PROJET DE MASTER 2

Autoscope
Second compte rendu

Auteurs :
Thomas ABGRALL
Clément AILLOUD
Thibaud LE DOLEDEC
Thomas LEPOIX

MASTER Systèmes Embarqués

E.S.T.E.I.
École Supérieure des Technologies Électronique, Informatique, et Infographie
Département Systèmes Embarqués

9 janvier 2019

Table des matières

Première partie

Partie de groupe

Chapitre 1

Évolution du cahier des charges

Plusieurs éléments du cahier des charges ont été amenés à changer depuis la première revue de projet.

1.1 Nécessité du traçage d'astre

Le traçage d'astre, vu au départ comme une fonctionnalité intéressante, s'est imposé comme une fonctionnalité nécessaire. En effet il peut être particulièrement difficile pour une personne non initiée à l'astronomie de positionner le télescope vers un astre précis ou de reconnaître un astre que l'on observe. Il nous est donc apparu primordial que le télescope permette d'affranchir l'utilisateur de la nécessité d'avoir des bases en astronomie pour observer le ciel.

En étudiant les solutions disponibles nous avons trouvé des logiciels de simulation du ciel, comme par exemple le logiciel libre Stellarium.

FIGURE 1.1 – Captures d'écran de Stellarium

Celui-ci permet notamment :

- De connaître les coordonnées d'un astre par rapport à l'endroit sur terre où se situe l'observateur.
- De s'orienter dans le ciel selon des coordonnées.
- De s'interfacer avec d'autres systèmes logiciels et/ou matériels

Nous avons donc décidé dans un premier temps de développer une interface pour piloter le télescope depuis un ordinateur distant doté de Stellarium. Puis éventuellement d'embarquer Stellarium dans l'ordinateur du télescope. Ainsi Stellarium fera partie intégrante de son interface utilisateur.

Celle-ci pourrait être alors un menu discret permettant de passer de l'exploration virtuelle du ciel à la vue correspondante à travers le télescope à d'autres élément comme un dispositif d'amélioration de la qualité des images prises.

1.2 Nécessité d'une centrale inertielle et d'un GPS

L'utilisation d'un logiciel de traçage d'astre tel Stellarium nécessite la compatibilité du télescope avec les coordonnées d'azimut et d'élévation couramment utilisées en astronomie. Il est également nécessaire pour cela de savoir de quel endroit sur terre le télescope observe le ciel, d'où l'utilisation d'un GPS.

Pour connaître l'azimut et l'élévation, il faut avoir des repère dans les deux dimensions. Un magnétomètre permet de déterminer la direction du nord et un accéléromètre permet de connaître la direction du sol, c'est à dire la verticale.

Une centrale inertielle est un composant intégrant un magnétomètre, un accéléromètre et un gyroscope. Elle permet de connaître directement les coordonnées absolues de son orientation dans l'espace.

1.3 Changement de SoC

L'utilisation de Stellarium requiert au minimum 512MiB de RAM et 1GiB pour une utilisation optimale. Or la carte PICO-PI ne dispose que de 512MiB de RAM, elle est donc incapable de faire fonctionner Stellarium correctement.

Nous avons choisi une Raspberry Pi 3 B avec 2GiB de RAM en remplacement. En dépit de ses faibles capacités d'industrialisations, la Raspberry Pi a l'avantage d'être populaire dans le milieu de l'électronique amateur, c'est-à-dire le publique le plus susceptible d'être intéressé par ce genre de projet.

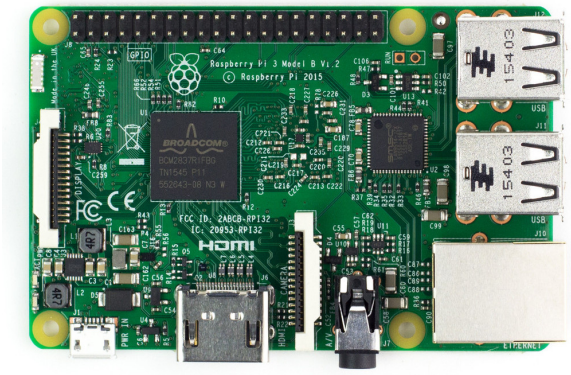

FIGURE 1.2 – Captures d'écran de Stellarium

La caméra et l'écran utilisés ne seront donc plus ceux fournis avec la PICO-PI mais ceux de la Raspberry Pi, à savoir :

- Le module Raspicam v2.1 intégrant une caméra IMX219 de $8Mpx$.

L'écran demeure toutefois une option, celui-ci sera intégré si le télescope embarque Stellarium.

Deuxième partie

Thomas LEPOIX

Chapitre 2

Architecture

2.1 Architecture système

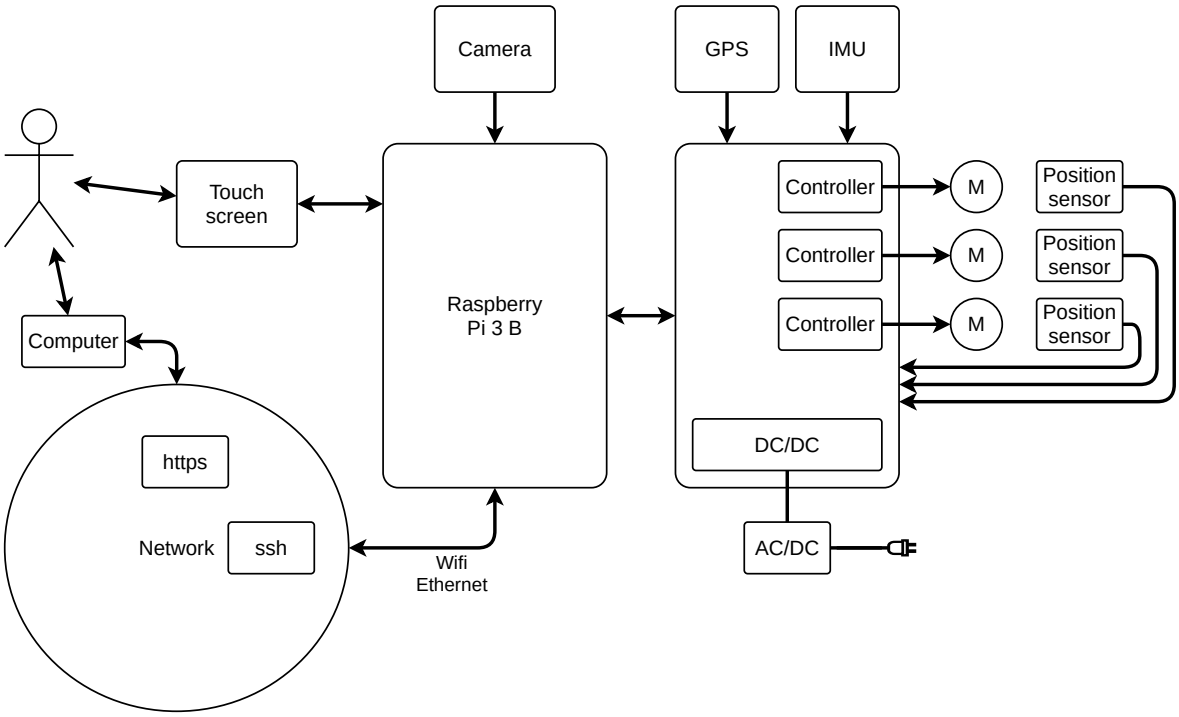

FIGURE 2.1 – Schéma fonctionnel du télescope

Le télescope sera composé d'une carte Raspberry Pi à laquelle sera connecté une carte de même format accueillant le module GPS, le module IMU (centrale inertielle), les contrôleurs des moteurs ainsi que l'alimentation du système. Les modules GPS et IMU seront probablement déportés afin de les éloigner des moteurs et de leurs perturbations électromagnétiques.

La caméra et l'éventuel écran tactile seront directement connectés à la Raspberry Pi.

Trois moteurs permettront de mouvoir le télescope :

- Un pour l'azimut.
- Un pour l'élévation.
- Un pour le zoom.

Pour interagir avec le télescope, l'utilisateur aura le choix d'utiliser l'écran tactile embarqué ou bien un ordinateur connecté au télescope via le réseau. Cette solution étant la voie d'accès préférentielle aux contrôles télescope.

2.2 Architecture logicielle

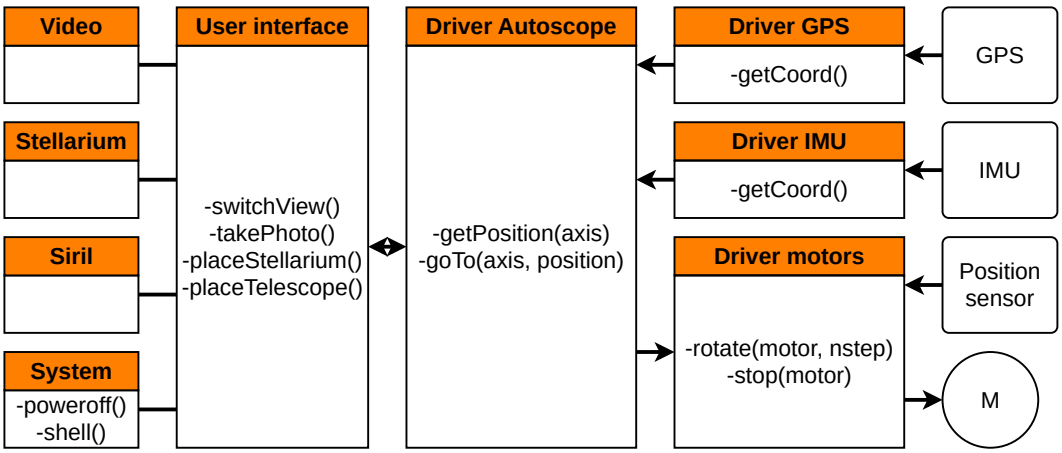

FIGURE 2.2 – Architecture logicielle du télescope

Le fonctionnement du télescope reposera sur plusieurs briques logicielles.

- Au plus bas niveau les drivers élémentaires permettant de recueillir les données que fournissent l'IMU et le GPS, ainsi que d'actionner les moteurs, ceux-ci étant asservis par des capteurs de positions extrêmes.
- À l'étage suivant un driver de tout le télescope permettant de connaître les coordonnées précises du télescope (position géographique, orientation spatiale) et d'ordonner au télescope d'adopter une orientation particulière.
- À l'étage supérieur l'interface utilisateur permettant de prendre des clichés du ciel, d'orienter le télescope selon Stellarium, de placer Stellarium selon l'orientation du télescope, mais avant tout de basculer entre les différents modes du télescope :
  - Observation du ciel via la caméra.
  - Exploration virtuelle du ciel avec Stellarium (local ou distant).
  - Amélioration d'images avec Siril.
  - Interaction avec le système d'exploitation du télescope.

2.3 Architecture matérielle

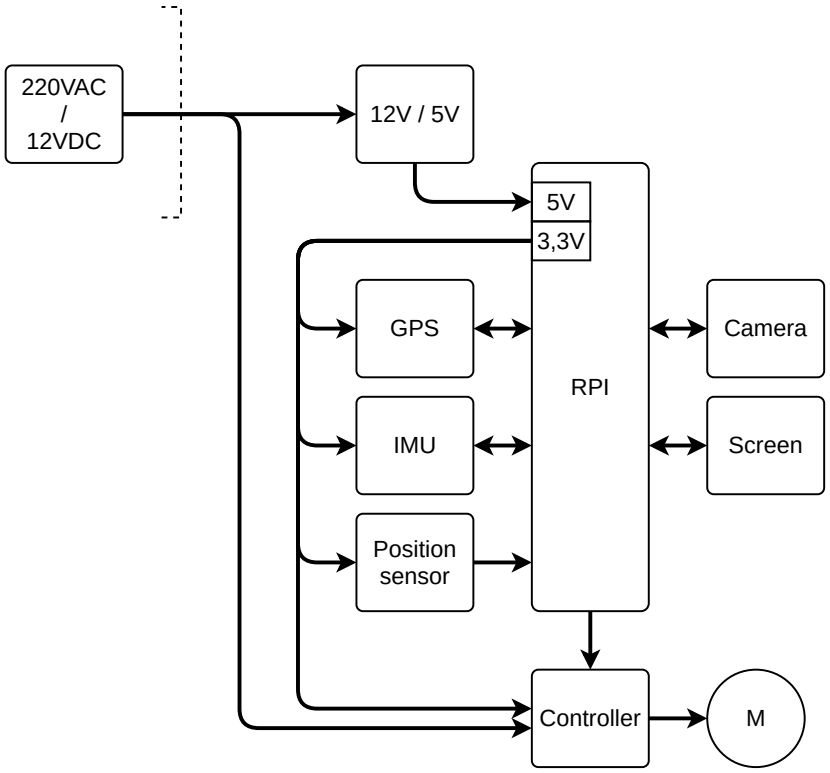

FIGURE 2.3 – Schéma structurel de premier niveau du télescope

La première question de l'étude structurelle du télescope est celle de l'alimentation.

Le télescope sera raccordé au secteur par un module externe $220VAC/12VDC$. Ensuite les moteurs seront alimentés en $12V$ via leurs contrôleurs respectifs et la Raspberry Pi sera alimentée en 5V. Un convertisseur $12V/5V$ est donc a ajouter.

Tous les autres éléments seront alimentés en $3,3V$ par la Raspberry Pi. Il est toutefois important de s'assurer que la consommation maximale en courant de ces élément ne dépasse pas ce que peut fournir la Raspberry Pi, à savoir $500mA$.

- Contrôleurs des moteurs ($\times 3$) : $8mA$
- GPS : $25mA$
- IMU : $3,7mA$
- Capteurs de position des moteurs ($\times 4$) : $33\mu A$

La consommation en courant totale des périphériques de la Raspberry Pi est largement inférieure à $500mA$, le montage est donc réalisable sans risque de dysfonctionnement ou de dommages.

La question des connectiques sera également déterminante dans l'élaboration de la carte pluggable sur la Raspberry Pi

Chapitre 3

Software

3.1 Support de la caméra

Pour gérer les différentes configurations matérielles de façon plus "userfriendly", la Raspberry Pi dispose d'un fichier de configuration **`config.txt`** que le *bootloader* interprétera pour sélectionner les *overlays* correspondants et composer le *devicetree* qui convient à l'architecture matérielle utilisée. Celui-ci étant ensuite passé au *kernel* lors du démarrage.

Cette subtilité propre aux Raspberry Pi permet à l'utilisateur de ne pas avoir besoin d'avoir affaire au *devicetree* quand il s'agit de configuration. Par exemple l'activation du support d'un élément courant sur les Raspberry Pi comme le module Raspicam.

le logiciel **`raspi-config`** n'est autre qu'une interface à ce fichier de configuration.

Pour activer le support matériel de la caméra, il faut ajouter les lignes suivantes à ce fichier :

```
start_x=1
gpu_mem=128
```

À travers Yocto, cela passe par l'ajout des lignes ci-dessous au fichier **`local.conf`** et donc à son modèle le fichier **`meta-autoscope/conf/local.conf.sample`** dans la *layer* dédiée au projet.

```
VIDEO_CAMERA = "1"
GPU_MEM = "128"
```

Quant à l'utilisation de la caméra, il existe des logiciels tels que **`raspivid`** pour filmer ou **`raspistill`** pour prendre des clichés. Tous deux font partie de la suite **`userland`** que l'on ajoute à notre image via la ligne ci-dessous dans le fichier **`meta-autoscope/recipes-autoscope/images/autoscope-console-image.bb`** :

```
IMAGE_INSTALL += "userland"
```

À l'usage, la commande suivante permet d'afficher en plein écran le flux vidéo jusqu'à ce qu'on le stoppe :

```
~$ raspivid -t 0
```

3.2 Étude de l'embarcation de Stellarium

Une première évaluation de la faisabilité de l'embarcation de Stellarium sur la Raspberry Pi a été de l'installer via le gestionnaire de paquet sur une Raspberry Pi dotée de Raspbian.

```
1 ~$ sudo apt-get install stellarium
```

Avec le compositeur graphique logiciel, Stellarium est totalement inutilisable. Cependant il est possible d'activer le compositeur graphique matériel. Pour cela il faut activer l'option **`Advanced Options -> GL Driver`** dans le menu **`raspi-config`**.

La différence de rendu peut être observée avec l'outil **`glxgears`** prévu à cet effet. Celui-ci s'installe par la ligne suivante :

```
1 ~$ sudo apt-get install mesa-utils
```

FIGURE 3.1 – Captures d'écran de glxgears

Le résultat est alors meilleur, Stellarium est utilisable bien qu'il lui arrive de planter lorsque l'on fait un déplacement trop brusque dans le ciel.

Il existe une version mobile de Stellarium disponible notament pour Android et iOS qui est une version allélégée et optimisée pour l'usage sur un smartphone.

Ses sources sont disponibles à l'adresse suivante :
`https://noctua-software.com/stellarium-mobile`

Il semble que cette version puisse fonctionner telle quelle sur Linux. À l'heure actuelle nous n'avons pas réussi à compiler Stellarium mobile pour un problème de dépendance à une librairie (Qt5Declarative, faisant partie du paquet qtdeclarative5-dev) qui n'existe plus dans les versions récentes de Qt, ultérieures à la version Qt5.6 pour être précis.

Le problème pourrait sans doute être résolu en récupérant ladite librairie dans les dépots d'une distribution plus ancienne, Ubuntu Xenial (Qt5.5.1) ou Trusty (Qt5.2.1) par exemple.

Chapitre 4

Organisation prévisionnelle

Me concernant, pour le prochain cycle de travail le plus urgent est d'étudier l'aspect hardware du télescope et de produire la carte qui accueillera les différents périphériques de la Raspberry Pi.

Ensuite ma contribution pourrait être la plus utile dans l'intégration au système d'exploitation des différents éléments qui le composent, comme le logiciel principal de l'Autoscope ou les logiciels Siril et éventuellement Stellarium. Ou bien dans les choix ergonomiques préalables au développement de l'interface utilisateur.

Troisième partie

Thomas ABGRALL

Chapitre 5

Motorisation

Le projet repose sur un projet de telescope manuel, qui devait être placé à la main pour viser les étoiles. Nous allons voire ici les choix et mise en oeuvre réalisé pour automatisé notre télescope.

5.1 Placement des moteurs

Le télescope est composé de deux axes à automatiser : la rotation et l'inclinaison.

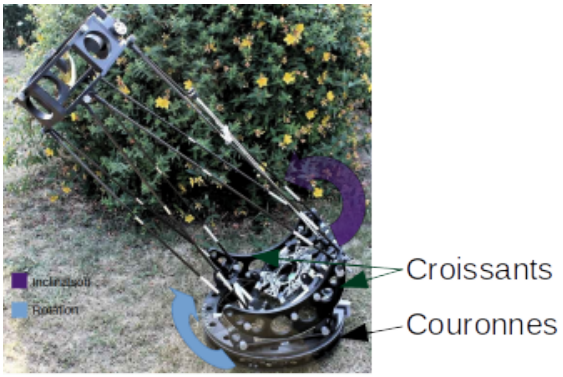

FIGURE 5.1 – Illustration des axes telescope

Pour mouvoir le telescope en rotation, nous avons choisi de placer un moteur sur la couronne fixe de la base du telescope. Qui ce dernier ferai tourner la couronne du dessus qui sera équipée d'une courroie sur ça face intérieure.

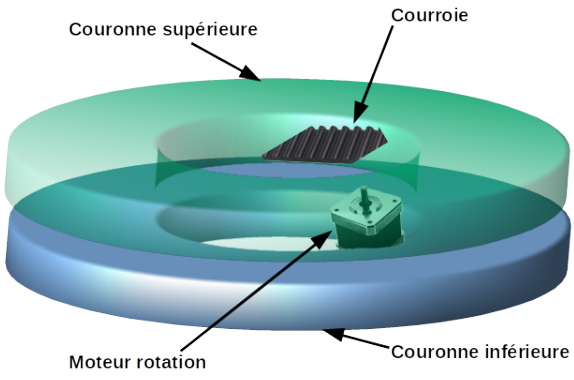

FIGURE 5.2 – Illustration placement moteur rotation

Avant de choisir cette soluction nous avions réfléchi sur plusieurs autres. Nous avions envisagé de placer une courroie sur la face intérieur du croissant. La forme non-circulaire de l'interieur du croissant empêche de placer un axe FIXE pour entrainer une courroie car la forme du croissant va se faire rapprocher la courroie de l'arbre du moteur durant ça progression. Une autre solution visée à placer le moteur au centre de la courroie, cependant vous avons conclus que la surface sur laquelle va s'appliquer la force du moteur est plus petite que dans la solution que nous avons retenue.

Pour agir sur l'inclinaison, nous relions les deux barre qui maintiennent les croissants avec une courroie. Elle sera tendu grâce à l'arbre du moteur et un pignion. Le tout sera placé entre des deux croissants, sous le miroire principale.

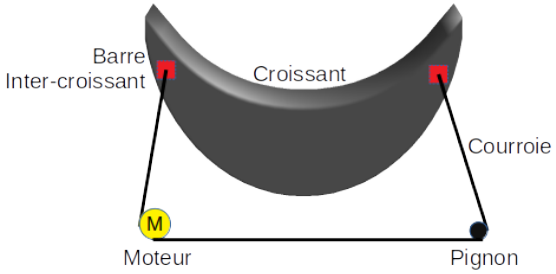

FIGURE 5.3 – Illustration placement moteur inclinaison

Il nous reste ensuite une derenière chose a automatisé, le zoom. Le zoom sera placé entre le miroire secondaire et la caméra. Il sera entouré par une courroie qui fera le lien entre le zoom et une autre courroie entrainé par le moteur zoom.

FIGURE 5.4 – Illistration moteur zoom

5.2 Choix des moteurs

D'après la quantité de matière au bout du télescope, les matériels qui seront installés au bout (miroire, zoom, caméra, moteur pour le zoom, fixation) nous avons estimé que doit pouvoir soulever au minimum 4 kilogrammes. Nous avons donc choisi le moteur de référence : 17HM15-0904S. De plus nous disposions d'un tel moteur, avec lequel nous avons pu réaliser des testes pour nous assuré qu'il réponde bien a notre cahier des charges. Le moteur du zoom a besoin de moin d'énergie pour agir sur le zoom, nous avons donc choisi un moteur moin puissant celui retenu est le S20TH30-0604A.

FIGURE 5.5 – Photo du moteur

5.3 Contrôle des moteurs

Pour contrôler le moteur nous avons choisi le composant "A4988 Stepper Motor Driver Carrier". Il permet de contrôler la rotation du moteur, choisir entre le mode pas, demi-pas, quart de pas, huitième de pas et seizième de pas. Il y a également une pin activation et endormissement mais que nous n'utiliserons pas.

FIGURE 5.6 – Photo du controleur

5.4 Présentation du code

Je développe un C un driver linux pour permettre le contrôle des moteurs, mais également de surveiller des interrupteurs.
Le choix de réaliser un drive linux au lieu d'un logiciel est justifié parcequ'il permettra à un programme en couche supérieure d'utiliser ces fonctions pour utiliser les moteurs.
Pour contrôler les moteur faut générer une impultion, à chaque front montant le moteur réalise un pas si il se trouve dans le mot pas à pas, sinon une demi-pas etc. Il faut également choisir le sens de rotation du moteur

Fonctionnement :
Lors de l'initialisation du drive les interruptions de chacun des interrupteurs se mette en service, dès lors si les interrupteurs placer en fin de course de la rotation s'active alors le moteur de rotation est immédiatement arrêté. Il en est de même pour l'inclinaison. Cette partie est autonome, et ne doit pas être utilisé par des logiciles tiers. A l'inverse du contrôle des moteurs. Car chaque moteur dispose d'un fonction qui sera appelé par un logiciel tiers pour lancer un moteur sur un certain nombre de pas et dans une certaine direction.

Avancement :
Le contrôle des moteurs rotation et inclinaison sont réalisés. Les interruptions des interrupteurs de rotations sont également réalisées.

Travail à venir :
Réalisation du code permettant de contrôler le moteur zoom.
Contrôle des modes des moteurs (mode de pas) en fonction de la distance à parcourir. Quand le moteur s'approche de la fin on réduit la taille des pas pour augmenter la précision, afin d'arriver exactement là où il faut.

5.5 En soutien

J'ai participer en soutien dans la partie optique.
Pour cela j'ai démarché plusieures entreprises dans le domaine de l'optique tel que STEMMER IMAGING S.A.S dont Alexis Bouras Ingénieur Technico-Commercial / Région Sud-Ouest, ce dernier m'a explique en fonction de nos contraites et materiels ce qu'il nous faut comme technologie réellement. J'ai également pris contact avec un amis travaillant dans le milieu.
J'en ai conclus qu'il nous faut un zoom variable et qu'une focale variable nous est inutile.