STEMKIT 4SCHOOLS

UTILISATION DE CAPTEURS IR DANS LES SYSTÈMES D'ALARME

Plan de leçon 1

Cofinancé par le programme Erasmus+ de l'Union européenne

Ce projet a été financé avec le soutien de la Commission européenne.

Table des matières

1 Utilisation de capteurs IR dans les systèmes d'alarme 2
1.1 Informations générales 2
1.1.1 Description succincte 2
1.1.2 Objectifs d'apprentissage 2
1.1.3 Liens avec le curriculum 2
1.1.4 Matériel requis 3
1.1.5 Durée 3
1.2 Plan de leçon 4
1.2.1 Introduction 4
1.2.2 Préparation 4
1.2.3 Enquête 8
1.2.4 Conclusion 9
1.2.5 Exercice de suivi (facultatif) 9
1.3 Références ou Ressources 9

1 Utilisation de capteurs IR dans les systèmes d'alarme

1.1 Informations générales

1.1.1 Description succincte

Cette leçon présente l'utilisation du capteur optique réfléchissant TCRT5000 avec sortie transistor pour concevoir un circuit simple qui agira comme une alarme pour les fenêtres qui auraient pu être ouvertes par une personne non autorisée. Le positionnement d'un tel capteur peut être introduit en configuration réelle à côté du cadre de fenêtre en raison de la plage de fonctionnement du capteur qui est de 0,2 mm à 15 mm.

1.1.2 Objectifs d'apprentissage

Les principaux objectifs d'apprentissage de ce plan de cours ou de cette activité éducative sont:

- Compréhension du concept et du contenu du matériel pour mettre en place un code dans Scratch.
- Familiarisation avec les activités pratiques de construction de consoles pour améliorer l'expérimentation dans des sujets liés aux STEM.
- Familiarisation avec les lectures des broches GPIO.
- Comprendre les étapes de codage dans Scratch.
- Conception du codage dans Scratch.
- Utilisation de capteurs, émetteur infrarouge, phototransistor et autres éléments pour expérimenter la console STEMKIT.
- Réalisation d'un codage sur une console STEMKIT sur l'exemple des capteurs IR dans les systèmes d'alarme.
- Réalisation de l'assemblage de base des capteurs sur une maquette.
- Expérimentation de la connexion de capteurs à la console Raspberry Pi et STEMKIT.
- Autonomie dans la création d'un circuit simple pouvant servir de démonstration d'une installation de sécurité dans un bâtiment.
- Autonomie dans l'introduction des concepts de codage dans l'environnement de la classe.

1.1.3 Liens avec le curriculum

Les domaines, sous-domaines, sujets / sujets auxquels ce plan de cours peut être lié sont:
Science (Physique / Chimie / Biologie / Géologie): tension, puissance, circuits, déclencheurs d'alarme, méthode scientifique, investigation, expérimentation, analyse et interprétation des résultats
Informatique / Informatique: unité de traitement et périphériques, interfaces, langage de programmation et structures principales, codage

Technologie: électronique, matériel et logiciel open source, capteurs, signal numérique, ordinateurs monocarte, console

1.1.4 Matériel requis

Afin de mener à bien ce plan de cours, la console STEMKIT avec Raspberry Pi est nécessaire avec les éléments suivants:

- 3 x capteurs TCRT5000
- 1 x buzzer avec générateur
- 2 x fils de cavalier femelle à femelle
- 5 x fils de raccordement mâle-femelle
- 3 résistances 10kΩ
- 3 résistances 330Ω
- 1 x planche à pain

1.1.5 Durée

La durée de ce plan de cours est estimée à environ 45 à 60 minutes, soit une heure de classe.

1.2 Plan de leçon

Le plan de leçon est divisé en quatre phases, qui sont l'introduction, la préparation, l'investigation et la conclusion. En guise de suivi, il y a aussi un exercice facultatif à la fin.

1.2.1 Introduction

Le TCRT5000 est un capteur composé de deux éléments dans un boîtier. L'émetteur émet la longueur d'onde de 950 nm qui doit ensuite être reçue par le phototransistor faisant office de détecteur. Plus l'objet est proche du capteur, plus la lecture de la tension du phototransistor sera élevée. Comme Raspberry Pi propose des broches GPIO, cette lecture peut ensuite être suivie pour vérifier si l'obstacle (le cadre de la fenêtre) est proche du capteur et par conséquent si la fenêtre est fermée.
Dans cette leçon, Scratch sera utilisé pour montrer l'exemple de code qui peut être utilisé pour suivre ce circuit simple.

1.2.2 Préparation

La phase de préparation nécessite de réaliser un assemblage de base des capteurs sur une maquette et la mise en place du code dans Scratch. Commençons par la maquette en premier.
Placez trois capteurs TCRT5000 sur une maquette vide en connectant l'émetteur du phototransistor et la cathode de l'émetteur infrarouge au rail de masse. Après cela, connectez le collecteur du phototransistor à l'aide d'une résistance de 10 kΩ au rail positif sur une maquette. De même, connectez l'anode de l'émetteur infrarouge au rail positif à l'aide d'une résistance de 330Ω. Répétez cette opération pour les deux autres capteurs TCRT5000.
Il est maintenant temps de connecter les capteurs au Raspberry Pi. Connectez le cavalier d'une longueur appropriée à chaque capteur TCRT5000. Il doit se connecter au rail où le collecteur du phototransistor est connecté et alimenté par un rail à tension positive utilisant une résistance de 10 kΩ. À ce stade, il devrait y avoir trois fils de liaison - un pour chaque capteur TCRT5000. Ces fils de cavalier doivent être connectés aux broches GPIO du Raspberry Pi marquées 35, 33 et 31 (ou en GPIO: 19, 13 et 6). Il est maintenant également temps de connecter le buzzer directement à Raspberry Pi. Vous pouvez le faire en attachant la broche GND du buzzer à la broche 39 et son fil positif à la broche 38 (ou GPIO: 26). Enfin, la maquette doit être alimentée par Raspberry Pi. Pour cela, vous pouvez utiliser un rail + 5V de Raspberry Pi (broche 4) et GND (broche 6). Utilisez des cavaliers pour alimenter la carte d'expérimentation. La configuration matérielle est terminée, nous pouvons donc maintenant passer à Scratch.
À l'intérieur de Scratch, vous pouvez sélectionner n'importe quelle toile de fond comportant au moins trois fenêtres visibles. Pour cette leçon, nous allons utiliser la toile de fond urban1. Réglez-le pour toute la scène. En même temps, vous devez ajouter trois objets qui changeront leurs costumes une fois l'alarme déclenchée. Ici, la suggestion est d'utiliser un sprite qui combine button4-a et button5-b. Le code changera les costumes en fonction de la lecture des capteurs. Comme vous avez besoin de trois sprites (un pour

chaque fenêtre), veuillez les dupliquer pour avoir la configuration suivante (le sprite du milieu a l'autre costume actif pour montrer la différence):

Image 1. Environnement Scratch avec tous les éléments requis positionnés sur l'écran
Source: Projet STEMKIT4Schools

Avant de passer au code principal, travaillons sur les trois sprites qui sont placés sur les fenêtres.

Pour simuler l'alarme, nous allons utiliser les messages de diffusion de Scratch pour réagir en conséquence aux lectures des capteurs. Lorsque vous basculez vers l'onglet Scripts, vous devrez ajouter deux réactions pour les messages entrants. Supposons que pour le premier capteur, nous émettrons des messages window1-open et window1-closed. Lorsque le message reçu est window1-open, nous devons déclencher l'alarme et changer le costume d'une coche verte à une croix rouge. De même, lorsque la fenêtre est fermée, nous devons à nouveau avoir le costume de coche verte. L'exemple de code est présenté ci-dessous:

Image 2. Script pour les sprites réagissant aux lectures des broches GPIO
Source: Projet STEMKIT4Schools

Répliquez cette configuration pour les deux sprites restants placés sur les fenêtres et n'oubliez pas de changer les titres des messages (window2-open, window2-closed, window3-open, window3-closed).

Le code principal est là où tout devient intéressant. Une fois que vous avez sélectionné le chat, passez à l'onglet Scripts et commencez à ajouter le code. La première chose que nous aimerions configurer est de faire savoir à Scratch que les broches GPIO 19, 13 et 6 doivent être lues comme des broches d'entrée.

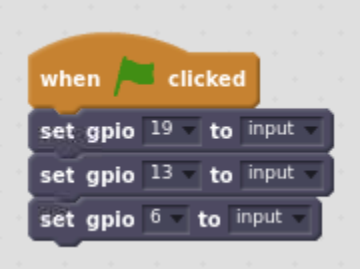

Image 3. Début du code - définition des broches GPIO pertinentes en tant que broches d'entrée
Source: Projet STEMKIT4Schools

Début du code - Définition des broches GPIO pertinentes en tant que broches d'entrée

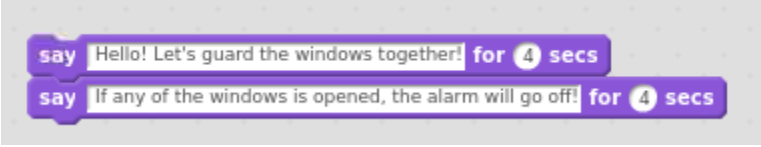

Image 4. Messages d'introduction exécutés par le code
Source: Projet STEMKIT4Schools

Dans l'étape suivante, nous allons ajouter une boucle de répétition qui fera 30 cycles, compris comme des «vérifications» des capteurs.

Image 5. Boucle de répétition principale
Source: Projet STEMKIT4Schools

Dans chaque boucle, nous sommes intéressés à atteindre les trois capteurs pour voir si les fenêtres sont fermées ou non. La structure est très simple, car nous pouvons vérifier avec une instruction if si la lecture des broches GPIO respectives est élevée et agir en conséquence. Vous trouverez ci-dessous un exemple de broche GPIO numéro 19.

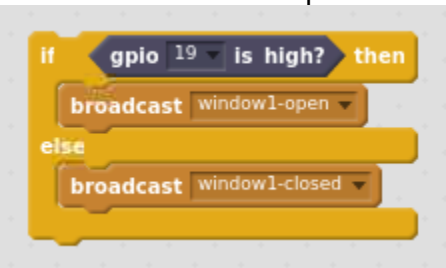

Image 6. Une instruction if lisant la valeur GPIO et diffusant des messages
Source: Projet STEMKIT4Schools

Notez que la seule action que nous prenons ici est d'émettre un message window1-open ou window1-closed en fonction de la valeur de la broche GPIO 19. Similaire si des blocs doivent être ajoutés pour le GPIO restant: 13 et 6. Encore une fois, ne le faites pas. oubliez de changer le numéro de pin GPIO et aussi le message à diffuser!
Dans l'étape suivante, nous voulons également contrôler notre buzzer. La condition ici est que si toutes les fenêtres sont fermées, l'alarme est silencieuse. Lorsqu'au moins une fenêtre est ouverte, nous émettrons le son d'alarme et laisserons également le chat dire Alarme! d'avoir également un avis visuel. Pour cela, nous utiliserons une instruction if avec trois conditions combinées par une instruction ou logique.

Image 7. Une instruction if pour déclencher l'alarme si nécessaire
Source: Projet STEMKIT4Schools

Juste avant de terminer la boucle en cours, nous mettrons en pause l'exécution pendant 1 seconde. Cela définit la fréquence de nos contrôles à exécuter une fois par seconde.

Image 8. Suspendre l'exécution du code pendant 1 seconde juste avant de quitter la boucle
Source: Projet STEMKIT4Schools

Enfin, comme nous ne voulons pas que le buzzer émette un son après avoir terminé tout notre code, juste après la boucle de répétition, nous devons ajouter l'instruction pour définir le numéro de broche GPIO 26 (celui avec le buzzer) pour rester dans le bas état (ce qui signifie pas de son).

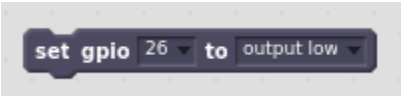

Image 9. Réglage du GPIO du buzzer sur faible après la boucle principale
Source: Projet STEMKIT4Schools

Le résultat final définissant notre code est présenté ci-dessous.

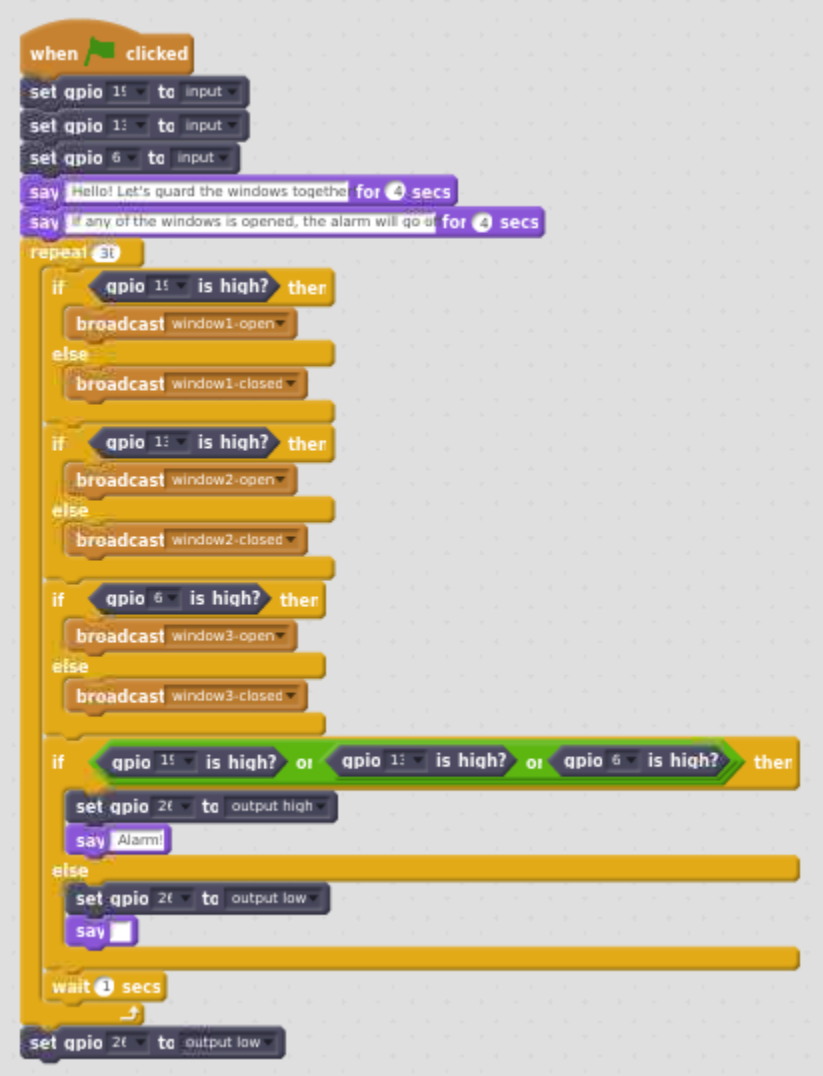

Image 10. Code principal du circuit d'échantillonnage
Source: Projet STEMKIT4Schools

1.2.3 Enquête

Nous pouvons enfin exécuter notre code! Veuillez suivre les sous-tâches suivantes pour en savoir plus sur cet exemple. Pour éviter de fausses lectures, assurez-vous que tous les éléments utilisés pour construire le circuit (fils de liaison ou résistances) ne recouvrent aucun des capteurs TCRT5000.
Collecte de données
Pour la première exécution, assurez-vous que les capteurs ne sont pas couverts, ou, en d'autres termes, qu'il n'y a pas d'éléments qui refléteraient le faisceau infrarouge émis et donc transformeraient les broches GPIO en état haut. Lors de la prochaine exécution, essayez de couvrir un capteur avec votre doigt ou tout autre élément. Essayez d'observer

le comportement lorsque les trois capteurs sont couverts, en simulant la situation où toutes les fenêtres sont fermées.
Analyse de données
Sur la base des données observées, êtes-vous en mesure de dire si la distance de l'obstacle au capteur est conforme à la plage de travail prévue? Essayez de remarquer à quelle distance l'obstacle doit être au-dessus du capteur pour le faire passer la broche GPIO à l'état haut. Le comportement du code est-il conforme aux attentes? Les sprites à l'écran changent-ils en réponse à des capteurs spécifiques couverts / découverts? L'alarme se déclenche-t-elle lorsqu'au moins une fenêtre est ouverte et le chat dit-il Alarme! dans cette situation?
Présentation des résultats
À ce stade, nous sommes invités à partager les résultats de nos travaux avec d'autres groupes. Est-ce que tout a bien fonctionné? Y a-t-il eu des difficultés à mettre en place l'ensemble du circuit? Des modifications ont-elles été apportées au code? Si oui, de quel genre? Les lectures sur la proximité de l'obstacle pour transformer le port GPIO du capteur en état haut étaient-elles cohérentes pour tous les capteurs? Était-ce également le même cas dans d'autres groupes?

1.2.4 Conclusion

Nous avons réussi à créer un circuit très simple pouvant servir de démonstration d'une installation de sécurité dans le bâtiment. À ce stade, nous pouvons échanger des idées avec d'autres groupes, ce qui a été fait de quelle manière et dans quel ordre et clarifier les questions qui pourraient apparaître.

1.2.5 Exercice de suivi (facultatif)

L'exercice de suivi peut inclure un multimètre, mesurant la tension des capteurs en mode direct à l'approche de l'obstacle. De cette façon, nous pourrons dire à quelle tension le Raspberry Pi considère que l'entrée est à l'état haut. Nous pouvons également essayer de changer la boucle principale utilisée dans le code de répétition à pour toujours.

1.3 Références ou Ressources

Ressource liée à ce plan de cours: https://www.vishay.com/docs/83760/tcrt5000.pdf