

Livrable G

Prototype 2

Soumis à

Emmanuel Bouendeu

Khansaa Salhi

Katrine Labonté

Par

Mariame Ba, 300384093

Ayman Diarra, 300393706

Mackenzie Elora Dutrisac, 300438937

Tristan Larabie, 300441655

Charlotte Marchand, 300446893

Gamila Norelden, 300419887

Dans le cadre du cours

GNG1503 – Génie de Conception

Université d'Ottawa

2025-03-09

Résumé

Dans ce livrable nous allons présenter notre deuxième prototype, une version améliorée du premier, intégrant les améliorations issues de la rétroaction de notre client. Il comprend la mise à jour de notre conception, le code de prototypage en Python, le nouveau plan d'essai, ainsi que la révision de notre NDM. Pour finir, nous y allons détailler notre plan de prototypage pour le développement du troisième prototype.

Table des matières

Résumé	1
Liste des tableaux	3
Liste des figures	3
1. Introduction	4
2. Rétroaction du client et du prof	4
3. Prototype 2	8
3.1. Partie logicielle.....	8
3.1.1. Montage du Arduino.....	8
3.1.2. Code du prototype en Python	10
3.2. Partie interface	11
3.3. Partie Mécanique.....	12
4. Mise-à-jour des spécifications cibles	12
5. Plan d'essai de prototypage.....	13
6. Mise-à-jour du NDM	16
7. Liste des risques importants	17
8. Conclusion	17
Annexes	18

Liste des tableaux

Tableau 1. Résultats de l'exécution du code sous forme de fichier CSV	11
Tableau 2. Les critères de conceptions du produit, mis-à-jour	12
Tableau 3. Plan détaillé du prototype 3.....	13
Tableau 4. Nomenclature des matériaux mise à jour.....	16

Liste des figures

Figure 1. Photo du circuit Arduino monté	9
Figure 2. Circuit Arduino monté pour le prototype 2	9
Figure 3. Capture d'écran de l'exécution du code	10

1. Introduction

Dans le développement de ce projet, l'amélioration continue du prototype est essentielle pour assurer sa fiabilité et sa performance. Après avoir présenté le premier prototype, nous avons pris compte des rétroactions du client afin de perfectionner la conception du prototype 2. L'objectif est d'améliorer la détection du bouton, la stabilité de la rampe et la solidité du montage en ajoutant une base et un ressort de torsion. Ce livrable fait en sorte de présenter les modifications apportées et inclut un plan d'essai pour évaluer la performance du système après ces ajustements.

2. Rétroaction du client et du prof

Le client a exprimé une satisfaction générale face au prototype 1. En effet, le prototype 1 était purement basé sur les informations et remarques collectées des rencontres client précédentes.

Quelques points à améliorer ont été soulevés. L'un des points majeurs qu'il a mentionnés était que si le bouton est trop petit, cela pourrait entraîner des problèmes de détection, donc une bonne solution pour cela consiste à améliorer la sensibilité du capteur pour garantir une meilleure détection.

La fixation de la rampe a également été discutée : le client a suggéré d'ajouter un ressort de torsion au point de pivot pour améliorer la stabilité, ainsi qu'une base sous le bouton pour renforcer la structure. Une rampe plus basse a été proposée, car la force exercée par le passage de la voiture est suffisante. Enfin, le client a conseillé d'ajouter un poteau ou un élastique pour maintenir le montage en place lors de l'utilisation. Ces suggestions seront intégrées dans la prochaine version du prototype.

Commentaire de : Eddy Dutrisac, Gestionnaire des technologies de l'information, Pioneer Construction. Nous avons un groupe appelé "Makers Initiative", où nous

travaillons actuellement sur un robot autonome de peinture de lignes guidé par GPS

Impression Générale

Le **Prototype 2** montre des améliorations significatives par rapport au premier prototype. L'optimisation de la **sensibilité du bouton**, la **stabilisation de la rampe**, et le **renforcement de la structure** sont des avancées qui témoignent d'une bonne prise en compte des retours du client. L'intégration d'un programme en **Python** permettant d'enregistrer les temps de passage dans un **fichier CSV** est également une fonctionnalité utile qui améliore la collecte et l'analyse des données.

Améliorations Possibles

Compte tenu du budget limité, voici quelques améliorations **peu coûteuses** qui pourraient être intégrées :

1. Améliorer la Précision du Capteur

- a. Une alternative plus fiable au bouton pourrait être d'utiliser un **capteur infrarouge (IR) ou un micro-interrupteur** pour détecter le passage des voitures.
- b. **Avantage** : Meilleure **détection des voitures** et réduction des risques d'erreur.

2. Fixation Plus Sécurisée de la Rampe

- a. Ajouter des **supports imprimés en 3D** ou des **pieds en caoutchouc ajustables** pourrait aider à stabiliser l'ensemble et éviter qu'il ne bouge sous l'impact des voitures.
- b. **Avantage** : Amélioration de la **stabilité** et réduction de l'**usure mécanique**.

3. Source d'Alimentation Alternative pour l'Arduino

- a. Une **batterie externe rechargeable** permettrait d'utiliser le système sans dépendance à un ordinateur.
- b. **Avantage** : Permettrait une utilisation en **mode portable**, idéal pour des tests en extérieur.

4. Affichage et Analyse des Données en Temps Réel

- a. L'ajout d'un **graphique en temps réel** via **Matplotlib** ou une interface en **Tkinter** permettrait de visualiser immédiatement les résultats.
- b. **Avantage** : Un affichage **en direct** des classements, rendant le système plus interactif.

5. Boîtier de Protection pour l'Électronique

- a. Actuellement, le circuit est exposé, ce qui peut entraîner des **dommages accidentels**.
- b. Une simple **boîte en plastique** ou un boîtier récupéré pourrait le protéger.
- c. **Avantage** : Augmentation de la **durabilité** et protection contre **la poussière et les chocs**.

Possibilités d'Application pour d'Autres Projets

Ce prototype pourrait être adapté pour plusieurs autres **domaines d'utilisation**, notamment :

1. Éducation & Apprentissage STEM

- a. Les écoles pourraient utiliser ce système pour enseigner **les bases de la programmation, des capteurs et de la collecte de données**.
- b. Il pourrait aussi être adapté pour des **compétitions de robotique** ou de **voitures miniatures**.

2. Courses de Modélisme & Chronométrage Amateur

- a. Ce système pourrait être utilisé pour des courses de **voitures télécommandées**, de **karting** ou d'autres compétitions de vitesse à petite échelle.
- b. Une version améliorée avec plusieurs **capteurs de détection** permettrait de suivre **plusieurs véhicules simultanément**.

3. Surveillance du Trafic à Petite Échelle

- a. Avec des capteurs améliorés, ce projet pourrait être utilisé pour **étudier la circulation dans des zones spécifiques** (ex : écoles, parkings).

- b. Cela pourrait permettre de **mesurer la vitesse moyenne des véhicules** et d'évaluer la **sécurité routière**.
- 4. Chronométrage pour Activités Sportives**
- a. Ce système pourrait être **adapté pour le chronométrage d'activités sportives**, comme **les courses à pied, le parkour ou les parcours d'obstacles**.
 - b. Une version modifiée pourrait inclure **un capteur de mouvement ou un capteur de pression au sol**.
- 5. Application dans notre group "Makers Initiative"**
- a. Dans le cadre de notre projet de **robot autonome de peinture de lignes guidé par GPS**, nous voyons **un potentiel intéressant** pour certaines des solutions développées ici.
 - b. L'interface de suivi des temps pourrait être adaptée pour **suivre la progression du robot sur son parcours**.
 - c. L'amélioration de la détection via capteurs pourrait également être utile pour **assurer un alignement précis lors de la peinture des lignes**.

Conclusion

L'équipe a fait un **travail remarquable** en améliorant le prototype tout en respectant les contraintes budgétaires. **Les prochaines améliorations** devraient porter sur :

- **L'optimisation de la détection** (capteurs IR ou micro-interrupteurs),
- **L'alimentation mobile** pour plus de flexibilité,
- **Un affichage plus interactif** des résultats,
- **Une meilleure protection et stabilisation** du montage.

Ces ajustements **peu coûteux** rendraient le projet **plus fiable, plus polyvalent et plus facilement réutilisable** pour d'autres applications. 🚀

3. Prototype 2

Quoi? Nous avons fait un code qui, à l'aide de l'actionnement d'un bouton, détectera la voiture, le nombre de tours accomplie ainsi que le temps pris pour accomplir chaque tour.

Pourquoi? Ce prototype a pour but de présenter un code qui fonctionne efficacement et montre dans un tableau la voiture, la durée de chaque tour ainsi que le numéro du tour. Le prototype nous permettra de vérifier et de tester le code pour qu'il puisse rencontrer les besoins du client.

Quand? Ce prototype a été faite lors de la semaine du 3 mars et a été accomplie avec succès en deux jours de travail (environ 20 h).

3.1. Partie logicielle

Le prototype 2 intègre une solution logicielle permettant de suivre les temps de passage des voitures et d'enregistrer les résultats dans un fichier CSV. Cette solution repose sur un programme en Python communiquant avec un Arduino via un port série. Le contenu de ce programme va être en annexe dans la section 0.

3.1.1. *Montage du Arduino*

Le montage électronique a été optimisé pour réduire le nombre de connexions filaires. Cette réduction permet d'améliorer la précision des mesures en minimisant les pertes de signal, un aspect critique sachant que les câbles du système final pourraient atteindre une longueur de 2 m.

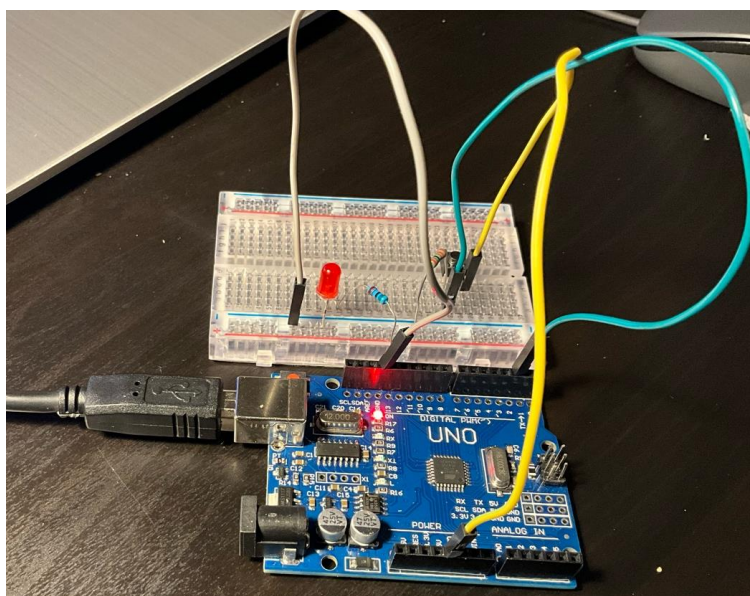


Figure 1. Photo du circuit Arduino monté

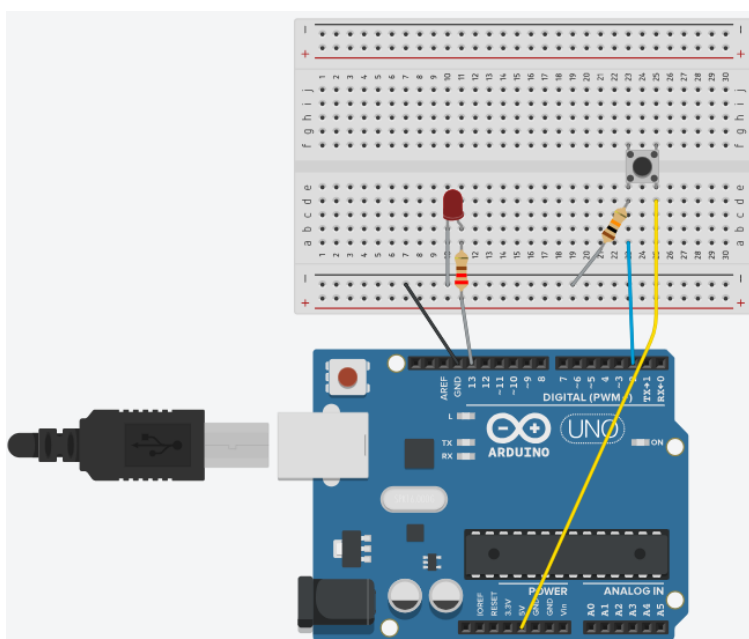


Figure 2. Circuit Arduino monté pour le prototype 2¹

¹ Montage inspiré du site web : <https://www.programmingelectronics.com/tutorial-17-using-a-button-old-version/>

3.1.2. Code du prototype en Python

Le programme Python gère la réception des données envoyées par l'Arduino, effectue un tri des résultats par ordre croissant de temps de course et enregistre ces informations dans un fichier CSV pour analyse ultérieure.

Trois bibliothèques sont nécessaires pour exécuter le programme :

- [pip 25.0.1](#)
- [pyserial 3.5](#)
- csv (intégrée nativement dans Python)

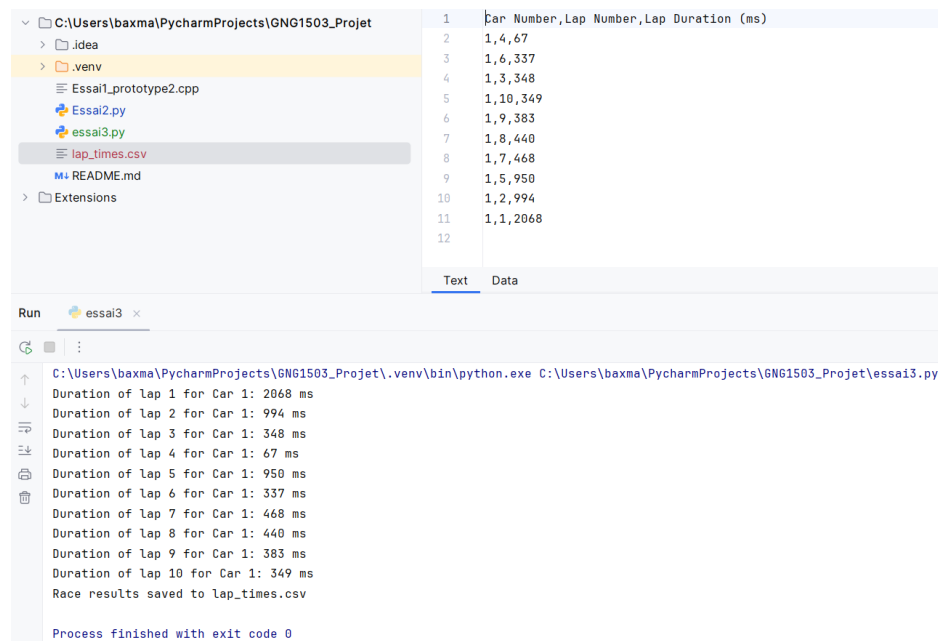


Figure 3. Capture d'écran de l'exécution du code

Le tableau suivant illustre les résultats enregistrés dans le fichier CSV lors d'un test manuel.

Tableau 1. Résultats de l'exécution du code sous forme de fichier CSV

# Voiture	# tour	Durée (ms)
1	3	7
1	7	8
1	10	9
1	6	10
1	5	11
1	9	23
1	8	53
1	2	61
1	4	104
1	1	2846

Les temps varient fortement en raison des tests manuels, Ils sont très courts car on voulait vérifier la précision de lecture du signal du bouton. Cependant, lors de la course réelle, ces variations devraient être plus faibles (durée moyenne de 11 500 ms ou 11.5 secondes par tour si la vitesse de la voiture est de 25 km/h et la distance est de 80 m).

3.2. Partie interface

L'objectif est de développer une interface utilisateur permettant d'afficher en temps réel le classement des voitures. L'interface devra inclure les fonctionnalités suivantes :

- Affichage des résultats sous forme d'un leaderboard.
- Possibilité d'entrer les noms des équipes/écoles avant le démarrage de la course.
- Affichage adapté à un grand écran pour une meilleure visibilité du public.
- Application Windows.

Une référence utile pour développer cette interface en Python: [APPRENDRE LE PYTHON #9 ? INTERFACE GRAPHIQUE \(avec Tkinter\)](#).

Vous pouvez suivre la progression de la conception de l'interface sur notre page [Github](#).

3.3. Partie Mécanique

Cette partie a été répondue, voir la section 2, Rétroaction du client et du prof.

4. Mise-à-jour des spécifications cibles²

Tableau 2. Les critères de conceptions du produit, mis-à-jour

No.	Critères de Conception	Relation (=, < ou >)	Valeur	Unité	Méthode de Vérification
Exigences Fonctionnelles					
1	Compter le nombre de tours des voitures	=	Oui	S.O.	Essai
2	Calculer le temps moyen des tours par les voitures	=	Oui	S.O.	Essai
3	Identifier le gagnant de la course, énumérer les voitures en ordre	=	Oui	S.O.	Essai
4	Fiabilité	>	80	%	Essai
5	Largeur de la rampe	>	370	Mm	Calculs
6	Angle de la rampe	<	15	Degrés	Calculs
Contraintes					
7	Facile à monter	=	Oui	S.O.	Essai Final
8	Accessibilité Windows	=	Oui	S.O.	Programmation et Essai

² Les Spécifications Cibles demeure sans changement du livrable F.

9	Coût	<=	100	\$	Estimation, Vérification
10	Interface	=	Oui	S.O.	Essai
Exigences Non-Fonctionnelles					
11	Rangement Facile (dans une boîte)	=	Oui	S.O.	Essai Final
12	Temps de familiarisation	<	1	Heure	Essai
13	Exporter les données sur Excel	=	Oui	S.O.	Essai
14	L'esthétique	=	Oui	S.O	Essai

5. Plan d'essai de prototypage

Tableau 3. Plan détaillé du prototype 3

# Test		1	2	3
Problème critique probable	<i>Assomptions qui sont faites.</i>	La rampe est assez forte afin de supporter le poids de la voiture lorsqu'elle passe dessus.	Quand les ressorts se compressent et la rampe descend, le bouton est pesé.	Lorsque la voiture passe sur la rampe et le bouton est pesé, l'interface est mise à jour avec les informations actuelles.
Objectif du test	<i>Communication, mesure de la performance, gestion des</i>	Mesurer la durabilité et	Vérifier la fiabilité/actuatio	Vérifier la mise à jour des informations

	<i>risques, apprentissage/compréhension.</i>	stabilité de la rampe.	n du système complet.	dans l'interface sur l'ordinateur.
Description du test	<i>Qu'allez-vous tester précisément ? Quelle est votre hypothèse ?</i>	Faire passer la voiture plusieurs fois dessus et voir si elle craque/brise/penche	Faire passer la voiture dessus plusieurs fois afin de voir si la rampe descend assez pour que le bouton soit pesé.	Faire passer la voiture sur la rampe plusieurs fois, et lorsque le bouton est pesé, on vérifie le logiciel/interface afin de voir si les informations sur la voiture sont bien enregistrées.
Méthode d'analyse	<i>Plus précisément, comment allez-vous tester, en incluant des éléments tels que la durée, la séquence de test, l'équipement, les critères de réussite/échec, etc. Comment les résultats seront-ils collectés ?</i>	Regarder la rampe afin de voir si elle est bien stable et si elle remonte. Tester environ une vingtaine de fois, et si elle fonctionne/est stable, on passe au prochain test.	Vérifier le « output » du logiciel et si la lumière sur le poteau allume. S'il s'allume, la voiture elle est passée. On vérifie aussi le logiciel afin de	Vérifier l'interface sur un ordinateur après que la voiture est passée. Vérifier aussi la vitesse avec laquelle l'interface est mise à jour. Quand la

			voir si le tour sera compté. Quand la fiabilité atteint 90%, les tests arrêtent. Quand la fiabilité atteint 90%, les tests arrêtent.	fiabilité atteint 90%, les tests arrêtent.
Déterminer les éléments mesurables	<i>Que testez-vous avec votre concept (attributs mesurables cibles) ?</i>	Stabilité, rigidité, capacité des ressorts à soutenir la rampe.	Fiabilité, efficacité, vitesse de traitement des donnés.	Fiabilité, vitesse de mise à jour.
Métriques	<i>Quelles mesures allez-vous tester ? Quelles sont les unités associées ?</i>	Masse(g) et masse soutenue (g)	Vitesse de traitement (s) Fiabilité (%)	Vitesse de mise à jour (s) Fiabilité (%)
Type de prototype	<i>Analytique, Physique</i>	Physique, complet.	Physique (électrique) et logiciel, complet.	Physique (électrique) et logiciel, complet.
Niveau de fidélité	<i>HiFi/LoFi Focused, HiFi/LoFi Compréhensive</i>	HiFi, compréhensive,	HiFi, compréhensive.	HiFi, compréhensive

6. Mise-à-jour du NDM

Tableau 4. Nomenclature des matériaux mise à jour

# item	Nom de l'item	Description	Unité de mesure	Quantité	Coût unitaire	Coût étendu
1	Bread Board	Plaque utiliser lors de notre prototypage pour avoir une méthode non permanente.	Unité	1	\$ 0,01	\$ 0,01
2	Arduino UNO	Arduino UNO R3 utilisable avec un câble USB Type A/B	Unité	1	\$ 5,00	\$ 5,00
3	Vis	Pour le Arduino.	Unité	4	\$ 0,01	\$ 0,04
4	Lumière LED	LED de 5 mm	Unité	4	\$ 0,01	\$ 0,04
5	Fils électriques	Fil de 5pi (1 sous par pied)	Unité	5	\$ 0,01	\$ 0,05
6	Proto Board	4 x 6 cm	Unité	4	\$ 1,50	\$ 6,00
7	Resistance	10kΩ	Unité	1	\$ 0,01	\$ 0,01
8	Résistance	220Ω	Unité	4	\$ 0,01	\$ 0,04
9	Câble USB Type A/B	Pour connecter le.s Arduino.s à un ordinateur	Unité	1	\$ 2,75	\$ 2,75
10	MDF	Panneau prédécoupé polyvalent. Épaisseur: 1/8". Dimension: 18 x 24"	Unité	4	\$ 3,00	\$ 12,00
11	Bouton poussoir marche/arrêt momentané	En appuyant sur le bouton, cela s'active et en laissant le bouton, il revient à son état initial.	Unité	25	\$ 0,52	\$ 13,00
12	Acrylique	12"x24"	Unité	1	\$ 14,00	\$ 14,00
13	Ressort	Ressort élastique de trampoline	Unité	10	\$ 1,10	\$ 11,00
Coût total du produit (sans taxes ou livraison)						\$ 63,92
Coût total du produit (avec taxes et livraison)						\$ 72,23

Pour le prototype 2, nous avons utilisé l'équipement suivant: 1 Arduino Uno, 3 fils male-male, 1 DEL, 2 résistors, 1 breadboard ainsi qu'un bouton-poussoir. Sur le côté logiciel, nous avons utilisé les 3 bibliothèques mentionnées plus haut.

A noter, les carreaux rouges indiquent que les éléments sont encore en train d'être analysés par le groupe. Tous les autres carreaux sont bien choisis, les seules modifications pour ces carreaux serait une modification de la quantité du matériel ou le matériel lui-même (si on trouve quelque chose de meilleur).

7. Liste des risques importants

- Le cas où la rampe ne fonctionne pas, et cause des délais majeurs, puisque nous devrions repartir presque à zéro sur le concept mécanique.
- Le cas où le bouton n'est pas bien attaché, et ne pourra pas être activé ou qu'il casse lorsque la voiture passe sur la rampe. Cela est aussi en lien avec les raccordements électriques.
- Délai de livraison du matériel retardé.

8. Conclusion

Ce livrable présente le prototype 2, il contient le développement de la partie interface ainsi que le montage du Arduino et les risques importants. Ce livrable inclus des commentaires du client réel et d'un autre client potentiel qui a été identifié. Leurs commentaires serviront comme guidance pour améliorer le projet avec le prototype 3.

Annexes

Code en python pour le projet

```
"""
Programme de comptage de tours pour une voiture en course.\n
\n
Ce programme reçoit des données de l'Arduino via un port série et enregistre:\n
- Le numéro de la voiture\n
- Le numéro du tour\n
- La durée du tour en millisecondes (ms).\n
\n
Les résultats sont triés en fonction du temps le plus court et enregistrés dans un fichier nommé lap_times.csv.\n
\n
Prototype 2 - Projet GNG1503
\n
Auteur : Mariame Ba (300384093)\n
Date : 08 Mars 2025\n
@uOttawa
"""

# Importation des bibliothèques nécessaires

import serial
import csv

# Définition des variables

# Pour la configuration du port
POSSIBLE_PORTS = ['COM3', 'COM4', 'COM5']
BAUD_RATE = 9600 # Vitesse de communication avec l'Arduino
TIMEOUT = 1 # Délai d'attente pour la lecture série

# Fichier CSV pour enregistrer les résultats
FILENAME = "lap_times.csv"
FIELDNAMES = ['# Voiture', '# tour', 'Durée (ms)']

# Définition des fonctions

def detecter_port():
    """
    Tente de se connecter à l'Arduino en testant plusieurs ports.

    Returns:
        ser (serial.Serial) : Objet Serial si la connexion réussit, sinon None.
    """
    for port in POSSIBLE_PORTS:
```

```

try:
    ser = serial.Serial(port, BAUD_RATE, timeout=TIMEOUT)
    return ser

except serial.SerialException:
    continue

return None

def recevoir_donnees(ser):
    """
    Lit et traite les données reçues via le port série de l'Arduino.

    Args:
        ser (serial.Serial) : Port série actif.

    Returns:
        results (list) : Liste contenant les données de la course (dictionnaires).
    """
    lap_count = 0
    results = [] # Liste pour stocker les tours

    while lap_count < 10: # On suppose une course de 10 tours maximum
        if ser.in_waiting > 0: # Vérifie si des données sont disponibles
            data = ser.readline().decode().rstrip() # Lire et décoder les données

            # Vérifie que la donnée reçue contient bien des nombres
            if not data.replace(',', '').isdigit():
                continue # Ignore cette donnée et continue la boucle

            try:
                car_number, lap_duration = map(int, data.split(',')) # Convertir en entier
                lap_count += 1

                # Ajouter les données du tour à la liste des résultats
                results.append({'# Voiture': car_number,
                               '# tour': lap_count,
                               'Durée (ms)': lap_duration})

                print(f"La voiture {car_number} a fait le tour {lap_count} en {lap_duration} ms. ")

            except ValueError:
                break

    return results

def sauvegarder_csv(results):
    """

```

Enregistre les résultats de la course dans un fichier CSV après les avoir triés.

Args:

results (list) : Liste des résultats de la course.

"""

`sorted_results = sorted(results, key=lambda x: x['Durée (ms)'])` *# Tri des résultats*

`with open(FILENAME, mode='w', newline='') as file:`

`writer = csv.DictWriter(file, fieldnames=FIELDNAMES)`

`writer.writeheader()`

`writer.writerows(sorted_results)`

`def main():`

"""

Fonction principale du programme:\n

- Détecte l'Arduino\n

- Récupère les données de la course\n

- Enregistre les résultats dans un fichier CSV\n

"""

`ser = detecter_port()` *# Connexion au port série*

`if ser:`

`print("La course peut commencer.")`

`results = recevoir_donnees(ser)` *# Lecture des données*

`sauvegarder_csv(results)` *# Enregistrement des résultats*

`ser.close()` *# Fermeture propre du port série*

Exécution du code

`if __name__ == "__main__":`

`main()`

Lien d'accès du Trello

[Projet de conception - GNG 1503 - FF031 | Trello](#)

Projet de conception - GNG 1503 - FF031

☆

👤

📅

Tableau

▼

📅

🔍

⚡

⌵

Filtres

DA

C

GN

MD

MB

+3

Tâches

Rédaction du livrable H

👁 ⌚ 23 mars

DA

C

GN

MD

MB

TL

plan d'essai de prototypage et les résultats des tests du prototype 3

Rétroactions du client

Mise à jour de la NDM

+ Ajouter une carte

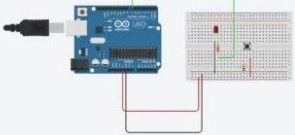
📄

Terminé

plan d'essai détaillé du prototype 3

🕒 9 mars 1 1

C



Modèle analytique, numérique ou expérimentale

🕒 9 mars 1

C

MB

GN

+ Ajouter une carte

📄

Documents

Prototypes à faire

🔗 1

Information sur le projet

👁 ⌵ 1

MB

DA

C

GN

TL

Notes du meeting 1 avec le client

⌵ 🔗 1

+ Ajouter une carte

📄

21