Plan :

I. Présentation du projet

A/ Introduction.

Notre projet pluritechnique encadré consiste en la conception et la réalisation d'un prototype d'un distancemètre, permettant de mesurer la longueur d'une courbe à l'aide d'une roue. Nous avons pensé réaliser le projet à partir d'un élément tournant et d'un système électrique (système compteur), fonctionnant à l'aide d'un programme.

Notre projet se compose de :
-Génie Electrique : programmation du Pic, conception du circuit électrique et de l'affichage.
-Génie mécanique : modélisation numérique (logiciel SolidWorks) et conception mécanique réelle.

Notre groupe se compose de cinq membres. Nous avons réparti notre charge de travail en différentes catégories. Cependant, chacun s'est intéressé à l'ensemble du projet.

La répartition s'est faite de la façon suivante :
-Quentin et Florian se sont occupés en grande partie au domaine du Génie Electrique et donc, plus particulièrement de la programmation du pic (programme permettant le comptage de la distance parcourue).
-Nancy et Zélie ont préféré la partie Génie Mécanique, c'est-à-dire la conception de la maquette numérique réalisée sous Solidworks ainsi que l'étude du choix des composants (type, taille…).
-Paul, quant à lui, s'est intéressé à la logistique et à la recherche des composants (commande et réception). Suite à cela, il a suivi l'avancement des travaux.

B/ Cahier des charges et objectif final.

Cahier des charges :
- La précision de la distance sera de 50 à 100mm.
- La distance maximum mesurable : 99m.
- Autonomie de huit heures environ.

Objectif final :
- Réalisation d'une carte d'acquisition de la rotation de la roue.
- Réalisation de la carte d'affichage de la distance parcourue.
- Choix du capteur.
- Maquette numérique du système complet.
- Réalisation d'un prototype.

C/ Analyse fonctionnelle.

Nous avons débuté nos recherches par l'analyse fonctionnelle du produit grâce au tableau et aux schémas étudiés l'an dernier en cours de Génie Mécanique. Ainsi, nous allons pouvoir définir précisément les critères et les niveaux exigés selon les fonctions principales et les fonctions contraintes définies dans le cahier des charges.

Graphe des prestations :

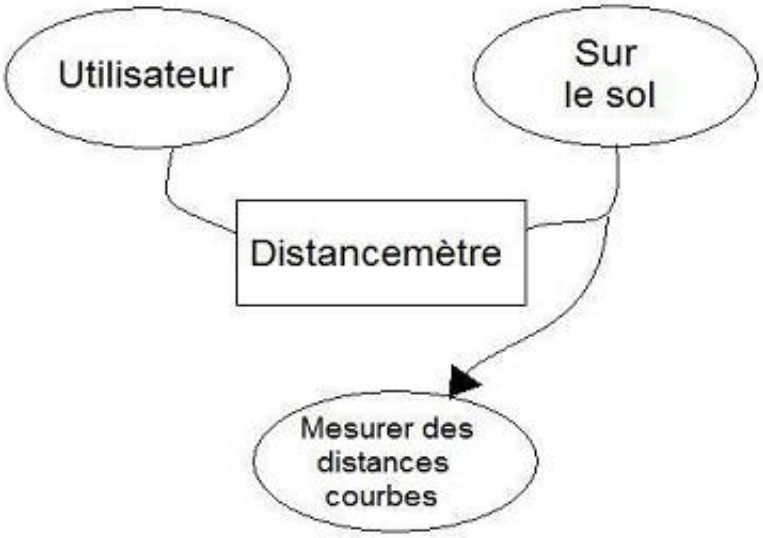
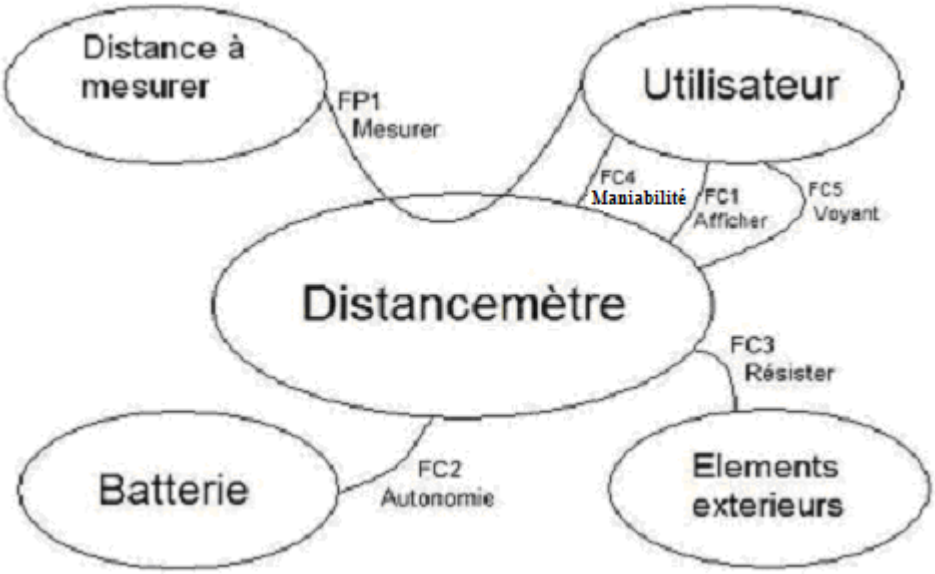


Diagramme des interacteurs :



Cahier des charges fonctionnel :

FONCTION	CRITERE	NIVEAU
FP1 : mesurer une distance à partir de la rotation de la roue	Diamètre de la roue choisi en fonction de la précision voulue	Précision : 50 à 100 mm Distance : 99m Rayon roue : 26cm
FC1 : Informer l'utilisateur sur la distance mesurée	Lisibilité	Deux afficheurs + possibilité de changer les unités affichées
FC2 : Autonomie	Durée	8h mini
FC3 : Résister aux éléments extérieurs	Solidité, étanchéité	Relatif aux conditions d'utilisation
FC4 : Maniabilité	Masse	< 5 kg
FC5 : Informer l'utilisateur sur le niveau de la batterie	Niveau de charge	Batterie faible (voyant allumé) Batterie pleine (voyant éteint)

Suite à cette analyse, nous avons pu déterminer les composants dont nous avions besoins pour la réalisation de notre prototype :

- roue (vélo, trottinette, skate…en fonction de la taille voulu)
- interrupteur à lame souple ILS
- aimants
- batterie (petite batterie ou ensemble de piles)

II. Différentes fonctions :

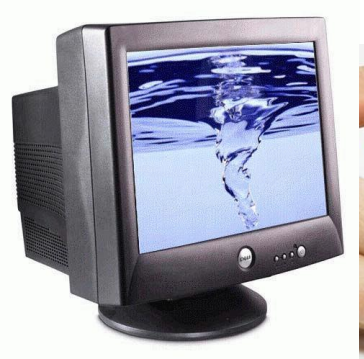
A/ Afficher.

Nous avons besoin d'un afficheur afin de pouvoir informer l'utilisateur de la distance parcourue par notre roue.
Pour cela, nous allons analyser différents types d'afficheurs :

-**Ecran LCD (Cristaux liquides), CRT (Tube à rayon cathodique), OLED (Diode électroluminescente) :**

Ces types d'écrans permettent d'afficher énormément d'informations et de qualité élevée (affichage avec pixels).
Prix : Elevé
Performance/Fiabilité : Excellente
Encombrement : Très important
Facilité d'emploi : Très dur pour programmer l'affichage

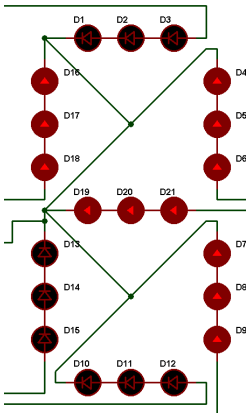

Écran LCD

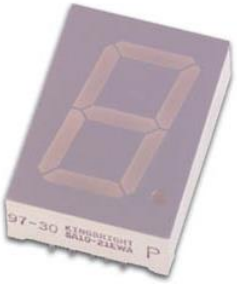

Ecran CRT

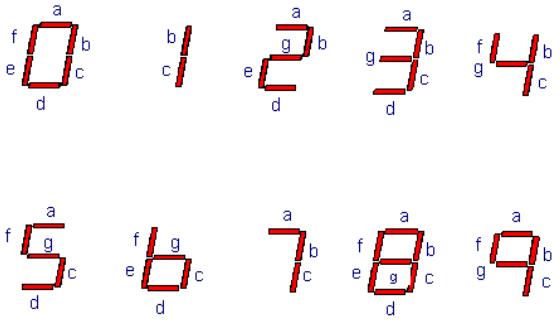

écran OLED (stade expérimental)

-**Afficheur 7 segments :**

Ce type d'écran permet d'afficher un nombre ou une lettre.
L'intérieur de l'afficheur est constitué de 7 segments (3 diodes par segment).







Combinaisons de segments permettant d'obtenir les chiffres de 0 à 9.

Prix : Faible
Performance/Fiabilité : Elevé (pour notre projet)
Encombrement : Faible
Facilité d'emploi : Facile

Nous avons choisi l'afficheur 7 segments pour ses contraintes faibles (performance élevé pour un prix faible).
Nous avons choisi un afficheur 7 segments et le dossier technique nous oblige à afficher une distance jusqu'à 99 mètres avec une précision de 5 à 10 cm. Nous avons donc trouvé utile de n'utiliser que 2 afficheurs (pour afficher un nombre entre 0 et 99) et un interrupteur pour afficher soit les centimètres soit les mètres selon la position de l'interrupteur (ce qui permet d'afficher un nombre entre 0 et 9999). Ainsi nous n'avons à gérer que deux afficheurs ce qui simplifie grandement le programme, diminue le prix et la complexité de la carte à réaliser.

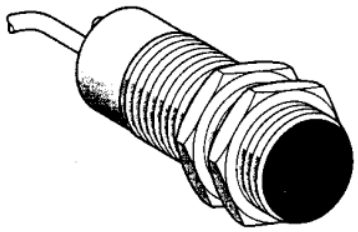
B/ Mesurer.

1. Choix du capteur.

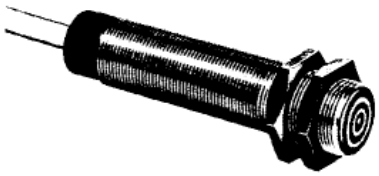
Nous cherchons un capteur capable de transmettre un signal logique (passage d'un objet) pour pouvoir interpréter et utiliser le signal émis par celui-ci.

Voici la liste des capteurs possibles :

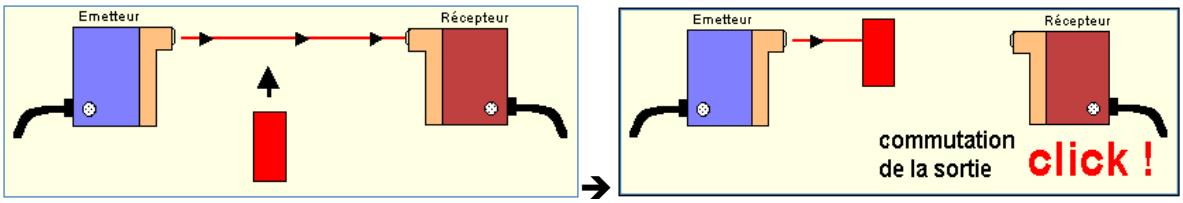
-**Capteur de proximité capacitif** :
Capteur qui permet de détecter des objets métalliques ou isolants à distance (par oscillation de la capacité d'un condensateur).
Prix : Elevé
Performance/Fiabilité : Moyenne (l'objet à détecter doit être très près)
Encombrement : Moyen
Type de signal : logique (ou analogique)
Facilité d'emploi : Moyen



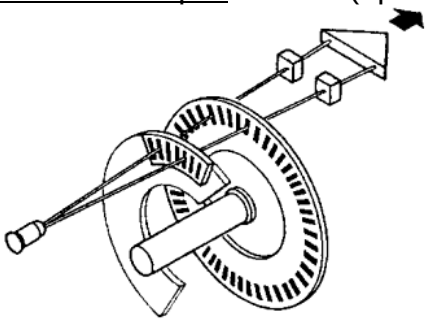
-**Capteur de proximité inductif** :
Capteur qui permet de détecter des objets métalliques à distance (par atténuation d'un champ magnétique crée par le capteur).
Prix : Elevé
Performance/Fiabilité : Moyenne (l'objet métallique doit être près)
Encombrement : Moyen
Type de signal : Logique (ou analogique)
Facilité d'emploi : Moyen



-**Capteur optique (photoélectrique)** :
La détection d'un objet se fait par coupure d'un faisceau lumineux crée par le capteur.
Prix : Elevé
Performance/Fiabilité : Elevé
Encombrement : Moyen
Type de signal : Logique (ou Analogique)
Facilité d'emploi : Moyenne

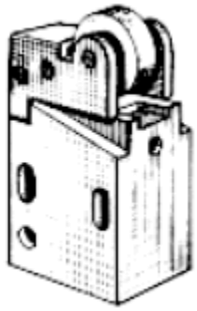


-**Capteur de position angulaire (codeur rotatif incrémental) ou roue codeuse** :
Un disque est divisé en x fentes qui permet à un faisceau lumineux de passer ou pas, selon la position de la roue. Il y a incrémentation à chaque fois que le faisceau est coupé.
Prix : Très élevé
Performance/Fiabilité : Excellente
Encombrement : Elevé
Type de signal : Logique (par incrémentation)
Facilité d'emploi : Facile (après montage)



-**Capteur de position** :

Capteur qui fonctionne par contact.
Prix : Faible
Performance/Fiabilité : Basse (pour ce que l'on veut faire)
Encombrement : Faible
Type de signal : Logique
Facilité d'emploi : Facile



-**Capteur ILS (Interrupteur à Lame Souple)** :

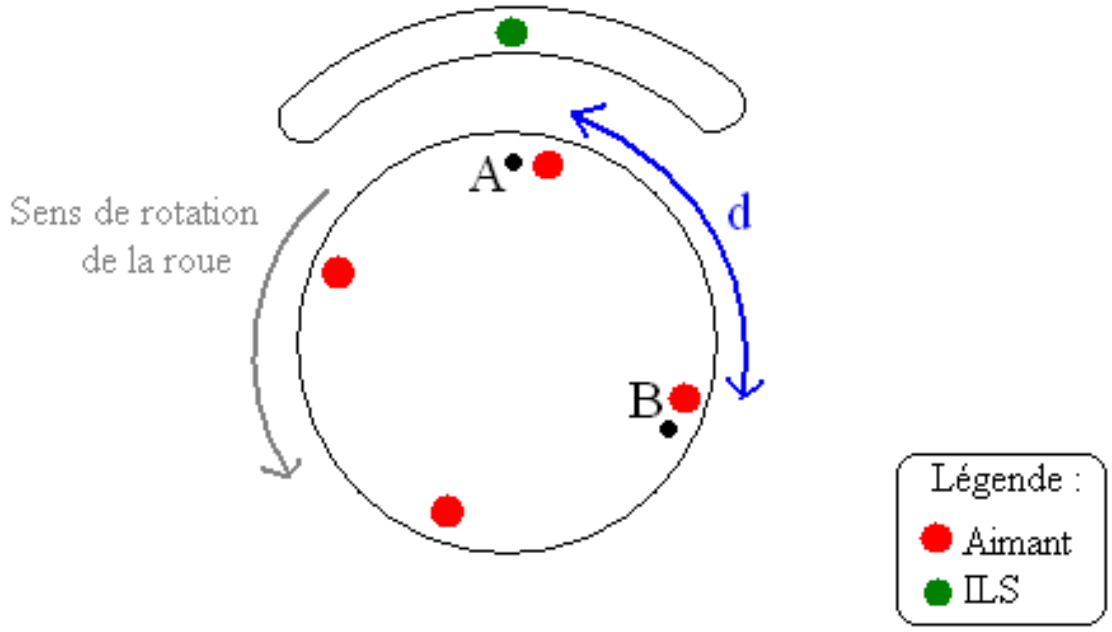
Capteur à distance sensible au champ magnétique qui fonctionne comme un interrupteur :
Si il détecte un aimant alors se ferme sinon, il reste ouvert (ou le contraire).
Prix : Faible
Performance/Fiabilité : Moyenne (l'aimant doit être à moins de 3 cm, ce qui est largement suffisant mais l'ILS est très fragile)
Encombrement : Minime
Type de signal : Logique
Facilité d'emploi : Facile

En considérant tous les facteurs (prix, encombrement …), nous avons décidé d'utiliser les ILS comme capteur même si ils sont fragiles. Ils ont toutes les qualités requises (rapidité et efficacité) pour réaliser notre projet.
On peut noter que pour l'utilisation des ILS, il faudra un nombre important d'aimants pour avoir une précision qui soit en accord avec le cahier des charges (cf « étude de la précision »).

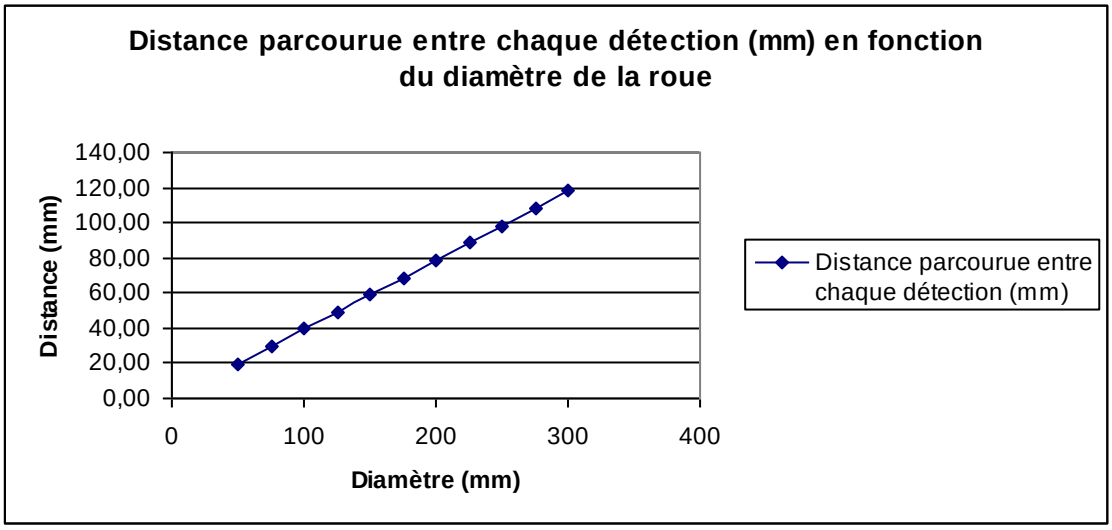
2. Etude de la précision.

(grâce à l'écart entre chaque détection d'aimant)

Dans le pire des cas, il y aura une imprécision équivalente à une fois la distance parcourue par la roue entre deux détections d'aimants. En effet, par exemple, si l'utilisateur débutait sa mesure au point A et la terminait au point B, le distancemètre considèrerait que la roue aurait parcouru deux fois la distance d puisqu'il y aurait eu 2 détections d'aimant. En réalité, elle n'aurait parcouru qu'une distance très légèrement supérieure à une fois la distance d.



De plus, il peut y avoir une imprécision dans le cas où l'utilisateur ne garde pas la même position du guidon tout au long de la mesure, étant donné que le guidon est en rotation par rapport à la roue.



Cas 1 :

Nombre d'ILS
2

Nombre d'aimants
18

Diamètre de la roue (mm)	Distance parcourue entre chaque détection (mm)
250	21,81661565
400	34,90658504
520	45,37856055
600	52,35987756
700	61,08652382
800	69,81317008
1000	87,2664626
1100	95,99310886
1110	96,86577349
1150	100,356432
1200	104,7197551

Ainsi, d'après ce tableau (issu des calculs d'un tableur), le diamètre de la roue doit être inférieur ou égal à 1140mm (si on choisit de ne prendre que 2 ILS et 18 aimants) car dans ce cas la distance parcourue entre chacune des détections d'aimant serait de 99,92 mm ce qui donne alors une imprécision maximale de 99,92mm. Le cahier des charges indique que le distancemètre devra offrir une précision de 50 à 100mm, cette condition est alors respectée.
Cependant, afin de prévoir d'éventuelles erreurs supplémentaires dues aux arrondis de nos calculs, par exemple, il vaut mieux choisir un diamètre de roue inférieur à 1140mm, lequel offre une marge d'erreur plus réduite.

Cas 2 :

Nombre d'ILS	Nombre d'aimants
2	4

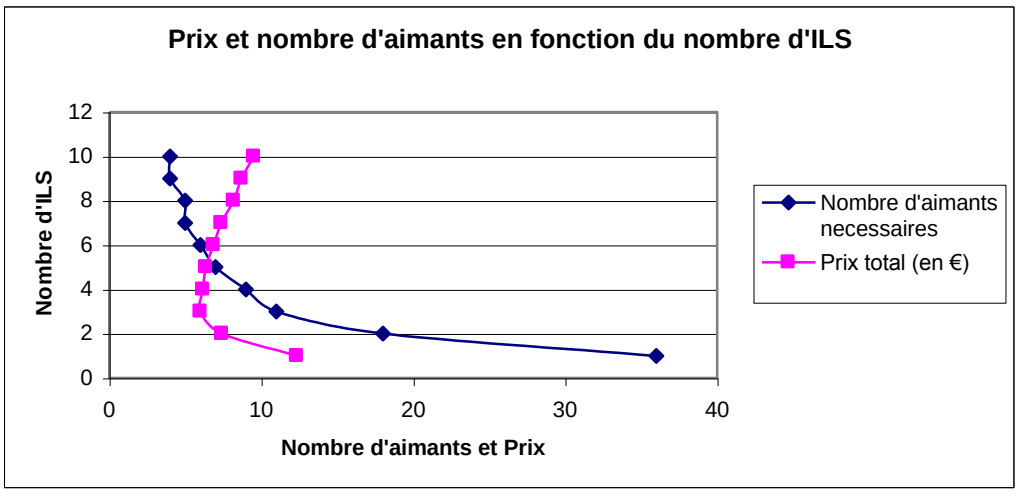
Diamètre de la roue (mm)	Distance parcourue entre chaque détection (mm)
50	19,63495408
75	29,45243113
100	39,26990817
125	49,08738521
150	58,90486225
175	68,7223393
200	78,53981634
250	**98,17477042**
400	157,0796327
520	204,2035225
600	235,619449

Ainsi, d'après ce tableau (issu des calculs d'un tableur), le diamètre de la roue doit être inférieur ou égal à 250mm (si on choisit de ne prendre que 2 ILS et 4 aimants) car dans ce cas la distance parcourue entre chacune des détections d'aimant serait de 49,09 mm ce qui donne alors une imprécision maximale de 98,17mm. Le cahier des charges indique que le distancemètre devra offrir une précision de 50 à 100mm, cette condition est alors respectée.
Cependant, afin de prévoir d'éventuelles erreurs supplémentaires dues aux arrondis de nos calculs, par exemple, il vaut mieux choisir un diamètre de roue inférieur à 250mm, lequel offre une marge d'erreur plus réduite.

3. Etude du prix.

La question qui se pose ensuite est : de quelle façon réduire le prix sans perte de précision ?
Il faut pour cela adapter le nombre d'ILS et d'aimants en fonction de leur prix respectif. Grâce à un graphique créé à l'aide d'un tableur, il est facile de repérer le couple (nombre d'ILS; nombre d'aimants) qui permettrait d'obtenir un coût moindre.

Cas 1 :



Diamètre de la roue (mm)
520

Distance désirée (mm)
50

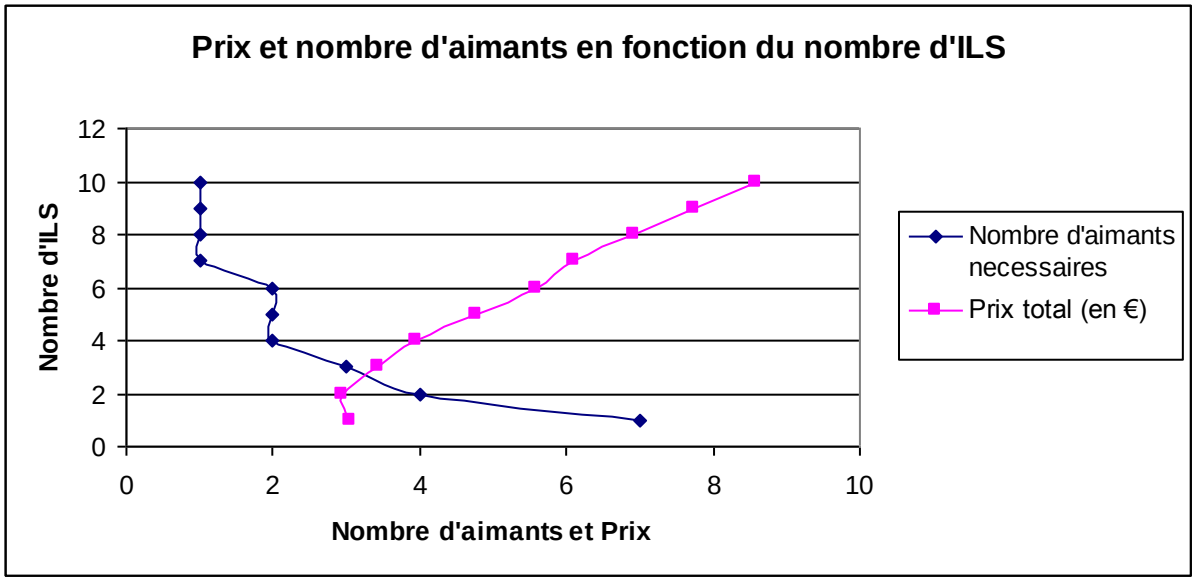
Prix ILS (€/unité)
0,825

Prix aimant (€/unité)
0,32

Nombre d'ILS	Nombre d'aimants nécessaires	Prix total (€)
1	36	12,345
2	**18**	**7,41**
3	11	5,995
4	9	6,18
5	7	6,365
6	6	6,87
7	5	7,375
8	5	8,2
9	4	8,705
10	4	9,53

D'après la courbe, le moins cher serait d'utiliser 3 ILS et 11 aimants. Cependant, les ILS seraient plus difficiles à fixer car il faudrait un support supplémentaire assez grand. Nous avons finalement choisi de prendre 2 ILS et 18 aimants, qui offrent tout de même un prix relativement réduit (support moins difficile à faire avec 2 ILS plutôt qu'avec 3).

Cas 2 :



Diamètre de la roue (mm)
100

Distance désirée (mm)
50

Prix ILS (€/unité)
0,825

Prix aimant (€/unité)
0,32

Nombre d'ILS	Nombre d'aimants nécessaires	Prix total (€)
1	7	3,065
2	4	2,93
3	3	3,435
4	2	3,94
5	2	4,765
6	2	5,59
7	1	6,095
8	1	6,92
9	1	7,745
10	1	8,57

D'après la courbe, le moins cher serait d'utiliser 2 ILS et 4 aimants, ce qui est tout à fait réalisable.

4. Nombre, pose et mesure grâce aux ILS.

Roue de diamètre 520 mm (roue de vélo) :

a) Faut-il un ou deux ILS ?

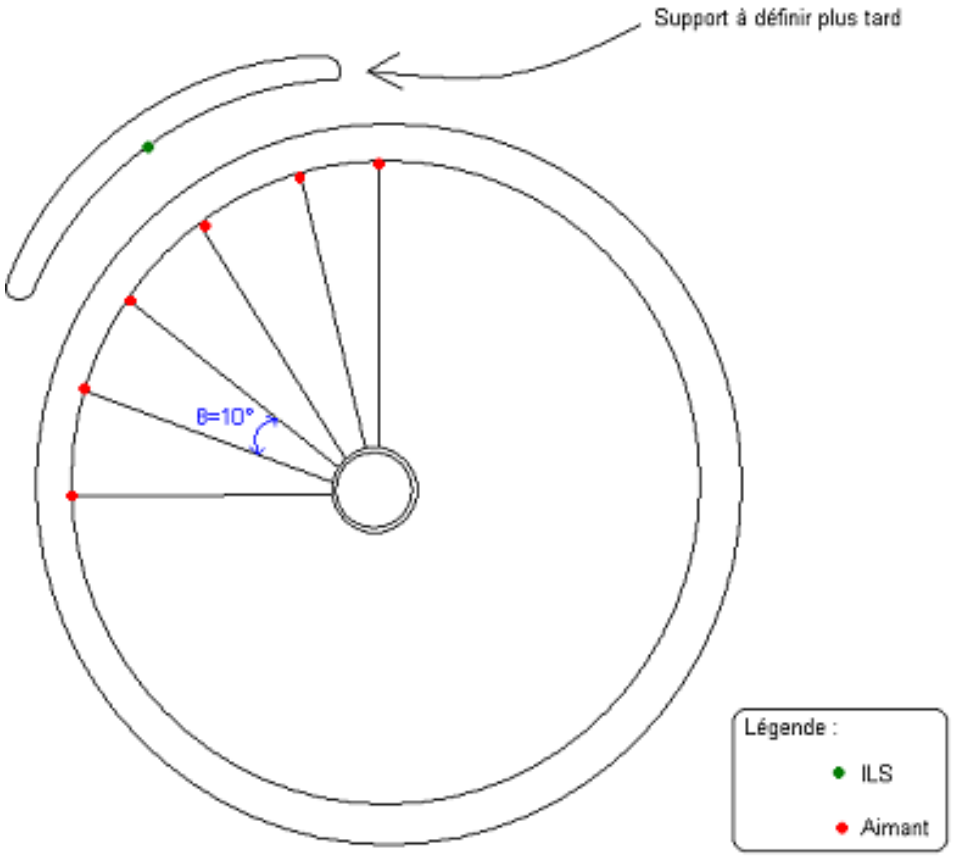
Pour commencer, nous avons trouvé une roue de diamètre égal à 52 cm et comportant 36 rayons également répartis. Nous avons pensé fixer les aimants sur les rayons afin de faciliter leur mise en place, les rayons étant également espacés.
La distance parcourue par la roue entre deux détections correspond à la longueur d'un arc de cercle de même rayon que la roue et dont l'angle au centre dépend du nombre d'aimants et d'ILS.

$$d = r * \theta \qquad \text{avec } \theta = \frac{2\pi}{a * ils}$$

Avec :

d =distance parcourue par la roue entre chaque détection d'aimant
r = rayon de la roue **(même unité que d)**
θ = angle entre un aimant, le centre de la roue et un deuxième **(en rad)**
a = nombre d'aimants
ils = nombre d'interrupteurs à lame souple

Solution 1 :



Avec un interrupteur à lame souple et un aimant sur chaque rayon (c'est-à-dire 36 aimants), entre chaque détection d'aimant par l'ILS (chaque incrémentation, donc), la roue aurait parcouru une distance de :

$$d = \frac{r * 2\pi}{a * ils} = \frac{26 * 2\pi}{36} = 4{,}54 \text{ cm}$$

C'est une précision suffisante par rapport à la valeur demandée par le cahier des charges mais le grand nombre d'aimants (36 aimants) risque d'être très coûteux.

Solution 2 :

Ainsi, d'après la formule, en multipliant par deux le nombre d'ILS, on peut aussi diviser par deux celui des aimants nécessaires pour une même précision :

$$d = \frac{r * 2\pi}{a * ils} = \frac{26 * 2\pi}{18 * 2} = 4{,}54 \text{ cm}$$

Idée supplémentaire :

Avec une roue de diamètre égal à 57cm, entre chaque détection, la roue aurait parcouru :

$$d = \frac{r * 2\pi}{a * ils} = \frac{28{,}5 * 2\pi}{18 * 2} = 4{,}97 \text{ cm}$$

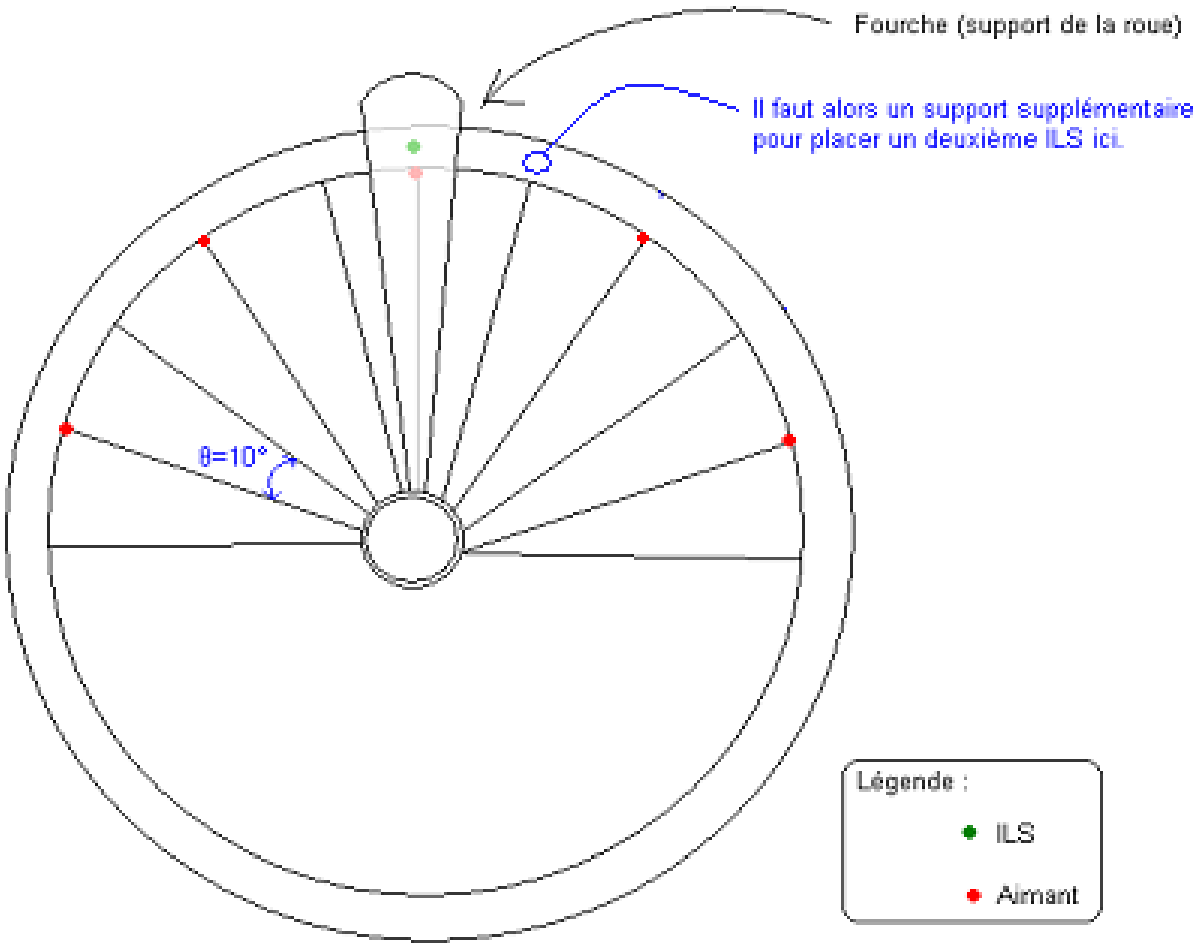
Cela peut, sans trop d'imprécision, être arrondi à 5,0 cm (contrairement aux 4,54cm pour la roue de diamètre 52cm). Cela rendrait le programme plus facile à mettre au point pour Quentin et Florian car il n'y aurait pas de millimètres à traiter. Cependant, ce n'est malheureusement pas le cas de la roue que nous avions et nous n'en avons pas trouvé qui respecte ces dimensions.

b) A-t-on besoin d'un support supplémentaire pour les ILS ?

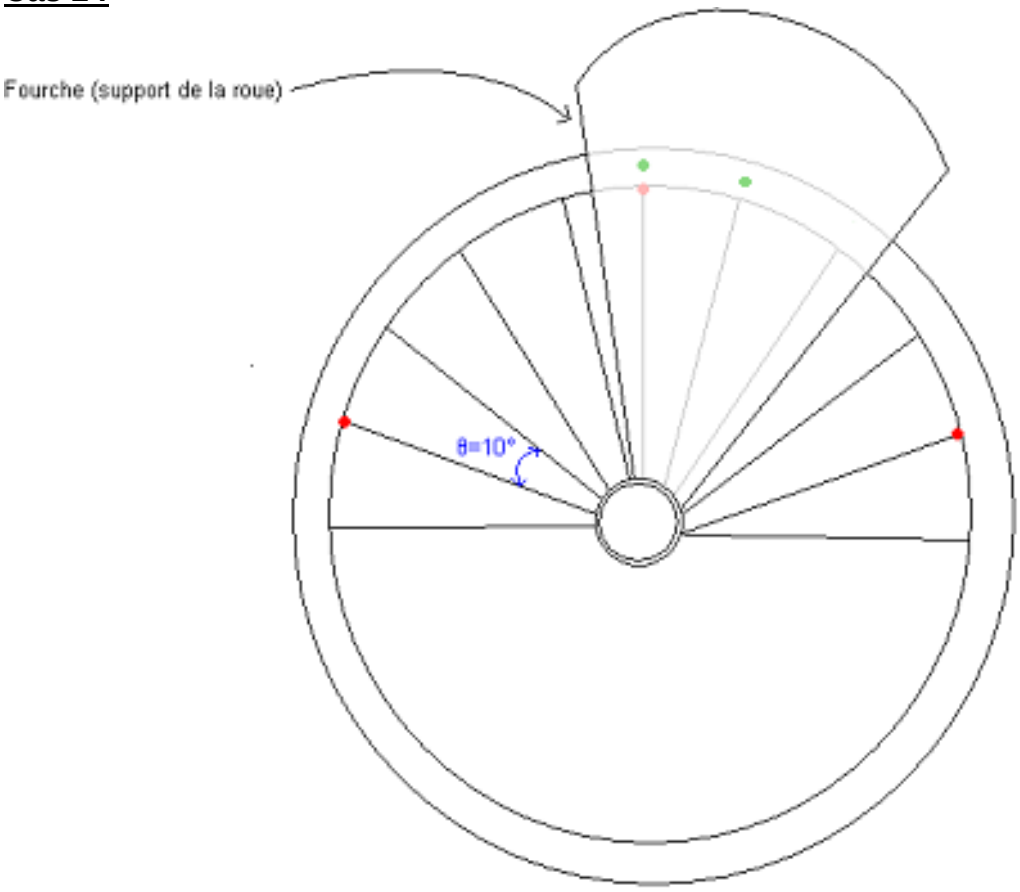
Pour la solution nécessitant un seul interrupteur à lame souple (dans la solution 1), nous avons prévu de placer celui ci à l'intérieur de la fourche.
Pour la solution à deux interrupteurs à lame souple, il pourrait y avoir plusieurs possibilités :
-Si la fourche est assez large pour que deux ILS soient fixés à 10 degrés de décalage (ou 30° ce qui revient au même), il serait donc possible de ne rien avoir à ajouter à la fourche actuelle. (cas 1)
-Sinon, il faudrait ajouter un support sur lequel serait fixé le deuxième. (cas 2)

Exemples :

Cas 1 :



Cas 2 :



Il faut placer les ILS avec un espacement de 10° (ou 30°, ce qui revient au même) pour ne pas qu'ils détectent les aimants en même temps.
Pour cela, il est nécessaire de rajouter un support afin que les conditions soient respectées.

Cependant, 18 aimants restent un achat assez conséquent. D'après nos recherches, l'idéal en terme de prix serait 4 aimants. Or, d'après les formules précédentes, pour un nombre d'aimants et d'ILS donnés (4 aimants, donc, et 1 ILS pour commencer) on obtient :

$$d = r * \theta$$

$$<=> \quad r = \frac{d}{\theta} = \frac{5}{\frac{\pi}{2}} = 3{,}18 \text{ cm}$$

Avec deux ILS, on peut multiplier ce rayon par 2, on obtient alors un rayon de 6,36 cm environ. Il s'agit du même ordre de grandeur que les dimensions d'une roue de trottinette. C'est pourquoi nous avons décidé d'en faire également l'étude.

Roue de diamètre 100mm (de type trottinette) :

a) Adaptation des calculs précédents sur la nouvelle roue

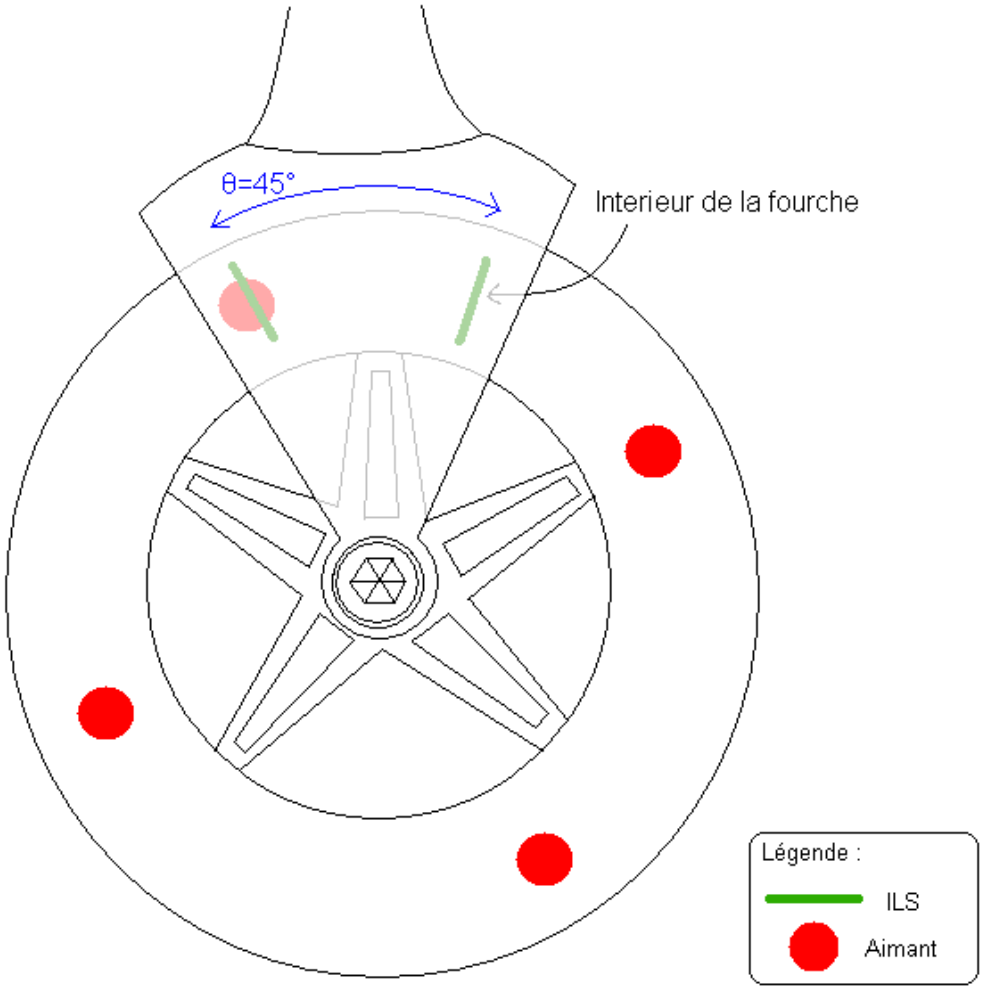
Nous nous sommes procurés une trottinette standard. Notre nouvelle roue a un rayon de 5,0cm. Nous avons décidé d'utiliser 2ILS et 4 aimants.
La distance parcourue par la roue entre chaque détection d'aimant sera alors :

$$d = \frac{r * 2\pi}{a * ils} = \frac{5{,}0 * 2\pi}{4 * 2} = 3{,}93 \text{ cm}$$

Cette distance pourrait peut être, contrairement à celle qui correspondait à la roue de vélo, être arrondie à 4cm, ce qui faciliterait la conception du programme.

De plus, la fourche de la trottinette est assez large pour y fixer deux ILS (à 45° d'écart, puisque les aimants sont espacés de 90° entre eux), ce qui permet de ne pas avoir à concevoir un support supplémentaire.

Aperçu :



b) Calcul de la précision

Nous avons vu précédemment que la perte de précision dans la distance totale parcourue était égale au maximum à une fois la distance parcourue entre deux détections d'aimant. (voir B/d/)
Dans le cas de notre trottinette, notre résultat serait correct à 3,93 cm près. Cependant, avec l'arrondi (voir d/1.) qui permet de faciliter la conception du programme, cette perte de précision serait augmentée. Vérifions-en la valeur :

On aurait une imprécision supplémentaire de **4 - 3,93 = 0,07cm** à chaque détection d'aimant, c'est-à-dire à chaque fois que la roue parcours 3,93cm. On compte mesurer jusqu'à 99m. Donc on aurait une imprécision à ajouter au bout du parcours de :

Nombre de détection = Distance totale / Distance par détec. = 9900/3,93 = 2519

Imprécision à ajouter = Nombre de détec. * imprécision par détec. = 2517 * 0,07 = 176cm

Nul besoin d'ajouter le manque de précision initial maximum de 3,93 cm pour se rendre compte que ce manque de précision ne respecte pas le cahier des charges qui offre une marge d'erreur maximum de 10cm. Il n'est donc pas raisonnable d'arrondir la distance parcourue par la roue entre deux détections d'aimant à 4cm.

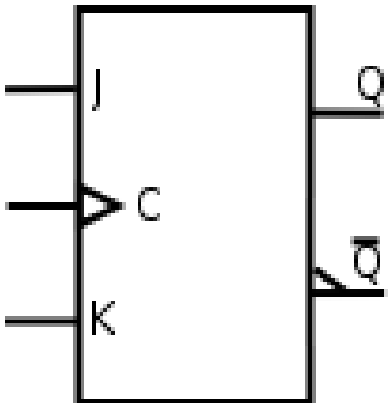
Finalement, l'utilisation d'une roue de type trottinette serait avantageuse au niveau du prix (nombre d'aimants réduit).
En revanche, elle ne respecterait pas le cahier des charges au niveau de la résistance aux éléments extérieurs (la fourche étant trop étroite, il serait impossible de protéger les ILS).
C'est pour cela que pour notre prototype nous utiliserons la roue de vélo.

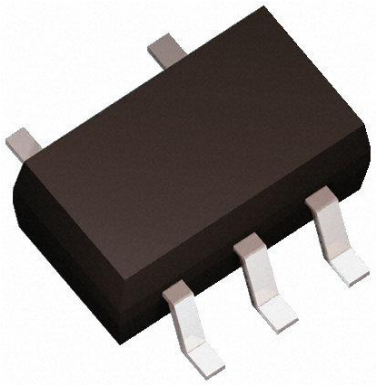
C/ Gérer l'information

1. Choix du PIC

-**Les bascules** :

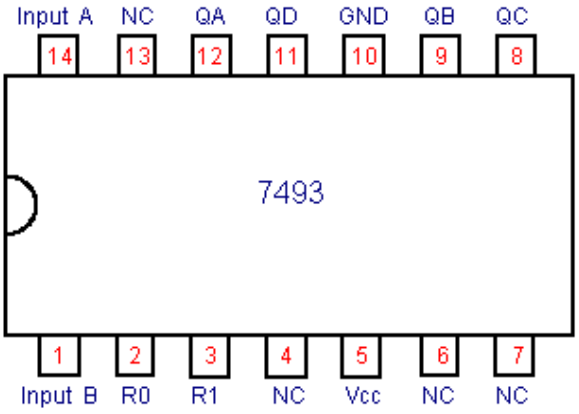
Avec un grand nombre de bascules, les une à la suite des autres, nous aurions pu obtenir la distance parcourue avec les différentes unités voulues (cm, m) en prenant les valeurs à certains endroits du circuit. Cependant, face à la complexité de ce moyen, et surtout au nombre de bascules dont nous aurions besoin pour le mettre en place, nous y avons renoncé au profit de l'utilisation d'un PIC ou d'un compteur intégré.





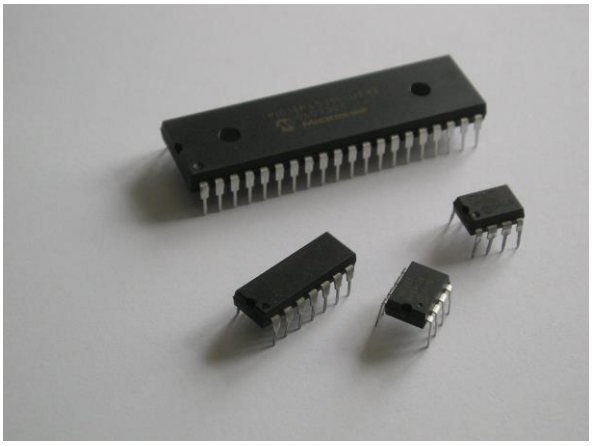
-**Le compteur intégré :**

Le compteur intégré est un circuit contenant un grand nombre de bascules.
Il permet de faire office de compteur.
Cet organe d'acquisition est donc plus attractif que le précédent car nous n'avons pas tout le câblage à faire.
Mais un problème persiste : la compatibilité avec les 2 afficheurs 7 segments.
En effet, il serait assez compliqué de gérer le multiplexage des afficheurs avec le compteur intégré.



-**Le pic** :

Le microcontrôleur PIC se programme en C et peut réaliser de nombreuses fonctions. Son utilisation est donc simple (si on le maîtrise bien). Il est peu cher et effectue l'essentiel de ses actions en un cycle d'horloge (soit très vite). Il dispose de 13 ports que l'on peut définir comme sortie ou entrée au besoin. Il disposerait donc de suffisamment de sorties pour utiliser un afficheur 7 segments et de suffisamment d'entrées pour gérer le capteur et les différents interrupteurs dont on pourrait avoir besoin.



Nous avons donc choisis d'utiliser un PIC.

2. Programme

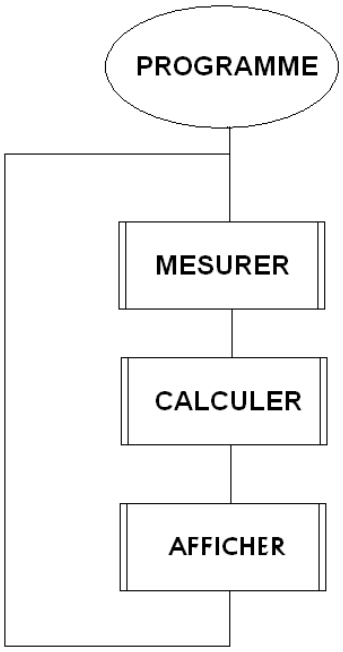
Le programme qui va suivre est en C et a été créé sur le logiciel DevPic84c.
Nous avons aussi fait un organigramme pour avoir un aperçu plus clair de notre programme et avons mis sur le programme en C des commentaires pour mieux le comprendre.

Pour commencer, nous appelons les librairies C et nous définissons nos variables :

```
//-----------------------------------------------Les declarations---------------------------------

#include std84.h //appel des librairies C
#include bit84.h
#define ZERO           portb.0=1; portb.1=1; portb.2=1; portb.3=1; portb.4=1; portb.5=1; portb.6=0;
#define UN             portb.0=0; portb.1=1; portb.2=1; portb.3=0; portb.4=0; portb.5=0; portb.6=0;
#define DEUX           portb.0=1; portb.1=1; portb.2=0; portb.3=1; portb.4=1; portb.5=0; portb.6=1;
#define TROIS          portb.0=1; portb.1=1; portb.2=1; portb.3=1; portb.4=0; portb.5=0; portb.6=1;
#define QUATRE         portb.0=0; portb.1=1; portb.2=1; portb.3=0; portb.4=0; portb.5=1; portb.6=1;
#define CINQ           portb.0=1; portb.1=0; portb.2=1; portb.3=1; portb.4=0; portb.5=1; portb.6=1;
#define SIX            portb.0=0; portb.1=0; portb.2=1; portb.3=1; portb.4=1; portb.5=1; portb.6=1;
#define SEPT           portb.0=1; portb.1=1; portb.2=1; portb.3=0; portb.4=0; portb.5=0; portb.6=0;
#define HUIT           portb.0=1; portb.1=1; portb.2=1; portb.3=1; portb.4=1; portb.5=1; portb.6=1;
#define NEUF           portb.0=1; portb.1=1; portb.2=1; portb.3=1; portb.4=0; portb.5=1; portb.6=1;
#define initialisation portb.0=0; portb.1=0; portb.2=0; portb.3=0; portb.4=0; portb.5=0; portb.6=0;
 /* simplification des nombres affichés sur les afficheurs 7 segments
(ex: pour former le 0, on utilise 6 sorties du pic : RB0,RB1,RB2,RB3,RB4,RB5) */

char unitecm, dizainecm, millimetre, unitemetre, dizainemetre;
// définition globale des variables
```

Programme principal :

- Organigramme :

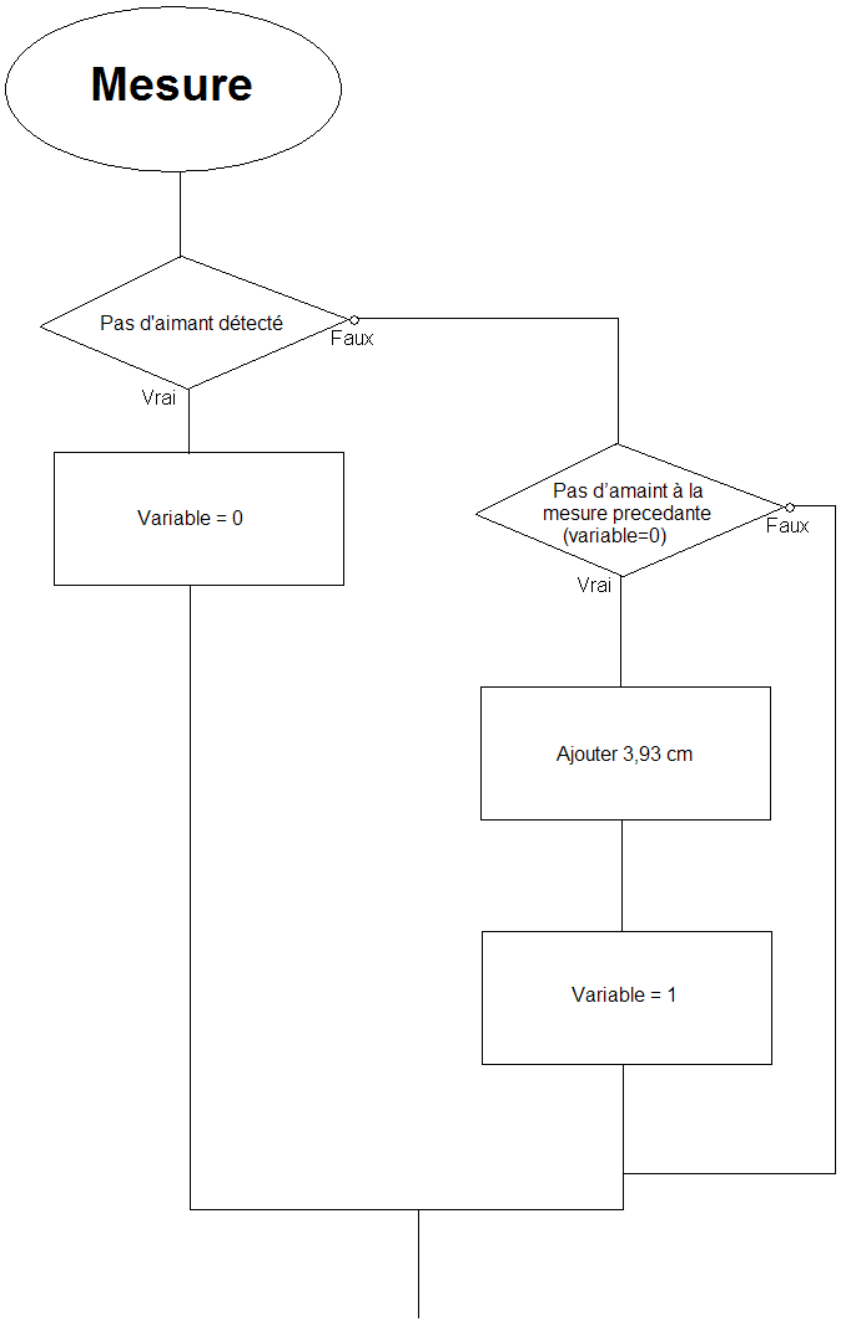


- Programme C :

```
//---------------------programme principal-----------------

void main()
    {
    trisa = 0b11111111; //définir les ports A comme entrée
    trisb = 0b00000000; //définir les ports B comme sortie
    unitecm=0;  //initialisation des variables à 0
    dizainecm=0;
    unitemetre=0;
    millimetre=0;
    dizainemetre=0;

    while(1) // boucle infinie
        {
        mesurer();
        calculer(); // 10mm->1cm, 10cm->1 dizaine de cm ...
        afficher(); // afficher soit les cm (RA1=0) soit les mètres (RA1=1)
        }

    }
```

Programme « Mesurer » :

- Organigramme mesure:

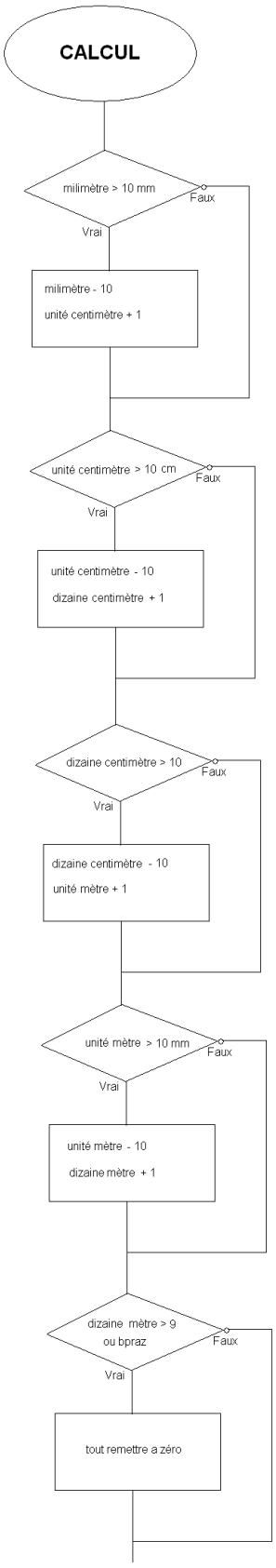


- Programme mesure:

```
//----------------------Fonction Mesurer-------------------------

void mesurer()
    {
      if (porta.0) //si détection d'un aimant
        {
        while(porta.0); //attendre jusqu'à ne plus détecter l'aimant
        millimetre=millimetre+35; //phase de calcul de la distance parcourue
        unitecm=unitecm+3;
        }

      else
        {
        afficher()
        }
    }
```

Programme « Calculer » :

- Organigramme « calculer »:

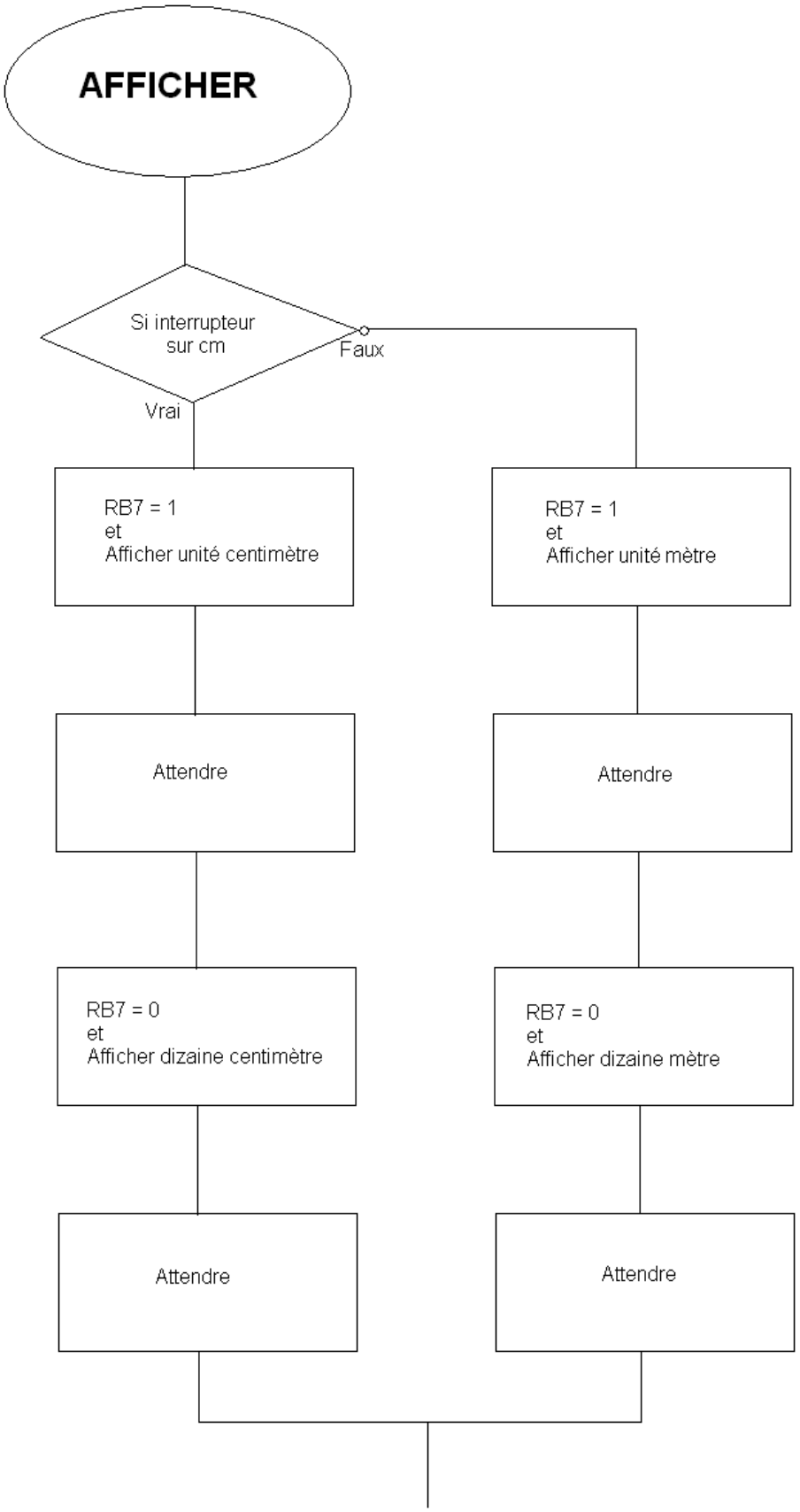


- Programme « calculer »:

```
//----------------------Fonction Calculer-----------------------

void calculer() /* cette fonction convertit 10mm en 1cm,
10cm en 1 dizaine de cm, 10 dizaines de cm en 1 mètre,
10 mètres en 1 dizaine de mètre. */
    {
        if(millimetre>99)
            {
            unitecm++;
            millimetre=millimetre-100;
            }

        if(unitecm>9)
            {
            unitecm=unitecm-10;
            dizainecm++;
            }

        if(dizainecm>9)
            {
            dizainecm=dizainecm-10;
            unitemetre++;
            }

        if(unitemetre>9)
            {
            unitemetre=unitemetre-10;
            dizainemetre++;
            }

//remise à zero si la distance dépasse 100 mètres :
        if(dizainemetre>9)
            {
            unitecm=0;
            dizainecm=0;
            unitemetre=0;
            dizainemetre=0;
            }
    }
```

Programme « Afficher » :

- Diagramme « afficher » :

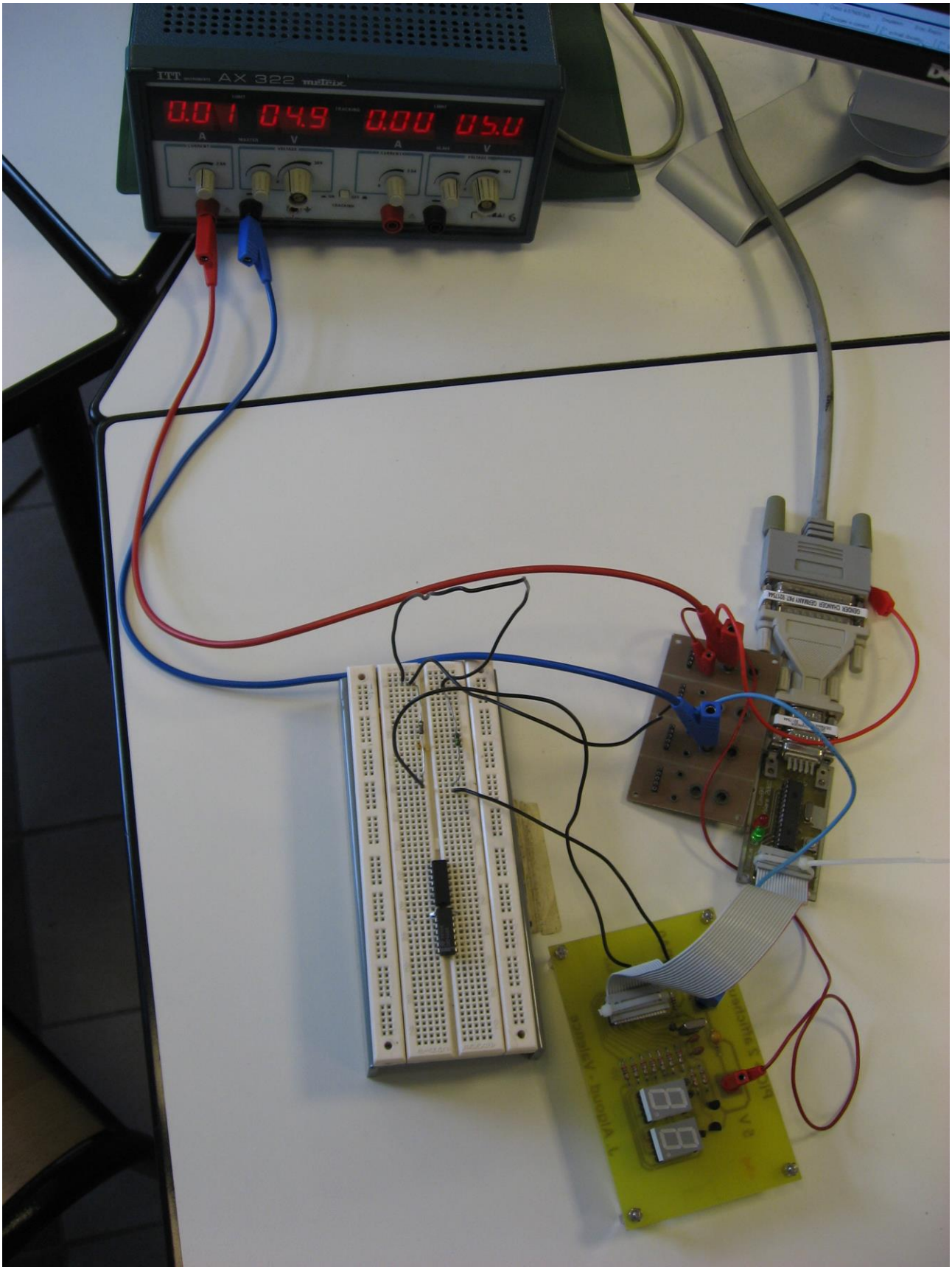


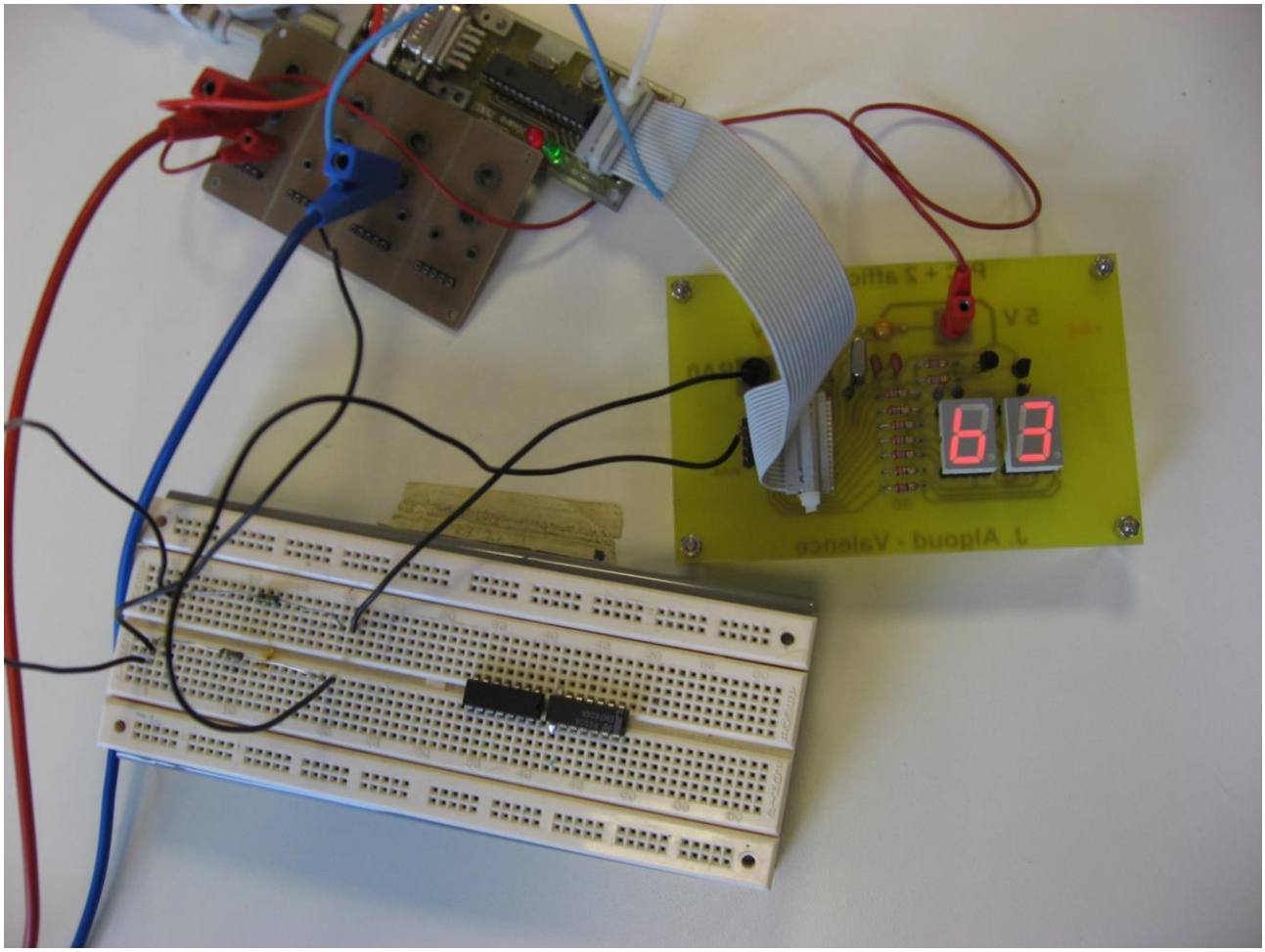
- Programme « afficher » :

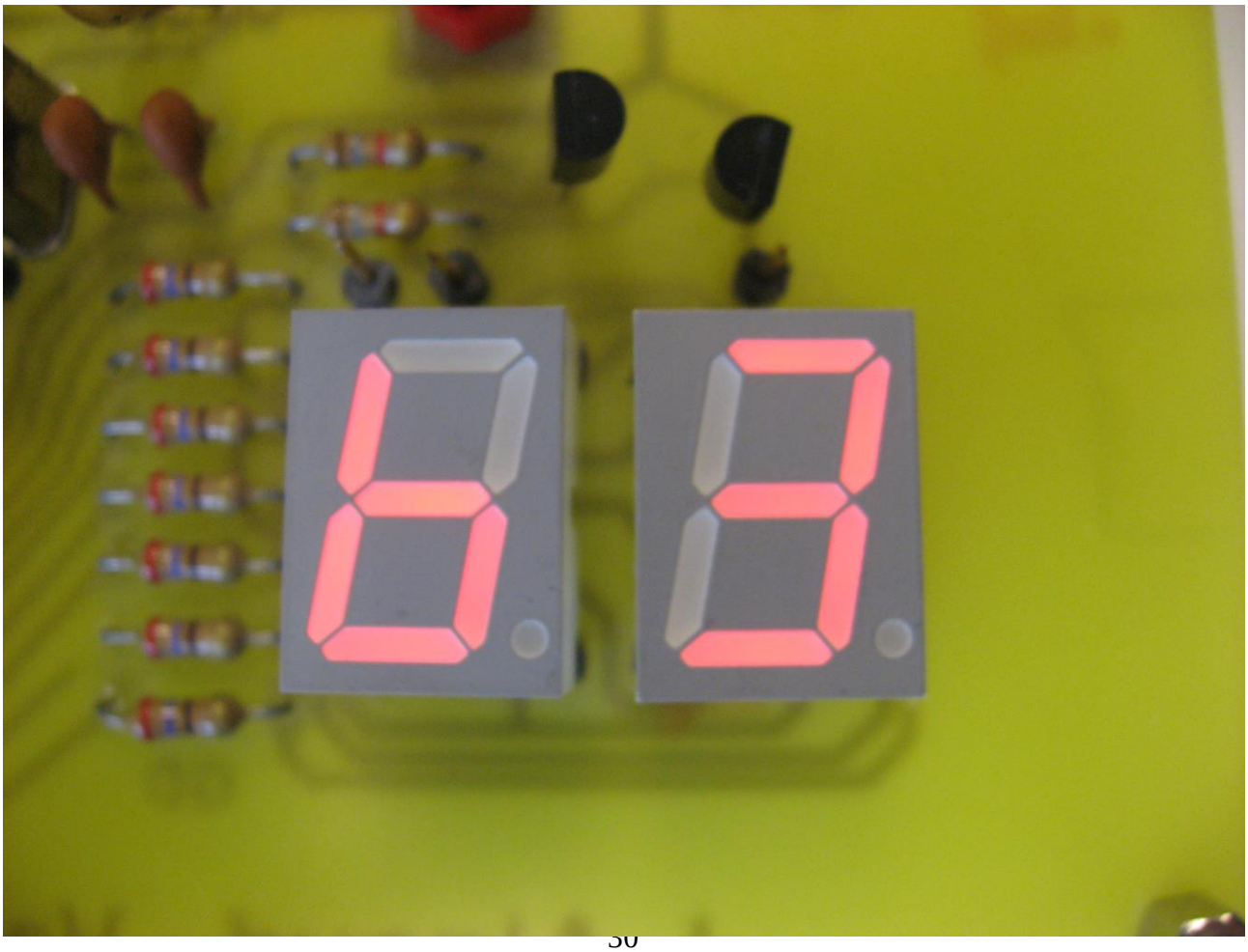
```
//--------------------Choix entre M et Cm---------------

 void afficher()
    {
    if(porta.1)
// appel de la fonction 'afficher' voulue (cm si RA1=0, mètre si RA1=1)
        {
        affichermetre();
        }
    else
        {
        affichercm();
        }
    }
//--------------------Affichage des Centimètres--------------


void affichercm() // cette fonction affiche les valeurs en cm
 {
portb.7=0;   // l'afficheur des dizaines est allumé

if((dizainecm==0))
{ZERO;}
if((dizainecm==1))
{UN;}
if((dizainecm==2))
{DEUX;}
if((dizainecm==3))
{TROIS;}
if((dizainecm==4))
{QUATRE;}
if((dizainecm==5))
{CINQ;}
if((dizainecm==6))
{SIX;}
if((dizainecm==7))
{SEPT;}
if((dizainecm==8))
{HUIT;}
if((dizainecm==9))
{NEUF;}

microdelay(5);   //détermine la durée de l'affichage de chaque chiffre
initialisation;   /*mise à 0 de toutes les sorties du pic
afin d'éviter une superposition des chiffres qui rendrait la mesure illisible   */

portb.7=1; // l'afficheur des unités est allumé

 if(unitecm==0)
 {ZERO;}
 if(unitecm==1)
 {UN;}
 if(unitecm==2)
 {DEUX;}
 if(unitecm==3)
 {TROIS;}
 if(unitecm==4)
 {QUATRE;}
 if(unitecm==5)
 {CINQ;}
 if(unitecm==6)
 {SIX;}
 if(unitecm==7)
 {SEPT;}
 if(unitecm==8)
 {HUIT;}
 if(unitecm==9)
 {NEUF;}

 microdelay(5);
 initialisation;

  }
```

3. Simulation

Voici différentes photos du programme sur simulateur :





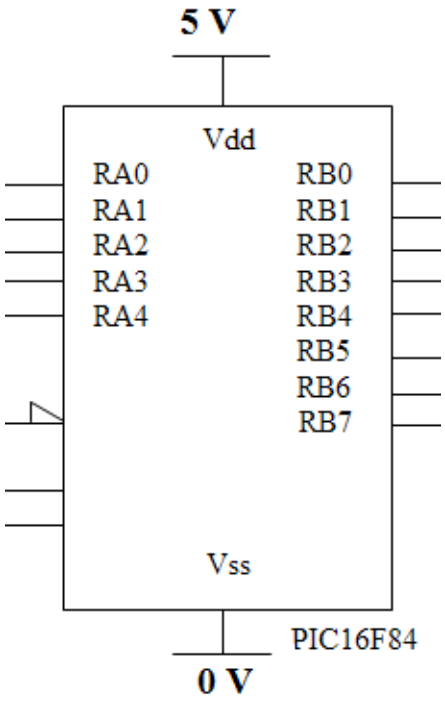


D/ Alimenter

1. Calculs de consommation

Nous considérons notre circuit comme parfait et y ajouterons une marge d'erreur de 25 %.
On considère aussi que seul le PIC et un Afficheur 7 segments (+ résistances) consomment de l'énergie.
En effet, les 2 afficheurs ne consomment jamais de l'énergie en même temps mais le font l'un après l'autre à des intervalles de temps imperceptibles à l'œil nu (affichage multiplexé).
L'autonomie de notre circuit doit être de 8 heures au minimum.

a. Consommation maximale du PIC :



Imax = 10mA (donnée constructeur)

Qmax(Ah) = I(A) x t(h)
= (10 x 10^-3) x 8
= 8,0 x 10^-2 Ah

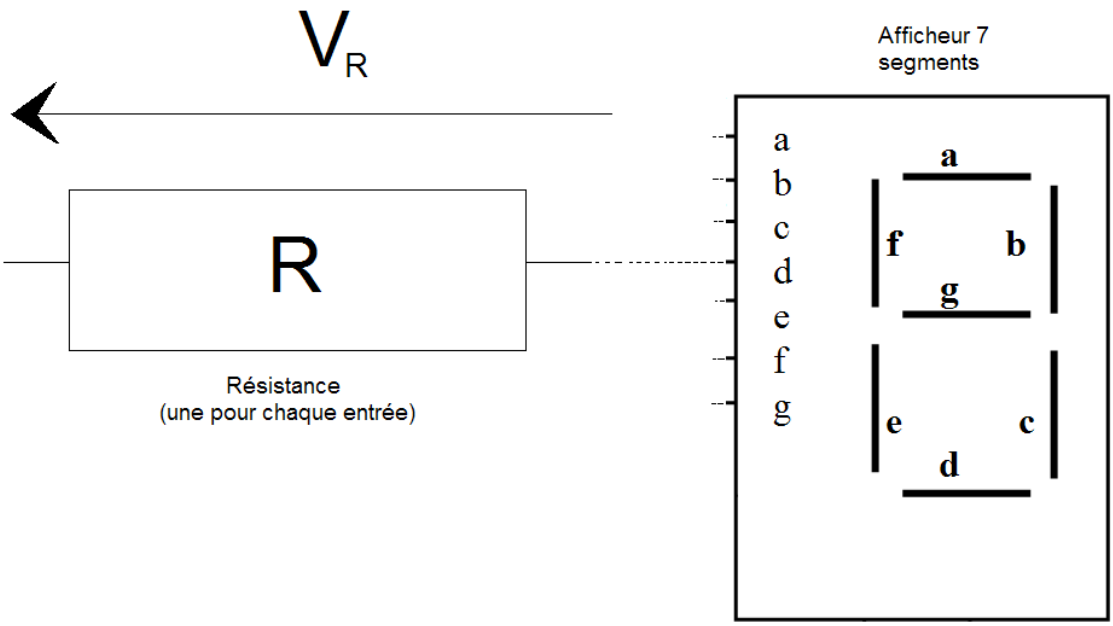
Pour ne prendre aucun risque, nous prenons une marge de sécurité de 25%.

Ainsi, Q = 1,25 x (8,0 x 10^-2) = 0,1 Ah

Qpic = 0,1 Ah

b. Consommation maximale de l'afficheur 7 segments (+ résistances) :

Nous supposons ici que l'afficheur 7 segments a tous ses segments d'allumés (affichage d'un 8) et donc que sa consommation est maximale.



Nous avons mesuré les tensions de chaque entrée et avons trouvé :

R = 270 Ω
Vmoy = 1,10 V

Ainsi, Isegment = V/R = 1,10/270 = 4,08 x 10^-3 A par segment

D'où Iafficheur = 7 x Isegment = 28,56 x 10^-3 A (car il y a 7 segments par afficheur)

Il en résulte : Qafficheur(Ah) = I(A) x t(h)
= (28,56 x 10^-3) x 8
= 0,23 Ah

Par mesure de sécurité, on considère que nous avons 25% d'erreur :
Qmax afficheur = 1,25 x 0,23 = 0,29 Ah

Qmax afficheur = 0,29 Ah

c. Consommation moyenne de l'afficheur 7 segments (+ résistances) :

Attention : Le calcul suivant n'est valable que si l'on suppose la vitesse de l'utilisateur parfaitement constante. Et que tous les segments ont la même consommation.

			6 segments
			2 segments
			5 segments
			5 segments
			4 segments
			5 segments
			6 segments
			3 segments
			7 segments
			6 segments

Ainsi, pour 10 chiffres affichés, il y aura en moyenne 49 segments allumés.

Moyenne du nombre de segments qui affichent
= (Somme des segments affichable pour les chiffres de 0 à 9) / (le nombre de chiffres)
= (6+2+5+5+4+5+6+3+7+6)/10
= 4.9 segments éclairés en continu

 Au lieu de 70 segments, il y en a 49 affichés (en moyenne) qui sont affichés.

Il y a donc environ 30% de l'énergie qui est conservée pour 49 affichés au lieu des 70 affichés.

Energie moyenne= 70% Energie si les 7 segments sont allumés

D'où : Qmoy afficheur = 0,7 x Qmax afficheur = 0,7 x 0,29 =

Qmoyen afficheur = 0,20 Ah

d. Consommation totale :

Qtotal max = Qpic + Qafficheur max
= 0,1 + 0,39 Ah
= 0,39 Ah

Qtotal moyen = Qpic + Qafficheur moyen
= 0,1 + 0,20
= 0,30 Ah

Qtotal max = 0,39 Ah
et
Qtotal moyen = 0,30 Ah
Pour
8 Heures de fonctionnement

2. Choix de l'alimentation

Le cahier des charges impose une autonomie de 8 heures.
Or, 0,39Ah correspond à l'énergie nécessaire pour que notre système marche pendant 8 h (voir la partie « Consommation du circuit électrique »).

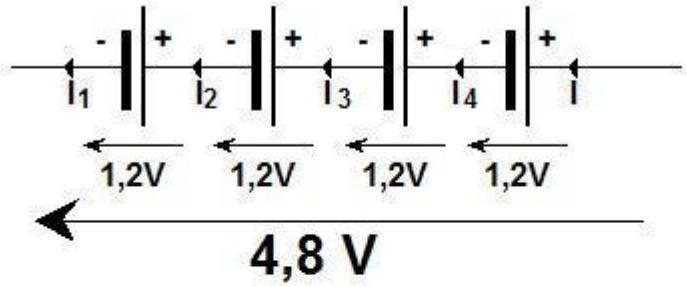
Il nous faut donc une alimentation de 5V et 0,39 Ah (ou plus).

Voici les différents types d'alimentation :

-**La Pile** : Alimentation qui transforme de l'énergie chimique en énergie électrique.
Prix : Faible
Performance : Moyenne
Encombrement : Faible

- **batterie 5V, Accumulateur** (ou « Pile rechargeable ») : Alimentation qui transforme de l'énergie chimique en énergie électrique et qui peut (parfois) inverser cette transformation pour se recharger.
Prix : Important
Performance : Excellente
Encombrement : Moyen

La pile semble être le meilleur choix (prix et performance convenables).
Suite à quelques recherches, nous nous sommes aperçus que les piles peuvent fournir une énergie largement au dessus de 0,39Ah mais ont une tension de 1,2V. Normalement, la tension doit être de 5,0V mais 4,8V suffisent pour que le montage fonctionne.
Nous avons donc décidé d'utiliser 4 piles en série pour avoir une tension de 4,8 V et une quantité d'énergie égale à celle d'une seule pile (600mA.h) :



$$I_{Pile1} = I_{Pile2} = I_{Pile3} = I_{Pile4} = I_{Total}$$

$$V_{Pile1} + V_{Pile2} + V_{Pile3} + V_{Pile4} = V_{Total}$$

3. Détection de batterie faible

Nous avons trouvé judicieux de pouvoir savoir quand notre batterie serait déchargée. Ce circuit est basé sur celui du portail Avidsen. Mais la tension d'alimentation que nous voulons tester est de 4.8V alors qu'elle est de 12V pour le portail. Nous sommes donc obligés de refaire des calculs.
Voici donc le schéma électrique correspondant (sans les valeurs des résistances):

Ce circuit utilise 2 alimentations :
-une alimentation que l'on veut tester (notre alimentation avec pile qui peut baisser au cours du temps).
-une alimentation de comparaison STABLE qui ne doit pas dépendre des fluctuations de l'alimentation à pile (on peut par exemple utiliser un régulateur 3V).

Nous voulons trouver les résistances adéquates pour le circuit ci dessus :

a. Trouver les valeurs d'une combinaison des 4 résistances de comparaison (R1, R2, R3 et R4) :

L'AOP ici présent est en mode non linéaire non inverseur.
Ainsi, Si e- > e+ alors la sortie de l'AOP délivre 0V
Si e+ > e- alors la sortie de l'AOP délivre 3V

Pour éviter de compliquer les calculs, nous déterminons arbitrairement les valeurs des résistances du seuil bas (R3 et R4) :
R3 = 1.2KΩ (valeur normalisée E12)
R4 = 1 KΩ (valeur normalisée E12)

Ainsi : e- = V(R4) = (3 x R4)/(R3 + R4) □ diviseur de tension
= (3 x 1)/(1.2 + 1)
= 1.36 V
e- =1,36 V

Il nous faut maintenant trouver la valeur de R1 et R2.
Or, notre 4.8 V correspond à un signal logique de niveau haut.
Un signal reste au niveau logique 1 entre 3,5 et 5V environ.

Ainsi, il faut que l'AOP bascule à la valeur la plus basse de l'alimentation que l'on peux admettre (soit 3,5 V).
Ainsi :
e+min = (3,5x R1)/(R1 + R2) = e- = 1,36 V

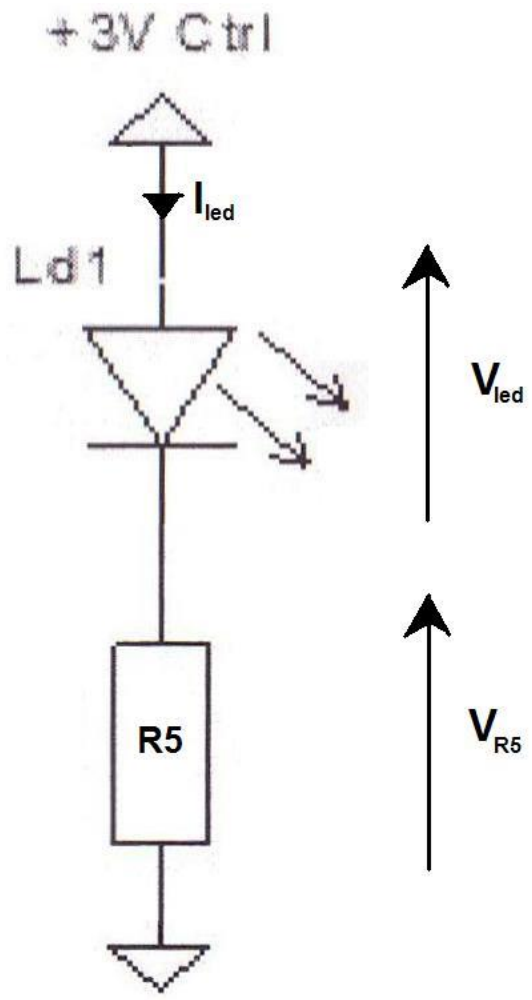
C'est un système à 2 inconnus mais avec une seule équation.
Nous somme donc encore obligés de prendre arbitrairement une valeur d'une des 2 résistances :
R1 = 330 KΩ (valeur normalisée E12)

Nous pouvons maintenant résoudre l'équation :
R2 = [(3,5 x R1)/(e+min)] –R1
= [(3,5 x 330)/1,36]-330
= 519 KΩ
Donc, d'après les valeurs normalisés E12 : R2= 470 KΩ (< 519 car on veut prévenir l'utilisateur qu'il y a une baisse de tension avant que celle-ci n'arrive).

De plus, d'après les caractéristiques des piles : Pmax = Vpile x Ipile = 4.8 x (0.6/8) = 0.36 W
Ainsi, une possible combinaison de résistance est :
R1 = 300 KΩ ½ W
R2 = 470 KΩ ½ W
R3 = 1,2 KΩ ½ W
R4 = 1 KΩ ½ W

b. Trouver la valeur de la résistance de protection aux bornes de la LED :

Iled = 2 mA (donnée constructeur)
Vled = 1,7 V (donnée constructeur)



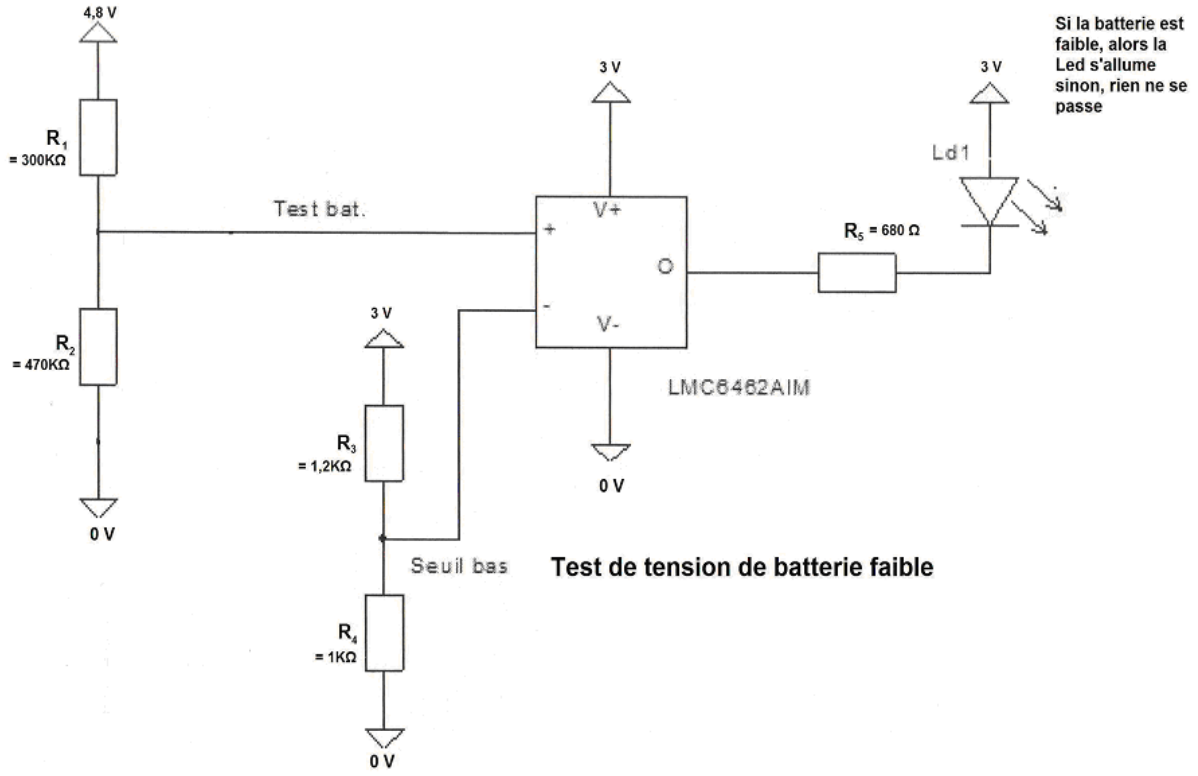
Loi des mailles : Vtotal = Vled + V(R20)
Ainsi : V(R5) = Vtotal – Vled = 3 – 1.7 = 1.3 V
De plus, I(R5) = I(led) = 2,0 mA (car les 2 composants sont en série).
D'où : R5 = V(R5)/I(R5) = 1,3/(2,0 x 10^-3) = 650 Ω
Cette résistance est une résistance de protection; on prend donc, (dans la série E12) la valeur au dessus de 650 Ω soit 680 Ω.

Or, P(R5) = (V(R5)²/R5) = 1,3²/680 = 2,5 x 10^-3 W

Ainsi :

R5 = 680 Ω ¼ W

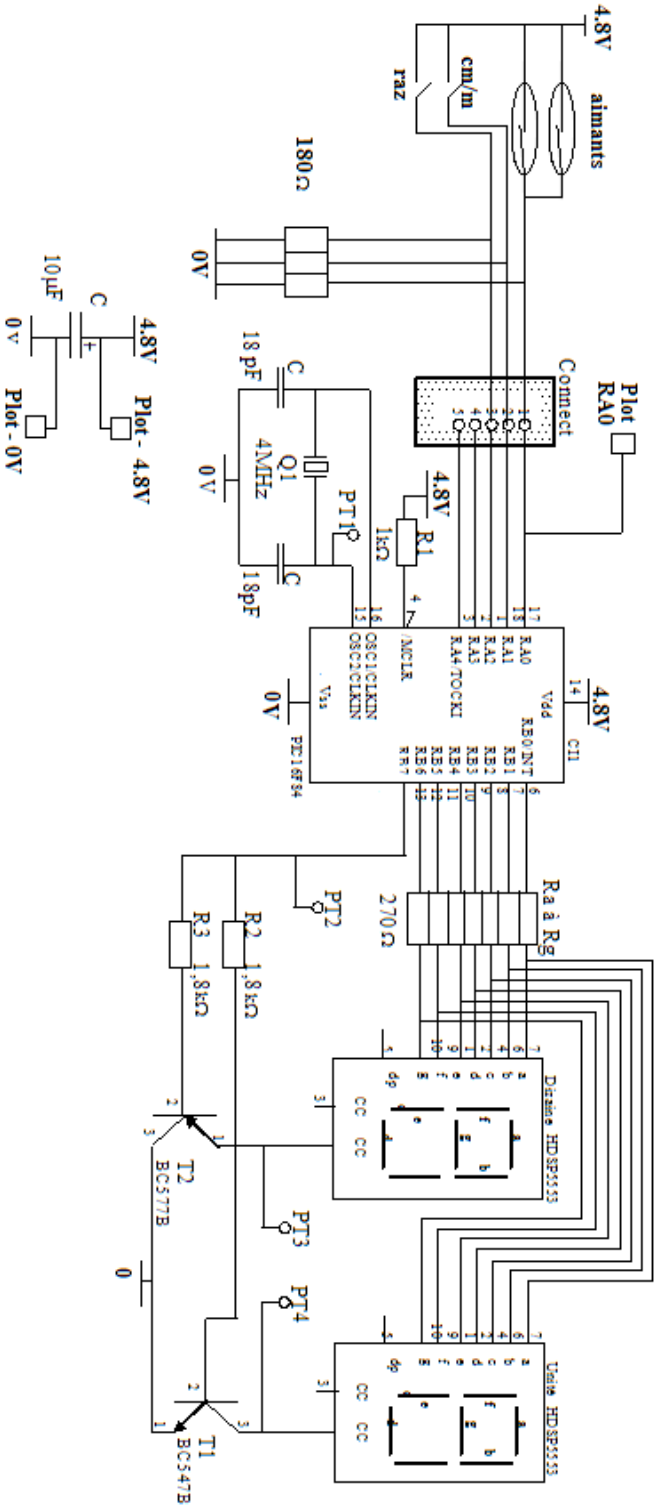
Voici donc le circuit électrique de la détection faible avec toutes les informations :



III. Conclusion

Notre prototype (ainsi que la maquette numérique) vous seront présentés lors de l'examen oral. Il s'agit d'un prototype réalisé avec les moyens que nous avions.
Le prototype fonctionne avec le circuit électrique que nous avons réalisé (sauf raccord PIC/afficheurs)

Voici donc le schéma électrique de notre prototype :



Cependant, en cas de commercialisation du distancemètre, nous aurions dû ajouter et modifier différents éléments :

- Une ergonomie plus performante, avec un système moins lourd et plus maniable
- Une partie électronique plus légère et moins encombrante.
- Installation du système de détection de batterie faible.
- Augmenter la capacité de mesure du distancemètre. En effet, dans le cas d'utilisation à plus grande échelle (chantier), les distances à mesurer sont beaucoup plus importantes.
- Améliorer l'autonomie du distancemètre (batterie).
- Une roue plus adaptée au but du distancemètre : Périmètre constant (ce qui risque de ne pas être le cas avec une roue à pneu).
- Améliorer les afficheurs afin d'afficher la distance parcourue en mètres et centimètres en même temps plutôt que de devoir utiliser un interrupteur.
- Améliorer la précision avec peut-être plus d'ILS utilisés (au centimètre prés par exemple).

Annexe : Code source

```
// Includes pour les fonctions lie au PIC
#include std84.h
#include bit84.h

// Denitions pour les afficheurs 7 segments
// Ex: pour former le 0, on utilise 6 sorties du pic : RB0,RB1,RB2,RB3,RB4,RB5).
#define ZERO portb.0=1; portb.1=1; portb.2=1; portb.3=1; portb.4=1; portb.5=1;
#define UN portb.1=1; portb.2=1;
#define DEUX portb.0=1; portb.1=1; portb.3=1; portb.4=1; portb.6=1;
#define TROIS portb.0=1; portb.1=1; portb.2=1; portb.3=1; portb.6=1;
#define QUATRE portb.1=1; portb.2=1; portb.5=1; portb.6=1;
#define CINQ portb.0=1; portb.2=1; portb.3=1; portb.5=1; portb.6=1;
#define SIX portb.2=1; portb.3=1; portb.4=1; portb.5=1; portb.6=1;
#define SEPT portb.0=1; portb.1=1; portb.2=1;
#define HUIT portb.0=1; portb.1=1; portb.2=1; portb.3=1; portb.4=1; portb.5=1;
portb.6=1;
#define NEUF portb.0=1; portb.1=1; portb.2=1; portb.3=1; portb.5=1; portb.6=1;
#define bpraz porta.2

// Denition des unitees
char unitecm,
     dizainecm,
     unitemetre,
     dizainemetre,
     milimetre,
     aimant;

//----------------------------PROGRAMME PRINCIPAL----------------------

void main()
{
    unitecm=0;
    dizainecm=0;
    unitemetre=0;
    millimetre=0;
    dizainemetre=0;

    trisa = 0b11111111; // defini le port A comme entree
    trisb = 0b00000000; // defini le port B comme sortie

    while(1) // Boucle principale
    {
        mesure(); // ajouter 3.93 cm quand un aimant passe (distance entre deux
aimants)
        calcul(); // transformer 10 mm en 1cm, 10cm en 1 dizaine de cm, 10 dizaines
de cm en 1 m et 10m en une dizaine de m.
        afficher(); // afficher la distance sur les 2 afficheurs 7 segments
    }
}

//----------------------------PROGRAMME AFFICHER----------------------
void afficher()
{
    if(porta.1) // si le bouton cm/m est en mode cm :
    {
        portb.7=1;
        switch(unitecm) //afficher la variable unitecm sur l'afficheur 7 segments
de droite
        {
            case 0 : {ZERO ; break;}
            case 1 : {UN ; break;}
```

```
            case 2 : {DEUX ; break;}
            case 3 : {TROIS ; break;}
            case 4 : {QUATRE ; break;}
            case 5 : {CINQ ; break;}
            case 6 : {SIX ; break;}
            case 7 : {SEPT ; break;}
            case 8 : {HUIT ; break;}
            case 9 : {NEUF ; break;}
        }
        portb.7=0; //afficher la variable dizainecm sur l'afficheur 7 segments de
gauche

        switch(dizainecm) //afficher la variable unitecm sur l'afficheur 7 segments
de droite
        {
            case 0 : {ZERO ; break;}
            case 1 : {UN ; break;}
            case 2 : {DEUX ; break;}
            case 3 : {TROIS ; break;}
            case 4 : {QUATRE ; break;}
            case 5 : {CINQ ; break;}
            case 6 : {SIX ; break;}
            case 7 : {SEPT ; break;}
            case 8 : {HUIT ; break;}
            case 9 : {NEUF ; break;}
        }
    }
    else // si le bouton cm/m est en mode m :
    {
        portb.7=1; //afficher la variable unitemetre sur l'afficheur 7 segments de
droite

        switch(unitemetre) //afficher la variable unitecm sur l'afficheur 7
segments de droite
        {
            case 0 : {ZERO ; break;}
            case 1 : {UN ; break;}
            case 2 : {DEUX ; break;}
            case 3 : {TROIS ; break;}
            case 4 : {QUATRE ; break;}
            case 5 : {CINQ ; break;}
            case 6 : {SIX ; break;}
            case 7 : {SEPT ; break;}
            case 8 : {HUIT ; break;}
            case 9 : {NEUF ; break;}
        }
        portb.7=0; //afficher la variable dizainemetre sur l'afficheur 7 segments
de gauche

       switch(dizainemetre) //afficher la variable unitecm sur l'afficheur 7
segments de droite
        {
            case 0 : {ZERO ; break;}
            case 1 : {UN ; break;}
            case 2 : {DEUX ; break;}
            case 3 : {TROIS ; break;}
            case 4 : {QUATRE ; break;}
            case 5 : {CINQ ; break;}
            case 6 : {SIX ; break;}
            case 7 : {SEPT ; break;}
            case 8 : {HUIT ; break;}
            case 9 : {NEUF ; break;}
        }
    }
}

//----------------------------PROGRAMME MESURE----------------------
void mesure()
```

```
{
    if(!porta.0) // si il n'y a pas de detection de l'aimant
    {
        aimant=0; // enregistrer dans une variable qu'il n'y a pas d'aimant
    }
    else
    {
        if(aimant==0) // et si pas d'aimant avant (aimant=0), alors il faut ajouter
3.93 cm :
        {
            milimetre = milimetre +93; // ajouter 93/10 mm = 9.3mm □a variable
milimetre
            unitecm = unitecm + 3; // ajouter 3 cm a la variable unitecm
            aimant = 1; // recommencer la detection de l'aimant.
        }
    }
}

//----------------------------PROGRAMME CALCUL----------------------
void calcul()
{
    if(milimetre>=100) //si millimetre □lus de 10mm (100/10 mm)
    {
        unitecm++; // ajouter 1cm
        milimetre=milimetre-100; // enlever 10 mm (100/10 mm)
    }
    if(unitecm >= 10) //convertir 10cm en 1 dizaine de cm
    {
        unitecm=unitecm-10;
        dizainecm++;
    }
    if(dizainecm >= 10) //convertir 10dizaines de cm en 1 metre
    {
        dizainecm=dizainecm-10;
        unitemetre++;
    }
    if(unitemetre >= 10) //convertir 10 metres en 1 dizaine de metres
    {
        unitemetre=unitemetre-10;
        dizainemetre++;
    }
    if(dizainemetre>9 || bpraz) // si 100m atteint ou remise □ par bouton poussoir
(bpraz)
    {
        // tout remettre a 0 (pour eviter un probleme d'affichage)
        unitecm=0;
        dizainecm=0;
        unitemetre=0;
        dizainemetre=0;
        milimetre=0;
    }
}
```