

Projet d'électronique

ASI3 2013-2014
Ludovic Henriet

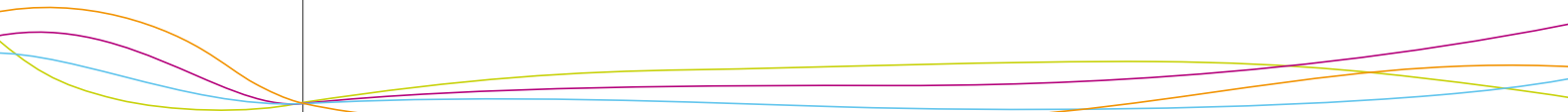
Antoine AUGUSTI
Etienne BATISE
Jean-Claude BERNARD
Thibaud DAUCE

PROJET D'ÉLECTRONIQUE

UN RÉVEIL INTELLIGENT

Table des matières

I	Introduction	2
1	Introduction	3
2	État de l'art	4
II	Cahier des charges	8
1	Cahier des charges	9
III	Étude technique	11
1	Électronique	12
IV	Documents de réalisation	15
1	Électronique	16
2	Calendrier prévisionnel	20
V	Étude logicielle	21
1	Raspberry	22
2	Arduino	23
VI	Estimation du coût total	25
1	Bon de commande	26
VII	Bilan et conclusion	27
1	Conclusion	28



Première partie

Introduction

Chapitre 1

Introduction

Dans le cadre du cours d'Électronique pour ASI, nous avons dû réaliser un projet de notre choix. Il nous a été proposé de reprendre un projet de l'année dernière ou de concevoir notre propre projet. Nous avons choisi de reprendre le projet de deux de nos camarades de l'année dernière. La durée approximative de ce projet était de 6 mois.

Notre groupe était composé de quatre membres : Antoine Augusti, Étienne Batise, Jean-Claude Bernard et de Thibaud Dauce. Le projet, que nous avons choisi, était le réveil intelligent. Celui-ci fonctionnait à l'aide du couplage de la carte Arduino et de la Raspberry Pi. Grâce à cette interaction, le réveil sonnait à une certaine heure et des informations s'affichaient à l'écran. Notre objectif a été de reprendre les outils qu'ils nous avaient laissés pour améliorer le réveil.

Dans ce rapport, nous abordons les nouveautés que nous avons apportées ainsi que les démarches effectuées pour mener le projet à son terme. Ainsi, dans une première partie nous présenterons le cahier des charges, suivi par les études et recherches menées ainsi que les documents de réalisation associés, pour enfin terminer par le bon de commande et le bilan de notre projet.

Chapitre 2

État de l'art

Introduction

Dans le cadre de notre première année de cycle ingénieur dans le département ASI¹, nous avons pour mission de réaliser pour le cours d'Électronique pour l'ingénieur dispensé par Monsieur HENRIET un projet d'une durée de 6 mois en rapport avec le domaine de l'électronique. Une partie de notre groupe ayant réalisé un robot auparavant dans le cadre du projet de P6, nous avons choisis de découvrir ensemble de nouvelles technologies de façon à agrandir notre vision dans ce domaine.

Notre attention s'est très rapidement portée sur l'un des projet déjà réalisé par un groupe d'étudiant l'année dernière : *Un réveil intelligent*. En effet, ce projet est véritablement dans la vague des nouvelles technologies domotiques qui deviennent de plus en plus populaires. C'est pourquoi nous nous sommes fixés comme but d'améliorer ce projet par de nouvelles fonctionnalités.

Malgré la réussite de ce projet, il est néanmoins indispensable d'effectuer pour nous un état de l'art sur les différents caractéristiques actuelles du projet d'une part, et de nos projets d'autres part. C'est pourquoi dans ce dossier nous allons d'abord vous parler des différents modèles de carte à programmer telles que les Arduino ou les Raspberry Pi. Ensuite nous présenterons les différents langages de programmation à utiliser et leur utilité. Enfin nous expliquerons quels nouveaux types de matériel nous allons utiliser pour arriver à notre but.

2.1 Les différents modèles de Raspberry

2.1.1 Qu'est-ce qu'un Raspberry ?

Un Raspberry, ou pour son nom complet un Raspberry Pi, est une mini-ordinateur, de la taille d'une carte de crédit, tournant avec un processeur ARM. Un processeur ARM est du même type que ceux que l'on trouve dans nos tablettes ou smartphone, leur principale caractéristique est leur faible consommation d'énergie. Le Raspberry Pi est composé uniquement d'une carte mère sans boîtier, sans alimentation ni stockage mais il possède un grand nombre d'entrées sorties dites standards. Les plus importantes sont : un lecteur de carte SD (pour le stockage), une prise HDMI (afin de le connecter à un écran), un ou des ports USB (afin en particulier de pouvoir y brancher un clavier et / ou une souris mais aussi des clés USB ou d'autres périphériques USB tel que un dongle Wi-Fi).

1. ASI : Architecture des Système d'Information



FIGURE 2.1 – Raspberry Pi

Le principal problème de ce mini-ordinateur est qu'il est alimenté en USB (5 Volts) ce qui explique le choix d'un processeur ARM, peu gourmand en énergie certes, mais peu puissant en contrepartie. Cette faible alimentation est aussi un problème concernant les périphériques USB. Par exemple, un clavier rétro-éclairé peut consommer trop d'énergie et rendre le Raspberry Pi défaillant.

Toutefois le Raspberry Pi possède de nombreux avantages. En premier lieu, il est capable de faire tourner un système d'exploitation de type GNU / Linux très puissant et permettant d'effectuer des tâches très variées. Il est en effet possible d'installer sur un Raspberry Pi un serveur web (de type Apache), un serveur mail, un système de partage de fichier (type Samba) ou encore une seedbox (afin de partager des fichiers torrent). Là où l'alimentation était un problème, sa faible consommation devient un avantage si il reste allumé 24h / 24 7j / 7. Ce mini-ordinateur se distingue aussi par ses incroyables capacités graphique car malgré un processeur peu puissant, il arrive à décoder des vidéos full HD (1080p) sans aucun problème. Il est souvent utilisé comme *media center*, branché à une télévision et au réseau, permettant ainsi de regarder des films stockés sur son ordinateur facilement.

Pour finir, le prix du Raspberry d'environ 30 euros le rend très attractif aux vues de ses capacités très vastes. Il peut être intégré dans de nombreux projets de robotique allant du ballon sonde à un serveur de domotique.

2.1.2 Deux grands types de Raspberry Pi

Il existe deux types de Raspberry Pi : le modèle A et le modèle B. Le modèle B se distingue de son homologue moins cher sur plusieurs points : une RAM plus importante (512 Mo au lieu de 256 Mo), un port USB supplémentaire, un port Ethernet 10/100 et une meilleure puissance électrique (700mA au lieu de 400mA).

Pour notre projet, nous n'avons pas le choix du Raspberry Pi car nous allons reprendre celui utilisé pour le projet de P6 du "Réveil intelligent". C'est un modèle B muni donc d'un port Ethernet mais malheureusement plus vieux que l'actuelle version donc n'ayant quand même que 256 Mo de RAM. Si nous avions dû choisir, c'est ce modèle plus performant que nous aurions pris en grande partie pour son port Ethernet permettant des applications réseau mais aussi pour ses meilleures performances d'alimentation qui pourront nous être utiles dans le futur.

Nous tenons à lever un problème du Raspberry Pi modèle B, les deux ports USB et le port Ethernet sont reliés au même composant "LAN9512" qui est lui-même connecté au CPU via un port USB2 : les débits sont donc divisés entre les trois éléments. Dans notre cas, ce n'est pas un problème car nous ne comptons pas effectuer des transferts importants de données en USB ou en réseau.

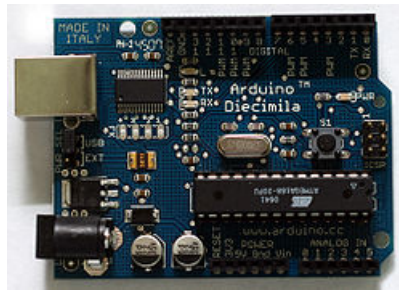


FIGURE 2.2 – Carte Arduino

2.1.3 La carte Arduino

Arduino est un circuit imprimé possédant un microcontrôleur lui permettant d'analyser des signaux électriques. Ce composant est déjà présent dans le projet que nous comptons reprendre et nous pouvons donc le réutiliser.

Contrairement à un Raspberry Pi, la carte Arduino possède un grand nombre d'entrées / sorties (une vingtaine alors que le Raspberry Pi en possède uniquement 6) mais ne peut pas exécuter de système d'exploitation et donc de programme en temps que tel. Le microcontrôleur peut quand même effectuer des calculs qui seront développés en C / C++. Le dernier point intéressant sur la Arduino est de pouvoir recevoir des signaux analogiques alors que le Raspberry Pi ne peut lire que des signaux numériques, il aura donc en charge de transformer les signaux analogiques des capteurs (luminosité par exemple) en numérique (0 ou 1 en fonction d'un seuil).

2.2 Les différents langages

Nous allons être amenés à utiliser plusieurs langages de programmation bien différents les uns des autres pour pouvoir réaliser ce projet. La séparation la plus marquée pour ces langages peut se faire par leur but : une partie des langages sera utilisée pour l'intelligence de nos applications, tandis que les autres langages seront dédiés à la création d'une IHM².

2.2.1 Les langages dédiés à l'intelligence de notre application

La logique métier et les différentes fonctions de notre application seront écrites en PHP³ et en script shell.

PHP est un langage de programmation compilé à la volée libre principalement utilisé pour produire des pages Web dynamiques via un serveur HTTP, mais pouvant également fonctionner comme n'importe quel langage interprété de façon locale. Le langage PHP permettra de programmer les différentes fonctions de notre application ainsi que de stocker et d'aller chercher les données utiles à son fonctionnement.

Les scripts shell (également appelés scripts bashs) permettent d'automatiser une série de commandes exécutées dans un terminal. Un script shell se présente sous la forme d'un fichier contenant une ou plusieurs commandes qui seront exécutées de manière séquentielle. Ces scripts nous permettront d'automatiser des opérations de maintenance, de mises à jour ou de modifications de notre Raspberry.

2. IHM : Interface Homme Machine

3. PHP : *PHP Hypertext Preprocessor*.

2.2.2 Les langages dédiés à la création de l'IHM

L'Interface Homme Machine de notre application sera accessible depuis un navigateur web (Firefox ou Google Chrome par exemple). Pour pouvoir proposer une IHM accessible depuis un navigateur web, il est obligatoire d'utiliser les langages associés au monde du web à savoir HTML⁴ et CSS⁵. Nous utiliserons également du JavaScript pour pouvoir proposer une interface plus dynamique.

Le HTML est le format de données conçu pour représenter les pages web. C'est un langage de balisages qui permet d'indiquer qu'un élément est un titre, un paragraphe, un lien, une liste, un élément d'une liste etc.

Les feuilles de style CSS, quand elles sont associées à du HTML, permettent de mettre en forme une page web en spécifiant comment sont positionnés les éléments, quelles couleurs il faut utiliser, quels sont les tailles et espaces à respecter. Le CSS définit l'aspect graphique de la page web tandis que le HTML ne s'occupe que de la sémantique du contenu de celle-ci.

Le JavaScript (souvent abrégé JS) est un langage de programmation de scripts principalement utilisé dans les pages web interactives mais aussi côté serveur. Il permet de modifier des pages web quand un événement est déclenché (le survol d'un élément, un délai dépassé, une position atteinte etc.) sans devoir rafraîchir celle-ci, donnant la possibilité d'avoir des modifications instantanées à l'écran dès qu'une action de la part de l'utilisateur est effectuée. Le JavaScript est capable de modifier du CSS et du HTML.

2.3 Les choix de la source de lumière

On trouve trois grandes familles d'ampoules : les lampes à incandescences, les sources à décharge lumineuses pressurisées et les sources électroluminescentes.

La lampe à incandescence classique, inventée en 1879 par Joseph Swan et améliorée par les travaux de Thomas Edison, ou à halogène, inventée en 1959, produit de la lumière en portant à incandescence un filament de carbone (à l'origine) ou de tungstène.

Ensuite, on a les sources à décharge lumineuses pressurisées telles que les lampes fluorescentes compactes. Celles-ci produisent de la lumière grâce à un mélange de gaz et/ou de vapeur excité par une décharge électrique. Contrairement aux lampes à incandescences, on a une plus grande luminosité (jusqu'à 115 lumens par watt) et le coût est similaire. Toutefois l'allumage n'est pas instantané.

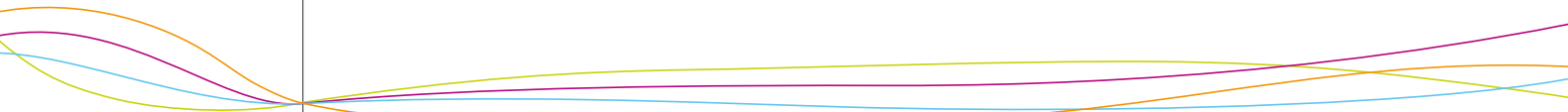
Enfin, les lampes électroluminescentes sont constituées d'un matériau semi-conducteur traversé par un courant électrique et émettent une couleur bleue. Par des procédés chimiques, on va pouvoir convertir la lumière en jaune. Ces types de lampes ont la plus grande longévité et chauffent beaucoup moins que les lampes à incandescences. Néanmoins, ce sont les ampoules les plus chères du marché actuellement.



FIGURE 2.3
— Ampoule
à incandescence

4. HTML : *Hypertext Markup Language*.

5. CSS : *Cascading Style Sheets*



Deuxième partie

Cahier des charges

Chapitre 1

Cahier des charges

1.1 Électronique

- Faire un circuit imprimé pour remplacer le shield actuel
- Implémenter un système d'ampoule
- Implémenter une alimentation autonome
- Effectuer tous les branchements

Le shield existant permet, grâce à une photorésistance, de déterminer la luminosité dans la pièce où se situe le réveil. Il est fonctionnel mais nous devons créer notre propre carte pour répondre aux besoins du projet. Cette carte aura les mêmes objectifs que le shield, c'est-à-dire retourner une valeur analogique en fonction de la luminosité de la pièce. L'ancien shield possède une LED, contrôlée par la Arduino, permettant de savoir quand la luminosité dépasse un certain seuil. Pour notre projet, nous comptons intégrer le réveil dans une boîte et nous pensons réutiliser la LED afin de prévenir l'utilisateur que la batterie du réveil est faible.

Nous devons choisir l'ampoule et la connecter à notre circuit imprimé afin de pouvoir contrôler la quantité de courant à envoyer et ainsi proposer un éclairage progressif.

Le dernier objectif de cette partie d'électronique est d'effectuer des montages et par extension des câblages propres afin de produire un travail de qualité et de faciliter l'intégration des différentes cartes dans un boîtier.

1.2 Arduino

- Implémentation de la gestion de l'ampoule
- Réutiliser la LED pour signaler que la batterie est faible

Au niveau de la Arduino, le code présent permet déjà de déterminer lorsque le seuil de luminosité est dépassé. Il nous faudra donc ajouter l'allumage progressif de l'ampoule. Nous devons aussi résoudre les conflits entre l'allumage de l'ampoule et le capteur de luminosité afin de ne pas précipiter l'arrêt du réveil. Nous pourrions pour cela adapter le code déjà produit par l'équipe précédente et implémenter, par exemple, une variation du seuil de luminosité en fonction du stade d'allumage de l'ampoule.

La gestion de la batterie est importante et nous devons trouver un moyen de déterminer quand le

niveau de batterie du réveil est faible pour pouvoir prévenir l'utilisateur (afin de ne pas se retrouver sans réveil du jour au lendemain). À ce stade de la conception, nous ne savons toujours pas si cette information pourra être récupérée directement via la Arduino ou si (plus probablement) nous devrions récupérer certaines informations via notre circuit imprimé. Cette batterie va nous permettre d'alimenter le Raspberry Pi et la Arduino sans câble, si nous voulons un boîtier autonome nous avons aussi besoin de supprimer le câble RJ-45 permettant l'accès au réseau. Nous devons donc rajouter un dongle Wi-Fi qui peut consommer très rapidement beaucoup de courant. Il faut donc aussi implémenter une gestion du Wi-Fi : bouton on / off ou allumage intelligent (un peu avant le déclenchement du réveil par exemple).

1.3 Raspberry Pi

- Affichage des prochains métros
- Gestionnaire de sons
- Implémentation de l'affichage LCD
- Paquet magique sur le réseau (broadcast)

L'affichage des prochains métro semble ne pas fonctionner actuellement sur le projet, sans doute à cause d'un changement sur le site de la TCAR. Il nous faudra réparer cette fonctionnalité.

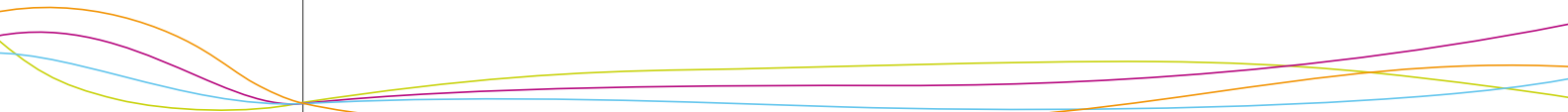
Au niveau des améliorations, il n'existe pas de gestionnaire de son sur l'interface web permettant l'administration du réveil et donc, le changement de la sonnerie du réveil est laborieux. Nous comptons implémenter une nouvelle page web permettant d'ajouter des nouveaux sons et de choisir celui à jouer au prochain réveil.

Les deux dernières fonctionnalités sont optionnelles, même si nous aimerions les implémenter, il est peu probable que nous ayons le temps vu les délais à respecter pour le projet. La première consiste à gérer l'affichage de l'heure via un écran LCD et la deuxième, un peu plus technique mais très utile, permettrait d'envoyer un paquet "magique" TCP / IP sur le réseau local de l'utilisateur lors de la sonnerie du réveil (et pourquoi pas lors de son extinction) afin de donner la possibilité de réutiliser ce projet. En effet, un paquet "magique" est un paquet qui peut être lu par tous les appareils connectés au réseau, cela donne donc la possibilité de créer un autre projet d'électronique (par exemple un autre Raspberry Pi) analysant le réseau et exécutant une action lors de la détection de ce paquet (par exemple, allumage de l'ordinateur, de la cafetière ou autre...).

1.4 Bricolage

- Construire une boîte

Il ne faut pas oublier de prendre le temps de construire une boîte afin de protéger les circuits et rendre le projet plus esthétique.



Troisième partie

Étude technique

Chapitre 1

Électronique

1.1 La lumière : Shield LDR

1.1.1 Schéma du Shield LDR

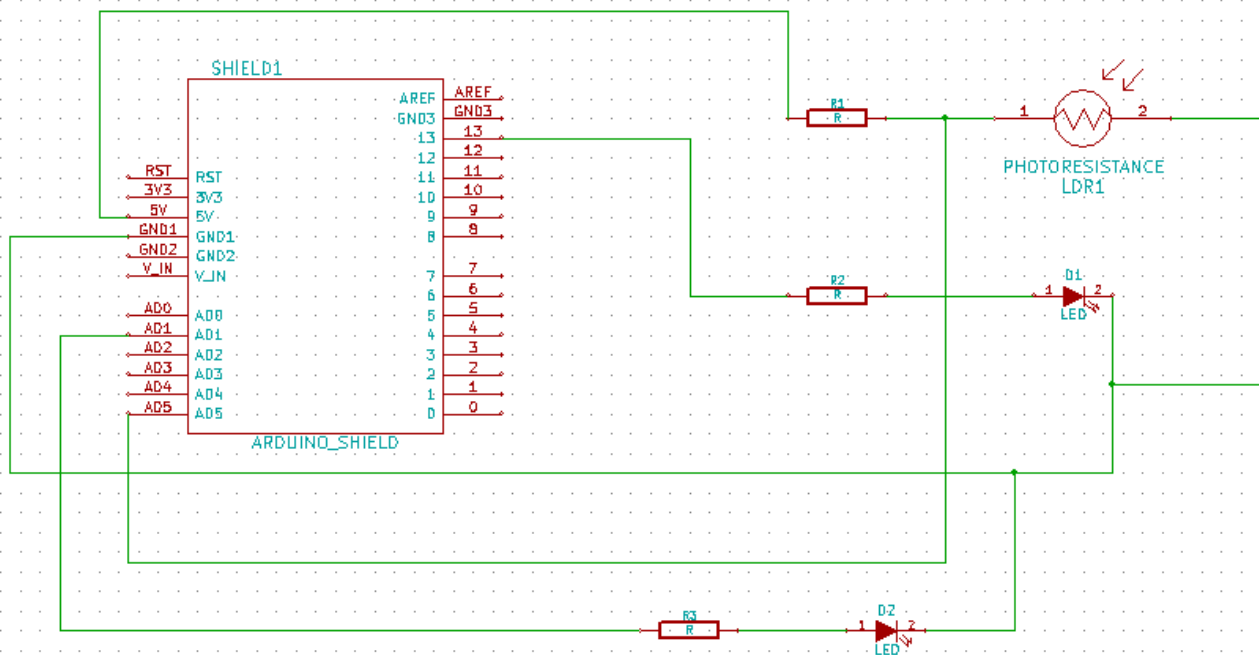


FIGURE 1.1 – Logiciel utilisé : Eeschema de Kicad

1.1.2 Choix des composants

Notre projet étant une reprise d'un projet de l'année dernière, nous avons dû refaire le shield. Nous avons donc récupéré leurs composants à savoir :

- Une Arduino Mega 2560
- Deux LEDs

- Une photorésistance LDR
- Trois résistances

1.2 Alimentation autonome : 7805

1.2.1 Schéma du régulateur de tension

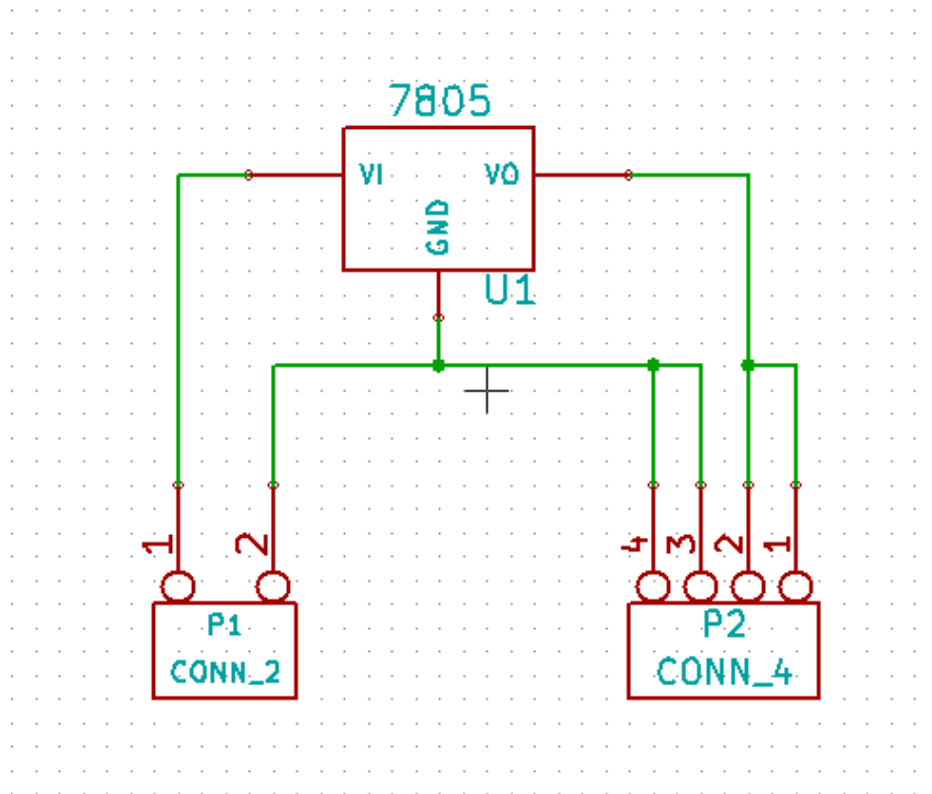


FIGURE 1.2 – Logiciel utilisé : Eeschema de Kicad

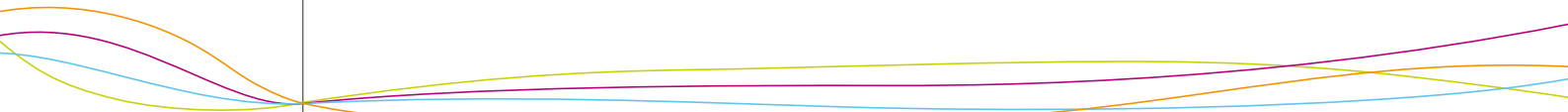
1.2.2 Choix des composants

Nos camarades de l'année dernière alimentaient leurs Arduino et Raspberry avec une prise secteur. Cette année, nous avons décidé d'en faire un réveil portable. Et qui dit portable, dit alimentation autonome. Les deux appareils étant alimentés en 5V, nous avons commandé un régulateur de tension 7805 qui délivre du 5V/1A. On met donc en entrée du régulateur un pack d'accus de 7V et le régulateur va se charger de délivrer du 5V à l'Arduino et à la Raspberry branchées en dérivation sur celui-ci.

1.3 Communication Arduino-Raspberry

Comme la Arduino reçoit les informations du capteur de luminosité et allume la lampe, et que le Raspberry est en charge du réveil, il a fallu établir une communication entre les deux cartes. Pour cela nous utilisons deux pins du Raspberry et deux pins de la Arduino afin de pouvoir envoyer et recevoir des informations grâce à une liaison bi-directionnelle.

D'un côté, le Raspberry va envoyer un signal de réveil à la Arduino afin qu'elle puisse commencer à allumer la LED. Et inversement, lorsque que la Arduino détecte une source de lumière, elle envoie un signal au Raspberry afin qu'il puisse éteindre la sonnerie du réveil.



Quatrième partie

Documents de réalisation

Chapitre 1

Électronique

1.1 Shield LDR

1.1.1 Circuit du Shield LDR

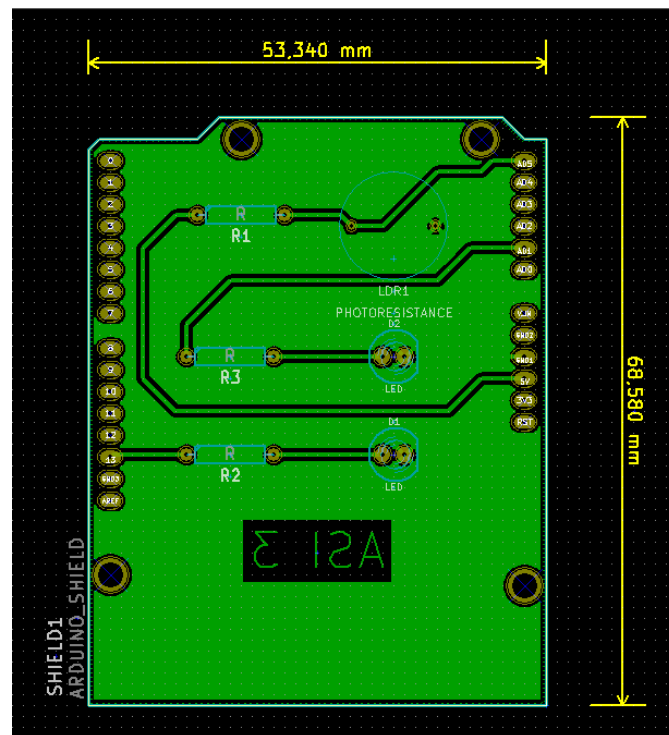


FIGURE 1.1 – Logiciel utilisé : Pcbnew de Kicad

1.1.2 Représentation de la carte en 3D

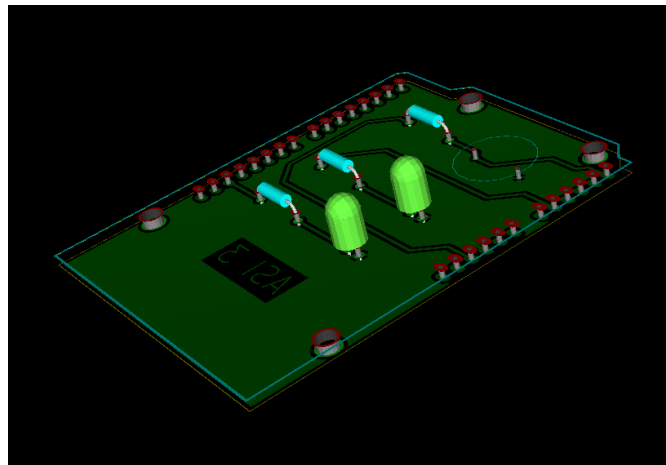


FIGURE 1.2 – Représentation 3D du shield LDR

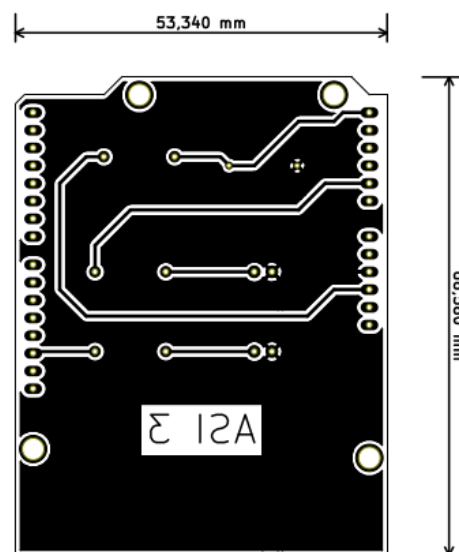


FIGURE 1.3 – Typon du shield LDR

1.1.3 Manoeuvres réalisées sur la carte

Vérification des cartes

Test de court-circuit : On a vérifié à l'aide d'un multimètre en mode "test de continuité", si il n'y avait aucunes microcoupures ou court-circuit. Nous en avons rencontrés aucun.

Assemblage des composants et soudure

1.2 Alimentation autonome : 7805

1.2.1 Circuit du régulateur de tension

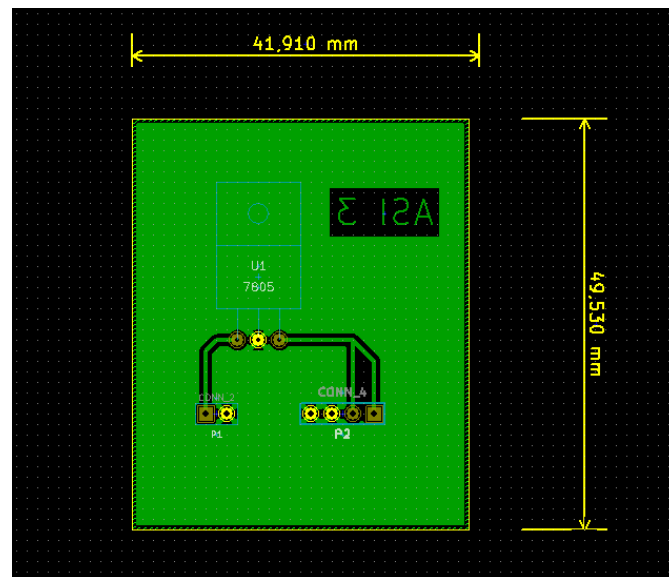


FIGURE 1.4 – Logiciel utilisé : Pcbnew de Kicad

1.2.2 Représentation de la carte en 3D

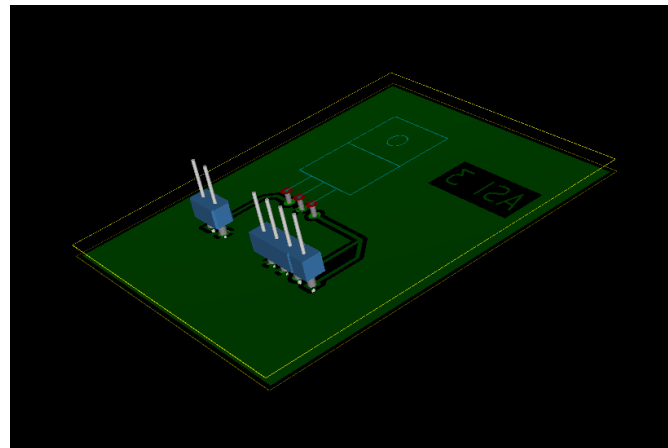


FIGURE 1.5 – Représentation 3D de la carte du régulateur

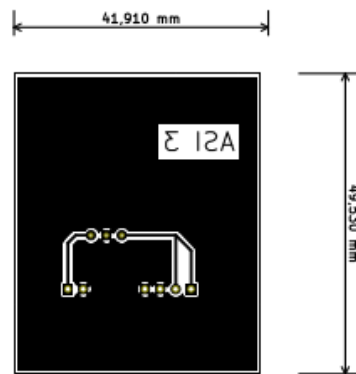


FIGURE 1.6 – Typon du régulateur

1.2.3 Manoeuvres réalisées sur la carte

Vérification des cartes

Test de court-circuit : On a vérifié à l'aide d'un multimètre en mode "test de continuité", si il n'y avait aucunes microcoupures ou court-circuit. Nous en avons recontrés aucun.

Assemblage des composants et soudure

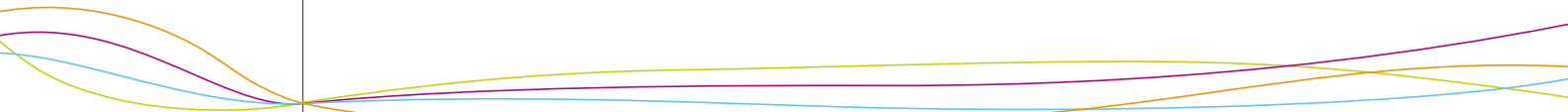
Nous avons eu quelques soucis lors de la soudure des composants. En effet deux d'entre nous n'étaient pas habitués à souder. Mais finalement, on eu aucun problème de court circuit. Cependant, dû à la mauvaise qualité du connecteur pour la Raspberry, nous avons des faux contacts qui entraîne la mise sous hors-tension de celle-ci lorsqu'il y a un choc.

Chapitre 2

Calendrier prévisionnel

Retrouvez ci-dessous le calendrier prévisionnel de la réalisation de notre projet.

Dates	Électronique	Arduino	Raspberry Pi
25/09/2013	Determiner le travail à fournir	Prendre en main le logiciel Arduino	Découvrir l'interface RaspberryPi
09/10/2013	Remplacer le Shield Arduino par une carte PCB	Commencer le développement de la gestion de l'ampoule	
23/10/2013	Ajouter une ampoule au montage et trouver une solution pour une alimentation autonome	Effectuer les tests de la gestion de l'ampoule	
13/11/2013	Implémentation de l'alimentation autonome	Développer un indicateur de la charge restante de la batterie	Commencer le développement du gestionnaire de mètres
27/11/2013	Effectuer et vérifier tous les branchements définitivement		Commencer le développement du gestionnaire de sons
11/12/2013			Commencer la gestion d'un affichage sur écran LCD



Cinquième partie

Étude logicielle

Chapitre 1

Raspberry

1.1 Réseau

Le principal problème que nous avons rencontré était au niveau de la gestion des contraintes réseau : nous avons besoin d'utiliser le serveur local du Raspberry pour proposer l'interface web de configuration du réveil et un accès au web complet pour proposer des services extérieurs : les prochains métros, la météo et les news.

Pour nous connecter au web nous avons à notre disposition dans les salles de TP le réseau de l'INSA qui oblige à passer par le proxy¹ de l'INSA pour atteindre le web.

Malheureusement l'utilisation du proxy de l'INSA est incompatible avec l'utilisation d'un serveur local simultanément. Pour pouvoir utiliser ces deux services simultanément nous avons besoin de spécifier des règles de proxy spécifiques et supplémentaires par rapport à ce que le proxy de l'INSA propose naturellement. Une fois la solution trouvée, nous avons alors créé un fichier de configuration pour le proxy répondant à notre besoin.

Enfin, le navigateur web proposé par défaut sur Raspberry n'étant pas capable de lire ce type de fichier de configuration, nous avons téléchargé le navigateur web Chromium.

1.2 Gestionnaire de sons

Pour être efficace, notre réveil devait jouer un son pour réveiller la personne ayant programmé l'alarme. L'alarme avait été préalablement conçue pour que le son joué pour réveiller soit toujours le même.

Nous avons voulu améliorer ceci en jouant un son aléatoire pour réveiller la personne. Nous avons réalisé cette fonctionnalité à l'aide d'un script Bash qui joue aléatoirement un des fichiers de musique se trouvant dans un répertoire spécifique.

1. En informatique, un proxy est un composant logiciel qui joue le rôle d'intermédiaire en se plaçant entre deux autres pour faciliter ou surveiller leurs échanges.

Dans le cadre plus précis des réseaux informatiques, un proxy est alors un programme servant d'intermédiaire pour accéder à un autre réseau, généralement internet. Par extension, on appelle aussi proxy un matériel (un serveur par exemple) mis en place pour assurer le fonctionnement de tels services.

Chapitre 2

Arduino

2.1 Arduino IDE

Afin de développer le programme principal de la carte Arduino, nous avons utilisé le logiciel officiel Arduino IDE.

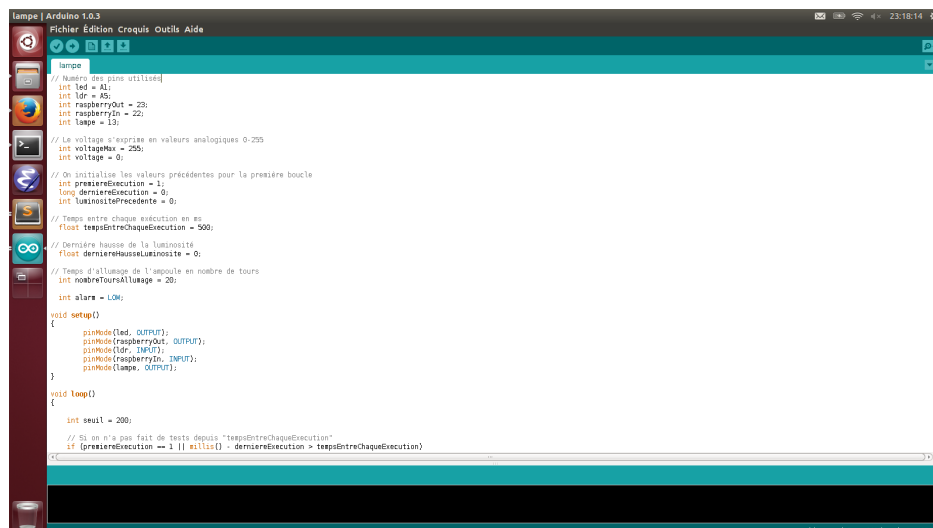


FIGURE 2.1 – Logiciel utilisé : Arduino IDE

La prise en main de ce logiciel c'est fait très simplement, c'est un éditeur de code simple. La seule subtilité rencontrée a été de régler le modèle de la carte Arduino avant de transférer le programme sur la carte.

2.2 Le langage Arduino

Le langage utilisé pour programmer la carte Arduino est un dérivé du langage C. Cette particularité nous a facilité la programmation des fonctionnalités de notre réveil intelligent grâce aux cours d'algorithmique donnés en parallèle en ASI3.

Les limitations de langage ne nous ont posés aucun soucis. Les seules nouveautés à appréhender ont été les diverses connexions, lectures et écritures sur les pins de la carte Arduino.

2.3 Le programme

Le programme installé sur la carte Arduino n'est pas très complexe mais nous avons tout fait pour le garder évolutif.

Tout d'abord, le programme vérifie le pin d'entrée provenant du Raspberry afin de savoir si l'alarme a été déclenché ou pas. Si c'est le cas, il va commencer l'allumage progressif de la LED en augmentant la valeur de sortie du pin de la LED. Dans un deuxième temps, la Arduino va vérifier la valeur envoyée par la photorésistance, si cette valeur dépasse un certain seuil (défini à 200 dans notre cas), le programme éteint la LED et envoie un signal sur le pin allant au Raspberry qui se chargera alors d'éteindre l'alarme.

Ces instructions se répètent dans une boucle infinie. Afin d'éviter les pauses et que la carte Arduino soit toujours active, nous avons voulu ne pas utiliser la fonction "sleep". La carte Arduino fait donc tourner à son rythme la boucle, et les instructions sont déclenchées en fonction du temps écoulé. La variable *tempsEntreChaqueExecution* nous permet de réguler la vitesse de réponse de la carte Arduino. Cette astuce permet de rajouter très facilement des instructions pour un autre projet en rajoutant simplement une autre condition et une autre variable définissant un temps entre chaque exécution.

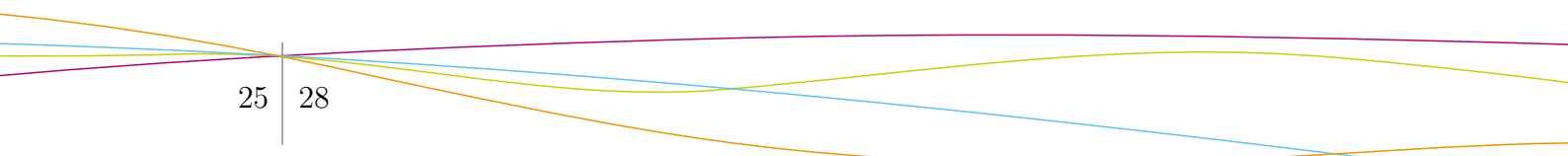
Toujours dans un souci de garder le programme évolutif, nous n'avons pas inscrit en dur les informations concernant l'augmentation de voltage de la LED. Cette information est calculée à partir de deux variables, la première que nous venons d'évoquer est le *tempsEntreChaqueExecution* et la deuxième est le *nombreToursAllumage*. Grâce à ces deux variables nous pouvons calculer avec précision le temps d'allumage de la LED. Nous pouvons aussi régler la continuité de l'allumage (en diminuant le *tempsEntreChaqueExecution* et en augmentant le *nombreToursAllumage*) afin d'éviter de trop grand saut de voltage. Nous pouvons imaginer dans un nouveau projet une interface permettant à l'utilisateur de facilement régler les paramètres d'allumage de sa LED.

Notre programme permet aussi de renseigner le voltage maximal si, par exemple, l'utilisateur ne veut pas que la LED s'allume complètement.



Sixième partie

Estimation du coût total



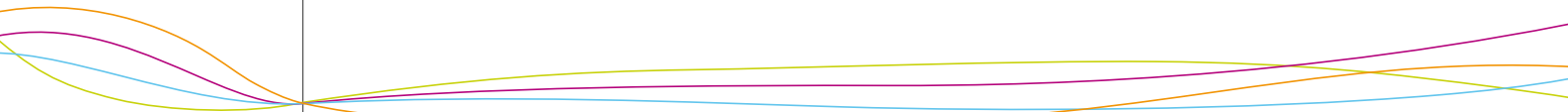
Chapitre 1

Bon de commande

Retrouvez ci-dessous le bon de commande pour le matériel nécessaire à notre projet.

Prévision Matériel	Référence	Prix HT	Prix TTC	Quantité
Régulateur 5V / 1A	7805	0.42 €	0.50 €	1
Batterie Li-po 3.9V	MPERI-K750I	11.66 €	13.95 €	1
Led blanche diffusante 5mm	L-5WDN/1	0.29 €	0.35 €	1
Platine Arduino Mega 2560	MEGA-2560	30.00 €	35.88 €	1
Kit shield prototypage	KIT-LEXSP001	6.50 €	7.77 €	1
Led verte	LED5GLN	0.17 €	0.20 €	1
Photorésistance 11 mm	LDR1000	1.18 €	1.48 €	1
Raspberry Pi (Modèle A)	RASPB1	32.00 €	38.27 €	1

FIGURE 1.1 – Bon de commande



Septième partie

Bilan et conclusion

Chapitre 1

Conclusion

La réalisation de ce projet d'électronique nous a permis tout d'abord d'acquérir des compétences dans de multiples domaines : l'électronique, la programmation et le réseau. Nous avons dû faire appel à de nombreuses connaissances et nous avons effectué de nombreuses recherches pour pouvoir réaliser l'évolution du réveil intelligent, particulièrement en électronique pour la Raspberry et la Arduino.

Bien que nous étions à travailler en groupe avec les précédents projets que nous avons pu réalisé lors de notre scolarité à l'INSA, ce projet a été différent car pour la première fois nous devions rendre un objet au terme du projet.

La partie qui nous a demandé le plus de travail a été l'interaction entre les différentes parties de notre projet : l'interface web de configuration, le système d'exploitation de la Raspberry ainsi que la communication entre la Raspberry et la Arduino.

Bien que notre projet puisse être amélioré et que celui ne soit absolument pas présentable en tant que réveil intelligent complet, nous sommes satisfaits du travail que nous avons fourni car nous avons rencontré divers obstacles avant d'arriver au résultat actuel.