Rapport de BE C++ : Détecteur par ultrasons

INTRODUCTION

Grâce au microcontrôleur ESP8266, nous avons voulu construire un système de surveillance avec alarme capable d'informer l'utilisateur par mail en temps réel. Nous voulions un système simple d'utilisation et peu onéreux qui n'a pas besoin de caméra pour fonctionner, mais qui reste complet et efficace. Notre première approche consistait en de la surveillance par détection de vibrations. Ce système ne paraissait pas assez efficace car le capteur de vibration n'était pas assez sensible pour détecter des pas par exemple. Nous nous sommes donc penchés sur autre solution : la détection par ultrasons. Le capteur par ultrasons possède des caractéristiques très intéressantes : celui-ci a une portée de plusieurs mètres et est facilement manipulable.

MÉTHODOLOGIE

Notre travail s'est décomposé en plusieurs étapes. Dans un premier temps, nous choisissons nos composants pour mettre en place le système de détection. Notre choix s'est porté sur un détecteur à ultrasons, un buzzer et un microphone. Dans un second temps, nous testons chaque composant (Wi-Fi inclus) avec des exemples disponibles sur internet. Ensuite, nous créons et testons une fonction spécifique à notre projet pour chaque composant. Enfin, nous regroupons ces fonctions dans un seul et même programme principal et ajoutons d'autres fonctionnalités.
L'ensemble du projet a été réalisé en binôme. Nous avons préféré ne pas nous répartir le travail pour bien assimiler l'utilisation de chaque composant et privilégier l'entraide dans la création des fonctions. Chaque membre du binôme a ainsi une parfaite connaissance de la structure et du fonctionnement du code lié au projet.

LE FONCTIONNEMENT MATÉRIEL DU DÉTECTEUR

Le détecteur à ultrasons se configure facilement. Le schéma de fonctionnement matériel se présente ainsi :

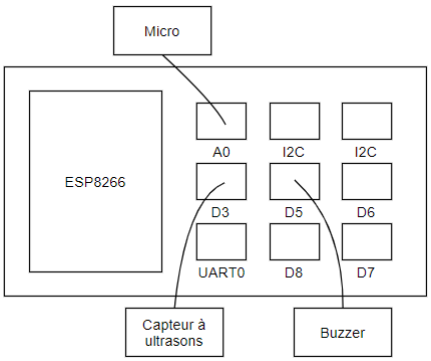

Figure 1 : Schéma de fonctionnement matériel

Le microphone s'installe sur le port analogique tandis que le buzzer et le capteur à ultrasons se branchent sur des ports numériques. Le microphone doit être installé à un endroit accessible et de préférence silencieux. Le capteur à ultrasons doit être placé en face d'une surface plane située à moins de 3 mètres pour une performance optimale.

LA STRUCTURE DU CODE

Le programme utilisé pour le détecteur à ultrasons se décompose en plusieurs classes présentées ci-dessous :

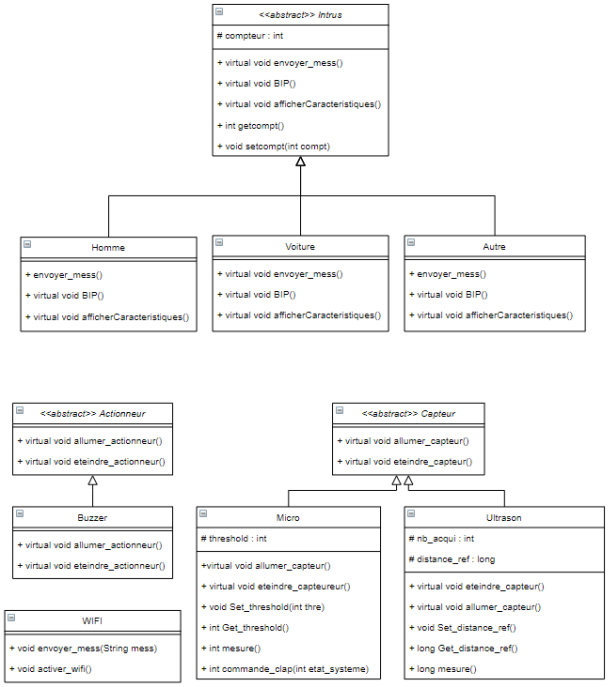

Figure 2 : Diagramme de classes

Nous avons choisi de créer une classe abstraite pour les capteurs et pour les actionneurs. Ceux-ci partagent en effet des fonctions qui doivent être redéfinies pour chaque capteur ou actionneur. Nous avons également créé une classe abstraite Intrus car nous souhaitons pouvoir distinguer chaque intrus par des fonctions spécifiques. Enfin, une classe WIFI est créée pour la gestion des mails et de la connexion réseau.

SCHÉMA DE FONCTIONNEMENT LOGICIEL

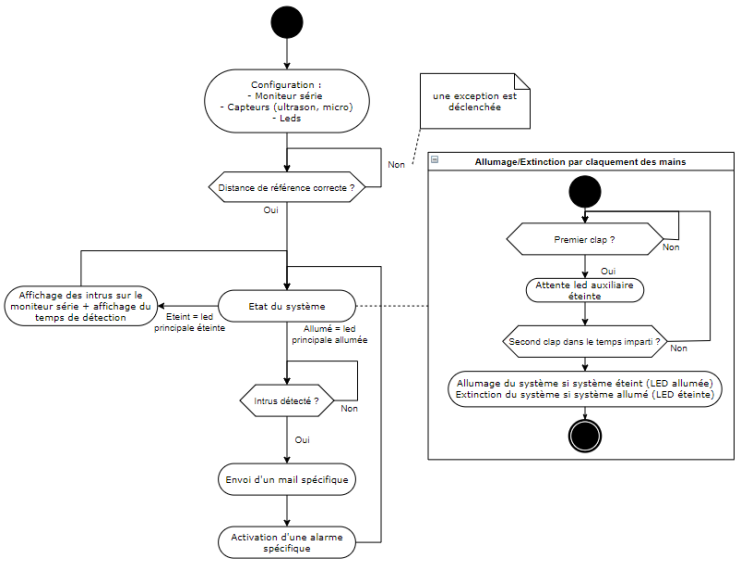

Figure 3 : Schéma de fonctionnement logiciel

Ci-dessus, vous trouverez le schéma fonctionnel du programme (une image vaut parfois mille mots !). C'est un programme d'apparence simple qui demande une courte phase d'initialisation. Par la suite, il se comporte différemment en fonction de son état : alarme active ou non. Nous avons tenté de lui ajouter des fonctionnalités plus complexes telles que l'envoi de mail par un module Wi-Fi, la reconnaissance du type détecté (Homme, voiture, autre) ou encore un changement d'état par claquement de mains. Cette dernière fonction demande par ailleurs une certaine robustesse.

LA ROBUSTESSE

La robustesse a été un point clé à traiter durant notre projet, le côté ludique de changer l'état de l'alarme en claquant des mains apportant en effet son lot de soucis. Par exemple, il est crucial que l'alarme ne se désactive pas lorsque le moindre bruit ambiant se fait entendre dans un périmètre proche de la carte. Pour contrer cela, nous avons adopté deux choix principaux. Le premier a été d'augmenter le seuil nécessaire de bruit pour déclencher l'activation/désactivation à un niveau sonore assez haut de telle sorte qu'une personne parlant normalement à un mètre du capteur ne déclenche pas un changement du système. Cette solution ne nous semblant pas suffisante, nous avons décidé d'ajouter une autre condition. Quelques secondes après le premier clap, un deuxième clap est nécessaire dans un temps imparti très court. La probabilité pour que l'alarme se désactive autre que par la volonté de l'utilisateur est maintenant très faible, mais reste non nulle. En effet, si un bruit fort et continu est détecté, il est fort probable que l'alarme se désactive. Cet évènement reste toutefois assez rare.

RESPECT DU CAHIER DES CHARGES

Au-delà de la création de classes et du mécanisme d'héritage, nous devions respecter trois critères de plus dans notre code.
Nous avons utilisé la STL, surchargé un opérateur et défini une exception de manière à ce qu'elles aient un réel intérêt dans notre code.
La STL nous a permis de créer une liste d'intrus afin de pouvoir afficher les caractéristiques des objets passés devant le capteur pendant son fonctionnement, dès lors que nous l'éteignons.
La surcharge de l'opérateur += nous permet de savoir combien de secondes des objets sont restés ou passés devant le capteur. Nous incrémentons un objet de type Intrus avec d'autres objets du même type avec +=. Cela permet d'additionner le compteur de chaque intrus. A la fin du programme, nous récupérons et convertissons ce nombre en secondes.
Enfin, l'exception nous permet de traiter le cas où l'utilisateur aurait fait une mauvaise manipulation lors de l'instant où il fixe son capteur (par exemple, passer sa main au début de l'exécution). Si le capteur reçoit une distance référence inférieure à 60 cm, il lève une exception et indique à l'utilisateur qu'une nouvelle distance va être mesurée.

CONCLUSION

Nous avons fini par atteindre notre objectif : avoir une alarme facile à mettre en place, pas chère, relativement discrète et robuste aux événements extérieurs. Celle-ci peut vous avertir rapidement de l'intrusion d'une personne dans une pièce ou sur une propriété.
Le projet s'est déroulé sans accroc et nous semble complet. Avec du recul, l'exception aurait sûrement pu être mieux utilisée pour s'assurer que l'utilisateur a bien fixé le capteur. Par exemple, prendre une première mesure, une deuxième au bout de 5 secondes et vérifier qu'elles correspondent. De plus, l'utilisation du capteur à ultrasons a nécessité d'instancier un objet Ultrasonic pour chaque fonction associée à celui-ci. Cela implique la présence de lignes de codes supplémentaires qui ne sont pas forcément nécessaires.