```
01 //ARDUINO Variables Entiers
02 int commande_ouverture = 2; // Déclaration broche Bouton ouverture
03 int commande_fermeture = 3; // Déclaration broche Bouton fermeture
04 int fin_de_course_ouvert = 4; // Capteur fin de course haut
05 int fin_de_course_ferme = 5; // Capteur fin de course bas
06 int MotorPin1 = 6;  // Déclaration broche IN1 L293D
07 int MotorPin2 = 7;  // Déclaration broche IN2 L293D
08 int MotorPin3 = 8; // Déclaration broche IN3 L293D
09 int MotorPin4 = 9; // Déclaration broche IN4 L293D
10 int LuminositePin = A0;  // Déclaration broche LDR
11 int Luminosite = 0; // Variable de la luminosité
12 int Tour = 0;
13 int delayTime = 25; // Temps entre chaque pas 25ms
14 int Seuil_Jour = 400; // Variable de luminosité seuil pour le jour
15 int Seuil_Nuit = 250; // Variable de luminosité seuil pour la nuit
16 int Tempo_luminosite = 20000; // Temporisation luminosité 20 secondes = 20000ms
17 //ARDUINO Variables booléennes
18 boolean porte_fermee = false; //Déclaration variable porte fermée
19 boolean porte_ouverte = false; //Déclaration variable porte ouverte
20 boolean fdc_ferme = false; // Déclaration variable Fin de Course Haut
21 boolean fdc_ouvert = false; // Déclaration variable Fin de Course Bas
22 boolean etat_bp_h = false, etat_bp_b = false; // Déclaration des variables bas et haut
23 boolean mem_h = false, mem_b = false, mem_fdc_ferme = false,  mem_fdc_ouvert = false; // Déclaration des mémoires
24 boolean mem_mouvement = false; // Déclaration de la mémoire mouvement
25 boolean mem_lumiere = false; // Déclaration de la mémoire lumière
26 boolean mem_init = false; // Déclaration de la mémoire initialisation
27 boolean Detecte_lumiere = false; // Déclaration variable détection lumière
28 boolean Jour = true; // Déclaration variable Jour = 1 | Nuit = 0
29 boolean Initialisation = false; // Déclaration variable initialisation
30 boolean tempoActive = false; // État d'activation de la tempo
31 //ARDUINO Variables numériques
32 unsigned long tempoDepart = 0; // Temps à l'activation de la tempo
33
34 void setup() {
35   Serial.begin(9600); // Ouverture du port série et debit de communication fixé à 9600 bauds
36   pinMode(commande_ouverture, INPUT_PULLUP); // Déclaration entrée pull-up sur entrée BP haut
37   pinMode(commande_fermeture, INPUT_PULLUP); // Déclaration entrée pull-up sur entrée BP bas
38   pinMode(fin_de_course_ferme, INPUT_PULLUP); // Déclaration entrée pull-up sur entrée Fin de course haut
39   pinMode(fin_de_course_ouvert, INPUT_PULLUP); // Déclaration entrée pull-up sur entrée Fin de course bas
40   pinMode(MotorPin1, OUTPUT);
41   pinMode(MotorPin2, OUTPUT);
42   pinMode(MotorPin3, OUTPUT);
43   pinMode(MotorPin4, OUTPUT);
44 }
45
46 void Lance_initialisation() {
47   Fermer_porte_Initialisation();
48 }
49
50 void loop() {
51   Luminosite = analogRead(LuminositePin);
52   if (Initialisation) {
53     if (Luminosite >= Seuil_Jour)
54     {
55       Detecte_lumiere = true;
56     }
```

```
  if (Luminosite <= Seuil_Nuit)
  {
    Detecte_lumiere = false;
  }
  if (Detecte_lumiere != mem_lumiere) {
    tempoActive = true;
    tempoDepart = millis();
    Serial.println("La lumière lance tempo"); // Affichage sur le moniteur série du texte
  }
  if (Detecte_lumiere && tempoActive && ((millis() - tempoDepart) >= Tempo_luminosite))
  {
    Jour = true;
    Serial.println("Il fait jour"); // Affichage sur le moniteur série du texte
    tempoActive = false;
  if (!fdc_ouvert  && !porte_ouverte) {
    Ouvrir_porte();
    }
  }
  mem_lumiere = Detecte_lumiere;

  if (!Detecte_lumiere && tempoActive && (millis() - tempoDepart) >= Tempo_luminosite)
  {
    Jour = false;
    Serial.println("Il fait nuit"); // Affichage sur le moniteur série du texte
    tempoActive = false;
    if (!fdc_ferme  && !porte_fermee){
    Fermer_porte();
  }}
  mem_lumiere = Detecte_lumiere;
 }

//Lecture fins de course et boutons poussoir
 etat_bp_h = !digitalRead(commande_ouverture); // Inverse de la lecture sur entrée BP haut
 etat_bp_b = !digitalRead(commande_fermeture); // Inverse de la lecture sur entrée BP bas
 fdc_ferme = !digitalRead(fin_de_course_ferme); // Inverse de la lecture sur entrée Fin de course haut
 fdc_ouvert = !digitalRead(fin_de_course_ouvert); // Inverse de la lecture sur entrée Fin de course bas

 if (fdc_ferme != mem_fdc_ferme) // Changement d'état du fin de course haut (front montant ou descendant)
 {
  if (fdc_ferme)
  {
    Serial.println("Porte fermée !"); // Affichage sur le moniteur série du texte
  }
  if (!fdc_ferme)
  {
    Serial.println("Porte non fermée"); // Affichage sur le moniteur série du texte
  }
 }
 mem_fdc_ferme = fdc_ferme; // Mémorisation du nouvel état du fin de course haut

 if (fdc_ouvert != mem_fdc_ouvert) // Changement d'état du fin de course bas (front montant ou descendant)
 {
  if (fdc_ouvert) // Appui sur BP haut mais pas sur le bas
  {
    Serial.println("Porte ouverte !"); // Affichage sur le moniteur série du texte
  }
```

```
113     if (!fdc_ouvert)
114     {
115       Serial.println("Porte non ouverte"); // Affichage sur le moniteur série du texte
116     }
117   }
118   mem_fdc_ouvert = fdc_ouvert; // Mémorisation du nouvel état du fin de course bas
119
120
121   if (etat_bp_h != mem_h) // Changement d'état du bouton poussoir haut (front montant ou descendant)
122   {
123     Serial.println("Appui BP Haut"); // Affichage sur le moniteur série du texte
124     if (etat_bp_h && !etat_bp_b && !fdc_ferme && !porte_fermee) // Appui sur BP haut mais pas sur le bas
125     {
126       Fermer_porte(); // Lancer la fonction sens normal
127     }
128   }
129   mem_h = etat_bp_h;  // Mémorisation du nouvel état du bouton haut
130   if (etat_bp_b != mem_b) // Changement d'état du bouton poussoir bas (front montant ou descendant)
131   {
132     if (etat_bp_b && !etat_bp_h && !fdc_ouvert && !porte_ouverte) // Appui sur BP bas mais pas sur le haut
133     {
134       if (!fdc_ouvert) {
135         Ouvrir_porte();
136       }
137     }
138   }
139   mem_b = etat_bp_b; // Mémorisation du nouvel état du bouton bas
140 }
141
142 void Fermer_porte_Initialisation() {
143       delay(5000);
144   while (!fdc_ferme) {
145       digitalWrite(MotorPin1, HIGH);
146       digitalWrite(MotorPin2, HIGH);
147       digitalWrite(MotorPin3, LOW);
148       digitalWrite(MotorPin4, LOW);
149       delay(delayTime);
150
151       digitalWrite(MotorPin1, LOW);
152       digitalWrite(MotorPin2, HIGH);
153       digitalWrite(MotorPin3, HIGH);
154       digitalWrite(MotorPin4, LOW);
155       delay(delayTime);
156
157       digitalWrite(MotorPin1, LOW);
158       digitalWrite(MotorPin2, LOW);
159       digitalWrite(MotorPin3, HIGH);
160       digitalWrite(MotorPin4, HIGH);
161       delay(delayTime);
162
163       digitalWrite(MotorPin1, HIGH);
164       digitalWrite(MotorPin2, LOW);
165       digitalWrite(MotorPin3, LOW);
166       digitalWrite(MotorPin4, HIGH);
167       delay(delayTime);
168       Serial.println("Fermer porte"); // Affichage sur le moniteur série du texte
```

```
169
170       fdc_ferme = !digitalRead(fin_de_course_ferme);
171       etat_bp_b = !digitalRead(commande_fermeture); // Inverse de la lecture sur entrée BP bas
172       if (fdc_ferme)
173       {
174         Serial.println("Porte en haut"); // Affichage sur le moniteur série du texte
175         Arret();
176         porte_fermee = true;
177         delay(2000);
178         porte_ouverte = false;
179         Initialisation = true;
180         break;
181       }
182     }
183   }
184
185   void Fermer_porte() {
186     while (!fdc_ferme) {
187       digitalWrite(MotorPin1, HIGH);
188       digitalWrite(MotorPin2, HIGH);
189       digitalWrite(MotorPin3, LOW);
190       digitalWrite(MotorPin4, LOW);
191       delay(delayTime);
192
193       digitalWrite(MotorPin1, LOW);
194       digitalWrite(MotorPin2, HIGH);
195       digitalWrite(MotorPin3, HIGH);
196       digitalWrite(MotorPin4, LOW);
197        delay(delayTime);
198
199       digitalWrite(MotorPin1, LOW);
200       digitalWrite(MotorPin2, LOW);
201       digitalWrite(MotorPin3, HIGH);
202       digitalWrite(MotorPin4, HIGH);
203       delay(delayTime);
204
205       digitalWrite(MotorPin1, HIGH);
206       digitalWrite(MotorPin2, LOW);
207       digitalWrite(MotorPin3, LOW);
208       digitalWrite(MotorPin4, HIGH);
209       delay(delayTime);
210       Serial.println("Fermer porte"); // Affichage sur le moniteur série du texte
211
212       fdc_ferme = !digitalRead(fin_de_course_ferme);
213       etat_bp_b = !digitalRead(commande_fermeture); // Inverse de la lecture sur entrée BP bas
214       if (fdc_ferme || etat_bp_b)
215       {
216         porte_fermee = true;
217         porte_ouverte = false;
218         Serial.println("Porte fermée"); // Affichage sur le moniteur série du texte
219         Arret();
220         break;
221       }
222     }
223   }
224
```

```
225 void Ouvrir_porte() {//Tour < 280
226  while (!fdc_ouvert) {//Tour < 280
227    digitalWrite(MotorPin1, LOW);
228    digitalWrite(MotorPin2, LOW);
229    digitalWrite(MotorPin3, HIGH);
230    digitalWrite(MotorPin4, HIGH);
231    delay(delayTime);
232
233    digitalWrite(MotorPin1, LOW);
234    digitalWrite(MotorPin2, HIGH);
235    digitalWrite(MotorPin3, HIGH);
236    digitalWrite(MotorPin4, LOW);
237    delay(delayTime);
238
239    digitalWrite(MotorPin1, HIGH);
240    digitalWrite(MotorPin2, HIGH);
241    digitalWrite(MotorPin3, LOW);
242    digitalWrite(MotorPin4, LOW);
243    delay(delayTime);
244
245    digitalWrite(MotorPin1, HIGH);
246    digitalWrite(MotorPin2, LOW);
247    digitalWrite(MotorPin3, LOW);
248    digitalWrite(MotorPin4, HIGH);
249    delay(delayTime);
250     Serial.println("Ouvrir porte"); // Affichage sur le moniteur série du texte
251
252     fdc_ouvert = !digitalRead(fin_de_course_ouvert);
253     etat_bp_h = !digitalRead(commande_ouverture); // Inverse de la lecture sur entrée BP haut
254     if (fdc_ouvert || etat_bp_h)
255     {
256       porte_fermee = false;
257       porte_ouverte = true;
258       Serial.println("Porte ouverte"); // Affichage sur le moniteur série du texte
259       Arret();
260       break;
261     }
262    }
263 }
264
265 void Arret() {
266  digitalWrite(MotorPin1, LOW);
267  digitalWrite(MotorPin2, LOW);
268  digitalWrite(MotorPin3, LOW);
269  digitalWrite(MotorPin4, LOW);
270  tempoActive = 0;
271 }
```