Heilmann G4

Hugo

Compte-rendu séance du 19/01/2023

Suite à la soutenance, je suis allé chercher les pièces d’échec qui ont été imprimé et nous nous sommes malheureusement rendu compte que nous avons fait une erreur de mesure des aimants et que ceux-ci sont trop gros pour rentrer dans les trous présents sous les pièces. Nous allons donc de voir limer l’intérieur des pièces afin de permettre aux aimants de rentrer car la marge manquante est extrêmement faible.



Suite à cela, j’ai passé le reste de la séance à essayer d’arranger le code et à la débuguer.

Voici les quelques lignes de code qui sont complètement nouvelles ou qui ont subi de lourdes modifications :

// Valeur des coordonnées des cases à déplacer et à atteindre

int target\_line=0;

int target\_column=0;

int wanted\_line=0;

int wanted\_column=0;

int jeu[10][10] = { // Instance du tableau de jeu

{-1,-1,-1,-1,-1,-1,-1,-1,-1,-1}, // Ligne "0" zone morte=-1

{-1,20,30,40,50,60,40,30,20,-1}, // Ligne 1 pion=1

{-1,10,10,10,10,10,10,10,10,-1}, // Ligne 2 tour=2

{-1,0,0,0,0,0,0,0,0,-1}, // Ligne 3 cavalier=3

{-1,0,0,0,0,0,0,0,0,-1}, // Ligne 4 fou=4

{-1,0,0,0,0,0,0,0,0,-1}, // Ligne 5 reine=5

{-1,0,0,0,0,0,0,0,0,-1}, // Ligne 6 roi=6

{-1,1,1,1,1,1,1,1,1,-1}, // Ligne 7 vide=0

{-1,2,3,4,5,6,4,3,2,-1}, // Ligne 8 piece noire=piece blanche\*10

{-1,-1,-1,-1,-1,-1,-1,-1,-1,-1}, // Ligne "0bis" tout le plateau est entouré de cases mortes

};

void loop() {while(true){

// Afficher la couleur du joueur actuel

if (white\_turn) {

Serial.println("White turn");

} else {

Serial.println("Black turn");

}

// Demander la ligne de la pièce à déplacer;

Serial.print("Ligne de la pièce à déplacer ? ");

if (Serial.read()!= ""){

boolean verified=false;

for (int t=0;t<8;t++) {

if (Serial.read()==line\_possibilities[t]){

verified=true;

}

}

if (verified){

target\_line=Serial.read();

Serial.println(target\_line);

} else {

Serial.println("Inexistant");

break;

}

}

// Demander la colonne de la pièce à déplacer;

Serial.print("Colonne de la pièce à déplacer ? ");

if (Serial.read()!= ""){

boolean verified=false;

for (int t=0;t<8;t++) {

if (Serial.read()==transcribe(column\_possibilities[t])){

verified=true;

}

}

if (verified){

target\_column=transcribe(String(Serial.read()));

Serial.println(target\_column);

} else {

Serial.println("Inexistant");

break;

}

}

// Vérifier que la case ne soit pas vide

if (jeu[target\_line][target\_column]==0){

Serial.println("There is no one here");

break;

}

// Vérifier que la couleur de la pièce à déplacer soit raccord avec la couleur de la personne dont c'est le tour

if ((white\_turn==true) and (jeu[target\_line][target\_column]>=10)){

Serial.println("This is not your color.");

break;

}

if ((white\_turn==false) and (jeu[target\_line][target\_column]<10)){

Serial.println("This is not your color.");

break;

}

// Demander la ligne de la case à atteindre

Serial.print("Ligne de la case à atteindre ? ");

if (Serial.read()!= ""){

boolean verified=false;

for (int t=0;t<8;t++) {

if (Serial.read()==line\_possibilities[t]){

verified=true;

}

}

if (verified){

wanted\_line=Serial.read();

Serial.println(wanted\_line);

} else {

Serial.println("Inexistant");

break;

}

}

// Demander la colonne de la case à atteindre

Serial.print("Colonne de la case à atteindre ? ");

if (Serial.read()!= ""){

boolean verified=false;

for (int t=0;t<8;t++) {

if (Serial.read()==transcribe(column\_possibilities[t])){

verified=true;

}

}

if (verified){

wanted\_column=transcribe(String(Serial.read()));

Serial.println(wanted\_column);

} else {

Serial.println("Inexistant");

break;

}

}

// avancer/reculer l'électroaimant de ax colonnes

void avancer\_caseX(int ax) {

int nbPas=ax\*50; // Demi-case environ 50px ?

for (int i=0;i<nbPas;i++) {

digitalWrite(moteurX,LOW); // on allume le moteur ligne pendant nbPas

}

digitalWrite(moteurX,HIGH); // on éteint le moteur ligne

posx=posx+ax; // on actualise la nouvelle ligne sur laquelle se trouve l'électroaimant

}

// avancer/reculer l'électroaimant de ay lignes

void avancer\_caseY(int ay) {

int nbPas=ay\*50; // Demi-case environ 50px ?

for (int i=0;i<nbPas;i++) {

digitalWrite(moteurY,LOW); // on allume le moteur colonne pendant nbPas

}

digitalWrite(moteurY,HIGH); // on éteint le moteur colonne

posy=posy+ay; // on actualise la nouvelle colonne sur laquelle se trouve l'électroaimant

}

// Déplace l'électroaimant

void aller\_case(int wantedX, int wantedY) {

int ecartX=wantedX-posx; // Calcule la distance entre la ligne de départ et d'arrivée

int ecartY=wantedY-posy; // Calcule la distance entre la colonne de départ et d'arrivée

avancer\_caseX(ecartX); // déplace l'aimant selon la distance x calculée

avancer\_caseY(ecartY); // déplace l'aimant selon la distance y calculée

posx+=ecartX; // Modifie la position x de l'électroaimant

posy+=ecartY; // Modifie la position y de l'électroaimant

}

// gérer ici l'endroit où sont envoyés les pièces mortes

void dead(int i, int j) {

for (int k=1;k<=10;k++){ // on cherche sur chaque ligne

for (int l=1;l<=10;l++){ // sur chaque colonne de chaque ligne s'il y a une zone morte disponible

if (jeu[k][l]==-1){ // dès qu'on en trouve une :

deplacer\_piece(i,j,k,l); // on y déplace la pièce

break; // on arrête la recherche

}

}

}

}

// Transforme une lettre en nombre

int transcribe(String f){

if (f=="A"){

return 1;

}

if (f=="B"){

return 2;

}

if (f=="C"){

return 3;

}

if (f=="D"){

return 4;

}

if (f=="E"){

return 5;

}

if (f=="F"){

return 6;

}

if (f=="G"){

return 7;

}

if (f=="H"){

return 8;

}

}