Rebah Mehdi
Jorge Alexandre
Lawniczak Corentin

IE3-00-4B

Compte-rendu du projet de SPI

Sujet n°2 – Roue de la fortune

I – Présentation du projet

A) Sujet et déroulement

L'objectif de ce projet est la création d'une « roue de la fortune » composée de :

- 1 ruban cuivré
- 1 roue en carton
- 1 moteur pas à pas
- 1 écran d'affichage
- 1 buzzer
- au moins 3 leds

Le déroulement doit être le suivant :

Un bouton permet la mise en route d'un moteur pas à pas entraînant une roue découpée en huit portions (dont deux gagnantes), le moteur s'arrête après une courte période. Un message est affiché sur l'écran et sur l'ordinateur afin de donner le résultat. En cas de victoire, un buzzer retentis et les leds s'éclairent.

B) Répartition du travail

Rebah Mehdi et Jorge Alexandre étaient principalement chargés de la mise en place du circuit et de la création du programme et Lawniczak Corentin était principalement chargé de la

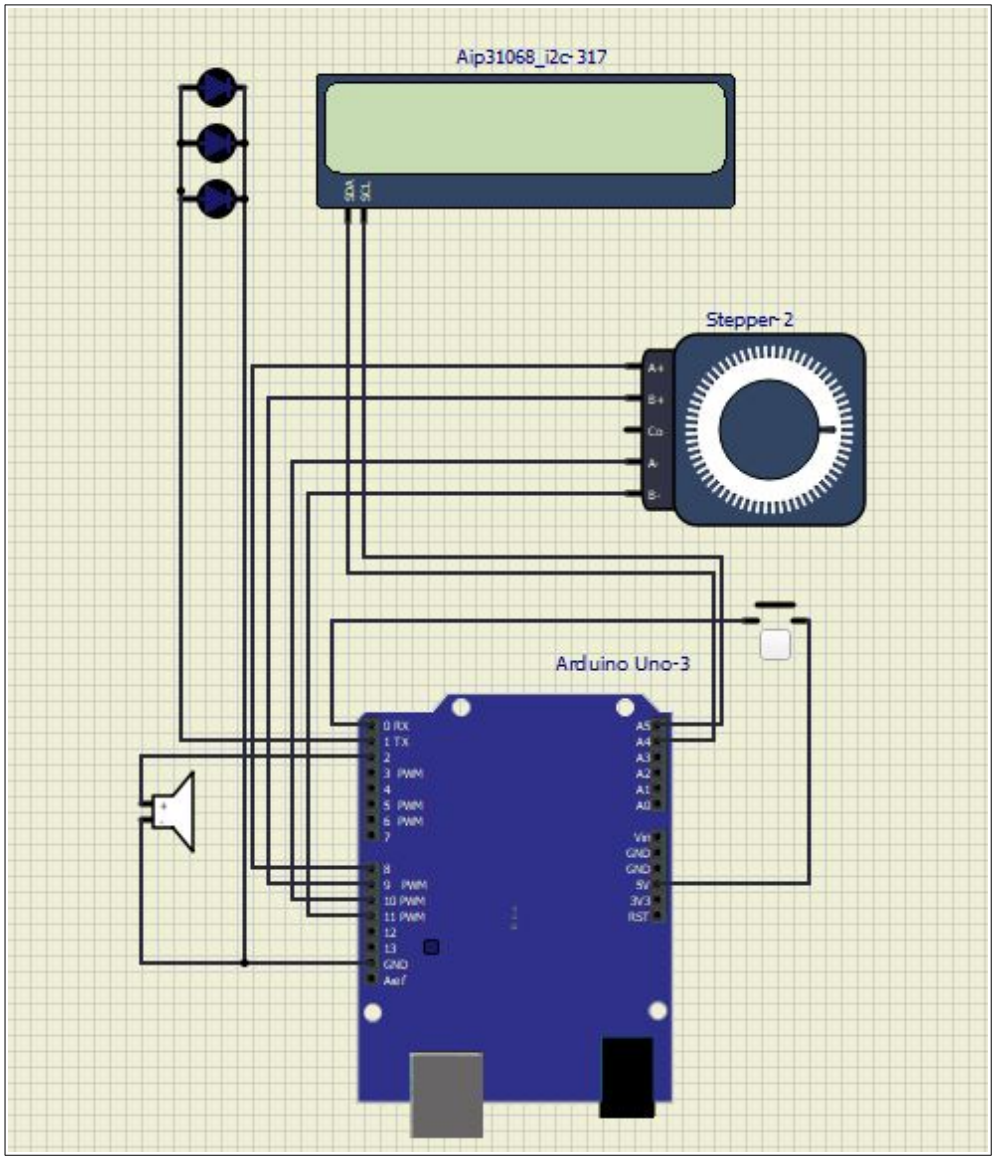
rédaction du compte-rendu et de la présentation.

II – Caractéristiques des capteurs et dispositifs

Le matériel utilisé pour la réalisation de ce projet est le suivant :

- 1 Arduino Uno : microcontrôleur possédant 14 broches d'entrée/sortie numérique et 6 entrées analogiques.
- 1 Moteur pas à pas : moteur continu dans laquelle la rotation est divisée en un certain nombre de pas (ici 64 pas)
- 1 écran LCD AIP31068_I2C : afficheur LCD 2x16 caractères rétro-éclairé se raccordant via le bus I2C.
- 1 buzzer
- 3 leds
- 1 bouton-poussoir

III – Schéma du circuit



IV – Algorithme

A) Initialisation des variables

On inclut la bibliothèque *EEPROM* afin de permettre d'utiliser la mémoire du microcontrôleur de la carte afin de conserver des valeurs lorsque la carte est éteinte et la bibliothèque *LiquidCrystal_AIP3168_I2C* afin de permettre à la carte Arduino d'utiliser l'écran LCD.

Ensuite, on déclare un écran LCD de 16 colonnes et 2 lignes et on initialise toutes les variables nécessaires à l'exécution du programme :

Type	Nom	Description
int	currentStep	Stocke la valeur de la position actuelle du moteur.
int	steps	Représente le nombre de pas nécessaires pour que le moteur fasse un tour complet.
int	pin1, pin2, pin3, pin4	Permettent de stocker les positions des broches connectés au moteur, elles contiennent respectivement les valeurs 8, 9, 10 et 11.
int	bouton	Permet de garder en mémoire la valeur lue sur la broche numérique connectée au bouton
int	aleatoire	Représente les valeurs tirées aléatoirement.
int	etat	Gardes en mémoire les différents états de l'algorithme.
int	victoire	Définis si l'utilisateur à remporté la partie.
int	p_execution	Permet de savoir si c'est la première exécution du programme
boolean	done	Permet de déterminer si quelque chose est affiché sur l'écran LCD et si les instructions en cas de victoire ou de défaite ont été exécutés.

B) La fonction *setup*()

On entre ensuite dans la fonction *setup*() qui sera appelée une seule fois au démarrage du programme et dans laquelle on configure la broche 0 comme étant une entrée et les broches 1, 2, 8, 9, 10 et 11 comme étant des sorties. Puis on initialise la communication avec le moniteur sur 9 600 bps.

Toujours dans la fonction *setup*() , on récupère la valeur contenue à l'adresse 8 008 de l'EEPROM et on la stocke dans la variable p_execution. Une condition vérifie si c'est la première exécution du programme (si la variable p_execution contient une autre valeur que 2 345) et si c'est le cas alors on stocke la valeur 2 345 dans l'EEPROM à l'adresse 8 008 et la valeur 0 à l'adresse 8 000.

Pour finir l'exécution de la fonction *setup*(), on récupère la valeur stockée à l'adresse 8 000 de l'EEPROM , on l'attribue à la variable currentstep , on initialise l'écran LCD grâce à sa fonction *init*() et on positionne son curseur à la position 0,0.

c) La fonction *loop*()

Ensuite on entre dans la fonction *loop*() qui sera lue continuellement jusqu'à l'extinction de la carte.

On commence par stocker dans la variable currentStep la valeur stockée à l'adresse 8 000 de l'EEPROM et on divise cette valeur par 16 afin qu'elle corresponde à une des parts de notre roue.

L'algorithme comporte 3 états :

L'état **0** permet l'initialisation ou la réinitialisation de la partie. Une condition vérifie si la roue est bien a sa position initiale et si ce n'est pas le cas, la fonction *step*() est appelée afin de repositionner celle-ci. Ensuite une autre condition vérifie l'état du bouton, lorsque l'on appuie dessus, on passe à l'état suivant.

L'état **1** va permettre de lancer la roue et de déterminer si on a gagné ou non. Un nombre aléatoire est tiré et détermine un nombre de pas a faire , compris entre 0 et 16 plus 10 à 20 tours de roue. La fonction *step*() exécute ce nombre de pas et une condition vérifie si le lancer est gagnant, si c'est le cas victoire prend la valeur 1. Dans tous les cas, on passe à l'état 2.

L'état **2** correspond à la fin de partie. Une condition vérifie si le lancer est gagnant et si c'est le cas, elle appelle la fonction *gagner*() dans le cas contraire, elle appelle la fonction *perdu*(). Ensuite, on vérifie l'état du bouton, si on appuie dessus, le booléen done prend la valeur false, on efface ce qui est affiché à l'écran grâce à la fonction *clear*() et on passe à l'état 0.

d) les fonctions

La première fonction est la fonction *step*() , elle ne retourne rien et prend en paramètres un entier x. Elle permet de répondre au problèmes de précision rencontrés lors des essais de différentes fonctions de librairies pour les moteurs pas à pas. Elle permet d'avancer du nombre pas passé en paramètres en activant les unes après les autres les différentes broches connectées au moteur. De plus, elle incrémente la valeur contenue dans la variable currentstep.

La seconde fonction est la fonction *gagner*() , elle ne retourne rien et ne prend aucuns paramètres. Elle passe d'abord le booléen done sur la valeur true et affiche un message de victoire sur le moniteur. Puis elle alimente les broches où sont connectés les leds et le buzzer , elle affiche le message de victoire sur l'écran LCD et après un délai, arrête d'alimenter les broches liées aux leds et au buzzer.

La fonction suivante est la fonction *perdu*() , elle ne retourne rien et ne prend aucuns paramètres. Elle exécute les mêmes actions que la fonction *gagner*(), mais affiche un message de défaite sur le moniteur et l'écran LCD et n'alimente pas les broches liées au buzzer et aux leds.

La dernière fonction est la fonction *generate_seed*(), elle retourne une variable non signée de type long et ne prend pas de paramètres. On initialise la variable seed à 0 puis dans une boucle for, on fait un **ET** logique entre la valeur retournée par un port analogique vide et la valeur 1. Puis on fait un **OU** logique entre cette valeur et notre seed décalé d'un bit sur la gauche à chaque itération. Ce qui donne une seed aléatoire et donc un nombre aléatoire.

VI – Déroulement

Si la roue n'est pas placée sur sa position par défaut alors celle-ci se replace, ensuite un appui sur le bouton permet de lancer la roue, celle-ci effectue 10 à 20 tours et se positionne aléatoirement. En cas de victoire, un message est affiché sur l'écran LCD et sur le moniteur, les leds s'allument et le buzzer retentis pendant un court délai. En cas de défaite, un message est affiché sur l'écran LCD et sur le moniteur. Un nouvel appui sur le bouton permet de réinitialiser le lancer.