Rapport projet
Plateau tournant
2021-2022

22 Avril

M1-EEA

Créé par : Grossard Mathieu
Schultz Quentin

FST
Faculté des Sciences et Techniques
UNIVERSITÉ HAUTE-ALSACE

Professeur responsable :
Urban Jean-Philippe

Les remerciements :

Nous sommes Mathieu Grossard et Quentin Schultz. Nous sommes ensemble pour travailler sur notre projet qui est un plateau tourant.

Nous tenons à remercier toutes les personnes actives sur les différents forums Arduino qui nous ont permis de peaufiner notre code pour le projet ainsi que les différentes fiches techniques de nos modules qui possèdent pour la plupart un code de démarrage.

Ainsi que le corps enseignant, plus particulièrement Mr. Urban qui nous a imprimé notre boîte et nous a conseillé sur certains branchements et composants ainsi que Mr. Hermann et Mr. Kihl qui ont été disponibles pour répondre à nos questions concernant le matériel.

Sommaire

Introduction :

Nous avons dû réfléchir sur une thématique de plateau tourant pour la prise de photos fonctionnant avec une application mobile. Pour cela nous avons réfléchi sur ce projet en présentiel et à distance en utilisant Discord pour pouvoir communiquer entre nous. Afin aussi de choisir le matériel nécessaire, savoir s'il pouvait correspondre au cahier des charges ainsi que de son fonctionnement recherché.

Nous sommes partis sur un plateau tournant fonctionnant aussi bien manuellement que de façon autonome. Ce plateau a été voulu pour ressembler à des modèles déjà existants. Ce rapport va donc contenir notre démarche de réflexion sur ce projet.

Nous allons alors parler de notre étude sur le projet, son but, puis nous verrons les étapes de réalisation du projet. Et enfin, le bilan de cette étude.

1. Contexte et cahier des charges :

Nous avons pour but du projet de créer un plateau tournant qui sera utilisé principalement lors de prise de photos. Ce qui nous permettra de faire des photos sous différents angles de vus. Pour cela notre plateau est prévu pour communiquer avec un smartphone qui nous permet de le contrôler à distance.

Ainsi nous avons fait plusieurs choix ou plutôt nous imposer des contraintes. Nos choix ou contraintes sont :

- Un moyen de communication à distance
- Un plateau portatif d'une taille réduite entre 12 et 15 cm et alimenté sur batterie
- Une application smartphone pour l'interface utilisateur permettant de sélectionner un angle, une vitesse, un mode automatique et sens contraire ainsi que des boutons permettant la connexion et déconnexion de la communication à distance.
- Un moteur assez petit avec suffisamment de couple pour réussir à faire tourner au moins 1kg sans difficulté et pouvoir faire tourner le plateau à au moins 3 tours par minutes
- Mise en place d'un interrupteur pour la mise hors tension, d'un bouton pour sélectionner des vitesses ainsi que d'un port pour recharger la batterie

Il s'agit donc de toutes les contraintes que nous avons reçues ou que nous nous sommes données, dans un sens imposé, lors de l'élaboration du projet.

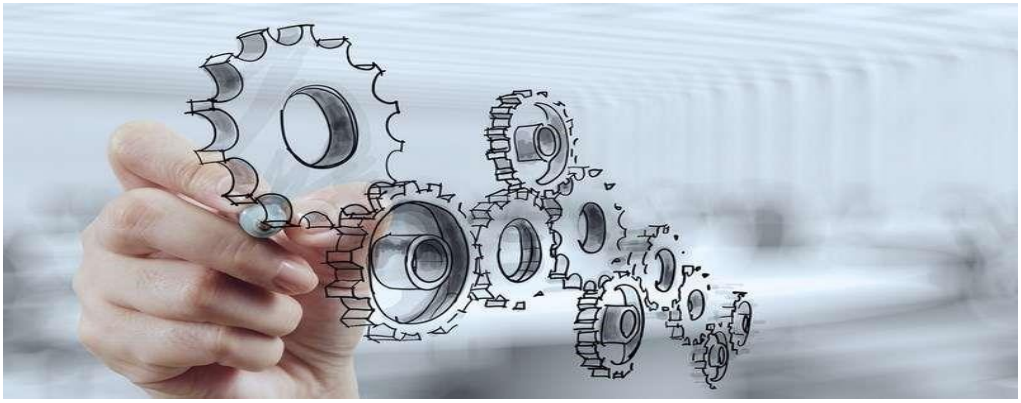
Cahier des charges optionnel :

Nous avons eu comme tâches optionnelles d'être capable de prendre des photos de façon synchrone avec le mode automatique du plateau (lors d'un arrêt à certains angles, on prend une photo).

2. Etudes :

Nous allons dans cette partie expliquer notre procédé de réflexion pour notre étude. Cela concerne notre démarche de réflexion du public visé, le fonctionnement voulu de notre projet et donc nous aurons une grande partie avec les différents composants utilisés.

Cette partie comportera le but d'utilisation du composant, ce que nous avons souhaité faire, ce qui a pu être fait et ce que l'on n'a pas pu réaliser finalement.



2.1. But :

Notre projet a été conçu pour être utilisé dans le milieu de la photo. Son principe premier est de permettre à l'utilisateur de faire des photos sous une multitudes d'angles différents.

Bien sûr, il peut servir dans d'autre domaine en tant que présentoir notamment dans un magasin de jouet par exemple pour mettre un produit en avant.

Puisque nous avons choisi le milieu de la photographie nous devions donc décider de leur besoin, comment l'utiliser et des contraintes. Ainsi nous avons d'abord discuté de la forme que cela devrait prendre et nous nous sommes concentrés sur une boite et un plateau rond. Etant donné qu'on en retrouve le plus souvent de cette forme. Nous avons dû décider de la manière de communiquer, de la manière de facilité la rotation plateau, afin d'avoir une rotation avec le moins de frottements possible et celle-ci a pris plus de temps que prévu. Au final nous avons choisi une communication Bluetooth et un roulement à billes déjà tout fait, ce qui était moins contraignant. Donc nous avons décidé d'utiliser un roulement Lazy Susan qui est généralement utilisé pour les plateaux tournants.

Étant donnée cela, notre première étape consistait à une recherche sur d'autres projets pour pouvoir s'inspirer du fonctionnement, de la forme et du fond pour la création du nôtre. La première étape dans cette recherche consistait déjà à observer l'exemple qui a été fourni lors de la présentation du projet. Celui-ci était une bonne idée pour le point du fonctionnement de base mais pas plus. Puisque pour sa réalisation nous devions avoir besoin d'utiliser une découpeuse laser et que nous, nous souhaitons réaliser ce projet avec une imprimante 3D. Ensuite la taille et la forme du projet était différente de la norme de notre cahier des charges qui correspond à une envergure comprise entre 12 et 15 cm. Il ne disposait pas d'une connexion en Bluetooth permettant de le commander à distance et avait un écran supplémentaire pour lire des données qui pouvaient être remplacées. Le plateau pouvait changer de vitesse à l'aide d'un potentiomètre. Nous allons utiliser le même principe mais en utilisant notre application que nous verrons plus tard.

L'étape qui a suivi, a été de faire des recherches pour faire une forme optimale pour correspondre à la taille des caractéristiques du cahier des charges fournies. Cette forme est devenue optimale et tel que l'on la connaît avec l'arrivée de l'utilisation du Lazy qui nous sert de roulement à billes pour permettre d'aider lors des rotations du moteur.

Comme nous avons décidé de viser le milieu de la photo nous devons être capable d'indiquer un angle pour faire différentes prises de photos avec une pause d'une certaine durée pour nous laisser le temps de la prendre. C'est pourquoi, nous avons décidé de faire une application pour faciliter l'utilisation. Sur laquelle nous avons afficher un cadran permettant une saisie facile des différentes valeurs d'angles, ainsi qu'un bouton pour accéder et quitter l'appareil photo pour nous permettre de prendre plus simplement des captures d'image sans quitter l'application.

Finalement nous avons aussi rajouté un interrupteur pour la mise hors tension, un bouton sur l'application pour la déconnexion ainsi qu'un bouton poussoir pour pouvoir modifier la vitesse de rotation du plateau tournant lorsque celui-ci n'est pas connecté en Bluetooth.

2.2. Composants utilisés :

Carte Node ESP 32 :

Nous avons décidé d'utiliser comme microcontrôleur la carte Node ESP 32 pour son prix faible et sa petite taille mieux adaptée pour notre étude. Malgré son coût plus faible il ne faut pas croire à une faible performance. Au contraire pour ce qui est des entrées et sorties (E/S) l'ESP 32 possède 4 pins digitaux et 15 pins analogiques. En ce qui concerne la capacité, il possède une mémoire RAM[1] de 520 kB, EEPROM[2] de 448 kB[3] et aussi une mémoire Flash de 4000 kB (pour la programmation et l'enregistrement de données). En utilisant le pin VIN on peut connecter une alimentation externe jusqu'à 5Volt[4] que l'on abordera plus tard.

Le choix de l'ESP est aussi en occurrence avec l'environnement de travail qui est Arduino IDE. Celui-ci nous permet de créer nos programmes et de les transférer à la carte à l'aide de la prise USB. Mais aussi car elle intègre la gestion du wifi et du Bluetooth.

Bouton poussoir :

Nous avons décidé d'intégrer à notre étude un bouton poussoir. Ce bouton se trouve sur le pourtour de la boite.
Il sert à changer la vitesse. Le fonctionnement du bouton est simple. Nous avons défini 3 vitesses différentes à savoir la vitesse initiale de 5 tours par minutes, l'appui sur le bouton nous permettra de passer à une vitesse de 2 tours par minutes, puis 10 tours par minutes jusqu'à revenir sur la valeur initiale.

Nous n'avons pas eu de problèmes lors de l'utilisation de ce bouton puisque le principe est courant.

Batterie :

L’accu Li-Ion[5] a été notre choix pour un prototype de notre de plateau. C’est l'une des batteries les plus efficaces. On la retrouve dans plusieurs types d’appareil notamment dans les ordinateurs. Elle nous permet de durer plus longtemps qu’une batterie Lipo[6] par exemple. Elles sont rechargeables, durables, légères et ont la meilleure densité d'énergie de stockage que toute autre batterie.

Après étude l’accu Li-Ion que nous utilisons pour le prototype possède 3,7 Vcc et 3500 mAh[7]. L’accu a pour dans notre projet une faible autonomie mais selon le calcul de la consommation qui est capacité de la pile / consommation de l’appareil = autonomie en heures.

Soit consommation de l’appareil = ESP avec Bluetooth (80-180) + moteur (55) +driver (500) + pin de l’ESP (40*6) + GND de l’ESP (28) = 1003mAh. Ainsi la durée de vie maximum pouvant être atteinte avec les composants toujours actif nous avons une autonomie = 3500/1003 =3 heures et 30 minutes environ, en se plaçant dans le cas le plus défavorable.

Commutateur à glissière :

Nous utilisons un commutateur unipolaire à deux directions qui parmi tous les types d’interrupteur est le plus simple. En effet, il coupe ou il rétablit le circuit électrique selon si l’on souhaite faire fonctionner ou non le plateau. Pour cela on interrompt le courant du câble d’alimentation.

Recharge :

Pour pouvoir recharger notre plateau nous utilisons un coupleur pour un accumulateur de type 18650 à sortie sur fils qui correspond au modèle d'accu que nous avons choisi.

Nous avons opté pour un chargeur USB type-C car ce type est vouer à être le nouveau standard et peut charger la batterie à une puissance de 15 watts.

A savoir que les prises USB Type A et Type B n'offrent que 4 connexions alors que les prises USB 3.1 de Type-C en proposent 24, ce qui explique l'avantage d'énergie et une meilleure vitesse.

Booster :

Comme nous l'avons vu notre accu délivre 3.7 Vcc mais nous devons alimenter les drivers[8] du moteur avec 5v. Nous utilisons donc ce régulateur, qui permet de délivrer une tension de 5 Vcc à partir d'une tension de 0,5 à 5,5 Vcc.

Moteur et driver :

On utilise le driver fournit avec le moteur (le driver STP01). Il nous permet de convertir des signaux électriques pour que le moteur puisse les interpréter.

La vitesse de rotation et le couple des moteurs pas-à-pas dépendent de la tension d'alimentation et du courant.

Pour le choix du moteur nous l'avons choisi en fonction de notre couple maximal puis vérifier si les caractéristiques du moteur correspondent avec la vitesse demander mais également aussi en fonction de ses dimensions. Notre moteur est donc un moteur 5V possédant un couple de 0.03 Nm^9. A savoir qu'il est déjà composé d'un réducteur, 1/64, qui permet de modifier le rapport de vitesse et le couple entre l'axe d'entrée et l'axe de sortie

Totale :

Le budget correspond aux dépenses réalisées pour construire la version finale que l'on propose.

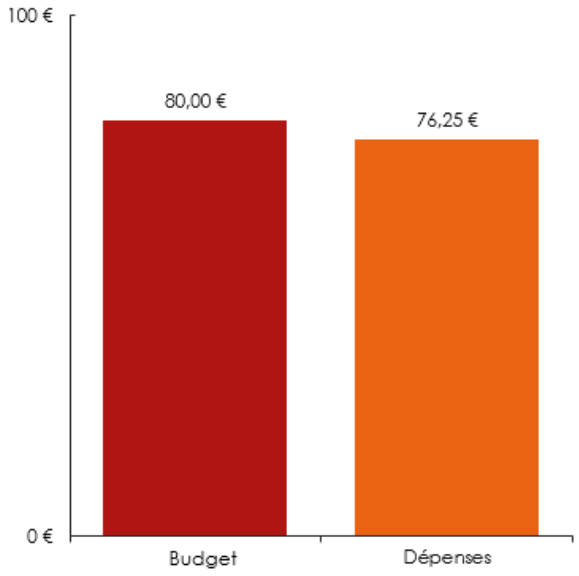
BUDGET DU PROJET	
Élément	Montant
Budget initial d'un plateau	80,00 €

POURCENTAGE DE BUDGET DÉPENSÉ	
	95%

SYNTHÈSE	
Budget total	80,00 €
Dépenses totales	76,25 €

DÉPENSES LOCAL OUTILLAGE	
Élément	Montant
Accu Li-Ion 18650-3.4A	16,80 €
Coupleur d'accu BH18650-1	1,95 €
Module NodeMCU ESP32	15,45 €
BP subminiature	0,60 €
Moteur pas-à-pas + driver STP01	5,90 €
Commutateur à glissière	4,61 €
Chargeur LiPo USB Type-C DFR066	6,30 €
Régulateur 3,3 Vcc ADA4711	3,25 €
Lazy Susan 5"	15,49 €
Divers	5,90 €

SOLDE	3,75 €



L'analyse nous montre que le budget initial a été entièrement respecté puisque le budget de départ est de 80€ et nous avons au final 76.25€ avec un solde de +3.75 €. La raison pour cela c'est que nous avons recherché le moins cher possible que ça soit au niveau du matériel ou même au niveau de la création de l'application (entièrement gratuite).

2.3. Supports utilisés :

Pour les supports que nous avons choisis nous avons deux types. Le support pour créer notre application mobile et notre support pour programmer l'esp32. Pour l'esp 32 le choix a été vite effectué mais pour l'application cela a nécessité plus de temps. Puisque nous pouvions créer une application depuis le début en utilisant soit python soit java. Mais avec tous ce que nous souhaitions faire cela aurait demandé trop de travail pour la durée du projet.

Ainsi nous avons dû nous diriger vers des outils de créations d'applications préexistantes telles que MIT App Inventor, Thunkable, AppyBuilder et bien d'autres. Notre choix pour la sélection de l'application s'est joué sur l'accessibilité aux ressources pour comprendre son fonctionnement et pouvoir l'adapter au principe de notre projet. Cela commence par des recherches sur leurs forums respectifs pour vérifier si ceux-ci sont assez fournis ou non.

Notre choix final c'est dirigé vers une application bien précise pour deux raisons. Des extensions directement accessibles pour l'aide à notre projet pour le finaliser mais aussi que lors de notre recherche pour une application à utiliser avec Arduino IDE, qui a bien entendu été pour programmer l'esp, c'est celui-ci qui revenait le plus souvent en tête de liste et était bien fourni.

Ainsi comme dit nous avons choisi MIT App Inventor pour la création de notre application qui nous permettra de gérer notre projet. Ainsi que Arduino IDE pour pouvoir contrôler notre carte qui est l'esp 32. Nous avons utilisé pour chaque support des extensions que nous allons maintenant présenter.

Ces extensions sont celles qui ont permis la réalisation finale du projet même les parties optionnelles du cahier des charges. Certaines de ces extensions ont pu être modifiées pour assurer leur bon fonctionnement

Pour créer notre application Android nous avons décidé d'utiliser App Inventor qui est un logiciel de développement d'applications Android créé par Google. Ce logiciel est adapté et orienté pour les débutants. Il est basé sur une interface graphique et nous permet de faire abstraction de lignes de code.

Ce qui nous simplifie le développement de notre application sous Android.

Nous utilisons deux extensions pour notre projet :

- L'extension ProCamera
  Elle nous donne accès à plus de fonctionnalités sur la caméra Android du smartphone (elle nous permet d'initialiser une caméra en choisissant la caméra frontale ou avant et permet la sauvegarde de photos par exemple).

- L'extension Taifun Tools
  Elle permet d'obtenir une collection de plusieurs outils, dans notre cas elle nous permet de garder notre écran de l'appareil allumé tant que notre fenêtre est visible pour l'utilisateur.

Pour programmer l'ESP 32 nous avons utilisé Arduino IDE

Ce logiciel nous permet d'éditer, de téléverser, de compiler un programme en langage C.

Nous utilisons les extensions CheapStepper, crée par Tyler Henry, qui est une bibliothèque pour le moteur pas à pas 28BYJ-48 5v et fonctionne avec la carte de commande ULN2003 qui correspond à notre matériel. Nous avons apporté quelques modifications pour l'adapter à notre projet.

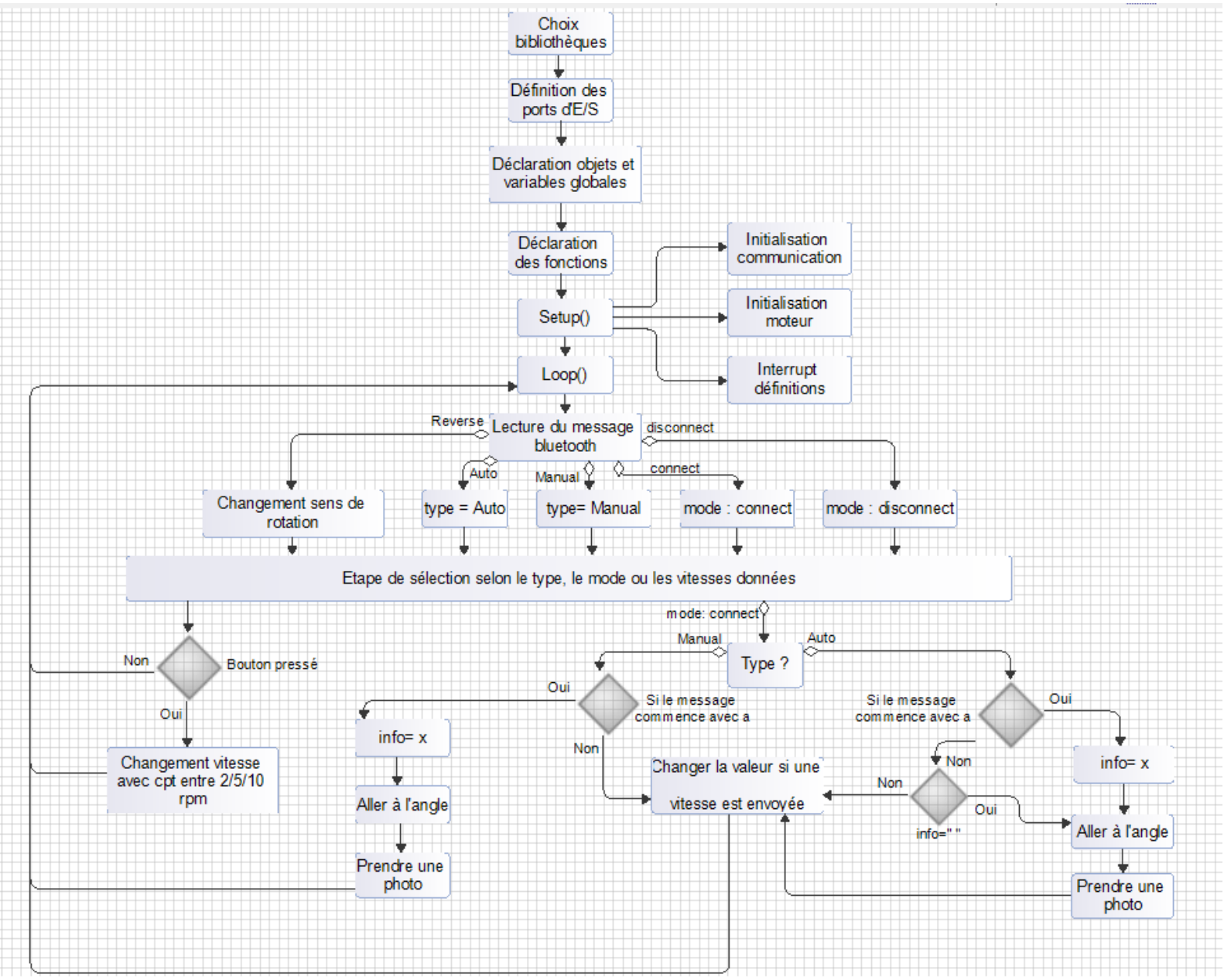
3. Réalisations :

Dans cette partie nous allons expliquer la réalisation du projet. C'est-à-dire nous allons décrire le fonctionnement de notre code dans cette étude. Ses différents buts et fonctions.

Ensuite nous expliquerons la démarche que nous avons eue pour la réalisation du contenant pour ce prototype et pourquoi nous l'avons fait de telles manières.

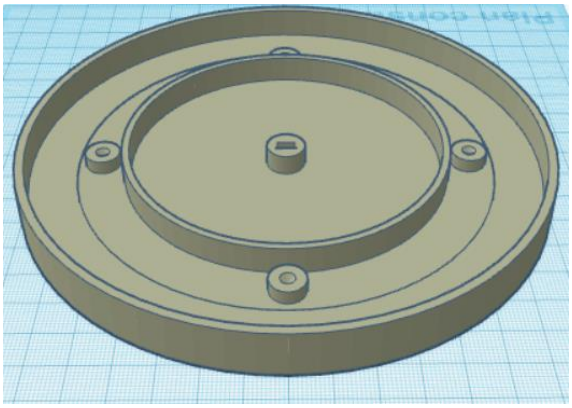
3.2. Principe de Fonctionnement :

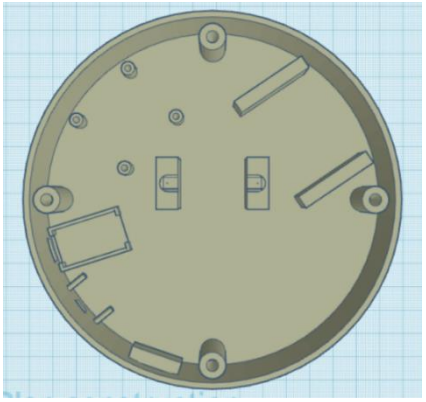
Tout d'abord commençons par un organigramme du code :



3.3. Forme Final :

Nous nous sommes réparti les tâches pour finaliser le projet. Mathieu s'est chargé de la modélisation 3D de la boîte et de la programmation Arduino, quant à Quentin, il s'est occupé de souder les composants entre eux et du code de l'application Android. Pour la forme finale du plateau nous avons décidé de faire une boîte ronde de 14.5 cm de diamètre et environ 3cm de hauteur selon le couvercle. La boîte est assez ressemblante comparée à des plateaux déjà existants, dans laquelle nous avons réussi à y incorporer tout notre matériel. Ce qui suit maintenant seront des images de la boîte en 3D numériques et ensuite réels :





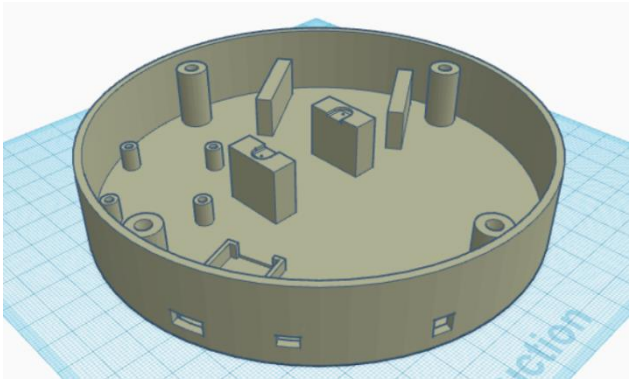
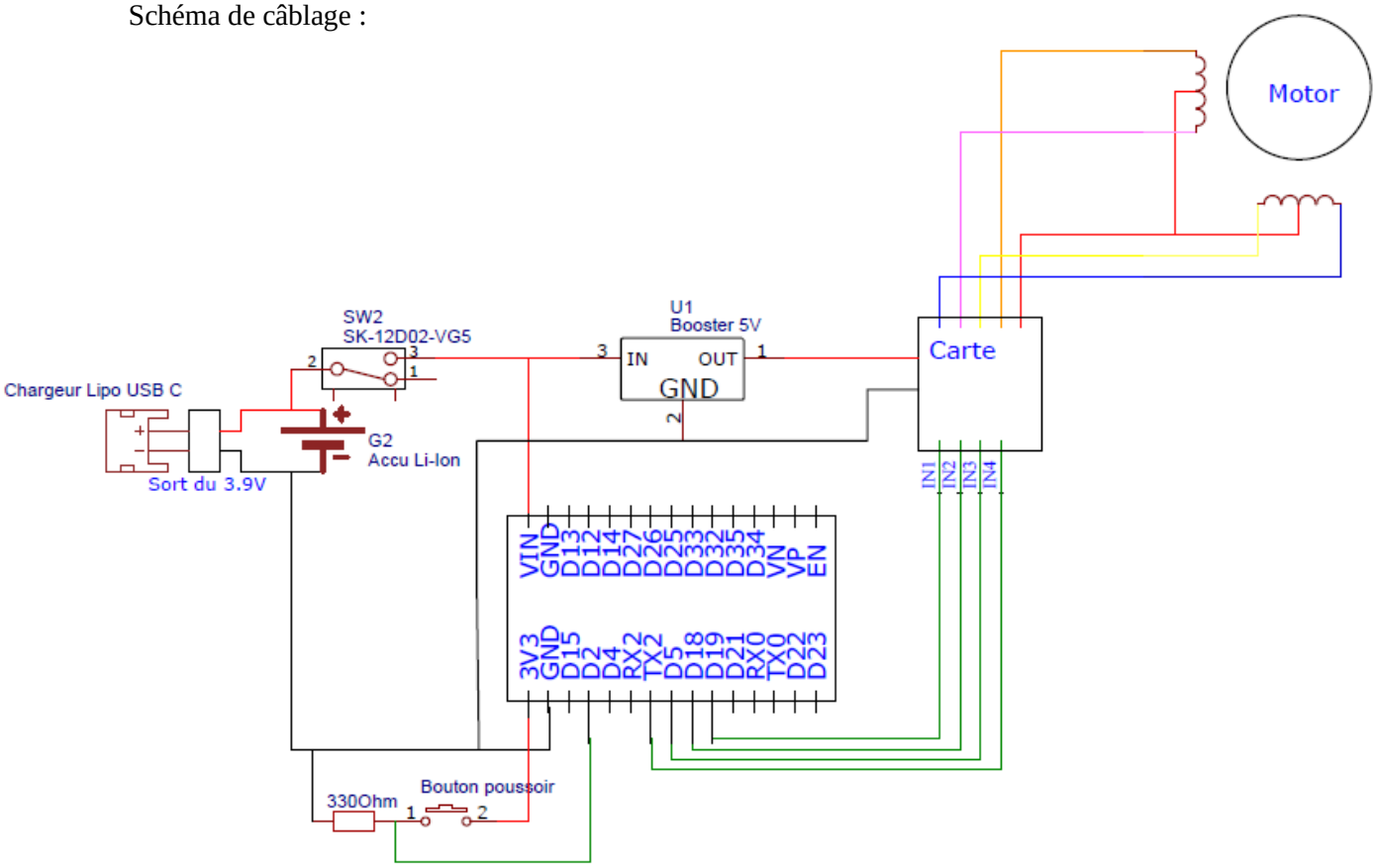
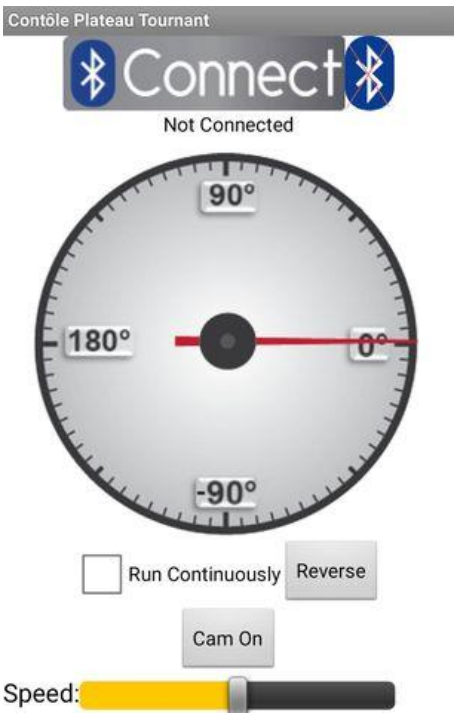
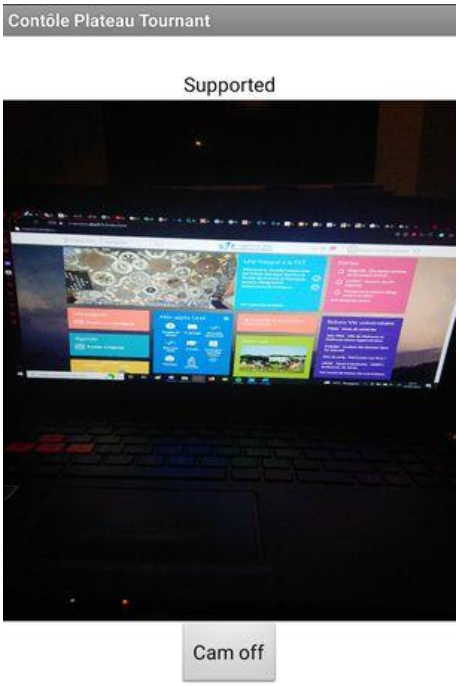


Schéma de câblage :



Application :





(Appareil photo ouvert)

3.4. Code et application :

Comme nous l'avons expliqué nous utilisons la communication Bluetooth qui nous permet d'envoyer ou recevoir des messages de l'application vers l'ESP. Selon le type de message reçu de l'application nous allons effectuer l'action qui y est associé dans le code.

Pour cela nous utilisons des conditions qui nous permettent de différencier les différentes actions à réaliser. Pour ce qui est de l'envoi de messages de l'ESP vers l'application nous les utilisons pour la prise de photo.

Nous pouvons également faire fonctionner le plateau sans connexion Bluetooth en utilisant le bouton sur le pourtour du plateau qui nous permettra comme vu précédemment de modifier la vitesse selon 3 valeurs différentes (5tr/min vitesse initiale, 2 tr/min et 10 tr/min).

Pour créer notre application App Inventor nous avons dû gérer dans une première fenêtre (Designer) la partie esthétique (l'interface) de l'application puis dans une seconde fenêtre (Blocs) la partie programmation des blocs.

Dans la partie Designer, nous avons défini un titre mais surtout nous avons créé cinq boutons :

- Deux boutons, un pour la connexion en Bluetooth et l'autre pour la déconnexion ainsi qu'un affichage de l'état de connexion.
- Un bouton pour changer le sens de rotation du moteur
- Deux boutons, un pour ouvrir l'appareil photo de l'application et l'autre pour le fermer

Nous avons également ajouté un sélecteur d'angle, une checkbox permettant d'activer ou désactiver le mode automatique ou manuel ainsi qu'un curseur permettant de changer la vitesse de rotation.

Dans la partie Blocs, nous allons programmer tous les éléments sur lesquels on va agir au niveau de l'écran. On va donc dire ce qu'il va se passer quand on appuiera sur les boutons, lorsque l'on fera glisser le curseur.

Dans un premier temps nous avons d'abord dû gérer la communication Bluetooth en ajoutant un client Bluetooth à l'application avec un sélectionneur de liste nous permettant, lorsque l'on va cliquer dessus de lister tous les périphériques qui ont déjà été appairé à notre smartphone.

Le fonctionnement est assez similaire pour nos différentes actions. Nous indiquons quand agir (lorsque l'on clique, lorsque l'on enfonce, lorsque l'on change une valeur) en fonction de l'action on décide de ce que doit faire cette action grâce au client Bluetooth à savoir si l'on doit se connecter ou se déconnecter, indiquer une position, envoyer un message, etc….

Grâce à l'envoie de message fait par la fonction Bluetooth "SendText" nous pouvons envoyer quoi faire à l'ESP et selon cette information l'ESP sera capable de savoir quelle action effectuer.

Pour la prise de photo il nous faut définir quelle caméra utiliser (la frontale, la caméra arrière ou celle par défaut), et définir le format d'image de la photo ainsi que quoi faire de cette photo. Dans notre cas il s'agit de l'enregistrer à un endroit précis que nous définirons également.

Pour la prise de photo en synchronisation avec les arrêts du moteur nous utilisons une horloge en chronomètre avec la possibilité de recevoir des messages. L'ESP enverra à l'application le moment ou le moteur est à l'arrêt ce qui déclenchera la prise de photo.

Pour le mode manuel/automatique nous utilisons une checkbox. Donc, si la case à cocher est cochée, nous enverrons le texte « Auto » à l'Arduino qui activera le moteur pas à pas pour tourner en continu et nous permettra si une valeur d'angle est saisie de tourner tous les x degrés en marquant une pose de 5 secondes à chaque fois pour laisser le temps à l'application de faire une photo. Si nous décochons la case à cocher, nous reviendrons en mode manuel en envoyant le texte « Manual ».

Pendant que nous sommes dans un de ces deux modes si nous appuyons sur le bouton "Reverse", nous enverrons le texte "Reverse" à l'Arduino qui changera le sens de rotation du moteur. De plus, pendant que nous sommes dans l'un des deux modes, nous pouvons modifier la vitesse de rotation. Si nous changeons la position du curseur, la valeur actuelle de la position du curseur sera envoyée à l'Arduino qui modifiera la vitesse de rotation du moteur (nous ne prenons en compte que la dernière valeur obtenue de la chaîne de caractère convertie en valeur arrondie).

Pour la saisie d'angle, nous utilisons une image qui nous servira de pointeur (l'aiguille dans notre cas) qu'on va faire pivoter, nous utilisons la fonction ImageSprite « .PointInDirection » qui fait pivoter l'image de la position 0° aux coordonnées X et Y où le canevas a été touché. Après cela, nous appelons une procédure personnalisée, ou une fonction qui est en fait un délai de 20 milli secondes. Si nous changeons la position du pointeur, la valeur actuelle des coordonnées du pointeur sera envoyée à l'Arduino grâce au message « ax » avec x l'angle à faire au moteur.

4. Bilan et perspectives :

Nous avons choisi de créer un plateau tournant pour la prise de photo fonctionnant principalement en Bluetooth avec une application mobile de type Android. Cette application nous permet de contrôler le plateau mais aussi la prise de photo. Nous avons également ajouté la possibilité de faire varier la vitesse manuellement à l'aide d'un bouton poussoir comme il nous a été demandé et ajouté la possibilité de couper toute l'alimentation du plateau à l'aide d'un interrupteur. Pour cela nous nous sommes fixé un budget initial de 80€ et nous voulions faire un prototype le plus ergonomique possible.

En ayant ces points en tête nous avons eu au total une commande de 76,25€ pour l'ensemble de nos composants qui regroupent la partie alimentation, connectique, commande et communication du plateau.

L'impact de notre projet est d'amener un côté pratique à la prise de photo en particulier les photos à 360° qui nous permet de capturer et de présenter des produits sous différents angles. Pour cela nous avons un fonctionnement automatique qui nous permet la prise de photo en autonomie à un angle donné.

Au final nous avons encore un certain nombre d'améliorations qui peuvent être apportées et nous allons n'en citer qu'un certain nombre. La première amélioration est sans aucun doute le fait de pouvoir avoir une version de l'application sous Apple/IOS. La deuxième serait la possibilité de pouvoir voir la led de charge de la batterie sans avoir besoin d'ouvrir le couvercle. La troisième est possiblement un ajout d'un bouton poussoir qui permet la rotation selon des angles prédéfinis (45,90,180 et 360°). La quatrième serait d'améliorer le soft pour pouvoir envoyer de longs messages et extraire les informations souhaitées.

4.1. Conclusion :

GROSSARD Mathieu :

Pour moi ce projet nous a apporté une vraie expérience sur un travail d'équipe et de collaboration. Puisque pour démarrer nous avons dû réfléchir entre nous de ce que nous voulions réaliser en respectant le cahier des charges ; que ce soit pour les déplacements selon les angles ou encore sur la possibilité de changement de vitesses et tout le reste.

Ensuite nous avons chacun fait de notre côté sur les logicielles. D'une part l'un créait l'application mobile avec toutes les fonctionnalités et l'autre s'occupait de créer le soft pour pouvoir contrôler notre moteur pas à pas comme nous l'entendions. Une fois cela fait il ne nous restait plus qu'à mettre les deux projets en commun pour que la communication entre la carte de contrôle et l'application puisse se dérouler comme souhaitée.

Pour ce qui est de la création en 3D et de la soudure cela était un travail mis en commun pour que l'on soit tous les deux d'accord avec la forme et les soudures a effectué. Nous avons aussi eu certains liens avec notre cursus puisque nous avons dû répondre à la fois à des réponses mécaniques pour correspondre au cahier des charges mais aussi à la partie sur la programmation et électroniques pour finaliser le fonctionnement en choisissant les composants nécessaires.

SCHULTZ Quentin :

Ce projet a été très formateur, il a permis de mobiliser des connaissances acquises au cours de notre formation et d'explorer de nouveaux domaines non enseignés comme avec la création d'une application Android et des différentes possibilités qu'offre le site que nous avons utilisé.

Il m'a également permis de mieux planifier et partager les tâches à effectuer pour être le plus productif possible. Les différentes recherches que nous avons faites ont été très enrichissantes. Cela nous a permis de prendre conscience de l'importance de considérer les diverses parties prenantes lors de la réalisation d'un projet et les réalisations du code et de la modélisation de la boîte ainsi que le raccordement avec les soudures.

Dans l'ensemble, j'ai pu complémenter mon savoir-faire sur la programmation et intégrer de nouvelles connaissances notamment sur la programmation en blocs et la création d'une application mobile.

4.2. Lexique :

Volt : Le volt (symbole : V) est une unité, de force électromotrice et de différence de potentiel (ou tension) et dérivée du SI.

Ampère : L'ampère (symbole A) est l'unité de mesure du Système international d'unités de l'intensité du courant électrique. Un courant d'un ampère correspond au transport d'une charge électrique d'un coulomb par seconde à travers une surface.

E/S : E/S correspond à entrer et sortie, ici cela correspond au différent pin de l'Arduino qui seront déclaré comme une entrée d'informations ou une sortie d'informations.

RAM : RAM (Random Access Memory) correspond à la mémoire vive, qui est la mémoire informatique dans laquelle peuvent être enregistrées les informations traitées par un appareil informatique.

EEPROM : abréviation de electrically erasable programmable read only memory.Type de mémoires électroniques mortes reprogrammables, effaçables électriquement.

Accu Li-Ion : La batterie lithium-ion est un accumulateur électrochimique qui utilise le lithium sous une forme ionique.

Ah : L'ampère-heure (symboles : Ah) est une unité de charge électrique. C'est la quantité de charges (portées par les électrons) traversant la section d'un conducteur parcouru par un courant d'intensité de 1 ampère pendant 1 heure.

Kb : Un kilobit est une unité d'information (symbole kbit ou kb). La valeur standard d'un kilobit est 10^3 bit = 1 000 bit. Les kilobits sont généralement utilisés pour exprimer la vitesse de communication numérique

Batterie Lipo : Les batteries LiPo sont des accumulateurs électrochimiques dont la réaction est basée sur le lithium non pas à l'état ionique mais à l'état de polymère (l'électrolyte est sous forme de gel)

Driver : C'est un ensemble qui permet de convertir des signaux électriques pour que le moteur puisse les interpréter.

Nm : Le Newton-mètre, unité de mesure, noté N m, qui traduit le moment de force, c'est-à-dire la possibilité pour une force donnée de faire tourner un système mécanique autour d'un pivot. Un newton-mètre correspond donc à une force de 1 newton appliqué à un bras de levier mesurant un mètre.

4.3. Webographie :

Réalisé par Dominique Meurisse (MCHobby) mercredi 11 juillet 2018 Arduino, Tutoriel

https://arduino103.blogspot.com/2018/07/mit-app-inventor-creer-facilement-des.html, MIT APP INVENTOR - CRÉER FACILEMENT DES APPLICATIONS ANDROID -OU- CONTRÔLER UN CROQUIS ARDUINO VIA BLUETOOTH

Réalisé par KUMARASWAMY_B.G, 1 juin 2021

https://community.appinventor.mit.edu/t/pro-camera-the-pro-custom-camera/25353?page=2,Pro camera: The pro custom camera

Réalisé par Tyler Henry, 22 juillet 2019

https://github.com/tyhenry/CheapStepper, CheapStepper

Réalisé par Rui Santos, 14 août 2018

https://randomnerdtutorials.com/installing-the-esp32-board-in-arduino-ide-windows-instructions/, Install the ESP32 Board in Arduino IDE

Fiche technique de Joy-It

https://www.gotronic.fr/pj2-sbc-moto1-fr-1519.pdf, Manuel d'utilisation du module driver

moteur pas-à-pas STP01

Réalisé par Arduino, 23 mai 2022

https://docs.arduino.cc/learn/electronics/stepper-motors/, Arduino and Stepper Motor Configurations.

Réalisé par circuito.io team, 20 février 2018

https://www.hackster.io/circuito-io-team/automatic-360-photography-turntable-586b1e, Automatic 360° Photography Turntable

5. Annexe

https://drive.google.com/drive/folders/1Xmg3eg1IbklL_H9qgbeC2R4ym0wFXmz-

5.1. Code du projet

5.1.1. Code Arduino

```
#include "BluetoothSerial.h"
#include <CheapStepper.h>
#include <String>

// next, declare the stepper
#define IN1 19
#define IN2 18
#define IN3 5
#define IN4 17
//pin bouton
#define BP 2

CheapStepper stepper (IN1,IN2,IN3,IN4);

BluetoothSerial SerialBT;

 // let's also create a boolean variable to save the direction of our rotation
 // and a timer variable to keep track of move times

bool moveClockwise = true;
unsigned long moveStartTime = 0; // this will save the time (millis()) when we started each new move
String message=" ";
String info = " ";
String type = "Manual";
String mod = "Disconnect";
String a = "a";
int stepsLeft;
int cpt = 2;
volatile bool bouton = false;

/*****************************************************************************
 * INTERRUPT ROUTINES
 *****************************************************************************/
void BP_Function(){
  bouton = true;
}

void setup() {

  // let's run the stepper at 12rpm (if using 5V power) - the default is ~16 rpm

  stepper.setRpm(cpt);

  SerialBT.begin("ESP32");
  Serial.begin(115200);

  //Bouton poussoir déclaration
  attachInterrupt(digitalPinToInterrupt(BP), BP_Function, FALLING);
```

```
  // let's print out the RPM to make sure the setting worked

  stepper.newMoveTo(moveClockwise, 2048);
  moveStartTime = millis(); // let's save the time at which we started this move

}

void loop() {
  stepper.run();
  if (SerialBT.available()) // réception d'un message
  {
    stepper.stop();
    message = SerialBT.readString(); // lecture du message reçu
    message.trim();
    Serial.println(message);
  }
  if ( message == "Reverse")
    moveClockwise = !moveClockwise;
  else if (message == "Auto")
    type = "Auto";
  else if (message == "Manual")
    type = "Manual";
  else if (message == "Connect")
    mod = "Connect";
  else if (message == "Disconnect")
    mod = "Disconnect";
  if (mod=="Connect")
  {


    if (type == "Auto")
      {
        if (message.startsWith(a)==true)
        {
          message.remove(0,1);
          if (message.startsWith("-")==true)
          {
            moveClockwise = !moveClockwise;
            message.remove(0,1);
          }
          info = message;
          SerialBT.println(info);
          message = " ";
        }
        if (info!=" ")
        {
            stepper.moveDegrees(moveClockwise, info.toInt());//mouvement bloquant
            delay(1000);//pause de 1s
        }
      }
```

```
else if (type == "Manual")
{
  if (message.startsWith(a)==true)
  {
    message.remove(0,1);
    if (message.startsWith("-")==true)
    {
      moveClockwise = !moveClockwise;
      message.remove(0,1);
      info = message;
      SerialBT.println(info);
      message = " ";
      stepper.moveDegrees(moveClockwise, info.toInt());//mouvement bloquant
      delay(1000);//pause de 1s
    }
    else
    {
      info = message;
      SerialBT.println(info);
      message = " ";
      stepper.moveDegrees(moveClockwise, info.toInt());//mouvement bloquant
      delay(1000);//pause de 1s
      }
    }
}


      else
      {
        if (message != "")
          {
            message.remove(1,message.length()-1);
          }
        stepper.setRpm(message.toInt());
      }

  }
  if (mod == "Disconnect")
  {
    if(bouton == true)
    {
      cpt+=1;
      stepper.stop();
      if (cpt>=9)
        {
          cpt = 2;
        }
      stepper.setRpm(cpt);
      bouton = false;
    }
  }
```

```
  // let's check how many steps are left in the current move:
  stepsLeft = stepper.getStepsLeft();

  if (stepsLeft == 0)
  {
  // if the current move is done...
      unsigned long timeTook = millis() - moveStartTime; // calculate time elapsed since move start
      stepper.newMove (moveClockwise, 2048); // move 180 degrees from current position
      moveStartTime = millis(); // reset move start time
  }
  message = " ";
}
```

5.1.2. Code Cheapstepper.cpp (partie modifiée)

```
void CheapStepper::run(){

        if (micros() - lastStepTime >= delay) { // if time for step
                if (stepsLeft > 0) { // clockwise
                        stepCW(false);
                        stepsLeft--;
                } else if (stepsLeft < 0){ // counter-clockwise
                        stepCCW(false);
                        stepsLeft++;
                }

                lastStepTime = micros();
        }
}

void CheapStepper::stop(){

        stepsLeft = 0;
}


void CheapStepper::step(bool clockwise,bool block){

        if (clockwise) seqCW(block);
        else seqCCW(block);
}
```

```
  //////////////
  // PRIVATE //
  //////////////

  int CheapStepper::calcDelay (int rpm){

          if (rpm < 1) return delay; // will overheat, no change
          else if (rpm >= 24) return 600; // highest speed

          unsigned long d = 60000000 / (totalSteps* (unsigned long) rpm);
          // in range: 600-1465 microseconds (24-1 rpm)
          return (int) d;

  }

  int CheapStepper::calcRpm (int _delay){

          unsigned long rpm = 60000000 / (unsigned long) _delay / totalSteps;
          return (int) rpm;

  }

  void CheapStepper::seqCW (bool block){
          seqN++;
          if (seqN > 7) seqN = 0; // roll over to A seq
          seq(seqN,block);

          stepN++; // track miniSteps
          if (stepN >= totalSteps){
                  stepN -=totalSteps; // keep stepN within 0-(totalSteps-1)
          }
  }

void CheapStepper::seqCCW (bool block){
        seqN--;
        if (seqN < 0) seqN = 7; // roll over to DA seq
        seq(seqN,block);

        stepN--; // track miniSteps
        if (stepN < 0){
                stepN +=totalSteps; // keep stepN within 0-(totalSteps-1)
        }
}

void CheapStepper::seq (int seqNum,bool block){

        int pattern[4];
        // A,B,C,D HIGH/LOW pattern to write to driver board

        switch(seqNum){
                case 0:
                {
                        pattern[0] = 1;
                        pattern[1] = 0;
                        pattern[2] = 0;
                        pattern[3] = 0;
                        break;
                }
```

...

```
                case 7:
                {
                        pattern[0] = 1;
                        pattern[1] = 0;
                        pattern[2] = 0;
                        pattern[3] = 1;
                        break;
                }
                default:
                {
                        pattern[0] = 0;
                        pattern[1] = 0;
                        pattern[2] = 0;
                        pattern[3] = 0;
                        break;
                }
        }

        // write pattern to pins
        for (int p=0; p<4; p++){
                digitalWrite(pins[p], pattern[p]);
        }
        if(block){
            delayMicroseconds(delay);
        }
}
```

5.1.3. Code Cheapstepper.h (partie modifiée)

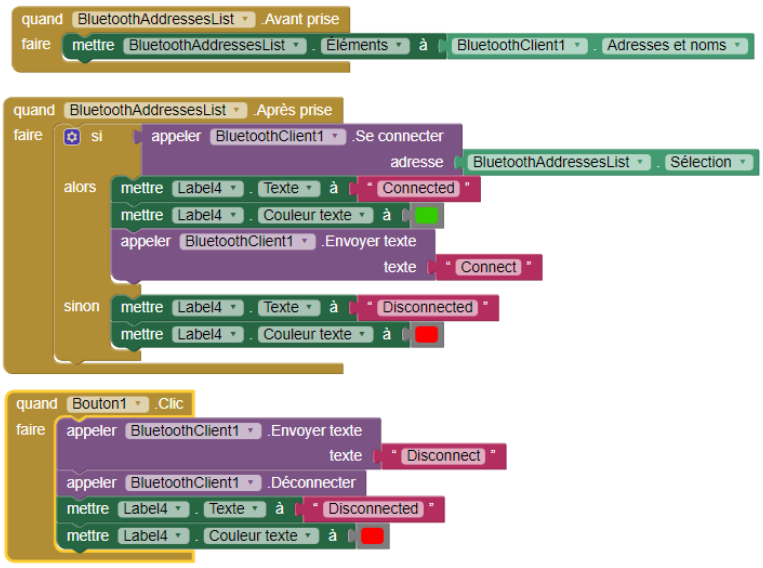
```
void step (bool clockwise, bool block = true);
// move 1 step clockwise or counter-clockwise

void stepCW (bool block=true) { step (true,block); } // move 1 step clockwise
void stepCCW (bool block=true) { step (false,block); } // move 1 step counter-clockwise

void seqCW(bool block);
void seqCCW(bool block);
void seq(int seqNum,bool block); // send specific sequence num to driver
```

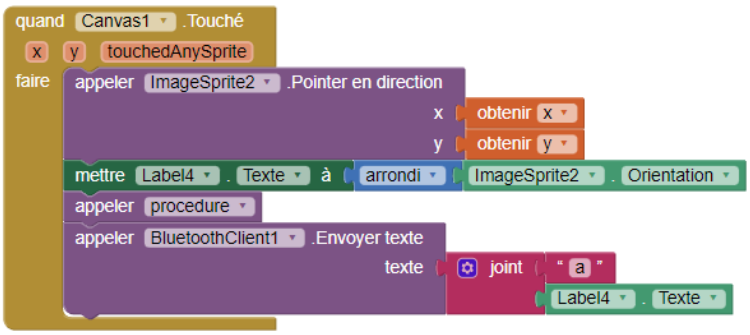
5.1.3. Code de l'application

Partie connexion Bluetooth :

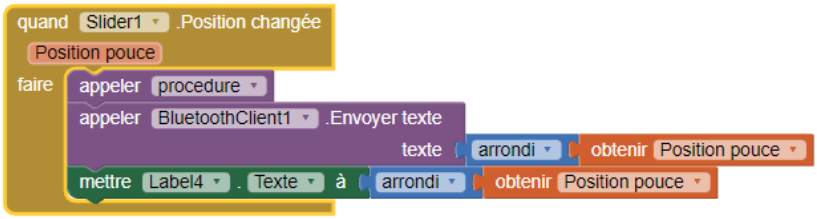


Plateau tournant

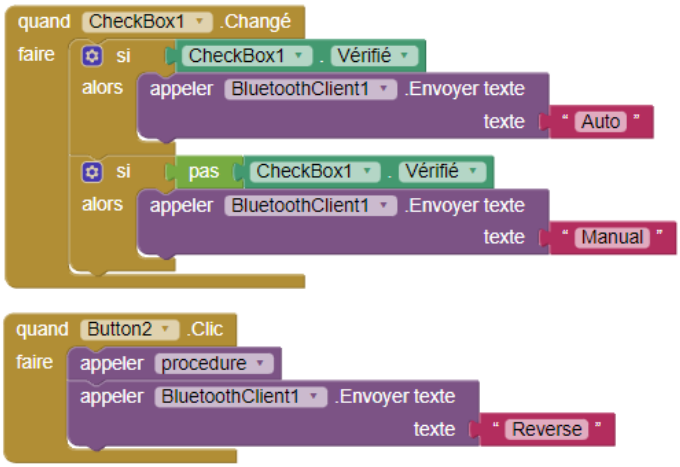
Obtention d'angle :



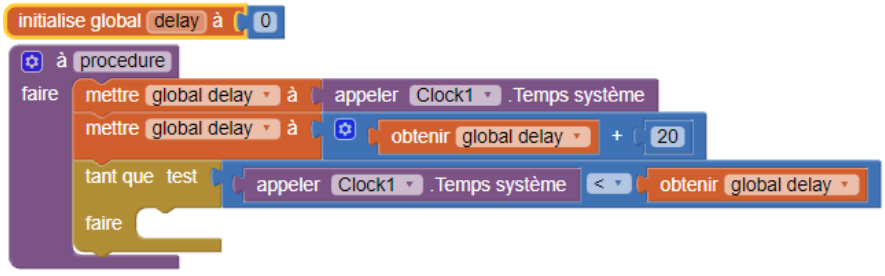
Variation de vitesse :



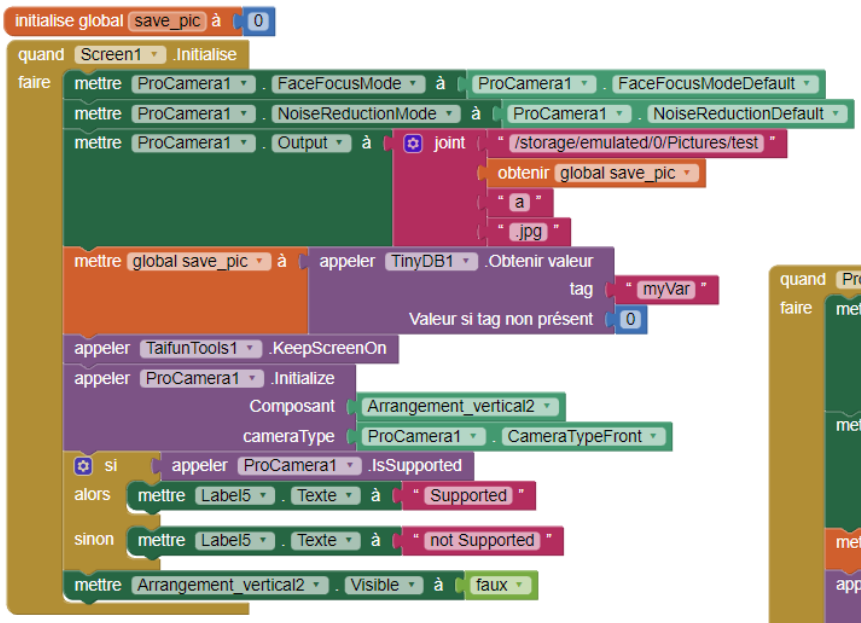
Changement de sens et de mode (manu/auto) :

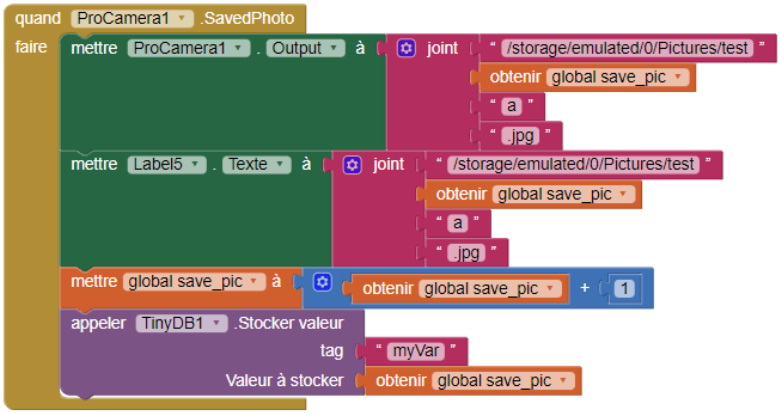


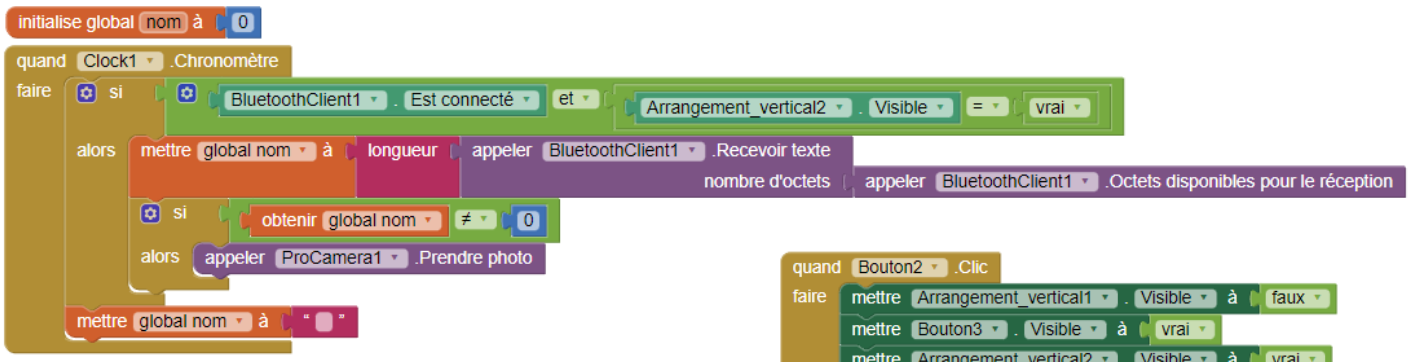
Délai :

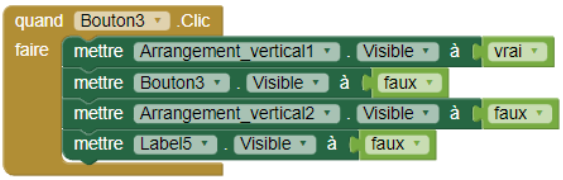


Partie gérant la photo :









Résumé en anglais :

Our project is a turntable that will be used for the photo. We will have a mobile app to communicate in Bluetooth or a button on the box to communicate manually.

We printed an ergonomic 3D box in which we have a rechargeable battery allowing to have a transportable turntable, ideal for taking pictures indoors or outdoors.

We will therefore see the various choices and reflections that allowed us to arrive at the version of our project.