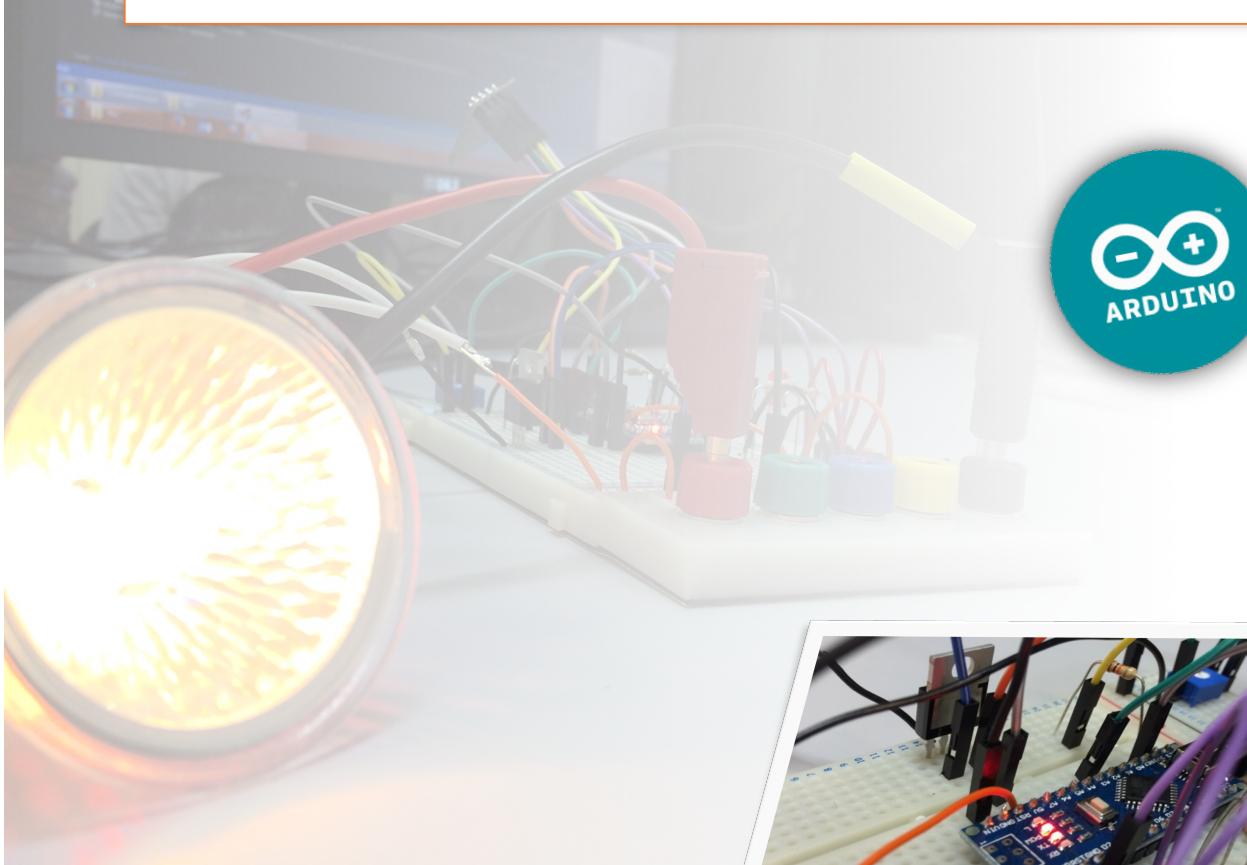


## Projet d'Etudes et Réalisations

Domotique : Gestion d'un système d'éclairage

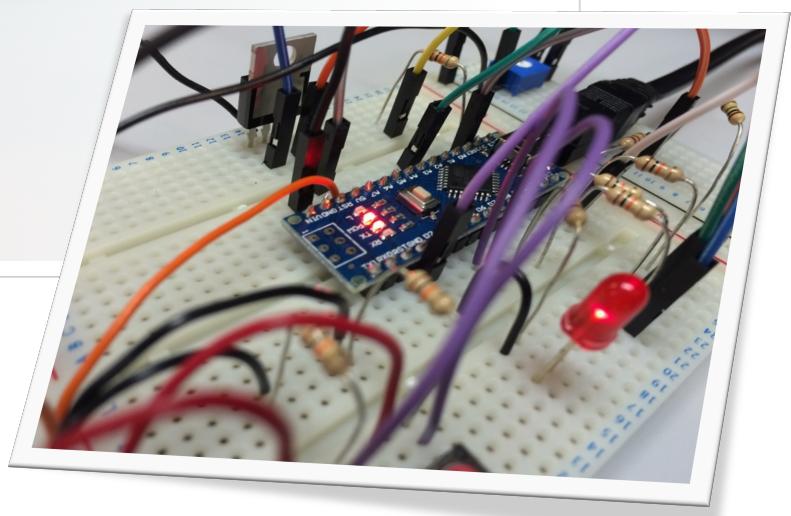


**GEii**  
Génie Electrique &  
Informatique Industrielle



Alan BELLO  
Maxime MARMONT

en collaboration avec  
Bruno MANILLIER  
Jean-Noël MARTIN



# Projet d'Etudes et Réalisations – Semestre 2

## Domotique : Gestion d'un système d'éclairage avec microcontrôleur Arduino

DUT1 Génie Electrique et Informatique Industrielle – IUT d'Annecy – Université Savoie Mont-Blanc

Un projet réalisé par Alan BELLO et Maxime MARMONT,  
en collaboration avec Bruno MANILLIER, Jean-Noël MARTIN et Thierry SUATON

## Sommaire

---

<b>1. Cahier des charges initial : connexion filaire entre les différentes parties .....</b>	<b>3</b>
1.1. Description et principe de fonctionnement.....	3
1.2. Gestion des Modes de Marche et d'Arrêt .....	5
1.3. Conception électrique et câblages .....	6
1.3.1. Entrées .....	6
1.3.1.1 Boutons poussoirs pour la sélection de modes.....	6
1.3.1.2 Capteur de lumière.....	7
1.3.1.3 Consigne .....	7
1.3.2. Sorties .....	7
1.3.2.1 LED de visualisation du mode actif.....	7
1.3.2.2 Commande de la lampe 50W (partie puissance) .....	7
1.4. Configuration du microcontrôleur Arduino.....	8
<b>2. Amélioration du prototype : communication sans fil entre le capteur et le système de commande.....</b>	<b>9</b>
2.1. Description avec prise en charge de la communication sans fil .....	9
2.2. Ajout des modules radio.....	9
2.3. Configuration des microcontrôleurs Arduino .....	9
2.3.1. Logiciel du nœud émetteur (partie capteur) .....	9
2.3.2. Logiciel du nœud récepteur (partie opérative).....	10
2.4. Rendu final.....	10
<b>3. Bilan critique .....</b>	<b>11</b>
<b>4. Annexes .....</b>	<b>12</b>
4.1. Diagramme fonctionnel (avec communication sans fil).....	12
4.2. Gestion de projet.....	13
4.2.1. Diagramme de Gantt .....	13
4.2.2. Nomenclature.....	13
4.2.3. Devis du prototype (coûts horaires et prix du matériel).....	14

The subject of our project is Automation of light. The aim of this project is to manage the lighting according to the brightness. When the brightness is low the light of lamps increases and if the brightness is high the light of lamps decreases automatically. But there is also a manual mode to manage the lighting with push-button. This project is very interesting for our everyday use at home as an example because you can save some energy or if you want a soft light on the evening.

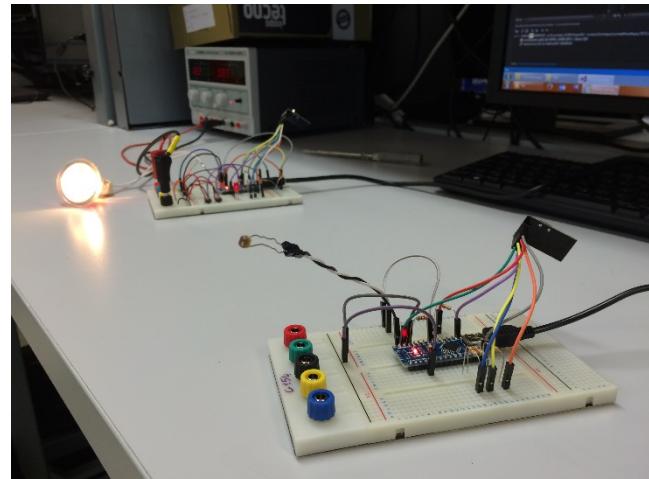
We will begin with the description of specifications. Secondly we will present you the analog part. Then we will explain the software configuration. We will conclude with a Critical balance sheet of project.

## 1. Cahier des charges initial : connexion filaire entre les différentes parties

### 1.1. Description et principe de fonctionnement

L'objectif du projet « Lumière Domotique » réside dans la capacité du prototype à gérer de manière autonome un système d'éclairage présent dans une pièce d'un bâtiment. Celui-ci sera représenté par une lampe halogène 50W dans notre prototype. Le système d'éclairage doit pouvoir être utilisé selon deux modes précisés ci-après : premièrement un mode automatique et deuxièmement un mode manuel. Le switch entre les deux modes sera géré par un simple appui sur un bouton poussoir nommé « **Mode** » (ou S2).

En mode **automatique** (F1), le système doit pouvoir se réguler en fonction de deux paramètres : tout d'abord une consigne qui va indiquer l'intensité lumineuse désirée dans la pièce en question, et ensuite la luminosité ambiante présente dans la pièce. Le système chargera la consigne demandée, vérifiera si elle est atteinte par la luminosité ambiante, et dans le cas où celle-ci est insuffisante l'intensité lumineuse doit être compensée par l'allumage plus ou moins fort de l'éclairage jusqu'à atteindre la consigne. Dans notre cas, on considérera que si lumière émise par notre lampe halogène est inférieure à 20%, on préférera éteindre la lampe puisque l'intensité lumineuse fournie par celle-ci en dessous de ce seuil est de faible utilité (lampe quasiment éteinte) et consommera de l'énergie inutilement. L'appui sur un bouton « **Option** » (ou S1) activera/désactivera l'allumage de la lampe.



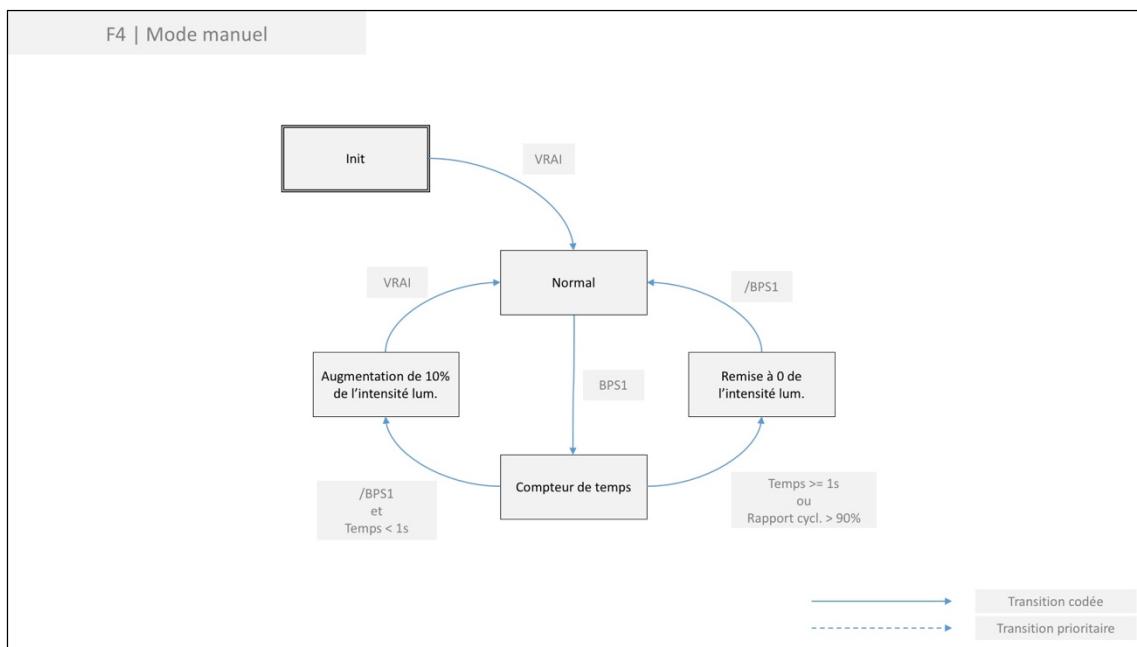
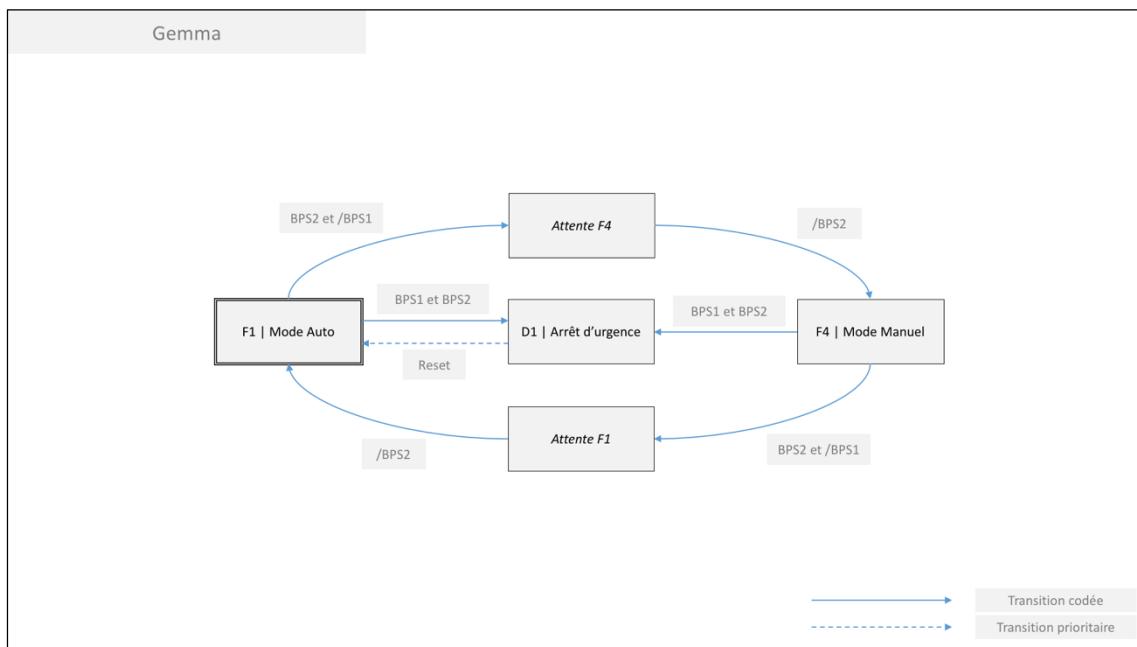
En mode **manuel** (F4), on doit pouvoir augmenter l'intensité lumineuse générée par notre lampe de 10% à chaque appui inférieur à 1 seconde sur le bouton « **Option** » (S1). Si l'intensité lumineuse est strictement supérieure à 90% ou si l'appui sur « **Option** » (S1) est supérieur ou égal à 1 seconde, la lampe doit s'éteindre (intensité lumineuse à 0%).

On ajoutera en parallèle de ces deux modes un mode « Arrêt d'urgence » qui éteindra immédiatement le système d'éclairage lors de l'appui simultané sur « Mode » (S2) et « Option » (S1). La remise en marche du système ne devra se faire que lors de la réinitialisation du microcontrôleur (gérée par matériel : appui sur le bouton « Reset » ou mise hors-tension de la carte).

De plus, une LED devra permettre de visualiser le mode dans lequel se situe le programme : clignotement à une fréquence de 1Hz pour le mode auto et allumée en permanence pour le mode manuel. Enfin, les microcontrôleurs utilisés seront des cartes Arduino Nano (voir en annexe 4.1 le schéma fonctionnel du prototype).

## 1.2. Gestion des Modes de Marche et d'Arrêt

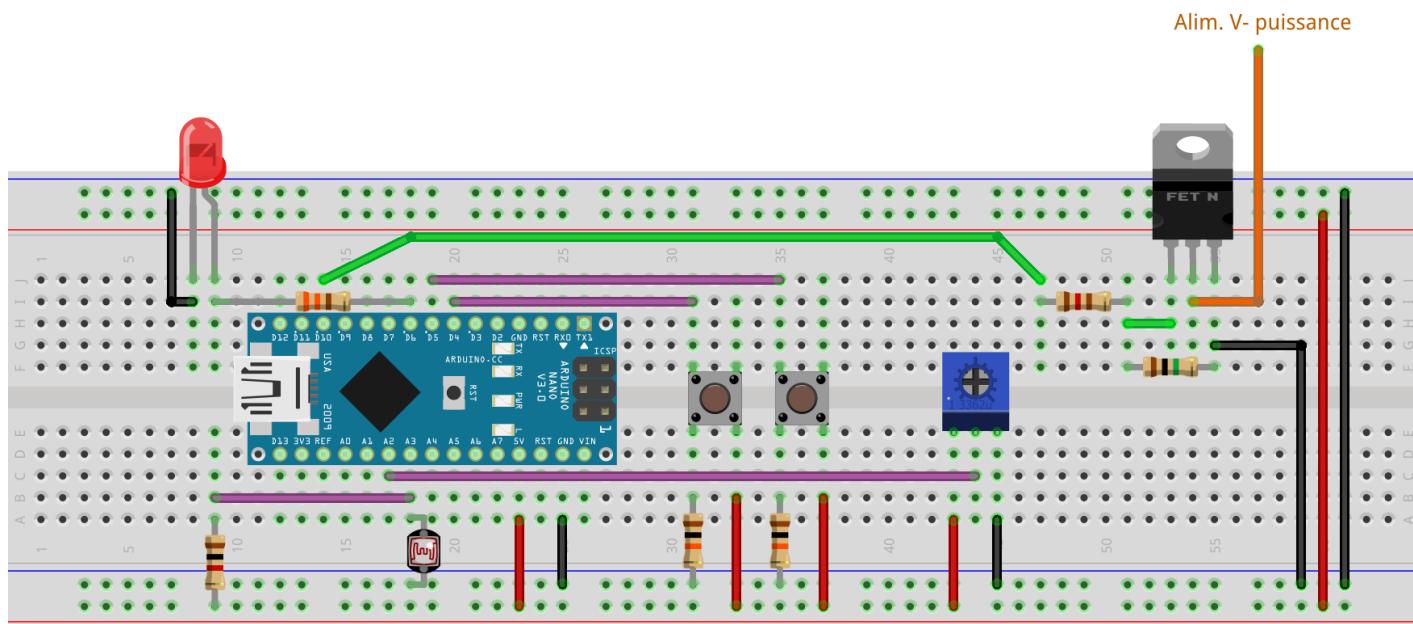
Sachant que le basculement entre les modes (manuel et auto) se réalise de manière séquentielle, et de même pour le fonctionnement en mode manuel, on choisit d'utiliser un GEMMA (Gestion des Modes de Marche et d'Arrêt) simple qui va gérer l'ensemble du fonctionnement du système domotique. Sachant que le mode auto est purement combinatoire, il ne nécessite pas d'être géré par une machine d'état. Ci-dessous la structure GEMMA appliquée à notre système :



### 1.3. Conception électrique et câblages

Les microcontrôleurs mis à notre disposition fonctionnent suivant la norme TTL (0-5V) et peuvent être alimentés directement en 5Vdc (régulé) sur leur borne **5V**. Sinon, on peut les alimenter soit par USB, soit via une alimentation délivrant une tension continu comprise entre 7 et 12V appliquée sur la borne **Vin**. Ci-après sont décrites toutes les parties électroniques reliées en entrée et en sortie de notre carte Arduino.

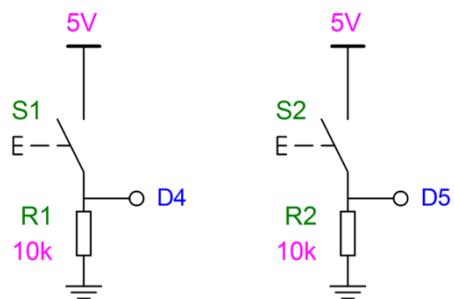
Sur les schémas, en vert est indiquée la nomenclature du composant, en fuchsia la valeur du composant ou de la tension appliquée et en bleu la borne de connexion à l'Arduino.



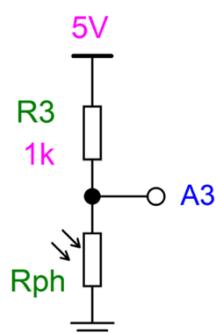
#### 1.3.1. Entrées

##### 1.3.1.1 Boutons poussoirs pour la sélection de modes

Les boutons poussoirs, qui trouvent leur utilité dans la sélection de mode ou dans la modification d'une option, sont câblés en *actif haut*, c'est-à-dire qu'ils renvoient un 1 logique lors de l'appui ou un 0 logique au repos. Afin de limiter au mieux la consommation d'énergie, les résistances choisies sont de l'ordre du kilo-ohm.



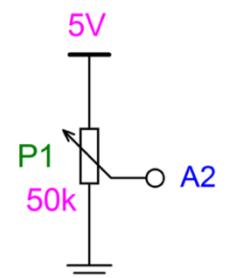
### 1.3.1.2 Capteur de lumière



Bien que cela n'ai pas de réelle importance étant donné qu'un programme informatique gère ce capteur, celui-ci a été câblé de la même manière qu'on l'aurait fait en électronique classique (avec un AOP en mode comparateur par exemple), c'est-à-dire avec la photorésistance entre le point de tension et la masse. Etant donné que notre interface entre la physique et l'électricité est réalisée via une résistance dont la valeur varie inversement proportionnellement à l'intensité lumineuse ambiante, on réalise un simple pont diviseur.

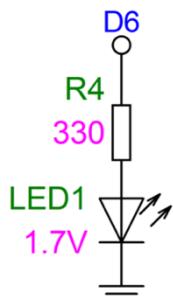
### 1.3.1.3 Consigne

Pour réaliser une consigne allant de la valeur minimum (0V) à la valeur maximum (5V), rien de plus simple, on place un potentiomètre (si possible d'une grande valeur pour avoir un courant très faible) entre V+ et GND.



## 1.3.2. Sorties

### 1.3.2.1 LED de visualisation du mode actif

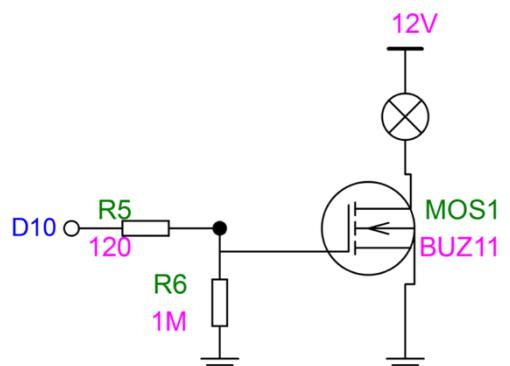


Pour la LED de visualisation du mode actif, on dispose d'une LED rouge dont la tension de seuil est égale à 1,7V à 10mA. On fait une loi des mailles pour trouver la valeur de la résistance à placer en série avec la diode pour limiter le courant :

$$5 - 1.7 - R * 0.01 = 0 \Leftrightarrow R = \frac{5 - 1.7}{0.01} = 330\Omega$$

### 1.3.2.2 Commande de la lampe 50W (partie puissance)

En ce qui concerne la lampe 12V-50W, cela se complique un peu. En effet, la carte Arduino ne pouvant délivrer une tension de 12V et encore moins un courant de plusieurs ampères, il faut créer une maille de puissance. Cette maille est réalisée via un transistor, une sorte d'interrupteur commandé qui peut faire circuler un fort courant lors de la présence d'un faible courant (ou faible tension) sur sa base (ou gate). On choisira nous un MOS-N, un transistor commandé par une tension, ce qui est idéal pour notre cas puisque l'Arduino ne devra délivrer qu'un très faible courant. On ajoute des résistances de sécurité, une de faible valeur en sortie de l'Arduino et une de très grande valeur entre la base du transistor et la masse (pour éviter des dommages liés à un éventuel court-circuit). Le transistor que l'on a pu obtenir est un

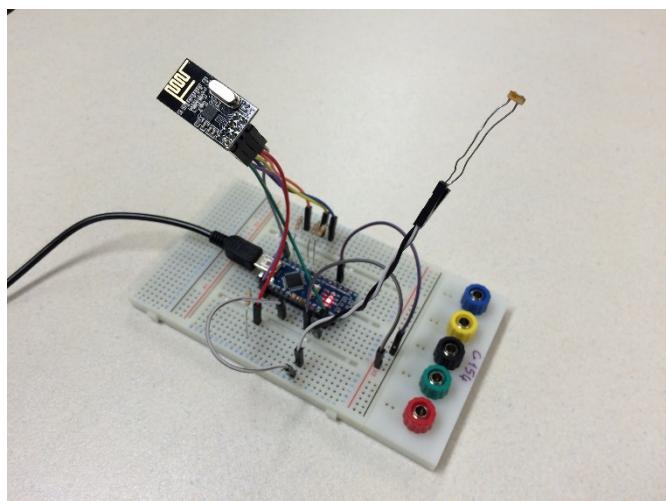


BUZ11, qui peut supporter les 4A que demandent la lampe mais qui chauffe fort très rapidement. Un radiateur serait une bonne solution dans le cas d'une utilisation prolongée du dispositif.

## 1.4. Configuration du microcontrôleur Arduino

Le code fourni lors de la soutenance technique peut se résumer en quelques lignes mais nous ne donnerons pas la liste complète des fonctions C/C++ étant donné le grand nombre que l'on a créées et/ou utilisées. Les machines d'état sont gérées via des fonctions switch().

Un microcontrôleur Arduino peut accepter sur certaines entrées des signaux analogiques et sur d'autres des signaux numériques. Or ce n'est pas tout à fait le cas pour les sorties. Toutes ne peuvent délivrer que 0 ou 5V exclusivement. Cependant, il existe une méthode pour simuler un signal « analogique » (c'est-à-dire variable sur une grande plage de valeur). Cette méthode utilise les propriétés de la valeur moyenne, vous l'aurez donc compris nous utiliserons des signaux dont on peut faire varier le rapport cyclique (0% pour éteint, et entre 20 et 100% pour allumer la lampe).



Pour pouvoir faire fonctionner un tel mécanisme, il faut d'une part utiliser une sortie compatible PWM, ainsi qu'un Timer approprié (nous avons choisi le Timer1 dans notre cas, un Timer 8-bit). Une fois le Timer correctement configuré, il suffit d'insérer dans le registre approprié la valeur du rapport cyclique voulue (0 pour 0% et 255 pour 100%, les valeurs intermédiaires étant proportionnelles).

En ce qui concerne le capteur, on utilisera le Convertisseur Analogique-Numérique (CAN) intégré au microcontrôleur qui va nous renvoyer un nombre entre 0 et 1023 proportionnel à la tension  $U_{Rph}$ , qui est comprise entre 0 (luminosité forte) et 5V (luminosité faible). Pour lire la valeur de la consigne, il suffit simplement de changer la voie d'acquisition via le registre ADMUX, et ainsi de suite.

Pour faire clignoter la LED d'état (lorsque le mode auto est activé), on utilise une interruption sur débordement du Timer2. On configure le Timer pour que son horloge soit une division de 1/1024 de l'horloge 16MHz (ce qui donne une période de 64μs) et le registre TIMSK2 pour activer l'interruption sur le drapeau TOV2. Sachant que l'on veut une période de 1Hz pour le clignotement, il faut compter 61 débordements pour avoir une demi-période. Ce comptage sera effectué dans la fonction d'interruption ISR et lorsque les 61 débordements sont atteints, on change l'état de la LED.

(→ Se référer au code ainsi qu'aux commentaires fournis pour de plus amples informations sur le logiciel développé sur ce microcontrôleur)

## 2. Amélioration du prototype : communication sans fil entre le capteur et le système de commande

### 2.1. Description avec prise en charge de la communication sans fil

Le cahier des charges initial reste le même, à la différence que cette fois-ci nous introduirons une communication sans fil entre le capteur et le système d'éclairage. On déportera seulement le capteur de lumière sur une seconde carte Arduino Nano et on ajoutera un module radio (RF) sur chacune des cartes.

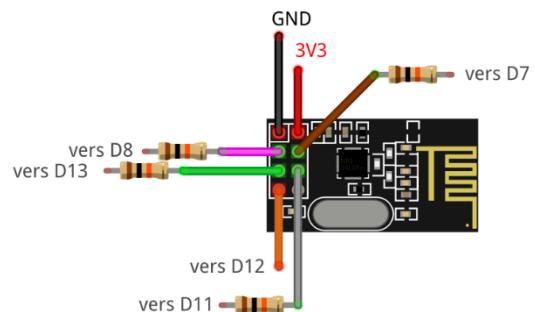
On choisit de déporter le capteur de la partie commande puisque dans le cas où la connexion entre les deux cartes serait perdue ou interrompue, il faut pouvoir garder le contrôle sur la lampe quoi qu'il en soit, quitte à passer en mode manuel.

Les deux cartes devront chacune transmettre des informations de télémétrie concernant la liaison sans fil sur la liaison série USB pour permettre un diagnostic en cas de dysfonctionnement.

### 2.2. Ajout des modules radio

L'ajout des modules radio sur les cartes est relativement simple, il suffit de suivre le brochage de la documentation technique associée aux modules. Le câblage est le suivant et est le même pour les deux cartes.

Les modules utilisés sont des nRF24L01. Une fois configurés correctement par logiciel, les modules RF établissent une liaison série entre eux pour pouvoir communiquer. Le câblage nécessaire au bon fonctionnement de la liaison série nous a été fourni dans une fiche technique.



### 2.3. Configuration des microcontrôleurs Arduino

Contrairement au cahier des charges initial, il va falloir créer deux logiciels Arduino : un pour la *partie opérative* et un autre pour la *partie capteur*. La première partie héritera du logiciel créé précédemment auquel on ajoutera la fonctionnalité sans fil (en tant que serveur) et la seconde aura son logiciel propre qui n'aura pour fonctionnalité que de lire la valeur du capteur de lumière et de renvoyer cette valeur via la liaison sans fil (en tant que client) sur demande du serveur.

#### 2.3.1. Logiciel du noeud émetteur (partie capteur)

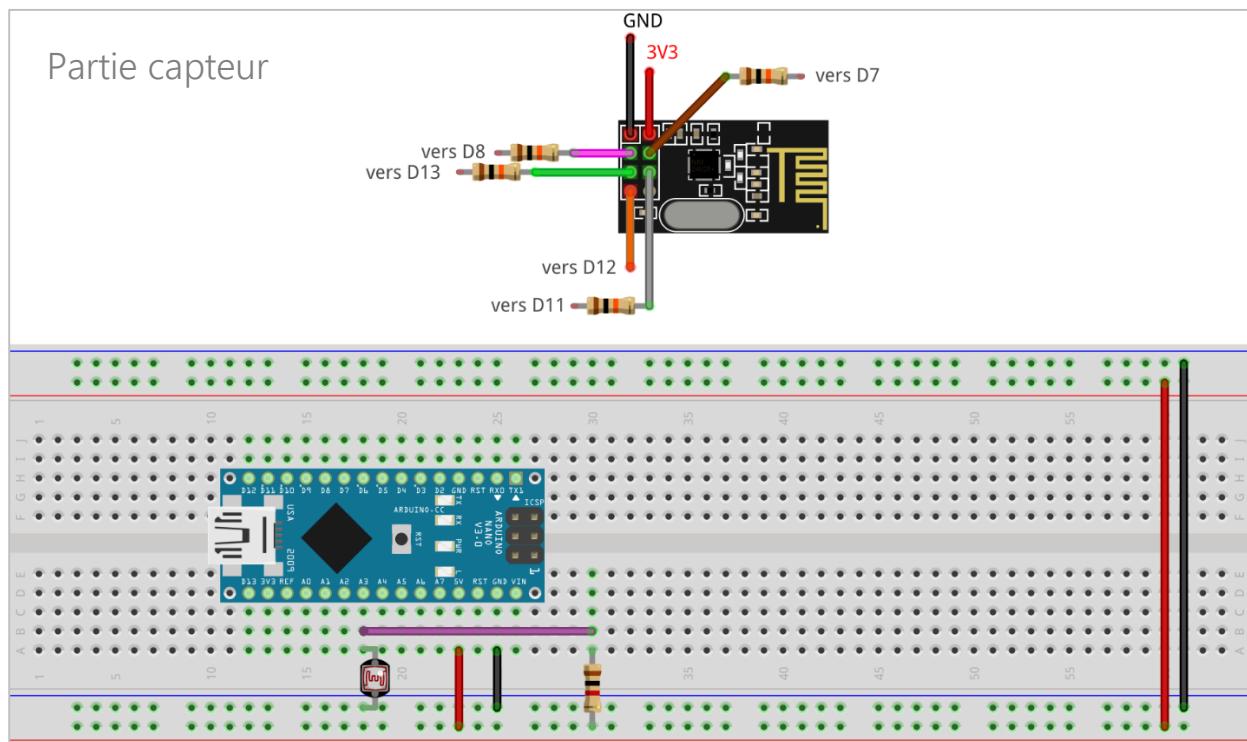
Comme dit précédemment, le capteur n'enverra la valeur du capteur au serveur uniquement lorsque ce dernier l'aura interrogé. Finalement, le logiciel de la *partie capteur* intégrera comme fonctionnalités la conversion analogique-numérique (CAN) et la communication sans fil (client).

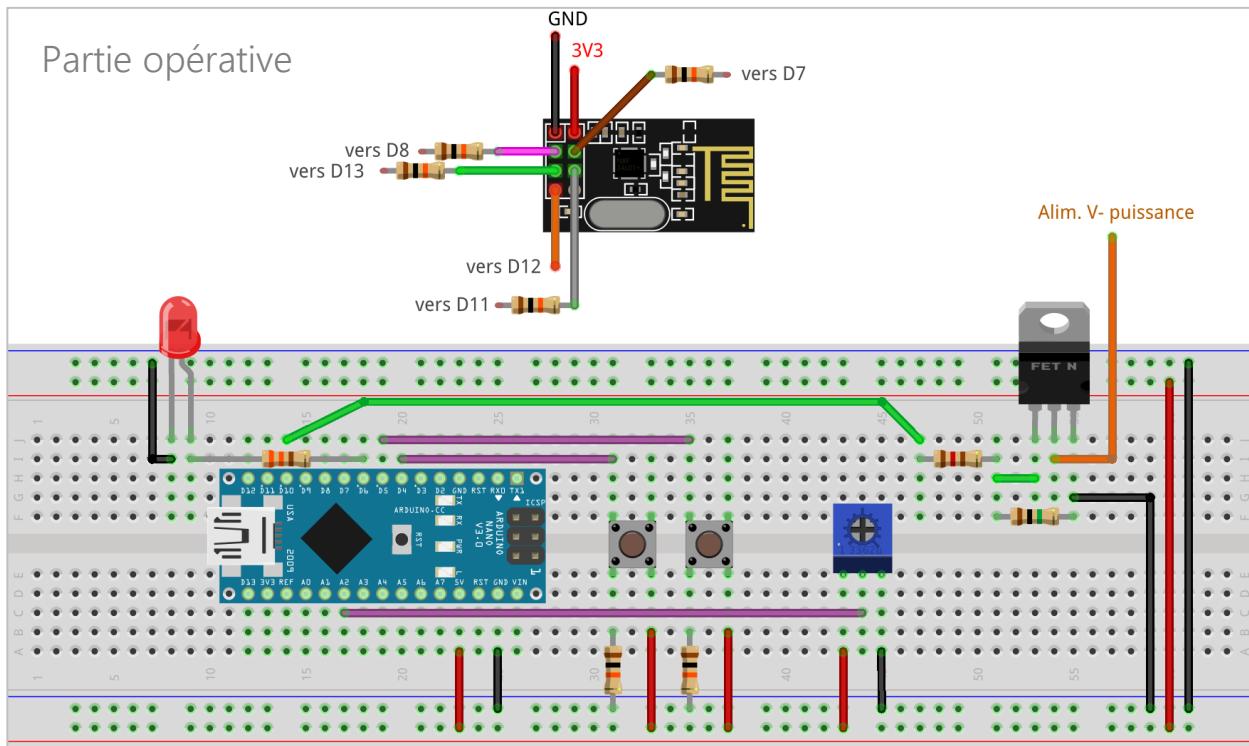
Lorsque le capteur ne reçoit aucun ordre ou si l'ordre n'est pas reconnu, le contrôleur n'envoie aucune donnée.

### 2.3.2. Logiciel du nœud récepteur (partie opérative)

Le logiciel du serveur ne subira finalement que peu de changement. Lors de la lecture des entrées au début du Gemma, la valeur de l'intensité lumineuse ambiante ne sera plus récupérée dans le CAN mais dans les données reçues par le module sans fil. Pour récupérer cette valeur, le serveur va interroger le client en envoyant une requête qui demandera la valeur du capteur, puis va attendre de recevoir les données pour les analyser et enfin retourner la valeur attendue au Gemma.

## 2.4. Rendu final





### 3. Bilan critique

Everything works in our project, we have the light which is managed according to the ambient brightness in automatically mode and we have a manual mode for managed the light of lamp with a push bouton.

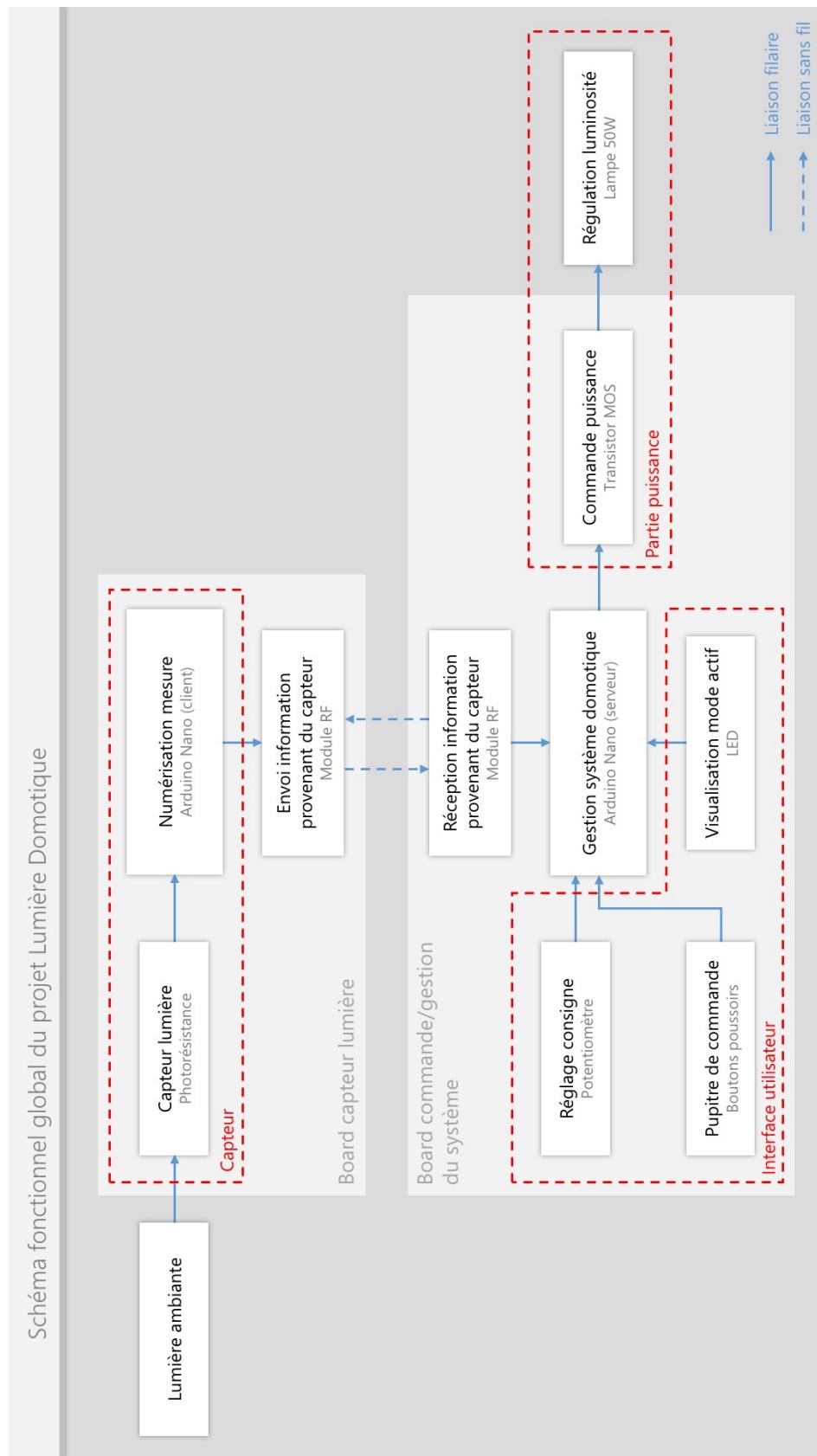
It works with and without wire, but lack of time forced us to realize wireless communication without using the JEEDOM server. Knowing that only two modules communicate, using a central server was not mandatory. This allowed us to recover the time we had lost.

We lost this time when we had to realize the separation in automatic and manual mode because we had a simple little mistake in our programming.

Thanks to this project, we developed our skills in programming and project management as well as our knowledge of Arduino hardware.

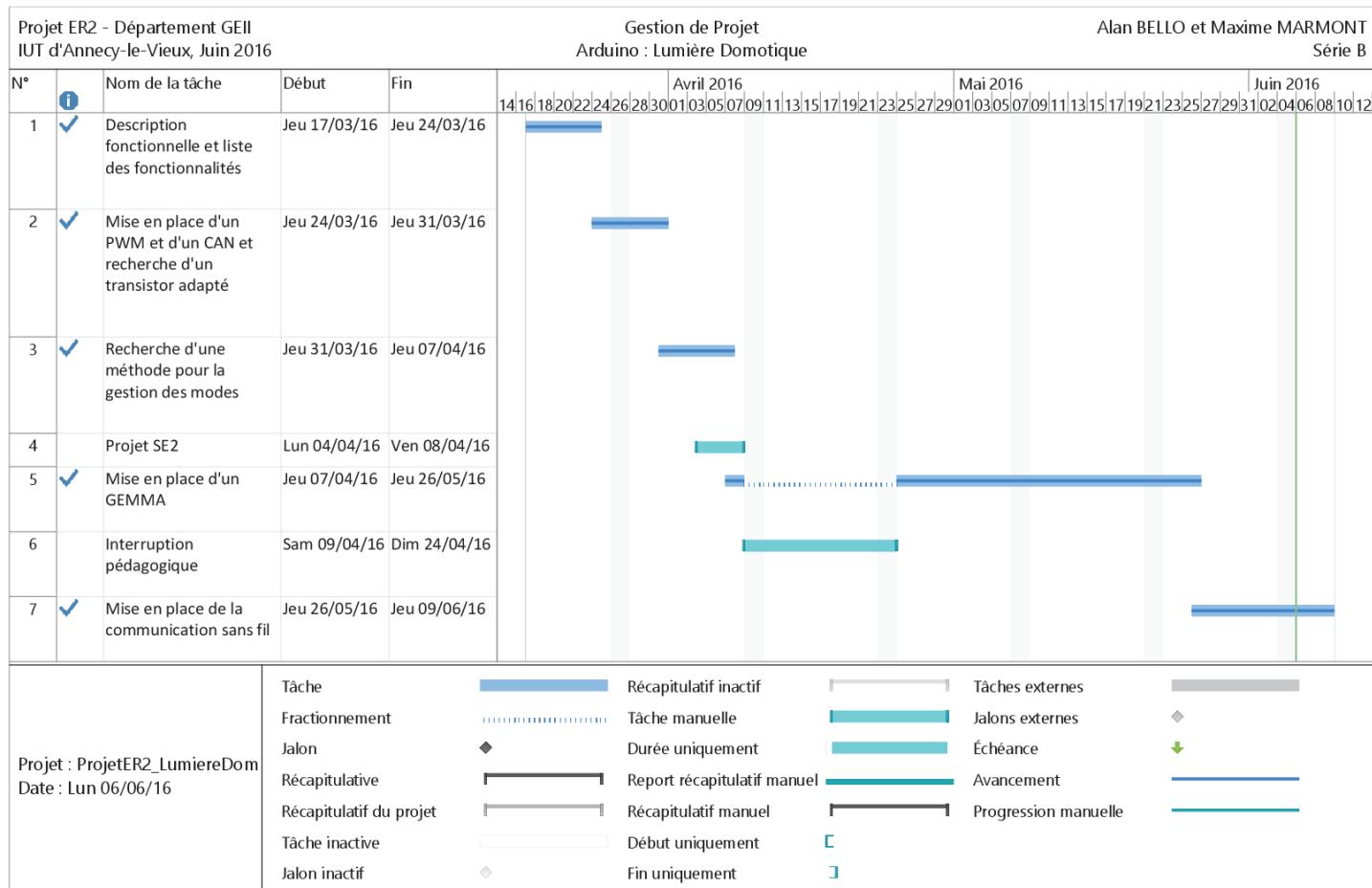
## 4. Annexes

### 4.1. Diagramme fonctionnel (avec communication sans fil)



## 4.2. Gestion de projet

### 4.2.1. Diagramme de Gantt



### 4.2.2. Nomenclature

NOMENCLATURE :		
Références:	Noms:	Valeurs:
LED1	LED	
MOS1	Transistor	
P1	Potentiomètre	50Kohm
Rph	PhotoRésistance	
R1,R2,R7,R8,R9,R10,R11, R12,R13,R14	Résistance	10Kohm
R3	Résistance	1Kohm
R4	Résistance	330ohm
R5	Résistance	120ohm
R6	Résistance	1Mohm
S1	Bouton Poussoir	
S2	Bouton Poussoir	

### 4.2.3. Devis du prototype (coûts horaires et prix du matériel)

IUT d'Annecy (Université Savoie Mont-Blanc)  
9, Rue de l'Arc-en-Ciel  
74940 Annecy-le-Vieux

**Devis Projet Lumière Domotique**  
**Alan BELLO - Maxime MARMONT**

Facture 20160614001002

juin 2016

MATERIEL						
Réf. Fact.	Produit/Description	REF. CONSTR.	Fournisseur	Prix unitaire	Qté	Prix
<b>10</b>	<b>MICROCONTROLEURS</b>					
11	ARDUINO NANO Atmega328, 8-bit, 16MHz			24,30 €	2	48,60 €
12	MODULE RF POUR ARDUINO Radio 2.4GHz, Contrôleur 16MHz	NRF24L01		5,49 €	2	10,98 €
<b>20</b>	<b>ELECTRONIQUE CLASSIQUE</b>					
21	BOUTON POUSSOIR (PACK DE 12 PCS)			3,24 €	1	3,24 €
22	LED ROUGE 1.7V			0,30 €	1	0,30 €
23	PHOTORESISTANCE			1,75 €	1	1,75 €
24	LAMPE HALOGENE 12V, 50W			2,00 €	1	2,00 €
<b>30</b>	<b>COMPOSANTS PASSIFS E12</b>	<b>VALEUR</b>				
31	POTENTIOMETRE	50k		1,15 €	1	1,15 €
32	RESISTANCE	10k		0,01 €	10	0,10 €
33	RESISTANCE	1k		0,01 €	1	0,01 €
34	RESISTANCE	330		0,09 €	1	0,09 €
35	RESISTANCE	1M		0,01 €	1	0,01 €

RESSOURCES HUMAINES					
Réf. Fact.	Département	Prs affectées	Coût hr net /prs	Coût hr net	Tps travail
40	RECHERCHE & DEVELOPPEMENT Etudiants	2	7,54 €	15,08 €	35h

<b>TOTAL MATERIEL</b>	<b>68,23 €</b>
<b>TOTAL RESSOURCES HUMAINES</b>	<b>527,80 €</b>
<b>TOTAL NET</b>	<b>596,03 €</b>

## ER2 Project – Automation Light

### Abstract

#### ▪ Keywords

- Arduino
- C/C++ language
- Automation
- Light sensor
- Wireless communication
- Energy saver

#### ▪ Abstract

The main objective is to command a lighting system in two different ways: a manual mode where we can vary the intensity of the light, and an automatic mode which introduces a light sensor. The intensity of the light must be manageable directly by the sensor, that is to say the higher the ambient light, the lesser the light. Moreover, the sensor is in a place while the light is in another room, and all the information must pass through an automation software installed on a computer in a different room also. It implies that we must create a wireless communication between all the elements.

To achieve our goal, we used two Arduino Nano cards each with an RF module (wireless module), liting with the same frequency. Then, we wired the photocell with a voltage divider bridge and the light with a MOSFET transistor because this one must have a higher voltage than the Arduino to work.