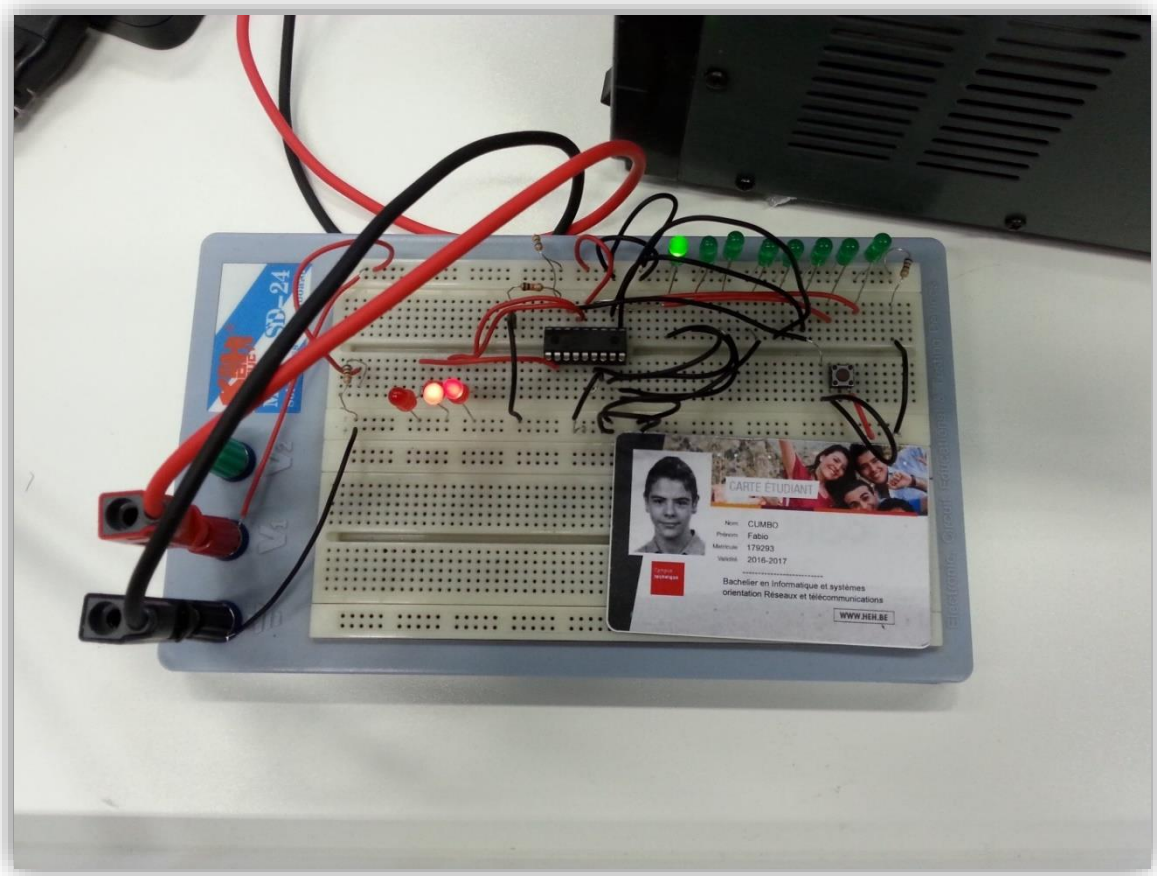
HAUTE ECOLE EN HAINAUT
Campus technique
8a, avenue V. Maistriau B-7000 Mons
Tél : +32 (0)65 33 81 54
Fax : +32 (0)65 31 30 51
E-mail : tech-mons@heh.be
www.heh.be



Internet of Everything – Project
Guide technique

Mai 2017
Version 3.0
Fabio Cumbo

Enseignant :
ARNAUD David
DESMET Erwin

1. Informations générales sur le document

Contact

Pour toute question ou remarque concernant ce document, merci de contacter :

Fabio Cumbo - Etudiant
Mail : fabio.cumbo@std.heh.be
HEH Campus Technique
8A Avenue Maistriau,
7000 Mons

Confidentialité

Ce document contient des informations confidentielles et exclusives de la Haute École en Hainaut (HEH). Le service informatique ne peut divulguer les informations confidentielles contenues dans ce document à un tiers sans le consentement écrit de la HEH, hormis aux employés, enseignants ou directeurs qui ont besoin de connaître son contenu à des fins d'évaluation du document. Le service informatique se doit d'informer ces personnes de la nature confidentielle de ce document et d'obtenir leur accord pour préserver sa confidentialité.

Informations sur le document

Nom du document	IoE project Technique.Docx
Version	Version 3.00
Niveau de confidentialité	Utilisation interne uniquement
Auteur du document	Fabio Cumbo
Contributeur(s)	
Révisé par	
Approuvé par	

Versions

Version	Date de parution	Modification réalisée par	Modification(s) apportée(s)
1.00	25/04/2017	Fabio Cumbo	Création du document
2.00	29/05/2017	Fabio Cumbo	Modification du code pour régler un problème lorsqu'on relâchait le bouton à la dernière LED dans le mode automatique.
3.00	30/05/2017	Fabio Cumbo	Amélioration de l'optimisation du code.

Documents connexes et/ou de référence

Nom du document	Description	Date
Q1 ETUDE DE PROJET 2016_Testeur_de_cables_reseauV1	Document du cours d'étude de projet du premier quadrimestre	16 sept 2016
Q2 EDPsuite 2016, Projet dinterfacage informatique.pdf	Document du cours de projet d'interfaçage informatique du deuxième quadrimestre	6 fevrier 2017

Table des matières

2. Introduction

L'objectif de cette fiche technique est de documenter la réalisation d'un testeur de câble réseau (RJ45) dans le cadre du cours d'étude de projet et du cours de projet d'interfaçage.
Les outils et explications nécessaire à la conception de ce projet seront donc élaboré dans ce document.

3. Logiciels et matériels nécessaire à la réalisation du projet

Matériels :

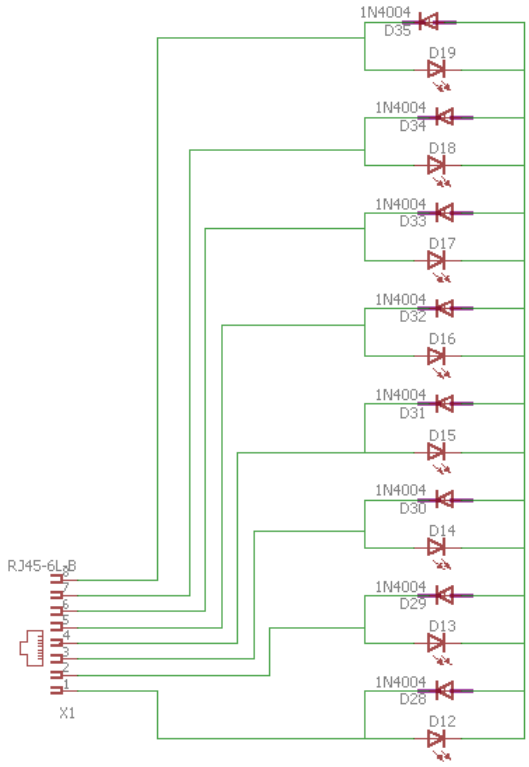
- ✓ La liste des composants électronique (5.1.)
- ✓ Une breadboard : Simulation réelle
- ✓ Une pile 9V pour la breadboard
- ✓ Le Pickit 3 pour injecter le code dans le PIC

Logiciels :

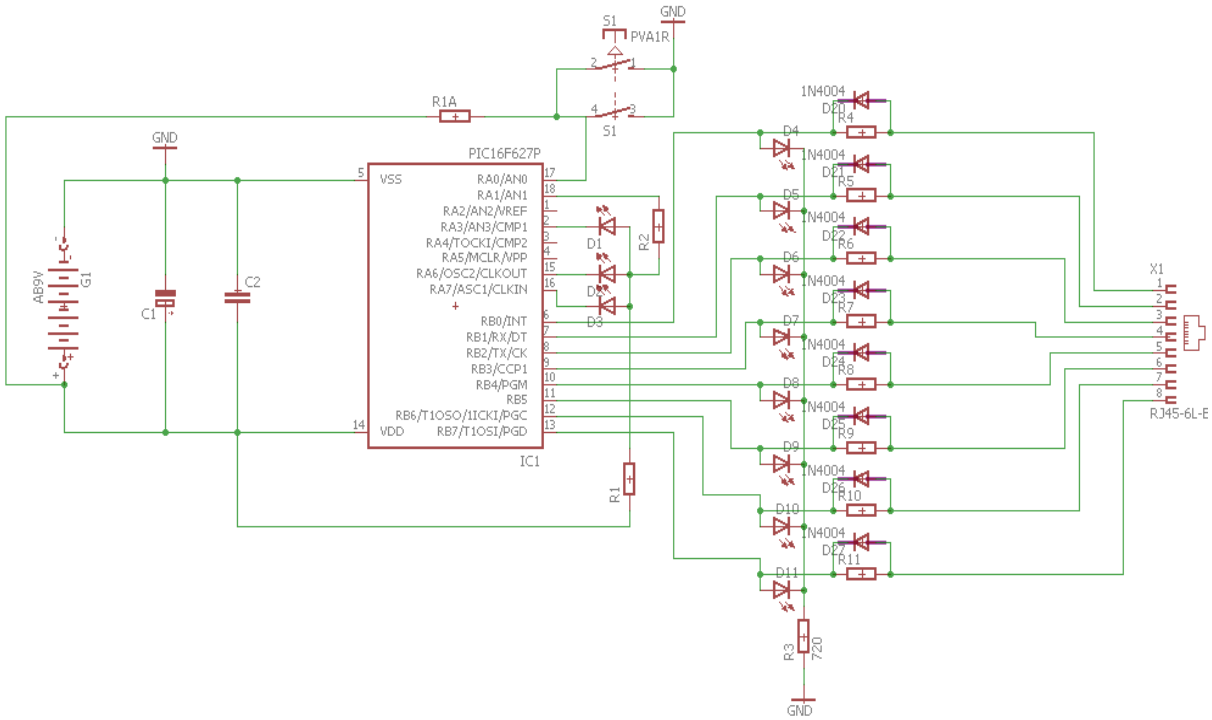
- ✓ Eagle : Conception de la schématique et réalisation de la board
- ✓ LARP : Conception d'ordinogrammes pour aider a réaliser l'algorithme
- ✓ MikroC : IDE pour programmer le microcontrôleur PIC du testeur de câble
- ✓ Proteus : Simulateur pour tester le programme
- ✓ Fritzing : Simulation du câblage de la breadboard
- ✓ MPlab : IPE pour injecter le code dans le PIC

4. Schéma générale sur Eagle

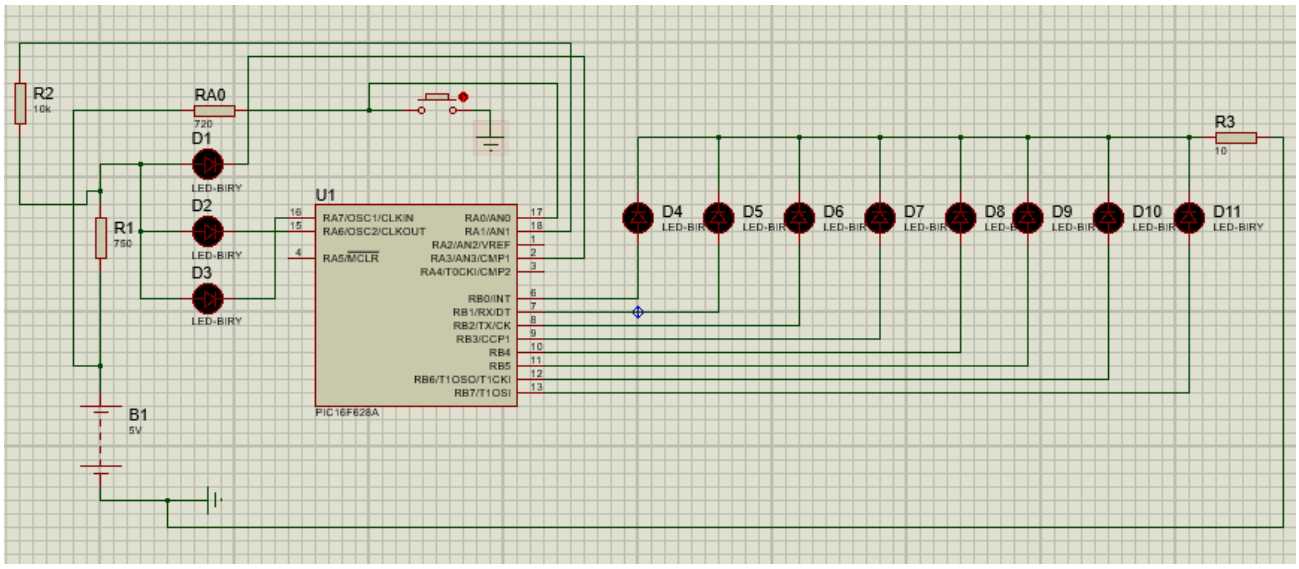
4.1. Module LED



4.2. Module PIC



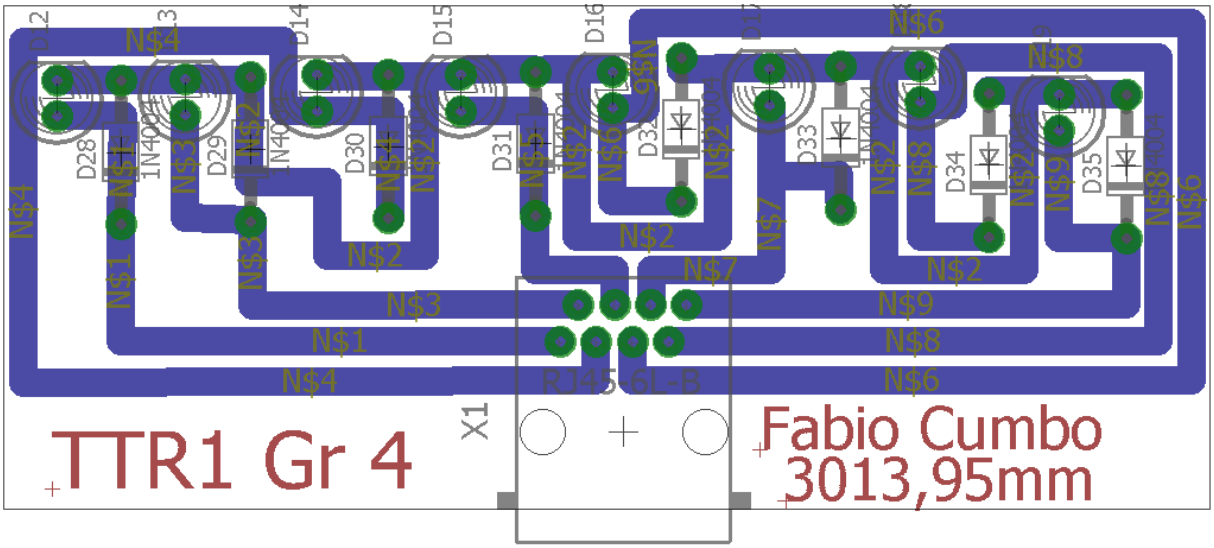
4.3. Schéma sur Proteus



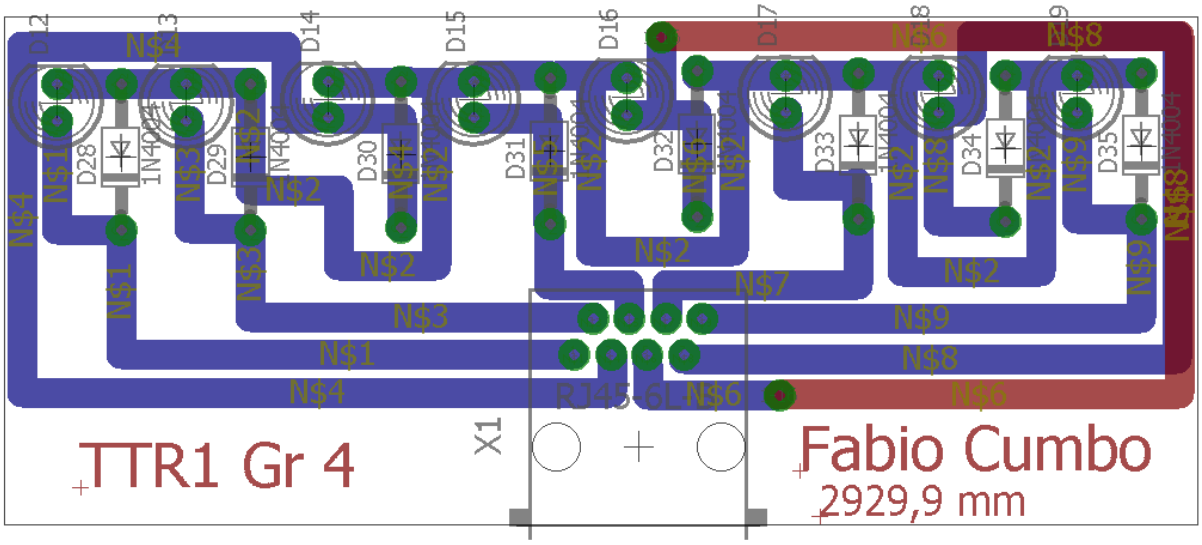
5. Les Boards

5.1. Module LED simple face et double face

Simple face :

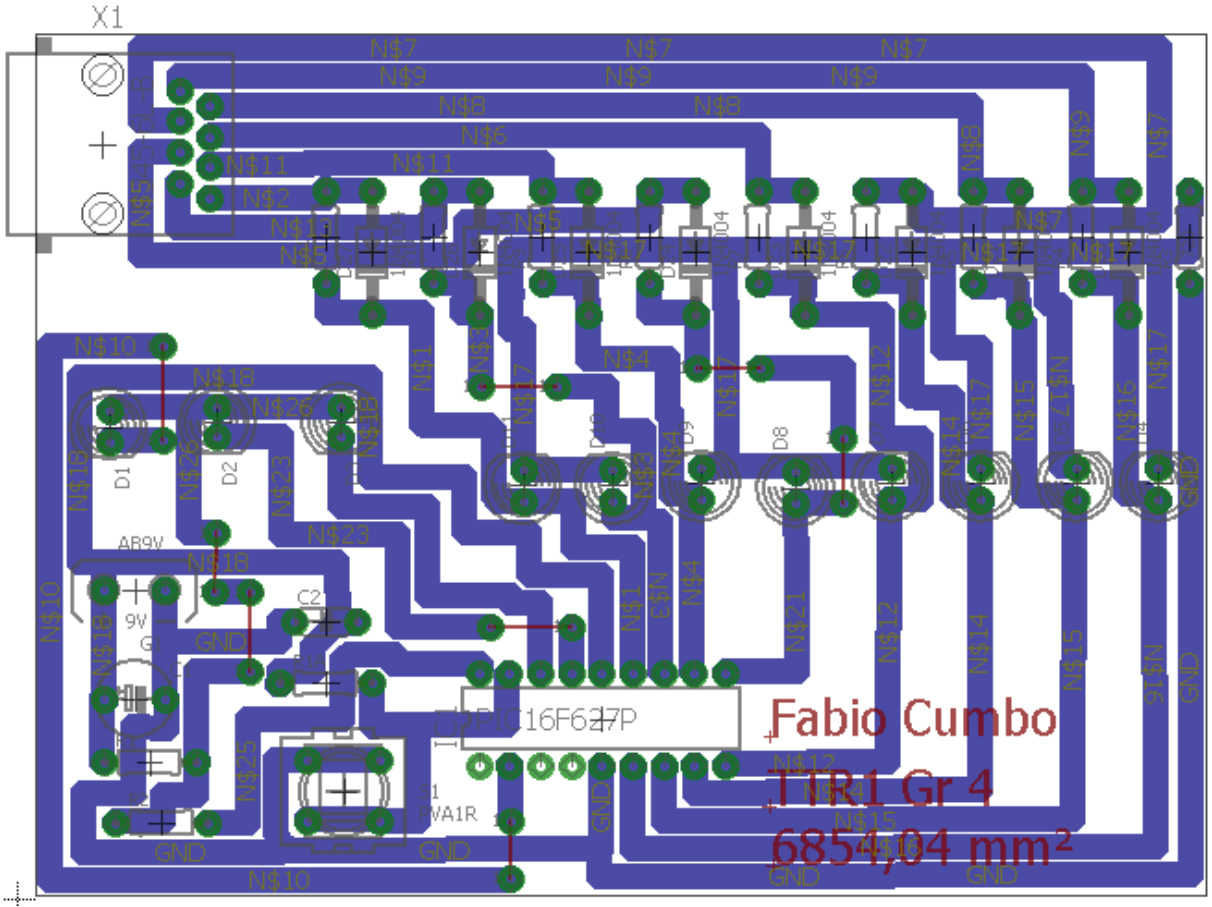


Double face :

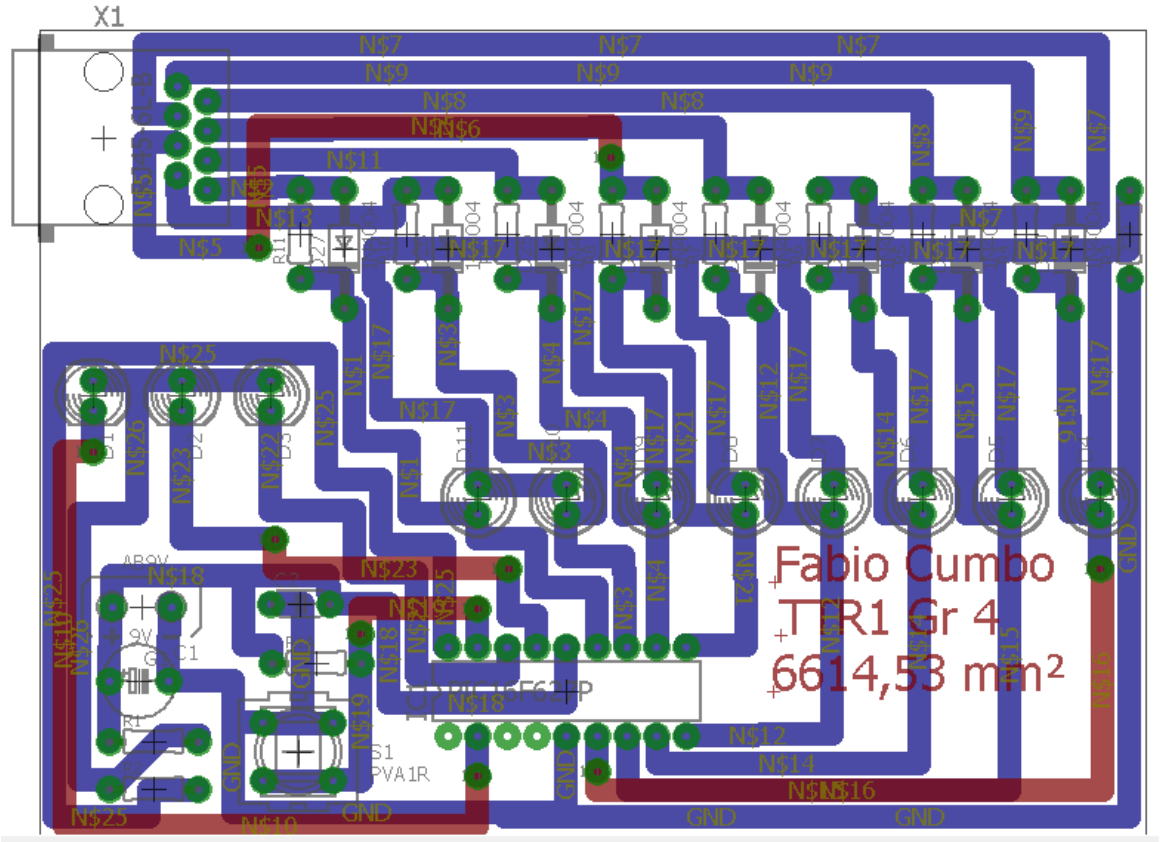


5.2. Module PIC simple face et double face.

Simple face :



Double face :



6. Les composants.

6.1. *Liste des composants*

Liste des composants	Quantité	Référence Eagle
Les LEDs 5 mm	11	LED5MM
Les diodes 1N4004	8	1N4004
Un connecteur RJ 45	1	RJ45-6L-B
Un PIC 16F627A	1	PIC16F627P
Les résistances	8x470, 1x680 et 1x10K	R-EU_0204/7
Un condensateur	1x100uF/16V	CPOL-EUE5-6
Un condensateur polarisé	1x100nF	C-EU050-024X044
Un Micro switch	1	PVA1R
Une pile	1	AB9V
Les LEDs 5 mm	8	LED5MM

Les diodes 1N4004	8	1N4004
Un connecteur RJ 45	1	RJ45-6L-B

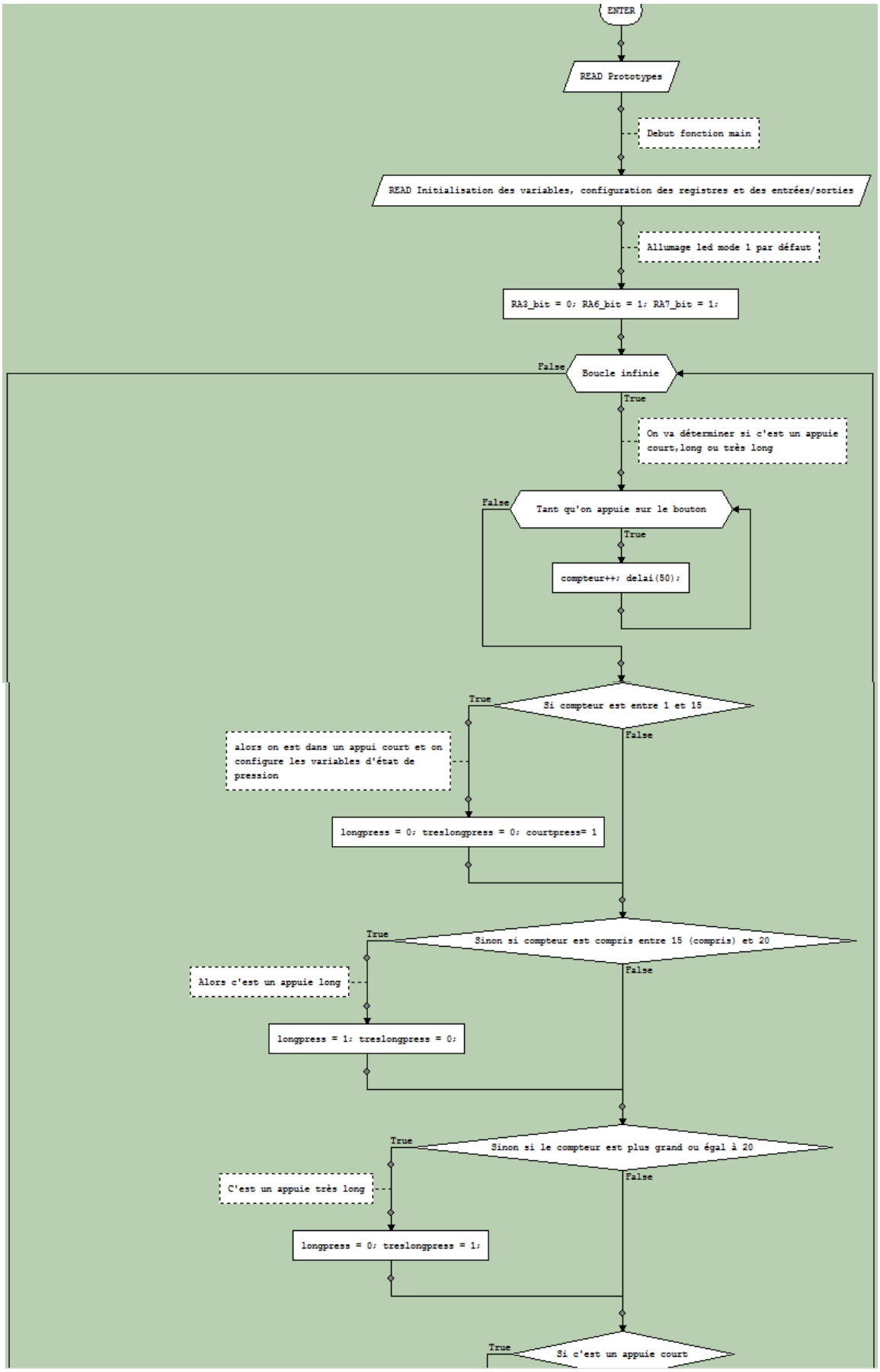
6.2. Comment obtenir des composants gratuitement

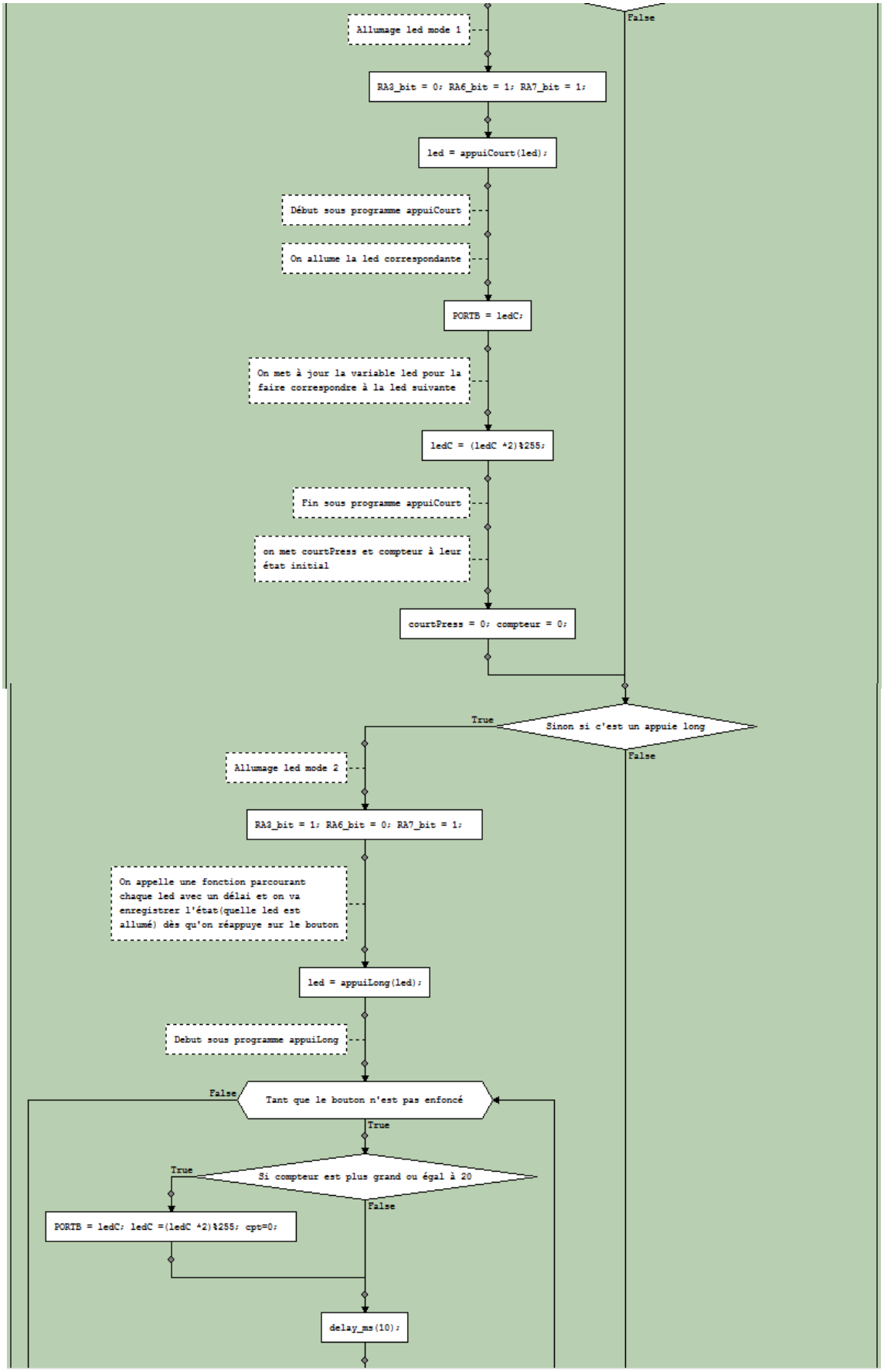
L'entreprise Microchip offre la possibilité de se procurer gratuitement jusqu'à 6 microcontrôleurs PIC par trimestre (2 x 3).

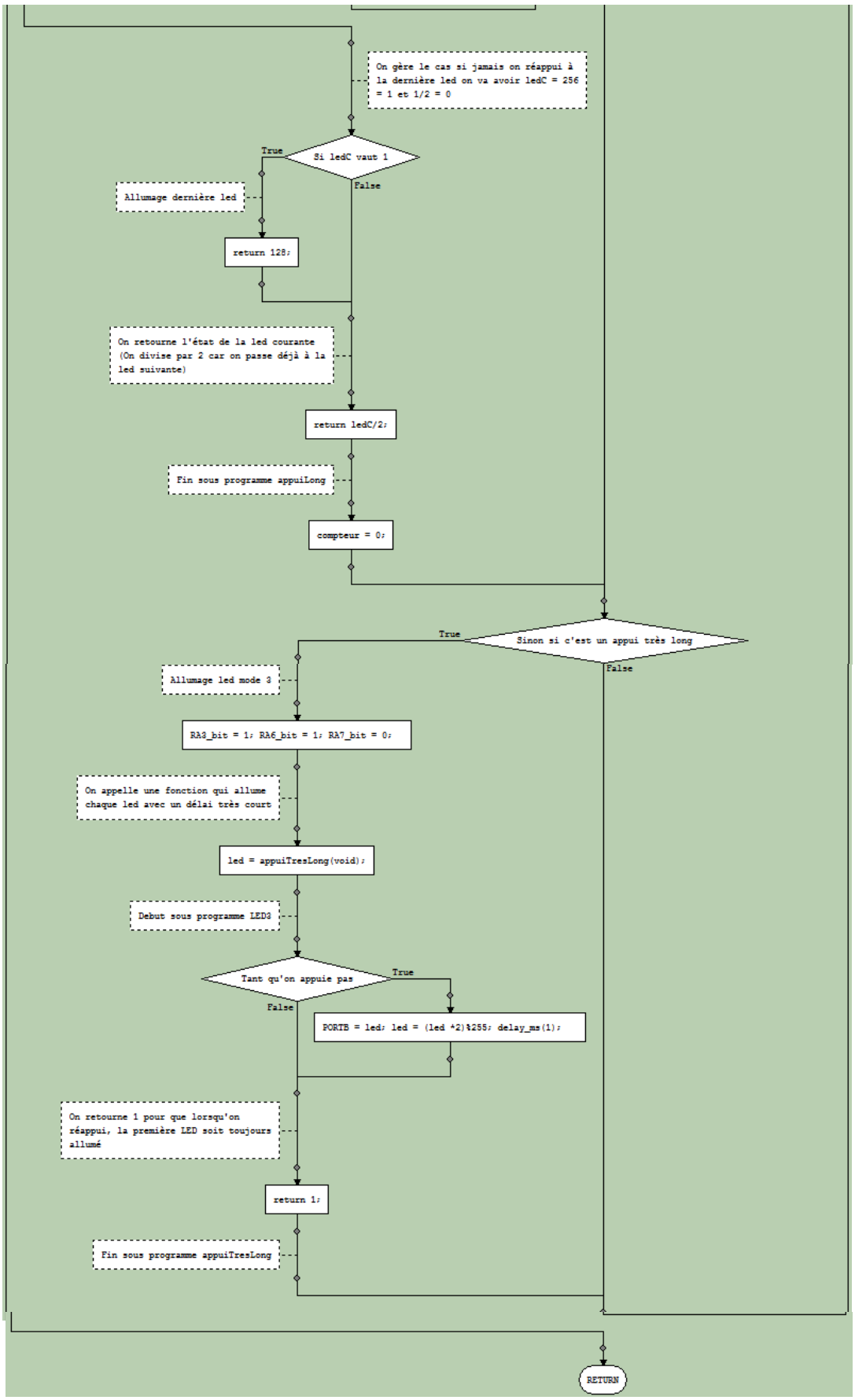
Il suffit de s'inscrire sur leur site internet : http://www.microchip.com/ et ensuite aller sur l'onglet « SAMPLE AND BUY et de choisir ces microcontrôleur ainsi que la quantité voulu.

7. Ordinogramme

7.1. LARP







8. Programmation

8.1. Programmation du PIC

Pour développer le code du testeur de câble, nous avons utilisé le langage C ainsi que l'IDE MikroC développé par l'entreprise Mikroelektronika.
Il existe 2 possibilités pour pouvoir injecter le code dans le microcontrôleur PIC :

- Avec une Plaquette de développement EasyPic réalisé par Mikroelectronika
- Avec un programmateur PicKit et l'IPE MPlab

Nous avons utilisé la deuxième solution. Un fichier. HEX a été généré après compilation du code par MikroC et ce fichier a été introduit dans le PicKit pour pouvoir l'injecter dans le PIC.

Par ailleurs, nous avons utilisé le logiciel Proteus pour simuler le testeur de câble et s'assurer que le code est fonctionnel avant de passer à l'injection.

8.2. Premier

Commande	Description

`while(RA0_bit==0)`
`{`
`  compteur++ ;`
`  delay_ms(50);`
`}`
	Cette partie va permettre d'incrémenter un compteur tous les 50 ms pour déterminer par la suite le type d'appui

`if(compteur >= 1 && compteur < 15`
`{`
`longPress = 0;`
`tresLongPress = 0;`
`courtPress = 1;`
`}`
	On configure les paramètres pour un appui court

`else if (compteur >= 15 && compteur <20)`
`{`
`longpress = 1;`
`treslongpress = 0;`
`}`
	On configure les paramètres pour un appui long

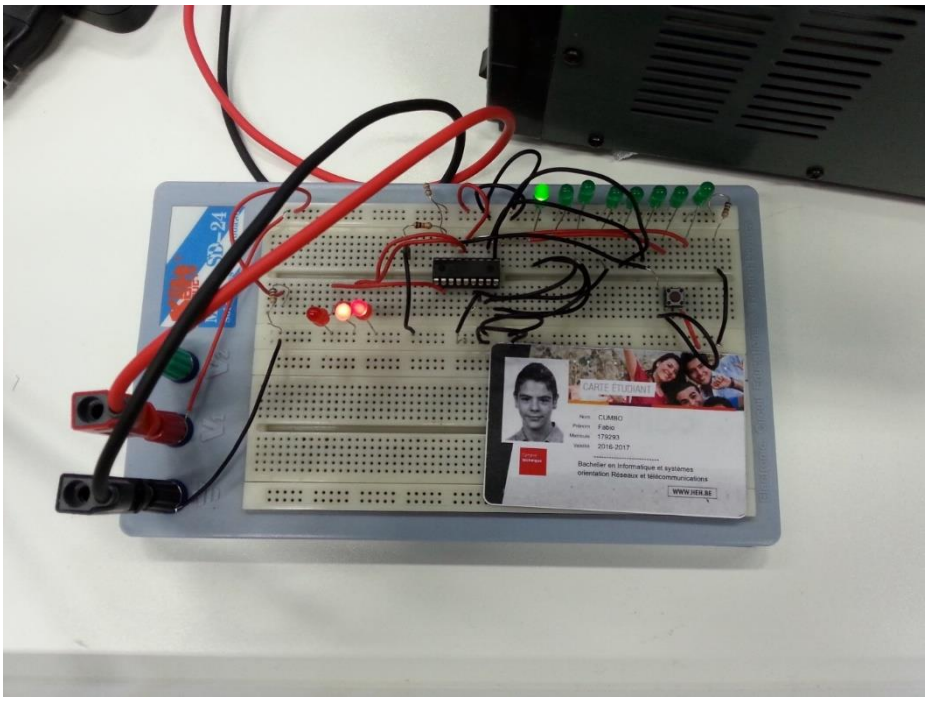
`else if (compteur >= 20)`
`{`
`treslongpress = 1;`
`longpress =0;`
`}`
	On configure les paramètres pour un appui à 1 KhZ

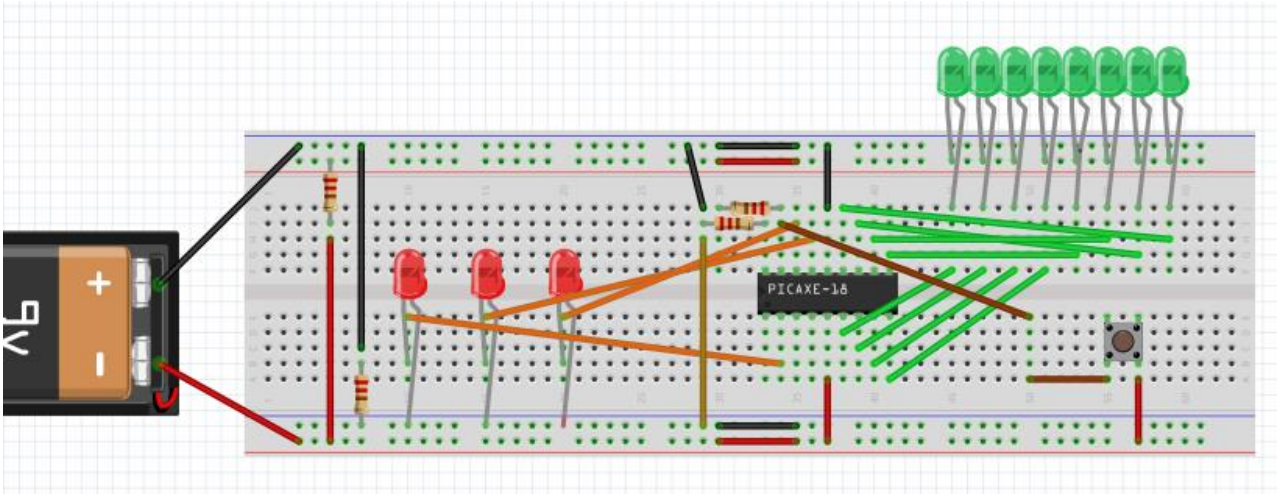
`char appuiCourt(char ledC)`
`{`
`  PORTB = ledC;`
`  ledC = (ledC *2)%255;`
`}`
	Sous-programme permettant d'allumer la led correspondante en mode appui court. A chaque appui on va repasser par cette procédure qui va mettre à jour la led en faisant *2 pour "décaler" un bit vers la droite et ainsi passer à la led suivante et %255 pour retourner à la première si on est a 256
	
	PORTB va donc prendre va donc prendre une de ses valeurs : 1-2-4-8-16-32-64-128

`char appuiLong(char ledC)`
`{`
`  int cpt =0;`
`  for(cpt = 0; RA0_bit == 1; cpt++)`
`  {`
`  if(cpt>=20)`
`  {`
`  PORTB = ledC;`
`  ledC = (ledC *2)%255;`
`  cpt=0;`
`  }`
`  delay_ms(10);`
`  }`
`  if (ledC == 1)`
`  {`
`  return 128;`
`  }`
`  return ledC/2;`
`}`
	Sous-programme pour allumer chaque led en mode automatique (chenillard)
	
	Tant qu'on appui pas, on incrémente un compteur pour déterminer le temps entre chaque led correspondante qui vont s'allumer avec un délai supplémentaire
	
	Si on réappuie pour changer de mode, on renvoie la valeur de ledC/2 car on est déjà passé à la suivante et on veut s'arrêter à la led courante si on réappuie.
	
	Cependant si ledC = 128 et qu'on passe à la suivante, on va avoir ledC = (128 * 2) %255 = 1 et si on reappuie sur le bouton pour changer de mode, on va avoir ledC = ½ = 0 et donc on aura aucune led d'allumer par la suite car (0 * 2) % 255 fera toujours 0. On doit donc gérer ce cas en renvoyant la dernière led si on réappuie à la dernière (donc 128)

`char appuiTresLong()`
`{`
`  char led = 1;`
`  while(RA0_bit==1)`
`  {`
`  PORTB = led;`
`  led = (led *2)%255;`
`  delay_ms(1);`
`  }`
`  PORTB = 0`
`  return 1;`
`}`
	Sous-programme permettant d'allumer toutes les leds en même temps
	
	Tant qu'on n'appuie pas, on fait la procédure pour passer à la led suivante mais avec un délai très court (1ms)
	
	Si on réappuie, on va toujours retomber sur la première led autrement, une led au hasard s'allumerait à chaque fois qu'on réappuie sur le bouton

9. Test sur plaquette d'essai ou sur board.





10. Amélioration du code

Une amélioration du code permettant d'améliorer les fonctionnalités du testeur serait d'allumer la LED du mode correspondant avant de relâcher le bouton pour que la personne sache dans quel mode il se trouve avant de relâcher le bouton

Une autre amélioration aurait été de simplifier le moyen de détecter le type d'appui en utilisant seulement la variable compteur pour cela.

11. Synthèse et conclusion

11.1. Problème rencontré avec analyse des solutions.

- La disposition des composants du testeur pouvait être problématique. En effet pour permettre un routage plus facile, la disposition devait être logique et cohérente. Dans le cas contraire, il était parfois compliqué de réaliser les ponts.

- La difficulté à optimiser la taille du PCB m'a un peu posé problème. En effet, certains composants ont dû être déplacé pour diminuer la taille de la plaquette mais il fallait également penser à améliorer le routage de manière à éviter les pistes obliques pour permettre de gagner un maximum de place.

- Au niveau de la programmation du PIC, il a été difficile au départ de trouver le moyen de créer une boucle pour générer directement la séquence d'allumage et extinction automatique des LEDs sans devoir réécrire plusieurs fois la même chose. Il a fallu se rendre compte que l'on pouvait contrôler les LEDs facilement avec la variable PORTB et faire varier son état à chaque itération pour les 2 modes automatiques

11.2. Synthèse aboutissant sur une conclusion avec appréciation.

Nous avons abordé une thématique qui m'était pratiquement inconnu, j'ai donc appris de nouvelles choses qui je pense, peuvent m'être utile pour l'avenir. J'ai donc trouvé le cours intéressant pour ces raisons. Le projet s'est en général assez bien passé malgré une mauvaise appréhension au départ et les quelques problèmes rencontrés qui ont pu être résolu.

Annexes

11.3. Toute la programmation.

```
// ******** Déclaration des prototypes ********
 char appuiLong(char ledC);
 char appuiCourt(char ledC);
 char appuiTresLong();

void main()
{
// ********  Déclaration des variables ********
char longpress = 0;     //Ici , on choisis des char pour ne pas allouer des espaces mémoires
inutiles
char treslongpress = 0;
int compteur = 0;
char led = 1;
char courtpress = 0;

TRISA = 0b00000011 ;  // port A en sortie sauf  RA0 et RA1
PORTA = 0;          // mettre les sorties à zéro
TRISB = 0b00000000;     // Port B en sortie
PORTB = 0;          // mettre les sorties à zéro
pcon.OSCF = 1;       // configure le bit 3 du registre pcon  pour 4 mhz
CMCON = 0b00000111;

      RA3_bit = 0;  // On allume la première led par défault pour savoir si c'est bien alimenté
      RA6_bit = 1;
      RA7_bit = 1;
       while(1)
       {
         while(RA0_bit==0)    // Tant qu'on presse le bouton
         {
          compteur++;       // On va incrémenter un compteur pour déterminer ensuite le type
d'appui
          delay_ms(50);
         }

// ******** On va déterminer le type d'appui ********

          if(compteur >= 1 && compteur < 15)   //si compteur est entre 1 et 15
          {
           //On configure les paramètres pour un appui court
```

```
                longpress = 0;
                treslongpress = 0;
                courtress = 1;
               }
               else if (compteur >= 15 && compteur <20)
               {
                 //On configure les paramètres pour un appui long
                 longpress = 1;
                 treslongpress = 0;
               }
               else if (compteur >= 20)
               {
                 //On configure les paramètres pour un appui très long
                 treslongpress = 1;
                 longpress =0;
               }

// ******** On effectue les actions en fonction du type d'appui ********

              if(courtPress == 1)  //Si c'est un appuie court
              {
                  RA3_bit = 0;  //Allumage led mode 1
                  RA6_bit = 1;
                  RA7_bit = 1;

                  led = appuiCourt(led);  //Appel fonction pour un appui court, la led va retenir la
position et va devoir prendre une de ces valeurs: 1-2-4-8-16-32-64-128
                  courtPress = 0;
                  compteur = 0;
              }
              else if (longpress == 1)   //Si on est dans un appuie long
              {
                RA3_bit = 1;   //Allumage led mode 2
                RA6_bit = 0;
                RA7_bit = 1;

                led=appuiLong(led); //Appel fonction appui long
                compteur = 0;
              }
              else if (treslongpress == 1) // Appui très long
              {
                RA3_bit = 1; //Allumage led mode 3
                RA6_bit = 1;
```

```
            RA7_bit = 0;

            led=appuiTresLong(void); //appel fonction mode 1khz
            compteur = 0;
          }
      }
}

// ******** Sous- programmes ********

 char appuiCourt(char ledC)    //Appui simple
 {
   PORTB = ledC;  //On allume la led correspondante
   ledC = (ledC *2)%255;  //On met à jour cette led pour la faire correspondre à la led suivante
                  //On fait donc *2 pour "décaler" un bit vers la droite et ainsi passer à la suivant
                 // et %255 pour retourner à la première si on est a 256
 }

 char appuiLong(char ledC)  //Appui long
{
   int cpt =0;
   for(cpt = 0; RA0_bit == 1; cpt++)   //Tant qu'on appui pas
   {
     if(cpt>=20)     // On incrémente un compteur pour déterminer le temps entre chaque led
allumé
     {
       PORTB = ledC;
       ledC = (ledC *2)%255;
       cpt=0;
     }
     delay_ms(10);
   }
   if (ledC == 1)  //On gère ce cas pour éviter d'avoir ledC = 256 = 1 et 1/2 = 0
   {
     return 128;   //La dernière led s'allume
   }
   return ledC/2;     //Si on relache on divise par 2 pour avoir la led courante car on a déjà
passé à la suivante
}

 char appuiTresLong()     //Appui très long
 {
   char led = 1;
```

```
  while(RA0_bit==1)  // Tant qu'on appui pas
  {
    PORTB = led;
    led = (led *2)%255;
    delay_ms(1);
  }
  PORTB = 0;   //si on réappuie, on allume d'office la première led
  return 1;
}
```