Projet final

BRAS ROBOTISÉ avec CamÉra

Samuel Duclos | 247-616 Élaborer l'installation et l'intégration d'un système ordiné | 20/9/22



Table des matières

[**Description du projet** 2](#_Toc51846224)

[**Déroulement de la présentation de fin de projet** 2](#_Toc51846225)

[**Requis systèmes** 3](#_Toc51846226)

[**Requis matériels** 4](#_Toc51846227)

[**Requis logiciels** 5](#_Toc51846228)

[**Architecture système** 6](#_Toc51846229)

[**Architecture matérielle** 7](#_Toc51846230)

[**Architecture logicielle** 7](#_Toc51846231)

[**Identification et gestion des risques** 9](#_Toc51846232)

[**Planification** 9](#_Toc51846233)

[**Références** 9](#_Toc51846234)

[Liste des figures 10](#_Toc51846235)

[Liste des tableaux 10](#_Toc51846236)

Description du projet

Le projet sert à démontrer les compétences acquises durant la technique ainsi qu’à en développer de nouvelles. Il s’agit d’un système embarqué contrôlant un bras robotisé avec la rétroaction d’une caméra montée au bout afin de contrôler une suce qui attrape des objets. Le bras étant initialement dépourvu de capteur visuel, il est supposé que sa capacité à d’atteindre des objets d’atteindre des objets soit limitée dans un environnement changeant ; le but est d’y remédier. En bonus, une plus grande échelle de précision pour la préhension, la diversification des catégories préalablement définies d’objets à trier et la préservation de la durée de vie des servomoteurs en adoucissant les mouvements et en conservant l’énergie sont des buts à atteindre lorsque le projet sera bien avancé.

Déroulement de la présentation de fin de projet

La présentation débutera par un gros plan du robot sur un projecteur en arrière-plan avec une description de l’architecture matérielle et des modifications apportées. Lorsque le matériel aura été présenté, l’audience se fera demander de sélectionner un objet à trier. Le bras se fera présenter une série d’objets et devra en attraper certains mais pas d’autres. Une description de l’architecture logicielle s’ensuivra.

Requis systèmes

L’analyse de la description du projet traduit les exigences. En reflétant les compétences acquises dans la technique et peaufinées en recherche telles la conception et le montage de circuits imprimés, la planification, la réutilisation d’éléments pour simplifier la conception, la communication, la définition de la solution à entreprendre pour adresser un problème donné à résoudre et faciliter la planification. Le rapport doit être entamé d’avance et être mis à jour régulièrement afin d’assurer la bonne planification et la cohérence avec le projet fini. Le bras robotisé devra être muni d’une caméra. Un circuit imprimé connectera toutes les composantes électroniques du circuit, sauf la caméra qui est déjà montée sur un circuit optimisé pour la capture d’images. Ce système devra être une valeur ajoutée définie par une préhension et un positionnement s’adaptant à un environnement changeant. La plus grande échelle de précision, la diversification des catégories préalablement définies d’objets à trier, la préservation de la durée de vie des servomoteurs et la conservation d’énergie par adoucissement des mouvements seront en bonus pour prioriser certains éléments et assurer une marge de manœuvre dans l’exécution de la planification du projet.

REQUIS CLIENT

Buts:

* Au moins un circuit imprimé de caméra avec une image sur un Nvidia Jetson (préférablement un Nano Devkit).
* Détection d’objets arbitraire à partir de l’image reçue sur un Nvidia Jetson (préférablement un Nano Devkit).
* Commandes moteur fonctionnelles.
* Mode manuel : contrôle du bras.
* Mode manuel : feedback de la caméra.
* Rapport.

Bonus :

* Capacité à cibler, attraper et déplacer des objets arbitrairement placés.
* Auto-build et auto-deploy.
* ROS (Robot Operating System).
* Docker containers.
* Simulation sur Gazebo.
* Simulation d’apprentissage visuomoteur par renforcement.
* Apprentissage visuomoteur par renforcement dans le monde réel.
* Supporte Bash sur Ubuntu Linux.
* Bash sur Ubuntu Linux sur WSL2 (écrire une procédure d’installation pour Windows 10 afin que n’importe qui avec un GPU sur x86\_64 puisse utiliser le code).
* Bash sur Brew pour MacOS.

Les requis systèmes qui découlent de la traduction des exigences mentionnées précédemment figurent dans le tableau 1

|  |  |  |
| --- | --- | --- |
| Numéro | Nom | Description |
| 1 | Compétences en TSO | Le projet reflète les compétences acquises en Technologie de Systèmes Ordinés. |
| 2 | Compétences en recherche | Le projet reflète les compétences acquises durant le projet. |
| 3 | Bras robotisé et caméra | Le projet comporte un bras robotisé muni d’une caméra |
| 4 | Circuit imprimé | Le circuit imprimé du bras est remplacé et amélioré |
| 5 | Adaptation | Le bras s’adapte à un environnement changeant |
| 6 | Visages | Le bras détecte et attrape des photos de visages |
| 7 | La précision du bras (bonus) | La précision du bras en utilisant une caméra et de électronique plus précis est améliorée pour attraper des objets |
| 8 | Type d’objets (bonus) | Le bras peut attraper plusieurs types d’objets |
| 9 | Mouvements adoucis (bonus) | Les mouvements du bras sont adoucis |

Tableau 1: Identification et description des requis systèmes

Requis matériels

La définition du matériel est simplifiée pour permettre la mise à jour lors de l’itération visant à établir le matériel à utiliser de manière plus précise. Le *uARM* est choisi pour son accessibilité au magasin de génie électrique, pour limiter les coûts des pièces à acheter et pour réduire le temps de développement. L’électronique sera entièrement refait sur un circuit imprimé pour augmenter les capacités de calcul, de réception du capteur et de contrôle du bras. Un ESP32 sera utilisé comme élément central du centre de commandes. Une mémoire externe de huit mégaoctets accessible par SPI devra être ajoutée pour mettre les poids du réseau neuronal qui devront faire moins de quatre mégaoctets. Il est supposé qu’une caméra montée sur le bras permettra de développer une boucle de rétroaction plus précise et versatile qu’avec un capteur de proximité. Un connecteur MiniUSB-A et un circuit intégré FTDI servira à la liaison USB/série permettant la programmation du microcontrôleur. Trois servomoteurs seront contrôlés par PWM et un convertisseur analogue à digital servira à lire l’angle des flexions du bras. Un connecteur servira à brancher la caméra, trois connecteurs serviront à brancher les servomoteurs et un autre connecteur servira à brancher la pompe. Un interrupteur de fin de course servira pour déterminer la fin de succion de la pompe. Un circuit d’alimentation est prévu pour protéger les composantes. La caméra communiquera sans fil pour accélérer le flux vidéo, mais devra avoir une alimentation filaire. Trois diodes électroluminescentes serviront au débogage et indiqueront l’état du système.

Les requis matériels qui découlent des requis systèmes mentionnés précédemment figurent dans le tableau 2

|  |  |  |
| --- | --- | --- |
| Numéro | Nom | Description |
| 1 | ESP32-D0WDQ6 | Circuit intégré principal |
| 2 | MiniUSB-A (3-1734035-2) | Connecteur USB pour programmation |
| 3 | 3 servomoteurs | ADC pour lire l’angle  Sortie PWM pour contrôler les moteurs |
| 4 | FTDI (FT232RL) | Conversion USB/série |
| 5 | DC Jack (DCJ235-05-A-XX-X\_REVA) | Alimentation 3.3V/5V |
| 6 | Connecteurs (215297-4) | 3 connecteurs pour les servomoteurs  1 connecteur pour la pompe  1 connecteur pour le capteur |
| 7 | Interrupteur de fin de course | Fin de succion |
| 8 | ESP32-CAM | Contient OV2640DS |
| 9 | Diodes électroluminescentes | 3 DELs pour le débogage et l’indication de l’état du système |
| 10 | PSRAM | 8MB, SPI |
| 11 | Régulateur (LM3940IT-3.3) | Alimentation 3.3V/5V |
| 12 | 1 MP OV9281 (x2) | Modules de caméra avec obturateur global et lentilles M12 à faible distortion |
| 13 | Arducam Camarray | HAT caméra stéréo |
| 14 | Câbles de caméra | 150mm/6inch 22pin câbles de caméra |
| 15 | Câbles de caméra | 300mm/12inch 22pin câbles de caméra |
| 16 | Câbles de caméra | 73mm/2.87inch 22pin câbles de caméra |
| 17 | M2.5\*10mm nylon hex standoffs (x4) | M2.5\*10mm nylon hex standoffs |
| 18 | M2.5 nylon nuts (x4) | M2.5 nylon nuts |
| 19 | M2.5\*6mm écrous en nylon (x4) | M2.5\*6mm écrous en nylon |
| 20 | ABM8-40.000MHZ-10-1-U-T | Oscillateur crystal (40MHz) |

Tableau 2: Identification et description des requis matériel

La spécification des servomoteurs commandés par modulation de largeur d’impulsion figure dans le tableau 3. L’étendue du mouvement du servomoteur de gauche oscille entre 0° et 150°. Le servomoteur droit oscille entre 20° et 150°.

|  |  |
| --- | --- |
| Signal commande | Modulation de largeur d’impulsion |
| Fréquence de pulsation | 50Hz |
| Étendue de la largeur de la pulsation | 500 à 2500 microsecondes |
| Position neutre | 1500 microsecondes |
| Degré de course | 180°±3° (quand entre 500 et 2500 microsecondes) |
| Position neutre | 1500 microsecondes |
| Largeur de la zone morte | 2 microsecondes |
| Direction de la rotation | Sens horaire (quand entre 500 à 2500 microsecondes) |

Tableau 3: Encodage PWM pour servomoteurs

Requis logiciels

Une routine de démarrage vérifie et configure les périphériques attachés, puis le bras robotisé assume la position de départ pendant que le point d’accès Wi-Fi est initialisé avec son serveur HTTP. Une page web sur le port HTTP sert à configurer l’appareil et attend l’entrée de l’utilisateur pour sélectionner (en bonus) une catégorie d’objets à attraper. Les poids pré-entraînés sont téléchargés par le port série puis envoyés par wifi au ESP-CAM (en bonus). La page web diffuse périodiquement des images avec les boîtes de l’inférence tout en laissant disponibles les annotations pour le second ESP. Le ESP32 qui contrôle le bras se connecte au serveur HTTP pour télécharger les annotations, puis fait les calculs requis pour s’approcher le plus près possible de l’objet pendant que la boucle de rétroaction continue. Lorsqu’un signal signifiant que la suce est assez proche, la boucle de rétroaction est interrompue pour attraper l’objet et le dépose à une position prédéterminée. Le système utilise la caméra pour détecter puis attraper les objets d’une certaine catégorie mais doit ignorer les autres catégories. Le mouvement du bras peut être interrompu en tout temps pour être contrôlé manuellement.

Les requis logiciels qui découlent des requis systèmes mentionnés précédemment figurent dans le tableau 4

|  |  |  |
| --- | --- | --- |
| Numéro | Nom | Description |
| 1 | TensorFlow-Lite Micro | Simplifie le déploiement et les tests d’un réseau neuronal sur système embarqué |
| 2 | FreeRTOS | Possiblement utilisé pour une gestion simplifiée des processus et du matériel |
| 3 | ESP-IDF | Librairies disponibles pour accélérer le développement de prototypes pour l’Internet Connecté des Objets sur ESP |
| 4 | GCC | Utilisé pour le « cross-compile » d’un ESP-IDF personnalisé |

Tableau 4: Identification et description des requis logiciels

Architecture système

La figure 1 présente l’architecture système

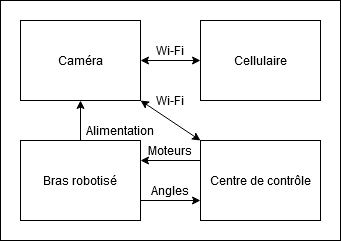


Figure 1: Architecture système

Architecture matérielle

La figure 2 présente l’architecture matérielle

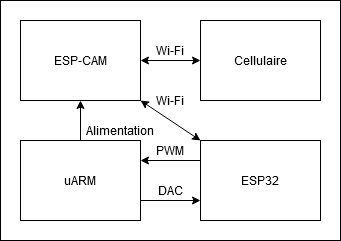


Figure 2: Architecture matérielle

Architecture logicielle

La figure 3 présente l’architecture logicielle

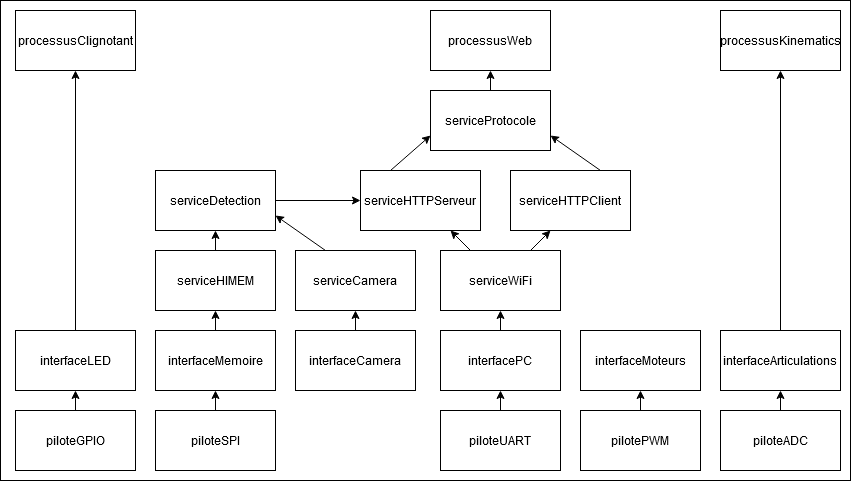


Figure 3: Architecture logicielle

La figure 4 présente l’architecture originale MobileNet

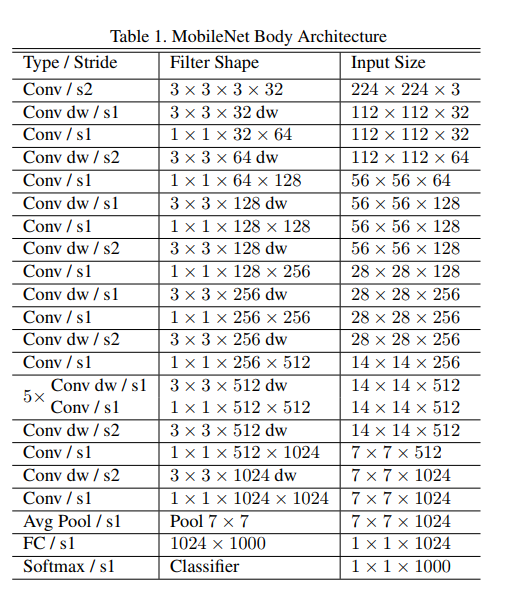


Figure 4: Architecture logicielle: réseau neuronal

Identification et gestion des risques

* Fonctionnement non complet
* Démonstration non complète
* Changement de direction
* Demander une rétroaction trop tard
* Choisir les mauvais outils
* Mauvais cas d’utilisation
* Attentes irréalistes
* Implémenter trop d’automatisation
* Bris/arrêt de fonctionnement impromptu
* Implémentation tardive de tests ou de documentation
* Maintenance ou débogage coûteux
* Mauvaise planification/gestion des risques

Planification

# Planification :

1. Requis :
   1. Système
   2. Matériel
   3. Logiciel
2. Schémas blocs :
   1. Architecture logicielle
   2. Ordinogramme
3. Fiche techniques :
   1. Comprendre le fonctionnement des éléments.
4. Montage :
   1. Schéma de connexion
   2. Faire le Montage
5. Programme initial :
   1. Pseudocode
   2. Faire les modules
   3. Tester les éléments
6. Programme principal :
   1. Faire le processus
7. Programme final :
   1. Faire fonctionner la station avec le reste du système
8. Aider mes coéquipiers à finir si je finis avant eux.

EN TOUT TEMPS :

* Accumuler des outils de conception de circuit imprimé haute-vitesse.
* Accumuler des outils de simulation
* Accumuler de la documentation.
* Prendre 15 minutes chaque jour pour évaluer ce qui pourrait avoir fonctionné ou pas et porter attention au déficit d’attention pour rectifier rapidement lorsque ça arrive. Noter les succès et les échecs pour le rapport et pour changer de stratégie régulièrement.
* Demander l’approbation du projet et prendre des notes des rétroactions des professeurs.
* Poser des questions à qui je peux pour maximiser l’apprentissage ultra-rapide.
* Tester tout ce qui génère des doutes (surtout ce qui est relié à la conception haute-vitesse).
* Sauvegarder tout partout en tout temps.

SEMAINE 1 :

* Écrire requis client (buts vs bonus). (fait)
* Activer VNC dans Docker sur ARM64v8 et sauver un écran, clavier et souris! (fait)
* Réparer bug de udev (fait)
* Planifier les prochaines semaines. (en cours)

SEMAINE 2 :

* Tester le container pour uARM dans Docker avec ROS sur Nvidia Jetson Nano. (fait, deadline lundi)
* Choisir/définir matériel et commander/emprunter le matériel. (fait, deadline mardi)
* Planifier les prochaines semaines avec des chemins alternatifs où il reste des risques et officialiser dans Trello. (en cours, deadline mercredi)
* Schématiser l’architecture matérielle dans DRAWIO et produire Bill Of Materials. (en cours, deadline jeudi)
* Tests de build/run en utilisant le même code sur ARM64v8 que sur x86\_64, avec émulation optionnelle en Docker avec QEMU. Ne pas dépenser plus de 3h sur ça, parce que ça fonctionne déjà sur ARM64v8. Ferait gagner du temps de build. (deadline samedi)

SEMAINE 3 :

* Schématiser l’architecture système dans DRAWIO. (deadline lundi)
* Schématiser l’architecture logicielle dans DRAWIO. (deadline lundi)
* Conception de la schématique du circuit imprimé. (deadline mardi)
* Description préliminaire des algorithmes utilisés (attendre avant de mettre trop de détails). (deadline mercredi)
* Schémas préliminaires des algorithmes utilisés. (deadline jeudi)
* Ajouter la caméra sur ROS. (deadline mercredi)
* Simuler le circuit électrique. (deadline jeudi)
* Étudier conception haute-vitesse (voir Texas Instruments application notes). (deadline vendredi)

SEMAINE 4 :

* Début du plan A (architecture logicielle) : tracking par sélection du canal de couleurs. (deadline lundi)
* Officialiser en réécrivant la publication (très brouillon, simple mais détaillé pour sauver du temps plus tard). (deadline mardi)
* Empreintes terminées. (deadline mercredi)
* La réception du matériel devrait débuter. (deadline jeudi)
* Tester l’entrée de la caméra. (deadline vendredi)

SEMAINE 5 :

* Continuation du plan A.
* Début de l’installation d’OpenAI Gym sur ROS. (deadline lundi)
* Début du plan B (architecture logicielle) : installer jetson-inference sur ROS et tester. (deadline mercredi)
* Conception de la disposition du circuit imprimé. (deadline vendredi)

SEMAINE 6 :

* Continuation du plan B.
* Début de l’écriture de la définition de l’environnement OpenAI Gym. (deadline lundi)
* Complétion du plan A. (deadline mercredi)
* Les pièces devraient être toutes arrivées (deadline vendredi).

SEMAINE 7 :

* Début du plan C (architecture logicielle) : étudier la conception d’environnement sur OpenAI Gym en écrivant un crypto\_bot ou stocks\_bot simple qui vend et achète automatiquement en apprenant sur le tas par renforcement. NE PAS prendre plus de 2 jours pour perfectionner le bot. Le but est d’apprendre les conventions de la librairie; pas de faire de l’argent. (deadline lundi)
* Complétion du plan B. (deadline mardi)
* Remise des fichiers de production du circuit imprimé. (deadline mercredi)
* Commander au moins 2 circuits imprimés (avoir pièces prêtes au moins en double). (deadline mercredi)

SEMAINE 8 :

* Continuer plan C : écrire environnement OpenAI Gym pour le uARM dans le Docker ROS.
* Souder un seul circuit imprimé sur 2. (deadline vendredi)

SEMAINE 9 :

* Début de la réécriture de FERM avec l’environnement OpenAI Gym pour le uARM dans container Docker sur ROS. (deadline lundi)
* Entraîner réseaux neuronaux et tester. (deadline vendredi)

SEMAINE 10 :

* Valider les réseaux neuronaux. (deadline mercredi)
* S’assurer de la cohésion matériel/logiciel avec le plan A et B. (deadline jeudi)
* Complétion du plan C. (deadline vendredi)
* Procéder aux derniers tests matériels. (deadline samedi)

SEMAINE 11 :

* Continuation et complétion de tout.
* Débogage, réglages, perfectionnement. (no deadline)
* Perfectionner publication. (deadline vendredi)

SEMAINE 12 :

* Semaine réservée aux imprévus.
* Sauvegardes partout!!!
* Temps en extra pour finaliser et nettoyer.

Références

Code à date:

<https://github.com/abstractguy/TSO_project>

Références matérielles:

<https://grabcad.com/library/uarm-metal-1>

<http://download.ufactory.cc/docs/en/uArm-Metal-Developer-Guide.pdf>

<https://www.uctronics.com/download/cam_module/OV2640DS.pdf>

<https://dl.espressif.com/dl/schematics/esp32_devkitc_v4-sch.pdf>

<https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf>

https://github.com/realizator/stereopi-kicad-schematic/

<https://www.raspberrypi.org/documentation/hardware/camera/schematics/rpi_SCH_Camera2_2p1.pdf>

https://www.raspberrypi.org/documentation/hardware/camera/

<https://www.raspberrypi.org/documentation/hardware/camera/schematics/rpi_SCH_HQcamera_1p0.pdf>

<https://github.com/RidgeRun/NVIDIA-Jetson-IMX477-RPIV3>

<https://www.buyapi.ca/product/raspberry-pi-8mp-camera-board-v2/>

https://www.uctronics.com/arducam-1mp-2-stereoscopic-camera-bundle-kit-for-raspberry-pi-nvidia-jetson-nano-xavier-nx-two-ov9281-global-shutter-monochrome-camera-modules-and-camarray-stereo-camera-hat.html

Références logicielles:

<https://arxiv.org/abs/1704.04861>

<https://github.com/espressif/tensorflow/blob/master/tensorflow/lite/micro/examples/doorbell_camera/training_a_model.md>

<https://github.com/tensorflow/models/blob/master/research/slim/nets/mobilenet_v1.py>

https://github.com/JetsonHacksNano/CSI-Camera

<https://github.com/DrYerzinia/RaspberryPiCamerav21>

<https://developer.ridgerun.com/wiki/index.php?title=Jetson_Nano/Development/Building_the_Kernel_from_Source>

http://wiki.ros.org/ROS/Tutorials

<http://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber%28python%29>

<https://www.arducam.com/docs/camera-for-jetson-nano/multiple-cameras-on-the-jetson-nano/using-arducam-camarry-on-ros/>

<https://github.com/ArduCAM/MIPI_Camera>

Liste des figures

[Figure 1: Architecture système 6](#_Toc51845922)

[Figure 2: Architecture matérielle 7](#_Toc51845923)

[Figure 3: Architecture logicielle 7](#_Toc51845924)

[Figure 4: Architecture logicielle: réseau neuronal 8](file:///U:\Avant-derniere%20session\Yves\DOC\Projet.docx#_Toc51845925)

Liste des tableaux

[Tableau 1: Identification et description des requis systèmes 3](#_Toc51846192)

[Tableau 2: Identification et description des requis matériel 4](#_Toc51846193)

[Tableau 3: Encodage PWM pour servomoteurs 5](#_Toc51846194)

[Tableau 4: Identification et description des requis logiciels 6](#_Toc51846195)