|  |
| --- |
|  |
| Jeux de lumières |
| *Découverte de la programmation sur microcontrôleur* |
|  |
| **BARBESANGE Benjamin, GARCON Benoît** |
| **mars 2014** |

|  |
| --- |
|  |

Table des matières

[Introduction 2](#_Toc384069880)

[Présentation du projet 2](#_Toc384069881)

[Carte EasyPIC 2](#_Toc384069882)

[LED RGB 3](#_Toc384069883)

[Présentation du programme 3](#_Toc384069884)

[Partie 1 - La matrice de LED 4](#_Toc384069885)

[Partie 2 - L'écran GLCD 5](#_Toc384069886)

[Partie 3 - La LED RGB 6](#_Toc384069887)

[Port PWN "classqique" 6](#_Toc384069888)

Introduction

# Présentation du projet

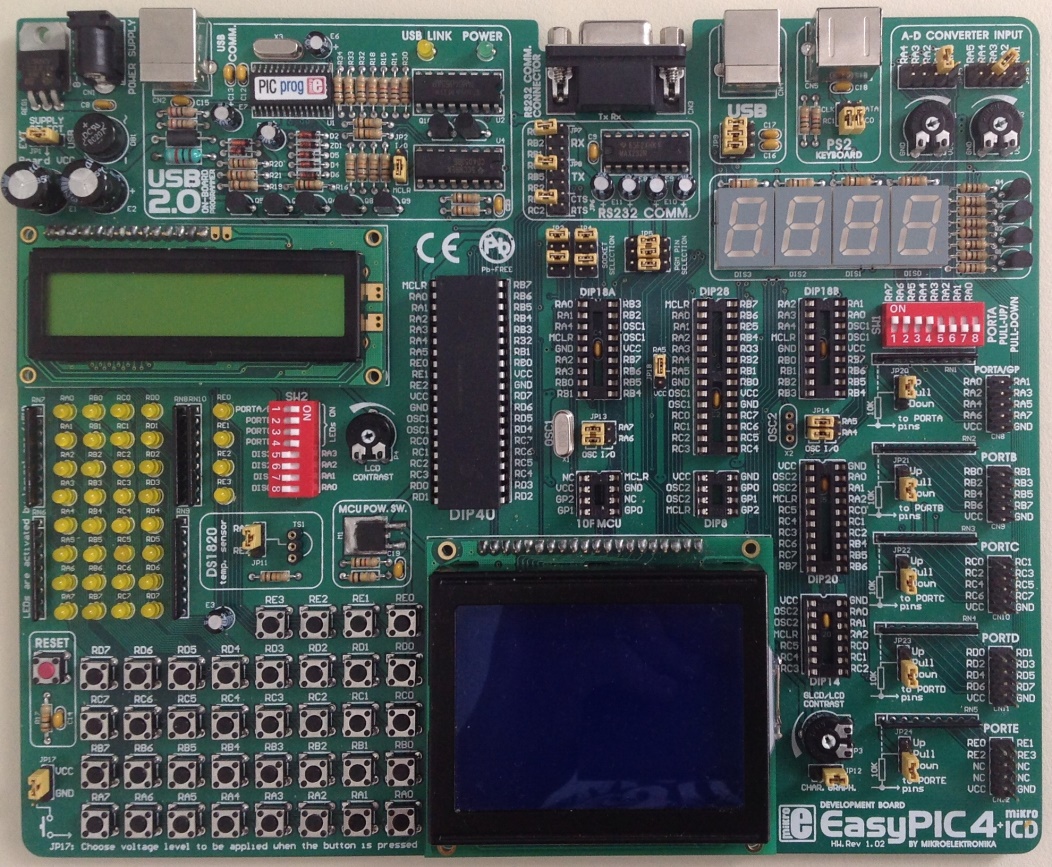
Dans le cadre de la deuxième année de Prep' ISIMA, il nous est demandé d'effectuer un travail de groupe occupant 24h de travail. Le sujet que nous avons ainsi choisi est le sujet "Jeux de lumière".

Afin de pouvoir réaliser ce projet, le matériel suivant nous a été fourni :

* Une carte EasyPIC
* 6 résistances
* Une LED RGB
* Une plaquette de branchement

## Carte EasyPIC

La carte EasyPIC fournie est le modèle 4. C'est la base du projet, qui va nous permettre d'interagir avec les différents composants que nous avons a disposition. De plus, notre carte est composée d'un écran LCD ainsi que d'un écran GLCD sur lequel il est possible d'afficher des images. Ce sont les deux principaux éléments sur lesquels nous intervenons.



Matrice de LED

Ecran GLCD

Figure - Carte EasyPIC 4

## LED RGB

En plus de la carte PIC, une LED RGB nous est fournie. Celle-ci vient accompagnée de plusieurs résistances de 1k Ohms (afin d'éviter de griller la LED). De plus, une plaquette de branchement ainsi que des fils viennent s'ajouter afin de pouvoir relier cette LED à la carte PIC, qui va ainsi déterminer de fonctionnement de cette LED.

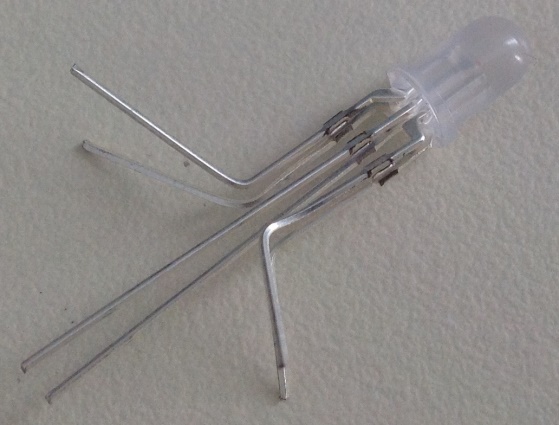


Figure - LED RGB

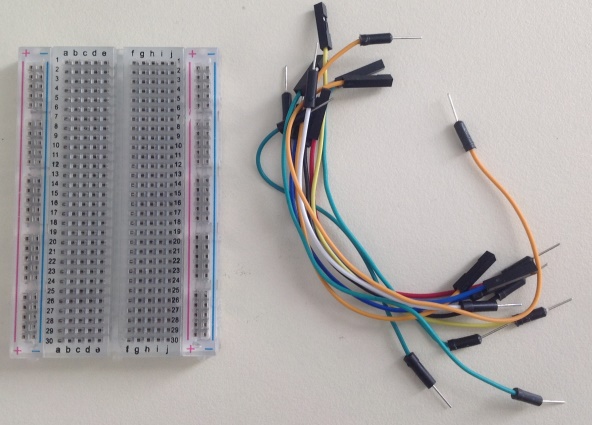


Figure - Plaquette de branchement et fils

# 

# Présentation du programme

Plusieurs langages de programmation sont disponibles pour la carte PIC. Etant plutôt entrainés en C, c'est donc tout naturellement que nous nous sommes orientés vers ce langage.

1. La matrice de LED

Nous nous sommes mis à travailler sur le projet le mardi 21 janvier mais jusqu’au jeudi 23 nous ne sommes pas parvenus à faire fonctionner l'exemple LedBlinking. Nous avons installé le driver ainsi que les programmes associés : mikroProg Suite For PIC qui est une interface avec la carte et mikroC PRO for PIC qui est notre IDE. Ensuite nous avons ouvert le projet LedBlinking, puis changé la montre (20Mhz) et changé le modèle (P18F452). Au branchement de la carte les voyants POWER et USB se sont allumés et une faible lueur a jaillit d'une des LED de la matrice. Lorsque nous avons chargé le programme sur le MCU cette LED "défectueuse" clignote une fois très faiblement sans qu'il ne se passe quoi que ce soit.

Pour essayer de trouver la source du problème nous avons essayé de changer le type de MCU mais le logiciel nous dit bien que le bon est le P18F452, nous avons changé de PC (OS plutôt) pensant que c'était dû à un problème de driver, nous avons essayé avec tout un tas d'autres programme, etc.

Lorsque l'on s’est documenté sur internet, nous avons bien vu qu'une simple "compilation écriture" ou pression sur Ctrl+F11 devait suffire à faire tout le travail mais chez nous la partie logicielle ne renvoie aucune erreur et la partie matérielle ne répond pas (hormis un sursaut faible de la LED faiblement allumée).

Nous nous sommes donc rendu dans le local d’ISIBOT et les conclusions ont été rapides : le EasyPIC 4 était défectueux, nous en avons donc reçu une nouvelle ainsi qu’un nouveau quartz de 8Mhz par la même occasion. Nous nous sommes ainsi lancés dans la programmation le jeudi 23 janvier.

Nous nous sommes ainsi lancés dans l’élaboration de notre alphabet numérique et avons ainsi pu découvrir le fonctionnement pratique de la matrice de diodes. Après avoir déterminé un style adapté à la matrice 4\*8 pour chaque chiffre nous les avons codés et testés. Tout fonctionnait parfaitement excepté les LED RA4, RA6 et RA7. Effectivement RA6 et RA7 sont utilisées pour l’oscillateur et non pour les entrées/sorties. On nous a donc conseillé de considérer ces LED comme des "pixel mort" et de faire comme s’ils fonctionnaient mais nous avons continué à chercher un moyen d’utiliser ces LED. Ainsi nous avons pu mettre en place un programme de compte et de décompte de 0 à 9 secondes.

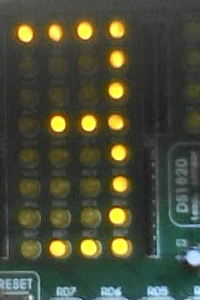


Figure - Affichage du chiffre 3 sur la matrice de LED

1. L'écran GLCD

Nous nous sommes ensuite intéressés au reste du matériel et avons aussi réussi à faire afficher le logo de l’ISIMA sur l’écran GLCD.



Figure - Affichage du logo ISIMA sur l'écran GLCD

1. La LED RGB

Nous disposons à présent d’une LED RGB à quatre pattes, de six résistances identiques et de deux LED dites classiques. Après avoir reçu la LED RGB nous l'avons branchée en série avec les résistances associées nous avons réussi à faire clignoter chaque couleur allant de l'intensité la plus faible à l'intensité la plus forte.

# Port PWM "classqique"

Les deux ports PWM correspondent au port RC1 et RC2. N'ayant que deux port PWM nous ne pouvons donc utiliser que deux couleur à la fois nous avons choisi dans un premier temps d'utiliser la LED rouge et la LED verte.

Afin d'utiliser le maximum de possibilités de la LED RGB nous avons décidé de faire une fondue de la couleur verte à la couleur rouge puis inversement en passant par toutes les gammes d'intensité disponibles. La prise en main de la librairie dite PWM a été assez rapide. En effet une fonction très simple permet de régler le " duty " : paramétrable de 0 à 255 cela règle la largeur d’impulsion de RC1 et RC2 respectivement de 0 à 100%. Il a alors été assez aisé d’établir des mélanges de couleur.

# Création d'un port PWM

Une fois notre travail terminé avec le port pwm integré sur la carte, il nous a été demandé de recréer un tel port sur une patte n'ayant pas la possibilité de créer de base un signal de type "carré". Pour cela, nous avons effectuées des mesures avec l'oscilloscope afin de déterminer la demi-période a recréer.

* delay µs
* temsp de calcul
* improvements avec registres