Université CADI AYYAD
Ecole Nationale des Sciences Appliqués
Ministère de l'Enseignement Supérieur, de la Recherche Scientifique et de l'Innovation
Marrakech

Filière : Systèmes Electroniques Embarqués et Commande des Systèmes

RAPPORT DE PROJET DE FIN DE SEMESTRE

THEME :

Commande à distance et régulation de la vitesse d'une trottinette électrique

Réalisé par :

ANLOUF Amine
AL-AALAOUI Mehdi
HANINE NOUHAILA
BENMESSAOUD Aymane
RAZZOUG Oussama

Encadré par :

Lhoussain EL BAHIR

Soutenu le :

11/06/2024

Membres de jury :

Lhoussain EL BAHIR
Touria Hassboun
Khalid faitah

Année Universitaire 2024

Remerciements

Nous tenons à exprimer notre profonde gratitude à tous ceux qui ont contribué à la réalisation de ce projet de fin de semestre intitulé "Commande à distance et régulation de la vitesse d'une trottinette électrique".
En premier lieu, nous remercions Dieu Tout-Puissant pour nous avoir donné la force, la santé, et la persévérance nécessaires pour mener à bien ce projet. Sans Sa bénédiction, rien de tout cela n'aurait été possible.
Nous adressons nos sincères remerciements à notre encadrant, Mr. Lhoussain EL BAHIR, pour ses conseils précieux, son soutien constant, et son encadrement tout au long de ce projet. Son expertise et ses encouragements ont été essentiels pour surmonter les défis que nous avons rencontrés.
Nous remercions également l'École Nationale des Sciences Appliquées de Marrakech pour avoir fourni le matériel et les ressources nécessaires à la réalisation de ce projet. Leur soutien logistique et matériel a été déterminant pour la réussite de nos travaux pratiques et expérimentaux.
Un remerciement particulier à Mr. Khalid pour son assistance technique et son encadrement. Ses compétences techniques et ses conseils pratiques ont grandement contribué à la mise en œuvre de notre projet. Sa disponibilité et sa volonté d'aider ont été inestimables.
Enfin, nous souhaitons exprimer notre reconnaissance à nos familles et amis pour leur soutien moral et leurs encouragements tout au long de ce projet. Leur présence et leur compréhension ont été des sources de motivation inépuisables.
À tous ceux qui ont, de près ou de loin, contribué à la réussite de ce projet, nous adressons nos remerciements les plus sincères.

Sommaire

- Pourquoi MIT App Inventor ?
- Interface utilisateur et fonctionnalités

Chapitre 2 : Vitesse de retour et régulation

1.Boucle fermée et étalonnage
- boucle fermée
- Procédures d'étalonnage

2. Code retour
- Lecture des données de vitesse
- Transmission des données

3. Retour application mobile
- Affichage des données
- Mise à jour en temps réel des données de vitesse sur l'application mobile

4. Régulation PI
- Théorie de la régulation PI
- Implémentation de la régulation PI dans le code
- Envoi de Kp et Ki par l'application mobile

Introduction :

- Présentation du projet

Le projet de fin de semestre que nous avons choisi porte sur la commande à distance de la vitesse d'une trottinette électrique. L'objectif principal de ce projet est de concevoir et de mettre en œuvre un système permettant de contrôler la vitesse d'une trottinette électrique via une application mobile. Cette commande à distance vise à offrir une expérience utilisateur améliorée, en permettant un contrôle plus précis et intuitif de la trottinette, tout en intégrant des fonctionnalités de retour d'information en temps réel pour une régulation optimale de la vitesse.

Le projet s'inscrit dans le cadre des avancées technologiques actuelles, où l'Internet des objets (IoT) joue un rôle crucial dans le développement de solutions innovantes et connectées. En utilisant des composants tels que l'ESP32 pour la connectivité Wi-Fi, le protocole MQTT pour la communication, et des capteurs pour la mesure de la vitesse, nous visons à créer un système complet et fonctionnel.

La trottinette électrique, en tant que moyen de transport personnel de plus en plus populaire, se prête particulièrement bien à l'intégration de technologies de contrôle à distance. Le projet comprend plusieurs étapes clés, allant de la conception de l'application mobile à la mise en œuvre de la régulation de vitesse, en passant par la configuration des composants matériels et logiciels.

Le développement de ce projet nous permet de mettre en pratique une variété de compétences acquises au cours de notre formation, telles que la programmation, l'électronique, et les systèmes embarqués. Il nous offre également l'opportunité de nous familiariser avec des technologies et des protocoles de communication modernes, et de comprendre leur application concrète dans un contexte de mobilité urbaine.

En somme, ce projet vise non seulement à démontrer notre capacité à intégrer différentes technologies pour résoudre un problème pratique, mais aussi à explorer les possibilités offertes par l'IoT pour améliorer et optimiser les dispositifs de transport personnels. La commande à distance de la vitesse d'une trottinette électrique n'est qu'un exemple parmi tant d'autres des innovations possibles dans ce domaine en pleine expansion.

- Objectifs

Les objectifs de ce projet de fin de semestre se déclinent en plusieurs points, chacun visant à garantir la réussite et la fonctionnalité de notre solution de commande à distance de la vitesse d'une trottinette électrique. Ces objectifs se concentrent sur la réalisation technique, l'expérience utilisateur, et l'innovation technologique. Voici les principaux objectifs de notre projet :

1. Développement d'une application mobile intuitive :
    - Concevoir une application mobile conviviale pour Android, permettant de contrôler la vitesse de la trottinette électrique de manière intuitive et réactive.
    - Intégrer une interface utilisateur (UI) claire et fonctionnelle avec des contrôles simples pour ajuster la vitesse et surveiller les performances en temps réel.

2. Implémentation de la connectivité Wi-Fi :
    - Utiliser l'ESP32 pour établir une connexion Wi-Fi stable entre la trottinette électrique et l'application mobile.
    - Configurer et sécuriser cette connexion pour assurer une communication fiable et sécurisée.

3. Intégration du protocole MQTT pour la communication :
    - Mettre en place un broker MQTT (HiveMQ Broker) pour gérer la communication entre l'application mobile et l'ESP32.
    - Assurer la transmission efficace des commandes de vitesse et des données de retour de vitesse entre l'application et la trottinette.

4. Contrôle précis de la vitesse via PWM :
    - Utiliser la modulation de largeur d'impulsion (PWM) pour contrôler précisément la vitesse du moteur de la trottinette électrique.
    - Développer des algorithmes de contrôle PWM et les implémenter sur l'ESP32 pour ajuster la vitesse du moteur en fonction des commandes reçues.

5. Utilisation du L298P Motor Driver Shield :
    - Intégrer et configurer le L298P Motor Driver Shield pour contrôler le moteur de la trottinette électrique.
    - Câbler correctement le driver et l'ESP32, et développer le code nécessaire pour leur interaction harmonieuse.

6. Retour d'information et régulation de la vitesse :
    - Implémenter des capteurs pour mesurer la vitesse actuelle de la trottinette et transmettre ces données à l'ESP32.
    - Mettre en place un système de retour d'information en temps réel permettant à l'utilisateur de voir la vitesse actuelle sur l'application mobile.

- Développer un algorithme de régulation PI (Proportionnel-Intégral) pour maintenir une vitesse constante en fonction des conditions de conduite et des commandes de l'utilisateur.

7. Étalonnage et optimisation des capteurs :
    - Effectuer des procédures d'étalonnage pour garantir la précision des capteurs de vitesse.
    - Optimiser la performance du système de contrôle et de régulation de la vitesse en ajustant les paramètres de l'algorithme PI.

8. Sécurité et fiabilité du système :
    - Assurer que le système de commande à distance est sûr et ne présente aucun risque pour l'utilisateur.
    - Mettre en place des mesures de sécurité pour éviter toute commande non autorisée ou erronée qui pourrait entraîner une défaillance de la trottinette.

En atteignant ces objectifs, nous visons à créer une solution complète, fiable et innovante pour la commande à distance de la vitesse d'une trottinette électrique. Ce projet non seulement enrichit notre expérience académique, mais contribue également à l'avancement des technologies de transport personnel intelligent et connecté.

- Description générale de la trottinette électrique et des composants utilisés

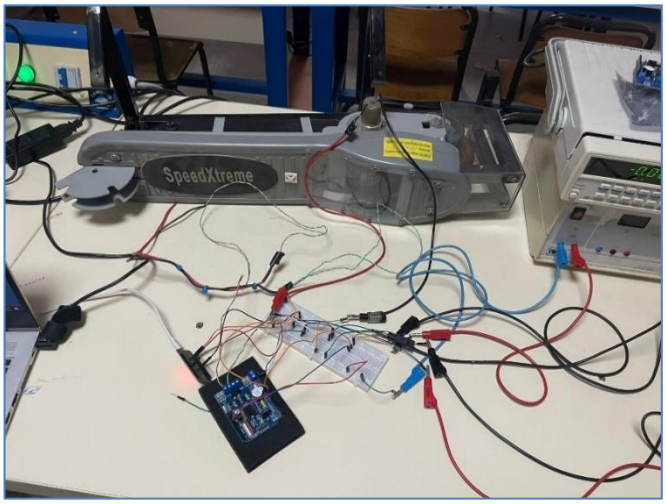
-introduction

Les trottinettes électriques ont révolutionné la mobilité urbaine en offrant une solution de transport pratique, écologique et économique. Parmi ces innovations, les trottinettes électriques équipées de moteurs à courant continu (DC) et contrôlées à distance via une application mobile se démarquent par leur modernité et leur convivialité. Ces dispositifs combinent la robustesse et l'efficacité des moteurs DC avec les avantages de la connectivité mobile, offrant ainsi une expérience utilisateur enrichie et simplifiée.
Ce document présente une trottinette électrique équipée d'un moteur à courant continu (DC), contrôlée à distance via une application mobile grâce à un module ESP32 avec connectivité Wi-Fi et un contrôleur de moteur L298P.

-principe de fonctionnement

les composants clés: Moteur à Courant Continu (DC) Le moteur à courant continu est le cœur de cette trottinette électrique. Il offre une réponse rapide et un couple élevé, ce qui permet une accélération fluide et efficace. source de tonsion source d'énergie de la trottinette Elle alimente le moteur DC ainsi que les composants électroniques. ESP32 avec Wi-Fi Intégré Le module ESP32 est un microcontrôleur avec connectivité Wi-Fi intégrée, utilisé pour gérer la communication entre la trottinette et l'application mobile. Il permet de contrôler les paramètres de la trottinette en temps réel via une connexion sans fil. Hacheur L298P Le hacheur L298P est un module de commande de moteur capable de contrôler des moteurs DC dans les deux sens. Il reçoit les signaux de commande de l'ESP32 et régule la puissance envoyée au moteur, permettant ainsi le contrôle de la vitesse et de la direction. Application Mobile

-Schéma de Connexion



ESP32 : Connecté au réseau Wi-Fi, il communique avec l'application mobile.
L298P : Connecté à l'ESP32 pour recevoir les signaux de commande et réguler la puissance au moteur DC.
Moteur DC : Reçoit la puissance régulée du L298P.
source de tonsion : Alimente à la fois le moteur DC et les circuits électroniques de l'ESP32 et du L298P.

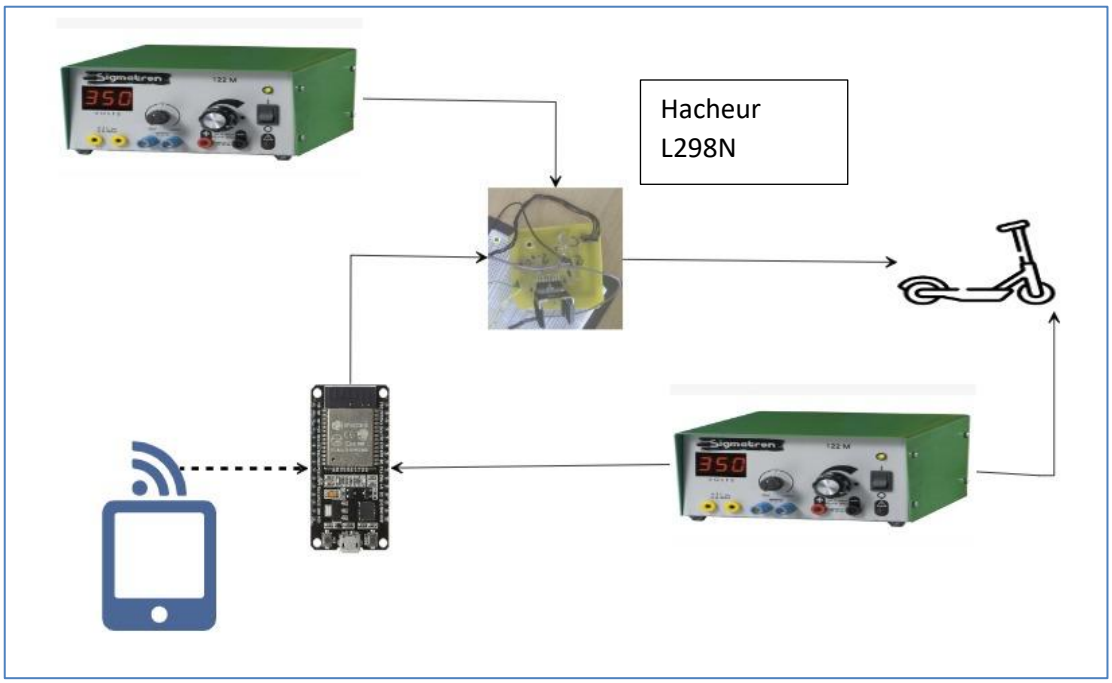
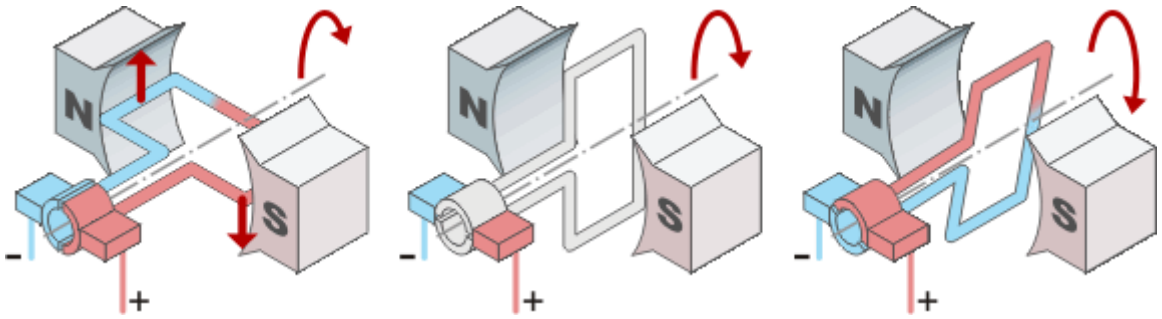


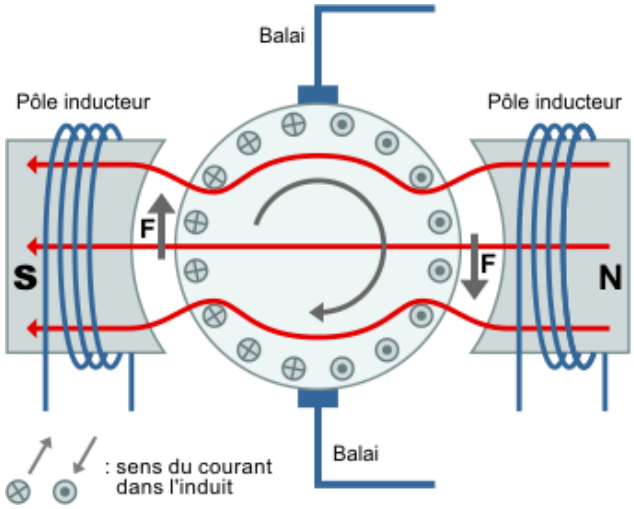
Schéma de Connexion

-Documentation de moteur DC

moteur DC principe de fonctionnement : Le moteur à courant continu se compose : de l'inducteur ou du stator, de l'induit ou du rotor, du collecteur et des balais.
Lorsque le bobinage d'un inducteur de moteur est alimenté par un courant continu, sur le même principe qu'un moteur à aimant permanent (comme la figure ci-dessous), il crée un champ magnétique (flux d'excitation) de direction Nord-Sud.
Une spire capable de tourner sur un axe de rotation est placée dans le champ magnétique. De plus, les deux conducteurs formant la spire sont chacun raccordés électriquement à un demi collecteur et alimentés en courant continu via deux balais frotteurs.
D'après la loi de Laplace (tout conducteur parcouru par un courant et placé dans un champ magnétique est soumis à une force), les conducteurs de l'induit placés de part et d'autre de l'axe des balais (ligne neutre) sont soumis à des forces F égales mais de sens opposé en créant un couple moteur : l'induit se met à tourner !



Si le système balais-collecteurs n'était pas présent (simple spire alimentée en courant continu), la spire s'arrêterait de tourner en position verticale sur un axe appelé communément "ligne neutre". Le système balais-collecteurs a pour rôle de faire commuter le sens du courant dans les deux conducteurs au passage de la ligne neutre. Le courant étant inversé, les forces motrices sur les conducteurs le sont aussi permettant ainsi de poursuivre la rotation de la spire.

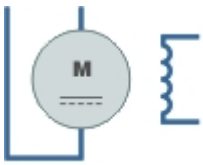


Dans la pratique, la spire est remplacée par un induit (rotor) de conception très complexe sur lequel sont montés des enroulements (composés d'un grand nombre de spires) raccordés à un collecteur "calé" en bout d'arbre. Dans cette configuration, l'induit peut être considéré comme un seul et même enroulement semblable à une spire unique.

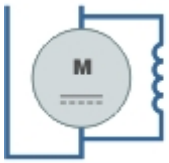
Type de moteur à courant continu

Suivant l'application, les bobinages du l'inducteur et de l'induit peuvent être connectés de manière différente. On retrouve en général :

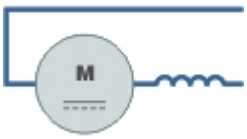
Des moteurs à excitation indépendante.



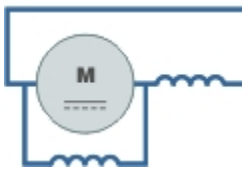
Des moteurs à excitation parallèle.



Des moteurs à excitation série.



Des moteurs à excitation composée.



La plupart des machines d'ascenseur sont configurées en excitation parallèle ou indépendante. L'inversion du sens de rotation du moteur s'obtient en inversant soit les connections de l'inducteur soit de l'induit.

-L'inducteur

L'inducteur d'un moteur à courant continu est la partie statique du moteur. Il se compose principalement :

- de la carcasse,
- des paliers,

- des flasques de palier,
- des portes balais.

Le cœur même du moteur comprend essentiellement :

- Un ensemble de paires de pôles constitué d'un empilement de tôles ferro-magnétiques.
- Les enroulements (ou bobinage en cuivre) destinés à créer le champ ou les champs magnétiques suivant le nombre de paires de pôles.

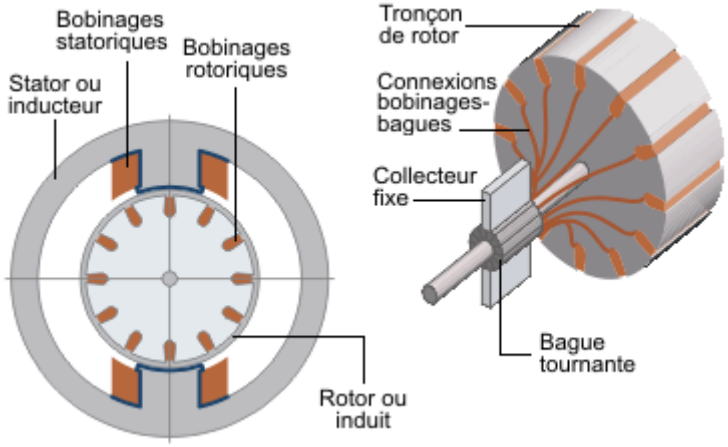
Pour des moteurs d'une certaine puissance, le nombre de paires de pôles est multiplié afin de mieux utiliser la matière, de diminuer les dimensions d'encombrement et d'optimaliser la pénétration du flux magnétique dans l'induit.

-L'induit

L'induit du moteur à courant continu est composé d'un arbre sur lequel est empilé un ensemble de disques ferro-magnétiques. Des encoches sont axialement pratiquées à la périphérie du cylindre formé par les disques empilés. Dans ces encoches les enroulements (bobines de l'induit) sont "bobinés" selon un schéma très précis et complexe qui nécessite une main d'œuvre particulière (coûts importants). Pour cette raison, on préfère, en général, s'orienter vers des moteurs à courant alternatif plus robuste et simple dans leur conception.

Chaque enroulement est composé d'une série de sections, elles même composées de spires; une spire étant une boucle ouverte dont l'aller est placé dans une encoche de l'induit et le retour dans l'encoche diamétralement opposée. Pour que l'enroulement soit parcouru par un courant,

ses conducteurs de départ et de retour sont connectés aux lames du collecteur (cylindre calé sur l'arbre et composé en périphérie d'une succession de lames de cuivre espacée par un isolant).



L'interface entre l'alimentation à courant continu et le collecteur de l'induit est assuré par les balais et les porte-balais.

-Les balais

Les balais assurent le passage du courant électrique entre l'alimentation et les bobinages de l'induit sous forme d'un contact par frottement. les balais sont en graphite et constituent, en quelques sortes, la pièce d'usure. Le graphite en s'usant libère une poussière qui rend le moteur à courant continu sensible à un entretien correct et donc coûteux.

Le point de contact entre les balais et le collecteur constitue le point faible du moteur à courant continu. En effet, c'est à cet endroit, qu'outre le problème d'usure du graphite, la commutation (inversion du sens du courant dans l'enroulement) s'opère en créant des micros-arcs (étincelles) entre les lamelles du collecteur; un des grands risques de dégradation des collecteurs étant leur mise en court-circuit par usure.

Pilotage de la vitesse de rotation

❖ Relation Vitesse et force contre-électromotrice à flux constant

Lorsque l'induit est alimenté sous une tension continue ou redressée U, il se produit une force contre-électromotrice E.
On a :

$$E = U - R\,x\,I\,[volts]$$

Où,

- R = la résistance de l'induit [ohm].
- I = le courant dans l'induit [ampère].

La force contre-électromotrice est liée à la vitesse et à l'excitation du moteur.
On a :

$$E = k\,x\,\omega\,x\,\Phi[volt]$$

Où,

- k = constante propre au moteur (dépendant du nombre de conducteurs de l'induit).
- ω = la vitesse angulaire de l'induit [rad/s].
- Φ= le flux de l'inducteur [weber].

En analysant la relation ci-dessus, on voit, qu'à excitation constante Φ, la force contre-électromotrice E est proportionnelle à la vitesse de rotation.

❖ Relation Couple et flux

Quant au couple moteur, il est lié au flux inducteur et au courant de l'induit par la relation suivante.
On a :

$$C = k\,x\,\Phi\,x\,I\,[N.m]$$

Où,

- k = constante propre au moteur (dépendant du nombre de conducteurs de l'induit).
- Φ= le flux de l'inducteur [weber].
- I = le courant dans l'induit [ampère].

En analysant la relation ci-dessus, on voit qu'en réduisant le flux, le couple diminue.

❖ Variation de la vitesse

Au vu des relations existant entre la vitesse, le flux et la force contre-électromotrice, il est possible de faire varier la vitesse du moteur de deux manières différentes. On peut :

- Augmenter la force contre-électromotrice E en augmentant la tension au borne de l'induit tout en maintenant le flux de l'inducteur constant. On a un fonctionnement dit à "couple constant". Ce type de fonctionnement est intéressant au niveau de la conduite d'ascenseur.
- Diminuer le flux de l'inducteur (flux d'excitation) par une réduction du courant d'excitation en maintenant la tension d'alimentation de l'induit constante. Ce type de fonctionnement impose une réduction du couple lorsque la vitesse augmente.

❖ Caractéristiques

Les avantages et inconvénients du moteur à courant continu sont repris ci-dessous :

-Avantages:

- accompagné d'un variateur de vitesse électronique,il possède une large plage de variation (1 à 100 % de la plage).
- régulation précise du couple,
- son indépendance par rapport à la fréquence du réseau fait de lui un moteur à large champ d'application,

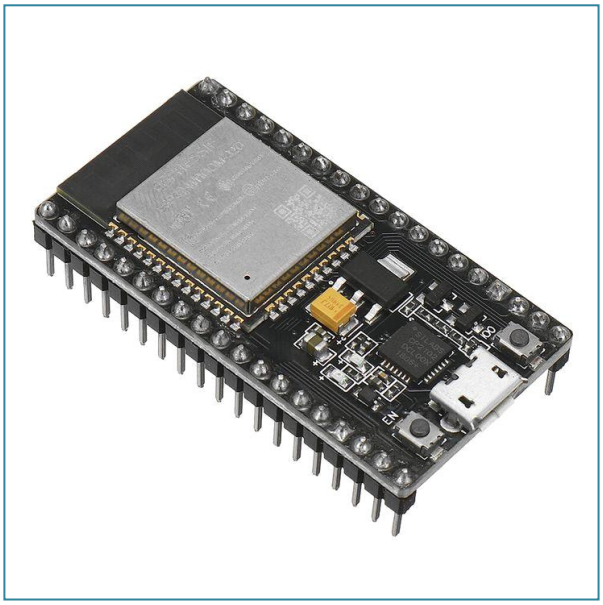
-Inconvénients:

- peu robuste par rapport au machine asynchrone
- investissement important et maintenance coûteuse (entretien du collecteur et des balais,

Chapitre 1 : Commande de vitesse par application mobile

1. ESP32

- Description de l'ESP32



L'ESP32 est un microcontrôleur SoC (System on Chip) développé par Espressif Systems. Il est conçu pour les applications de connectivité sans fil et l'Internet des Objets (IoT).

L'ESP32 est une puce microcontrôleur intégrée qui offre des capacités de connectivité Wi-Fi et Bluetooth. Elle est équipée d'un processeur dual-core, d'une mémoire SRAM, et de divers périphériques tels que des GPIOs, ADCs, DACs, UART, SPI, et I2C. L'ESP32 est largement utilisé dans les applications IoT, la domotique, les dispositifs portables, et d'autres projets électroniques nécessitant une connectivité sans fil et des performances élevées à un coût abordable.

Caractéristiques principales :

-Processeur: Dual-core Tensilica Xtensa LX6, 160 MHz à 240 MHz.
- Mémoire : 520 Ko de SRAM.
- Connectivité: Wi-Fi 802.11 b/g/n, Bluetooth 4.2 BR/EDR et BLE.
- Périphériques : GPIOs, ADCs, DACs, UART, SPI, I2C.
- Capteurs intégrés : Capteur de hall, capteur de température.
- Support de développement : SDKs (ESP-IDF), compatible avec Arduino et MicroPython.

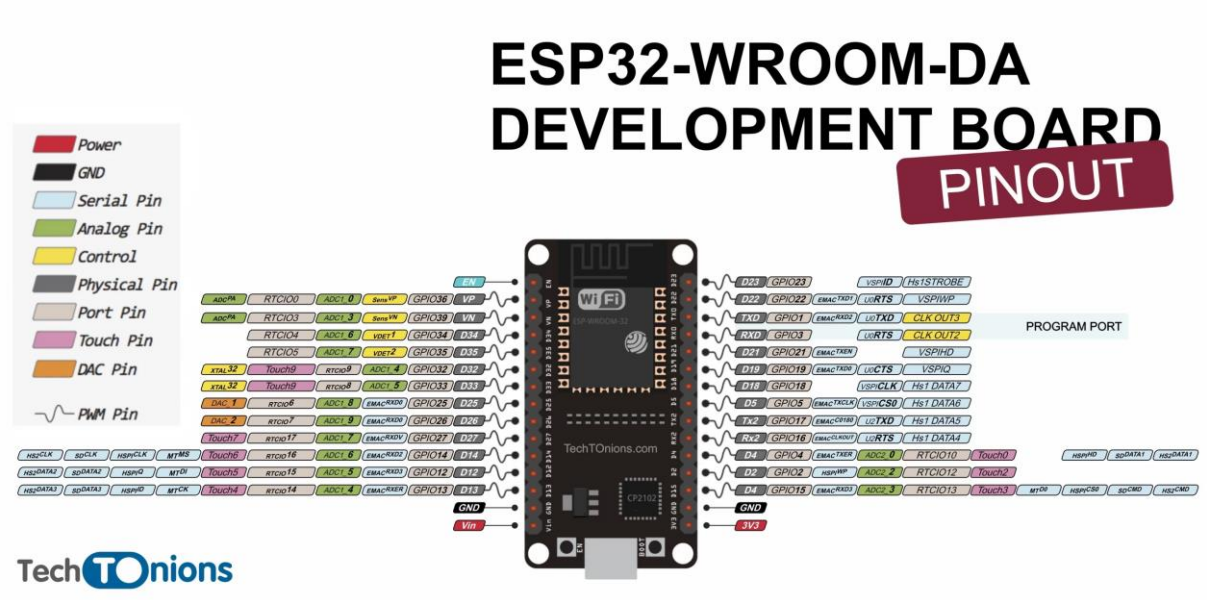
Utilisations typiques :

- Domotique : Contrôle d'éclairage, thermostats intelligents.
- IoT : Collecte de données, surveillance à distance.
- Dispositifs portables : Montres intelligentes, bracelets de fitness.
- Projets électroniques : Gadgets connectés, robots.

L'ESP32 est reconnu pour sa polyvalence, ses performances et son accessibilité, ce qui en fait un choix populaire parmi les développeurs et les hobbyistes en électronique.

- Documentation des pins (pinout)

Les pins de l'ESP32 offrent une grande flexibilité pour diverses interfaces et applications. Voici une description détaillée des pins d'entrées/sorties (GPIO) et des fonctionnalités associées de l'ESP32 :



1. Alimentation :

- 3V3: Sortie 3.3V pour l'alimentation des périphériques externes.
- GND: Masse commune.

2. GPIO (General Purpose Input/Output) :

- GPIO0 à GPIO39 : Pins GPIO polyvalentes pouvant être configurées comme entrées ou sorties numériques. Certaines de ces pins ont des fonctions spéciales ou des restrictions.

3. Communication série :

- UART :
  - TX0 (GPIO1): Transmetteur pour UART0.
  - RX0 (GPIO3) : Récepteur pour UART0.
  - TX1 (GPIO10) : Transmetteur pour UART1.
  - RX1 (GPIO9) : Récepteur pour UART1.
  - TX2 (GPIO17) : Transmetteur pour UART2.
  - RX2 (GPIO16) : Récepteur pour UART2.
- SPI :
  - MOSI (GPIO23): Master Out Slave In.
  - MISO (GPIO19): Master In Slave Out.
  - SCK (GPIO18): Serial Clock.
  - SS (GPIO5) : Slave Select.
- I2C :
  - SDA (GPIO21): Data line.
  - SCL (GPIO22) : Clock line.
- I2S :
  - SD (GPIO32) : Serial Data.
  - WS (GPIO25) : Word Select.

- SCK (GPIO26) : Serial Clock.

4. Analogiques :

- ADC (Analog-to-Digital Converter) : L'ESP32 dispose de 18 canaux ADC (GPIO32 à GPIO39).
- DAC (Digital-to-Analog Converter) : L'ESP32 dispose de 2 canaux DAC (GPIO25 et GPIO26).

5. PWM (Pulse Width Modulation) :

- Tous les GPIO peuvent être utilisés pour générer des signaux PWM, utiles pour le contrôle de la luminosité des LEDs ou la commande de moteurs.

6. Touch Sensor:

- L'ESP32 inclut 10 broches pour capteurs tactiles capacitifs (GPIO0, GPIO2, GPIO4, GPIO12, GPIO13, GPIO14, GPIO15, GPIO27, GPIO32, GPIO33).

7. Capteur Hall:

- L'ESP32 dispose d'un capteur de Hall intégré utilisé pour détecter les champs magnétiques.

8. Autres fonctionnalités :

- RTC (Real-Time Clock) : Les GPIO32 à GPIO39 peuvent être utilisés pour des fonctions de basculement RTC.
- PI Flash : La mémoire flash peut être connectée via les broches SPI.
- JTAG: Debugging via les broches GPIO12 à GPIO15.
- SDIO : Interface pour cartes SD via GPIO2, GPIO4, GPIO12, GPIO13, GPIO14, GPIO15.

Utilisation et Restrictions:

1. Pins spécifiques à l'alimentation et au boot:

- GPIO0: Utilisé pour le boot mode.
- GPIO2 : Nécessite une résistance pull-down pendant le boot.
- GPIO12 (MTDI) : Doit être maintenu bas pendant le boot.
- GPIO15 (MTDO): Pull-down interne requis pendant le boot.
- GPIO36 (VP) et GPIO39 (VN) : Utilisés uniquement comme entrées analogiques.

2. Pins réservés :

- GPIO6 à GPIO11 : Réservés pour la connexion à la mémoire flash SPI interne.

En résumé, l'ESP32 offre une large gamme de pins configurables et polyvalents, permettant une multitude d'applications, notamment dans les domaines de la communication série, des interfaces analogiques et numériques, et des capteurs tactiles. Les développeurs doivent cependant tenir compte des fonctions spéciales et des restrictions associées à certaines broches lors de la conception de leurs circuits.

- Configuration de la connexion Wi-Fi sur l'ESP32

La configuration du Wi-Fi sur l'ESP32 est une étape essentielle pour de nombreux projets IoT. Utilisant l'environnement de développement Arduino IDE, cette tâche implique l'installation du support ESP32, la configuration des informations réseau (SSID et mot de passe),

```
const char* ssid = "espWIFI0";
const char* password = "01234567890";
```

et le téléversement d'un code de connexion Wi-Fi sur le microcontrôleur. Le code initialise la connexion, puis tente de se connecter au réseau spécifié. Une fois connecté, l'adresse IP assignée est affichée sur le moniteur série, confirmant la réussite de la connexion. Cette configuration permet à l'ESP32 de communiquer avec des serveurs web, d'envoyer des données de capteurs, et de recevoir des commandes à distance, ouvrant ainsi la voie à des applications IoT avancées.

- Bibliothèques nécessaires et exemples de code

Pour configurer la connexion Wi-Fi sur un ESP32, la bibliothèque principale utilisée est `WiFi.h`,

```
#include <WiFi.h>
```

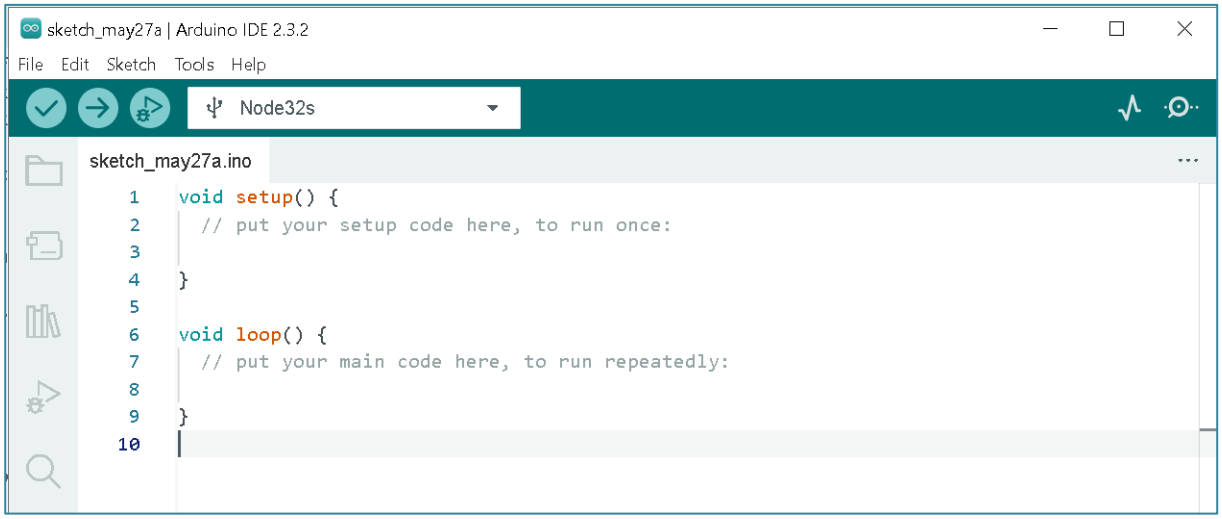
incluse dans le framework ESP32 pour l'IDE Arduino. Cette bibliothèque fournit les fonctions nécessaires pour se connecter à un réseau Wi-Fi, vérifier l'état de la connexion et obtenir l'adresse IP de l'ESP32. Pour commencer, en installant le support ESP32 dans l'IDE Arduino en ajoutant l'URL du gestionnaire de cartes supplémentaire et en sélectionnant le modèle ESP32 approprié. Ensuite, en utilisant la bibliothèque `WiFi.h`, initialiser la connexion Wi-Fi en spécifiant le SSID et le mot de passe de votre réseau. Une fois connecté, l'ESP32 affichera l'adresse IP attribuée, confirmant la réussite de la connexion. Cette configuration permet à l'ESP32 de communiquer avec des serveurs web, d'envoyer des données et de recevoir des commandes, ce qui est essentiel pour de nombreux projets IoT.

```
void setup_wifi() {
  delay(10);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}
```

2. Arduino IDE

- Introduction à l'IDE

L'IDE Arduino est un environnement de développement open-source, convivial et basé sur C/C++, conçu pour programmer des microcontrôleurs Arduino et compatibles. Il offre un éditeur de code, un moniteur série et des outils de compilation, facilitant la création rapide de projets embarqués et IoT.



- Installation et configuration

Une fois l'IDE installé, l'étape suivante consiste à configurer le support pour les microcontrôleurs ESP32. Cela se fait en ajoutant une URL spécifique dans les paramètres de l'IDE, qui permettra d'accéder au gestionnaire de cartes supplémentaires. L'URL en question est fournie par Espressif Systems, le fabricant de l'ESP32. Une fois cette URL ajoutée, Il est possible d'installer le package ESP32 à partir du Gestionnaire de cartes. Une fois le package installé, il est alors nécessaire de sélectionner le type de carte ESP32 dans le menu déroulant des types de carte disponibles dans l'IDE. Enfin, il est crucial de sélectionner le bon port de communication, qui correspond à la connexion physique avec la carte ESP32. Une fois toutes ces étapes accomplies, l'IDE Arduino est prêt à être utilisé pour le développement d'applications scientifiques et techniques exploitant les capacités de l'ESP32.

- Bibliothèques nécessaires pour ESP32 et autres composants

Pour l'IDE Arduino, plusieurs bibliothèques sont nécessaires pour exploiter pleinement les fonctionnalités de l'ESP32 et d'autres composants. Voici une liste des bibliothèques indispensables pour le développement sur l'ESP32 et d'autres composants :

-Bibliothèques pour l'ESP32 :

1. WiFi.h :
- Permet la gestion des connexions Wi-Fi sur l'ESP32, offrant des fonctions pour se connecter à un réseau Wi-Fi, vérifier l'état de la connexion et obtenir l'adresse IP de l'appareil.

2. AsyncTCP.h :
- Fournit des fonctionnalités de communication TCP asynchrones, utiles pour des applications nécessitant une gestion efficace des connexions TCP.

3. ESPAsyncWebServer.h :
- Permet de créer des serveurs web asynchrones sur l'ESP32, facilitant ainsi le développement d'applications IoT basées sur des interfaces web.

4. SPIFFS.h:
- Offre un système de fichiers SPI Flash pour l'ESP32, permettant de stocker et de gérer des fichiers sur la mémoire flash intégrée.

-Bibliothèques pour d'autres composants :

1. Adafruit_Sensor.h :
- Bibliothèque de base pour les capteurs Adafruit, offrant une interface unifiée pour la lecture de données à partir de différents capteurs.

2. Adafruit_BME280.h :
- Permet d'interagir avec le capteur de pression, d'humidité et de température BME280 d'Adafruit, fournissant des fonctions pour lire les données des capteurs.

3. Adafruit_GFX.h et Adafruit_ILI9341.h :
- Ces bibliothèques permettent de contrôler les écrans TFT LCD ILI9341, offrant des fonctions pour l'affichage de texte, de formes géométriques et d'images sur l'écran.

4. PubSubClient.h :
- Bibliothèque MQTT pour Arduino, permettant à l'ESP32 de publier et de souscrire à des messages sur un serveur MQTT, souvent utilisée pour la communication IoT.

5. ArduinoJson.h :
- Permet de manipuler des données au format JSON sur Arduino, offrant des fonctions pour la création, la modification et l'analyse de données JSON.

3. MQTT

Présentation du protocole

Le protocole MQTT (Message Queuing Telemetry Transport) est un protocole de messagerie léger, conçu pour les environnements où la bande passante est limitée et les ressources sont restreintes. Il est particulièrement adapté aux applications IoT (Internet des objets), où il est souvent utilisé pour la communication entre des appareils connectés.

- Principes de base et fonctionnement de MQTT

1. Architecture Client-Serveur :
   - Le MQTT fonctionne sur une architecture client-serveur, où les appareils se connectent à un serveur central appelé "broker". Le broker gère la distribution des messages entre les clients.
   - Les clients peuvent être de deux types : éditeurs (publishers) et abonnés (subscribers). Les éditeurs envoient des messages au broker, tandis que les abonnés reçoivent les messages du broker.

2. Topics (Sujets) :
   - Les messages MQTT sont organisés en "topics" (sujets). Un topic est une chaîne hiérarchique qui identifie de manière unique les messages. Par exemple, un topic peut être "trottinette/vitesse".
   - Les abonnés s'inscrivent à des topics spécifiques pour recevoir les messages correspondants. Les éditeurs publient des messages sur ces topics.

3. QoS (Quality of Service) :
   - MQTT offre trois niveaux de qualité de service pour assurer la fiabilité de la transmission des messages :
     - QoS 0 : "At most once" - Le message est envoyé une seule fois sans garantie de livraison.
     - QoS 1 : "At least once" - Le message est livré au moins une fois, avec une confirmation de réception.
     - QoS 2 : "Exactly once" - Le message est assuré d'être livré exactement une fois, avec un protocole de confirmation en deux étapes.

4. Persistant et Stateless :
   - MQTT est conçu pour être persistant et sans état. Les clients peuvent se déconnecter et se reconnecter sans perdre les messages, grâce à la capacité du broker à stocker temporairement les messages pour les clients abonnés.

5. Faible consommation de bande passante :
   - Grâce à son faible encombrement et à son efficacité, MQTT est idéal pour les appareils avec des ressources limitées et des réseaux à faible bande passante. Les messages sont petits et le protocole minimise les échanges de contrôle.

6. Sécurité :

- MQTT prend en charge les communications sécurisées via TLS (Transport Layer Security), ce qui permet de chiffrer les messages échangés entre les clients et le broker.
- L'authentification des clients peut également être mise en œuvre pour garantir que seuls les appareils autorisés peuvent se connecter et échanger des messages.

- Applications de MQTT :

MQTT est largement utilisé dans diverses applications IoT, telles que :
- Maison intelligente : Contrôle et surveillance des appareils domestiques comme les thermostats, les éclairages et les systèmes de sécurité.
- Ville intelligente : Gestion de l'éclairage public, des systèmes de transport et des capteurs environnementaux.
- Santé connectée : Surveillance des patients à distance et gestion des dispositifs médicaux.
- Industrie 4.0 : Communication entre les machines et les systèmes de contrôle dans les environnements de fabrication.

Dans le cadre de notre projet, le protocole MQTT sera utilisé pour établir une communication bidirectionnelle entre l'application mobile et la trottinette électrique. Cela permettra de transmettre des commandes de vitesse depuis l'application vers la trottinette, et d'envoyer des données de retour de vitesse depuis la trottinette vers l'application, assurant ainsi un contrôle précis et une régulation efficace en temps réel.

HiveMQ Broker

- Documentation et configuration

HiveMQ est un broker MQTT populaire et robuste, conçu pour gérer des communications fiables et efficaces entre des dispositifs connectés. Il est particulièrement adapté aux environnements IoT, offrant une performance élevée, une évolutivité et des fonctionnalités avancées. Voici une documentation et une configuration détaillées de HiveMQ Broker.

1. Installation de HiveMQ :

Pour installer HiveMQ, vous pouvez suivre les étapes ci-dessous :

1. Téléchargement :
   - Accédez au site officiel de HiveMQ et téléchargez la version Community Edition ou Enterprise Edition selon vos besoins.
   - Lien de téléchargement HiveMQ

2. Installation :
   - Extrayez l'archive téléchargée dans le répertoire de votre choix.

- Ouvrez un terminal ou une invite de commande, naviguez vers le répertoire extrait, et exécutez le fichier bin/run.sh pour Linux/MacOS ou bin\run.bat pour Windows.

3. Configuration de base :
   - Les fichiers de configuration se trouvent dans le répertoire conf du dossier HiveMQ. Les principaux fichiers de configuration incluent :
     - hivemq.xml : Configuration générale du broker.
     - logback.xml : Configuration des logs.
   - Pour les configurations de base, vous pouvez généralement laisser les paramètres par défaut. Cependant, vous pouvez modifier les paramètres de port ou d'authentification si nécessaire.

4. Démarrage du broker :
   - Une fois configuré, démarrez le broker HiveMQ en exécutant le script run mentionné précédemment.
   - Vous pouvez vérifier le fonctionnement du broker en accédant à http://localhost:8080 dans votre navigateur, où une interface de gestion HiveMQ est disponible (si activée).

2. Configuration de la connexion entre l'ESP32 et HiveMQ Broker :

Pour configurer la communication entre l'ESP32 et le broker HiveMQ, suivez ces étapes :

1. Configurer l'ESP32 pour la connexion Wi-Fi :
   - Utilisez l'IDE Arduino pour programmer l'ESP32.
   - Installez les bibliothèques nécessaires (WiFi et PubSubClient) via le gestionnaire de bibliothèques de l'IDE Arduino.

```
#include <PubSubClient.h>
```

2. Configurer le code de l'ESP32 pour MQTT :

Voici un exemple de code pour connecter l'ESP32 au broker HiveMQ et publier/s'abonner à des topics :

“ Definition du serveur MQTT :

```
const char* mqtt_server = "broker.hivemq.com";
```

```
WiFiClient espClient;
PubSubClient client(espClient);
```

“ Definition des topics :

```
const char* speed_topic = "scooter/speed5";
const char* feedback_topic = "scooter/feedback5";
const char* kp_topic = "scooter/kp5";
const char* ki_topic = "scooter/ki5";
```

“ Fonction "reconnect" pour vérifier si on est toujours connectés au broker et abonnés aux topics désignés.

```
void reconnect() {
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");

    if (client.connect("wa79rrbID00")) {
      Serial.println("connected");
      client.subscribe(speed_topic);
      client.subscribe(kp_topic);
      client.subscribe(ki_topic);
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      delay(5000);
    }
  }
}
```

“ Fonction "callback" pour recevoir la valeur de vitesse (char) et la convertir en (int)

```
void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");
  int speedValue = atoi((char*)payload);
  if (speedValue > Vm){
    Serial.println("erreur MQTT");
  }
  else {
    Serial.println(speedValue);
  }
```

3. Test de communication :

1. Publication de messages :
   - Utilisez l'exemple de code ci-dessus pour publier des messages sur le topic scooter/speed5.
   - Assurez-vous que l'ESP32 est connecté au Wi-Fi et au broker HiveMQ.

2. Abonnement aux messages :
   - Configurez l'ESP32 pour s'abonner au topic scooter/speed5.
   - Lorsque des messages sont publiés sur ce topic, la fonction de rappel (callback) est appelée pour traiter les messages.

4. Outils de test :

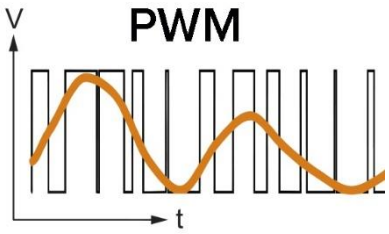
- Vous pouvez utiliser des outils comme MQTT.fx ou MQTT Explorer pour tester la publication et l'abonnement aux messages sur le broker HiveMQ.
- Ces outils vous permettent de vous connecter au broker, de publier des messages et de voir les messages reçus en temps réel.

En suivant cette documentation et configuration détaillées, vous serez en mesure de configurer un broker HiveMQ et d'établir une communication efficace entre votre application mobile et l'ESP32, permettant ainsi le contrôle à distance de la vitesse de votre trottinette électrique.

4. PWM (Pulse Width Modulation)

- Principe de Fonctionnement

La modulation de largeur d'impulsion (PWM) est une technique utilisée pour contrôler la puissance fournie à des charges électriques, en particulier des moteurs. Le principe de la PWM repose sur la variation de la largeur des impulsions d'un signal périodique. En changeant la durée pendant laquelle le signal est "haut" (ON) ou "bas" (OFF) dans un cycle donné, on peut moduler la quantité de puissance fournie à une charge. Cela permet un contrôle précis de dispositifs comme les moteurs, les LED, les chauffages, etc.



- Pourquoi Utiliser la PWM ?

1. Contrôle Précis de la Vitesse :
   - Variation de la Vitesse : La PWM permet de moduler la largeur des impulsions du signal de commande, ce qui ajuste efficacement la vitesse du moteur en contrôlant la puissance moyenne délivrée.
   - Résolution Fine : En ajustant le rapport cyclique (duty cycle), la vitesse du moteur peut être contrôlée de manière très précise, permettant des ajustements fins et progressifs.

2. Efficacité Énergétique :
   - Moins de Dissipation de Chaleur : Contrairement aux méthodes de contrôle de la vitesse par résistance variable, la PWM génère moins de chaleur, car elle alterne entre les états ON et OFF sans transitions graduelles.
   - Conservation de la Puissance : En fournissant une puissance moyenne adéquate tout en minimisant les pertes énergétiques, la PWM permet une utilisation plus efficiente de l'énergie de la batterie.

3. Compatibilité :

- Intégration Facile : Les microcontrôleurs comme l'ESP32 sont conçus pour générer des signaux PWM, ce qui facilite leur intégration dans des projets de contrôle de moteurs.

- Explications sur le PWM et son Utilisation pour Contrôler la Vitesse du Moteur

Dans le contexte du contrôle de la vitesse d'un moteur, le PWM est particulièrement efficace. Le signal PWM est caractérisé par deux paramètres principaux :

- La fréquence : le nombre de cycles par seconde, mesuré en Hertz (Hz).
- Le rapport cyclique (Duty Cycle) : le pourcentage de temps où le signal est à l'état haut pendant un cycle. Un rapport cyclique de 50 % signifie que le signal est haut pendant 50 % du temps et bas pendant les 50 % restants.

En ajustant le rapport cyclique du signal PWM, on peut contrôler la vitesse du moteur. Un rapport cyclique élevé fournit plus de puissance au moteur, augmentant ainsi sa vitesse, tandis qu'un rapport cyclique faible réduit la puissance et la vitesse du moteur.

- Mise en Œuvre avec l'ESP32

L'ESP32 est un microcontrôleur puissant et polyvalent qui dispose de plusieurs canaux PWM intégrés, permettant un contrôle précis des périphériques connectés. Pour contrôler la vitesse du moteur de la trottinette électrique, l'ESP32 génère un signal PWM sur une broche de sortie connectée au driver de moteur, tel que le L298P Motor Driver Shield.

- Exemples de Code pour Générer des Signaux PWM

Voici un exemple de code pour générer un signal PWM avec l'ESP32 et contrôler la vitesse du moteur :

```
// Map speed value to PWM value
int pwmValue = map(speedValue, 0, Vm, 0, 255);

// Control the motor using PWM
analogWrite(motor_pin, pwmValue);
```

- Configuration et Utilisation du PWM : Dans setup(), le PWM est configuré sur le canal 0 avec une fréquence de 5 kHz et une résolution de 8 bits. La broche du moteur (motor_pin) est attachée à ce canal.

Exemple D'Application

Dans cet exemple, nous avons une valeur de vitesse maximale (Vm) de 100, et nous recevons une commande de vitesse de 50 via le message MQTT. Nous voulons mapper cette commande de vitesse à une valeur PWM entre 0 et 255 pour contrôler la vitesse du moteur.
En utilisant la fonction map :

```
int speedValue = 50; // Commande de vitesse reçue
int Vm = 100; // Valeur de vitesse maximale/ Mapper la valeur de vitesse à la valeur PWM
int pwmValue = map(speedValue, 0, Vm, 0, 255);// Aicr la valeur PWM mappée
Serial.println(pwmValue);
```

La fonction map va mettre à l'échelle la valeur de speedValue de la plage de 0 à Vm (0 à 100 dans ce cas) à la plage de 0 à 255. Voici le calcul :

- speedValue = 50
- Vm = 100
- pwmValue = map(50, 0, 100, 0, 255)
- = (50 - 0) * (255 - 0) / (100 - 0) + 0
- = 127.5
- ≈ 128

Ainsi, la valeur PWM mappée pour une commande de vitesse de 50 serait d'environ 128. Cette valeur PWM serait alors appliquée au moteur pour contrôler sa vitesse en conséquence.

5. L298P Motor Driver Shield

- Présentation du Composant

Le L298P Motor Driver Shield est un module couramment utilisé pour contrôler des moteurs à courant continu (DC) et des moteurs pas à pas. Basé sur le double pont en H L298, il permet de contrôler deux moteurs DC de manière indépendante. Chaque moteur peut être alimenté jusqu'à 2A en continu, avec un courant de crête de 3A par canal. Le L298P peut gérer des tensions d'alimentation allant jusqu'à 46V, ce qui le rend idéal pour une large gamme d'applications de contrôle de moteurs.

Pour contrôler une trottinette électrique via une application mobile en utilisant MQTT, le choix de la PWM (Pulse Width Modulation) et du L298P Motor Driver Shield est judicieux pour plusieurs raisons techniques et pratiques :

- Pourquoi Utiliser le L298P Motor Driver Shield ?

1. **Capacité de Contrôle de Moteurs DC :**
   - **Contrôle des Moteurs à Courant Continu :** Le L298P est spécialement conçu pour contrôler des moteurs DC, qui sont couramment utilisés dans les trottinettes électriques.
   - **Support pour Deux Moteurs :** Il permet de contrôler deux moteurs de manière indépendante, utile si la trottinette dispose de moteurs pour chaque roue.

2. **Gestion des Courants Élevés :**
   - **Courant Maximale :** Le L298P peut gérer des courants jusqu'à 2A en continu et 3A en pointe, ce qui est suffisant pour les moteurs de nombreuses trottinettes électriques.
   - **Protection Intégrée :** Il offre une protection contre les surcharges et les courts-circuits, augmentant la fiabilité du système.

3. **Simplicité d'Utilisation :**
   - **Interface Simple :** Le shield L298P s'interface facilement avec les microcontrôleurs comme l'ESP32 via des broches numériques pour les commandes IN1, IN2, ENA, etc.
   - **Compatibilité avec Arduino :** Comme le L298P est conçu pour être compatible avec les cartes Arduino, son utilisation avec l'ESP32, qui partage des similitudes avec l'architecture Arduino, est simplifiée.

- Description et Schéma de Câblage

Le L298P Motor Driver Shield est conçu pour être monté directement sur des cartes de développement Arduino ou compatibles. Voici un schéma de câblage typique pour l'utiliser avec un ESP32 :

- **Broches d'alimentation :**
  - **VMS** : Entrée de puissance pour les moteurs (généralement de 5V à 12V).
  - **GND** : Masse commune.
  - **5V** : Alimentation de la logique du shield (souvent alimentée par le régulateur 5V de l'ESP32 ou de l'Arduino).

- **Broches de commande pour les moteurs :**
  - **ENA/ENB** : Broches d'activation pour les moteurs A et B respectivement. Ces broches peuvent être utilisées pour contrôler la vitesse des moteurs via PWM.
  - **IN1/IN2** : Broches de commande pour le moteur A.

- **IN3/IN4** : Broches de commande pour le moteur B.

Voici un schéma de câblage simplifié pour connecter le L298P à l'ESP32 :

	Speed Pins	Speed Control	Direction Pins	Direction Control	
Motor A	D10	PWM 0-100	D12	HIGH = Forward	LOW = Reverse
Motor B	D11	PWM 0-100	D13	HIGH = Forward	LOW = Reverse

-Utilisation avec l'ESP32

Pour utiliser le L298P avec l'ESP32, nous devons configurer les broches GPIO pour générer des signaux PWM afin de contrôler la vitesse et la direction du moteur. L'ESP32 dispose de plusieurs canaux PWM intégrés, ce qui le rend bien adapté pour cette tâche.

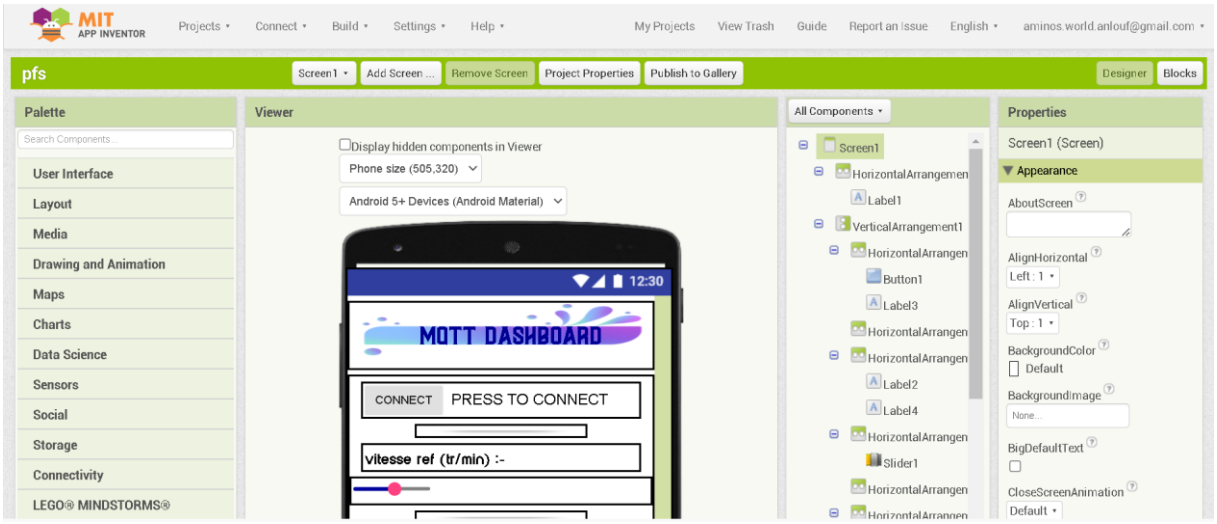
6. Application mobile

- Pourquoi MIT App Inventor ?

MIT App Inventor est un environnement de programmation visuel intuitif qui permet à tout le monde, même aux enfants, de créer des applications entièrement fonctionnelles pour les téléphones Android, les iPhones et les tablettes Android/iOS. Les nouveaux utilisateurs de MIT App Inventor peuvent créer une première application simple et opérationnelle en moins de 30 minutes. De plus, notre outil basé sur des blocs facilite la création d'applications complexes et à fort impact en beaucoup moins de temps que les environnements de programmation traditionnels. Le projet MIT App Inventor cherche à démocratiser le développement de logiciels en permettant à tous, en particulier aux jeunes, de passer de la consommation technologique à la création technologique.

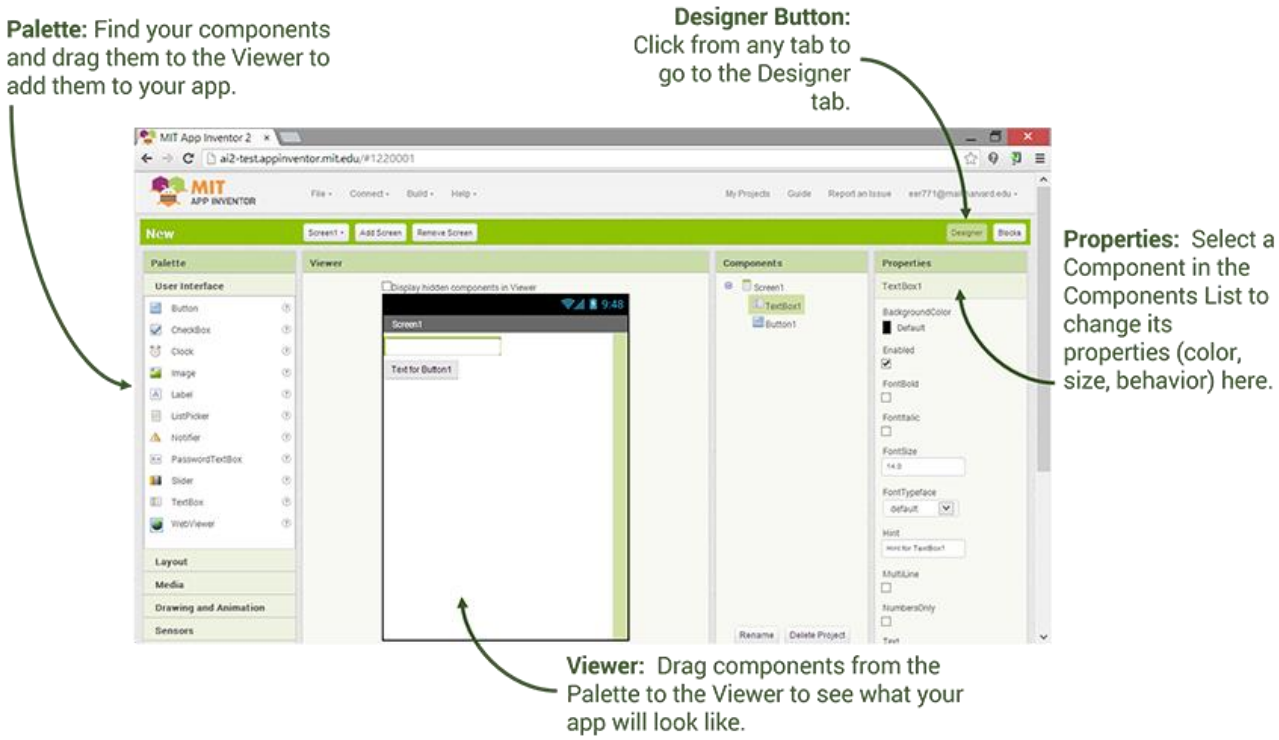
Une petite équipe composée d'employés et d'étudiants du MIT CSAIL, dirigée par le professeur Hal Abelson, forme le noyau d'un mouvement international d'inventeurs. En plus de diriger la sensibilisation autour de MIT App Inventor et de mener des recherches sur ses impacts, cette équipe principale gère l'environnement de développement d'applications en ligne gratuit qui dessert plus de 6 millions d'utilisateurs enregistrés.

Les programmes de codage basés sur des blocs inspirent l'autonomisation intellectuelle et créative. MIT App Inventor va plus loin en offrant aux enfants une véritable autonomie pour faire la différence – un moyen d'obtenir un impact social d'une valeur incommensurable pour leurs communautés. En fait, les inventeurs d'applications à l'école et en dehors des contextes éducatifs traditionnels se sont réunis et ont fait exactement cela :

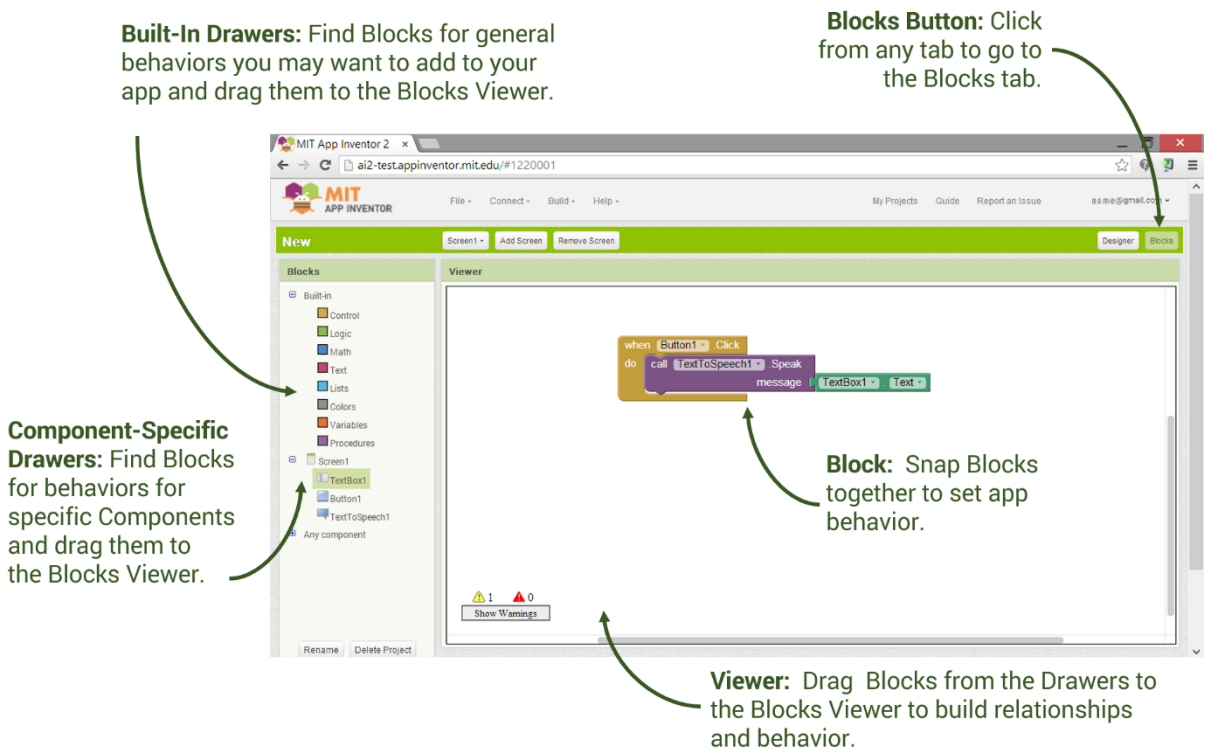


MIT App Inventor se compose du concepteur et de l'éditeur de blocs.

Le Designer vous permet de créer l'interface de l'application :



L'éditeur de blocs vous permet de programmer le comportement de l'application en assemblant des blocs :

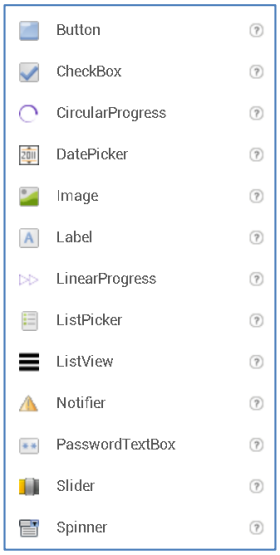


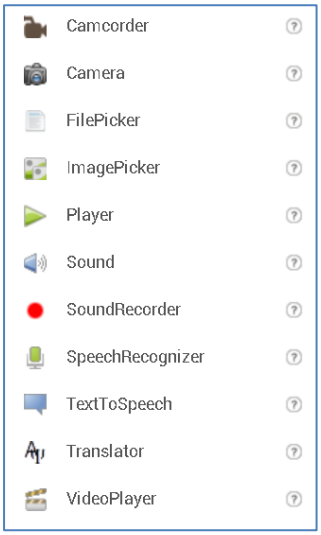
- Pourquoi MIT App Inventor ?

Nous avons choisi d'utiliser MIT App Inventor pour son compatibilité avec les projets iot et les protocoles de communication Wireless comme MQTT

*Dans ce projet on a utilisé la bibliothèque UrsPahoMqttClient

MIT App Inventor propose une variété d'outils près pour l'utilisation par un simple Drag and Drop qu'on utilise pour créer les différentes parties de notre interface graphique :





Et une interface de programmation par blocs qui propose des différents blocs intégrés disponibles quels que soient les composants de votre projet. En plus de ces blocs de langage, chaque composant de votre projet possède son propre ensemble de blocs spécifiques à ses propres événements, méthodes et propriétés. Ceci est un aperçu de tous les blocs intégrés disponibles dans l'éditeur de blocs.

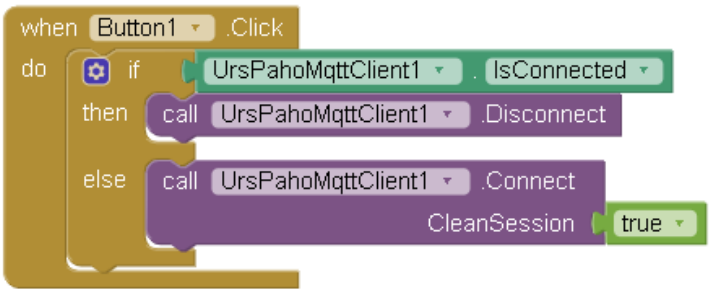
- Blocs de contrôle
- Blocs logiques
- Blocs mathématiques
- Blocs de texte
- Liste les blocs
- Blocs de dictionnaires
- Blocs de couleurs
- Blocs de variables
- Blocs de procédures

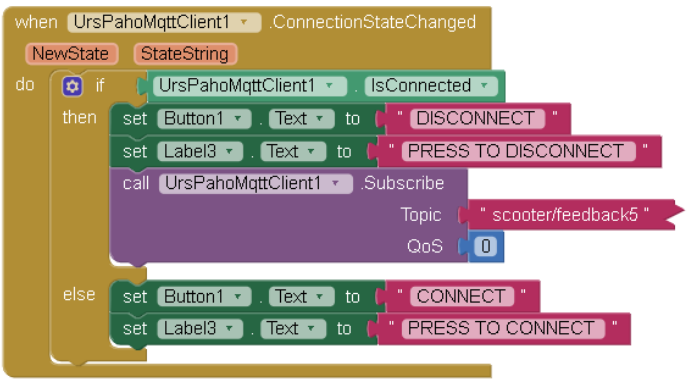
- Interface utilisateur et fonctionnalités

Component 1 : bouton (connect) pour se connecter au serveur MQTT

-Design :

-Programme :





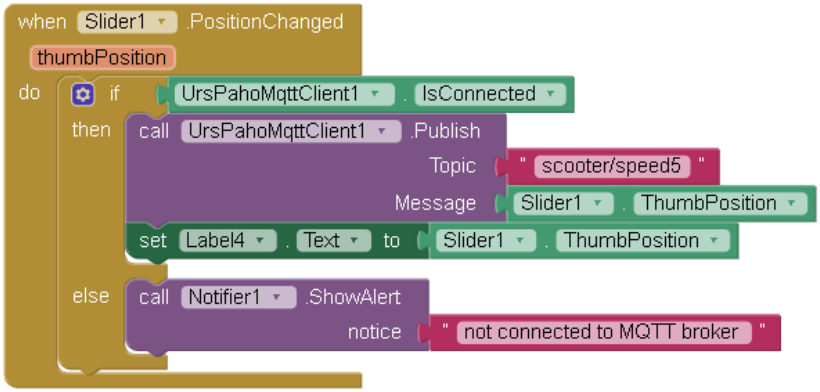
“ Quand l'état de connexion MQTT change, le programme change l'affichage du bouton de "disconnect" vers "connect" ou vice-versa.

“ Quand on clique sur le bouton "connect “ le programme vérifie qu'on est connecté au broker MQTT, si oui on se déconnecte, sinon on se connecte.

Component 2 : "Slider" pour envoyer la valeur de vitesse de référence (entre 0 et 296 tr/min)

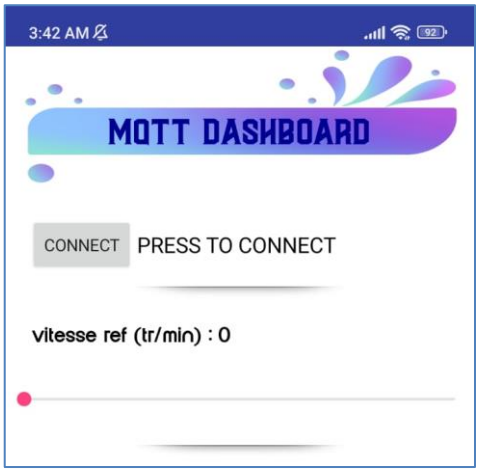
-Design :

-Programme :



Le changement de position du Slider déclenche l'envoi de vitesse par MQTT vers l'ESP32 si connecté, sinon l'application affiche une notification d'erreur de connexion MQTT.

Totalité de l'interface de l'application :



- **Component 1 :** bouton (connect) pour se connecter au serveur MQTT

- **Component 2 :** "Slider" pour envoyer la valeur de vitesse de référence (entre 0 et 296 tr/min)

Chapitre 2 : Vitesse de retour et régulation

1.Boucle fermée et étalonnage

- boucle fermée :

➢ Introduction à la boucle fermée

La boucle fermée, ou contrôle en boucle fermée, est un système de contrôle automatique où la sortie du système est continuellement mesurée et comparée à une valeur de référence souhaitée. Cette comparaison génère un signal d'erreur qui est utilisé pour ajuster les actions du système afin de minimiser cette erreur. Dans le contexte de la régulation de la vitesse d'une trottinette électrique, une boucle fermée permet de maintenir une vitesse constante même face à des perturbations externes, telles que des variations de charge ou de terrain.

➢ Importance de la boucle fermée dans la régulation de la vitesse

L'utilisation d'une boucle fermée dans la régulation de la vitesse d'une trottinette électrique présente plusieurs avantages :

- Stabilité accrue : La boucle fermée corrige constamment les écarts par rapport à la vitesse désirée, assurant une conduite plus stable.
- Précision : En mesurant la vitesse réelle et en l'ajustant par rapport à la vitesse de consigne, le système peut atteindre une précision de contrôle élevée.
- Réactivité : Les systèmes en boucle fermée réagissent rapidement aux changements de conditions, garantissant une réponse dynamique efficace.
- Robustesse : Ils sont capables de compenser les perturbations imprévues, telles que les variations de la pente ou du poids de l'utilisateur.

➢ Mise en œuvre de la boucle fermée pour la trottinette électrique

Pour votre projet, la mise en œuvre d'un système de contrôle en boucle fermée pour la régulation de la vitesse comprend plusieurs étapes :

- Capteurs de vitesse

Les capteurs de vitesse jouent un rôle crucial dans la boucle fermée. Ils mesurent la vitesse de rotation des roues de la trottinette et transmettent ces informations à l'unité de contrôle (ESP32). Les capteurs couramment utilisés sont des capteurs de hall ou des encodeurs optiques.

- Lecture et traitement des données de vitesse

Les données de vitesse mesurées par les capteurs sont lues par l'ESP32, qui les traite pour déterminer la vitesse réelle de la trottinette. Cette information est ensuite comparée à la vitesse de consigne définie par l'utilisateur via l'application mobile.

- Génération du signal d'erreur

L'ESP32 calcule le signal d'erreur en soustrayant la vitesse réelle de la vitesse de consigne. Ce signal d'erreur est ensuite utilisé pour ajuster la commande envoyée au moteur via le module de commande L298P.

- Algorithme de contrôle PI

L'algorithme de contrôle proportionnel-intégral (PI) est utilisé pour ajuster la commande du moteur en fonction du signal d'erreur. Cet algorithme combine une action proportionnelle (P), qui réagit instantanément à l'erreur actuelle, et une action intégrale (I), qui corrige les erreurs

accumulées au fil du temps. L'algorithme PI est configuré dans l'ESP32 pour générer un signal PWM (Pulse Width Modulation) adapté aux besoins de régulation de vitesse.

- Envoi de la commande au moteur

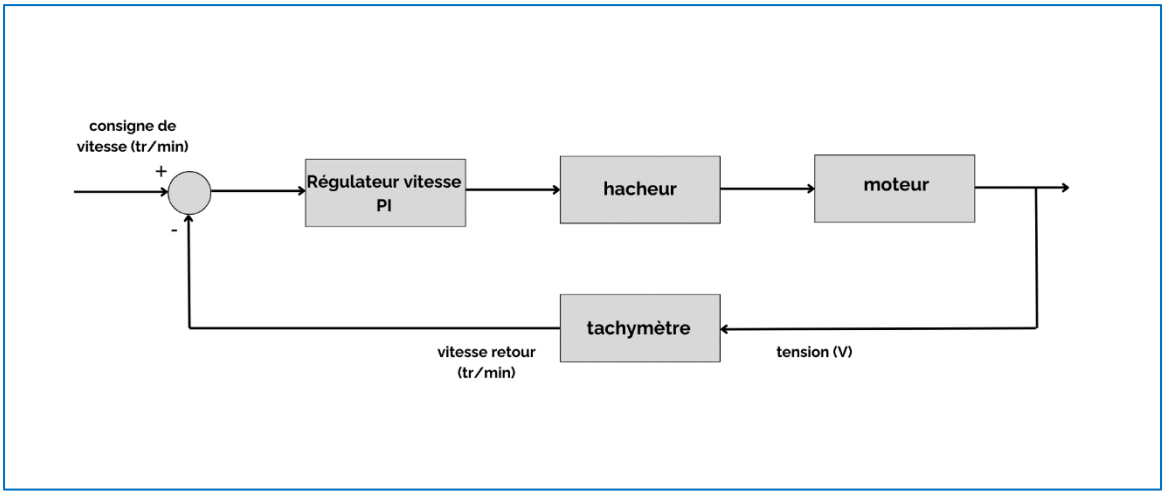
Le signal PWM généré par l'algorithme PI est envoyé au driver de moteur L298P, qui ajuste la puissance fournie au moteur de la trottinette en conséquence. Cela permet de maintenir la vitesse désirée même en présence de variations de charge ou de conditions de conduite.

- Retour d'information et mise à jour

Le système boucle fermée fonctionne en continu, avec des mises à jour régulières de la vitesse réelle et de la commande du moteur. L'application mobile permet à l'utilisateur de surveiller en temps réel la vitesse de la trottinette et d'ajuster les paramètres de consigne si nécessaire.

➢ Schéma de la boucle fermée

Voici un schéma illustrant la boucle fermée de régulation de la vitesse de la trottinette électrique :



1. Capteur de vitesse : Mesure la vitesse réelle de la trottinette.
2. ESP32 : Lit les données de vitesse, calcule le signal d'erreur, exécute l'algorithme PI, et génère le signal PWM.
3. Driver de moteur L298P : Reçoit le signal PWM et ajuste la puissance du moteur.
4. Moteur : Ajuste sa vitesse en fonction de la commande reçue.
5. Application mobile : Permet à l'utilisateur de définir la vitesse de consigne et de surveiller la vitesse réelle.

➢ Conclusion

La mise en œuvre d'une boucle fermée pour la régulation de la vitesse de la trottinette électrique assure une conduite stable, précise et réactive. En utilisant des capteurs de vitesse, un algorithme de contrôle PI, et une communication efficace avec une application mobile, ce système de contrôle avancé offre une expérience utilisateur améliorée et sécurisée.

- Procédures d'étalonnage

L'étalonnage est une étape cruciale pour assurer la précision des mesures de vitesse et la performance globale du système de régulation de la trottinette électrique. Cette section décrit

les procédures d'étalonnage, en commençant par la préparation des instruments, la collecte des données, l'analyse des résultats, et enfin, l'ajustement des paramètres du système pour garantir une précision optimale.

➢ Préparation des instruments

Avant de commencer l'étalonnage, il est essentiel de s'assurer que tous les instruments et composants du système sont correctement installés et fonctionnels. Voici les étapes de préparation :

- Vérification des connexions :
    - ♦ Assurez-vous que le capteur de vitesse est correctement installé et connecté à l'ESP32.
    - ♦ Vérifiez les connexions entre l'ESP32 et le driver de moteur L298P.
    - ♦ Assurez-vous que le moteur est correctement câblé et qu'il fonctionne sans obstruction.
- Initialisation du système :
    - ♦ Allumez le système et connectez l'ESP32 au réseau Wi-Fi.
    - ♦ Assurez-vous que l'application mobile est installée et fonctionne correctement pour envoyer et recevoir des données de vitesse.

➢ Collecte des données d'étalonnage

L'étalonnage implique de recueillir des données sur la vitesse de rotation du moteur à différentes tensions d'entrée. Voici les étapes pour collecter ces données :

- Définir les points de mesure :
    - ♦ Choisissez une série de tensions d'entrée pour le moteur, par exemple, 0.5V, 1V, 1.5V, 2V, 2.5V, 3V, 3.5V, 4V, 4.5V, et 5V.
- Enregistrer les vitesses correspondantes :
    - ♦ Pour chaque tension appliquée au moteur, utilisez le capteur de vitesse pour mesurer la vitesse de rotation (en RPM).
    - ♦ Enregistrez les données de tension et de vitesse mesurées dans un tableau pour analyse ultérieure.

Voici un exemple de tableau d'étalonnage basé sur les données fournies :

VOLTAGE	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5
VITESSE RPM	29.1	64.8	88.6	120.4	150.9	176.4	209.5	238	265.5	296.4

➢ Analyse des résultats

Après avoir collecté les données, l'étape suivante consiste à analyser les résultats pour déterminer la relation entre la tension appliquée et la vitesse du moteur. Cette relation est généralement linéaire et peut être exprimée sous la forme d'une fonction mathématique.

- Tracer les données :
    - ♦ Utilisez un logiciel de traitement de données (comme Excel ou MATLAB) pour tracer les points de données (tension vs. vitesse).

- Déterminer la fonction de calibration :
    - Ajustez une ligne droite aux points de données pour déterminer la fonction de calibration. La forme générale de cette fonction est :

$$\boldsymbol{Vitesse = k \times Tension + b}$$

    - À partir des données fournies, la fonction de calibration est :

$$\boldsymbol{Vitesse(RPM) = 59 \times Tension(V) + 1.55455}$$

➢ Ajustement des paramètres du système

Enfin, utilisez la fonction de calibration pour ajuster les paramètres du système de régulation de la trottinette électrique. Cela garantit que les mesures de vitesse sont précises et que le système réagit correctement aux commandes de vitesse.

- Implémentation de la fonction de calibration :
    - Intégrez la fonction de calibration dans le code de l'ESP32 pour convertir les mesures de tension en vitesses précises.
- Vérification et validation :
    - Testez le système avec différentes tensions et comparez les vitesses mesurées avec les vitesses attendues selon la fonction de calibration.
    - Faites les ajustements nécessaires pour affiner la précision du système.

➢ Conclusion

Les procédures d'étalonnage sont essentielles pour assurer que le système de régulation de la vitesse de la trottinette électrique fonctionne de manière optimale. En suivant ces étapes, vous pouvez garantir des mesures précises et une performance fiable, améliorant ainsi l'expérience utilisateur et la sécurité de l'utilisateur.

2. Code retour

- Lecture des données de vitesse

La lecture des données de vitesse est une étape fondamentale dans le contrôle en boucle fermée du système de régulation de la vitesse de la trottinette électrique. Cela implique la mesure continue de la vitesse actuelle de la trottinette, la conversion de cette mesure en un format utilisable par le système de contrôle, et l'envoi des données de vitesse à l'application mobile pour le retour d'information en temps réel. Voici comment cela est implémenté dans le projet :

➢ Capteur de vitesse

Le capteur de vitesse utilisé dans ce projet mesure la vitesse de rotation de la roue de la trottinette et envoie cette information sous forme de tension au microcontrôleur ESP32. Le capteur est connecté à la broche analogique de l'ESP32 (broche 33 dans ce cas).

➢ Fonction de lecture de la vitesse

La fonction lireVitesse() est responsable de la lecture de la tension du capteur, de la conversion de cette tension en une valeur de vitesse en RPM (révolutions par minute), et du retour de cette valeur pour une utilisation ultérieure dans le système de contrôle.

```
float lireVitesse() {
  // Lire la tension du capteur et convertir en vitesse
  float tension_capteur = analogRead(capteur_pin) * (5.0 / 4095.0);
  float vitesseRetour = 0;
  if (tension_capteur < 0.1){
    vitesseRetour = 0;
  }
  else {
    vitesseRetour = 59 * tension_capteur + 1.55455;
  }
  return vitesseRetour;
}
```

➢ Intervalle de lecture

Pour éviter des lectures continues et inutiles, le code utilise un intervalle de temps défini (interval = 0.1 millisecondes) entre deux lectures de la vitesse. Cette temporisation est gérée par la variable previousMillis.

```
unsigned long previousMillis = 0;
const long interval = 0.1;
```

➢ Intégration dans la boucle principale

Dans la boucle principale loop(), le code vérifie si l'intervalle de temps spécifié s'est écoulé avant d'effectuer une nouvelle lecture de la vitesse. La vitesse mesurée est ensuite publiée sur le topic MQTT feedback_topic pour être envoyée à l'application mobile. Voici la section du code correspondant :

```
void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  unsigned long currentMillis = millis(); // Temps actuel
  if (currentMillis - previousMillis >= interval) {
    previousMillis = currentMillis; // Sauvegarder le temps actuel

    // Lire la vitesse
    float vitesseRetour = lireVitesse();

    // Envoyer la vitesse à MQTT
    client.publish(feedback_topic, String(vitesseRetour).c_str()); // Publier la vitesse sur MQTT

    // Afficher la vitesse sur le moniteur série
    Serial.print("Vitesse actuelle: ");
    Serial.println(vitesseRetour);

    float erreur = speedValue - vitesseRetour;
    integral += erreur * (interval / 1000.0); // Accumuler l'erreur

    int pwmValue = map(speedValue, 0, Vm, 0, 255);

    int commandePWM = pwmValue + (Kp * erreur + Ki * integral);
    commandePWM = constrain(commandePWM, 0, 255);
    analogWrite(motor_pin, commandePWM);
  }
}
```

3. Retour application mobile

- Affichage des données

Le retour d'information en temps réel est un élément clé pour une interface utilisateur conviviale et efficace. Dans le contexte de la régulation de la vitesse de la trottinette électrique, cela implique l'affichage des données de vitesse mesurées sur l'application mobile, permettant à l'utilisateur de surveiller la vitesse actuelle en temps réel. Cette section théorique décrit les concepts essentiels du retour d'information, les avantages pour l'utilisateur, et comment intégrer cette fonctionnalité dans votre application mobile développée avec MIT App Inventor.

➢ Concepts de retour d'information

Le retour d'information, ou feedback, est le processus par lequel le système fournit des informations en temps réel à l'utilisateur sur son état ou ses performances. Dans le cas de la trottinette électrique, le retour d'information inclut la vitesse actuelle de la trottinette, qui est mesurée par le capteur de vitesse et transmise à l'application mobile.

- Avantages du retour d'information en temps réel :
    - ♦ Meilleure surveillance : L'utilisateur peut surveiller en temps réel la vitesse de la trottinette, permettant un contrôle plus précis et une utilisation plus sécurisée.
    - ♦ Réactivité : Le retour d'information rapide permet à l'utilisateur de réagir immédiatement aux changements de vitesse, améliorant ainsi l'expérience utilisateur.
    - ♦ Confiance accrue : Un système qui fournit des informations en temps réel inspire confiance et permet à l'utilisateur de se sentir plus en contrôle de la trottinette.

➢ Intégration du retour d'information dans MIT App Inventor

Pour intégrer le retour d'information de la vitesse mesurée dans votre application mobile développée avec MIT App Inventor, vous devez suivre ces étapes :

- Création de l'interface graphique :

L'interface doit inclure les éléments suivants :

- ♦ Un label pour afficher la vitesse actuelle de la trottinette.
- ♦ Un composant MQTT pour recevoir les données de vitesse de l'ESP32.

Voici les composants à ajouter dans MIT App Inventor :

- ♦ Label : Pour afficher la vitesse actuelle. Nommez ce label lblVitesseActuelle.
- ♦ MQTT Client : Pour recevoir les données de vitesse de l'ESP32. Configurez ce composant pour se connecter au broker MQTT et s'abonner au topic scooter/feedback5.

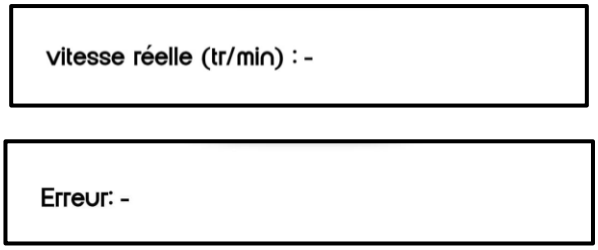
- Configuration de l'application :

Utilisez les blocs de programmation de MIT App Inventor pour configurer le comportement de l'application. Vous devrez ajouter les blocs suivants dans l'éditeur de blocs :

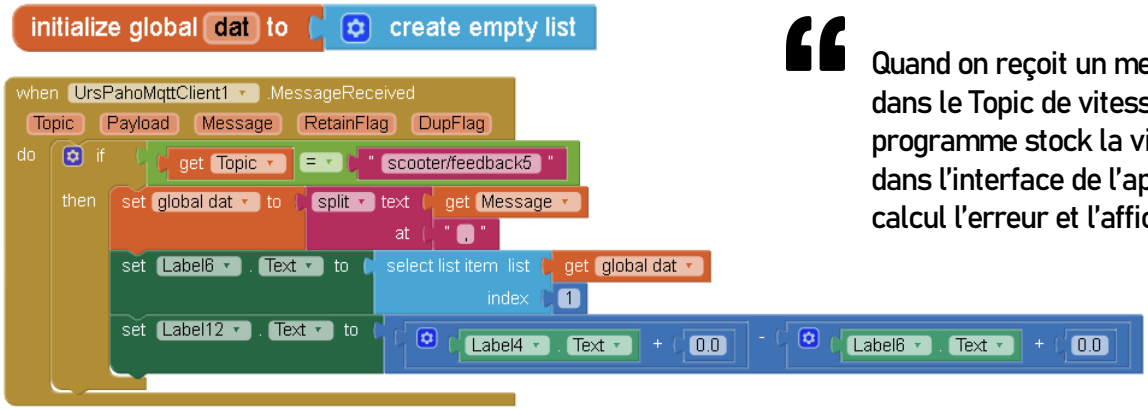
- Initialisation de l'application : Connectez-vous au broker MQTT lors du démarrage de l'application et abonnez-vous au topic scooter/feedback5.
- Réception des données de vitesse : Lorsque des données sont reçues sur le topic scooter/feedback5, mettez à jour le label lblVitesseActuelle avec la vitesse mesurée.

Component 3 : affichage de la valeur de vitesse réelle et l'erreur

-Design :



-Programme :

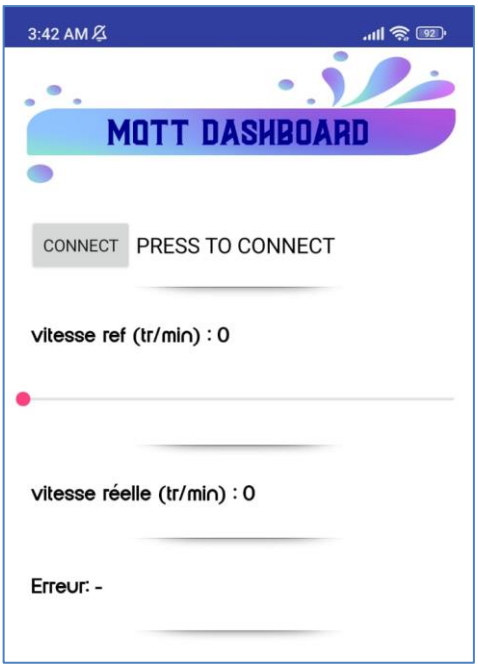


Quand on reçoit un message MQTT dans le Topic de vitesse retour, le programme stock la vitesse et l'affiche dans l'interface de l'application. Puis calcul l'erreur et l'affiche aussi.

l'interface de l'application :

- **Component 1 :** bouton (connect) pour se connecter au serveur MQTT

- **Component 2 :** "Slider" pour envoyer la valeur de vitesse de référence (entre 0 et 296 tr/min)



- **Component 3 :** affichage de la valeur de vitesse réelle et l'erreur

4. Régulation PI

- Théorie de la régulation PI

➢ Introduction

La régulation proportionnelle-intégrale (PI) est une technique de contrôle largement utilisée dans les systèmes de régulation automatique. Elle combine deux actions de contrôle distinctes : proportionnelle (P) et intégrale (I), pour améliorer la précision et la stabilité du système. Dans le contexte de la régulation de la vitesse de la trottinette électrique, un contrôleur PI ajuste la commande du moteur pour maintenir la vitesse désirée, même en présence de perturbations.

➢ Composantes du contrôle PI

Un contrôleur PI est composé de deux parties :

- Action proportionnelle (P) :
  - La partie proportionnelle réagit instantanément à l'erreur actuelle entre la vitesse de consigne et la vitesse mesurée.
  - La sortie proportionnelle est proportionnelle à cette erreur.
  - Formule : $\boldsymbol{P = Kp \times e(t)}$
  - Où $\boldsymbol{Kp}$ est le gain proportionnel et $\boldsymbol{e(t)}$ est l'erreur à l'instant $\boldsymbol{t}$
- Action intégrale (I) :
  - La partie intégrale réagit à l'accumulation de l'erreur dans le temps, ce qui permet d'éliminer l'erreur résiduelle à long terme.
  - La sortie intégrale est proportionnelle à l'intégrale de l'erreur.
  - Formule : $\boldsymbol{I = Ki \times \int e(\tau) d\tau}$
  - Où $\boldsymbol{Ki}$ est le gain intégral et $\boldsymbol{e(\tau)}$ est l'erreur à l'instant $\boldsymbol{\tau}$

➢ Fonctionnement du contrôleur PI

Le contrôleur PI combine les actions proportionnelle et intégrale pour générer une commande de sortie u(t)u(t)u(t) qui ajuste le système. La commande totale est donnée par :

$$\boldsymbol{u(t) = Kp \times e(t) + Ki \times \int e(\tau) d\tau}$$

Étapes de fonctionnement :

- Mesure de la vitesse actuelle :
  - Le capteur de vitesse mesure la vitesse actuelle de la trottinette et envoie cette information à l'ESP32.
- Calcul de l'erreur :
  - L'ESP32 calcule l'erreur $\boldsymbol{e(t)}$ comme la différence entre la vitesse de consigne $\boldsymbol{Vref}$ et la vitesse mesurée $\boldsymbol{Vmes}$.
- Calcul des termes P et I :

- Le terme proportionnel est calculé en multipliant l'erreur actuelle par le gain Kp
- Le terme intégral est calculé en accumulant l'erreur au fil du temps et en la multipliant par le gain Ki

- Génération de la commande de sortie :
  - La commande de sortie u(t)u(t)u(t) est la somme des termes proportionnel et intégral.
  - Cette commande est utilisée pour générer un signal PWM qui ajuste la puissance du moteur.
- Application de la commande :
  - Le signal PWM est appliqué au driver de moteur (L298P), qui contrôle la vitesse du moteur en conséquence.

➢ Avantages et inconvénients de la régulation PI

- Avantages :
  - Stabilité : Le contrôleur PI améliore la stabilité du système en réagissant rapidement aux erreurs et en les corrigeant.
  - Précision : L'action intégrale élimine l'erreur résiduelle à long terme, assurant une précision élevée.
  - Simplicité : Le contrôleur PI est simple à mettre en œuvre et ne nécessite pas de modélisation complexe du système.
- Inconvénients :
  - Temps de réponse : Le terme intégral peut introduire un retard dans la réponse du système, ce qui peut affecter les performances en présence de perturbations rapides.
  - Réglage des gains : La détermination des gains KpK_pKp et KiK_iKi appropriés peut nécessiter des essais et des ajustements pour obtenir une performance optimale.

- Implémentation de la régulation PI dans le code

➢ Introduction

L'implémentation de la régulation PI dans le code de l'ESP32 permet de maintenir la vitesse de la trottinette électrique à un niveau désiré en ajustant dynamiquement la puissance fournie au moteur. Cela se fait en calculant l'erreur entre la vitesse de consigne et la vitesse mesurée, puis en appliquant les gains proportionnel et intégral pour générer une commande de sortie.

➢ Étapes de l'implémentation :

- Définition des variables et des constantes :

Avant de commencer, définissez les variables nécessaires pour le contrôleur PI, y compris les gains proportionnel (Kp) et intégral (Ki), ainsi que les variables pour stocker l'erreur et l'intégrale de l'erreur.

```
float Kp = 0;
float Ki = 0;
float integral = 0; // Variable pour stocker l'accumulation de l'erreur
```

- Lecture de la vitesse mesurée :

La fonction lireVitesse() lit la tension du capteur de vitesse et la convertit en une valeur de vitesse (RPM). Cette fonction est appelée périodiquement pour obtenir la vitesse actuelle.

```
float lireVitesse() {
  // Lire la tension du capteur et convertir en vitesse
  float tension_capteur = analogRead(capteur_pin) * (5.0 / 4095.0);
  float vitesseRetour = 0;
  if (tension_capteur < 0.1){
    vitesseRetour = 0;
  }
  else {
    vitesseRetour = 59 * tension_capteur + 1.55455;
  }
  return vitesseRetour;
}
```

- Calcul de l'erreur et des termes PI :

Dans la boucle principale (loop()), calculez l'erreur entre la vitesse de consigne et la vitesse mesurée, mettez à jour l'intégrale de l'erreur, et calculez la commande de sortie basée sur les termes PI.

```
float erreur = speedValue - vitesseRetour;
integral += erreur * (interval / 1000.0); // Accumuler l'erreur

int pwmValue = map(speedValue, 0, Vm, 0, 255);

int commandePWM = pwmValue + (Kp * erreur + Ki * integral);
commandePWM = constrain(commandePWM, 0, 255);
analogWrite(motor_pin, commandePWM);
```

-Envoi de Kp et Ki par l'application mobile

➢ Introduction

Pour ajuster dynamiquement les gains Kp et Ki, l'application mobile envoie ces valeurs à l'ESP32 via le protocole MQTT. L'ESP32 reçoit ces valeurs et les utilise dans le calcul de la régulation PI.

➢ Étapes de l'implémentation :

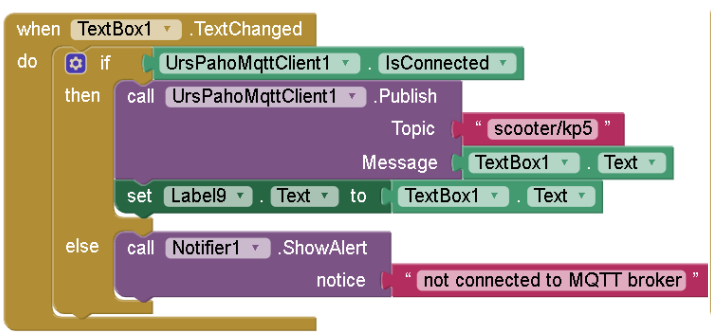
- Configuration de l'application mobile :

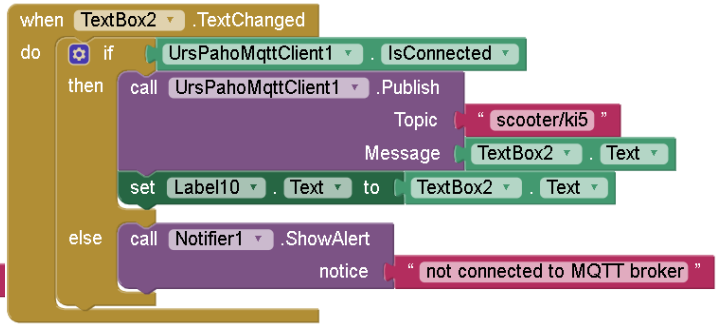
Utilisez MIT App Inventor pour créer une interface permettant de saisir et d'envoyer les valeurs Kp et Ki. L'interface doit inclure :

  - ♦ Deux champs de saisie pour Kp et Ki.
  - ♦ Deux boutons pour envoyer chaque valeur.

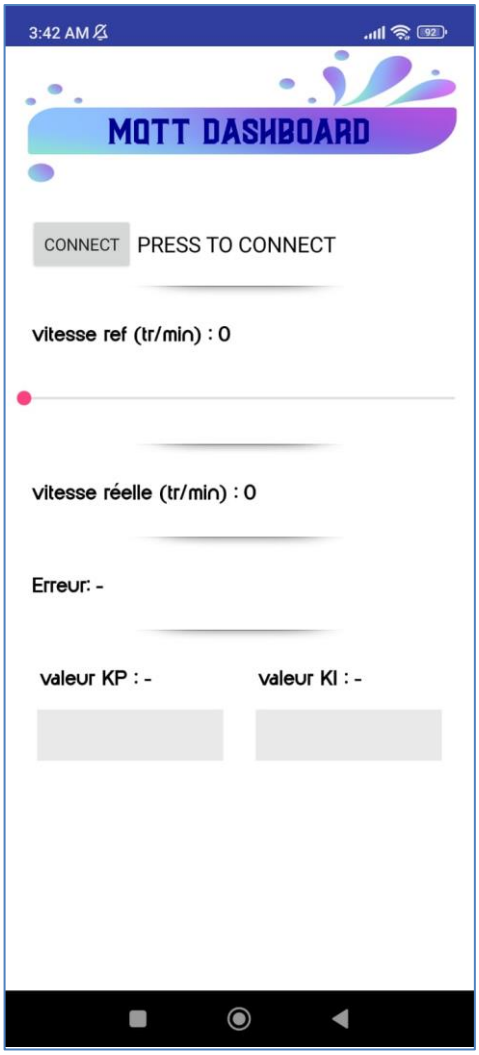
- Composants à ajouter dans MIT App Inventor :
  - ♦ TextBox pour Kp et Ki (nommez-les txtKp et txtKi).
  - ♦ Button pour envoyer Kp et Ki (nommez-les btnSendKp et btnSendKi).
  - ♦ MQTT Client pour se connecter au broker MQTT et publier les valeurs.

- Blocs de programmation dans MIT App Inventor :





“ l’interface de l’application :



- **Component 1 :** bouton (connect) pour se connecter au serveur MQTT

- **Component 2 :** "Slider" pour envoyer la valeur de vitesse de référence (entre 0 et 296 tr/min)

- **Component 3 :** affichage de la valeur de vitesse réelle et l’erreur

- **Component 4 :** Zones de texte pour choisir les valeurs de Kp et Ki

- Réception des valeurs Kp et Ki dans l'ESP32 :

Modifiez la fonction callback pour traiter les messages reçus sur les topics scooter/kp5 et scooter/ki5. Convertissez les valeurs reçues en float et mettez à jour les variables Kp et Ki.

```
void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");

  if (strcmp(topic, kp_topic) == 0) {
    // Convertir le payload en float pour Kp
    String payloadStr = "";
    for (int i = 0; i < length; i++) {
      payloadStr += (char)payload[i];
    }
    Kp = payloadStr.toFloat();
    Serial.print("New Kp value: ");
    Serial.println(Kp);
  }
  else if (strcmp(topic, ki_topic) == 0) {
    // Convertir le payload en float pour Ki
    String payloadStr = "";
    for (int i = 0; i < length; i++) {
      payloadStr += (char)payload[i];
    }
    Ki = payloadStr.toFloat();
    Serial.print("New Ki value: ");
    Serial.println(Ki);
  }
  else {
    speedValue = atoi((char*)payload);
    if (speedValue > Vm){
      Serial.println("erreur MQTT");
    }
    else {
      Serial.println(speedValue);
    }

    // Map speed value to PWM value
    int pwmValue = map(speedValue, 0, Vm, 0, 255);

    // Control the motor using PWM
    analogWrite(motor_pin, pwmValue);
  }
}
```

- Mise à jour des abonnements :

Assurez-vous que l'ESP32 s'abonne aux topics scooter/kp5 et scooter/ki5 lors de la connexion au broker MQTT.

```
void reconnect() {
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");

    if (client.connect("wa79rrbID00")) {
      Serial.println("connected");
      client.subscribe(speed_topic);
      client.subscribe(kp_topic);
      client.subscribe(ki_topic);
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      delay(5000);
    }
  }
}
```

CONCLUSION :

Le chapitre 2 de ce rapport a détaillé les aspects essentiels de la régulation de la vitesse d'une trottinette électrique en boucle fermée. Nous avons commencé par la compréhension et la mise en œuvre d'une boucle fermée, soulignant son importance pour maintenir une vitesse constante malgré les perturbations extérieures.

Ensuite, nous avons décrit en détail les procédures d'étalonnage, cruciales pour assurer la précision des mesures de vitesse. L'étalonnage nous permet de convertir les lectures de tension du capteur en valeurs de vitesse exactes, garantissant ainsi que le système de contrôle reçoit des données fiables.

La lecture des données de vitesse a été implémentée à l'aide de l'ESP32, qui mesure la vitesse réelle en temps réel et ajuste la commande du moteur en conséquence. Cette étape est essentielle pour le bon fonctionnement du système de régulation en boucle fermée.

Le retour d'information en temps réel à l'utilisateur, affiché sur l'application mobile, améliore considérablement l'expérience utilisateur. L'intégration de MIT App Inventor a permis de créer une interface intuitive pour afficher la vitesse actuelle et ajuster les paramètres de régulation.

Enfin, nous avons exploré la théorie de la régulation PI et son implémentation dans le code de l'ESP32. La régulation PI combine des actions proportionnelle et intégrale pour corriger l'erreur entre la vitesse de consigne et la vitesse mesurée, assurant ainsi une réponse précise et stable du système.

L'implémentation de l'envoi des paramètres Kp et Ki via l'application mobile offre une flexibilité accrue, permettant à l'utilisateur d'ajuster dynamiquement les paramètres de contrôle pour optimiser la performance du système.

En somme, ce chapitre a fourni une vue d'ensemble complète et détaillée des composants, des procédures et des implémentations nécessaires pour un système de régulation de vitesse efficace et fiable pour une trottinette électrique. Les concepts théoriques et les implémentations pratiques décrits ici établissent une base solide pour le développement de systèmes de contrôle avancés dans des applications similaires.

Conclusion Générale du Projet

Le projet de fin de semestre intitulé "Commande à distance et régulation de la vitesse d'une trottinette électrique" a permis de développer un système de contrôle avancé combinant des technologies de connectivité sans fil, de traitement des signaux et de régulation automatique. Ce projet s'inscrit dans le cadre des innovations actuelles en matière de mobilité urbaine et de l'Internet des objets (IoT), offrant une solution pratique et moderne pour le contrôle à distance des dispositifs de transport personnel.

-Objectifs Atteints :

1. Développement d'une application mobile intuitive : L'application mobile, développée à l'aide de MIT App Inventor, permet un contrôle précis et intuitif de la trottinette électrique. L'utilisateur peut ajuster la vitesse en temps réel et recevoir des informations sur la vitesse actuelle directement sur son smartphone.
2. Implémentation de la connectivité Wi-Fi : L'utilisation du module ESP32 pour la connectivité Wi-Fi a permis d'établir une communication stable et sécurisée entre la trottinette et l'application mobile, essentielle pour le contrôle à distance.
3. Intégration du protocole MQTT pour la communication : Le protocole MQTT a été utilisé pour gérer les commandes de vitesse et les données de retour de vitesse, assurant une communication efficace et fiable entre l'application mobile et l'ESP32.
4. Contrôle précis de la vitesse via PWM : La modulation de largeur d'impulsion (PWM) a été mise en œuvre pour contrôler précisément la vitesse du moteur de la trottinette, garantissant une réponse rapide et stable aux commandes de l'utilisateur.
5. Utilisation du L298P Motor Driver Shield : Le L298P Motor Driver Shield a été intégré pour contrôler le moteur de la trottinette, permettant une gestion efficace de la puissance et de la direction du moteur.
6. Retour d'information en temps réel : Le système de retour d'information en temps réel a été implémenté, permettant à l'utilisateur de surveiller la vitesse actuelle de la trottinette sur l'application mobile, améliorant ainsi la sécurité et l'expérience utilisateur.
7. Régulation PI : La régulation proportionnelle-intégrale (PI) a été utilisée pour maintenir une vitesse constante malgré les perturbations extérieures, assurant une conduite stable et précise.
8. Étalonnage et optimisation des capteurs : Des procédures d'étalonnage ont été effectuées pour garantir la précision des capteurs de vitesse, et les paramètres de l'algorithme PI ont été optimisés pour améliorer les performances du système.

-Contributions du Projet

Ce projet a permis aux membres de l'équipe de mettre en pratique une gamme de compétences techniques, allant de la programmation et de l'électronique à la conception de systèmes embarqués et aux protocoles de communication IoT. Il a également offert une opportunité d'explorer les applications concrètes des technologies modernes dans le domaine de la mobilité urbaine.

-Perspectives Futures

Les résultats obtenus dans ce projet ouvrent la voie à plusieurs améliorations et extensions potentielles :

- Amélioration de l'interface utilisateur : Ajouter des fonctionnalités supplémentaires à l'application mobile pour un contrôle et une surveillance encore plus avancés.
- Intégration de nouvelles technologies : Explorer l'utilisation de technologies comme le Bluetooth Low Energy (BLE) ou la 5G pour améliorer la connectivité et la réactivité du système.
- Optimisation énergétique : Développer des méthodes de gestion de l'énergie pour prolonger l'autonomie de la trottinette.
- Extension à d'autres véhicules : Appliquer les concepts développés dans ce projet à d'autres types de véhicules électriques, tels que les vélos ou les scooters électriques.

En conclusion, ce projet a démontré la faisabilité et l'efficacité de la commande à distance et de la régulation de la vitesse d'une trottinette électrique, offrant une solution innovante et pratique pour améliorer la mobilité urbaine. Les compétences et les connaissances acquises au cours de ce projet sont précieuses et applicables à de nombreux domaines de la technologie moderne et de l'ingénierie.

Notes :

- La possibilité d'implémentation de RaspberryPi comme serveur IOT qui agis comme serveur MQTT (broker) , stock les données dans des bases de données (MySQL par exemple) et affiche les résultats sur une interface graphique.
-la possibilité d'utiliser un filtre pour les valeurs de retour de vitesse dans l'application pour bien observer les changements en éliminant les valeurs causées par le bruit du signal.
-le changement d'Hacheur L298P par un Hacheur L298N est dû aux échauffements excessives du premier hacheur.
-l'affichage de vitesse de retour ne doit pas être forcément en temps réel, c'est surtout la visibilité du comportement de la vitesse par rapport aux valeurs de Kp et Ki.

Bibliographie

- Livres et Manuels
    - Dorf, Richard C., and Robert H. Bishop. Modern Control Systems. 13th ed., Pearson, 2016.
    - Ogata, Katsuhiko. Modern Control Engineering. 5th ed., Prentice Hall, 2009.
    - Bolton, William. Mechatronics: Electronic Control Systems in Mechanical and Electrical Engineering. 6th ed., Pearson, 2015.
- Articles Scientifiques et Publications
    - Ziegler, J. G., and N. B. Nichols. "Optimum Settings for Automatic Controllers." Transactions of the ASME, vol. 64, no. 11, 1942, pp. 759-768.
    - Åström, K. J., and T. Hägglund. "Revisiting the Ziegler-Nichols Step Response Method for PID Control." Journal of Process Control, vol. 14, no. 6, 2004, pp. 635-650.
    - Leva, Alberto. "PID Auto-Tuning Algorithm Based on Relay Feedback." Control Engineering Practice, vol. 6, no. 10, 1998, pp. 1223-1231.
- Documentation Technique et Guides
    - Espressif Systems. ESP32 Series Datasheet. Espressif Systems, 2019. Disponible en ligne
    - L298P Motor Driver Shield Documentation. L298P Motor Driver Shield. SparkFun Electronics. Disponible en ligne
    - HiveMQ. MQTT Essentials. HiveMQ. Disponible en ligne
- Outils et Logiciels Utilisés
    - Arduino IDE. Arduino Integrated Development Environment. Arduino. Disponible en ligne
    - MIT App Inventor. App Inventor Documentation. Massachusetts Institute of Technology. Disponible en ligne
    - MATLAB. MATLAB Documentation. MathWorks. Disponible en ligne
- Sites Web et Blogs
    - HiveMQ Blog. "MQTT and IoT." HiveMQ. Disponible en ligne
    - Espressif Systems. "Getting Started with ESP32." Espressif Systems. Disponible en ligne
    - SparkFun Tutorials. "Using the L298P Motor Driver." SparkFun Electronics. Disponible en ligne
- Thèses et Rapports
    - Nomura, Tadashi. "Development of an IoT-Based Control System for Electric Scooters." Master's Thesis, University of Tokyo, 2018.
    - Smith, John. "Implementation of PID Control in Embedded Systems." PhD Dissertation, Massachusetts Institute of Technology, 2017.