



**Travail de Bachelor**  
**2019 - 2020**

**Filière Informatique**

---

# **Accélérateur de tour télécom<sup>1</sup>**

---

**Rapport (*v1, 10.07.2020*)**

---

**Nicolas Maier<sup>2</sup>**

---

Superviseurs :           **Jacques Supcik<sup>3</sup>**  
                                 **Michael Mäder<sup>4</sup>**

Expert :                   **Frédéric Mauron<sup>5</sup>**

**Hes·so**  
Haute Ecole Spécialisée  
de Suisse occidentale  
Fachhochschule Westschweiz

---

1. [https://gitlab.forge.hefr.ch/nicolas.maier/tb\\_nm](https://gitlab.forge.hefr.ch/nicolas.maier/tb_nm)

2. nicolas.maier@edu.hefr.ch

3. jacques.supcik@hefr.ch

4. michael.maeder@hefr.ch

5. frederic.mauron@ftth-fr.ch

## Table des versions

<b>Version</b>	<b>Date de publication</b>	<b>Auteur</b>	<b>Description</b>
1.0	10.07.2020	Nicolas Maier	Premier rendu partiel du rapport pour juger la structure et le style

# Table des matières

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Acteurs . . . . .	1
1.2	Contexte . . . . .	2
1.3	Situation avant le projet . . . . .	2
1.4	Buts du projet . . . . .	3
1.4.1	Objectifs du projet . . . . .	3
1.4.2	Contraintes . . . . .	4
1.5	Méthode de travail . . . . .	4
1.5.1	Cahier des charges . . . . .	4
1.5.2	Planification initiale . . . . .	5
1.6	Structure du rapport . . . . .	6
<b>2</b>	<b>Analyse</b>	<b>7</b>
2.1	NeoPixel, contrôleurs ws2812b . . . . .	7
2.1.1	Organisation en série . . . . .	8
2.1.2	Signal (protocole) . . . . .	8
2.2	Choix du matériel . . . . .	10
2.2.1	ESP32 DEVKIT DOIT . . . . .	10
2.2.2	Sipeed Maix Bit . . . . .	10
2.2.3	BeagleBone Black . . . . .	11
2.2.4	Blue Pill (STM32F103C8T6) . . . . .	11
2.2.5	Évaluation et choix . . . . .	12
<b>3</b>	<b>Spécification</b>	<b>13</b>
3.1	Organisation physique . . . . .	13
3.2	Interface entre le Raspberry Pi et le Blue Pill . . . . .	14
3.2.1	Fichier "control" . . . . .	15
3.2.2	Fichier "data" . . . . .	15
3.3	Interface de la librairie (Raspberry Pi) . . . . .	15
<b>4</b>	<b>Conception</b>	<b>16</b>
4.1	Contrôle des LEDs . . . . .	16
4.2	Interfaçage USB . . . . .	16

4.3	Synchronisation	16
4.4	Envoi des valeurs côté Raspberry Pi	16
<b>5</b>	<b>Implémentation</b>	<b>17</b>
5.1	Environnement de développement	17
5.2	Programme embarqué	17
5.2.1	Librairies	17
5.3	Interface (librairie) côté Raspberry Pi	17
<b>6</b>	<b>Tests et validation</b>	<b>18</b>
6.1	Tests de fonctionnalité	18
6.2	Tests de performances	18
<b>7</b>	<b>Améliorations et travaux futurs</b>	<b>19</b>
7.1	Interfaçage USB pour d'autres périphériques	19
7.2	Extension de la capacité de la tour télécom	19
<b>8</b>	<b>Mise en place et exemple d'utilisation</b>	<b>20</b>
<b>9</b>	<b>Conclusion</b>	<b>21</b>
9.1	Comparaison avec les objectifs	21
9.2	Rétrospective sur la planification initiale	21
9.3	Conclusion personnelle	21
<b>10</b>	<b>Déclaration d'honneur</b>	<b>22</b>

# Chapitre 1

## Introduction

Depuis plusieurs années, un projet dénommé "Tour télécom" est développé au sein de la filière Informatique et Télécommunications. L'objectif de ce projet est de créer et utiliser une maquette de tour interactive afin de faire la promotion de l'informatique et des systèmes de communication auprès du public lors des manifestations telles que les portes ouvertes.

La tour télécom (figure 1.1) comporte une matrice de LEDs flexible permettant d'afficher du texte et des animations. C'est sur cet aspect de la tour que ce projet porte. Le but du projet est de permettre d'augmenter la vitesse d'affichage (nombre d'images par secondes) de la matrice de LED, ainsi que de permettre d'augmenter le nombre de LEDs présentes sur la tour.



FIGURE 1.1 – Photo de la tour télécom

### 1.1 Acteurs

Ce projet est suivi par les personnes suivantes :

- Jacques Supcik, *Superviseur*
- Michael Mäder, *Superviseur*
- Frédéric Mauron, *Expert*
- Nicolas Maier, *Étudiant*

## 1.2 Contexte

Les matrices de LEDs présentes sur la tour sont composées de contrôleurs ws2812b. Chacun de ces contrôleurs gère un pixel RGB. Ces contrôleurs fonctionnent en série, il suffit d'envoyer les données de toute la matrice de LEDs au premier contrôleur de la chaîne, qui va transmettre les informations aux suivants (une description détaillée du fonctionnement des contrôleurs ws2812b est disponible dans le chapitre 2.1).

Ce système de LEDs est simple et extensible, mais demande toutefois un timing très précis pour transmettre des données, c'est pourquoi un système temps réel avec des contraintes de temps de l'ordre de 150ns est nécessaire pour contrôler les matrices de LEDs.

## 1.3 Situation avant le projet

Actuellement, le système d'affichage de la tour télécom est composé d'un Raspberry Pi qui contrôle une matrice de 4 bandes de 32x8 pixels RGB (Adafruit "Flexible 8x32 NeoPixel RGB LED Matrix"<sup>1</sup>). Pour contrôler ces LEDs, le Raspberry Pi doit combiner un DMA et un PWM afin de générer un signal ayant un timing très précis, ce qui ne serait pas possible directement en contrôlant les GPIOs car le système Linux ne permettrait pas de garantir cette contrainte de temps réel avec la précision requise.

Les 4 bandes de 32x8 pixels sont branchées en série, le Raspberry Pi contrôle donc 1024 pixels comme indiqué sur la figure 1.2. Chaque pixel attend 24 bits afin d'obtenir une couleur RGB où chacune des 3 intensités est encodée sur 8 bits. Selon le protocole défini dans la datasheet du ws2812b<sup>2</sup>, les données doivent être transmises à 800 Kb/s. La solution actuelle permet donc d'afficher environ 30 images par seconde.

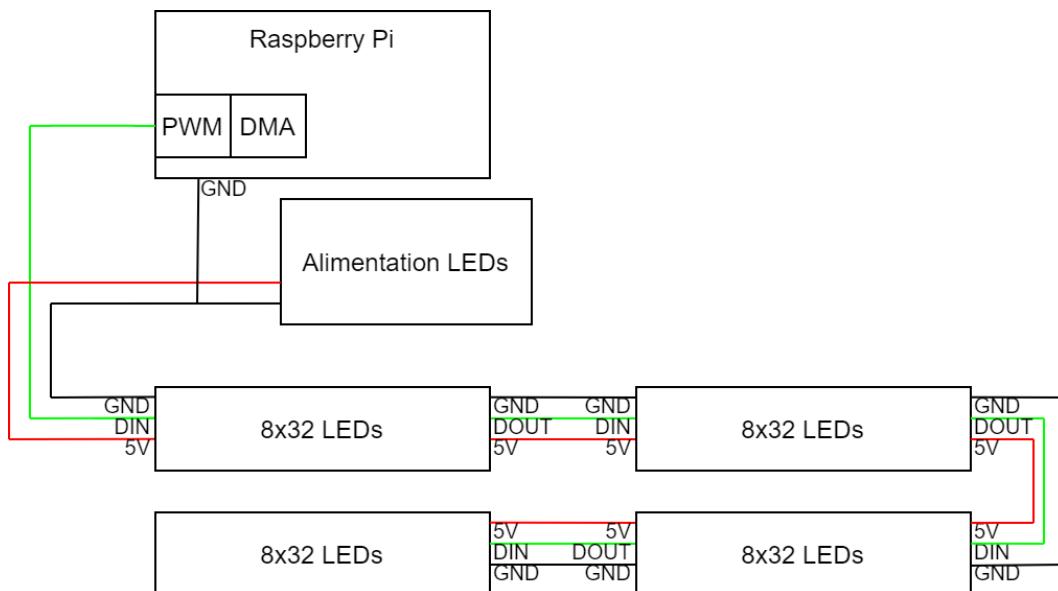


FIGURE 1.2 – Situation avant le projet

Cette solution est limitée en nombre de LEDs ou en images par seconde. En ajoutant des LEDs en série, on serait obligé de réduire le nombre d'images par seconde, et inversément.

1. <https://www.adafruit.com/product/2294>

2. [https://voltiq.ru/datasheets/WS2812B\\_datasheet\\_EN.pdf](https://voltiq.ru/datasheets/WS2812B_datasheet_EN.pdf)

## 1.4 Buts du projet

Le but principal de ce projet est d'ajouter un "coprocesseur" externe au Raspberry Pi qui s'occuperait du travail temps réel de contrôle des LEDs (génération du signal), afin de pouvoir augmenter la vitesse d'affichage (nombre d'images par secondes) de la tour télécom, ainsi que de permettre d'augmenter le nombre de LEDs présentes sur la tour, tout en déchargeant cette tâche du Raspberry Pi.

### 1.4.1 Objectifs du projet

L'objectif est de développer une solution comportant un microcontrôleur connecté au Raspberry Pi, comme indiqué sur la figure 1.3. Le microcontrôleur reçoit les informations (images à afficher) du Raspberry Pi, et s'occupe de générer le signal correspondant afin de contrôler plusieurs bandes NeoPixel en parallèle.

- Étude des microcontrôleurs intéressants dans le contexte du projet
- Étude de l'interfaçage entre le Raspberry Pi et le microcontrôleur
- Choix du meilleur microcontrôleur pour ce projet
- Implémentation d'un système capable de contrôler au minimum 4 bandes NeoPixel en parallèle
- Étude, description et justification des limites théoriques de la solution réalisée
- Tests et validation du système

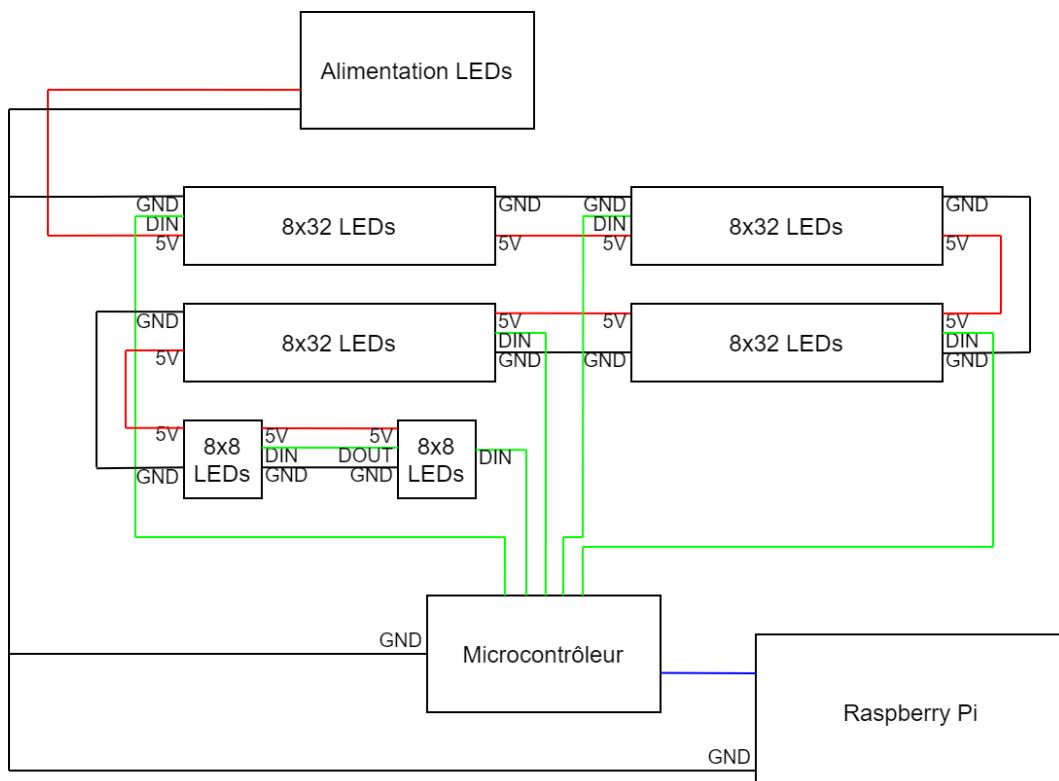


FIGURE 1.3 – Objectif du projet

### 1.4.2 Contraintes

Les contraintes suivantes doivent être respectées pour ce projet :

- Un minimum de 4 bandes de LEDs doivent pouvoir être contrôlées en parallèle, et une cinquième serait utile mais pas absolument nécessaire (le but serait de compléter le tour complet de la tour télécom avec deux carrés de 8x8 pixels<sup>3</sup>)
- Le système réalisé doit être simple à mettre en oeuvre (par exemple, l'idéal serait de pouvoir simplement brancher la tour télécom en USB au Raspberry Pi, de façon "plug-and-play")
- L'interface fournie pour contrôler les LEDs doit être inspirée de celle qui est actuellement utilisée<sup>4</sup>

## 1.5 Méthode de travail

Durant ce travail de Bachelor, il est important d'avoir un bon suivi tout au long de sa réalisation. C'est pourquoi des séances de projet ont eu lieu chaque semaine, auxquelles l'étudiant et les superviseurs ont participé. Deux séances supplémentaires, avec l'expert du projet et l'étudiant, ont également eu lieu. Chacune de ces séances a été résumée dans un procès-verbal<sup>5</sup>.

### 1.5.1 Cahier des charges

Un cahier des charges a été rédigé par l'étudiant et validé par les superviseurs après l'analyse des besoins et des objectifs du projet. Ce cahier des charges a permis de définir précisément le but et les contraintes du projet. Une rétrospective sur l'atteinte des objectifs se trouve dans le chapitre TODO.

3. <https://www.adafruit.com/product/1487>

4. [https://github.com/jgarff/rpi\\_ws281x](https://github.com/jgarff/rpi_ws281x)

5. [https://gitlab.forge.hefr.ch/nicolas.maier/tb\\_nm/-/tree/master/documentation/PVs](https://gitlab.forge.hefr.ch/nicolas.maier/tb_nm/-/tree/master/documentation/PVs)

### 1.5.2 Planification initiale

Voici la planification prévue au début du projet (figure 1.4) :

	P13	P14	P15	P16	P17	P18	P19	P20	P21
<b>Documentation</b>									
Cahier des charges									
<i>Validation cahier des charges</i>			Ve						
Rapport technique									
Manuel d'utilisation									
<i>Rendu du rapport</i>									Ve
<b>Analyse</b>									
Étude des bandes de LEDs NeoPixel									
Étude des microcontrôleurs									
Étude de l'interfaçage entre le RPi et le(s) µp(s)									
<i>Choix du matériel</i>				Je					
<b>Spécification</b>									
Spécification de l'interface entre le RPi et le(s) µp(s)									
<i>Validation de la spécification</i>									
<b>Conception</b>									
Définition de l'architecture, diagrammes correspondants									
<i>Validation de la conception et choix des technologies</i>									
<b>Réalisation</b>									
Implémentation du contrôle des bandes NeoPixel									
Implémentation de l'interface RPi/µp(s)									
<b>Tests</b>									
Tests appropriés									
<i>Validation de la réalisation</i>									Me
<i>Défense orale</i>									

(Semaine des examens)

FIGURE 1.4 – Planification du projet

Une rétrospective sur la planification et comment le projet s'est déroulé se trouve dans le chapitre TODO.

## 1.6 Structure du rapport

Ce rapport est séparé en 10 chapitres. Ces chapitres contiennent des sections, dans lesquelles sont expliquées les thèmes en détail. La liste ci-dessous contient une description de chaque chapitre :

— **Introduction**

Explique le contexte du projet, les buts à atteindre et son déroulement

— **Analyse**

Définit les besoins du système, et documente les choix technologiques réalisés durant le projet

— **Spécification**

Spécifie l'organisation physique des éléments du système ainsi que les protocoles utilisés entre ces éléments

— **Conception**

Indique la structure de l'implémentation, ainsi que les techniques qui seront utilisées afin de réaliser les fonctionnalités du projet

— **Implémentation**

Détaille les éléments importants de l'implémentation des différentes parties du projet

— **Tests et validation**

Indique quels tests ont été effectués pour vérifier le fonctionnement et les performances du système, ainsi que les résultats de ces derniers

— **Améliorations et travaux futurs**

Contient une description des suites ou dérivés possibles du projet

— **Mise en place et exemple d'utilisation**

TODO

— **Conclusion**

Donne une rétrospective sur différents aspects du projet, ainsi qu'une conclusion personnelle

— **Déclaration d'honneur**

Contient la déclaration d'honneur signée

## Chapitre 2

# Analyse

Ce chapitre décrit l'étude des différents besoins du système, et des différentes technologies et matériel choisies pour réaliser le projet.

### 2.1 NeoPixel, contrôleurs ws2812b

Dans les matrices de LEDs NeoPixel, chaque pixel est composé d'un contrôleur ws2812b capable de gérer l'intensité individuelle de 3 couleurs (rouge, vert et bleu). Les informations détaillées à propos de ce contrôleur sont disponibles dans leur datasheet<sup>1</sup>.

Voici une photo montrant en détail une partie d'une matrice NeoPixel, dans laquelle on voit plusieurs contrôleurs ws2812b sur un circuit imprimé flexible (figure 2.1)

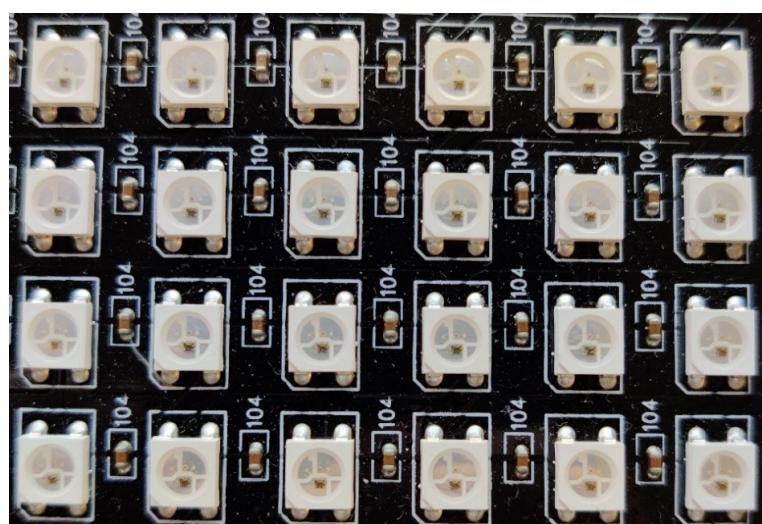


FIGURE 2.1 – Détails d'une matrice NeoPixel

1. [https://voltiq.ru/datasheets/WS2812B\\_datasheet\\_EN.pdf](https://voltiq.ru/datasheets/WS2812B_datasheet_EN.pdf)

### 2.1.1 Organisation en série

Ces contrôleurs sont conçus pour fonctionner avec autant de pixels que nécessaire : ils sont organisés en chaîne. Pour ajouter un pixel à une bande, il suffirait d'ajouter un contrôleur ws2812b à la chaîne. Cette organisation en cascade offre une grande flexibilité sur le nombre de pixels et leur organisation, tout en permettant d'adresser chaque LED individuellement.

Pour contrôler la couleur des pixels, un microcontrôleur doit envoyer les valeurs de chaque couleur de chaque LED (24 bits par pixel RGB) au premier contrôleur ws2812b de la chaîne. Chacun des contrôleurs lit et affiche la première valeur qu'il (24 bits) qu'il reçoit, et transmet toutes les autres au contrôleur ws2812b suivant. Cela permet de donner la couleur désirée à chacun des pixels, tout en n'ayant qu'à brancher une seule ligne de données pour toute la bande de LEDs. Lorsque le microcontrôleur a envoyé toutes les valeurs désirées, il doit envoyer un signal particulier ("reset code"). Ce signal indique à chaque contrôleur ws2812b qu'il devra à nouveau prendre une valeur, dans le but d'afficher l'image suivante.

Voici un schéma montrant comment les contrôleurs ws2812b doivent être câblés entre eux pour fonctionner en série (figure 2.2) :

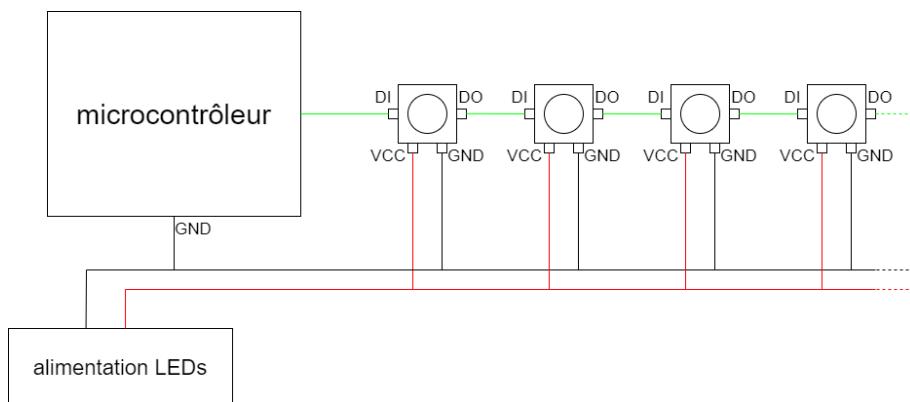


FIGURE 2.2 – Schéma du câblage des contrôleurs ws2812b

### 2.1.2 Signal (protocole)

Le protocole définit dans la datasheet des contrôleurs ws2812b est simple, mais nécessite une bonne précision dans le temps. Il s'agit d'un signal carré alternant entre haut et bas avec une période de  $1.25\mu\text{s}$  (fréquence de 800KHz), dont le rapport cyclique (temps à l'état haut) est modulé pour définir l'information envoyée.

Au début de la période, le signal est à "haut".

- Si on veut envoyer le bit "0", le signal passe à "bas" après  $0.4\mu\text{s}$
- Si on veut envoyer le bit "1", le signal passe à "bas" après  $0.85\mu\text{s}$

D'après la datasheet, ces temps doivent être précis à 150ns près.

Chaque contrôleur ws2812b consomme 24 bits et passe le reste du signal au contrôleur suivant. Ces 24 bits sont composés de l'information suivante (figure 2.3, à lire de gauche à droite) :

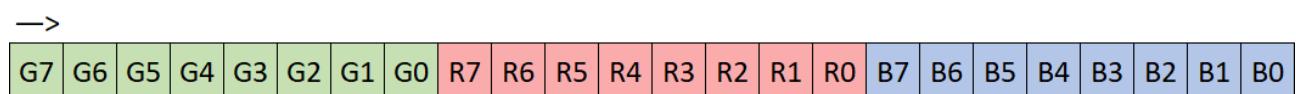


FIGURE 2.3 – Organisation des bits codant les couleurs

Les MSB des codes de couleurs sont envoyés avant les LSB. Chaque couleur est encodée sur 8 bit, et elles sont envoyées dans l'ordre GRB.

Le "reset code", permettant d'indiquer aux contrôleurs ws2812b qu'une nouvelle image va être envoyée, est un signal "bas" durant plus de  $50\mu s$ .

Voici un schéma résumant les 3 signaux différents permettant de contrôler les pixels (figure 2.4) :

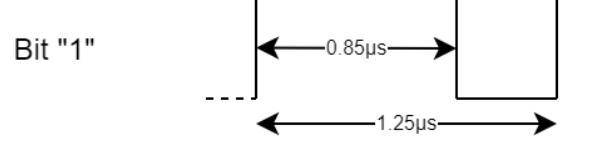
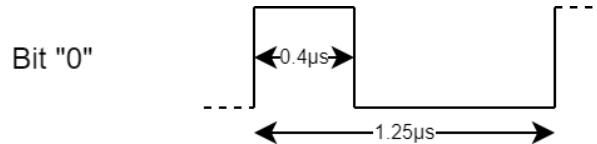


FIGURE 2.4 – Signaux permettant de contrôler les pixels

## 2.2 Choix du matériel

Pour réaliser les objectifs du travail, un microcontrôleur a été choisi. Voici un résumé des caractéristiques de différents microcontrôleurs intéressants retenus pour le projet :

### 2.2.1 ESP32 DEVKIT DOIT

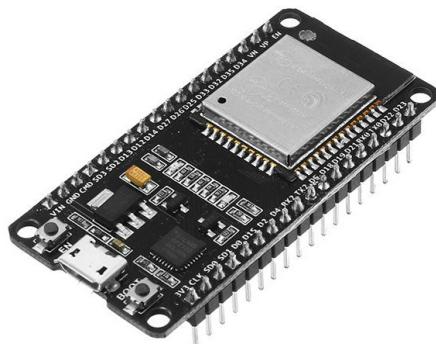


FIGURE 2.5 – ESP32 (source : [www.banggood.com](http://www.banggood.com))

Ce microcontrôleur est très populaire. Il propose de très bonnes caractéristiques tout en étant disponible pour un prix d'environ 10 CHF<sup>2</sup>.

Il comporte un processeur Xtensa dual-core cadencé à 240 MHz, ainsi que 500KB de RAM. Il possède un module wifi capable de transmettre environ 20Mbps<sup>3</sup>.

### 2.2.2 Sipeed Maix Bit

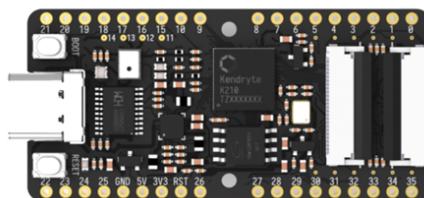


FIGURE 2.6 – Sipeed maix bit (source : [www.seeedstudio.com](http://www.seeedstudio.com))

Ce microcontrôleur moderne a la particularité d'être équipé d'un Kendryte K210 dual core, basé sur l'architecture RISC-V. Il possède 8MB de RAM, et son processeur est cadencé à 400MHz. Il est disponible pour un prix d'environ 16 CHF<sup>4</sup>.

Il comporte un port USB type C, qui peut être utilisé pour le programmer et pour transmettre des données entre l'hôte et le programme embarqué. Il comporte un chip USB-UART supportant un débit maximum de 2Mbps.

- 2. <https://www.banggood.com/ESP32-Development-Board...>
- 3. <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-guides/wifi.html#esp32-wi-fi-throughput>
- 4. <https://www.distrelec.ch/en/sipeed-maix-bit-for-risc-ai-iot-seeed-studio-102991150/p/30135127>

### 2.2.3 BeagleBone Black

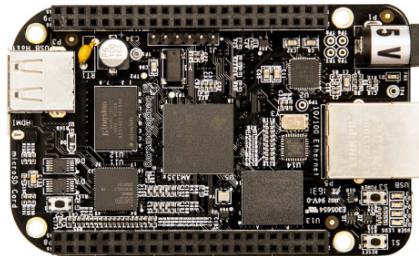


FIGURE 2.7 – BeagleBone Black (source : [www.element14.com](http://www.element14.com))

Il s'agit d'un "single-board computer" capable de faire tourner Linux. Il est disponible pour un prix d'environ 70 CHF<sup>5</sup>. Il est équipé d'un processeur mono cœur cadencé à 1GHz, ainsi que de 512MB de RAM.

Il a la particularité de comporter, en plus de son processeur principal, deux co-processeurs ("PRU") cadencés à 200MHz et capables d'interagir avec la mémoire principale ainsi que les composants du BeagleBone Black. Bien que le processeur principal fasse tourner Linux, un des PRU pourrait s'occuper de générer le signal des LEDs avec des contraintes temps réel.

Il comporte une interface Ethernet 100Mbps.

### 2.2.4 Blue Pill (STM32F103C8T6)



FIGURE 2.8 – Blue Pill (source : [stm32-base.org](http://stm32-base.org))

Cette carte est équipée d'un STM32F103C8T6. Il s'agit d'un MCU Cortex-M3, cadencé à 72MHz et proposant 20KB de RAM. Il est disponible pour un prix d'environ 3 CHF<sup>6</sup>.

Bien qu'extrêmement bon marché, ce microcontrôleur propose des avantages intéressants. Il possède une interface USB très flexible : il peut être vu par l'hôte comme différentes classes, et on peut même implémenter notre propre stack USB. Cette flexibilité permet au Blue Pill d'apparaître par exemple comme un périphérique de stockage auprès de l'hôte.

Étant donné qu'il fait partie de la famille des STM32, il est facilement interchangeable avec d'autres membres de cette famille. En effet, l'API permettant de contrôler les différents périphériques étant commune entre ces MCU, la majeure partie du code peut être gardée si il est nécessaire de passer à un MCU ayant plus de RAM par exemple.

5. <https://www.banggood.com/Embest-BeagleBone-BB-Black-Cortex-A8-Development-Board-REV-C-Version-p-954565.html>  
6. <https://www.banggood.com/STM32F103C8T6-...>

## 2.2.5 Évaluation et choix

Les microcontrôleurs présentés ci-dessus ont été évalués selon différents critères :

- Prix : Prix à l'achat du microcontrôleur
- Capacité (RAM) : Quantité de mémoire RAM disponible
- Popularité (moins de risque) : Facilité à trouver des informations en ligne, ancienneté du microcontrôleur
- Interface (vitesse) : Vitesse disponible pour le transfert d'informations entre le Raspberry Pi et le microcontrôleur
- Facilité à mettre en oeuvre : Facilité à connecter le Raspberry Pi au microcontrôleur et à envoyer les informations des images à afficher

Un poids de 1 à 5 a été donné à chaque critère (1 : le moins important, 5 : le plus important). Chaque microcontrôleur présenté a ensuite été évalué selon chacun des critères avec une note de 1 à 5, afin d'obtenir un total permettant d'indiquer si il est adapté au projet (figure 2.9) :

Poids	2	2	2	4	5	
Critères	Prix	Capacité (RAM)	Popularité (moins de risque)	Interface (vitesse)	Facilité à mettre en œuvre	Total
Sipeed Maix Bit	3	4	1	2	3	<b>39</b>
ESP32 DEVKIT DOIT	4	3	4	4	1	<b>43</b>
BeagleBone Black	1	5	4	5	4	<b>60</b>
Blue Pill	5	2	3	4	5	<b>61</b>

FIGURE 2.9 – Tableau multicritère

En prenant tous ces éléments en compte, le Blue Pill a été choisi pour réaliser ce projet. La flexibilité de son interface USB et son bas prix permettra au produit réalisé d'être très accessible pour quiconque voulant contrôler des LEDs NeoPixel.

## Chapitre 3

# Spécification

Ce chapitre spécifie l'organisation physique des éléments du système, ainsi que les interactions entre les différentes parties du système (protocoles et interfaces).

### 3.1 Organisation physique

Le système est composé de 3 éléments physiques principaux différents :

- Le Raspberry Pi : Il génère les images à afficher et les envoie au microcontrôleur en suivant le protocole défini ci-dessous
- Le Blue Pill : Il reçoit les images à afficher du Raspberry Pi et génère les signaux correspondant pour contrôler jusqu'à 8 bandes de LEDs NeoPixel en parallèle
- Les bandes de LEDs : Ce sont les bandes de contrôleurs ws2812b connectés en série

Voici un schéma (figure 3.1) montrant l'organisation physique de ces éléments, en omettant l'alimentation et le chip CD4050B permettant de convertir les signaux 3.3V en 5V (un schéma montrant le câblage complet à réaliser lors de la mise en place est disponible dans le chapitre TODO) :

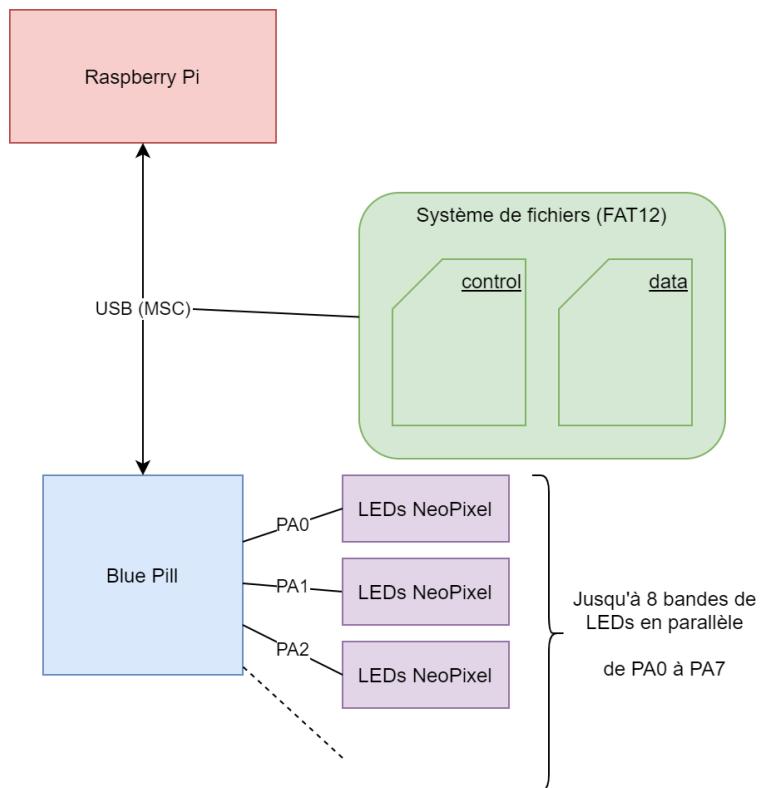


FIGURE 3.1 – Organisation physique du système

Le Raspberry Pi est connecté par USB au Blue Pill. Le Blue Pill est connecté aux différentes bandes de LEDs en parallèle sur les pins 0 à 7 de la banque de GPIOs A (PA0, PA1, ...).

## 3.2 Interface entre le Raspberry Pi et le Blue Pill

Le Blue Pill est configuré pour s'annoncer comme un périphérique de stockage USB (MSC). Il présente un système de fichiers FAT12 avec des blocs de 512 bytes contenant deux fichiers : "control" et "data".

- Le fichier "control" permet de configurer le système (nombre de LEDs, réduction des données envoyées, synchronisation)
- Le fichier "data" permet d'envoyer les données des LEDs à afficher

Le Raspberry Pi peut écrire dans ces deux fichiers par blocs de 512 bytes, pour obtenir le comportement désiré :

### 3.2.1 Fichier "control"

Le fichier "control" a une taille de 1 bloc (512 bytes). Ils ont la fonction suivante :

- byte 0 (uint8\_t) : synchronisation de l'affichage (lorsque cette valeur est modifiée, l'opération "write" est bloquante jusqu'à ce que l'image précédente ait fini de s'afficher et la prochaine image est envoyée aux bandes de LEDs, plus de détails dans la section TODO)
- byte 1 (uint8\_t) : mode de réduction des données (0 signifie aucune réduction, 1 signifie que seuls les 4 MSB des codes de couleurs sont envoyés, réduisant de moitié le temps du transfert d'une image et doublant la capacité totale de LEDs disponibles)
- byte 2 à 3 (uint16\_t) : configuration du nombre de LEDs (TODO)

### 3.2.2 Fichier "data"

Le fichier "data" a une taille de TODO blocs (TODO bytes). Ils contiennent les données des LEDs en suivant l'organisation suivante :

TODO (restera probablement d'abord toutes les leds de la bande 0, puis toute la bande 1, puis la 2...) (noter aussi la réduction des données : 4 MSB)

## 3.3 Interface de la librairie (Raspberry Pi)

TODO (interface C, structure, buffer pour chaque LED, fonction pour rafraîchir, + interface supplémentaire (abstraction) pour contrôler des matrices au lieu de bandes de LEDs)

# Chapitre 4

# Conception

Ce chapitre décrit la structure de l'implémentation du projet. Il explique les techniques qui seront utilisées lors de l'implémentation afin de réaliser les fonctionnalités nécessaires.

## 4.1 Contrôle des LEDs

TODO

## 4.2 Interfaçage USB

TODO

## 4.3 Synchronisation

TODO

## 4.4 Envoi des valeurs côté Raspberry Pi

TODO

# Chapitre 5

# Implémentation

Ce chapitre décrit la structure et les choix d'implémentation des différentes parties du projet.

## 5.1 Environnement de développement

Tout le code du système a été réalisé dans VS Code (v1.46.1)<sup>1</sup>.

La partie embarquée a été réalisée à l'aide de l'extension PlatformIO (v4.3.4)<sup>2</sup>. Il s'agit d'une plateforme qui s'intègre dans l'IDE (par exemple VS Code ou Atom), et qui offre des outils de développements pour des systèmes embarqués. Il est compatible avec des centaines de microcontrôleurs, notamment le Blue Pill STM32F103C8T6 utilisé dans ce projet.

## 5.2 Programme embarqué

TODO (utilisation HAL ; système de fichiers ; 3 DMAs)

### 5.2.1 Librairies

TODO (lib USB ; fichier leds basé sur M.H.)

## 5.3 Interface (librairie) côté Raspberry Pi

TODO (implémentation structure ; envoi (écriture fichier) ; sync)

---

1. <https://code.visualstudio.com>  
2. <https://platformio.org/>

## Chapitre 6

# Tests et validation

Afin de valider le fonctionnement de la solution et tester ses limites, deux types de tests ont été réalisés.

### 6.1 Tests de fonctionnalité

Ces tests permettent de tester le bon fonctionnement du système. Pour les effectuer, il suffit de brancher des bandes de LEDs sur chacune des 8 sorties en parallèle (le test peut être effectué par exemple en deux fois, avec 4 bandes en parallèle à la fois), et de démarrer le programme TODO sur le Raspberry Pi.

Résultats attendus : TODO

### 6.2 Tests de performances

Ces tests permettent de mesurer les performances du système dans différentes situations. Pour les effectuer, il suffit d'effectuer le branchement TODO, et de démarrer le programme TODO sur le Raspberry Pi.

Résultats attendus : TODO

Mesures des performances : TODO

## Chapitre 7

# Améliorations et travaux futurs

L'implémentation des fonctionnalités principales du projet est terminée, mais de nouvelles idées ont été évoquées lors des séances.

### 7.1 Interfaçage USB pour d'autres périphériques

Ce projet comporte des modules qui pourraient être utilisés pour d'autres projets, en particulier la partie interfaçage USB. En effet, un périphérique s'annonçant comme un périphérique de stockage et qui permet de contrôler du hardware spécifique en éditant simplement un fichier permettrait d'abstraire l'utilisation non seulement de LEDs, mais également d'autres périphériques comme par exemple des servomoteurs. Grâce à cet interfaçage, le Blue Pill peut très facilement devenir une extension pour n'importe quel PC lui permettant de contrôler du matériel spécifique.

### 7.2 Extension de la capacité de la tour télécom

Comme expliqué dans l'analyse, les différents MCU de la famille des STM32F ont beaucoup en commun. Il serait aisément de remplacer le STM32F103C8T6 utilisé actuellement par un autre microcontrôleur ayant plus de RAM, dans le cas où plus de LEDs devraient être contrôlées par le système. La plupart du code sera commun entre les deux contrôleurs.

## Chapitre 8

# Mise en place et exemple d'utilisation

TODO (démonstration, voir peut-être affichage d'une image ou vidéo simplement avec OpenCV)

## Chapitre 9

# Conclusion

TODO

### 9.1 Comparaison avec les objectifs

TODO

### 9.2 Rétrospective sur la planification initiale

TODO

### 9.3 Conclusion personnelle

TODO

## Chapitre 10

# Déclaration d'honneur

Je, soussigné, Nicolas Maier, déclare sur l'honneur que le travail rendu est le fruit d'un travail personnel. Je certifie ne pas avoir eu recours au plagiat ou à toute autre forme de fraude. Toutes les sources d'information utilisées et les citations d'auteur ont été clairement mentionnées.



# Glossaire

**Adafruit** Entreprise concevant et vendant des produits électroniques, notamment les NeoPixels. 23

**Blue Pill** Microcontrôleur équipé d'un STM32F103C8T6. 23

**LED** Light-Emitting Diode, source de lumière. 23

**MCU** MicroController Unit. 23

**microcontrôleur** Petit ordinateur sur un circuit intégré. 23

**MSB et LSB** Most Significant Bit (bit de poids fort) et Least Significant Bit (bit de poids faible). 23

**pixel** Point (RGB) sur une bande de LEDs adressable. 23

**Raspberry Pi** Série de "single-board computers". 23

**USB MSC** USB Mass Storage device Class, un ensemble de protocoles de communications pour les périphériques de stockage USB. 23

**ws2812b** Contrôleur de LEDs dans les bandes NeoPixel. 23

**ws281x** Nom générique pour désigner les ws2811, ws2812 et ws2812b. 23

# Bibliographie

- [1] Datasheet du ws2812b  
[https://voltiq.ru/datasheets/WS2812B\\_datasheet\\_EN.pdf](https://voltiq.ru/datasheets/WS2812B_datasheet_EN.pdf), 10.07.2020
- [2] Datasheet du STM32F103C8  
<https://www.st.com/resource/en/datasheet/stm32f103c8.pdf>, 10.07.2020
- [3] Datasheet du CD4050B  
<https://www.ti.com/lit/ds/symlink/cd4049ub.pdf>, 10.07.2020
- [4] Improved STM32 WS2812B DMA Library (Martin Hubacek)  
<http://www.martinhubacek.cz/arm/improved-stm32-ws2812b-library>, 10.07.2020