UCLouvain

UCLouvain - Sciences Informatiques

APP3 - Groupe B.20 - Rapport Microbit

Groupe :
Enzo ANDRADE ORLETTI
Jad M'RABTI
Viktor PROFIROV
Pedro SIBOMANA

Table des matières

10 décembre 2024

1 Introduction

L'idée qui sous-tend notre projet repose sur le fait que le gouvernement belge vise à rajeunir sa population et nous a recrutés précisément pour contribuer dans ce domaine. Notre mission consistait à développer une technologie facilitant le quotidien des parents et, surtout, améliorant le bien-être des enfants.

Pour cela, nous avons utilisé deux micro :bits que nous avons programmés tout au long du projet afin d'établir une communication claire et détaillée entre eux. Ainsi, cette solution offre une meilleure expérience à l'utilisateur, à savoir le parent, tout en simplifiant son rôle de responsable de l'enfant.

2 Tâches du projet

2.1 Commandes Git

Dans le cadre de l'activité préparatoire au Projet 3, nous avons réalisé les exercices liés aux commandes Git mentionnées dans le syllabus. Comme ce projet est plus long que les deux précédents, nous avons créé un dépôt sur GitHub pour stocker notre projet. Grâce aux commandes Git, nous avons pu intégrer notre machine locale avec le dépôt distant. Cela nous a permis de sauvegarder nos progrès de manière progressive, rendant ainsi le processus de développement du code moins éprouvant.

2.2 Découverte de l'environnement de programmation du microbit

Le micro :bit met à disposition une documentation riche en Python pour permettre aux utilisateurs de le programmer avec ce langage. Le projet peut ainsi être divisé en deux parties :

— La première consiste en une découverte approfondie de l'environnement.
— La seconde débute après l'introduction aux concepts de cryptographie.

Nous avons réussi à établir une corrélation efficace entre les concepts de Python et les méthodes disponibles pour le micro :bit dans cet environnement de programmation. Jusqu'au début de la seconde partie, nous avons finalisé un premier brouillon de notre code final, bien que les messages n'étaient pas encore cryptés.

2.3 Cryptographie et défi

Après le cours d'introduction aux concepts de cryptographie et de sécurité, il était temps de réfléchir à leur application concrète dans le projet. Pour ce faire, nous avons utilisé les structures de code pour le Parent et l'Enfant, disponibles sur Moodle, qui ont servi de base pour développer une communication cryptée entre eux.

Le premier objectif consistait à concevoir un défi crypté dans lequel, à la fin, le Parent et l'Enfant obtiendraient de nouvelles clés identiques. Un point crucial pour la réussite de ce défi a été la manipulation du paramètre `decryption` de la fonction `vigenere()`. En définissant `decryption=False`, nous avons indiqué que nous voulions chiffrer un message, tandis que `decryption=True` signifiait que nous voulions le déchiffrer.

Ce concept a été fondamental pour remplir les fonctions requises (présentes dans la structure de code) et, par conséquent, pour établir une communication chiffrée tout au long du projet.

2.4 Fonctionnalité supplémentaire

Étant donné la large présence de capteurs sur le micro :bit, nous ne nous sommes pas contentés d'implémenter une seule fonctionnalité supplémentaire. Une fois que nous avions déjà mis en place les fonctionnalités obligatoires, nous avons utilisé le temps restant pour explorer cette vaste gamme d'options. Notre stratégie a été de créer une fonctionnalité pour chaque capteur afin de mettre en

évidence cette grande diversité. Cela a également été un facteur déterminant pour décider, à la fin, quelles fonctionnalités resteraient dans la version finale du projet et lesquelles seraient supprimées en raison de la limitation de la mémoire du micro :bit. Les fonctionnalités supplémentaires qui réutilisaient des capteurs déjà utilisés ont donc été supprimées, libérant ainsi de l'espace pour de nouvelles fonctionnalités.

2.5 Tests

Pour la réalisation des tests, nous les avons également divisés en deux parties : les tests dans l'environnement de programmation du micro :bit lors de la première étape, puis les tests sur les micro :bits réels lors de la phase finale, après avoir transféré le code sur ces derniers.

Dans un premier temps, pour les tests dans l'environnement de programmation, nous avons adopté la stratégie d'ouvrir deux onglets simultanément : l'un contenant le code de l'*Enfant* et l'autre celui du *Parent*. Cela nous a permis de tester plus facilement les fonctionnalités utilisant la communication radio, en copiant et collant simplement les messages produits par l'un pour vérifier si l'autre micro :bit réagissait comme attendu à la réception du message.

Cette méthode nous a déjà permis de corriger de nombreux bugs détectés à ce stade. Ensuite, lors de la phase finale où nous testions directement sur les micro :bits physiques, il devenait plus évident d'identifier les erreurs restantes. Nous avons suivi une approche incrémentale, en implémentant et testant chaque fonctionnalité une à une, ce qui a facilité la détection et la correction des problèmes.

3 Fonctionnement du groupe

Tout comme dans les deux projets précédents, notre groupe a fait preuve d'une bonne synergie, et, encore une fois, la communication a joué un rôle fondamental dans l'avancement de notre projet.

Les différents points de vue au sein de l'équipe ont été essentiels pour l'implémentation de diverses fonctionnalités, reflétant la grande variété d'interactions possibles entre les Parents et les Enfants.

Tous les membres ont fait preuve d'une grande disponibilité et d'un engagement envers le projet, où nous avons recherché ensemble les matériaux et acheté la batterie pour les micro :bits, en divisant le coût entre nous.

Nous avons également réussi à respecter notre planning défini au début du projet sans grands obstacles, et cela a donc été un élément clé pour nous guider tout au long de notre parcours.

4 Problèmes rencontrés

4.1 Cryptographie

L'un des obstacles rencontrés a été l'implémentation de la cryptographie juste après un cours bref, mais chargé de nouveaux concepts. Pour surmonter cette difficulté, poser nos questions aux tuteurs a été d'une grande aide. Nous avons ainsi obtenu des informations précieuses : les fonctions `hashing(string)` et `vigenere(message, key, decryption)` n'avaient pas besoin d'être modifiées, et le concept de *nonce* ainsi que son rôle dans le *challenge* ont été clarifiés. Grâce à ces éclaircissements, nous avons pu mener à bien cette phase de cryptographie de manière autonome.

4.2 Mémoire du microbit

En raison de la taille de notre code, le micro :bit l'a rejeté lors du transfert, indiquant qu'il n'y avait plus suffisamment d'espace. Cela nous a conduit à effectuer une analyse globale du code afin de

l'optimiser.

Dans un premier temps, nous avons supprimé tous les commentaires du code, ce qui a fonctionné temporairement, mais n'a pas suffi à mesure que le projet progressait. Nous avons ensuite examiné le code de manière plus approfondie pour retirer toutes les fonctions et variables inutilisées.

4.3 Optimisation des boutons

En parallèle de l'optimisation nécessaire en raison des contraintes de mémoire, l'optimisation de l'utilisation des boutons du micro :bit a également été cruciale pour garantir leur bon fonctionnement.

Initialement, nous avons exploré les différents boutons disponibles et attribué à chacun une fonctionnalité spécifique. Cependant, après avoir constaté que certains boutons ne répondaient pas de manière optimale lorsqu'ils étaient pressés, nous avons décidé de créer un menu. Ce menu propose des options allant de 1 jusqu'au nombre total de fonctionnalités, chaque numéro correspondant à une fonctionnalité implémentée. Une fois notre choix confirmé, la fonctionnalité correspondante s'exécutait. Cette approche nous a permis d'obtenir un gain de performance significatif.

4.4 Environnements de développement (IDE)

Un des aspects qui a particulièrement retenu notre attention durant le projet a été le fait qu'il n'était pas possible de faire des *commits* directement depuis l'environnement de programmation du micro :bit. Nous avons donc eu recours à une IDE, en l'occurrence *PyCharm*, qui a servi d'intermédiaire pour permettre l'envoi du code vers notre dépôt GitHub.

La solution adoptée consistait à effectuer de nombreux *commits* pour sauvegarder chaque petite étape du projet. Cette méthode s'est avérée nécessaire, car en éteignant l'ordinateur, le code aurait été perdu, n'étant pas stocké dans une sorte d'« IDE traditionnel ».

5 Conclusion

À la fin du projet, nous estimons que notre autonomie a été davantage stimulée par rapport aux deux projets précédents, en raison de la complexité accrue de celui-ci. L'introduction d'un nouvel environnement de programmation, avec des méthodes spécifiques à celui-ci, ainsi que la nécessité d'implémenter une fonctionnalité supplémentaire, ont été des facteurs clés pour allier notre créativité à cette autonomie renforcée.

En termes d'apprentissage, nous avons acquis des compétences en gestion de versions de code grâce aux commandes Git et à l'utilisation de GitHub. Le fait de présenter un code propre, l'un des concepts de bonnes pratiques en Python, a été essentiel compte tenu de la taille du projet : plus le code est clair et simple, mieux c'est. Nous avons intensifié nos connaissances des concepts de base en Python, et grâce à la corrélation entre ce que nous avons appris dans le cours "Introduction à la Programmation" et ce qui était demandé dans le projet, nous avons pu développer le code de manière fluide.

Notre groupe, pour sa part, a fait preuve d'un bon niveau de coopération, où chacun a contribué afin que nous atteignions le résultat que nous nous étions fixé.