MASTER OF SCIENCE IN ENGINEERING

Hes·so
Haute Ecole Spécialisée de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences and Arts Western Switzerland

Master of Science HES-SO in Engineering
Av. de Provence 6
CH-1007 Lausanne

Master of Science HES-SO in Engineering

Orientation : Technologies de l'information et de la communication (TIC)

Interface standardisée de communication entre PLC et cartes embarquées

Fait par

Thibault Sampiemon

Sous la direction de
Prof. Yves Meyer
He-ARC(Hes-so)

Expert externe Schluep Roger Help-tec Automation AG

Lieu, HES-SO//Master, 2022

Accepté par la HES-SO//Master (Suisse, Lausanne) sur proposition de

Prof. Yves Meyer, conseiller de travail de Master
Schluep Roger, Expert principal

Lausanne, le 8 juillet 2022

Prof. Grunenwald David
Conseiller

Prof. Walther Philippe
Responsable de la filière

Table des matières

Remerciements

Je tiens à remercier l'entreprise Help-tec Automation AG de m'avoir fait confiance pour développer ce système dans le cadre de mon travail de Master.

Je remercie le directeur général et président du conseil d'administration de l'entreprise Help-tec Automation AG, Sacha Wittmann, d'avoir pris de son temps à plusieurs reprises pour s'informer du projet ou pour répondre à mes questions.

Je remercie tout particulièrement Roger Schluep, développeur logiciel au sein de l'entreprise Help-tec Automation AG, d'avoir suivi de très près le projet et également de m'avoir aidé à maintes reprises durant le projet. Je lui suis également très reconnaissant de m'avoir proposé ce travail de Master.

Je tiens à témoigner toute ma reconnaissance à Monsieur Yves Meyer, professeur au sein de la He-ARC, qui a accepté de me suivre lorsque je lui ai proposé ce travail de Master. Merci également pour son suivi hebdomadaire, mais, surtout un immense Merci pour avoir été là quand j'ai eu des problèmes et m'avoir aidé à les résoudre de la meilleure manière possible.

Je me dois aussi de remercier Monsieur David Grunenwald et Monsieur Philippe Walther qui m'ont aidé a inscrire mon travail de Master et ont accéléré certaines procédures administrative pour que je puisse effectuer ce travail.

Bien que ce ne soit pas de tradition de le faire, je tiens beaucoup à remercier Monsieur Jacques Bach, mon binôme dans les cours effectués durant mon travail de Master, pour s'être adapté à mon horaire très chargé et pour avoir effectué en ma compagnie tous les travaux pratiques avec bonne humeur et professionnalisme, malgré les horaires difficiles que je lui ai imposés.

Nomenclature

Publish	Envoi d'un message publish du protocole MQTT, ce qui correspond à une mise à jour de la valeur sur le broker lorsque le message est envoyé au broker ou l'envoi de la valeur aux subscribers depuis le broker lorsqu'une valeur a été mise à jour.
Publisher	Publisher d'une application publisher/suscriber du jargon réseau, ce qui correspond à un acteur du réseau qui envoie un message pour une mise à jour d'une valeur
Subscribe	Envoi d'un message suscribe du protocole MQTT, ce qui correspond à une demande de mise à jour lorsque la valeur change sur le broker
Subscriber	Suscriber d'une application publisher/suscriber du jargon réseau, ce qui correspond à un acteur du réseau qui veut recevoir une mise à jour d'une valeur lorsqu'elle change sur le broker MQTT
broker	Élément central d'un réseau MQTT auquel tous les acteurs du réseau MQTT sont reliés et qui maintient les valeurs des topics, les met à jour et informe les autres acteurs des changements de valeurs
Topic	Topic MQTT, une dénomination commune à tous les acteurs du réseau pour une valeur précise
QoS	Quality of Service du protocole MQTT
QoS de niveau 0	Quality of Service du protocole MQTT où il n'y aura qu'un publish sans confirmation de réception. Les liaisons MQTT avec ce type de QoS ne garantissent pas d'être mises au courant d'un changement.
QoS de niveau 1	Quality of Service du protocole MQTT où il n'y aura qu'un publish et un puback de confirmation de réception. Les liaisons MQTT avec ce type de QoS garantissent d'être mises au courant d'un changement au minimum une fois.
QoS de niveau 2	Quality of Service du protocole MQTT. Les liaisons MQTT avec ce type de QoS garantissent d'être mises au courant d'un changement exactement une seule et unique fois.
NOT	Inversion booléenne
PCB	Printed Circuit Board : désigne ici de manière générale la carte embarquée développée durant le projet
PoE	Power over Ethernet : vise à alimenter via un câble Ethernet en même temps que délivrer le signal (données)
IHM/HMI	Interface homme machine, dans ce projet un panel avec des boutons mécaniques et un potentiomètre
I/O	Input/output : entrées sorties

Abstract

Le système est un ensemble d'IHM mécaniques commandant et donnant des informations au PLC. Un réseau MQTT reliant les IHM au PLC devra également être développé. Le projet concerne le développement d'un réseau de cartes embarquées (IHM) MQTT connectées à un PLC via un switch PoE. Le software embarqué sera programmé en langage C sur une base de l'OS temps réel FreeRTOS. Le développement électronique d'un PCB de la carte embarquée comportant des boutons et LEDs gérés par des GPIO et un potentiomètre directement relié à un ESP32. Le tout est alimenté via PoE et communique avec le reste du système uniquement via la connexion Ethernet. Ces IHM seront reliées via le switch au PLC et le broker MQTT sera hébergé sur le PLC. L'attribution des IP se fera par un DHCP également sur le PLC. Un client MQTT Python (librairie MQTT Paho) modifiera des variables sur le PLC (grâce à la librairie ADS) directement sur le PLC lors d'un changement sur un topic et mettra à jour les topics en fonction des variations des variables dans le PLC. Le programme PLC (langage structuré) commandera les LEDs en fonction de la valeur des boutons. Un certain nombre de dysfonctionnements des composants du système ou du réseau MQTT sont résolus/réparés automatiquement.

Key Words: MQTT, IHM, PCB, ESP32, PoE, PLC, DHCP, ADS, Paho, Python, langage C, FreeRTOS, RTOS, langage structuré, automation, Software embarqué, Hardware embarqué, automatisation de résolution de dysfonctionnements.

1. Introduction

Le projet consiste à développer un système d'IHM pour une machine automatisée. Cette interface homme/machine comporte 8 boutons mécaniques (7 verts et 1 rouge) et parfois 1 potentiomètre.

La totalité du développement software et électronique sera fait par l'étudiant Thibault Sampiemon pour son travail de Master. Le développement mécanique se fera en parallèle en collaboration avec monsieur Roger Schluep. Ce sera également monsieur Roger Schluep au sein de Help-tec Automation AG qui suivra de près le travail et donnera des exigences supplémentaires du client durant le déroulement du travail. Monsieur Yves Meyer se chargera de suivre le projet au sein de la He-ARC et de proposer son aide à l'étudiant si nécessaire.

Toute l'analyse du produit, du travail à effectuer, de l'équipe et du management du travail se trouvent dans les documents « Management de projet » et « cahier des charges » en annexe.

1.1. Système demandé

Le système est un réseau d'interfaces comportant 8 boutons et parfois un potentiomètre. Ces interfaces seront alimentées via un câble Ethernet (PoE) qui sera également leur seule interface de communication avec l'extérieur.

Les cartes feront toutes partie d'un réseau MQTT et le développement de ce réseau et de l'acheminement des informations jusqu'au PLC feront partie du projet.

La partie software se trouvant sur le PLC sera faite sur le PC de monsieur Thibault Sampiemon. Le PLC se verra attribué un CPU et tournera sur ce même PC, il sera programmé/monitoré via Twincat car le PLC final est un PLC beckhoff. Mais un Windows 7 tournera sur l'automate, le système final ne sera donc pas si éloigné du système de développement.

1.1.1. Schéma bloc

Au niveau du Hardware, comme on peut le voir sur le schéma ci-après fourni dans les spécifications, seules les IHM (Button Panel) seront à développer. Ce développement consistera en la création à partir de zéro d'un PCB avec les boutons montés/soudés directement sur ce PCB et un potentiomètre connecté via des fils électriques au PCB.

Ces IHM seront connectées à un switch IP PoE classique via des ports PoE. Le switch sera lui-même connecté au PC via un port qui n'est pas PoE. Les connexions seront assurées par des câble Ethernet Rj45 normaux.

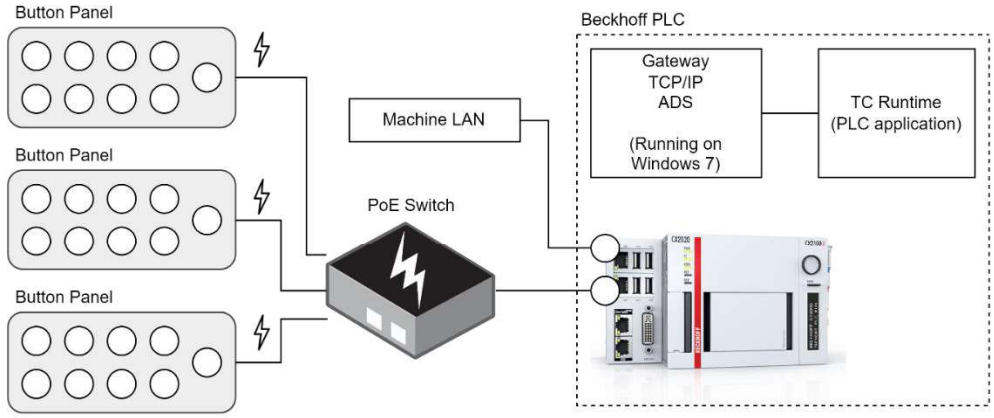


Figure 1 Spécifications du hardware

Pour ce qui est des spécifications mécaniques, deux spécifications différentes successives ont été données au cours du projet. Le schéma fourni pour la dernière version en date ci-dessous décrit les exigences mécaniques. Une autre exigence était qu'excepté la carte de pilotage Olimex ESP32-PoE-ISO qui se trouve à l'arrière du PCB, le reste des composants électroniques devait se trouver du même côté que les boutons. Comme les boutons sont soudés directement sur le PCB, cela sous-entendait que tous les composants électroniques devaient être significativement moins hauts que les boutons. Pour être plus précis, le bouton pressé devait arriver à fleur de la plaque métallique de protection placée au-dessus du PCB et les composants électroniques devaient passer entre cette plaque et le PCB. Le PCB devait également être fixé à la plaque par des vis M4 selon le schéma ci-dessous (les 5 points gris)

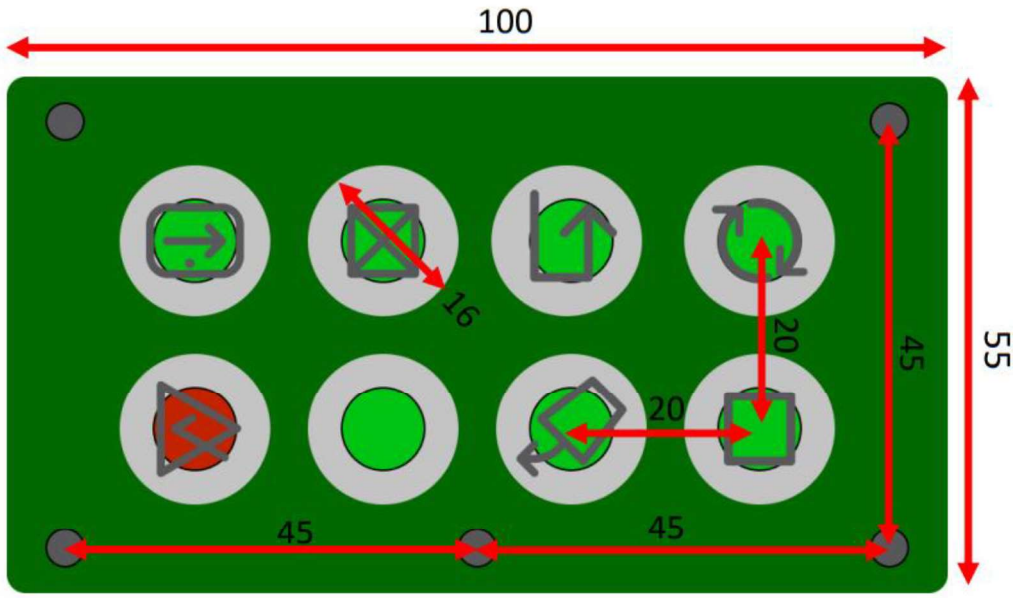


Figure 2 disposition mécanique du PCB

Du coté du software, le broker MQTT devait se trouver sur le PLC, sur un système Windows qui s'exécutait en parallèle avec le processus temps réel du PLC. Une Gateway permettant de synchroniser l'état des topics sur le broker et l'état de variables sur le PLC devait également être réalisée grâce à un client MQTT écrit en langage Python. Les cartes embarquées (Custom PCB) devaient être réalisées au niveau hardware (un PCB) et au niveau software durant le projet.

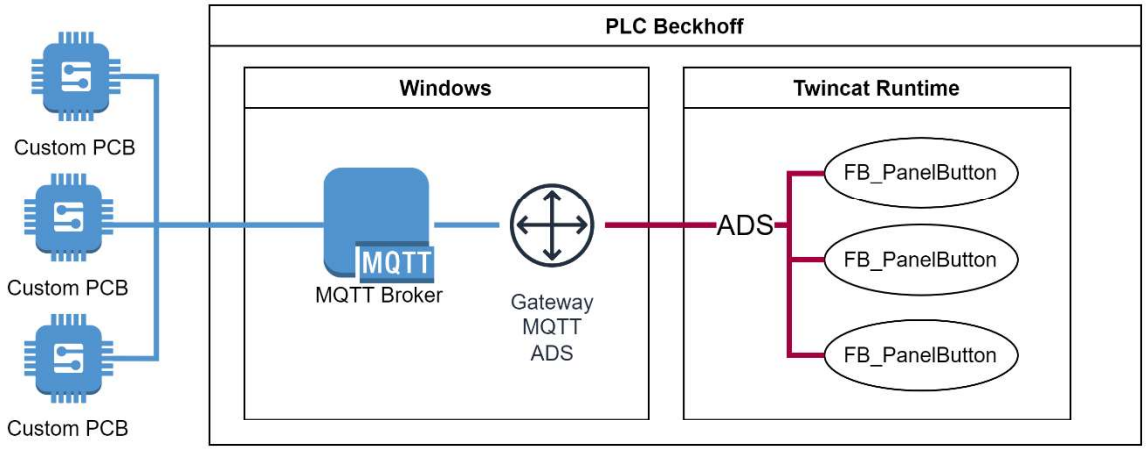


Figure 3 Spécifications du software

1.2. Solutions retenues et changements importants

La solution retenue ajoute à ces différentes parties un DHCP pour gérer les IP des cartes embarquées. Open DHCP a été retenu comme solution de serveur DHCP. Le broker, quant à lui, sera un broker mosquitto.

Un chose qui a changé depuis le premier cahier des charges est que le système PoE et l'ESP32 ainsi que toutes les fonctions utiles de la carte devaient, à terme, être intégrées au PCB. Mais, suite à des problèmes d'approvisionnement de composants (ce qui a aussi créé des problèmes pour des chips GPIO qui ont dû être changés en cour de route) dus à la crise des semiconducteurs de 2020- ?2022, il a été décidé de conserver la carte Olimex. Cela permet de laisser tous les problèmes d'approvisionnement à la discrétion d'Olimex et de délester Help-tech d'une charge de travail qui est devenue très importante les dernières années.

2. Management

La partie management a été détaillée en annexe dans le document « Management de projet » et dans le document « journal de travail » qui fournit les détails du déroulement effectif et les justifications de prolongement des tâches.

Cette section vise simplement à donner une idée générale sur la gestion de projet et les conséquences de certains problèmes rencontrés lors du déroulement du projet.

2.1. Idées générales préalables

Le client a participé à toutes les phases du projet de façon très active et beaucoup de modifications plus ou moins importantes ont été effectuées sur le cahier des charges en raison de décisions contextuelles (par exemple : crise des semiconducteur 2020- ?2022) ou techniques (par exemple l'absence de potentiomètres satisfaisants pour le client et assez fins pour se placer entre le PCB et la plaque métallique). Un retour du client sur ce qui était déjà développé a été effectué au minimum toutes les 3 semaines durant ce projet. Les objectifs finaux étaient déterminés lors de ces réunions et les solutions techniques ont quasiment toutes été choisies en concertation avec le client. Ce type de management de projet est un management Agile.

2.1.1. Gantt général du projet prévu

Ce Gantt général sert à situer temporellement l'exécution des différents work packages à l'échelle des semaines.

Pour plus de précisions ou explications sur les work packages ou le management de projet : voire le document « Management de projet » en annexe.

Tableau 1 Gantt du projet prévu avant le début du projet

Worck packages	Semaines (semaine 0 : du 21 au 28.02.22 puis incrément de 1 pour chaque semaine jusqu'au 08.07.22)
WP 1 : Hardware carte périf: Création du PCB	
WP 2 : Développement du software de la carte périf :	
WP 3 : Développement du software PLC :	
WP 4 : Test du système complet :	
WP 5 : Documentation et Rapport :	
Jalons/gates/deadline	J1 J2 J3 J4 J5

	Durée incompressible du package		Marge de manœuvre (certaines sous-tâches peuvent être commencées en amont et certaines peuvent être effectuées plus tard, sans impacter les packages dépendants)
	Semaine de pause		

Les jalons J1 à J5 représentent les dates butoirs où certaines parties du projet doivent impérativement être terminées. A chacune de ces dates butoirs, une réunion d'avancement du projet aura lieu. Les participants de ces réunions sont l'étudiant, le professeur référent et le mandant (responsable au sein de l'entreprise mandante).

Tableau 2 Jalons/deadline du projet

Jalon	Date	Titre du jalon	Objectifs du jalon
J1	21.03.2022	Fin de la conception carte prototype	Le PCB de la carte fille devra être terminé et en production.
J2	25.04.2022	Fin de la conception de la carte périf	Le hardware et le software du prototype de la carte périf devront être parfaitement fonctionnels et testés.
J3	16.05.2022	Fin de la conception du système software	Le hardware et le software du prototype du PLC devront être parfaitement fonctionnels et testés.
J4	13.06.2022	Finalisation de l'installation de démonstration	On pourra démontrer que le système composé de plusieurs cartes peut fonctionner en collaboration avec un système d'automation test. Le design hardware de la carte finale sera terminé.
J5	08.07.2022	Fin du projet	Le projet devra être fonctionnel, documenté et une intégration dans une fausse chaîne de production devra démontrer son fonctionnement.

2.1.1. Gantt général du projet effectué (difficultés, modifications)

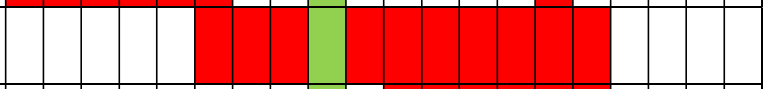
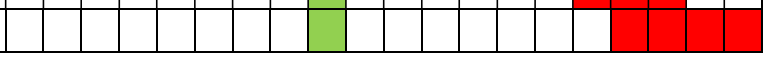
Le plan initial a pas été respecté à 100%, mais tous les jalons ont été respectés. La seule modification significative est que la partie communication MQTT de la carte a été développée et adaptée en même temps que la partie PLC. Mais il est plus logique de développer une communication de bout en bout, car dans une communication, il faut adapter la communication des deux cotés de la chaine d'information. Il est donc plus adapté de développer simultanément la partie communication sur les cartes embarquées et le PLC. De plus, 99% du développement sur le PLC concerne uniquement la communication. Cet aspect avait en effet été sous-estimé dans le Gantt initial.

Un autre changement est la prolongation du temps de développement matériel, mais cela reste dans le domaine du prévisible. Les changements dus aux reviews du client avaient été sous-estimés et suite à la crise des semiconducteur 2020- ?2022, les GPIO qui étaient encore en stock en grand nombre au début du développement étaient tombés en rupture de stock et avaient dus être remplacés.

Le dernier changement est la prolongation du temps de développement du software dû à un (voir deux) problème(s) technique(s) difficile(s) à résoudre. Le premier problème était dû a un temps de réponse du contrôleur Ethernet trop lent. Pour résoudre ce problème, trois délais ont dû être introduits dans l'initialisation (dont deux dans le driver fourni par ESP32) du contrôleur Ethernet. Ces délais n'étaient mentionnés dans aucune documentation (seul un message datant de 5 ans auparavant, mal référencé sur un forum, a été trouvé). Le deuxième était plus facile à résoudre une fois le premier résolu. Il était dû à une combinaison de problèmes venant du firewall et d'un changement de ce qui était autorisé sur la dernière version de mosquitto en termes d'authentification.

Pour plus de précisions sur les modifications du Gantt de projet, voir le document « Journal de travail » en annexe.

Tableau 3 diagramme de Gantt réel après le projet

Worck packages	Semaines (semaine 0 : du 21 au 28.02.22 puis incrément de 1 pour chaque semaine jusqu'au 08.07.22)
WP 1 : Hardware carte périf: Création du PCB	
WP 2 : Développement du software de la carte périf :	
WP 3 : Développement du software PLC :	
WP 4 : Test du système complet :	
WP 5 : Documentation et Rapport :	

	Travail effectué
	Semaine de pause

2.1.1. Améliorations

Lors de la phase de développement software, le Git n'a pas été mis à jour toutes les semaines comme demandé par le client. C'est clairement un problème de management dû au stress occasionné par les retards pris sur le planning et l'intensification des parties pratiques des cours à la fin du semestre. Pour remédier à cela, il aurait sûrement fallu inclure des deadlines pour cette partie également et calculer automatiquement des indices de mise à jour/modification par rapport à la dernière version dans la chaine de ci/cd du Git. Mais comme le projet ne s'est fini qu'un jours avant la fin prévue, le temps consacré à la chaine de ci/cd aurait pu résulter en un projet inachevé.

Une définition plus claire des reviews et une intransigeance quant au déroulement de ce processus auraient pu éviter une erreur d'inattention sur le design hardware.

3. Hardware

Cette section ne rentrera pas dans tous les détails de l'implémentation électronique. Elle a pour but de donner une compréhension générale de l'architecture du PCB.

Le développement hardware se centre sur la création d'un PCB qui fera office de carte de gestion des I/O de l'IHM et sera piloté par une carte ESP32-POE-ISO-IND de Olimex. Mais ce module embarqué devra communiquer avec le PLC (PC) via un switch et le matériel pour créer ce réseau MQTT fait aussi partie du développement hardware.

3.1. Système

Le système au niveau matériel est donc composé de :

- Un module embarqué composé lui-même de
  - Une carte ESP32-POE-ISO-IND de Olimex
  - Un PCB qui fera office de carte de gestion des I/O développé durant le projet
- Un switch comportant des ports Ethernet PoE et sans PoE
- Un PLC qui sera ici sur un PC

3.1.1. Architecture du réseau hardware

Les cartes embarquées sont des IHM munies de 8 boutons mécaniques et d'un potentiomètre. Ces IHM seront connectées à un port PoE du switch grâce à un câble Ethernet. Elles seront alimentées uniquement par le switch via PoE. Ce câble Ethernet sera la seule interface de communication entre le module embarqué et l'extérieur. Le switch est un simple switch IP avec au minimum un port Ethernet PoE par module embarqué et au moins un port sans PoE pour le PC. Le switch sera connecté au PC via un câble Ethernet sans PoE.

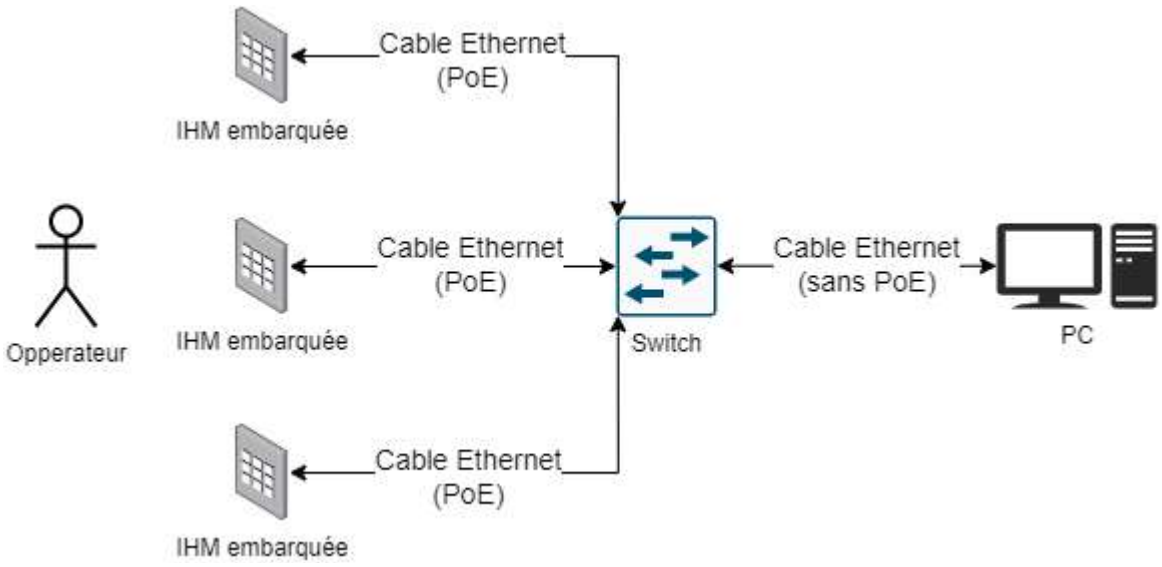


Figure 4 Schéma de l'installation hardware du point de vue réseau

3.2. Carte embarquée

Dans cette section, nous nous centrerons sur le PCB désigné durant ce projet qui est, dans les images ci-dessous, le PCB bleu. La carte ESP32-PoE-ISO est une carte développée par Olimex. Pour avoir plus de détails, la documentation de Olimex est suffisante.

La carte embarquée comporte 8 boutons qui comportent chacun une LED intégrée au centre de ce bouton. La disposition mécanique des boutons a été imposée (deux versions de cette disposition des boutons ont été imposées, une pour la première version du PCB et une pour la version finale). Pour la version finale, on peut se référer à la « Figure 2 disposition mécanique du PCB ». Les dimensions du PCB et la disposition des perçages étaient également imposés. La « Figure 5 photo de face du module embarqué » montre la version 1 du PCB avant les changements de spécifications mécaniques et électroniques. À l'origine l'électronique de la carte Olimex devait être réimplémentée sur le PCB, mais à la suite de certaines situations dues à la crise des semiconducteurs (2022), le client à préféré garder la carte Olimex. Il a été défini avec le client que cette carte sera montée à l'arrière du PCB et le reste de l'électronique à l'avant du PCB. Le potentiomètre devait, à l'origine, être soudé sur la carte embarquée, mais pour plus de flexibilité le client a préféré une connexion via des fils soudés au PCB. La valeur du potentiomètre à été conservée à 10 kOhms, mais il n'apparaitra plus de façon directe sur le PCB. Le client a pris part activement au design du PCB. Les composants et le design ont été changés plusieurs fois suite aux demandes du client. Il a également fallu changer certains composants suite à des ruptures de stocks dues à la crise des semiconducteurs (2022).

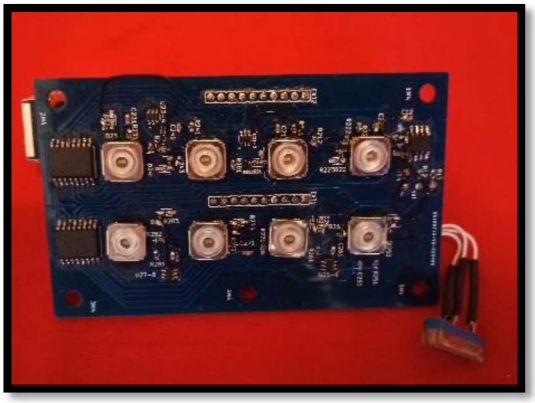


Figure 5 photo de face du module embarqué

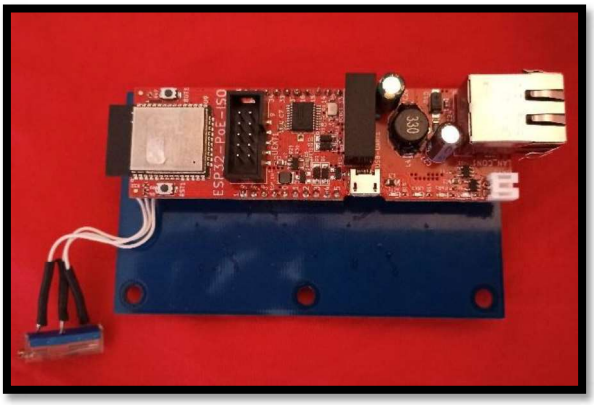


Figure 6 photo depuis l'arrière du module embarqué

3.2.1. Architecture électronique de la carte embarquée/du module embarqué

Ci-dessous « Figure 7 Schéma block de la carte » donne une idée générale du design de la carte. Quelques commentaires pour complémenter le schéma et vous aider à mieux comprendre l'architecture de la carte :

- La carte Olimex alimentée en PoE sera connectée au PCB et alimentera celui-ci.
- Bien que cela ait beaucoup varié au cours du développement, la version finale du PCB ne sera alimentée qu'en 3.3V (Vdd)

- Les deux GPIO sont branchés sur le même I2C. Ils sont branchés sur la même ligne d'interruption avec une pull-up et les chips en opendrain branchés sur la masse du chip (comme préconisé dans la documentation)
- Chaque GPIO gère 4 boutons et leurs LEDs
- L'ADC peut convertir une tension entre 0,5 et 2.45V, ce qui explique cette plage de tension sortant de l'électronique de branchement du potentiomètre

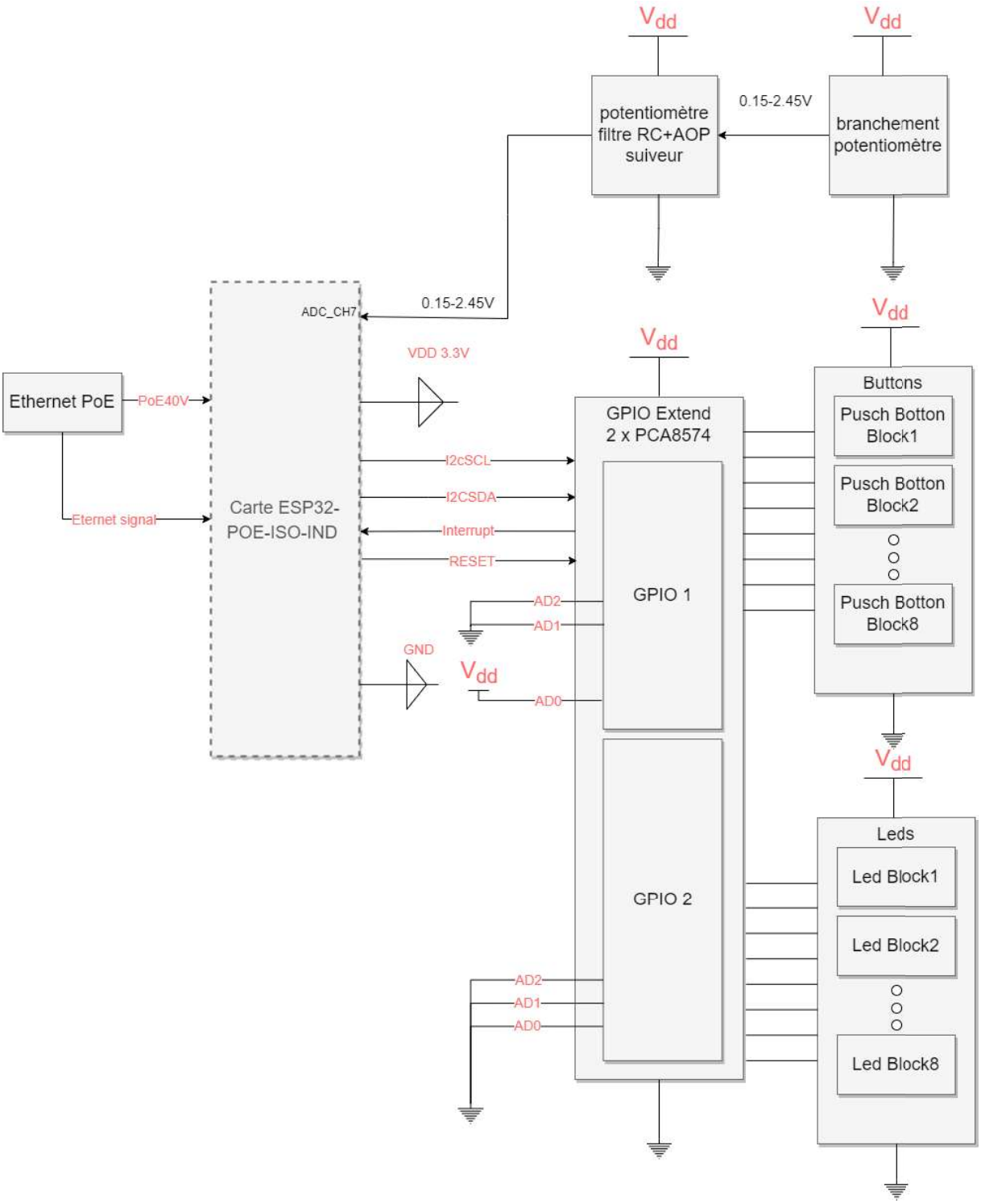


Figure 7 Schéma block de la carte

3.2.1. Idées générales concrètes d'implémentations et améliorations possible du produit

Dans cette section certaines réflexions concrètes effectuées au cours du projet seront exposées. Cette section est encore une fois non-exhaustive et représente une toute petite partie des réflexions effectuées au cours de ce projet. Seules les réflexions jugée les plus utiles à connaitre pour la suite du projet ou pour les améliorations possibles seront évoquées.

Alimentation du potentiomètre

Le branchement du potentiomètre alimente le potentiomètre de sorte que lorsque le potentiomètre est au plus bas, alors 0.15V seront envoyés dans la partie filtre etc. et respectivement au plus haut 2.45V. Cela correspond à la plage de tension que l'ADC peut convertir.

Le potentiomètre est de 10kOhm. Donc lorsqu'il est en position MIN = 0 Ohm, il est équivalent à un pont diviseur entre R14=680 Ohm et R15+Potentiomètre =3,9 kOhm + 10 kOhm = 13,9 kOhm. Cela donne donc, avec une tension d'alimentation de 3,3V : 3,3 *(680/(13900+680)) = 0,154 V, soit environ la limite basse de l'ADC (0,15V).

De même lorsque le potentiomètre est en position MAX=10 kOhm, il est équivalent à un pont diviseur entre R15=3,9 kOhm et R14+Potentiomètre = 680 Ohm + 10 kOhm = 10,68 kOhm. Cela donne donc avec une tension d'alimentation de 3,3V : 3,3 *(10,68/(10,68+3,9)) = 2,42 V, soit environ la limite haute de l'ADC (2,45V).

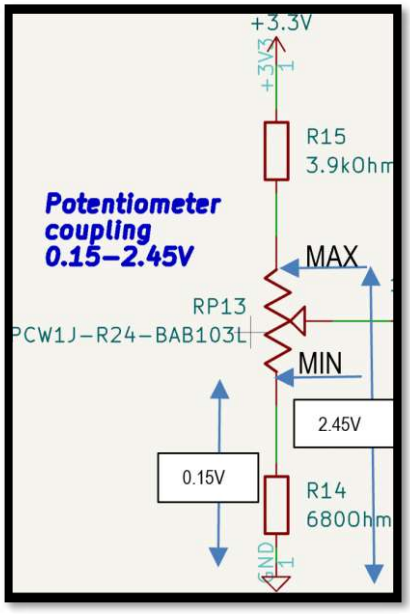


Figure 8 Schéma d'explication du branchement du potentiomètre

Ce système de pont diviseur avait du sens au début du projet. Le potentiomètre devait être soudé au PCB et donc être un modèle spécifique connu. La résolution évoquée lors d'un entretien était d'une dizaine voire d'une vingtaine de positions. Il n'y avait donc pas besoin d'être extrêmement précis et la valeur du potentiomètre était fixée. Mais à présent, le potentiomètre sera relié au PCB par des fils électriques et la précision voulue est de 100 valeurs (en pourcentage). Avec cette nouvelle situation une adaptation de la tension au niveau de l'AOP serait peut-être plus judicieuse.

Filtres RC

La sortie du bloc de branchement du potentiomètre est branchée sur un AOP en suiveur pour l'isoler électriquement du reste.

Puis la tension de sortie du suiveur passe dans un passe-bas du premier ordre (un RC). La fréquence de coupure du RC est de f = 1/(2*π*R*C)=159 Hz.

L'envoi d'un message depuis la carte jusqu'au PLC devait prendre au maximum 100 millisecondes, soit une fréquence d'envoi de messages garantie de 10 Hz minimum. Des changements avec des oscillations de la valeur du potentiomètre ne sont clairement pas utiles au-dessus de 10Hz, car le potentiomètre sera tourné par une main humaine incapable de créer ce type de variations. Donc 160 Hz nous permet de prendre des valeurs 16 fois plus fréquemment. On pourrait donc encore descendre la fréquence de coupure du filtre si nécessaire.

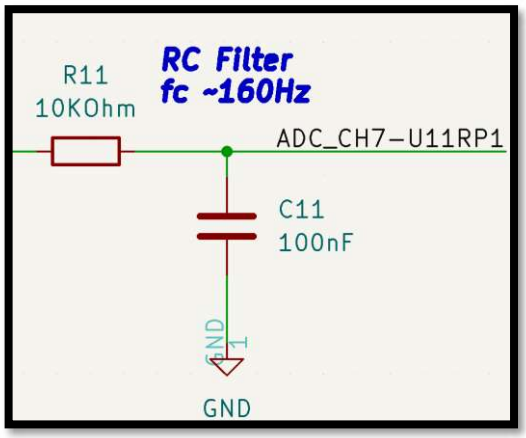


Figure 9 Schéma électronique du RC

Le même filtre a été implémenté sur la chaine de traitement de la valeur des boutons (le signal de sortie du bouton rentre directement dans le RC puis dans un trigger de schmitt et rentre ensuite dans le GPIO). De la même manière 160 appuis sur le bouton par seconde est irréalisable pour un humain. On pourrait donc encore descendre la fréquence de coupure du filtre si nécessaire.

Package des résistances et condensateurs

Un package avait été choisi pour être facile à monter à la main, ce qui n'est pas un problème car on disposait de beaucoup de place due aux spécifications mécaniques. Ce package avait un power rating typique de 0,125 W (1/8). Mais la puissance maximum dissipée par une résistance (celle en série avec la LED) est de 24mW. On pourrait donc descendre à un power rating typique de 0,031W (1/32) avec par exemple un style de package « 0402 » si dans l'avenir un besoin d'économie d'espace se faisait sentir.

3.2.2. Difficultés rencontrées

Dans la première version du PCB la Led était alimentée par une tension de 5V au lieu de 3,3V comme ci-dessous. La cathode (K) était et est toujours directement connectée à une sortie du GPIO.

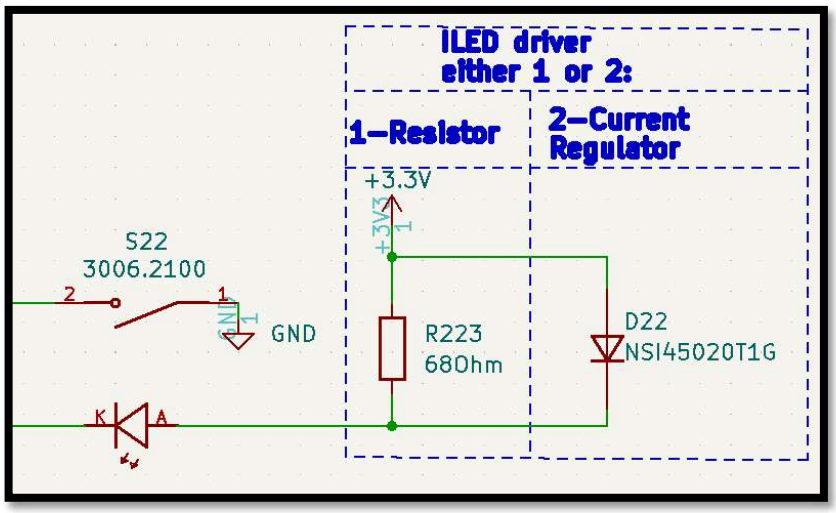


Figure 10 schéma électronique de la solution finale pour les LEDs

Le GPIO était et est toujours alimenté par du 3,3V. De ce fait, lorsque la sortie était active (1 logique) et que la LED devait être éteinte (logique inversée 0=LED allumée 1=LED éteinte) une tension subsistait entre l'alimentation de 5V et la sortie à 3,3V. Cette tension suffisait pour qu'un faible courant passe dans la LED, ce qui générait une très faible lumière dans les LEDs rouges (aucune lumière dans les LED vertes). Le problème a été réglé en descendant la tension d'alimentation à 3,3V (et en changeant la résistance de 150 Ohm à 68 Ohm).

Un autre problème vient d'une erreur d'inattention avec une masse connectée à un endroit où elle ne devait pas être connectée et qui n'a pas été remarquée lors des reviews du PCB. Imposer une checklist pour la review aurait pu être une bonne idée.

3.3. Tests

Plusieurs tests du Hardware ont été effectués.

Des tests mécaniques :

- Pour la sensation des boutons montés en mettant une étiquette en guise de membrane de protection qui a satisfait plusieurs membres de l'entreprise mandante

Figure 11 photo du test mécanique de la membrane (fausse)

- Et de montage qui a révélé que les GPIO étaient trop hauts pour un montage idéal, ce qui a résulté en un changement des GPIO vers le même GPIO mais avec un package plus fin et plus compact dans la version finale du PCB.

Figure 12 photo de la vue arrière du montage mécanique

Des tests électriques :

- Un test de consommation de courant a été effectué lorsque toutes les LEDs étaient allumées pour vérifier que l'on ne dépasse pas les limites du DC/DC ou du chip PoE.

Figure 13 photo du test de courant

- Les pistes ont été testées à l'ohmmètre pour vérifier que toutes les connexions voulues entre les composants existaient
- La tension à l'entrée de l'ADC a été comparée à la valeur convertie par l'ADC

4. Software

La partie software comprend un grand nombre de programmes indépendants écrits avec des langages différents sur des supports matériels différents. Les sous-chapitres suivants vous donneront une idée générale sur chacun d'eux, mais pour le rendre plus concis et clair, ce document n'a pas pour but d'être trop exhaustif sur les détails d'implémentations.

Les échanges de paquets de la communication MQTT entre le broker et la carte embaquée sont décrits dans la section Broker. L'échange entre le client python MQTT et le broker est décrit dans la section client MQTT de synchronisation.

4.1. Vue générale de l'interaction des différents programmes

Les programmes se trouvent sur deux types de hardware :

- Les cartes embarquées composées d'un PCB développé dans le cadre du projet et d'une carte embarquée OLIMEX (le soft se trouve sur les cartes finales dans le cadre de ce projet).
- Le PLC, mais qui dans le cadre de ce projet a été développé sur un PC avant de le migrer ultérieurement sur un PLC.

Les programmes développés et configurés sur le PC (PC sur le diagramme ci-dessous) sont :

- Un broker MQTT (broker MQTT sur le diagramme ci-dessous)
  - Le broker utilisé est mosquitto
  - La configuration se trouve sur un fichier de configuration
  - Le seul changement à effectuer pour adapter le système à la machine finale est de rajouter un nom d'utilisateur et un mot de passe par carte embarquée via les lignes de commande de mosquitto sur le fichier dédié.
- Un serveur DHCP (DHCP sur le diagramme ci-dessous)
  - Le serveur DHCP utilisé est OpenDHCP
  - La configuration se trouve sur un fichier de configuration
  - Aucune modification ne devrait être nécessaire à part changer l'IP fixe de contact du DHCP
- Client Python MQTT de mise à jour du PLC (Client MQTT synk sur le diagramme ci-dessous)
  - Utilise Paho et ADS
  - Il faut changer le nom des variables en fonction des variables définies dans le PLC
  - Il faux changer le nom des topics MQTT pour qu'ils correspondent au nom des topics définis sur la carte embarquée

Module embarqué (IHM embarquée sur le diagramme ci-dessous)

- Implémenté sur FreeRTOS
- Avec la librairie ESP32 (notamment pour la communication MQTT)

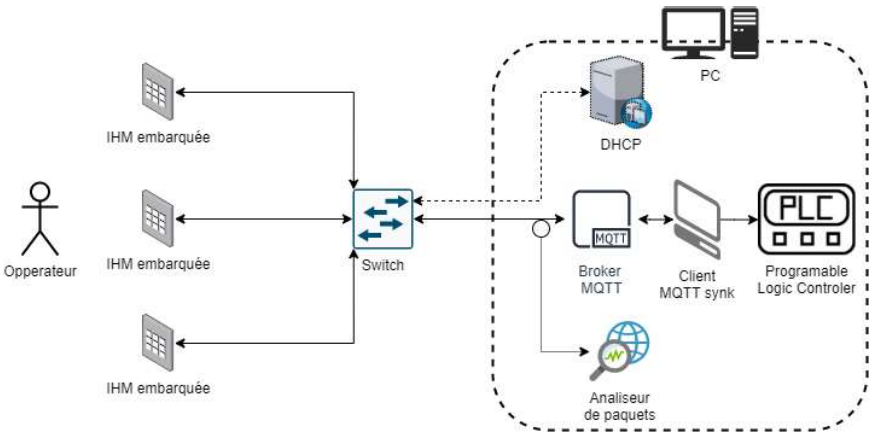


Figure 14 Architecture réseau du point de vue software

4.2. Carte embarquée

Il y a trois parties principales au software embarqué :

- La partie de gestion des boutons et des LEDs intégrées à ces boutons
- La partie d'échantillonnage/mise à jour/traitement des valeurs venant de l'ADC exprimant la position du potentiomètre
- La partie de gestion de la communication MQTT avec le broker

L'OS sur le module embarqué est FreeRTOS.

La priorité des tâches a été déterminée en fonction de la priorité des traitements et de la dépendance des tâches entres elles.

Type de tache		Tâches	Priorité
Tâches de traitement des données venant des GPIOs	Le traitement des entrées est prioritaire sur tout le reste car ce sont les informations que l'opérateur veut envoyer au PLC.	Tâche de traitement des données venant des Boutons	4
		Tâche de traitement des données venant du potentiomètre	3
Tâche de mise à jour des LEDs	Le traitement des sorties vient juste après, car c'est le retour que le PLC envoie à l'opérateur sur les conséquences ou sur la réception des informations des commandes envoyées par l'opérateur.	Tâche de mise à jour des LEDs	2
Tâche de communication avec le broker MQTT	La priorité basse de la tâche de communication est due au fait qu'elle dépend de la tâche des boutons. Les traitements importants sont faits suite au traitement des boutons.	Tâche de communication avec le broker MQTT	1

Tableau 4 Priorité des tâches (FreeRTOS)

4.2.1. Boutons & LEDs

Deux taches gèrent cette partie :

- Une tâche gère le traitement et la prise des valeurs des boutons via I2C et est déclenchée par une interruption lancée par un des deux GPIOs
- Une tâche gère l'allumage des LEDs et est déclenchée par en événement venant de la boucle de traitement événementielle (« event loop») de la pile MQTT de ESP32

Cette partie est également en étroite communication avec la tâche qui gère les communication avec le broker MQTT(Tâche d'envois/réceptions de messages).

Dans le diagramme ci-dessous on peut voir une communication classique du point de vue du module embarqué.

1. Un bouton est pressé.
2. Le GPIO du bouton détecte le changement de l'entrée.
3. Le GPIO envoie un signal d'interruption sur une pin de l'ESP32.
4. Le changement sur la pin déclenche une interruption et le handler est lancé.
5. Le handler envoie un événement (change un eventgroup flag) FreeRTOS.
6. La tâche de traitement des valeurs des boutons qui attendaient ce flag demande la valeur des deux GPIO via I2C
7. Les GPIOs lui renvoient la valeur des I/O et la tâche transforme ces données en un byte (un flag pour l'état de chaque boutons).
8. La tâche envoie ces données via une file de messages à la tâche de gestion des communications avec le broker.
9. La tâche de gestion des communications avec le broker publish cette donnée.
10. La donnée est envoyée jusqu'au PLC via la chaine de communication.
11. Le PLC demande d'allumer la LED de ce bouton et la valeur voulue des leds revient au module embarqué via la chaine de communication.
12. Un événement MQTT est détecté et un message est envoyé à la tâche de mise à jour.
13. La tâche de mise à jour des LEDs transforme le byte venant du PLC en byte à envoyer au GPIOs et elle les envoie via I2C.
14. La LED du bouton s'allume.

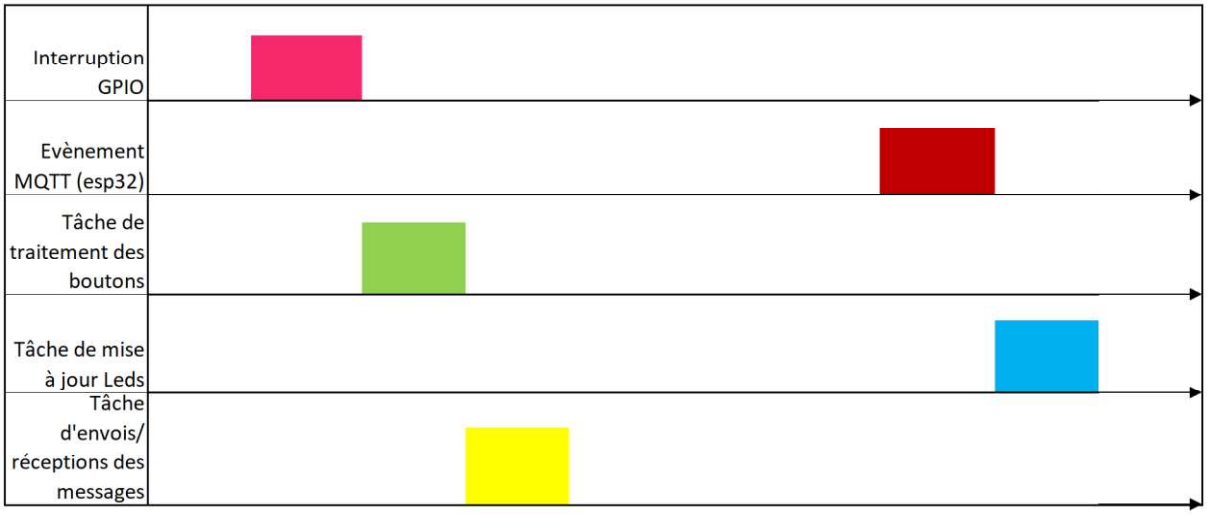


Figure 15 diagramme d'ordonnancement des tâches, exemple de traitement bouton/LEDs

4.2.1. Potentiomètre

La tâche de gestion du potentiomètre se déclenche de façon périodique. Elle échantillonne une valeur analogique de tension exprimant la position du potentiomètre via l'ADC de l'esp32. Puis, si la valeur a changé par rapport à la dernière valeur envoyée au broker MQTT (Tâche d'envois/réceptions de messages), la tâche de gestion du potentiomètre demande de publier cette valeur (en envoyant un message via une file de messages FreeRTOS à cette tâche).

Le diagramme ci-dessous illustre ce fonctionnement et on voit bien que la tâche d'envois/réceptions de messages déclenche un traitement (publish de la valeur) de façon conditionnelle (si la valeur a changé).

Figure 16 Diagramme d'ordonnancement des tâches de traitement des valeurs du potentiomètre

Le traitement effectué périodiquement par la tâche de traitement du signal venant du potentiomètre est le suivant :

1. Plusieurs échantillons de la valeur sont pris par une lecture de l'ADC.
2. On fait la moyenne de ces échantillons.
3. On rentre cette valeur dans le passe-bas informatique du premier ordre qui est un filtre de moyenne glissante à pondération pseudo-exponentielle.
4. La résolution sera diminuée pour avoir en sortie une valeur entre 0 et 100.
5. Une limite inférieure à min=0 et supérieure à max=100 sera imposée pour être sûr que la valeur envoyée ne dépasse pas les limites imposées.
6. Puis on fera un trigger de Schmitt autour de la valeur basse résolution (en se référant à la valeur haute résolution) pour que la valeur n'oscille pas sans arrêt entre deux valeurs lorsque la valeur haute résolution est à la limite entre deux paliers de la basse résolution.

Figure 17 Chaine de traitement du signal du potentiomètre

Il serait plus logique d'associer le trigger de Schmitt à la diminution de la résolution, mais le faire après l'imposition de la limite permet d'économiser ce traitement, car si après l'imposition de la limite la valeur reste la même que la valeur précédemment envoyée, on ne fera pas le traitement du trigger de Schmitt. Cette situation arrive chaque fois que le potentiomètre n'a pas bougé, donc pratiquement tout le temps. Vu que le traitement s'effectue à une fréquence relativement élevée cela représente une grande économie.

4.2.1. Tâche de Communication MQTT

La tâche de communication MQTT s'occupe de créer, maintenir la communication avec le broker MQTT. Elle se charge aussi du publish des messages ainsi que du maintien du subscribe aux topics MQTT nécessaires. Un subscribe au topic des LEDs est nécessaire. La tâche doit également assurer le publish des valeurs des boutons et du potentiomètre.

Le fonctionnement de cette tâche est décrit dans la machine à états suivante qui est lancée au reset ou à la mise sous tension de la carte.

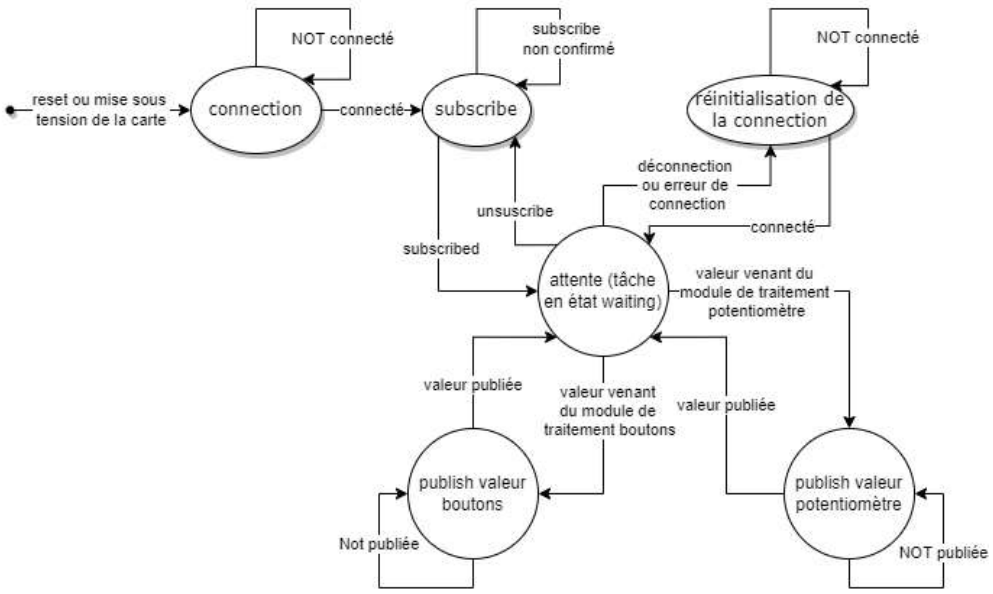


Figure 18 Machine à états de la tâche de communication au sein de la carte embarquée

Ci-dessous vous trouverez une table permettant de complémenter la machine à états pour la compréhension par le graphique du fonctionnement de la tâche.

Mot clé	explication	références
Connexion	Essai de se connecter au broker, cela comprend tous les aspects de connexion MQTT et soket TCP/IP (attribution de l'IP par le DHCP compris).	Voir la section broker : Figure 19 Attribution d'une IP par le DHCP & Figure 20 Connexion au broker MQTT échange de paquet.
Réinitialisation de la connexion	Même explication que pour la connexion.	Mêmes références que pour la connexion.
Attente	État lorsque la tâche se met en waiting à la suite de l'appel d'une fonction FreeRTOS attendant sur un événement.	
Publish valeur boutons	Publish la valeur des boutons reçue via une file de messages.	Figure 23 Échange de messages entre la carte embarquée et le broker lors de la mise à jour de la valeur des boutons ou du potentiomètre
Publish valeur potentiomètre	Publish la valeur du potentiomètre reçue via une file de messages.	Figure 23 Échange de messages entre la carte embarquée et le broker lors de la mise à jour de la valeur des boutons ou du potentiomètre
Subscibe	Subscribe aux topics nécessaires. (Dans cette application précise au topic des LEDs). Envoi du message « subscribe » au broker MQTT.	Figure 21 échange de messages pour un subscribe sur la carte embarquée
Connecté	Événement MQTT (ESP32) suite à la réception d'un message « connect Ack » venant du broker MQTT.	Figure 20 Connexion au broker MQTT échange de paquet
Suscribe non confirmé	Correspond à NOT suscribed, c'est-à-dire à la non réception d'un message MQTT « suback » de la part du broker	Figure 21 échange de messages pour un subscribe sur la carte embarquée
subscribed	Correspond à la réception d'un message MQTT « suback » de la part du broker	Figure 21 échange de messages pour un subscribe sur la carte embarquée
unsubscribe	Correspond à la réception d'un message MQTT « unsubscribe » de la part du broker	
Valeur publiée **Ou** **Publiée**	Correspond à la réception d'un message MQTT « published » de la part du broker. Ce message confirme dans une configuration de QoS de niveau 1 la réception du message publié, ici par la carte embarquée.	Figure 23 Échange de messages entre la carte embarquée et le broker lors de la mise à jour de la valeur des boutons ou du potentiomètre

Valeur venant du module de traitement boutons	Réception d'un message indiquant un changement de valeur des boutons depuis la « tâche de traitement des boutons » qu'il faut publish à destination du broker.	Section : Boutons & LEDs
Valeur venant du module de traitement potentiomètre	Réception d'un message indiquant un changement de valeur des boutons depuis la « tâche de traitement du potentiomètre » qu'il faut publish à destination du broker	Section : Potentiomètre
Déconnexion ou erreur de connexion	Correspond à la réception d'un message MQTT « disconnection »(le broker indique qu'il va se déconnecter (exemple : à la suite d'une demande d'arrêt)) ou «error » de la part du broker	

Tableau 5 Légende de la machine à états de la communication MQTT au sein de la carte embarquée

Cette machine à états peut paraitre complexe à première vue, mais en vérité il y a une raison à cette complexité. La raison est simple, elle permet de résoudre automatiquement un grand nombre de problèmes de communication entre les cartes embarquées et le broker.

Résolution automatique de problèmes au sein du système :

Dans tous les cas suivants, la connexion pourra être rétablie et les subscribe aux différents topics (ici uniquement au topic des LEDs) seront réeffectués. Les publish pourront également à nouveau être réeffectués. Les problèmes que cette machine à états/cette tâche peut résoudre sont :

- Reset de la carte embarquée ou arrêt et rétablissement de l'alimentation de la carte
- Coupure de connexion entre la carte et le switch
- Coupure de la connexion entre le switch et le PLC (PC)
- Mauvaise configuration initiale du firewall ou dérèglement et rétablissement du firewall
- DHCP pas atteignable pour l'attribution d'une IP puis de nouveau atteignable
- L'arrêt du broker sur le PLC (PC) et son redémarrage
- Mauvaise manipulation qui résulte avec un unsubscribe du Topic des LEDs

4.3. Automate sur l'ordinateur personnel (PLC)

Il y a 4 parties différentes sur le PLC, dans ce projet un PC a été utilisé en tant que PLC :

1. Le DHCP chargé d'attribuer les IP aux cartes embarquées
2. Le Broker MQTT, élément central du réseau
3. Le client MQTT de synchronisation Python
4. L'automate programmable en soi (PLC)

Tous ces softwares seront ébergés sur un PLC final avec un OS Windows, ce n'est donc pas un problème de le remplacer par un PC avec un OS Windows. Mais le PC n'étant pas le support final, certaines parties du développement (principalement la configuration du firewall) n'ont pas été poussées jusqu'à leurs formes finales, car elles risquent d'être relativement différentes sur le PLC.

4.3.1. Contrôle : DHCP

Le protocole MQTT est une couche implémentée au-dessus d'une pile TCP/IP. La création d'un soket TCP/IP est donc nécessaire au fonctionnement du protocole MQTT. Il faut donc définir tous les aspects habituels du réseau TCP/IP. Il a été choisi que l'IP serait attribuée aux cartes embarquées par un serveur DHCP.

Dans ce projet OpenDHCP a été choisi pour créer le serveur DHCP. Il a également été choisi de fixer l'IP du port Ethernet sur lequel sera connecté le switch. Le fichier de configuration sera fourni avec les différents programmes.

Dans ce fichier, il faut définir :

- L'IP du port Ethernet qui a été fixée
  - Section [LISTEN_ON] : écrire l'IP sans rien d'autre (ex : 192.168.5.10)
  - Sous « Router is default gateway » : définir la même IP (ex : Router=192.168.5.10)
- La plage d'IP disponible pour le DHCP
  - Section [RANGE_SET] : donner une plage d'IP (ex : DHCPRange= DHCPRange=192.168.5.100-192.168.5.125)

Scénario Idéal

La première action sur le réseau de la carte embarquée est donc d'envoyer un message en broadcast pour trouver le DHCP (message Discover) avec une adresse IP générique. Suite à ça, le DHCP envoie en broadcast un message contenant une IP disponible offerte à la carte (message offer). La carte envoie une requête (request) contenant l'IP offerte si elle lui convient. Le DHCP vérifie que cette IP est bien disponible, puis confirme l'attribution à la carte par un acknolege. Ce scénario est le scénario idéal.

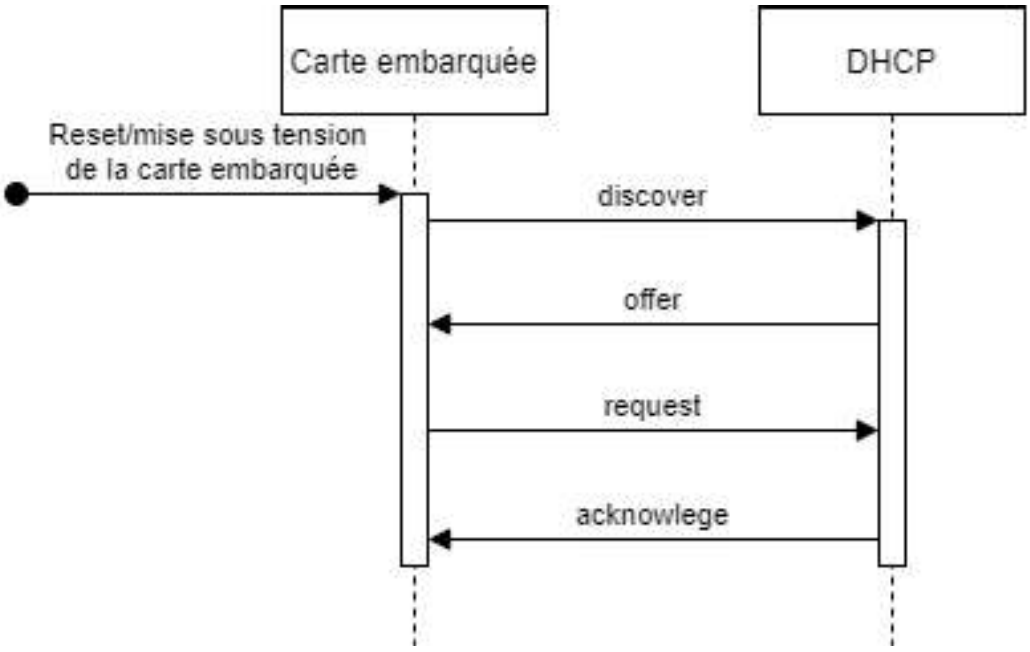


Figure 19 Attribution d'une IP par le DHCP

Problèmes possibles et leurs résolutions

Il se peut que des problèmes surviennent. Pour analyser ces problèmes, l'utilisation d'un analyseur de paquet comme wireshark sera nécessaire.

Si le message « discover » est émis et capturé par wireshark sur la connexion Ethernet **mais que le message « offer » n'est jamais émis** par le DHCP, cela peut venir du fait que le DHCP n'est pas en cours d'exécution. Cela peut aussi venir du firewall qui ne laisse pas passer les messages jusqu'au DHCP. Cela peut aussi venir d'une mauvaise configuration de l'IP sur laquelle le DHCP écoute les communications.

Ces problèmes sont déjà survenus, il est donc bien d'en avoir connaissance.

4.3.1. Données & Contrôle : Broker MQTT

Le Broker MQTT est l'élément central du réseau. Il met les valeurs des topics à jour lorsque qu'une des carte embarquée publish la valeur des boutons ou du potentiomètre ou bien que le client Python publish la valeur des LEDs. Il envoie aussi cette valeur à tous les suscribers de ce topic quand elle est mise à jour. Il envoie la valeur des LEDs aux cartes embarquées (lorsque le client Python la met à jour) et la valeur des boutons ou du potentiomètre au client Python (lorsque les cartes embarquées les mettent à jour).

Le broker choisi est mosquitto car il est très répandu. Cela a amené bon nombre de projets open source à l'utiliser et cela a aussi contribué à la création de tutoriels et de discussions sur des forums. Toute ces sources d'informations seront très utiles.

Sa configuration est la suivante :

- Connexion sur le port 1883 habituel (listener 1883)
- Pas de connexion sans nom d'utilisateur et mot de passe (allow_anonymous false)
- Mot de passe et nom d'utilisateur définis dans un autre fichier grâce aux lignes de commande mosquitto (password_file pwfile.example)
- Pas de certificat TLS/SSL nécessaire (require_certificate false)

Toutes les communications sont en QoS de niveau 1. QoS de niveau 0 n'offre aucune garantie que l'information a été transmise, ce type de communication a été jugée bien trop hasardeuse. La QoS de niveau 2, quant à elle, est surtout faite pour les connexions sans fil avec des module devant économiser leur énergie. Pour une application qui communique via câble Ethernet ,cela n'est pas nécessaire et ça ralentit les communications. La QoS de niveau 1 quant à elle garantit que les données seront transmises au minimum une fois, ce qui convient parfaitement à ce projet.

Connexion

Lorsqu'une carte embarquée s'est vue attribuer une adresse IP, elle peut initier un soket TCP. Puis, via ce soket TCP, initier une connexion MQTT.

C'est lors du message de connexion (MQTT connect) que l'identification via le nom d'utilisateur et le mot de passe se fera (le message contiendra ces deux informations).

Le broker répondra par un message contenant notamment le « returncode » qui donnera l'information si la session de communication a été établie avec succès et si ce n'est pas le cas, pourquoi (MQTT connect ACK aussi appelé CONNACK)

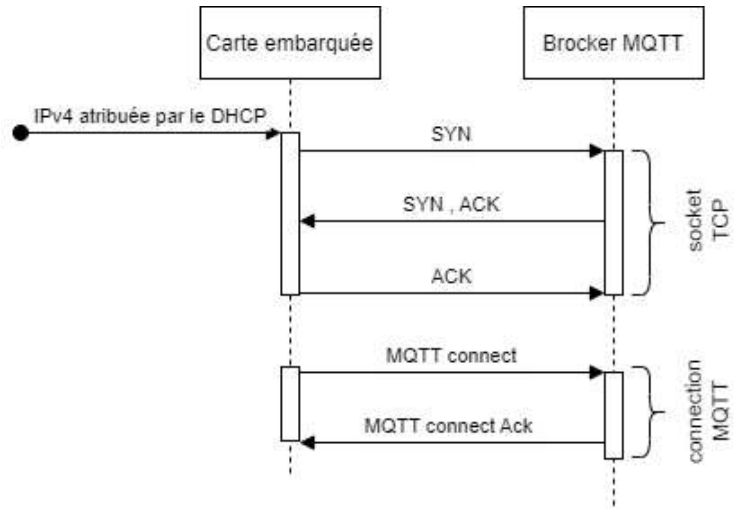


Figure 20 Connexion au broker MQTT échange de paquet

Problèmes de connexion possibles et leurs résolutions

Si le returncode du CONNACK (MQTT connect ACK) n'est pas égale à 0, alors il faut se référer à la documentation MQTT pour connaître le problème.

Une erreur courante (et qui est déjà arrivée) est de mal renseigner les mots de passe et noms d'utilisateurs sur la carte ou d'oublier de les ajouter au fichier des mots de passe via les lignes de commande de mosquitto. Cela sera sanctionné par un **returncode égal à 4**.

Le **returncode 3,** quant à lui, signifie un problème de connexion. D'expérience, cela vient souvent d'une interférence de Windows ou d'un autre programme. Toutefois cela ne se produira pas sur le PLC final, car cela venait, lorsque c'est arrivé durant le développement, d'autres projets et des configurations Windows liées à ceux-ci.

Subscribe au topic des LEDs par la carte embarquée

La carte embarquée doit subscribe au topic des LEDs pour être notifiée par le broker via un publish d'un changement des LEDs demandé par le PLC via Le client MQTT Python.

La carte réitérera le subscribe en boucle tant qu'il n'aura pas été acquitté à l'initialisation. Elle s'assurera du maintien de celui-ci parce que sans subscribe elle ne pourra pas mettre à jour les LEDs . De fait, elle ne fonctionnera pas correctement si ce n'est pas le cas (voir Figure 18 Machine à états de la tâche de communication au sein de la carte embarquée)

Si le suback n'est pas retourné, cela vient sûrement d'un problème de connexion (par exemple un câble qui dysfonctionne ponctuellement). Les câbles mal connectés mécaniquement (le « clapet » du câble était cassé) ont parfois créés des problèmes pendant le projet, mais quand ils ont été changés par des câbles neufs cela ne s'est plus produit.

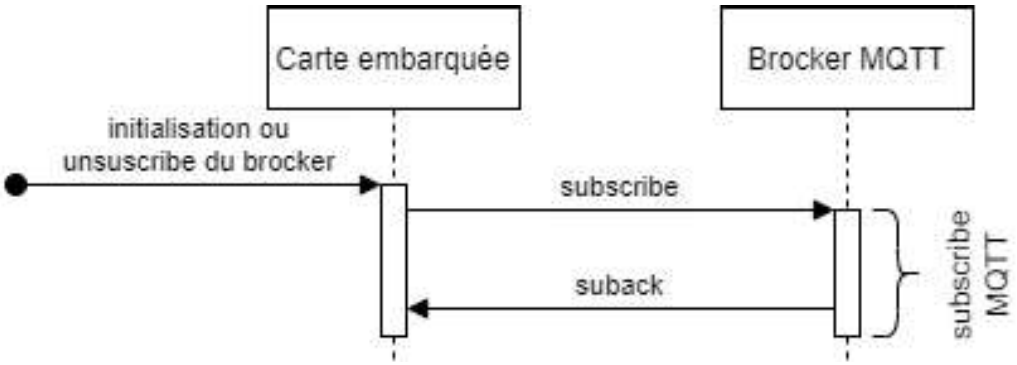


Figure 21 échange de messages pour un subscribe sur la carte embarquée

Le broker enverra un publish lorsque le PLC demandera un changement de l'état des LEDs.

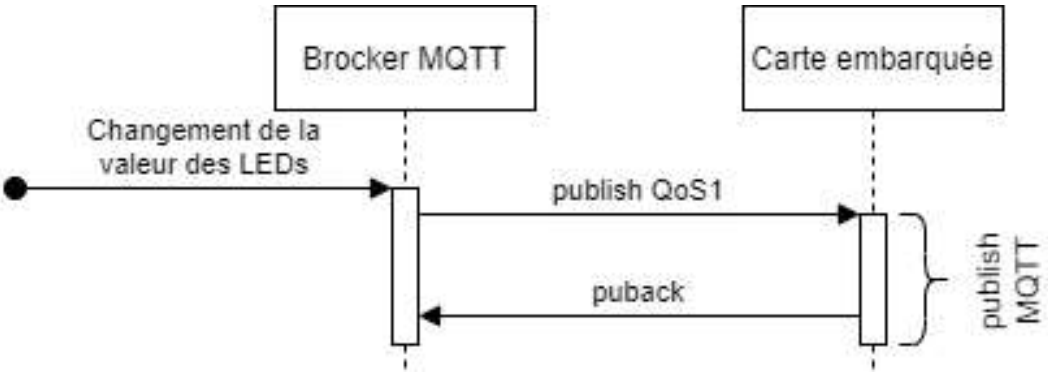


Figure 22 Echange de messages entre le broker MQTT et la carte embarquée lors de la mise à jour des LEDs

A chaque changement des valeurs des boutons ou du potentiomètre un publish sera envoyé et tant que le puback ne sera pas reçu du broker, le publish sera réitéré (voir Figure 18 Machine à états de la tâche de communication au sein de la carte embarquée)

Si le broker reçoit deux fois la même valeur, cela ne pose aucun problème puisque l'information envoyée est la valeur au moment de l'envoi et non pas une variation par rapport à une valeur précédente.

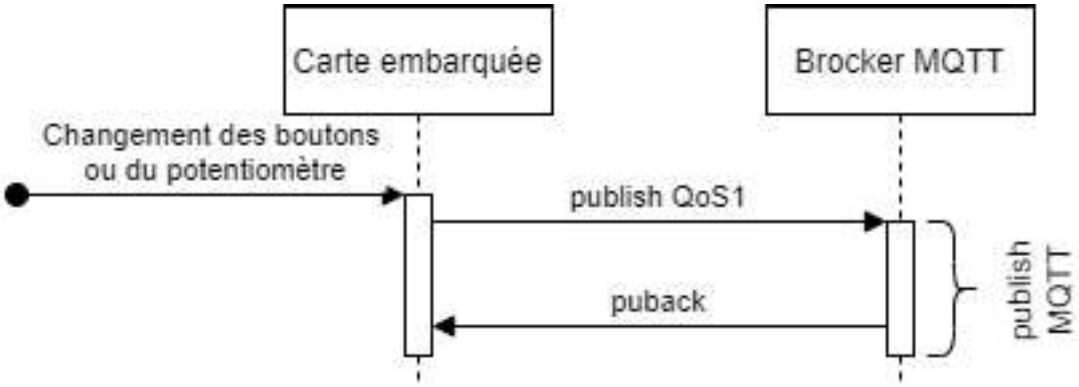


Figure 23 Échange de messages entre la carte embarquée et le broker lors de la mise à jour de la valeur des boutons ou du potentiomètre

4.3.1. Données & Contrôle : Client de synchronisation MQTT

Le client de synchronisation MQTT est codé en Python et utilise pleinement les facilités de ce langage. La librairie paho (librairie MQTT), par exemple, gère la reconnexion au broker et la seule action qui doit être développée par le développeur-même est de refaire les suscribe. La librairie ADS qui s'occupe de la synchronisation avec l'automate (PLC sur twincat tournant sur le PC), est moins efficiente, mais après connexion avec le PLC elle nous donne accès à une fonction pour changer une variable sur le PLC et une autre pour lire la valeur d'une variable. Ces deux fonctions ne nécessitent aucune intervention du PLC. L'état « vérifie la valeur sur le PLC » qui vérifie la valeur de la variable des LEDs ne le fait pas sur le signal du PLC, mais de façon périodique.

Les seules modifications dont ce client pourrait avoir besoin sont :

- Le changement des noms des variables avec lesquelles ADS travaille en fonction du nom de la valeur sur le PLC
- Le changement du nom des topics qu'il subscribe ou publish sur le broker

La machine à états suivante décrit le fonctionnement du client de synchronisation. On peut voir que deux thread s'exécutent en parallèle. Le Thread 1 se charge de vérifier périodiquement la valeur des LEDs sur le PLC et de notifier par un publish le broker si la valeur change. Le Thread 2, quant à lui, attend un publish de la valeur du potentiomètre ou des boutons de la part du brocker. Lorsqu'il recevra ce publish il changera la variable du potentiomètre ou des boutons sur le PLC.

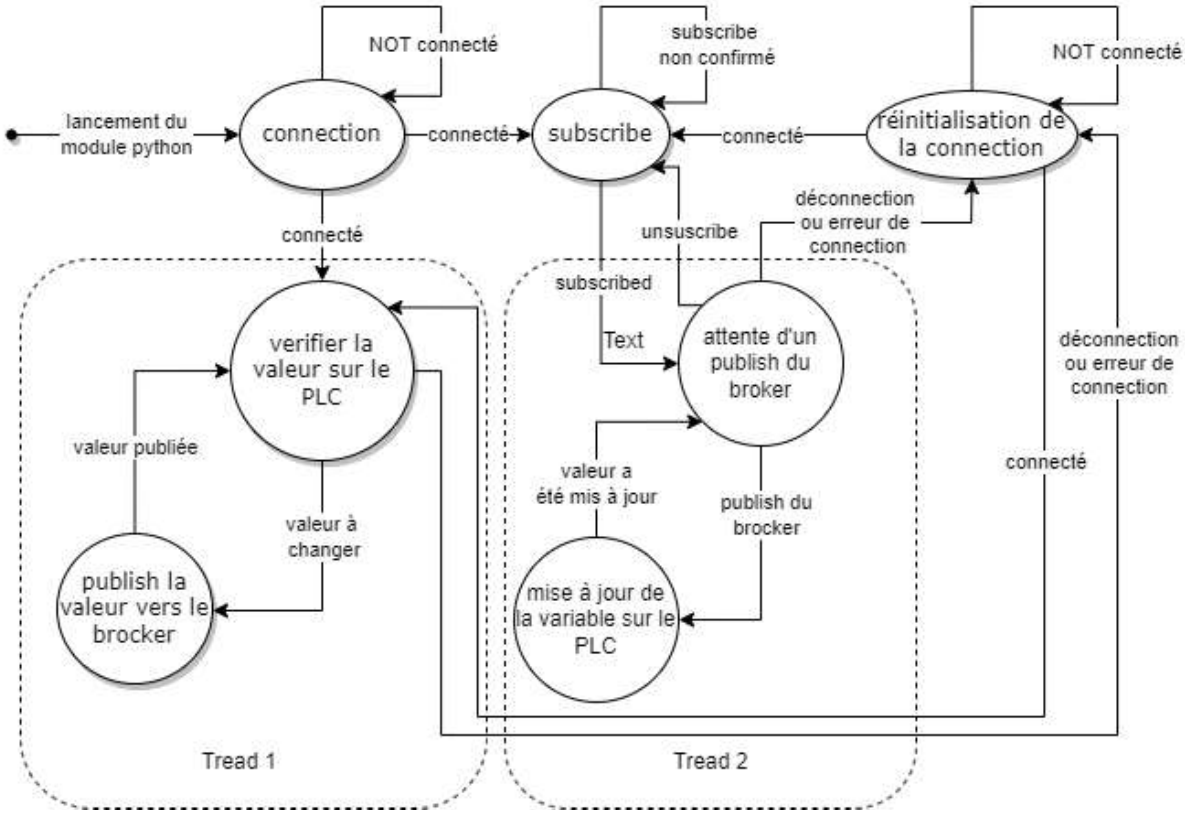


Figure 24 Machine à états du client Python

Ci-après : un tableau comportant une brève explication de chaque état et transition de la machine à états décrivant le fonctionnement interne du client Python.

Mot clé	explication	références
Connexion	Essai de se connecter au brocker, cela comprend tous les aspects de connexion MQTT et du soket TCP/IP (attribution de l'IP par le DHCP compris).	Voir la section broker : Figure 19 Attribution d'une IP par le DHCP & Figure 20 Connexion au broker MQTT échange de paquet Le processus est le même que pour une carte embarquée
Réinitialisation de la connexion	Même explication que pour la connexion.	Mêmes références que pour la connexion
Attente d'un publish du broker	Le thread attend de recevoir un message du broker (soit une valeur de bouton, soit une valeur du potentiomètre).	
Mise à jour des valeurs sur le PLC	Change la valeur de la variable potentiomètre ou de la variable bouton sur le PLC grâce à une fonction de la librairie ADS	
Publish la valeur sur le broker	Envoie un message MQTT « publish » au broker contenant la valeur des LEDs	Figure 25 échange de messages publish de la valeur des LEDs du client Python au broker

Vérifier la valeur sur le PLC	Prend la valeur de la variable LEDs sur le PLC de façon périodique.	
Subscribe	Subscribe au topic nécessaire (dans cette application précise au topic des boutons et à celui du potentiomètre). Envoi du message « suscribe » au broker MQTT.	Figure 26 échange de messages subscribe du client Python MQTT au broker
Connecté	Réception sur le client Python d'un « connect Ack » venant du broker MQTT.	Figure 20 Connexion au broker MQTT échange de paquet Le processus est le même que pour une carte embarquée
Subscribe non confirmé	Correspond à NOT subscribed, c'est-à-dire à la non-réception d'un message MQTT « suback » de la part du broker	Figure 26 échange de messages subscribe du client Python MQTT au broker
subscribed	Correspond à la réception d'un message MQTT « suback » de la part du broker	Figure 26 échange de messages subscribe du client Python MQTT au broker
unsubscribe	Correspond à la réception d'un message MQTT « unsubscribe » de la part du broker	
Valeur publiée **Ou** **Publiée**	Correspond à la réception d'un message MQTT « published » de la part du broker. Ce message confirme dans une configuration de QoS de niveau 1 la réception du message publié, ici par la carte embarquée.	Figure 25 échange de messages publish de la valeur des LEDs du client Python au broker
Publish du broker	Réception d'un message indiquant un changement de valeur des boutons ou du potentiomètre de la part du broker	Figure 27 publish du broker de la valeur du potentiomètre ou des boutons vers le client Python MQTT
Valeur a été mise à jour	La fonction ADS a été appelée et aucune erreur n'est survenue	
Déconnexion ou erreur de connexion	Correspond à la réception d'un message MQTT « disconnection »(le broker indique qu'il va se déconnecter (exemple : à la suite d'une demande d'arrêt)) ou «error » de la part du broker	

Tableau 6 légende de la machine à états du client MQTT Python

Tous comme la machine à états de la tâche de communication, cette machine à états peut paraitre complexe. Mais elle permet de résoudre automatiquement plusieurs problèmes de communication entre le client MQTT Python de synchronisation et le broker.

Résolution automatique de problèmes au sein du système :

Dans tous les cas suivants, la connexion pourra être rétablie et les subscribe aux différents topics (ici le topic des boutons et celui du potentiomètre) seront réeffectués. Les publish pourront également à nouveau être réeffectués. Les problèmes que la machine à états/que la tâche peut résoudre sont :

- Un arrêt et un redémarrage du broker
- Une coupure de la connexion entre le broker et le client de synchronisation, quelle que soit la raison
- Un arrêt et un redémarrage du client Python

Malheureusement, si le PLC s'arrête et redémarre le client s'arrêtera et devra être redémarré. La seule restriction dans l'ordre de démarrage des différents composants est que le PLC doit être démarré avant le client de synchronisation.

Communication entre le client de synchronisation et le broker :

Lorsqu'après un des contrôles périodique la valeur des LEDs a changé sur le PLC alors le client de synchronisation envoie un publish au broker avec la nouvelle valeur des LEDs.

Si le puback ne revient pas en retour, cela veut dire que le broker a été arrêté de façon impropre (le processus a été tué). Il faudra donc redémarrer le broker. Cela n'est jamais arrivé et a toujours été provoqué à des fin de test, cela ne devrait donc pas se produire.

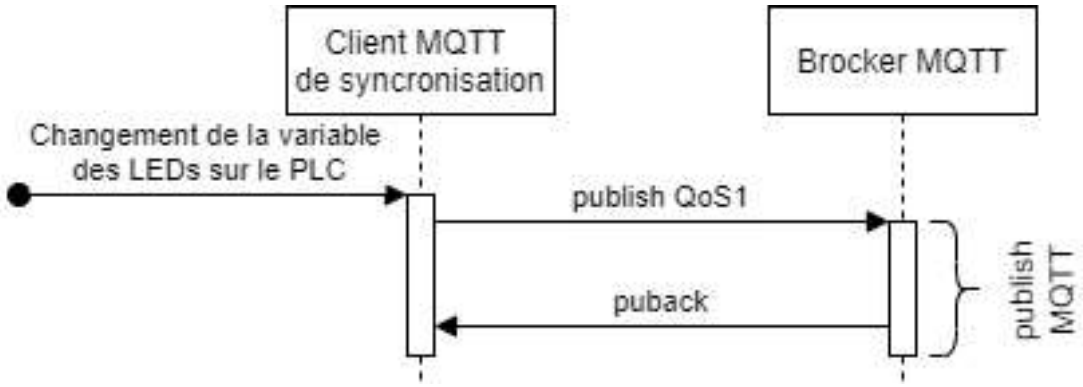


Figure 25 échange de messages publish de la valeur des LEDs du client Python au broker

Le client doit souscrire au topic des boutons et à celui du potentiomètre à l'initialisation ou lors d'une reconnexion.

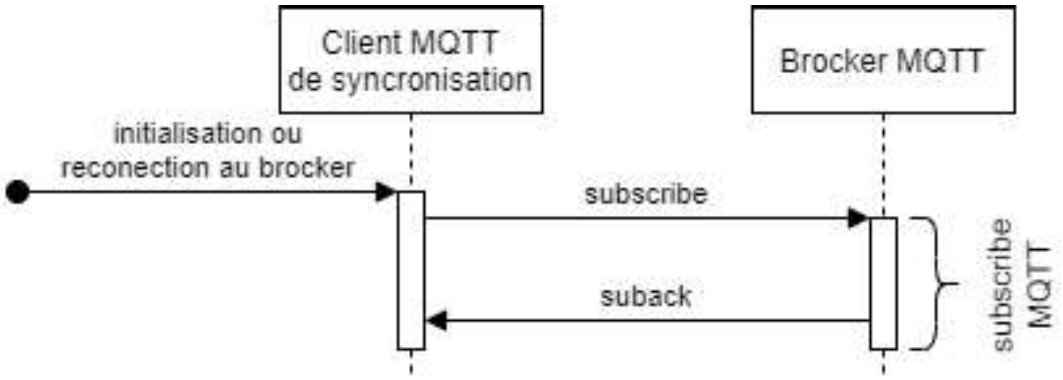


Figure 26 échange de messages subscribe du client Python MQTT au broker

En réponse à cela, un publish des valeurs des boutons et du potentiomètre est effectué de la part du broker.

Si la valeur de ces deux topics n'arrive pas au client de synchronisation, cela peut venir du topic qui n'est pas le même que celui publié par les cartes embarquées. Les topics sont visibles dans les messages de subscribe, comparer les subscribe et les publish avec un analyseur de paquet comme wireshark peut aider à trouver la solution au problème.

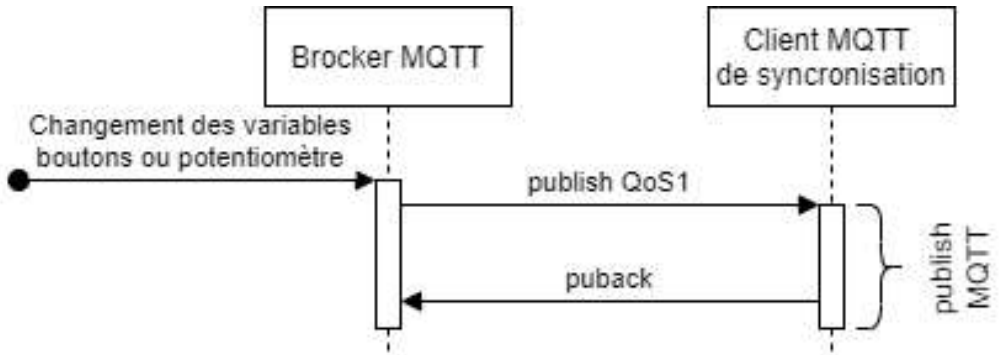


Figure 27 publish du broker de la valeur du potentiomètre ou des boutons vers le client Python MQTT

4.3.1. Données & Contrôle : Automate programmable (PLC)

La partie PLC est la partie la plus simple. Pour recevoir et envoyer des valeurs il suffit de :

- Déclarer des variables dans le PLC, exemples :
  - potentiometer
  - bouton
  - LED
- Les renseigner dans les fonctions ADS dans le client Python et les faire correspondre à un subscribe ou à un publish au sein du client Python

Dans ce projet, pour le test avec une seule carte, les 3 valeurs ci-dessus ont été créées sur le PLC. Les variables potentiometer et bouton étaient mises à jour depuis le client de synchronisation. Un programme a également été créé qui copie constamment la variable bouton dans la variable LED. Le client de synchronisation vérifiait périodiquement la variable LED et envoyait la valeur de la variable au broker lorsqu'elle changeait.

Pour le test avec un rajout de deux autres cartes envoyant des fausses données, quatre variables supplémentaires avaient été créées. Elles avaient été traitées comme un deuxième set de boutons et de LEDs.

4.4. Tests

Le software a surtout été testé de façon fonctionnelle et avec des simulations de pannes.

4.4.1. Test fonctionnel de mise en route du système

Ce projet contient un grand nombre de programmes tournant en parallèle de façon continue :

- Le DHCP
- Le broker MQTT
- Le client Python MQTT de synchronisation
- Le système temps réel du PLC
- Plusieurs cartes embarquées
- Un switch (aucune partie logicielle développée/configurée sur celui-ci)

Pour la mise en route du système, il a été testé tous les ordres de démarrage possibles des différents systèmes.

Le seul problème, et une possible amélioration dans le futur, est que le système temps réel du PLC doit être démarré avant le client Python MQTT de synchronisation.

À part cela, n'importe quel ordre successif ou simultané est supporté et ne créera aucun problème.

4.4.2. Test fonctionnel sur la chaine de communication du système en situation réelle avec 1 IHM et test du temps de transmission

Ce test consiste à implémenter le schéma « Figure 14 Architecture réseau du point de vue software » mais avec une seule IHM.

Sur cette installation, tous les boutons et la mise à jour de leurs LEDs ainsi qu'une grande quantité de combinaisons de boutons ont été testés. Le potentiomètre et l'envoi de sa valeur, uniquement quand elle change, a également été testé (voir la vidéo envoyée aux différents membres de l'équipe).

Il a été découvert que lorsque l'on branche l'USB pour flasher ou recevoir les messages de debug, cela peut créer parfois des variations sur l'alimentation de la carte et changer la valeur mesurée du potentiomètre de 1%. Ce n'est pas un problème grave, car dans l'installation finale seul le câble Ethernet sera branché et seule la source d'alimentation du PoE subsistera.

Des mesures du temps pour une information envoyée de la carte embarquée au PLC et revenir à la carte embarquée ont été effectuées. Le temps pour envoyer le message au PLC pour lui signifier un changement d'état des boutons puis pour envoyer la valeur de mise à jour des LEDs est en moyenne de 92 ms. Le temps reste en dessous de 100 ms tant que le PC a moins de 60% d'occupation du processeur. Cependant, il est arrivé que cela prenne jusqu'à 183 ms lorsque le PC avait un encombrement de plus de 90% dû à des mises à jour de logiciels autres que ceux utilisés au sein du projet. Cela remplit les objectifs de rapidité de communication au sein du réseau MQTT.

4.4.3. Test fonctionnel du système en situation réelle avec 1 IHM et deux fausses IHM

Cela consiste exactement en le même test que le test précédent « Test fonctionnel sur la chaine de communication du système en situation réelle avec 1 IHM » mais en rajoutant 2 cartes Olimex connectées au switch envoyant de façon périodique (4 fois par seconde) un message au PLC et recevant cette même donnée de la part du PLC en retour. Aucun changement significatif n'a été constaté sur le temps moyen d'un message pour passer au travers de la chaine de communication. Aucun autre changement n'a été constaté.

4.4.4. Test de simulations de pannes

Le test de simulations de pannes vise à tester les résolutions automatiques de problèmes évoqués dans les sections « 4.2.1 Tâche de Communication MQTT : Résolution automatique de problèmes au sein du système : » et « 4.3.1 Données & Contrôle : Client de synchronisation MQTT : Communication entre le client de synchronisation et le broker : ». Ce test a été effectué sur la même installation que la section « Test fonctionnel sur la chaine de communication du système en situation réelle avec 1 IHM »

Pour simuler ces pannes les processus ont été arrêtés puis redémarrés proprement. Ils ont également été arrêtés puis redémarrés de façon abrupte en tuant le processus depuis l'OS ou en arrêtant l'alimentation. Les connexions hardware Ethernet ont également été débranchées puis rebranchées.

Le fonctionnement de ces deux sections de résolutions automatiques des problèmes a été confirmé et aucun problème n'a été détecté.

5. Remarques, autocritiques et améliorations

Une brève review du projet et des différentes parties de ce projet avec une autocritique constructive.

5.1. Documentation :

Les tests du système ont pris du temps sur le temps qui était dédié à la documentation, il en résulte une documentation assez faible en dehors du rapport. Le temps dédié à la documentation pendant le développement a été trop faible également. Les tests ont été effectués mais pas documentés proprement. La documentation est par conséquent insuffisante pour un produit fini. Il manque notamment un manuel pour l'utilisateur/développeur. Le rapport contient néanmoins toutes les informations importantes. Les commentaires dans le code et sur le schéma électrique donnent toutes les informations nécessaires pour le comprendre avec l'aide du rapport.

5.2. Produit :

Tous les objectifs principaux ont été atteints. On est même allé plus loin que les objectifs en implémentant des résolutions automatiques de problèmes et en faisant en sorte qu'une séquence de démarrage précise des différents programmes ne soit pas nécessaire. Cependant, l'objectif secondaire consistant à pouvoir modifier la paramétrisation des différents logiciels via un simple envoi de fichier JSON n'a pas pu être rempli. La deuxième itération (itération finale) du PCB n'a pas encore été montée et testée, mais cela sera fait dans l'entreprise avant la défense du travail de Master.

5.3. Management

Le gros point noir du management est qu'à la fin du projet la mise à jour hebdomadaire du Git n'a pas été effectuée. Pour le reste, les problèmes et les délais sont justifiables et n'ont pas entrainés un problème majeur (comme par exemple un système non-fonctionnel à la fin du projet).

6. Conclusions

Ce projet était un projet de développement qui ne comportait pas de parties de recherche fondamentale ou applicative. Il n'est donc pas extrêmement innovant. Mais le but n'était pas là. Ce que le client voulait, c'était un système fonctionnel répondant à un problème concret.

J'ai été, moi, Thibault Sampiemon, en tant que développeur, très heureux de travailler sur un projet de produit ayant une application directe et concrète. Si tout se passe bien, je pourrai également voir le système en fonctionnement sur une machine qui sera réellement utilisée pour de la production industrielle dans une entreprise.

Je suis plutôt un développeur informatique et avoir pu développer l'électronique d'une carte embarquée m'a permis d'élargir mes compétences et de développer dans un projet concret des compétences auparavant plus théoriques que pratiques.

2108, Couvet, Suisse, juillet 2022

Thibault Sampiemon

7. Liste des figures

Annexes

Interface standardisée communication entre PLC’s et carte embarquée : Gestion de projet

Thibault Sampiemon

Hes-so, Help-tec Automation AG

Note de l’auteur

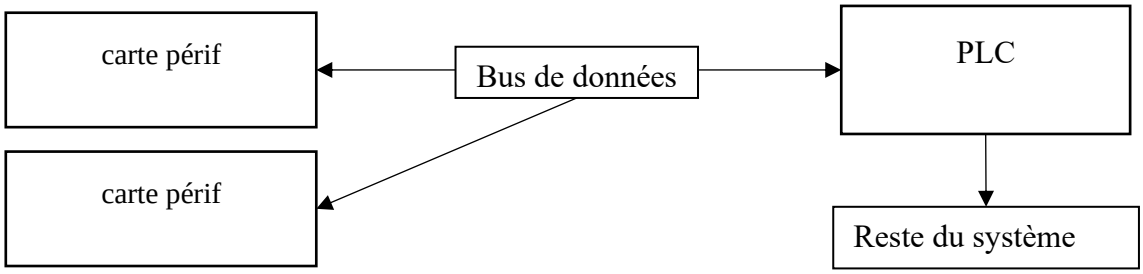
Ce dossier retrace les aspects de gestion du projet avant le début du projet qui commencera en mars 2022 en collaboration avec les contacts au sein des deux institutions : la Haute Ecole Spécialisée de Suisse Occidentale (hes-so)(professeur : Mr. Yves Mayer, gestionnaire de filière : Mr. David Grunenwald et conseiller aux études Mr. Philippe Walther) et l’entreprise Help-tec Automation AG (responsable du projet : Mr. Roger Schluep et chef d’entreprise Mr. Sacha Wittmann) dans le cadre d’un travail de master.

Table des matières

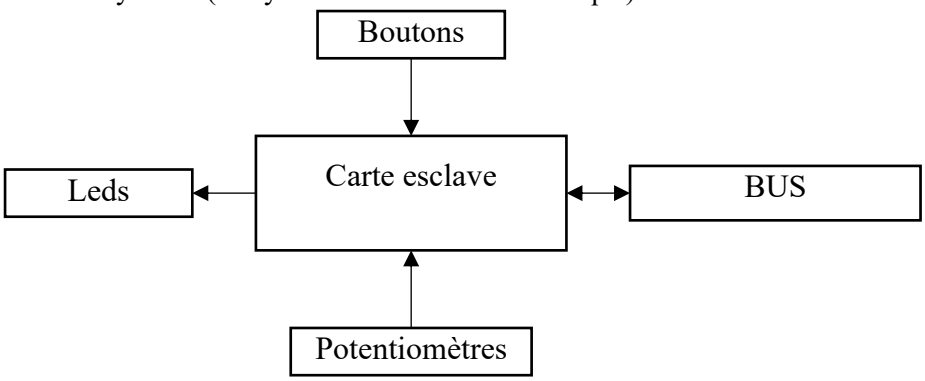
Contexte

Ce projet se déroulera dans le cadre d'un travail de master de 18 semaines à partir du 21.02.2022 jusqu'au 08.07.2022. Ce travail se fera dans un espace de travail attribué dans les locaux de la HE-Arc (site de St-Imier) et le responsable projet de l'entreprise se rendra sur place ou prendra contacte par camera interposée à intervalles réguliers pour prendre connaissance de l'avancée des travaux.

Le projet est un projet comportant 2 aspects : l'informatique et l'électronique, ainsi que trois parties : la partie liée à la carte de gestion des périphériques (carte périf), celle liée au PLC et celle liée au bus qui relie ces deux types de matériel.



Les cartes de gestion des périphériques peuvent être une ou plusieurs et communiquent toutes avec le PLC par un bus de données. La partie software sur le PLC fera le lien entre le bus de données et le reste du système (un système d'automation classique).



Les cartes de gestion des périphériques devront relever les données entrantes venant de certains périphériques, les transmettre sur le bus et changer les sorties en fonction des instructions venant du bus.

La partie software sur le PLC comporte deux parties distinctes : La gestion du BUS et la synchronisation des états des périphériques avec les variables locales accessibles depuis des systèmes d'automation classiques.

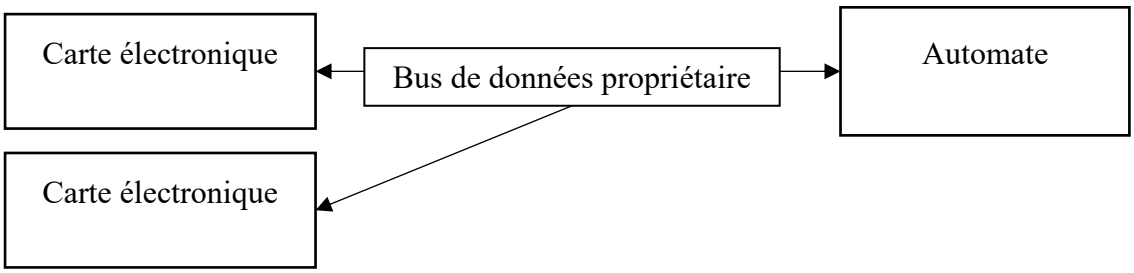
Interface standardisée communication entre PLC's et carte embarquée : Gestion de projet

CONOPS : intégration du nouveau système de remplacement

Ce projet a pour but de réduire les coûts du produit précédemment utilisé, de supprimer les fonctions non-utiles, d'ajouter des fonctions supplémentaires et d'améliorer la reconfigurabilité du produit. La topologie précédemment utilisée sera modifiée pour éviter des protocoles de communication propriétaires, très couteux pour chaque périphérique ajouté dans le système précédent.

CONOPS : Système existant

Le système existant est une carte électronique permettant de récupérer la pression sur un groupe de boutons et de renvoyer la pression de ce bouton sur un bus propriétaire à un automate programmable (Siemens, Beckhoff, …) et d'allumer des Leds lorsque l'automate programmable le demande via ce même bus propriétaire.



Fonctions opérationnelles de la carte électronique

- Envoyer l'état des boutons sur tous types de bus propriétaires
- Changer l'état des Leds en fonction des messages sur tous types de bus propriétaires

Fonctions opérationnelles du bus propriétaire

- Faire transiter des messages

Fonctions opérationnelles de l'automate

- Exécuter des actions en fonction de l'état des boutons
- Actionner des sorties en fonction de l'état du système

Coûts

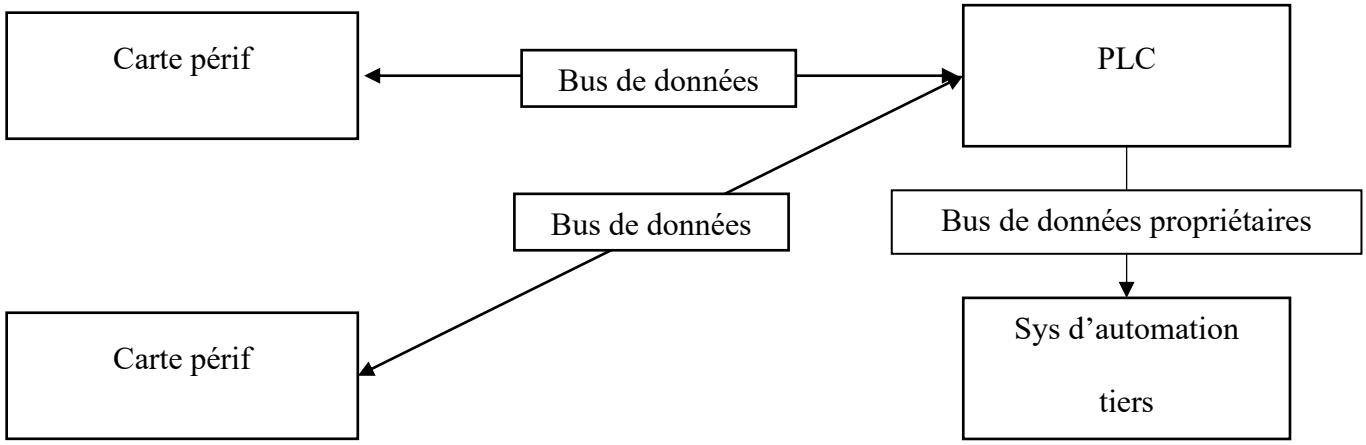
Pour chaque projet de chaine automatisée, l'entreprise pose un nombre donné de ces cartes électroniques égal au nombre d'interfaces bouton de cette chaine automatisée. La carte coûte

actuellement 400 CHF. Le coût de ce système est donc proportionnel au nombre de panel ajouté pour chaque projet.

CONOPS du Système de remplacement demandé en opération

Le système visé remplacera les cartes existantes vendue par une entreprise externe par des cartes spécifiquement crée pour cet usage au sein de ce projet et développées en interne du projet. Le PLC sera un PLC Beckoff avec un os windows sur lequel tournera le software de gestion du bus et de synchronisation des états et des variables internes.

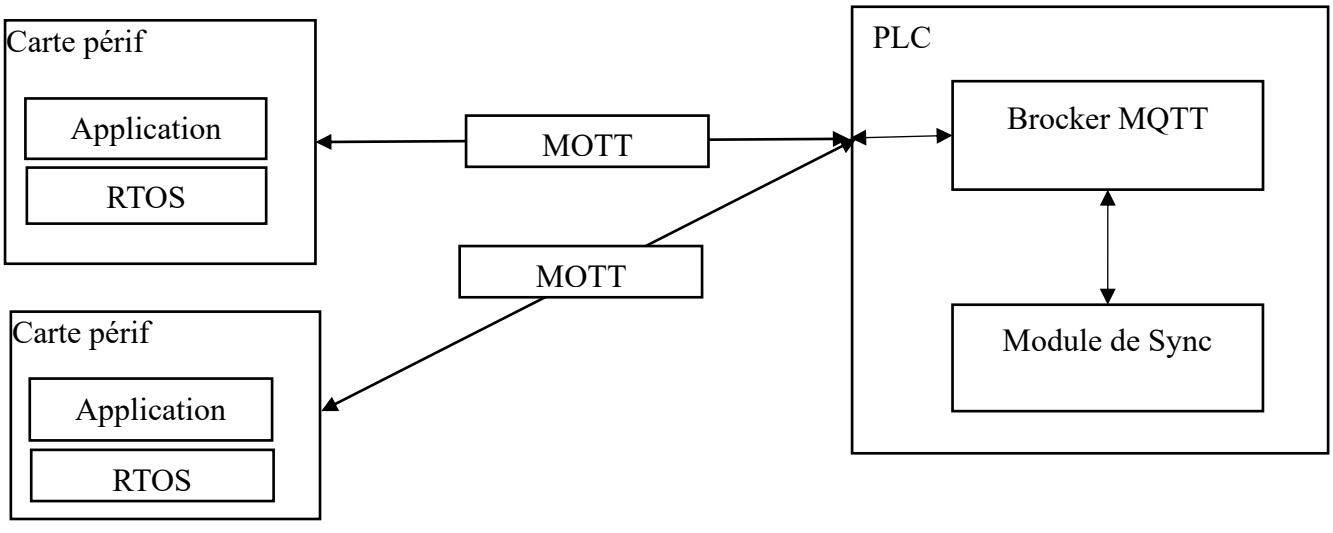
Topologie d'intégration du système :



Plusieurs choix ont déjà été fait :

- Les cartes périphériques comporteront des boutons, des LEDs et un potentiomètre monté directement sur le PCB et le système d'exploitation de la carte sera FreeRTOS
- Le Bus de donné sera un bus de données MQTT
- Le PLC sera un PLC Beckoff avec un os windows sur lequel tournera le software (Brocker+ Tâche de synchronisation variables)

Topologie du système :



Fonctions opérationnelles de la carte de gestion des périphériques

- Pouvoir connecter le maximum d'entrées-sorties possibles (boutons, leds, potentiomètres, panneaux d'affichage, …)
- Être ergonomique
- Pouvoir envoyer l'état des entrées sur le bus
- Pouvoir recevoir l'état des sorties depuis le bus et les mettre à jour
- Être alimentées directement par le bus

Fonctions constitutives de gestion des périphériques

- Un module hardware et un module software correspondant pour chaque type d'entrée-sorties visé
- Un module électronique de communication avec le bus et sa pile de communication software correspondante
- Microprocesseur
- Alimentation
- Un noyau software de gestion globale des différents modules
- Un système d'exploitation (temps réel)

Fonctions opérationnelles de la partie software sur le PLC

- Pouvoir modifier l'état des sortie des cartes de gestion de périphériques via MQTT et pouvoir réagir aux changement d'état des entrées des cartes de gestion périphériques
- Synchroniser les états des périphériques venant du bus MQTT avec des variables internes.
- Gérer les communications (Brocker MQTT)

Fonctions constitutives de la carte maître

- Un Brocker MQTT
- Une Tâche de synchronisation des états

Fonctions opérationnelles du bus

- Pouvoir transporter deux types de messages (de configuration et de communications de changement d'état des périphériques)
- Pouvoir alimenter les cartes de gestion périphériques

Fonctions constitutives du bus de données

- Un medium hardware adapté

Coûts

Pour ce nouveau système, l'équation des coûts doit être modifiée car, en plus des cartes gérant les entrée-sorties du système existant. Mais le coût idéal visé des n cartes de Gestion des périphériques est de 20 CHF.

CONOPS du projet

Le projet vise donc à s'approcher le plus possible du système demandé (décrit dans le conops précédant). Ce travail devra s'effectuer dans les 18 semaines prédéfinies, dans un bureau mis à disposition dans les locaux de la hes-so. Le coût de tous les objets et prototypes nécessaires à la réalisation de ce projet sera à la charge du mandant : l'entreprise Help-tec Automation AG.

Objectifs

- Diminuer les coûts de production (objectifs décrits dans les précédents conops)
- Augmenter les fonctionnalités
- Être capable d'assumer un fonctionnement temps réel
- Faciliter les futurs développements de chaines automatisées comportant des panels de contrôles
- Diminuer les dépendances aux fournisseurs en ramenant une tâche précédemment externalisée en interne.

Contraintes

- 18 semaines de travail avec un ingénieur
- Budget prototypes couvrant un maximum de 10 cartes électroniques prototypes esclaves (avec le fournisseur du mandant).

- Budget pour du matériel de test supplémentaire limité au maximum

Les Acteurs et leurs attentes

- Mandant Help-tec Automation AG
    - Réduire les coûts de production
    - Coût de développement le plus bas possible
    - Un prototype de système fonctionnel
    - Un maximum de fonctionnalités
    - Facile à maintenir et à adapter (maximiser la modularité et minimiser les interfaces de communication entre modules)
- Les clients de Help-tec Automation AG
    - Un système sans bugs
    - Ne pas remarquer la différence entre l'ancien et le nouveau système
- La Hes-so (plus précisément le professeur)
    - Un travail facilement évaluable
    - Des objectifs clairs
    - Un respect absolu de la date de fin du projet
    - Un travail à la hauteur des heures qui correspondent aux 30 crédits ECTS

Métriques de réussite et d'avancement produit

- Métriques de coût
    - % du coût de départ
    - % du coût cible
- Métriques de qualité
    - Modularité (idéalement un module par fonctionnalité)
    - Maximiser la fréquence maximale de communication sur le bus
    - Minimiser le temps moyen et maximum entre le changement d'une entrée sur la carte esclave et l'arrivée de l'information sur l'automate

- Minimiser le temps moyen et maximum entre l'envoi d'un ordre depuis l'automate et le changement d'une sortie sur la carte esclave

Métriques de santé du projet

- Nombre d'heures de retard/d'avance sur la fin des tâches
- Dépassements des jalons (gates) fixés
- Nombre de jours de retard sur la livraison des fournisseurs

Work packages du projet

Le projet sera fait entièrement par l'étudiant, sauf pour les tâches où il est expressément indiqué qu'un autre acteur la prend en charge. Ce qui veut dire que le gros du volume horaire incombera à l'étudiant. Chaque tâche comporte :

- Une mention de la personne chargée d'effectuer la tâche sous la rubrique « Responsable ».
- Un nombre d'heures estimé pour effectuer la tâche sous la rubrique « Nb d'heures », sauf pour les fournisseurs externes au projet. Pour eux, la durée est exprimée en semaines.
- Les autres tâches qui doivent êtres finies avant de commencer la tâche en question sous la rubrique « Tâches précédentes »
- Les coûts du matériel nécessaire sous la rubrique « Coûts ».

WP 1 : Hardware carte de gestion des périphériques : Création du PCB

1. Création du schéma électrique

1.1. Architecture hardware

1.1.1. Étude de la carte d'évaluation

Responsable :	Thibault Sampiemon	**Tâches précédentes :**	
Nb heures :	16h	**Coûts :**	

1.1.2. Choix des composants clés remplissant les fonctionnalités constitutives

Responsable :	Thibault Sampiemon	**Tâches précédentes :**	
Nb heures :	24h	**Coûts :**	

1.1.3. Schéma block de l'architecture hardware

1.1.3.1. Schéma block de l'architecture hardware carte fille

Responsable :	Thibault Sampiemon	**Tâches précédentes :**	
Nb heures :	8h	**Coûts :**	

1.1.3.2. Schéma block de l'architecture hardware carte finale

Responsable :	Thibault Sampiemon	**Tâches précédentes :**	
Nb heures :	16h	**Coûts :**	

Conception de l'électronique autour de chaque élément clé sous forme de module

1.1.4. Commande des composant carte fille

Responsable :	Thibault Sampiemon	**Tâches précédentes :**	WP 1.1.1(carte fille)
Nb heures :	4h	**Coûts :**	

1.1.5. Création de la librairie KiCAD

1.1.5.1. Création de la librairie KiCAD de la carte fille

Responsable :	Thibault Sampiemon	**Tâches précédentes :**	WP 1.1.1(carte fille)
Nb heures :	16h	**Coûts :**	

1.1.5.2. Création de la librairie KiCAD carte finale sur la base de la carte fille

et de :https://github.com/OLIMEX/ESP32-POE-ISO

Responsable :	Thibault Sampiemon	**Tâches précédentes :**	WP 1.1.1
Nb heures :	40h	**Coûts :**	

1.1.6. Création des schémas électroniques

1.1.6.1. Création des schémas électroniques carte fille

Responsable :	Thibault Sampiemon	**Tâches précédentes :**	WP 1.1.2.2(carte fille)
Nb heures :	24h	**Coûts :**	

1.1.6.2. Création des schémas électroniques carte finale

Responsable :	Thibault Sampiemon	**Tâches précédentes :**	WP 1.1.2.2
Nb heures :	60h	**Coûts :**	

2. Vérification du schéma électronique

2.1. Review du professeur responsable (un fois carte fille une fois carte finale)

Responsable :	Yves Mayer	**Tâches précédentes :**	WP 1.1.2.3
Nb heures :	2h	**Coûts :**	

2.2. Review du mandant (un fois carte fille une fois carte finale)

Responsable :	Roger Schluep	**Tâches précédentes :**	WP 1.1.2.3
Nb heures :	2h	**Coûts :**	

3. Création du PCB

3.1. Disposition des éléments

3.1.1. Disposition des éléments carte fille et routage

Responsable :	Thibault Sampiemon	**Tâches précédentes :**	WP 1.2.2(carte fille)
Nb heures :	40h	**Coûts :**	

3.1.2. Disposition des éléments carte finale et routage

Responsable :	Thibault Sampiemon	**Tâches précédentes :**	WP 1.2.2
Nb heures :	40h	**Coûts :**	

4. Vérification PCB

4.1. Review du professeur responsable

Responsable :	Yves Mayer	**Tâches précédentes :**	WP 1.3.2
Nb heures :	1h	**Coûts :**	

4.2. Review du mandant

Responsable :	Roger Schluep	**Tâches précédentes :**	WP 1.3.2
Nb heures :	1h	**Coûts :**	

5. Production du PCB

5.1. Envoi des donnés au fabricant dans le bon format

Responsable :	Thibault Sampiemon	**Tâches précédentes :**	WP 1.4.1 et WP 1.4.2
Nb heures :	2h	**Coûts :**	

5.2. Production (par une entreprise tierce)

Responsable :	Fournisseur PCB	**Tâches précédentes :**	WP 1.5.1
Nb heures :	1 semaine (risque de retard)	**Coûts :**	Coût du PCB ~20 CHF

5.3. Envoi (par une entreprise tierce)

Responsable :	Fournisseur PCB	**Tâches précédentes :**	WP 1.5.2
Nb heures :	1 semaine (risque de retard)	**Coûts :**	Coût de l'envoi ~5 CHF

5.4. Montage du PCB

Responsable :	Fournisseur PCB	**Tâches précédentes :**	WP 1.5.3
Nb heures :	4h	**Coûts :**	

6. Test du PCB

Responsable :	Thibault Sampiemon	**Tâches précédentes :**	WP 1.1.3
Nb heures :	40h	**Coûts :**	

WP 2 : Développement du software de la carte de gestion des périphériques :

1. Développement des modules softwares d'entrées-sorties

1.1. Développement des interfaces

Responsable :	Thibault Sampiemon	**Tâches précédentes :**	WP 1.2.2
Nb heures :	24h	**Coûts :**	

1.2. Développement des modules

Responsable :	Thibault Sampiemon	**Tâches précédentes :**	WP 2.1.1
Nb heures :	40h	**Coûts :**	

1.3. Test et adaptation des modules et interfaces

Responsable :	Thibault Sampiemon	**Tâches précédentes :**	WP 2.1.2 et WP1.6
Nb heures :	16h	**Coûts :**	

2. Intégration de la pile MQTT

2.1. Intégration de la pile MQTT

Responsable :	Thibault Sampiemon	**Tâches précédentes :**	
Nb heures :	16h	**Coûts :**	

2.2. Développement des couches supérieures du protocole

Responsable :	Thibault Sampiemon	**Tâches précédentes :**	WP1.2.2 et WP2.2.1
Nb heures :	40h	**Coûts :**	

2.3. Création des tâches de synchronisation des états et du bus

Responsable :	Thibault Sampiemon	**Tâches précédentes :**	WP2.2.2
Nb heures :	34h	**Coûts :**	

2.4. Test du bus avec simulation du maître sur ordinateur

Responsable :	Thibault Sampiemon	**Tâches précédentes :**	WP 2.2.3 et WP1.6
Nb heures :	16h	**Coûts :**	

3. Intégration software carte de gestion des périfériques:

3.1. Intégration de l'OS

Responsable :	Thibault Sampiemon	**Tâches précédentes :**	WP1.6
Nb heures :	22h	**Coûts :**	

3.2. Utilisation des modules au sein de tâches

Responsable :	Thibault Sampiemon	**Tâches précédentes :**	WP2.2.3 et WP2.2.4 et WP2.1.3
Nb heures :	32h	**Coûts :**	

WP 3 : Développement du software du PLC :

1. Intégration du broker:

1.1. Intégration du broker

Responsable :	Thibault Sampiemon	**Tâches précédentes :**	WP2.2.4
Nb heures :	24h	**Coûts :**	

1.2. Développement des couches supérieures du protocole

Responsable :	Thibault Sampiemon	**Tâches précédentes :**	WP3.1.1 et WP2.2.2
Nb heures :	40h	**Coûts :**	

2. Intégration PLC :

2.1. Intégration du Broker

Responsable :	Thibault Sampiemon	**Tâches précédentes :**	WP2.6.4
Nb heures :	32h	**Coûts :**	

2.2. Création d'un système de synchronisation avec les variables

Responsable :	Thibault Sampiemon	**Tâches précédentes :**	WP3.2.1 et WP3.1.2
Nb heures :	40h	**Coûts :**	

WP 4 : Test du système complet :

1. Tests de communications sur le bus et adaptations

Responsable :	Thibault Sampiemon	**Tâches précédentes :**	WP3.2.2 et W2.3.2
Nb heures :	40h	**Coûts :**	

2. Intégration du système dans une chaine de production de démonstration

Responsable :	Thibault Sampiemon	**Tâches précédentes :**	WP4.1
Nb heures :	32h	**Coûts :**	

3. Développement d'un programme de test du système

Responsable :	Thibault Sampiemon	**Tâches précédentes :**	WP4.2
Nb heures :	32h	**Coûts :**	

WP 6 : Documentation et Rapport :

1. Documentation du bus

Responsable :	Thibault Sampiemon	**Tâches précédentes :**	
Nb heures :	8h	**Coûts :**	

2. Documentation de la carte de gestion des périphériques

Responsable :	Thibault Sampiemon	**Tâches précédentes :**	
Nb heures :	16h	**Coûts :**	

3. Documentation de la partie PLC

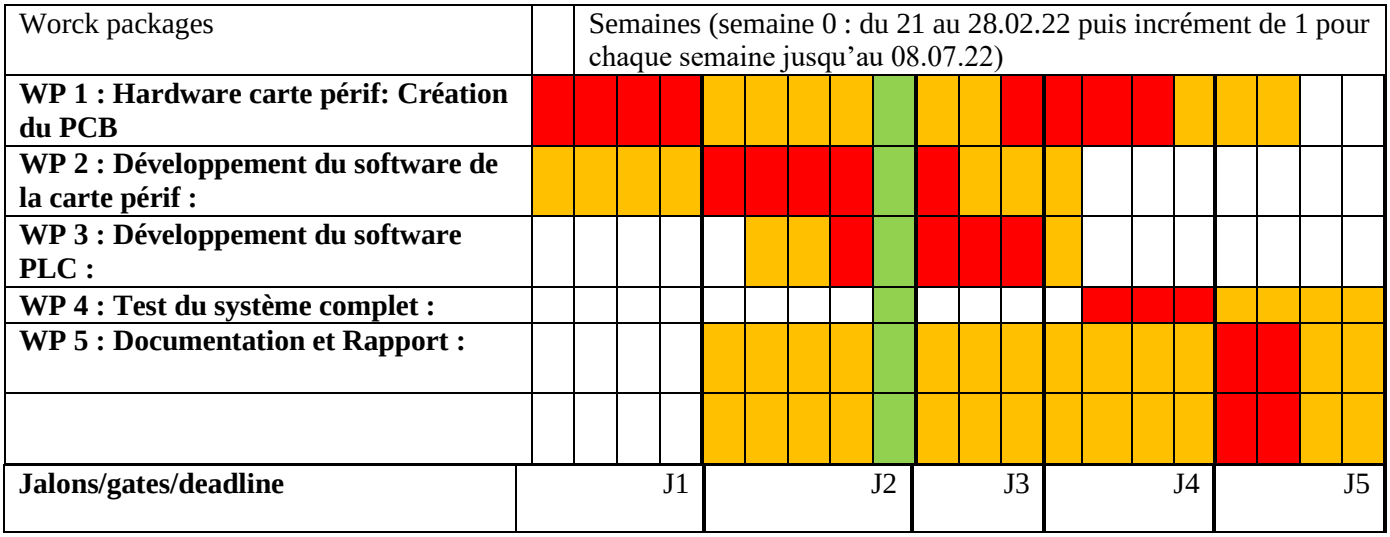
Responsable :	Thibault Sampiemon	**Tâches précédentes :**	
Nb heures :	16h	**Coûts :**	

4. Rapport final

Responsable :	Thibault Sampiemon	**Tâches précédentes :**	
Nb heures :	32h	**Coûts :**	

Gant général du projet

Ce Gantt général sert à situer temporellement l'exécution des différents work packages à l'échelle des semaines.



	Durée incompressible du package		Marge de manœuvre (certaines sous-tâches peuvent être commencées en amont et certaines peuvent être effectuées plus tard, sans impacter les packages dépendants)
	Semaine de pause		

Les jalon J1 à J5 représentent les dates butoirs où certaines parties du projet doivent impérativement être terminées. A chacune de ces dates butoirs, une réunion d'avancement du projet aura lieu. Les participants de ces réunions sont l'étudiant, le professeur référent et le mandant (responsable au sein de l'entreprise mandante).

Jalon	Date	Titre du jalon	Objectifs du jalon
J1	21.03.2022	Fin de la conception carte prototype	Le PCB de la carte fille devra être terminé et en production.
J2	25.04.2022	Fin de la conception de la carte périf	Le hardware et le software du prototype de la carte périf devront être parfaitement fonctionnels et testés.
J3	16.05.2022	Fin de la conception du système software	Le hardware et le software du prototype du PLC devront être parfaitement fonctionnels et testés.
J4	13.06.2022	Finalisation de l'installation de démonstration	On pourra démontrer que le système composé de plusieurs cartes peut fonctionner en collaboration avec un système d'automation test. Le design hardware de la carte finale sera terminé.
J5	08.07.2022	Fin du projet	Le projet devra être fonctionnel, documenté et une intégration dans une fausse chaîne de production devra démontrer son fonctionnement.

Suivi des tâches durant le projet

Le suivi des tâches durant le projet sera fait via un Kanban. Chaque tâche sera notée sur un post-it avec son titre et ses informations respectives (notamment les tâches qui doivent êtres finies avant cette tâche-là). Le post-it commencera sa course dans la colonne « Réservoir » et la finira dans la colonne « Fin ». Il parcourra le tableau de gauche à droite, une colonne à la fois, après avoir fini l'étape correspondante (en validant les conditions de fin d'étapes).

<table>
<tr><th>Réservoir</th><th>Implémentation</th><th>Review</th><th>Test fonctionnel</th><th>Intégration</th><th>Test d'intégration</th><th>Fin</th></tr>
<tr><td rowspan="3">Tâche WPx.x.x :
Responsable : Mr …
Tâches précédentes : WP…
Durée : …h
Coûts : …. CHF</td><td>En cours</td><td rowspan="2">Tâche WPx.x.z

Tâche WPx.x.y</td><td rowspan="2"></td><td rowspan="2"></td><td rowspan="2"></td><td rowspan="3">Tâche WPx.x.w</td></tr>
<tr><td>Finie</td></tr>
<tr><td></td><td></td><td>Tâche WPx.x.k</td><td></td><td></td></tr>
</table>

Réservoir

Contient toutes les tâches à effectuer dont les work packages ont déjà débuté

Condition de passage en étape d'implémentation : les tâches devant être effectuées précédemment sont finies (colonne fin) et leur implémentation a débuté.

Implémentation

Contient toutes les tâches en cours d'implémentation

Condition de passage en étape de Review : le module a été implémenté

Rewiew

Contient toutes les tâches dont le code doit être relu

Condition de passage en étape de test fonctionnel : le module a été relu

Test fonctionnel

Contient toutes les tâches en cours de test du software indépendant du matériel ou le matériel non encore intégré au design du PCB.

Condition de passage en étape d'intégration : le module a été testé

Intégration

Contient toutes les tâches en cours d'intégration matérielle ou d'intégration au PCB.

Condition de passage en étape de Test d'Intégration : le module a été intégré

Test d'intégration

Contient toutes les tâches en cours de test du software intégré matériel ou test du PCB.

Condition de passage en étape de Review : le module a été testé

Fin

Contient toutes les tâches terminées

Équipe

La haute école spécialisée de suisse occidentale (hes-so) :

- Professeur : Mr. Yves Meyer

  Rôle : évaluer l'étudiant et l'aider dans le développement en tant que personne ressource.

- Gestionnaire de filière : Mr. Philippe Walther

  Rôle : Approuver le Travail de Master et le professeur choisi

- Conseiller aux études : Mr. David Grunenwald

  Rôle : Approuver les décisions du gestionnaire de filière

- Étudiant effectuant le travail de master : Mr Thibault Sampiemon

  Rôle : Développer le produit

L'entreprise Help-tec Automation AG

- Responsable du projet : Mr. Roger Schluep

  Rôle : Suivre le développement du projet, approuver les choix techniques et créer le cahier des charges

- Chef d'entreprise : Mr. Sacha Wittmann

  Rôle : Déterminer les budgets et approuver le cahier des charges

ANEXES

Cahier des charges

Interface standardisée communication entre PLCs et carte embarquée

Implémentation avec un panel boutons et MQTT

Préambule

Il existe un grand nombre de bus de terrain actuellement en utilisation dans le milieu de l'industrie. Ces bus de terrain suivent des standards bien définis et reposent sur des technologies diverses et variées. Malheureusement, le choix du protocole utilisé est aujourd'hui presque complètement lié au fabricant du PLC que nous souhaitons utiliser. En effet, chaque marque a son bus de terrain de prédilection :

- Beckhoff : EtherCAT
- B&R : PowerLink
- Siemens : profiNET / profiBUS
- Allen&Bradley / Rockwell: Ethernet/IP (DeviceNet™ and ControlNet™)
- Schneider Electric : canOPEN

Ce qui rend l'interopérabilité de ces systèmes difficile.

Cependant, tous ces automates un en règle générale un connecteur réseau et sont utilisable pour l'utilisation du protocole TCP/IP. Il est donc judicieux d'utiliser cette interface commune.

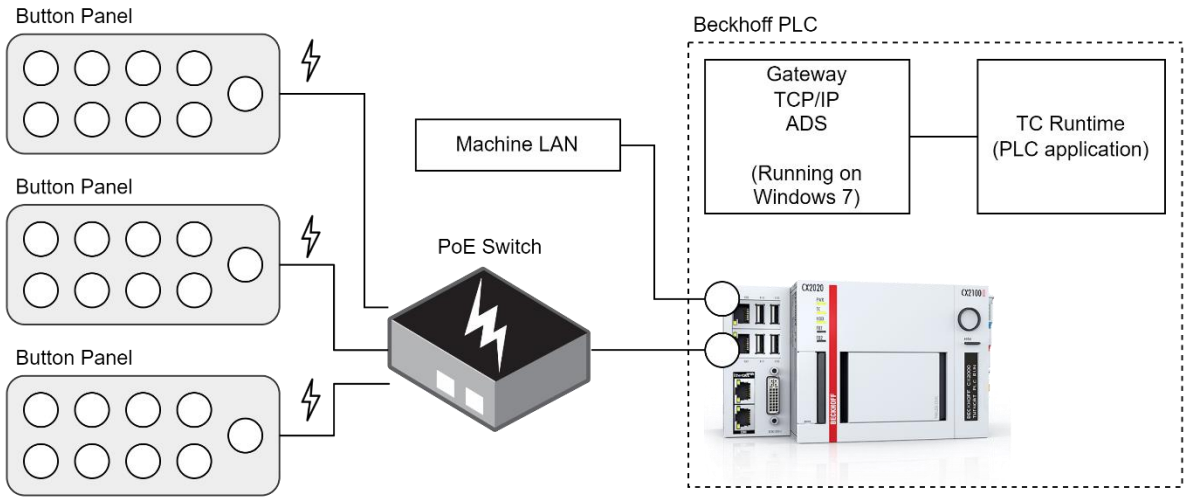
Besoin

L'objectif est de trouver un protocole de communication basé TCP/IP (ou UDP) répandu et qui sera facilement configurable depuis les différents IDE fournis par les différents fabricants de PLC. Ce protocole doit ensuite pouvoir être intégré dans une carte embarquée produite en petite quantité (< 1000 pces) en restant simple, flexible et peu onéreux.

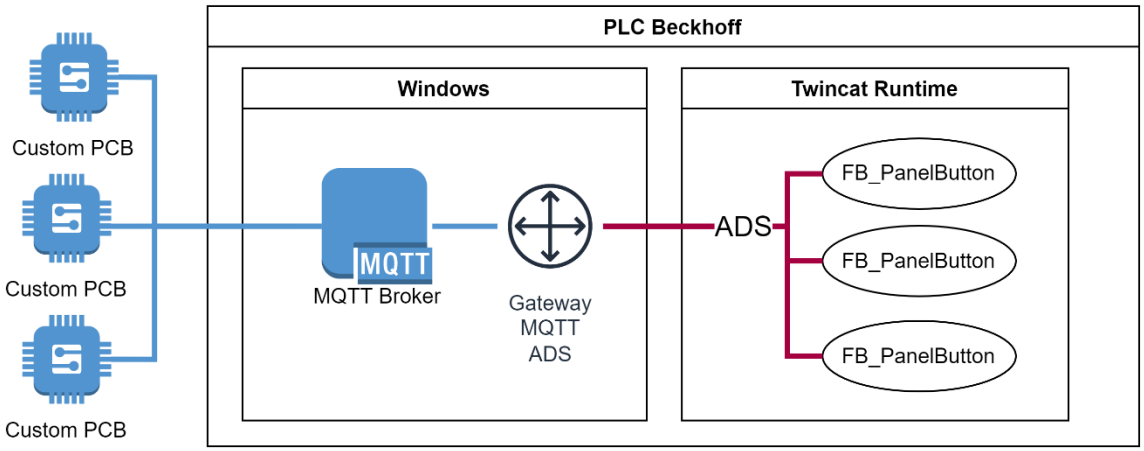
De plus, afin de limiter la configuration et le temps de câblages de ces différents éléments standard, Help-tec souhaite, si possible, que le protocole sélectionné utilise le même câble que l'alimentation des différents nœuds du réseau. Une des variantes de la technologie PoE (Power over Internet) est à privilégier. Un protocole de communication sans fil n'est pas souhaité actuellement, car les machines sont composées majoritairement d'aluminium relié à la terre. Certains nœuds pourraient alors être inaccessibles.

Schéma bloc

Montage :



Communication/Software :



Points à réaliser

- Création d'un PCB pour y mettre des boutons et un potentiomètre
    - Se baser sur la platine Help-tec pour les dimensions
    - MCU : ESP32
    - 1 interface Ethernet
    - Alimentation via PoE (standard à utiliser : TBD par l'élève)
    - Configuration IP statique (comment : Proposition par l'élève)
    - LED de statut (à définir par l'élève)
    - Au moins 8 boutons-poussoirs + Leds, dimension et montage à définir
    - 1 potentiomètre sur ADC de l'ESP32
    - Voir pour les protections galvaniques

- Software embarqué
    - Implémentation de MQTT Publisher/Subscriber,
    - Drivers nécessaires (ADC, stack TCP/IP, ...)
    - Mise à jour E/S de la carte via MQTT (Bouttons et LEDs)
    - Configuration MQTT via un fichier JSON
    - (SECONDAIRE) Configuration client MQTT
        - Ajouter un serveur web (requête REST) pour configurer MQTT ou autre solution OU
        - Utiliser l'antenne Bluetooth / WLAN de l'ESP32 pour le configurer

- Software MQTT Broker sur le PLC (windows)
    - Installation d'un Broker sur le PLC (Exemple : Mosquitto)

- Gateway MQTT <-> ADS
    - Programme en python
    - Lire écrire des variables dans le PLC
    - Lier les variables du PLC aux informations disponibles du Broker MQTT

- Mécanique
    - Montage des composants sur le PCB
    - Montage du PCB sur la platine Help-tec
    - Test avec une peau (Étiquettes autocollantes) A4 ou autre format à imprimer soi-même

Contraintes

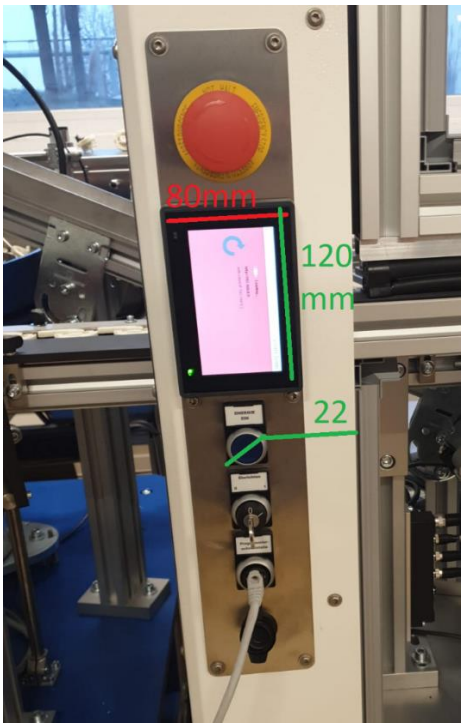
- Prix du PCB sans les boutons mécaniques : 20 - 30 CHF (PCB + composant)
- Latence entre pression du bouton et changement d'état dans la Runtime Twincat : 200 ms
- Utilisations de composants disponibles actuellement
- Design du PCB avec KiCAD
- MCU
    - ESP32-WROOM, révision au choix (S1/S2/S3)
        - Flash size : à définir
        - Choisir la version avec antenne en PCB
    - Code en C + FreeRTOS
    - Utilisation de Git, code héberger sur Github dans un repo public chez Help-tec
    - IDE : VScode ou Eclipse
    - toolchain de Espressif + ESP-IDF (pas la version pour Arduino)

Dimension

Le PCB doit s'intégrer sur une platine en acier. Celle-ci sera découpée à la commande pour correspondre aux dimensions du PCB. Le placement des boutons doit être le mieux possible pour une utilisation ergonomique. Une peau autocollante sera ajoutée au-dessus de la platine et des boutons pour afficher un symbole de la fonction du bouton et cacher le PCB.

- Les dimensions exactes du PCB sont à définir selon la disposition sur la platine de montage
    - Largeur : 40 – 60 mm
    - Longueur : 60 – 90 mm
    - Hauteur : Selon bouton et platine de montage
- Les boutons ou le capuchon du bouton doivent avoir un diamètre d'au moins 10mm. La disposition des boutons sur le PCB devra être validée par Help-tec.
- Fixation du PCB à la platine : TBD
    - Avec des goujons intégrés à la platine et filetage M3
    - Perçage puis vis + écrou
    - Autre

Exemple du montage de la platine avec la solution actuelle, avec dimension :



Matériel

Pour la réalisation du projet, les installations de l'HE-ARC St-Imier sont à privilégier. Seul le matériel pour le montage du PCB ainsi qu'un câble Ethernet est requis pour effectuer les tests sur le PCB. Afin de gagner du temps, Help-tec met à disposition un kit de développement comprenant une carte avec un MCU ESP32 alimenté via PoE et/ou usb.

La partie concernant le PLC peut être intégralement réalisée sur le Laptop de l'élève.

Si besoin et pour la démonstration, le matériel suivant peut être mis à disposition de la part de Help-tec à l'élève :

- Alimentation 24VDC
- PLC Beckhoff CX9020
- Switch PoE
- Divers câbles RJ45 Cat5 / Cat6

Fonctions

Roger Schluep est le chef de projet côté Help-tec

- Coordonne la réalisation de la partie software avec l'élève
- Mets à dispositions les ressources nécessaires en ce qui concerne Help-tec
- Valide les points importants

Sacha Wittmann est le directeur général de Help-tec

- Valide les points concernant les coûts et l'esthétique du produit (si nécessaire)

Yves Meyer est le professeur responsable du suivi de la thèse de Master

- Suivi hebdomadaire du projet

Journal de travail

RESUME DES ACTIVITES DE THBAULT SAMPIEMON AU SEIN DU TRAVAIL DE MASTER

Thibault Sampiemon | TM- Interface standardisée communication entre PLCs et carte embarquée| 21.02.2022

Table des activités

Date	Nombre d'heures	Travail effectué	Efficacité autoévaluée (0-10)
21.02.22	8	• Téléchargement des logiciels et bases de travail (ex : projet open source de carte de démonstration ESP32) • Entretien avec monsieur Meyer • Correction des planifications du projet suite à un entretien avec monsieur Meyer • Divers administratifs d'entrée en fonction dans les locaux de St-Imier (ex :accès) • Découverte de la carte exemple Olimex sur KiCAD • Découverte de KiCAD	5
22.02.22	7	• TUTO KiCAD • Etude de disponibilité des composants de la carte Olimex • Changements et étude des possibilités du changement des composants de la carte Olimex (utilisation d'une carte toute faite pour les parties « standards ») • Divers administratifs (accès suite et fin) • Recherche de composants de remplacement PoE pour refaire la carte Olimex avant la décision de garder la carte Olimex et créer une carte fille pour le premier prototype de PCB	5
24.02.22	8	• Choix composants carte fille et schéma block pour les boutons poussoirs • TUTO KiCAD • Choix GPIO	8
25.02.22	8	• Choix composants carte fille et schéma block pour le potentiomètre • Choix de l'entrée ADC • Changement de GPIO pour un GPIO avec interruption	8
26.02.22	5	• Calculs des composants électroniques et renseignements sur les composants électroniques • Choix des tensions • Recherche de bouton avec LED	7
28.02.22	7	• Réunion avec monsieur Meyer • Recherche d'un nouveau GPIO changement de I2C à SPI • Début schéma électrique	7
1.03.22	8	• Choix trigger de Schmitt	7

		• Suite de recherche d'un nouveau GPIO changement de I2C à SPI • Choix AOP • Choix boutons poussoirs	
3.03.22	8	• Choix Final GPIO • Choix de 2 types de boutons • Calculs de résistances pour les RC et les LEDs	8
4.03.22	7	• Implémentation du schéma sur KiCAD • Lecture de documentation pour le choix des composants • Choix final des boutons • Calculs pour la détermination des packages des résistances en fonction de la puissance	7
5.03.22	6	• Lecture doc +tuto divers • Changement de toutes les tensions pour du 5 V dans le but d'avoir une seule tension (les leds nécessitent 5 V pour avoir toutes les couleurs)	6
7.03.22	8	• Changement de GPIO vers un I2C (mais pas le même que le précédent) • Nouveau choix de Trigger de Schmitt • Changement avec 2 types de boutons suite à la demande du mandant	10
8.03.22	8	• Changement vers 1 seul type de bouton suite à demande du mandant • Installation software / Git avec mandant • Changement label au lieu de longues pistes • Affichage des tensions à la place de vcc suite à la demande du mandant	8
10.03.22	8	• Changement label à la place de longues pistes suite à la demande du mandant • Changement de tension pour du 3.3V suite à la demande du mandant • Recherche d'un régulateur de courant suite à la demande du mandant • Regroupement des composants par zones de schéma	6
11.03.22	8	• Changement des leds vers 5V suite à la demande du mandant (demande de 2 tensions différentes explicites) • Rajout du régulateur • Rajout de toutes les empreintes manquantes	5
12.03.22	6	• Début du placement des composants • Tuto KiCAD PCB	5

		• Regroupement des composants par zones de schéma : suite	
14.03.22	13	• Continuation du placement des composants • Changement du schéma, rajout de sous-schémas pour chaque bouton • Changement de police pour les commentaires /regroupements • Rajout de commentaires au niveau des LEDs • Superposition des alternatives résistances et régulateurs pour les LEDs • Assistance pour les changements du mandant d'AOP et des connexions de la carte ESP	4
15.03.22	8	• Vérification électrique de l'AOP • Changements schéma • Suppression reset et changements de pins • Repositionnement des composants	5
17.03.22	8	• Repositionnement des composants • Tuto KiCAD • Début routing	8
18.03.22	8	• Routing • Configuration du PCB • Changements par rapport à la documentation JLCPCB	8
19.03.22	8	• Routing + vias + fanout	10
21.03.22	8	• Fin du routage • Changements d'après les conseils de monsieur Meyer • Adaptation pour des trous M4 ISO • Adaptation/debug avec les outils KiCAD	8
22.03.22	12	• Révision + review + contrôle PCB + schéma • Adaptation du PCB pour la compatibilité avec JLCPCB • Debug de la carte et résolution des erreurs • Nettoyage général du PCB	7
24.03.22	8	• Génération des fichiers gerber selon la spécification du fabricant • Révisions de dernière minute	8
25.03.22	8	• Mise en place de l'environnement de développement software	6
26.03.22	8	• Mise en place de l'environnement de développement software	6
28.03.22	8	• Création et envoi du panier	6

		• Mise en place de l'environnement de développement software • Découverte du problème de non-support des chemins avec espace du build de la toolchain	
29.03.22	13	• Changement de GPIO du MCP23… vers deux PCA74.. schéma + PCB (absence de stock/crise Covid) • Vérification avec documentation des GPIO • Changements du panier, adaptations des résistances et GPIO	9
31.03.22	8	• Changement de GPIO du MCP23… vers deux PCA74.. schéma + PCB (absence de stock/crise Covid) • Finalisation et debug du PCB fini	9
1.04.22	8	• Vérification du panier • Génération fichier gerber	9
4.04.22	9	• Mise en place de l'environnement de développement (régler le problème de non support des noms de dossiers avec un espace pour certains outils) • Création nouvelle session et installation logicielle (bug chemin avec espace dans le dossier racine de l'utilisateur)	5
5.04.22	8	• Remise en place de esp-idf et vscode	4
7.04.22	10	• Debug environnement vscode • Abandon vscode pour esp-ide • Mise en place FreeRTOS	5
8.04.22	8	• Faire tourner FreeRTOS • Test hello world	7
9.04.22	8	• Début programme ADC	7
11.04.22	13	• Programme FreeRTOS squelette • Création veroboard de test	7
12.04.22	10	• Programme ADC minimum • Trouver une erreur de nomination dans le schéma PCB, un changement de pin pour l'ADC sans changement de chanel • Modification veroboard	8
14.04.22	8	• Intégration à FreeRTOS de l'ADC	7
15.04.22	11	• Mise au propre du programme • Régler le problème des CMAKE avec les fichiers multiples	7
16.04.22	8	• Fin ADC • Début interruption	8
18.04.22	9	• Changement veroboard pour test d'interruption • Fin interruption	9

		• Début I2C	
19.04.22	11	• Mise en place I2C • Lecture de trame avec analyseur logique	10
24.04.22	8	• Fin I2C pour partie testable avec analyseur logique	10
25.04.22	8	• Faire le point client + démo veroboard • Installation TIA portal + tests • Début montage PCB	7
26.04.22	8	• Montage PCB • Remplacement des composants manquants	9
28.04.22	8	• Test du PCB • Découverte et résolution du problème du suiveur (modification PCB)	7
29.04.22	8	• Modification pour des trigger de Schmitt non inverseurs • Découverte problème de différence de tension GPIO vs led et test d'une solution	7
30.04.22	8	• Modification du code pour adapter le code au PCB réel • Détection d'un problème pour l'interprétation des messages venant du GPIO par I2C	7
02.05.22	8	• Debug des problèmes d'interprétation des messages venant du GPIO par I2C	9
03.05.22	8	• Création de fonctions de tests supplémentaires pour les GPIO • Modifications du code demandé par le client	6
05.05.22	8	• Modifications du code demandé par le client (résolution ADC, trigger de Schmitt,..)	7
06.05.22	8	• Tests finaux du PCB • Début de la communication MQTT	6
07.05.22	8	• Commencement MQTT sur la carte avec TCP/IP esp32	6
09.05.22	8	• Continuation TCP/IP esp32	6
10.05.22	8	• Continuation TCP/IP esp32	6
12.05.22	8	• Changement pour une stratégie avec MQTT directement avec l'exemple esp32 • Configuration broker MQTT	6
13.05.22	8	• Continuation MQTT • Continuation configuration broker MQTT • Continuation modifications d'un client Python pour tester les valeurs du broker	7
14.05.22	8	• Continuation MQTT	6

		• Continuation configuration broker MQTT • Continuation modifications d'un client Python pour tester le broker	
16.05.22	8	• Changement sur conseil de monsieur Schluep sur une communication wifi + MQTT plus simple car directement intégrée à WROOM	7
17.05.22	8	• Wifi + MQTT communication avec un broker web (réussite)	6
19.05.22	8	• Création d'une meilleure communication entre le broker mosquitto et le client Python • Continuation configuration broker	6
20.05.22	8	• Création d'une communication entre le client Python et le PLC+ Test avec un programme simple sur le PLC	6
21.05.22	8	• Continuation configuration broker • Continuation partie esp32 de MQTT	6
23.05.22	8	• Debug de la communication (wireshark)	5
24.05.22	8	• Debug de la communication (wireshark)	5
25.05.22	8	• Debug de la communication (wireshark+ oscilloscope)	7
30.05.22	10	• Résolution du problème sur la carte (les délais à rajouter sur le driver)	10
31.05.22	8	• Création du DHCP	10
02.06.22	9	• Debug avec monsieur Florian Sauser et compréhension du problème du firewall	10
03.06.22	10	• Acheminement des messages jusqu'au client Python	10
04.06.22	10	• Retour du message du PLC à la carte embarquée	10
06.06.22	12	• Adaptation des messages pour que cela corresponde avec les tâches des boutons	10
07.06.22	9	• Adaptation des messages pour que cela corresponde avec les Tâches des LEDs	10
09.06.22	12	• Finalisation de la communication boutons + LEDs • Création de la communication pour le potentiomètre	10
10.06.22	12	• Adaptation du PCB pour le changement du GPIO • Changement pour la tension des LEDs sur le PCB	10
11.06.22	9	• Modification pour le branchement du potentiomètre • Reroutage et adaptation de la taille du PCB	10

		• Génération des fichiers gerber et contrôle des spécifications de JLCPCB	
13.06.22	11	• Création de résolutions automatiques de problèmes de connexion sur la carte embarquée	10
14.06.22	15	• Finalisation de la communication entre le broker et la carte embarquée • Vidéo de démo finale	10
15.06.22	8	• Amélioration du client Python pour qu'il résolve certains problèmes de connexion de façon automatique • Mise au propre des programmes et commentaires et directives de compilation pour le debug	10
16.06.22	8	• Mise au propre des programmes et commentaires et directives de compilation pour le debug	6
17.06.22	8	• Mise au propre des programmes et commentaires • Tests fonctionnels avec une seule carte embarquée	10
20.06.22	8	• Tests fonctionnels avec deux cartes envoyant des fausses données	3
23.06.22	9	• Tests fonctionnels avec deux cartes envoyant des fausses données • Tests fonctionnels des résolutions automatiques de problèmes	6
24.06.22	12	• Tests fonctionnels des résolutions automatiques de problèmes • Tests fonctionnels des démarrages • Mise au propre du journal de travail	6
25.06.22	8	• Documentation • Rapport	6
27.06.22	8	• Rapport	2
30.06.22	9	• Rapport	5
01.07.22	8	• Rapport	5
02.07.22	8	• Rapport	6
04.07.22	10	• Rapport	7
05.07.22	10	• Rapport	5
06.07.22	12	• Rapport	4
07.07.22	8	• Poster • Relecture et correction Rapport et Poster	4
08.07.22	6	• Rapport, Poster et réunion des document	3