```
//ETHERNET
#include <SPI.h> // Bibliothèqe pour SPI
#include <Ethernet.h> // Bibliothèque pour Ethernet
byte mac[] = {0x90, 0xA2, 0xDA, 0x0F, 0xDF, 0xAB}; // Adresse mac du Shield Ethernet
byte ip[] = {192, 168, 1, 40}; // Adresse IP réseau Local
EthernetServer server(80); // Déclare l'objet serveur au port d'écoute 80
unsigned long int refresh_num = 0; // Nombre de rafraîchissement de la page
String readString;
int Commande_ETH[4]={0, 1, 2, 3};
//ARDUINO Variables Entiers
int commande_ouverture = 2; // Déclaration broche Bouton ouverture
int commande_fermeture = 3; // Déclaration broche Bouton fermeture
int fin_de_course_ouvert = 4; // Capteur fin de course haut
int fin_de_course_ferme = 5; // Capteur fin de course bas
int MotorPin1 = 6;  // Déclaration broche IN1 L293D
int MotorPin2 = 7;  // Déclaration broche IN2 L293D
int MotorPin3 = 8; // Déclaration broche IN3 L293D
int MotorPin4 = 9; // Déclaration broche IN4 L293D
int LuminositePin = A0;  // Déclaration broche LDR
int Luminosite = 0; // Variable de la luminosité
int Tour = 0;
int delayTime = 25; // Temps entre chaque pas 25ms
int Seuil_Jour = 400; // Variable de luminosité seuil pour le jour
int Seuil_Nuit = 250; // Variable de luminosité seuil pour la nuit
int Tempo_luminosite = 20000; // Temporisation luminosité 20 secondes = 20000ms
//ARDUINO Variables booléennes
boolean porte_fermee = false; //Déclaration variable porte fermée
boolean porte_ouverte = false; //Déclaration variable porte ouverte
boolean fdc_ferme = false; // Déclaration variable Fin de Course Haut
boolean fdc_ouvert = false; // Déclaration variable Fin de Course Bas
boolean etat_bp_h = false, etat_bp_b = false; // Déclaration des variables bas et haut
boolean mem_h = false, mem_b = false, mem_fdc_ferme = false,  mem_fdc_ouvert = false; // Déclaration des mémoires
boolean mem_mouvement = false; // Déclaration de la mémoire mouvement
boolean mem_lumiere = false; // Déclaration de la mémoire lumière
boolean mem_init = false; // Déclaration de la mémoire initialisation
boolean Detecte_lumiere = false; // Déclaration variable détection lumière
boolean Jour = true; // Déclaration variable Jour = 1 | Nuit = 0
boolean Initialisation = false; // Déclaration variable initialisation
boolean tempoActive = false; // État d'activation de la tempo
//ARDUINO Variables numériques
unsigned long tempoDepart = 0; // Temps à l'activation de la tempo

void setup() {
  Serial.begin(9600); // Ouverture du port série et debit de communication fixé à 9600 bauds
  pinMode(commande_ouverture, INPUT_PULLUP); // Déclaration entrée pull-up sur entrée BP haut
  pinMode(commande_fermeture, INPUT_PULLUP); // Déclaration entrée pull-up sur entrée BP bas
  pinMode(fin_de_course_ferme, INPUT_PULLUP); // Déclaration entrée pull-up sur entrée Fin de course haut
  pinMode(fin_de_course_ouvert, INPUT_PULLUP); // Déclaration entrée pull-up sur entrée Fin de course bas
  pinMode(MotorPin1, OUTPUT);
  pinMode(MotorPin2, OUTPUT);
  pinMode(MotorPin3, OUTPUT);
  pinMode(MotorPin4, OUTPUT);
  Lance_initialisation();
  //ETHERNET
  Ethernet.begin (mac, ip); // Initialisation de la communication Ethernet
  Serial.print("\nLe serveur est sur l'adresse : ");
```

```
  Serial.println(Ethernet.localIP());  // On affiche l'adresse IP de la connexion
  server.begin(); // Démarrage de l'écoute
}

void Lance_initialisation() {
 Fermer_porte_Initialisation();
}

void loop() {
 //ETHERNET
 EthernetClient client = server.available();
 if (client) {
  while (client.connected()) {
   if (client.available()) {
    char c = client.read();
    if (readString.length() < 100) {
     readString += c;
    }
    //Vérifier si la requête HTTP est terminée
    if (c == '\n') {
     //Vérification réception chaîne
     Serial.println(readString);
     processCommand();
     generatePage(client);
     delay(1);
     //Stop client
     client.stop();
     //Effacement de la chaîne pour prochaine lecture
     readString = "";
    }
   }
  }
 }

 Luminosite = analogRead(LuminositePin);
 if (Initialisation) {
  if (Luminosite >= Seuil_Jour)
  {
   Detecte_lumiere = true;
  }
  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
   }
   if (!fdc_ouvert)
   {
    Serial.println("Porte non ouverte"); // Affichage sur le moniteur série du texte
   }
  }
  mem_fdc_ouvert = fdc_ouvert; // Mémorisation du nouvel état du fin de course bas


  if (etat_bp_h != mem_h) // Changement d'état du bouton poussoir haut (front montant ou descendant)
  {
   Serial.println("Appui BP Haut"); // Affichage sur le moniteur série du texte
   if (etat_bp_h && !etat_bp_b && !fdc_ferme && !porte_fermee) // Appui sur BP haut mais pas sur le bas
   {
    Fermer_porte(); // Lancer la fonction sens normal
   }
```

```
  }
  mem_h = etat_bp_h;  // Mémorisation du nouvel état du bouton haut
  if (etat_bp_b != mem_b) // Changement d'état du bouton poussoir bas (front montant ou descendant)
  {
    if (etat_bp_b && !etat_bp_h && !fdc_ouvert && !porte_ouverte) // Appui sur BP bas mais pas sur le haut
    {
      if (!fdc_ouvert) {
        Ouvrir_porte();
      }
    }
  }
  mem_b = etat_bp_b; // Mémorisation du nouvel état du bouton bas
}


void processCommand() {
 if (readString.indexOf("1") > 0)
 {
   if (readString.indexOf("?On") > 0) {
  Fermer_porte();
   }
 }
 if (readString.indexOf("2") > 0)
 {
   if (readString.indexOf("?On") > 0) {
  Ouvrir_porte();
   }
 }

 if (readString.indexOf("3") > 0)
 {
   if (readString.indexOf("?On") > 0) {
    Initialisation = false;
    Detecte_lumiere = false;
    Fermer_porte_Initialisation();
   }
 }
}

void generatePage(EthernetClient client) {
 if (refresh_num == 0) {
   //Header de la page HTML
   client.println(F("HTTP/1.1 200 OK"));
   client.println(F("Content-Type: text/html"));
   client.println();
 } else {
   //header
   client.print(F("<html><head><title>Arduino</title>"));
   client.print(F("<meta http-equiv='refresh' content=''>"));
   client.print(F("</head><body bgcolor='black'><br>"));
   client.print(F("<h1 style='color:green;font-size:28px;'  ><u>Etat de la porte & commandes</u></h1>"));
   client.println("<p style='color:white;'>");
   client.print(F("<br>Nombre d'actualisation de la page : "));
   client.print(refresh_num); //Nombre d'actualisation
   client.print("<br><br>");
//Détection porte ouverte (commande de fermeture)
```

```
  if(fdc_ouvert && !fdc_ferme) {
    client.print(F("<h2 style='color:red;font-size:26px;'><b>Porte Ouverte</b></h2>"));
    client.print(F("<br>"));
    client.print(F("<input style='font-size:24px;' type=button value=Fermer onmousedown=location.href='/?On"));
    client.print(Commande_ETH[1]);
    client.print(F(";'>"));
    client.print(F("<br>"));
  }
//Détection porte fermée (commande de ouverture)
  if(fdc_ferme && !fdc_ouvert) {
    client.print(F("<h2 style='color:red;font-size:26px;'><b>Porte Fermee</b></h2>"));
    client.print(F("<br>"));
    client.print(F("<input style='font-size:24px;' type=button value=Ouvrir onmousedown=location.href='/?On"));
    client.print(Commande_ETH[2]);
    client.print(F(";'>"));
    client.print(F("<br>"));
  }
//La porte n'est ni ouverte ni fermée (position inconnue = Initialisation ou fermeture ou ouverture)
  if(!fdc_ouvert && !fdc_ferme) {
    client.print(F("<h2 style='color:red;font-size:26px;'><b>Position porte inconnue</b></h2>"));
    client.print(F("<br>"));
    client.print(F("<input style='font-size:24px;' type=button value=Ouvrir onmousedown=location.href='/?On"));
    client.print(Commande_ETH[2]);
    client.print(F(";'>"));
    client.print(F("<br>"));
    client.print(F("<input style='font-size:24px;' type=button value=Initialisation onmousedown=location.href='/?On"));
    client.print(Commande_ETH[3]);
    client.print(F(";'>"));
    client.print(F("<br>"));
    client.print(F("<input style='font-size:24px;' type=button value=Fermer onmousedown=location.href='/?On"));
    client.print(Commande_ETH[1]);
    client.print(F(";'>"));
    client.print(F("<br>"));
  }
    client.print(F("</p><br>"));
    client.print(F("<input font size='6' type=button value=Rafraichir onmousedown=location.href='"));
    client.print(F("'>"));
    client.print(F("<br></body></html>"));
 }
 refresh_num = refresh_num + 1;
}

void Fermer_porte_Initialisation() {
     delay(5000);
  while (!fdc_ferme) {
     digitalWrite(MotorPin1, HIGH);
     digitalWrite(MotorPin2, HIGH);
     digitalWrite(MotorPin3, LOW);
     digitalWrite(MotorPin4, LOW);
     delay(delayTime);

     digitalWrite(MotorPin1, LOW);
     digitalWrite(MotorPin2, HIGH);
     digitalWrite(MotorPin3, HIGH);
     digitalWrite(MotorPin4, LOW);
     delay(delayTime);
```

```
281
282       digitalWrite(MotorPin1, LOW);
283       digitalWrite(MotorPin2, LOW);
284       digitalWrite(MotorPin3, HIGH);
285       digitalWrite(MotorPin4, HIGH);
286       delay(delayTime);
287
288       digitalWrite(MotorPin1, HIGH);
289       digitalWrite(MotorPin2, LOW);
290       digitalWrite(MotorPin3, LOW);
291       digitalWrite(MotorPin4, HIGH);
292       delay(delayTime);
293       Serial.println("Fermer porte"); // Affichage sur le moniteur série du texte
294
295       fdc_ferme = !digitalRead(fin_de_course_ferme);
296       etat_bp_b = !digitalRead(commande_fermeture); // Inverse de la lecture sur entrée BP bas
297       if (fdc_ferme)
298       {
299         Serial.println("Porte en haut"); // Affichage sur le moniteur série du texte
300         Arret();
301         porte_fermee = true;
302         delay(2000);
303         porte_ouverte = false;
304         Initialisation = true;
305         break;
306       }
307     }
308   }
309
310   void Fermer_porte() {
311     while (!fdc_ferme) {
312       digitalWrite(MotorPin1, HIGH);
313       digitalWrite(MotorPin2, HIGH);
314       digitalWrite(MotorPin3, LOW);
315       digitalWrite(MotorPin4, LOW);
316       delay(delayTime);
317
318       digitalWrite(MotorPin1, LOW);
319       digitalWrite(MotorPin2, HIGH);
320       digitalWrite(MotorPin3, HIGH);
321       digitalWrite(MotorPin4, LOW);
322        delay(delayTime);
323
324       digitalWrite(MotorPin1, LOW);
325       digitalWrite(MotorPin2, LOW);
326       digitalWrite(MotorPin3, HIGH);
327       digitalWrite(MotorPin4, HIGH);
328       delay(delayTime);
329
330       digitalWrite(MotorPin1, HIGH);
331       digitalWrite(MotorPin2, LOW);
332       digitalWrite(MotorPin3, LOW);
333       digitalWrite(MotorPin4, HIGH);
334       delay(delayTime);
335       Serial.println("Fermer porte"); // Affichage sur le moniteur série du texte
336
```

```
337       fdc_ferme = !digitalRead(fin_de_course_ferme);
338       etat_bp_b = !digitalRead(commande_fermeture); // Inverse de la lecture sur entrée BP bas
339       if (fdc_ferme || etat_bp_b)
340       {
341         porte_fermee = true;
342         porte_ouverte = false;
343         Serial.println("Porte fermée"); // Affichage sur le moniteur série du texte
344         Arret();
345         break;
346       }
347     }
348  }
349
350  void Ouvrir_porte() {//Tour < 280
351   while (!fdc_ouvert) {//Tour < 280
352     digitalWrite(MotorPin1, LOW);
353     digitalWrite(MotorPin2, LOW);
354     digitalWrite(MotorPin3, HIGH);
355     digitalWrite(MotorPin4, HIGH);
356     delay(delayTime);
357
358     digitalWrite(MotorPin1, LOW);
359     digitalWrite(MotorPin2, HIGH);
360     digitalWrite(MotorPin3, HIGH);
361     digitalWrite(MotorPin4, LOW);
362     delay(delayTime);
363
364     digitalWrite(MotorPin1, HIGH);
365     digitalWrite(MotorPin2, HIGH);
366     digitalWrite(MotorPin3, LOW);
367     digitalWrite(MotorPin4, LOW);
368     delay(delayTime);
369
370     digitalWrite(MotorPin1, HIGH);
371     digitalWrite(MotorPin2, LOW);
372     digitalWrite(MotorPin3, LOW);
373     digitalWrite(MotorPin4, HIGH);
374     delay(delayTime);
375       Serial.println("Ouvrir porte"); // Affichage sur le moniteur série du texte
376
377       fdc_ouvert = !digitalRead(fin_de_course_ouvert);
378       etat_bp_h = !digitalRead(commande_ouverture); // Inverse de la lecture sur entrée BP haut
379       if (fdc_ouvert || etat_bp_h)
380       {
381         porte_fermee = false;
382         porte_ouverte = true;
383         Serial.println("Porte ouverte"); // Affichage sur le moniteur série du texte
384         Arret();
385         break;
386       }
387     }
388  }
389
390  void Arret() {
391    digitalWrite(MotorPin1, LOW);
392    digitalWrite(MotorPin2, LOW);
```

```
393    digitalWrite(MotorPin3, LOW);
394    digitalWrite(MotorPin4, LOW);
395    tempoActive = 0;
396  }
```