

Ecole Polytechnique de l'Université de Tours

64, Avenue Jean Portalis

37200 TOURS, FRANCE

Tél. +33(0)2 47 36 14 14

Fax. +33(0)2 47 36 14 22

[www.polytech.univ-tours.fr](http://www.polytech.univ-tours.fr/)

Rapport de projet de électronique

**Arrosage Intelligent**

**DemeoBox**

Apprentis: Tuteur:

Quentin Chalopin Alexis Rolland

[Quentin.chalopin@etu.univ-tours.fr](mailto:Quentin.chalopin@etu.univ-tours.fr) [alexis.rolland@univ-tours.fr](mailto:alexis.rolland@univ-tours.fr)

Kévin Repillez

Kevin.repillez@etu.univ-tours.fr

Sommaire

Introduction 4

Contexte 4

L’origine du projet 4

Cahier des charges 5

Adaptation de tension 5

Identification des modules 5

Communication sans fil 5

Données météorologique 5

Humidité terrestre 5

Température 5

Luminosité 5

Hygrométrie 5

Détection de pluie 5

Protocole de communication 6

Conception 7

Discrimination des modules 7

Choix du nombre de vannes pilotables 7

Schéma fonctionnel 8

Module Garage 8

Module météo 10

Module d’humidité terrestre 12

Choix des composants 13

Régulateur de tension 13

Microcontrôleur 13

Roue codeuse 13

Relais 14

Module sans fil 14

Capteur de luminosité 14

Capteur température et humidité 14

Batterie 15

Elaboration du projet 16

Fonction commune à tous les modules 16

FP1 traitement numérique 16

FP2 Communication sans fil 17

FP4 Identification module 17

FA2 Programmation microcontrôleur 17

Module garage 18

FA1 Adaptation de tension 18

FP3 Commande en puissance 18

Test 18

Module météo 19

FA1 Adaptation de tension 19

FP5 Captage de la température et de l’humidité 19

FP6 Captage de la luminosité 20

Développement et débogage 20

Gestion de projet 21

Annexes 23

# Introduction

Dans le cadre de notre formation par apprentissage à Polytech’Tours en Informatique Industrielle, nous devons réaliser un projet orienté « Développement embarqué ». Nous avions la possibilité de proposer un projet personnel ou en partenariat avec notre entreprise.

Nous avons donc pensé un projet dans le domaine de la domotique permettant la gestion de la programmation d’un arrosage automatique qui se dit « intelligent ». Ce projet s’inscrit sur deux tableaux, celui du projet « développement » mais aussi sur le projet « électronique ». Tous deux seront réalisés dans le cadre de la formation de Polytech’Tours.

# Contexte

D’une manière générale, la domotique est un sujet très à la mode de nos jours. De plus en plus de projets/produits sont commercialisés pour le grand public dans cette catégorie. La domotique peut regrouper énormément de domaines différents. Dans beaucoup de cas, la domotique est utilisée dans les foyers pour commander des lumières, volets ou tout autre objet qui nous entoure de près ou de loin. Par exemple, il serait envisageable de commander des interrupteurs permettant l’allumage/extinction de l’arrosage de la pelouse de sa maison. Et mieux encore, pouvoir programmer à l’avance la gestion de cet arrosage. En plus d’être automatisé mécaniquement et/ou électriquement, il serait possible de rajouter de « l’intelligence » au système afin d’obtenir un arrosage précis, optimisé et intelligent.

# L’origine du projet

Le projet de réaliser un arrosage intelligent et autonome est issu d’un besoin personnel. En effet, nous avions tous deux un système d’arrosage intégré à nos jardins avec une commande d’ouverture des vannes manuelles ou avec un petit programmateur. Nous nous sommes dit qu’il serait intéressant d’automatiser ces ouvertures via une application utilisable sur un ordinateur ou sur un Smartphone. De plus, nous voulions que cette application permette la gestion de l’arrosage en fonction des conditions météorologiques. En effet, nous avons remarqué que la plupart des programmateurs ne prennent pas en compte l’environnement extérieur et ne sont pas programmable à distance.

# Cahier des charges

## Adaptation de tension

Le système pourra piloter des vannes fonctionnant sous 24V au moins. Le branchement sur la carte devra être facile et accessible.

## Identification des modules

Le système permettra une sélection facile d’un identifiant par l’utilisateur. Cette identifiant sera utilisé dans l’application pour différentier les modules. Le soft embarqué sur la carte sera capable de reconnaître si la trame lui est destinée ou non.

## Communication sans fil

Pour l’instant, un module xBee est utilisé pour la communication sans fil couplé avec un shield raspberry pi « Cooking Hacks » ainsi que leurs librairies. Une autre technologie sans fil peut-être utilisée.

Les principaux atouts doivent être :

* Faible consommation ou capable de mise en veille
* Bonne distance de communication
* Protocole de communication facile à mettre en place (UART par exemple).

## Données météorologique

### Humidité terrestre

Pour un arrosage dit « intelligent », il est nécessaire d’avoir un retour sur le taux d’humidité terrestre.

### Température

Remontée de la température ambiante.

### Luminosité

Remontée de la luminosité.

### Hygrométrie

Remontée du taux d’humidité de l’air ambiant.

### Détection de pluie

Si possible, avoir une détection de pluie.

## Protocole de communication

Comprendre les trames envoyées par le serveur et envoyer la confirmation lorsque la modification a été appliquée.

Message de ce genre :

**<idModule>\_<idBroche>\_OK**

# Conception

## Discrimination des modules

Pour bien séparer chacune des fonctionnalités, nous avons décidé de séparer le système en trois modules distinctifs :

* Module garage
* Module météo
* Module pour humidité terrestre

Le module **garage** que l’on appelle aussi module de commande, sera le module qui contrôlera toutes les vannes. Ce module sera placé sous un toit avec une alimentation secteur.

Le module **météo** regroupera tous les capteurs pour l’acquisition des différentes données météorologiques. Ce module sera placé à l’extérieur et sera facilement transportable.

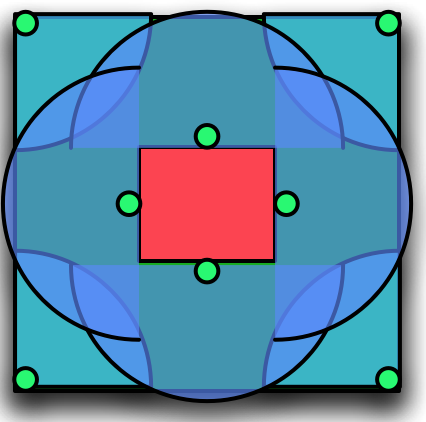
Le module pour **l’humidité terrestre** sera aussi mobile que le module météo. Nous avons choisi de le séparer pour apporter une mobilité supérieure. Ainsi, il est possible de relever l’humidité terrestre à des endroits précis.

## Choix du nombre de vannes pilotables

Ce nombre de vannes est primordial pour que ce système soit utilisable pour un utilisateur lambda. Pour connaître le nombre de circuits de commandes à installer sur notre module garage nous nous sommes fié à plusieurs facteurs :

* Superficie moyenne des terrains en France : 1000m2 avec maison de 100 m2
* Débit d’un petit circuit d’eau : 2.20 m3/h
* Limiter un seul arroseur par circuit d’eau : 1 vanne = 1 arroseur

Voici ci-dessous un schéma pour avoir un terrain totalement couvert avec les conditions citées :



En **rouge**, la superficie d’une maison.

En **vert,**  la position des arroseurs 180° et 90°.

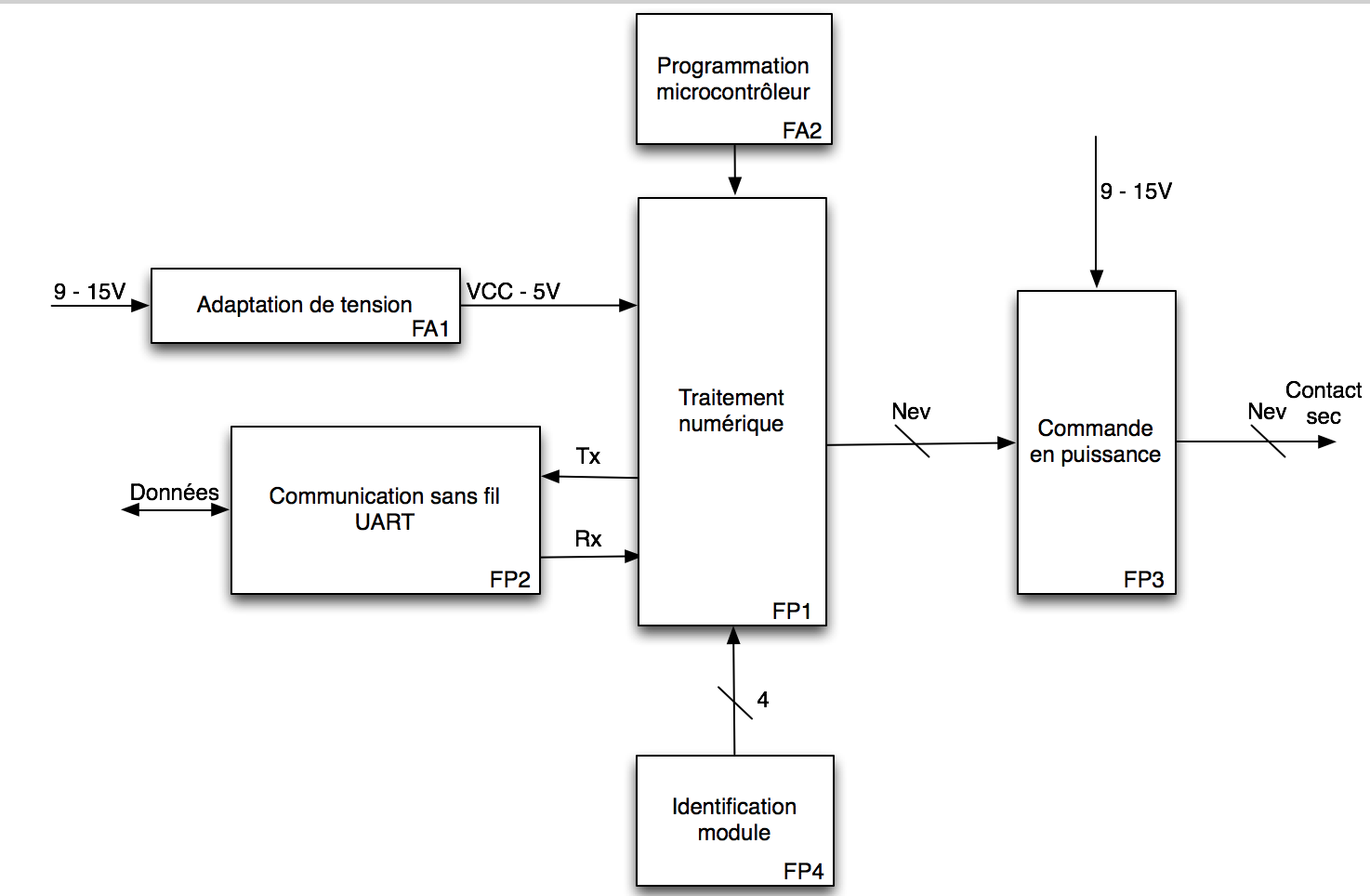
En **bleu,** la superficie couverte par les 8 arroseurs.

🡺 On peut voir qu’une superficie de taille moyenne est totalement couverte avec 8 arroseurs et donc 8 vannes.

🡺 Nous aurons donc 8 circuit de commandes sur notre module garage.

## Schéma fonctionnel

### Module Garage



|  |  |  |
| --- | --- | --- |
| Fonction | FA1 | Adaptation de tension |
| Description / Rôle | Il s’agit d’alimenter la carte entière. Cette alimentation doit fournir du **+5V** pour un courant de 200mA. Un régulateur à découpage LM2574-5 est au cœur de cette fonction . | |
| Signaux d’entrée | Une tension entre 9 et 15V | |
| Signaux de sortie | VCC | |
| Signaux E/S |  | |

|  |  |  |
| --- | --- | --- |
| Fonction | FA2 | Programmation µC |
| Description / Rôle | Cette fonction permet de flasher le µC mais aussi de le débuguer. Nous utilisons un PicKit v3. | |
| Signaux d’entrée |  | |
| Signaux de sortie | PGC/PGD/MCLR | |
| Signaux E/S |  | |

|  |  |  |
| --- | --- | --- |
| Fonction | FP1 | Traitement numérique |
| Description / Rôle | Fonction réalisant tous les traitements soft. Nous utilisons un PIC18F2520 pour cette fonction. | |
| Signaux d’entrée |  | |
| Signaux de sortie |  | |
| Signaux E/S | Toutes les broches des périphériques connectés | |

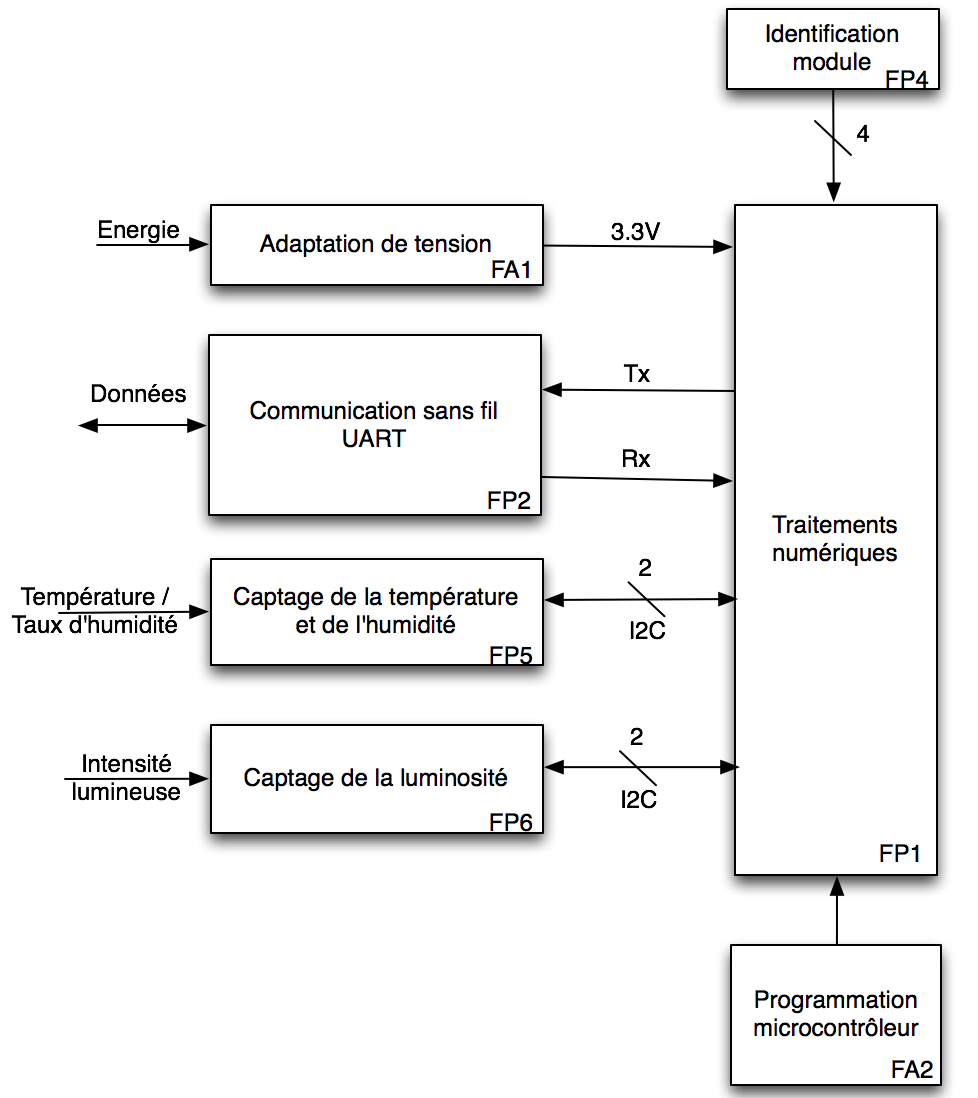
|  |  |  |
| --- | --- | --- |
| Fonction | FP2 | Communication sans fil UART |
| Description / Rôle | Fonction permettant de communiquer en sans fil. Nous utilisons ici un module xBee. | |
| Signaux d’entrée | Tx | |
| Signaux de sortie | Rx | |
| Signaux E/S |  | |

|  |  |  |
| --- | --- | --- |
| Fonction | FP3 | Commande en puissance |
| Description / Rôle | Fonction qui permet la commande des électrovannes. La fonction pourra piloter des électrovannes sous **220V** et **1A.** | |
| Signaux d’entrée | Broche du µC | |
| Signaux de sortie | Contacteur sec | |
| Signaux E/S |  | |

|  |  |  |
| --- | --- | --- |
| Fonction | FP4 | Identification module |
| Description / Rôle | Fonction qui donnera la possibilité à l’utilisateur de donner un identifiant à la carte. Une roue codeuse est au cœur de cette fonction. | |
| Signaux d’entrée |  | |
| Signaux de sortie | 4 sorties pondérées. | |
| Signaux E/S |  | |

### Module météo

|  |  |  |
| --- | --- | --- |
| Fonction | FA1 | Adaptation de tension |
| Description / Rôle | Il s’agit d’alimenter la carte entière. Cette alimentation doit fournir du **+5V** pour un courant de 200mA. Un régulateur à découpage LM2574-5 est au cœur de cette fonction . | |
| Signaux d’entrée | Une tension entre 9 et 15V | |
| Signaux de sortie | VCC | |
| Signaux E/S |  | |



|  |  |  |
| --- | --- | --- |
| Fonction | FA2 | Programmation µC |
| Description / Rôle | Cette fonction permet de flasher le µC mais aussi de le débuguer. Nous utilisons un PicKit v3. | |
| Signaux d’entrée |  | |
| Signaux de sortie | PGC/PGD/MCLR | |
| Signaux E/S |  | |

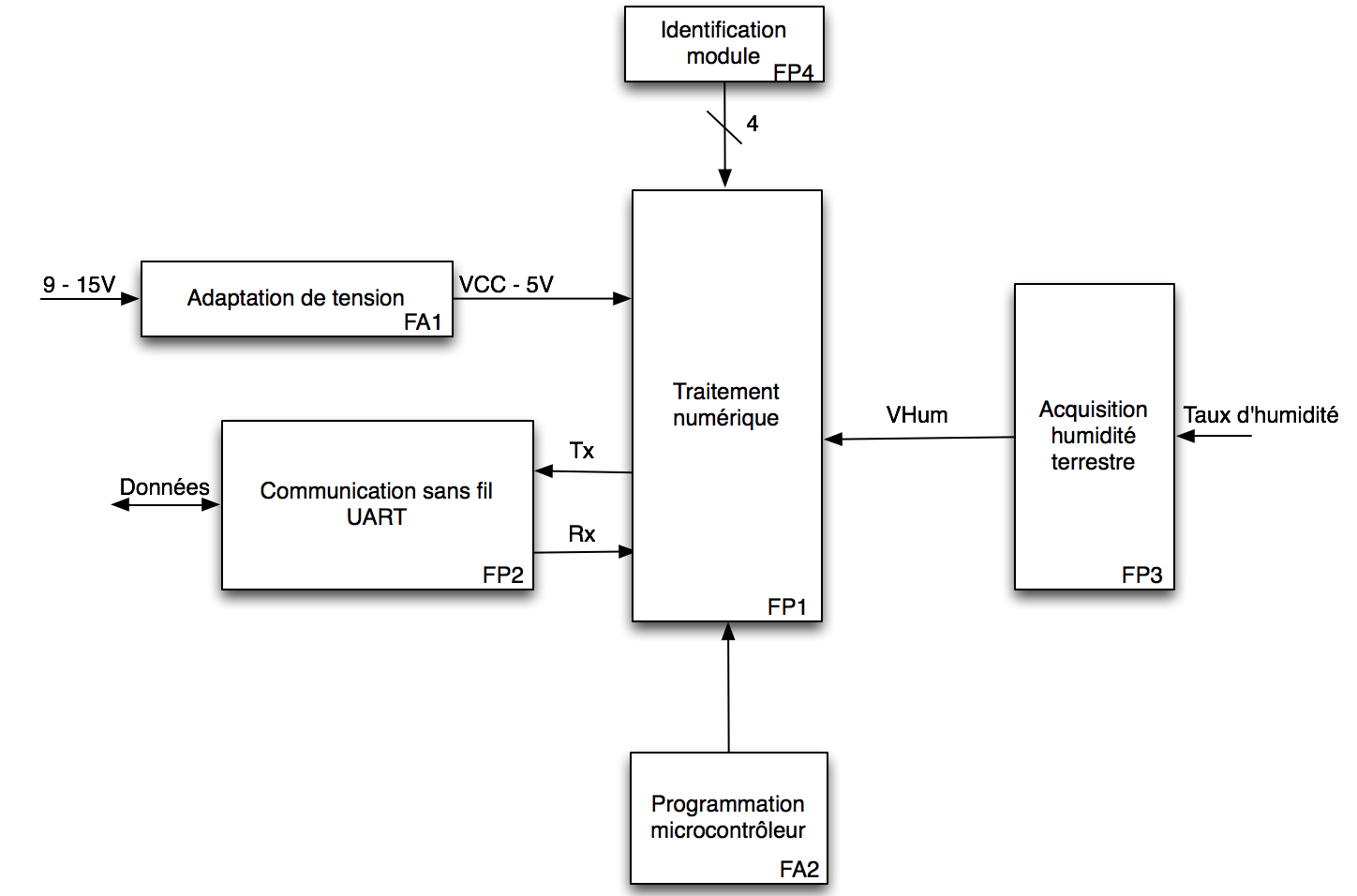
|  |  |  |
| --- | --- | --- |
| Fonction | FP1 | Traitement numérique |
| Description / Rôle | Fonction réalisant tous les traitements soft. Nous utilisons un PIC18F2520 pour cette fonction. | |
| Signaux d’entrée |  | |
| Signaux de sortie |  | |
| Signaux E/S | Toutes les broches des périphériques connectés | |

|  |  |  |
| --- | --- | --- |
| Fonction | FP2 | Communication sans fil UART |
| Description / Rôle | Fonction permettant de communiquer en sans fil. Nous utilisons ici un module xBee. | |
| Signaux d’entrée | Tx | |
| Signaux de sortie | Rx | |
| Signaux E/S |  | |

|  |  |  |
| --- | --- | --- |
| Fonction | FP5 | Captage de la température et humidité |
| Description / Rôle | Fonction qui permet d’acquérir la température et l’humidité de l’air ambiante. Un capteur HIH-6120-021-001 est au cœur de cette fonction. | |
| Signaux d’entrée | Température et le taux d’humidité | |
| Signaux de sortie | SDA / SCL | |
| Signaux E/S |  | |

|  |  |  |
| --- | --- | --- |
| Fonction | FP6 | Captage de la luminosité |
| Description / Rôle | Fonction qui donnera la luminosité actuelle. Un Capteur SEN10171N est au cœur de cette fonction. | |
| Signaux d’entrée |  | |
| Signaux de sortie | 4 sorties pondérées. | |
| Signaux E/S |  | |

### Module d’humidité terrestre



|  |  |  |
| --- | --- | --- |
| Fonction | FP3 | Acquisition humidité terrestre |
| Description / Rôle | Fonction qui permet de relevé le taux d’humidité terrestre. Cette fonction sera réalisée de façon « maison ». | |
| Signaux d’entrée | Taux d’humidité | |
| Signaux de sortie | Tension représentant le taux d’humidité | |
| Signaux E/S |  | |

## Choix des composants

### Régulateur de tension

L’alimentation de nos modules sera différente. En effet, le module garage sera alimenté sous 5V tandis que les modules météo et humidité terrestre seront en 3.3V.

Pour le module **garage**, nous avons dû dimensionner nos relais avant de pouvoir le choisir. Les relais Reed BE05-1A74-M consomme 15mA et sont au nombre de 8 soit 120 mA de consommation. Le régulateur LM2574 (500mA) était donc le meilleur choix.

Pour les deux autres modules, nous avons choisi un régulateur avec faible tension de chute qui permet de prendre une alimentation sous 5V. C’est pourquoi nous avons choisi le régulateur **MCP1825S-33**.

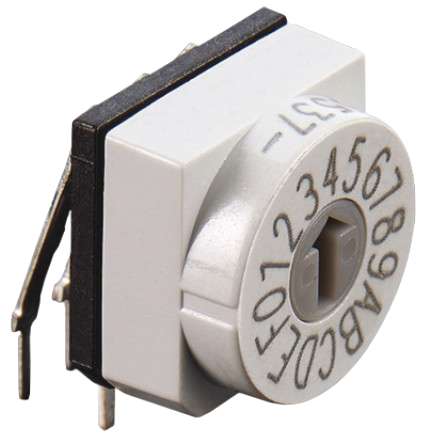
### Microcontrôleur

C’est le cœur de notre système. Après étude des périphériques utiles et du nombre de broches nécessaires, notre choix c’est porté vers un **PIC18F2520**. Ce microcontrôleur offre :

* Port I2C  🡪 Communication avec les différents capteurs (module météo)
* Port série 🡪 Communication sans-fil
* 4 broches 🡪 Roue codeuse
* 8 broches 🡪 Partie commande de puissance

### Roue codeuse

Pour l’identification des modules, il nous fallait un composant simple et accessible pour un utilisateur.



🡺 Nous avons décider d’utiliser une roue codeuse de ce type permettant de choisir un identifiant allant de 0 jusqu’à 9. Elle est également très simple à utiliser avec le microcontrôleur.

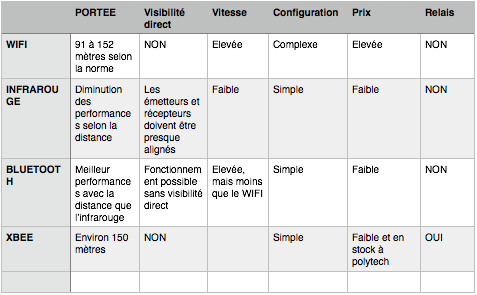
### Relais

Nous souhaitons réaliser un module garage qui peut commander toutes les électrovannes disponibles sur le marché. Pour cela nous devons choisir un relais capable de commuter avec une tension allant de 12V DC à 230V AC et de taille raisonnable.

🡺 Le seul relais correspondant à nos caractéristiques (site RS composant) est le relais REED BE05-1A74-M de la marque MEDER.

### Module sans fil

Voici un tableau comparatif des principales technologies sans-fil existantes :



Nous avons décidé d’utiliser les mêmes modules que lors du projet de développement embarqué.

### 

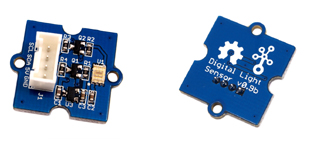
### Capteur de luminosité

Pour tous les capteurs du système, nous choisirons des capteurs numériques pour les avantages suivants :

* Plus simple à réaliser sur carte électronique
* Nous n’avons pas besoin d’une mesure extrêmement précise

Tout cela nous fera gagner du temps.

Pour le capteur de luminosité, nous avons besoin d’un capteur traversant communicant en I2C. Nous avons donc trouvé un capteur de luminosité I2C utilisé avec une carte Arduino.



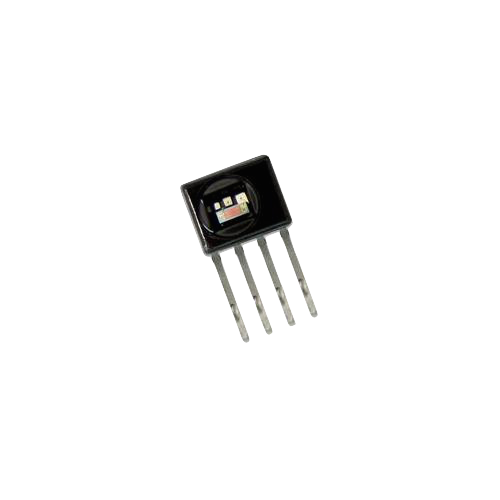
🡺 C’est le boîtier SEN10171P qui utilise le capteur TSL2561.

### Capteur température et humidité

Pour la température et l’humidité, nous avons décidé d’utiliser un capteur réunissant les deux informations. Ainsi, nous avons un gain de place sur la carte, de temps de conception et du développement soft.

Nous avons décidé d’utiliser des capteurs numériques pour les mêmes raisons qu’évoquées précédemment. En cherchant sur internet, nous avons trouvé deux capteurs :

* AM2302
* HIH-6120-021-001



🡺 En réunissant toutes les informations de chaque capteurs (Alimentation, Communication, Plage de fonctionnement,..).

Nous avons choisi le capteur HIH6120 car le AM2302 ne communique pas en I2C.

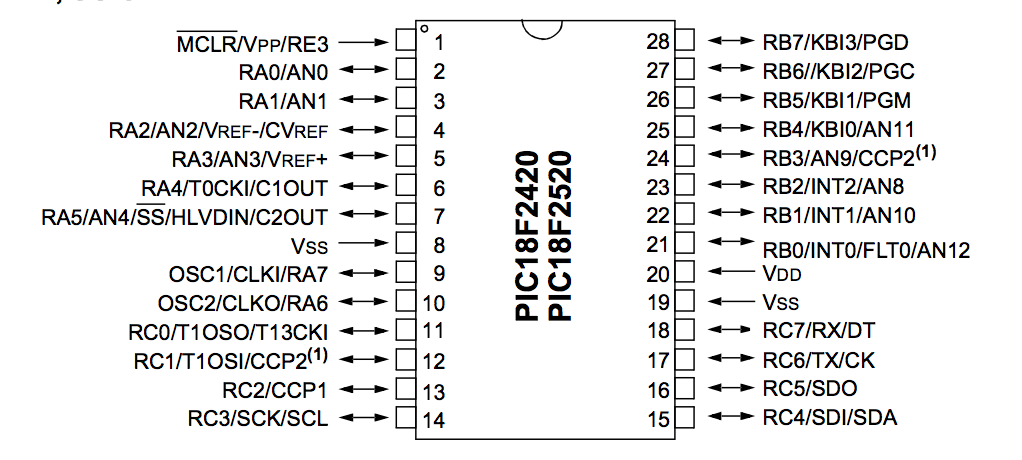
### Batterie

Pour le module **météo** et **humidité terrestre,** nous devions utiliser une pile classique. Seulement une pile n’est pas une solution très pratique. C’est pourquoi nous avons pensé à une batterie utilisée régulièrement pour la recharge des Smartphones. Nous en avons trouvé une pour 14€ sur RadioSpare. Sa capacité est de 2200mAh ce qui nous donne une autonomie vraiment meilleure qu’une simple pile.

# Elaboration du projet

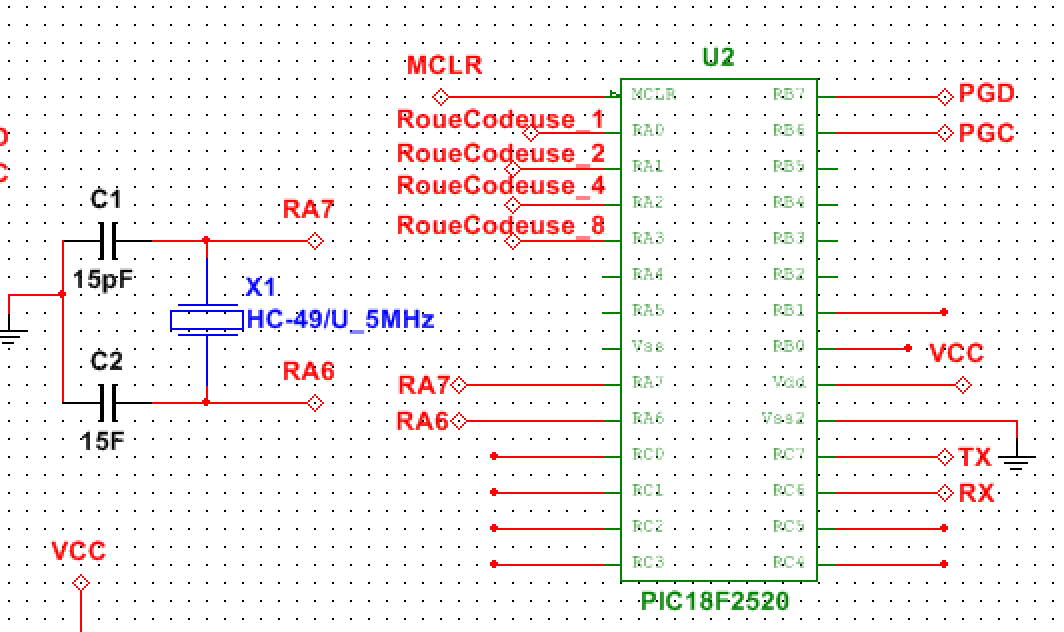
## Fonction commune à tous les modules

### FP1 traitement numérique



Pour la réalisation de cette fonction, nous avons ajouté un quartz de 8MHz au microcontrôleur avec deux condensateurs de 15pF, reliés à la masse. Nous avions calculé que le quartz nécessaire pour cette fonction serait de 5MHz, mais nous ne disposions que de quartz 8MHz à Polytech.

L’alimentation du microcontrôleur, la programmation, le reset, ainsi que la communication avec la fonction communication sans fil sont câblés sur les pins du microcontrôleur prévu à cet effet.

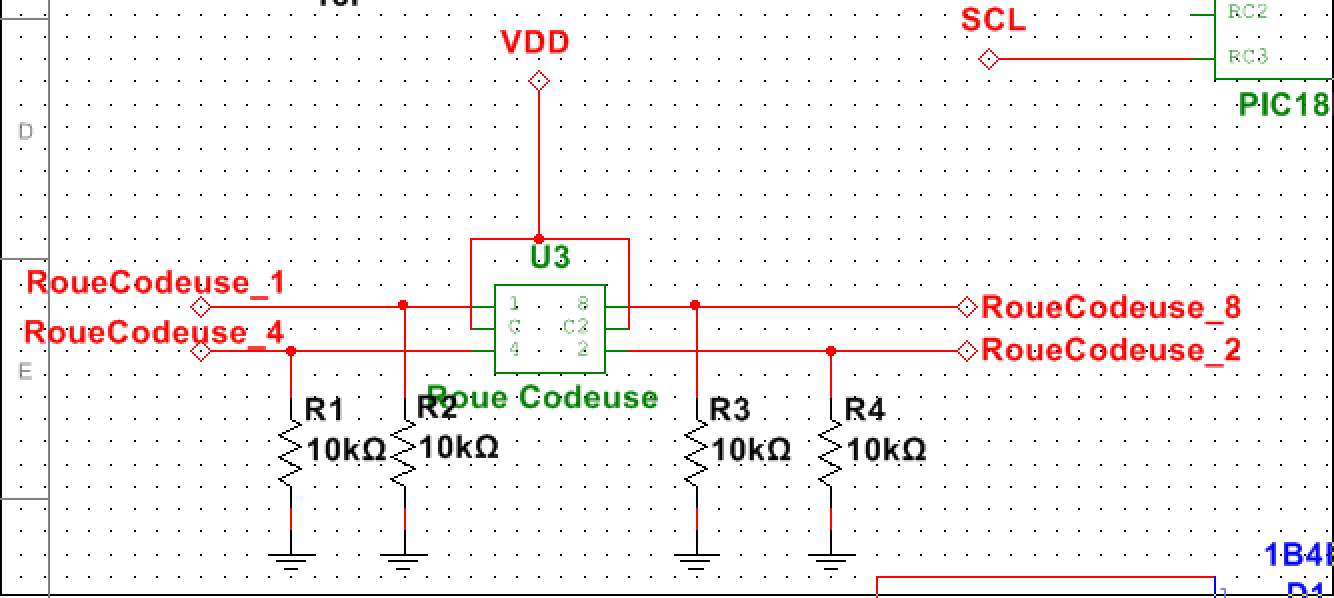
 La fonction identification module nécessite 4 pins du microcontrolleur, nous avons choisit les pins : RA0 à RA3.

### FP2 Communication sans fil

Nous avons placé les modules XBee sur des shields. Sur ces shields nous n’avions qu’à ajouter une source d’alimentation. 5V pour le module garage et 3.3V pour les modules météo. Et les broches de communication Tx-Rx raccordés au microcontrôleur.

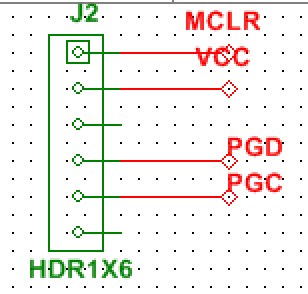
## 

### FP4 Identification module

 L’identification des modules ce fait grâce à des roues codeuses. Ces roues codeuses sont composées de 6 pins. 2 de ces pins sont reliés au VDD, les 4 autres sont reliés à la masse par le biais d’une résistance, ainsi qu’au microcontrôleur. Chacune de ces pins correspondent à une valeur, 1 2 4 ou 6. En fonction des pins active nous pouvons récupérer la valeur de la roue codeuse.

### FA2 Programmation microcontrôleur

La programmation du microcontrôleur se fait par un boitier, le pickit3. Pour communiquer avec ce boitier, les différents modules doivent être muni d’un connecteur à 6 pins. 2 pins reliés à l’alimentation, 1 pin relié au PGC, 1 pin relié au PGD, et enfin 1 pin relié au MCLR.

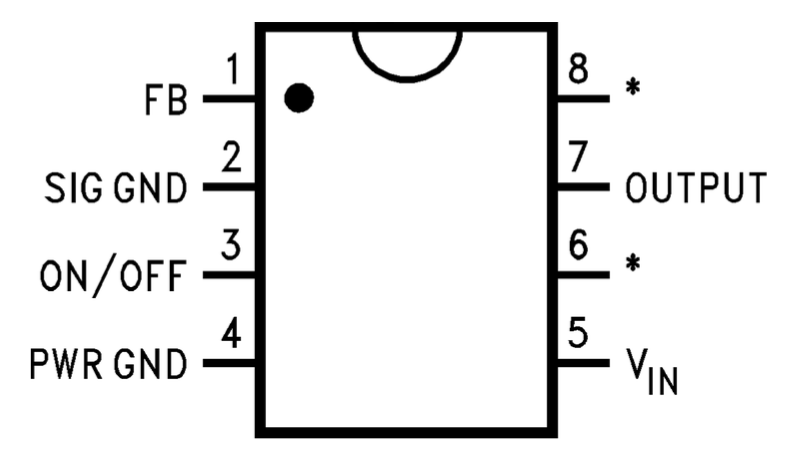


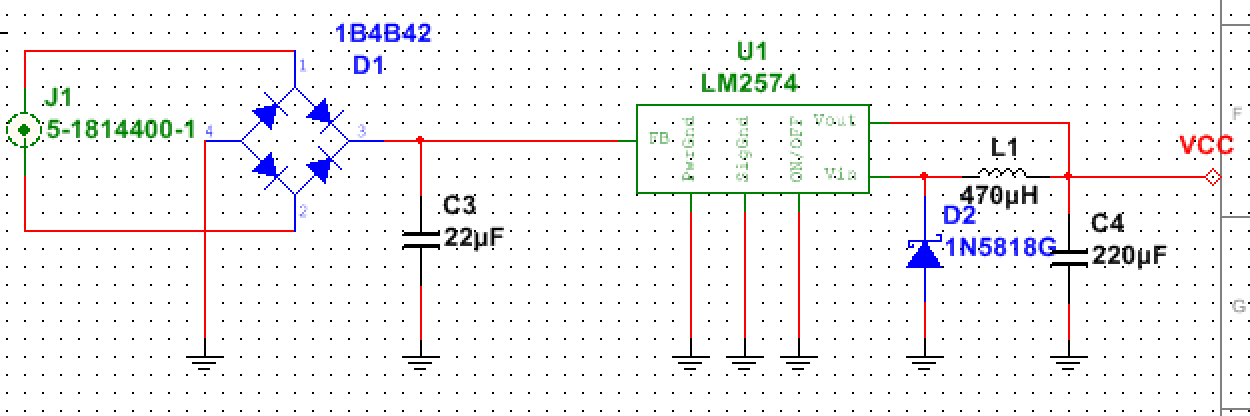
## Module garage

### FA1 Adaptation de tension

Nous avons câblé le régulateur de tension comme en suivant la documentation technique fournie avec le composant. Les pins Sig In, On/Off et Pwr GND sont relié à la masse. En sortie de la pin OUT, nous avons placé une diode shotcky et une self de 470µH. Le feedback du régulateur est relié à la sortie de la self. On retrouve ensuite un condensateur de 220µf connecté à la masse.

En amont du régulateur, nous avons placé un pont de diode, permettant de redresser le courant. Ainsi qu’un condensateur de 22µF pour le lisser.





### FP3 Commande en puissance

L’entrée de la bobine du relais est directement reliée à la sortie du régulateur de tension. La sortie de la bobine est quand à elle, relié au microcontrôleur. Pour que la bobine soit traversée par un courant, la sortie du microcontrôleur doit être à 0.

Entre l’entrer et la sortie du relais sont aussi relié par une diode de roue libre permettant à la bobine de ce décharger.

Le contact du relais laisse passer le courant nécessaire au fonctionnement des électrovannes lorsque la bobine est activée.

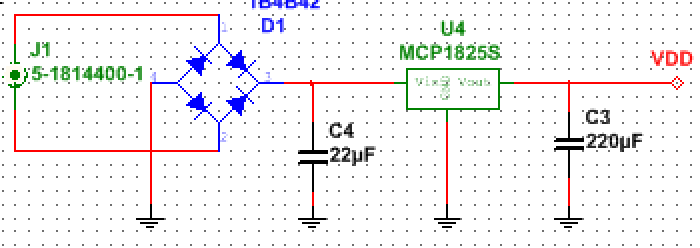
### Test

Pour la validation du module garage, nous avons procédé par étapes :

1. La tension de sortie du régulateur est de 5V avec peu d’ondulations sans charge
2. La tension de sortie à 5V avec microcontrôleur
3. Test des relais sans µC
4. Essai en flashant le µC avec un petit programme ON/OFF

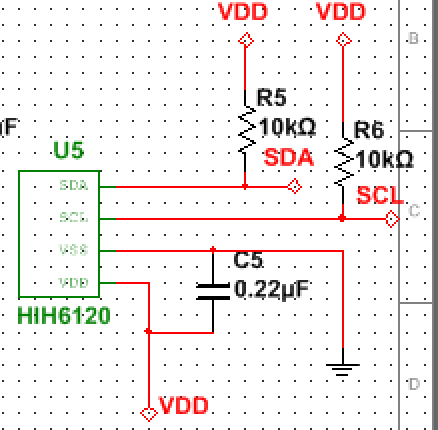
## Module météo

### FA1 Adaptation de tension



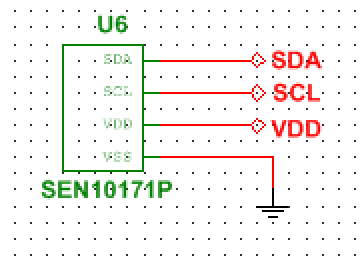
Pour la mise en place du régulateur de tension, nous avons installé un condensateur de 22µF en entrée, et un de 220µF en sortie.

### FP5 Captage de la température et de l’humidité



Pour les fonctions FP5 Captage de la température et de l’humidité et FP6 Captage de la luminosité. Nous avons câblé les capteurs comme indiqués sur la documentation technique.

### FP6 Captage de la luminosité



## Développement et débogage

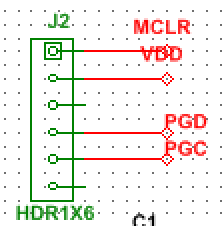
Pour le développement, le corps du programme est le même pour tous les modules. Il est composé de différentes fonctions :

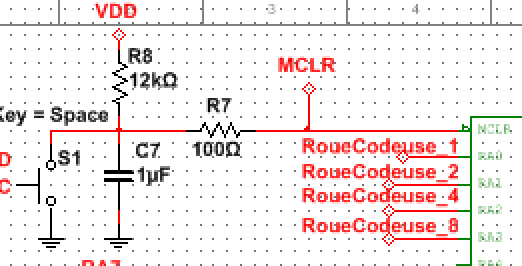
* Lecture de l’identifiant du module
* Lecture/Ecriture sur le port série
* Parsing du message reçu depuis le serveur

Pour le module **garage,**  nous avons une fonction supplémentaire permettant la commutation des différents relais.

Le module **météo** quant à lui, possède les fonctions nécessaires à la communication sur le bus i2C.

Lors de la conception du programme, nous avons utilisé l’IDE **MPLAB X** qui est très facile à configurer pour utiliser notre µC. De plus, le débogueur est très facile d’utilisation. Il nous a d’ailleurs beaucoup aidé tout au long du développement.

Pour flasher le microcontrôleur, nous avons câblé de la façon suivante :



J2 représente notre PICKIT v3. Nous avons également rajouté un bouton pour reset le microcontrôleur lorsque le système sera en mode **release** (principalement utile pour ce mode)**.**

# Gestion de projet

## 

Ci-dessus, notre planning prévisionnel ainsi que le réel. En **vert**, nous retrouvons les tâches réalisées dans les temps. En **orange,** celles qui ont pris plus de temps que ce qui était prévu mais qui n’est pas critique et enfin en **rouge**, les tâches qui ont pris beaucoup de retard.

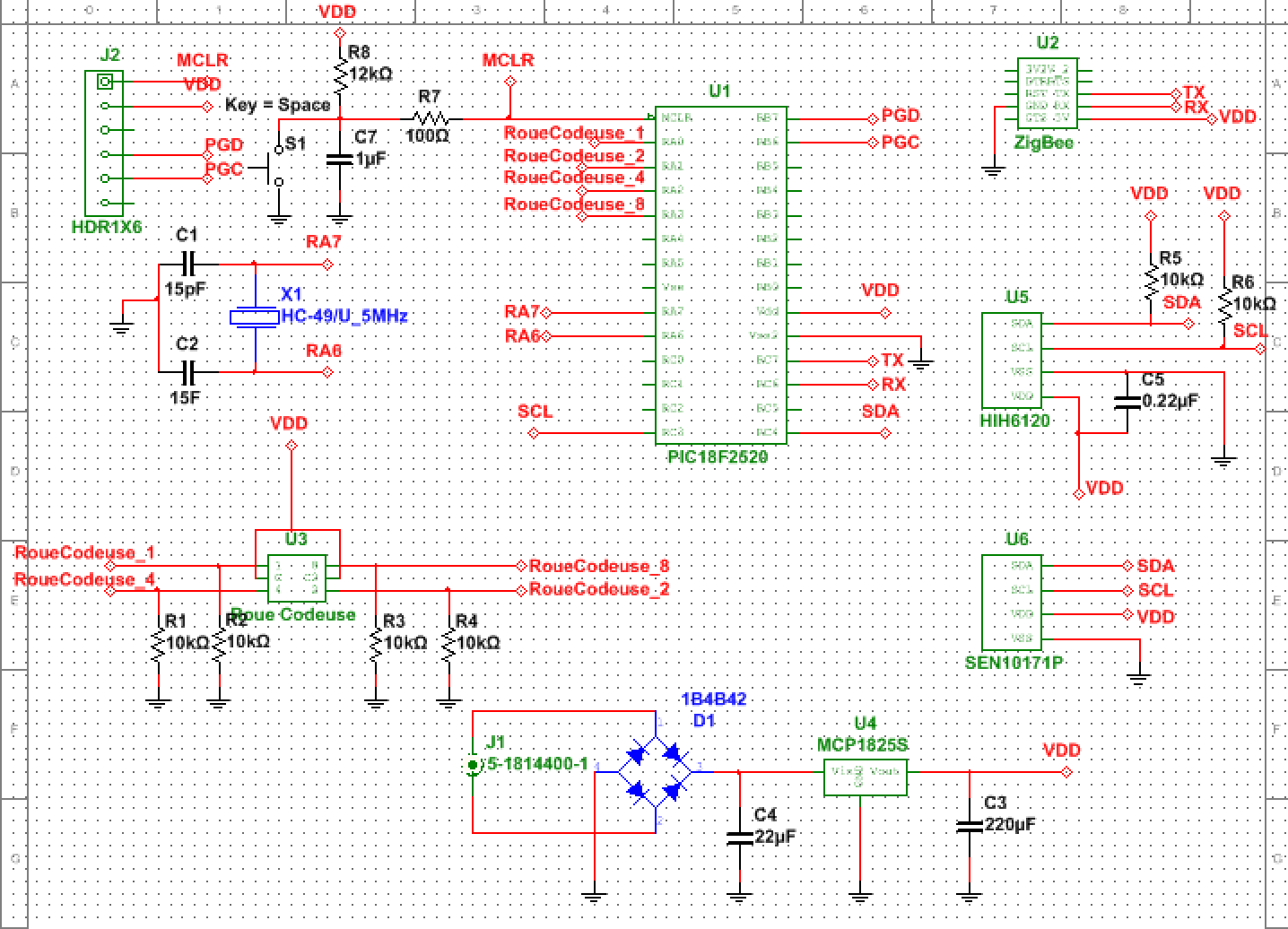
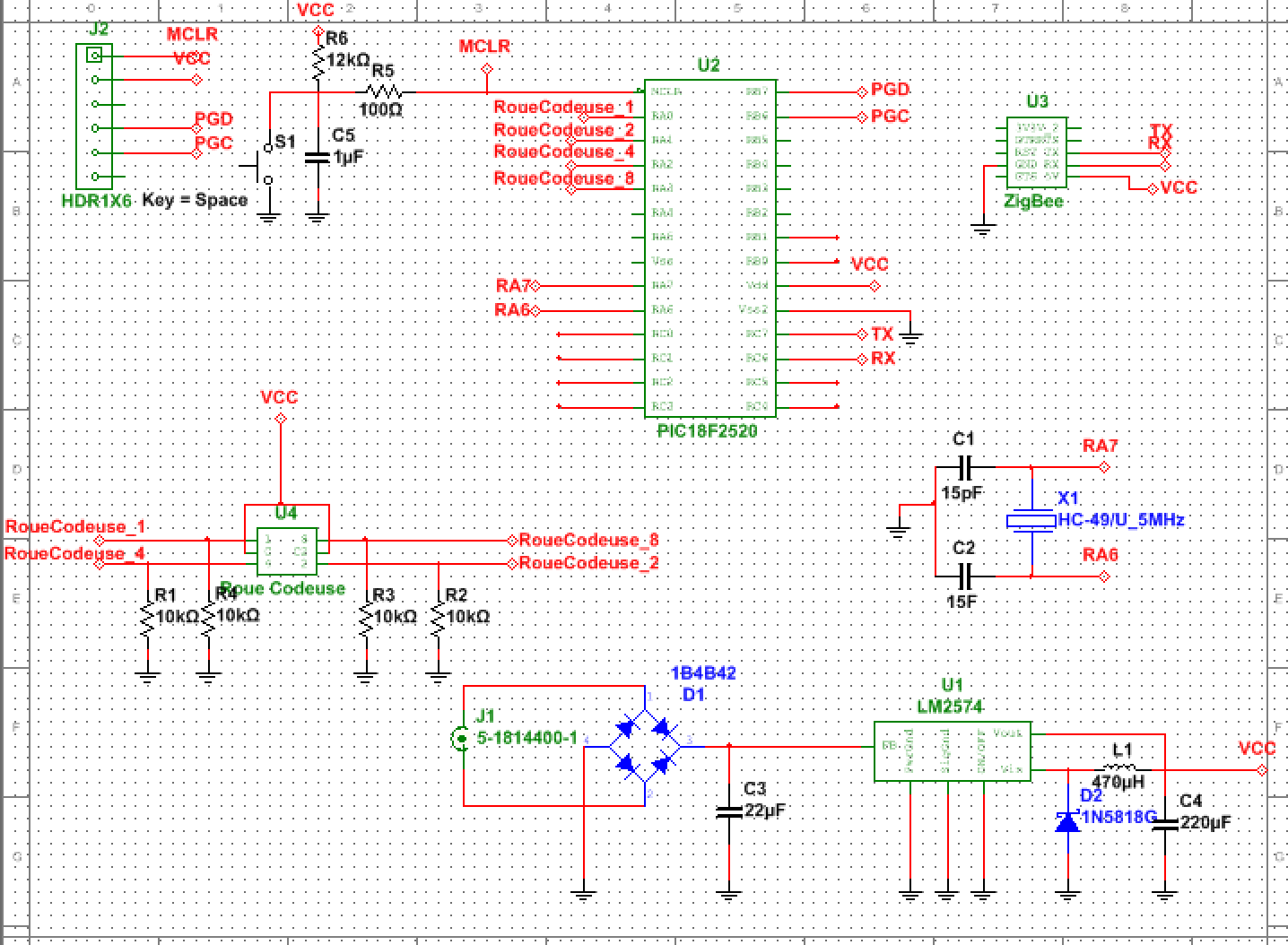
D’une manière générale, nos deadlines on été plutôt respectées. On peut s’apercevoir que le choix des composant nous a pris beaucoup plus de temps mais n’est pas pour autant un problème. Nous avons préféré faire le choix de composants en parallèle du schéma structurel. Cela nous a permis de faire moins d’erreur et ne pas reprendre notre schéma structurel à chaque changement de composants et/ou technologie.

En revanche, le développement sur microcontrôleur ne devait pas prendre autant de temps. Nous avons passé énormément de temps sur la configuration du PIC ainsi que sur ses périphériques.

# Annexes

Schéma structurel

Module garage

Module météo