Rémi BELLICOT
Johanna BOITEUX

SYNTHÈSE PROJET WATCH'ART

Introduction :

Ce projet a pour but de concevoir un système de sécurité pour protéger les œuvres les plus précieuses du musée. Chaque œuvre à protéger est équipée d'un capteur ultrasons, d'un buzzer et d'une carte Arduino, afin de mesurer la distance de sécurité entre l'œuvre et le public (ou un potentiel voleur).
Chaque capteur est relié à une carte Raspberry Pi. Si la distance de sécurité est franchie pour l'un des capteurs, le buzzer se déclenche afin de faire reculer la personne.
Puis, la carte Arduino envoie un signal sans fil contenant le nom du tableau et la salle dans laquelle se trouve le tableau à la carte Raspberry Pi. Cette dernière envoie ensuite par mail un message de détresse.
Dans notre projet, une LED et un signal sonore sont attribués à chaque tableau protégé sur un "panneau de contrôle". (Dans une application plus générale, le "panneau de contrôle" serait situé dans la salle de sécurité du musée).

Mode d'emploi :

Assurez-vous de disposer au préalable du kit GrovePi.
Les librairies python *grovepi*, *time* et *smtplib* nécessaires au bon fonctionnement du programme principal sont normalement déjà installées.
Aucun logiciel spécifique n'est nécessaire, seulement un compilateur python qui est déjà présent sur Raspbian.
Brancher la LED au port 3, le capteur ultrasons au port 4 et le buzzer au port 5 de la carte Arduino.
Disposer l'ensemble près de l'œuvre à protéger. Le capteur ultrason doit être dirigé vers le public.
Allumer la carte Raspberry Pi et la placer dans une autre pièce.
Se connecter à la boite mail de la personne à prévenir en cas d'intrusion.

(Attention : boîte gmail dans ce projet)
Pour tester le système de protection de l'œuvre, s'approcher à moins de 30 cm du capteur.
Une alarme visuelle et sonore doit normalement s'enclencher, puis un mail doit être envoyé à l'adresse spécifiée.

Moyens matériels :

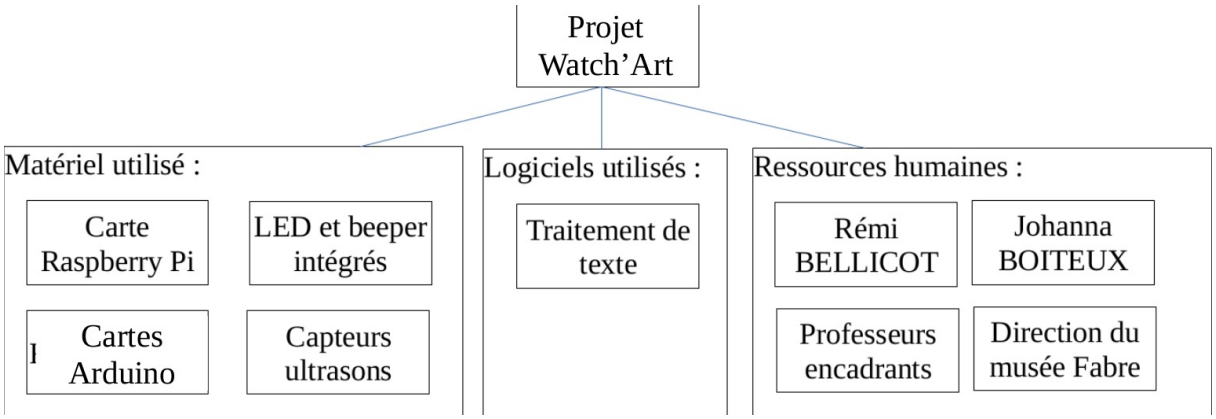
Cartes Raspberry Pi pour traiter les signaux d'alerte et envoyer le mail.
Carte Arduino, sur laquelle le programme de détection du capteur tourne en permanence.
Capteurs ultrasons pour mesurer la distance de sécurité entre l'œuvre à protéger et le public.
Des LEDs et des buzzers pour des signaux d'alarme visuels et sonores.
Tous ces capteurs ont été intégrés dans le kit Grove :
http://www.seeedstudio.com/wiki/Grove_-_Ultrasonic_Ranger
http://www.seeedstudio.com/wiki/Grove_-_Buzzer
http://www.seeedstudio.com/wiki/Grove_-_LED_Socket_Kit
Nous avions prévu d'utiliser un signal wi-fi comme dispositif de communication sans fil entre les cartes.

Moyens humains :

Rémi BELLICOT : programmeur
Johanna BOITEUX : programmeur

Architecture du projet :

Partie matérielle :
Nous avons utilisé une carte Raspberry Pi et des cartes Arduino avec des capteurs ultrasons pour les calculs de distance, ainsi que le buzzer et la LED intégrés pour les signaux d'avertissement sur le "panneau de contrôle".
Partie logicielle :
Nous avons utilisé l'éditeur de texte SublimeText pour la modification du code en langage python. Les données récoltées par le Raspberry Pi ont fait l'objet d'un traitement.
Schéma :



Code :

Le fichier *watch_art.py* contient le programme principal.
Celui-ci enclenche le buzzer, allume la LED et envoie un message d'alerte via Gmail. Il importe les librairies python *smtplib* pour la fonction d'envoi du message et *time* pour la durée d'action du buzzer, mais aussi le paquet MIMEMultipart (du module email.mime.multipart) nécessaire à l'envoi d'un message.
Les fonctions utilisées issues du module *grovepi* sont :

- UltrasonicRead(), qui récupère les distances mesurées par le capteur ultrasons. Ces données seront ensuite traitées dans le programme principal.
- AnalogWrite(), qui fait sonner le buzzer au volume maximal et permet de l'éteindre.
- DigitalWrite(), qui allume et éteint la LED.

Le module *time* fournit la fonction sleep() qui met en pause le buzzer pendant 0,5s.
Les fonctions utilisées issues du module *smtplib* sont :

- SMTP_SSL(), qui permet de se connecter au serveur à l'adresse smtp.gmail.com depuis le port 465
- login(), qui permet de se connecter au serveur en entrant un nom d'utilisateur et un mot de passe.
- sendmail(), qui envoie l'en-tête (adresse mail de l'expéditeur, adresse mail du destinataire) et le contenu message
- quit(), qui se déconnecte du serveur

Perspectives :

Au jour du rendu, notre projet n'a pas pu être terminé à 100%.
Les retards sur l'échéance sont dûs à plusieurs raisons.

Notre progression a principalement souffert de l'obligation de mener plusieurs projets de front, certains (comme celui d'Algorithmique) étant tout aussi conséquent que celui-ci,
Comme nous ne pouvions pas nous voir pendant les vacances, les heures d'autonomie ont été les seules où tous les membres du groupe pouvaient se retrouver en dehors des cours et travailler ensemble. Ces heures d'autonomie nous ont été très utiles et productives ; c'est pourquoi nous aurions voulu en avoir plus.
De plus, nous ne possédions pas non plus le matériel adéquat en dehors de celui fourni par l'établissement. Nous ne pouvions donc pas progresser séparément sur le projet. Il aurait ainsi été plus pratique pour nous que l'établissement nous laisse emprunter du matériel pour nous permettre de travailler chez nous.
Enfin, le matériel qui nous était réservé (les cartes Raspberry Pi notamment) avait disparu lors des dernières séances, celles où nous avions prévu de tester nos programmes. À cause de cela, nous n'avons pas pu effectuer ces tests.
Nous estimons ainsi qu'il nous faudrait encore un mois pour que les deux membres du groupe puissent terminer ce projet.
Si nous avions eu plus de temps, nous aurions voulu gérer 2 cas selon l'horaire où aurait lieu l'infraction :

- Le jour, la carte Raspberry Pi enverrait un message sur le portable du gardien (ou du propriétaire de l'œuvre) afin qu'il intervienne.
- La nuit, la carte Raspberry Pi enverrait un message au standard de la police afin qu'elle se rende sur place.