

Light Traffic

AUCLAIR William, LENOIR Adrien, 1G2TP5

14 Juin 2021



Contents

1	Introduction	3
1.1	Préambule	3
1.2	Problématique	3
2	Réalisations	3
2.1	Matériel à disposition	3
2.1.1	Descriptif	3
2.1.2	Hardware	3
2.1.3	Software	3
2.2	Conception	4
2.2.1	Principe de fonctionnement	4
2.2.2	Maquette	6
2.2.3	Système de détection	8
2.2.4	Feux de signalisations	10
2.2.5	Affichage de la temporisation	11
3	Conclusions	13
3.1	Résultats	13
3.2	Problèmes rencontrés	13
3.3	Coûts	14
3.4	Conclusion	14
4	Annexes	14

1 Introduction

1.1 Préambule

Régime de priorité à droite, croisement, manque de visibilité, cyclistes, piétons... Les intersections, jonction de deux ou de plusieurs routes, demeurent parmi les tronçons de la chaussée les plus dangereux et difficiles à aborder (ralentissement, contrôle des usagers en provenance des différentes voies ...). Afin de faciliter le passage des véhicules et d'assurer la sécurité des usagers, on munit certaines intersections de feux tricolores. La régulation de la circulation devient ainsi plus aisée.

1.2 Problématique

On choisit de modéliser physiquement sur une maquette de taille réduite la régulation du trafic à l'aide de feux tricolores à proximité d'une intersection en T entre deux routes principale et secondaire. Le contrôle de la circulation sera facilité par l'ajout de capteurs de présence détectant les usagers ainsi que de chronomètres affichant le temps restant à attendre aux feux rouge.

Nos objectifs sont multiples : Garantir une visibilité optimale sur les feux tricolores afin de permettre aux usagers d'anticiper et donc de limiter le risque d'accident, ainsi qu'afficher le temps d'attente estimé dans une optique environnementale et économique (On peut ainsi limiter le gaspillage de carburant et l'émission de CO₂).

2 Réalisations

2.1 Matériel à disposition

2.1.1 Descriptif

Light Traffic est une maquette réalisant la signalisation routière sur une intersection "en T", avec détection des véhicules, affichage des temps d'attente et gestion complète des cas.

2.1.2 Hardware

Nous avions à notre disposition les différentes ressources matérielles dont nous avions besoin, à savoir :

- Carte de développement STM32F407 ;
- Trois triplets (vert, jaune et rouge) de LEDs ;
- Trois duos d'afficheurs 7 segments ;
- Trois couples Laser - LDR (*Light detecting resistor*) .

2.1.3 Software

Pour ce qui est des ressources logiciels, tout le projet a été codé en langage C sous l'IDE KeilµVision, et pour tout le reste, nous avions accès aux logiciels suivants :

- EAGLE (Logiciel de conception de circuits imprimés) ;
- Fusion 360 (Logiciel de CAO gratuit avec licence étudiante) .

2.2 Conception

2.2.1 Principe de fonctionnement

On choisit premièrement la durée de la simulation (en termes de nombre de cas traités avant l'arrêt du programme). Ensuite s'opère une disjonction de cas à chaque itération de la boucle principale. Celle-ci s'appuie sur l'état actuel des feux tricolores et la présence ou non des véhicules au niveau des différents capteurs.

En fonction du résultat, les différentes LEDs (feux) s'allument, avec systématiquement une seule des trois LEDs allumée par poteau tricolore. Les différents cas possibles et interdits sont définis dans la section "Feux de signalisations".

Dans le code est définie une structure ***FEU*** qui décrit un feu de signalisation de par 6 champs contenant des *int* :

- L'identifiant *id* ;
- Le code correspondant au chiffre à afficher sur l'afficheur des dizaines *cpt_d* ;
- Le code correspondant au chiffre à afficher sur celui des unités *cpt_u* ;
- L'état de la LED verte *vert* ;
- L'état de la LED jaune *jaune* ;
- L'état de la LED rouge *rouge* .

C'est cette structure, dont trois exemplaires sont initialisés (un pour chaque feu), qui va être appelée par les différentes fonctions d'affichage et de temporisation au fur et à mesure des itérations de la boucle du main().

La carte STM32F407 possède une multitude de ports GPIO accessibles et disponibles. La figure 1 présente un tableau synthétisant ceux qui seront utilisés. Les noms associés seront explicités par la suite.

Pins utilisés	Utilisation	Pins utilisés	Utilisation	Pins utilisés	Utilisation
PA0	Anode 0 Diz	PB0	Cathode Diz 1	PC0	Capteur 1 (P1)
PA1	Anode 1 Diz	PB1	Cathode Diz 2	PC1	Capteur 2 (P2)
PA2	Anode 2 Diz	PB2	Cathode Diz 3	PC2	Capteur 3 (S)
PA3	Anode 3 Diz	PB3	Cathode Unit 1	PC3	
PA4	Anode 4 Diz	PB4	Cathode Unit 2	PC4	
PA5	Anode 5 Diz	PB5	Cathode Unit 3	PC5	
PA6	Anode 6 Diz	PB6	LED verte 1	PC6	
PA7	Anode 0 Unit	PB7	LED Jaune 1	PC7	
PA8	Anode 1 Unit	PB8	LED Rouge 1	PC8	
PA9	Anode 2 Unit	PB9	LED verte 2	PC9	
PA10	Anode 3 Unit	PB10	LED Jaune 2	PC10	
PA11	Anode 4 Unit	PB11	LED Rouge 2	PC11	
PA12	Anode 5 Unit	PB12	LED verte 3	PC12	
PA13	Anode 6 Unit	PB13	LED Jaune 3	PC13	
PA14		PB14	LED Rouge 3	PC14	
PA15		PB15		PC15	

Figure 1: Ports GPIO utilisés pour le projet, broche par broche

Toutes les anodes et les cathodes d'afficheurs sont reliées à la carte en configuration de sorties "NOPULL", tout comme les anodes des 9 LEDs de signalisation. Les 3 capteurs (LDRs) lui sont reliés en entrée "PULL DOWN". La figure 2 présente sur un fichier .sch réalisé sous EAGLE le branchement des différents composants opto-électroniques au pin-header principal (de taille 17*2), lui-même relié à la carte de développement. On y retrouve sous les appellations AD, AU, CD, CU, et CPT_0-2 les ports qui apparaissent dans le tableau de la figure 1. Leur rôle à tous sera précisé dans la partie "Affichage de la temporisation".

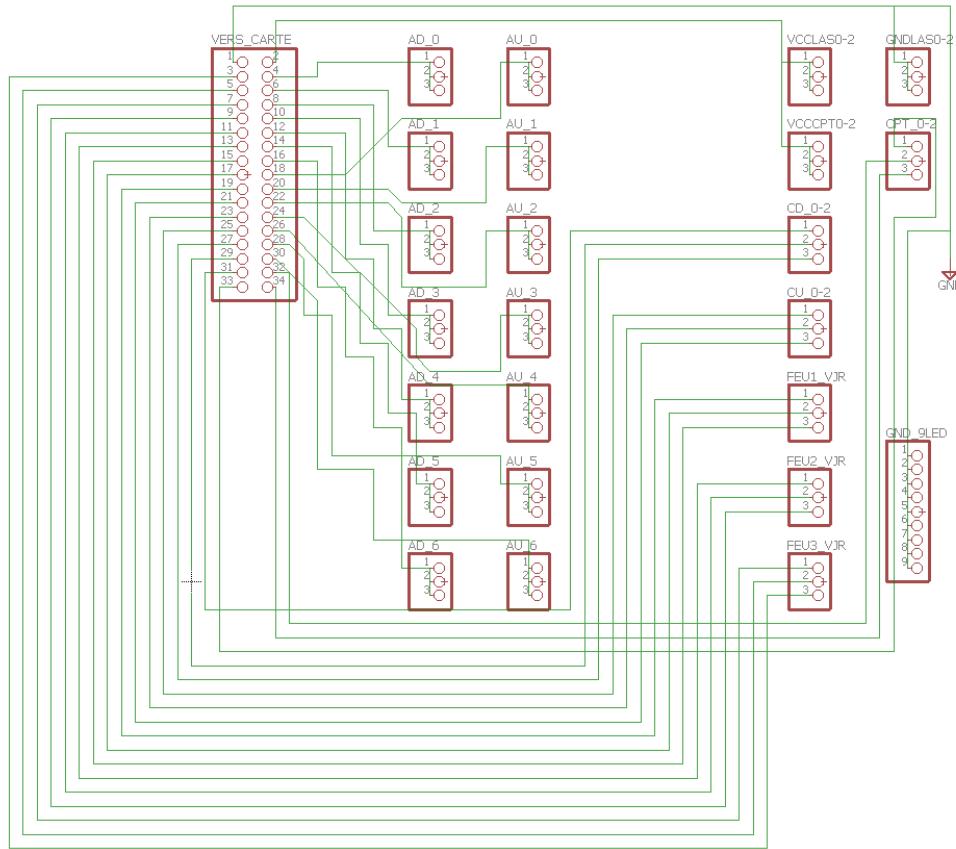


Figure 2: Fichier schematic, réalisé sous EAGLE

2.2.2 Maquette

On peut décomposer l'architecture matérielle du projet comme suit :

- Milieu urbain
- Intersection en T
- Bâtiments
- Véhicules
- Feux de signalisations
- Compteurs associés aux feux
- Capteurs de présence

Le but était donc de traduire un maximum de ces éléments sur la maquette. Deux bâtiments d'angles justifient l'utilité du système. En effet, ceux-ci empêchent les automobilistes de s'apercevoir. Sur chacune des trois branches de l'intersection, il y a un couple Laser-LDR officiant en capteur de présence, un duo d'afficheurs sept segments servant au décompte du temps d'attente, et un feu tricolore de signalisation.

Généralités

La maquette se présente comme un plateau de 25x25 cm, les routes sont larges de 5 cm et les voitures de 2 cm (et hautes de 1.3 cm) afin d'essayer de conserver un rapport de taille réel entre les deux.

Les supports des LASERs et des LDRs alignent leur centre à une hauteur de 1cm par rapport au sol.

Les différentes pièces de la maquette ont été conçues sur le logiciel Fusion 360, puis imprimées en 3D.

Tous les designs de pièces réalisés sous Fusion 360 sont visibles en figure 3. Dans l'ordre : bâtiment, afficheurs, feu, support LASER et LDR, Voiture. Chacunes de leur dimensions ont été prises de sorte à ce que l'on puisse y insérer le matériel optoélectronique prévu.

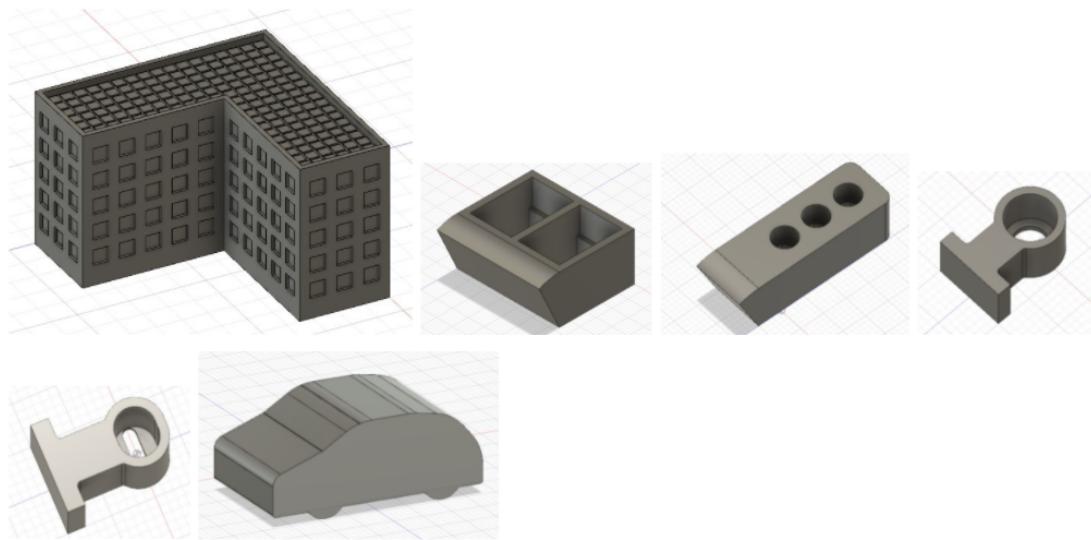


Figure 3: Modèles 3D de la maquette, captures d'écrans sur Fusion 360

Ces modèles exportés au format STL on pu être imprimés en 3D, donnant lieu aux résultats visibles en figure 4, une fois l'électronique implantée.

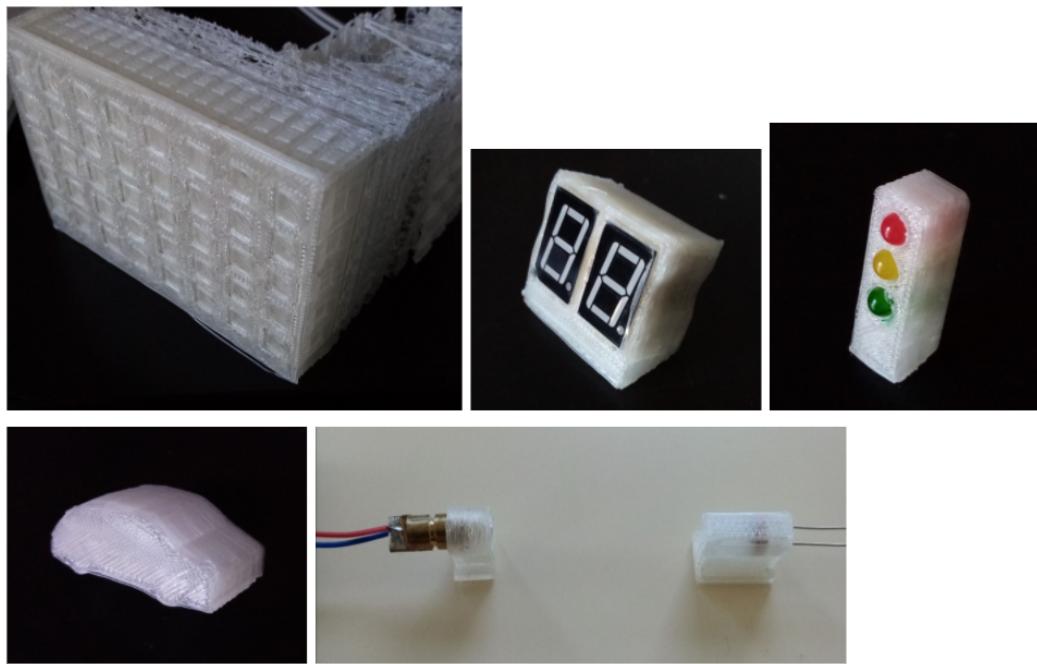


Figure 4: Impressions 3D de la maquette

Enfin tout les éléments électroniques du projet doivent être connectés à la carte STM32F407,

et ceci se fait par le biais d'un circuit imprimé, dont un visuel est donné en figure 5. Il a été réalisé par gravure chimique, à partir d'un fichier .brd conçu sur EAGLE et disponible en Annexes.

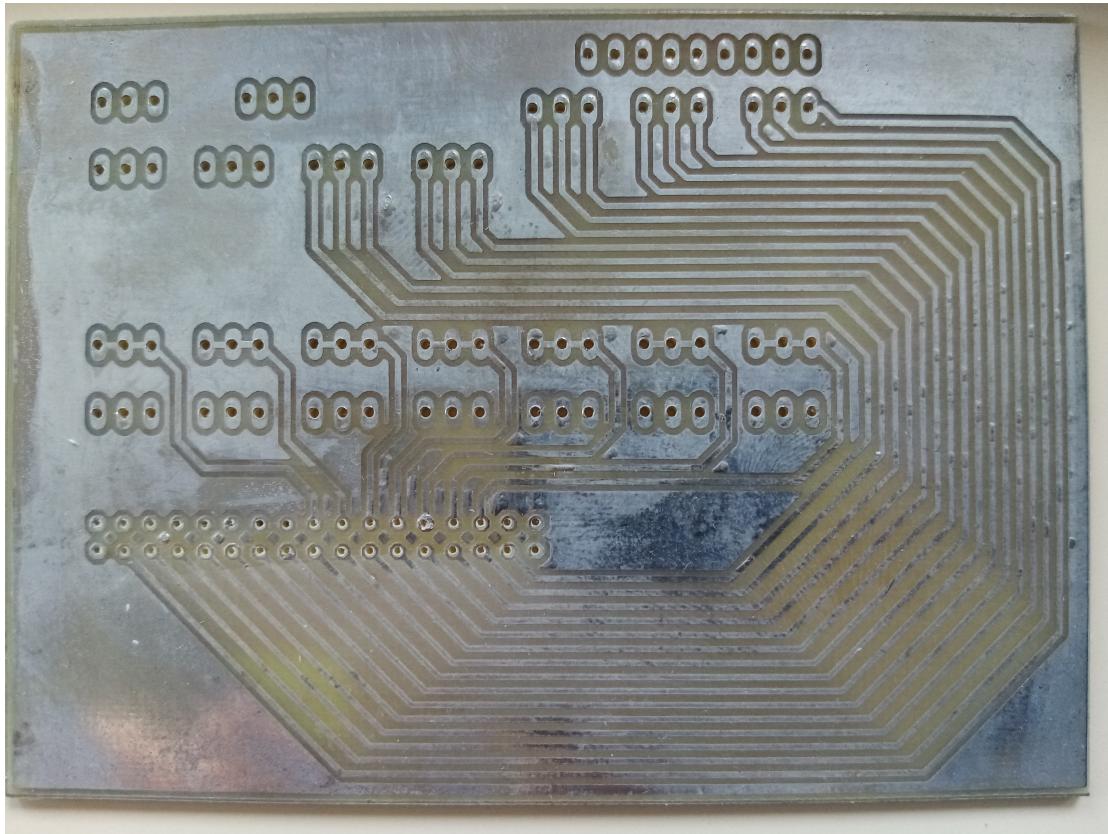


Figure 5: Circuit imprimé du projet, réalisé par gravure chimique

2.2.3 Système de détection

Solutions viables de détection

Plusieurs solutions s'offrent à nous en ce qui concerne le système de détection de véhicule. Bouton poussoir (\rightarrow Implémentation sous forme de plaque de pression), bouton poussoir manuel (Bien loin du fonctionnement réel), ou encore opto coupleur scindé (Diode LASER + LDR)

\rightarrow Notre Choix se porte sur l'opto-coupleur

Il faut que l'éclat du LASER soit important comparativement à la luminosité ambiante moyenne, autrement on ne peut pas placer un seuil de détection de luminosité tel que le dispositif fonctionne pour tous les jours de l'année sans regard de leur ensoleillement.

On pourrait alors être de lancer une phase de test : Pour quelques LDRs donnés qui correspondent aux ordres de grandeur de notre système, faisons un recensement de leurs valeurs résistives pour les luminosités minimales et maximales en particulier. (Entre un jour très

orageux et un jour d'été en plein zénith, il peut y avoir un facteur 10^3 entre les deux intensités d'ensoleillement.)

Toutes considérations faites, il est très compliqué de choisir un seuil qui fonctionnera à coup sûr. La solution est simple, nous allons nous cantonner aux LDRs que nous possérons et allons ajouter sur ceux-ci un cache en forme de tunnel ou tube, afin de minimiser l'impact des conditions extérieures. Ainsi les LDRs se trouvent dans l'obscurité tant que les faisceaux LASERs, seules sources de lumière dans leur alignement, sont coupés.

Principe du circuit de détection

D'un côté, une diode LASER alimentée de façon permanente. De l'autre, un LDR qui fait varier la valeur d'un pont résistif formé en association avec le pull-down d'une entrée de la carte, en fonction de l'éclat lumineux perçu.

Lorsque rien n'obstrue le passage de la lumière, le LDR est éclairé, sa valeur résistive est très faible, et donc l'entrée de la carte est tirée vers la tension d'alimentation ('1' logique). Lorsqu'un véhicule interfère, le LDR reste dans l'ombre, sa valeur résistive est très élevée, et donc l'entrée de la carte est tirée vers la masse ('0' logique). (Le LDR choisi présente une résistance de l'ordre du $M\Omega$ quand il n'est pas éclairé et inférieure à 100Ω en incidence directe du LASER.)

Schéma d'implémentation

La figure 6 présente le schéma d'implémentation du circuit de détection de véhicule.

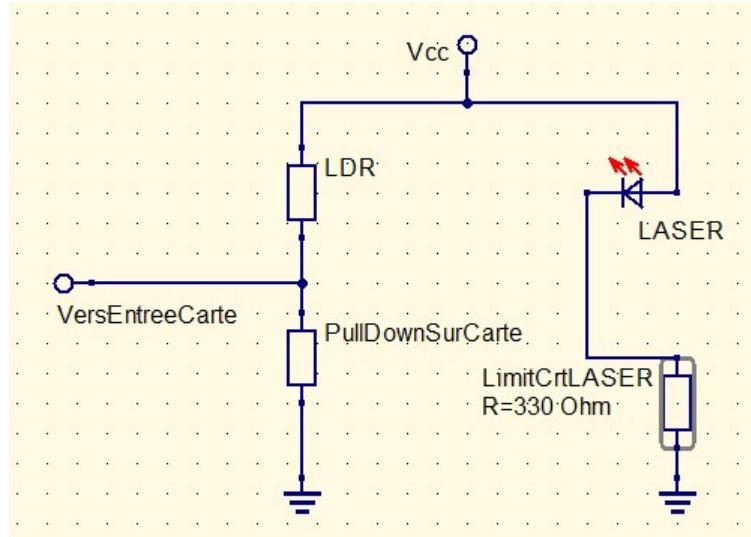


Figure 6: Circuit de détection réalisé sous Qucs, logiciel libre

Réalisation

Les socles vus dans la partie "Maquette" servent à fixer l'alignement des couples émetteurs/récepteurs,

comme on peut le voir sur la figure 7.



Figure 7: Système de détection, diode LASER à gauche, LDR à droite

2.2.4 Feux de signalisations

Descriptif

Les feux tricolores de la maquette seront simulés par un triplet de LEDs (Verte, jaune, et rouge) et celles-ci seront allumées une à une selon la gestion des cas présentée dans la figure 8. La maquette comporte trois feux tricolores, chacun positionnés à une entrée de l'intersection en "T".

Dans notre cas d'étude, il y a une route principale (La barre transversale du "T") et une secondaire (La barre verticale du "T"). Les deux feux de la route principale affichent toujours la même couleur. A chaque état donné, l'état suivant sera calculé à partir de l'état présent et de ce que détectent les capteurs de présence. Le calcul d'un nouvel état ne se fait qu'une fois la fin de la temporisation associée à l'état précédent, ceci assurant que seules les données les plus récentes soient prises en compte.

Cas de travail

Le tableau de la figure 8 récapitule les 9 cas possibles dans la gestion des couleurs affichées par les feux. (Les cases rouges et vertes du tableau récapitulent les 3 cas possibles dans l'étude simplifiée où nous n'aurions à faire qu'à des feux bicolores, verts et rouges.) Dans ce tableau, P signifie Principale et S signifie Secondaire.

Ici, la temporisation n'apparaît qu'en cas de feu rouge. Sa valeur dépend de l'encombrement des voies, mais se situe autour de 30 secondes.

Présence (P)	Présence (S)	Feux (P)	Feux (S)
OUI + était Vert	OUI, NON + était Rouge	Vert	Rouge + Tempo (30s)
OUI + était Rouge	OUI, NON + était Vert	Rouge + Tempo (5s)	Jaune + Tempo (5s)
OUI + était Rouge	OUI, NON + était Jaune	Vert	Rouge + Tempo (30s)
NON + était Vert	OUI + était Rouge	Jaune + Tempo (5s)	Rouge + Tempo (5s)
NON + était Rouge	OUI + était Vert	Rouge + Tempo (30s)	Vert
NON + était Jaune	OUI + était Rouge	Rouge + Tempo (30s)	Vert
NON + était Vert	NON + était Rouge	Vert	Rouge + Tempo (30s)
NON + était Rouge	NON + était Vert	Vert	Rouge + Tempo (5s)
NON + était Rouge	NON + était Jaune	Vert	Rouge + Tempo (30s)

Figure 8: Gestion des cas pour l'évolution de la signalisation

Le nombre de cas est largement amenuisé par les considérations suivantes :

- Le cas où un feu est vert et l'autre jaune n'existe pas, ce serait trop dangereux pour les automobilistes et le feu jaune perdrait ainsi son intérêt.
- Aussi, un feu qui était jaune ne peut devenir vert, puisque s'il était jaune, c'était pour prévenir son passage au rouge.
- Enfin, si un feu sur une voie est au vert, un feu de l'autre voie ne peut bien sûr pas l'être aussi.

Dans le programme, on nomme feuP1 et feuP2 les structures FEU (*cf. partie principe de fonctionnement*) représentant les deux feux présents sur la voie principale et feuS celle représentant celui présent sur la voie secondaire. Ces feux ont pour identifiants respectifs, dans l'ordre, 1, 2, et 3.

2.2.5 Affichage de la temporisation

Principe global

Un couple d'afficheurs sept segments est placé à chaque entrée de l'intersection en 'T', il y donc au total 3 couples, soit 6 afficheurs. Chaque couple d'afficheurs, lorsqu'il est sollicité, doit afficher une durée d'attente (en secondes) qui évolue, d'une valeur de départ choisie (ex : 50) jusqu'à 0. Cette durée correspond à un temps d'attente au feu rouge sur le feu tricolore correspondant parmi les trois de l'intersection.

Fonctionnement de l'algorithme

→ La fonction temporisation appelle en argument la structure représentant un des feux et la durée à patienter. Cette structure contient entre autres un champ int 'cpt_d' et un champ int 'cpt_u' pour l'afficheur des dizaines et des unités respectivement. Cette fonction comporte deux boucles imbriquées : une pour l'évolution des dizaines, et l'autre pour celle des unités. Chacune de ces deux boucles fait varier l'état du compteur 'cpt_x' parmi les dix états codés.

→ On décrémente le nombre unité en faisant passer l'objet 'cpt_u' de 'etat9' à 'etat0', état par

état, après temporisation (de 1 seconde) à chaque itération, puis le nombre dizaine décrémente d'une unité en faisant passer l'objet 'cpt_d' à l'état inférieur, et ainsi de suite jusqu'à ce que 'cpt_d' et 'cpt_u' soit tous deux rendus à 'etat0'. (ex : Début à cpt_d = 'etat4' pour une temporisation de 50 secondes, on couvre alors au total 50 couples : De 'etat4'; 'etat9' à 'etat0'; 'etat0'.)

→ La déclaration des différents états (qui correspondent aux dix nombres du système décimal) est faite dans un fichier d'entête. Ces 10 états, "ZERO" à "NEUF", correspondent à un code décimal qui sera lu par le programme pour commander l'allumage de tel ou tel segment des afficheurs concernés, pour afficher le nombre souhaité.

Fonction afficher(FEU*)

→ La fonction afficher observe l'état des objets 'cpt_d' et 'cpt_u' de la structure feu, les lit et les décode afin de pouvoir envoyer du courant aux segments (passer leur anode à l'état haut (afficheurs à cathode commune)) des afficheurs à alimenter subconséquemment. Le couple d'afficheurs d'intérêt est sélectionné en passant leur cathode à l'état bas. 20 broches sont nécessaires pour contrôler l'affichage des 6 afficheurs 7 segments : 10 pour les unités (3 cathodes communes et 7 anodes regroupées) et autant pour les dizaines.

Algorithme Temporisation

Début Temporisation(feu, durée) :

Pour i de durée[1]-1 à 0 :	<i>Boucle des dizaines</i>
feu[cpt_d] = etati;	
Pour j de 9 à 0 :	<i>Boucle des unités</i>
feu[cpt_u] = etatj;	
afficher(feu);	<i>Envoi du courant de la carte aux segments</i>
patiente();	<i>On patiente une seconde</i>

Fin Temporisation

Cas d'utilisation

Lorsque le programme doit signifier un temps d'attente à un feu (au rouge) de l'intersection, il appelle la fonction temporisation() prenant en argument la structure correspondante audit feu et modifie la valeur de ses champs cpt_d et cpt_u à chaque seconde qui passe. La fonction afficher() permet alors d'afficher au fur et à mesure les nombres correspondants au temps restant (en seconde) sur les deux afficheurs de l'intersection concernés par cette temporisation. C'est la fonction patiente() qui se charge de périodiser à une seconde le changement de l'affichage. Cette dernière fonction prend en argument un entier et génère un délai lui étant proportionnel (donc non fixé à une seconde), de façon à pouvoir tester les limites du système, mais cela sera abordé plus après dans la section améliorations.

Architecture matérielle

Voici comment s'organisent les différents afficheurs et leur commande. Si l'on numérote leurs segments de 0 à 6, chaque état (ZERO, UN, DEUX, TROIS, QUATRE, CINQ, SIX, SEPT, HUIT, NEUF) représente un chiffre de 7 digits en décimal, dont chaque digit est la commande (0 ou 1) à appliquer au segment dont le numéro correspond à sa place dans le nombre, en commençant de 0 à droite et en finissant à 6 à gauche. La figure 9 représente les segments physiques numérotés en concordance avec le digit auquel ils correspondent dans les codes ZERO, UN, DEUX, TROIS, QUATRE, CINQ, SIX, SEPT, HUIT et NEUF, définis ci-après comme dans une entête du projet en C :

ZERO : 111111 ; UN : 110 ; DEUX : 1101101 ; TROIS : 1001111 ; QUATRE : 1010110 ;
CINQ : 1011011 ; SIX : 1111011 ; SEPT : 1110 ; HUIT : 1111111 ; NEUF : 1011111

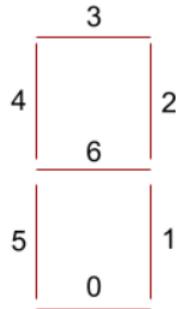


Figure 9: Numérotation des segments des afficheurs

3 Conclusions

3.1 Résultats

L'initialisation du microprocesseur reste incertaine. Toutefois, le réseau a été intégralement codé. Nous avons également pu imprimer les quelques pièces de la maquette (support des capteurs, feux tricolores et afficheurs, ainsi que des véhicules et des bâtiments venant altérer la visibilité de leur conducteurs). Nous avons également réaliser le circuit imprimé de liaison entre la carte de prototypage et tout les composants. Il nous aurait manqué au moins une séance de 4h pour souder tout les pin-headers au circuit imprimé, brancher les $\approx 100+$ câbles arrivant (x34) au circuit et en sortant (x78). Il n'aurait plus manqué qu'à compiler tout le code et espérer que cela fonctionne comme prévu.

3.2 Problèmes rencontrés

Le projet n'a pas pu aboutir (compilation de l'ensemble du programme, Liaison de tout les éléments de la maquette) faute de mauvaise gestion du temps. Nous avons notamment rencontré des difficultés lors du recensement des ports GPIOs disponibles sur la carte et de l'impression 3D des pièces.

3.3 Coûts

Le coût de la maquette avoisine 15 € (environ 14 € pour les 6 afficheurs 7 segments, et 1€ pour les 9 LEDs). Ceci sans compter biensûr le prix des différents matériels et services obtenus grâce à l'ENSEA (carte de développement STM32F407 et gravure du circuit imprimé entre autres).

3.4 Conclusion

La définition des critères et niveaux du Cahier des Charges Fonctionnel, de l'architecture du réseau Light Traffic, ainsi que l'écriture de son code s'est effectuée en respectant le planning. Toutefois, le recensement des ports GPIOs disponibles a pris plus de temps que prévu initialement. Avec plus de temps à disposition, il aurait été pertinent de s'orienter vers un modèle plus complet pouvant s'adapter à tous types d'usagers (piétons, cyclistes ...). Enfin pourquoi ne pas prévoir d'alimenter le système avec des sources renouvelables tel que le solaire ou l'éolien, à l'échelle de notre maquette ?

De plus, la fonction patiente() fonctionne sur la base de la miliseconde, on pourrait donc prévoir un code légèrement modifié et plus d'afficheurs pour transformer cette maquette en jeu de vitesse.

Le projet fut stimulant et accessible concernant sa réalisation (compte-tenu des moyens qui nous sont fournis). Une meilleure gestion du temps nous aurait permis de l'achever dans les délais impartis. Toutefois, cette expérience nous aura permis d'approfondir et de renforcer notre maîtrise du langage C et d'améliorer notre prise en main des logiciels KeilVision et EAGLE.

4 Annexes

Ici se trouve un lien vers les codes du projet regroupés sur un Github, ainsi que le fichier .brd réalisé sous EAGLE visible en figure 10.

Lien vers le GitHub du projet Light Traffic :

https://github.com/AdrienLenoir14159/Traffic_Lights_Model

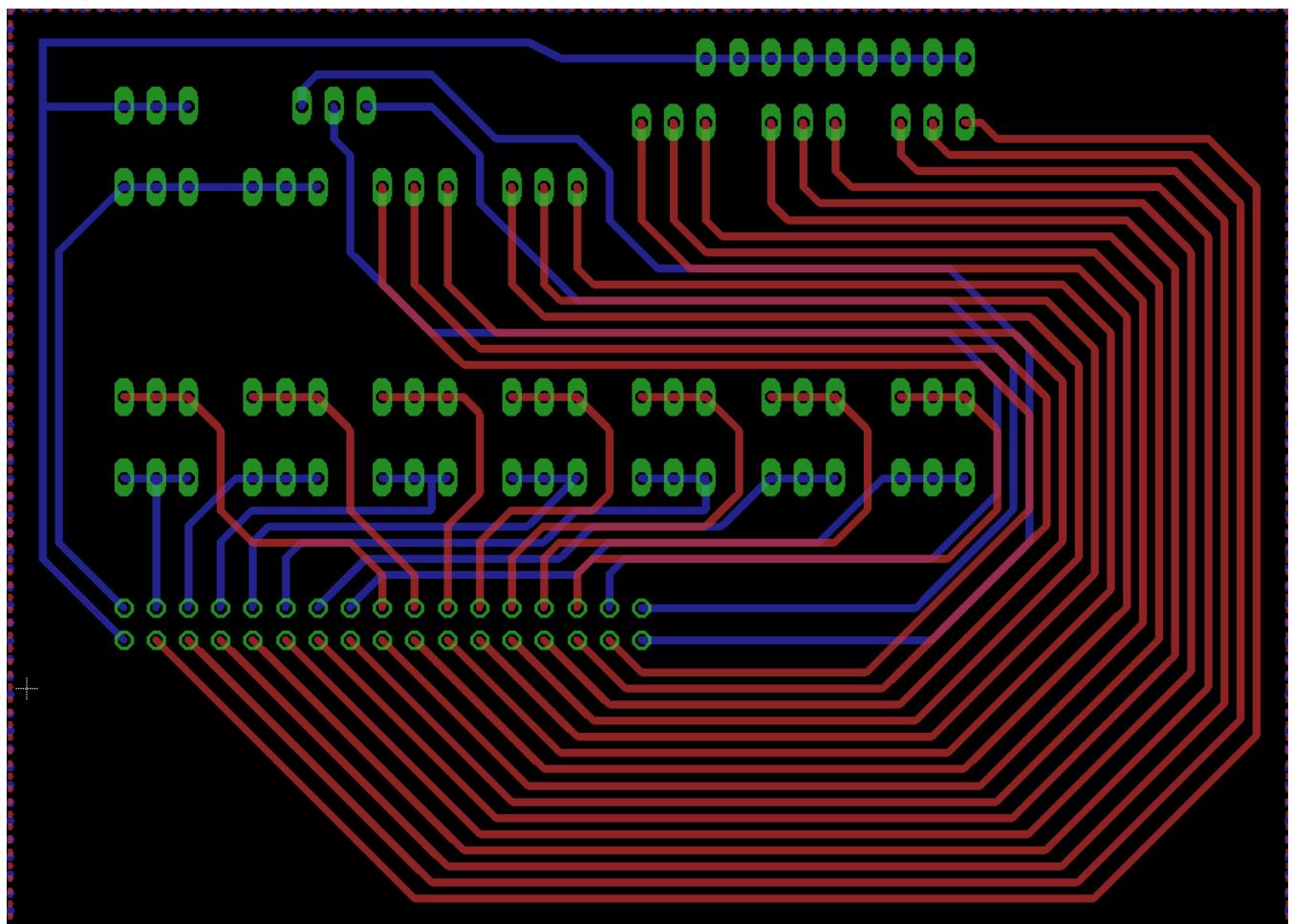


Figure 10: Board du projet, réalisé sous EAGLE