#include <Wire.h>

#include <RTClib.h>

#include <MFRC522.h>

#include <Servo.h>

#define SS\_PIN 10

#define RST\_PIN 9

#define PIR\_PIN 2

#define SERVO\_PIN 6

#define MOTOR\_STEP\_PIN 3

#define MOTOR\_DIR\_PIN 4

MFRC522 rfid(SS\_PIN, RST\_PIN);

Servo servo;

RTC\_DS3231 rtc;

const int stepsPerRevolution = 2048; // Nombre de pas pour une révolution complète du 28BYJ-48

int stepperSpeed = 15; // Vitesse du moteur en tours par minute

int angleOuvert = 90;

int angleFerme = 0;

bool estOuvert = false;

unsigned long lastMotorTime = 0;

unsigned long motorInterval = 8 \* 60 \* 60 \* 1000; // 8 heures en millisecondes

unsigned long motorDuration = 10000; // 10 secondes

void setup() {

Serial.begin(9600);

SPI.begin();

rfid.PCD\_Init();

servo.attach(SERVO\_PIN);

servo.write(angleFerme);

pinMode(PIR\_PIN, INPUT);

if (!rtc.begin()) {

Serial.println("Impossible de trouver le module RTC");

while (1);

}

if (rtc.lostPower()) {

Serial.println("L'heure n'est pas définie, initialisation...");

rtc.adjust(DateTime(F(\_\_DATE\_\_), F(\_\_TIME\_\_)));

}

}

void loop() {

DateTime now = rtc.now();

if (rfid.PICC\_IsNewCardPresent()) {

if (rfid.PICC\_ReadCardSerial()) {

Serial.println("Puce RFID détectée !");

if (!estOuvert) {

servo.write(angleOuvert);

estOuvert = true;

Serial.println("Couvercle ouvert.");

}

delay(1000);

}

}

if (estOuvert && digitalRead(PIR\_PIN) == LOW) {

servo.write(angleFerme);

estOuvert = false;

Serial.println("Couvercle fermé.");

delay(1000);

}

unsigned long currentMillis = millis();

if (currentMillis - lastMotorTime >= motorInterval && !estOuvert) {

Serial.println("Activation du moteur pas à pas...");

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

digitalWrite(MOTOR\_DIR\_PIN, HIGH);

delay(1000 / stepperSpeed);

digitalWrite(MOTOR\_DIR\_PIN, LOW);

delay(1000 / stepperSpeed);

}

delay(motorDuration);

lastMotorTime = currentMillis;

Serial.println("Moteur désactivé.");

}

}  
  
  
AVEC BOUTON  
  
#include <Wire.h>

#include <RTClib.h>

#include <MFRC522.h>

#include <Servo.h>

#define SS\_PIN 10

#define RST\_PIN 9

#define PIR\_PIN 2

#define SERVO\_PIN 6

#define MOTOR\_STEP\_PIN 3

#define MOTOR\_DIR\_PIN 4

#define BTN\_INC\_PIN 5

#define BTN\_DEC\_PIN 7

MFRC522 rfid(SS\_PIN, RST\_PIN);

Servo servo;

RTC\_DS3231 rtc;

const int stepsPerRevolution = 2048; // Nombre de pas pour une révolution complète du 28BYJ-48

int stepperSpeed = 15; // Vitesse du moteur en tours par minute

int angleOuvert = 90;

int angleFerme = 0;

bool estOuvert = false;

unsigned long lastMotorTime = 0;

unsigned long motorInterval = 8 \* 60 \* 60 \* 1000; // 8 heures en millisecondes

unsigned long motorDuration = 10000; // 10 secondes

unsigned long lastDebounceTimeInc = 0;

unsigned long lastDebounceTimeDec = 0;

unsigned long debounceDelay = 50;

void setup() {

Serial.begin(9600);

SPI.begin();

rfid.PCD\_Init();

servo.attach(SERVO\_PIN);

servo.write(angleFerme);

pinMode(PIR\_PIN, INPUT);

pinMode(BTN\_INC\_PIN, INPUT\_PULLUP);

pinMode(BTN\_DEC\_PIN, INPUT\_PULLUP);

if (!rtc.begin()) {

Serial.println("Impossible de trouver le module RTC");

while (1);

}

if (rtc.lostPower()) {

Serial.println("L'heure n'est pas définie, initialisation...");

rtc.adjust(DateTime(F(\_\_DATE\_\_), F(\_\_TIME\_\_)));

}

}

void loop() {

DateTime now = rtc.now();

if (rfid.PICC\_IsNewCardPresent()) {

if (rfid.PICC\_ReadCardSerial()) {

Serial.println("Puce RFID détectée !");

if (!estOuvert) {

servo.write(angleOuvert);

estOuvert = true;

Serial.println("Couvercle ouvert.");

}

delay(1000);

}

}

if (estOuvert && digitalRead(PIR\_PIN) == LOW) {

servo.write(angleFerme);

estOuvert = false;

Serial.println("Couvercle fermé.");

delay(1000);

}

unsigned long currentMillis = millis();

if (currentMillis - lastMotorTime >= motorInterval && !estOuvert) {

Serial.println("Activation du moteur pas à pas...");

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

digitalWrite(MOTOR\_DIR\_PIN, HIGH);

delay(1000 / stepperSpeed);

digitalWrite(MOTOR\_DIR\_PIN, LOW);

delay(1000 / stepperSpeed);

}

delay(motorDuration);

lastMotorTime = currentMillis;

Serial.println("Moteur désactivé.");

}

if (digitalRead(BTN\_INC\_PIN) == LOW && (currentMillis - lastDebounceTimeInc) > debounceDelay) {

lastDebounceTimeInc = currentMillis;

motorInterval += 30 \* 60 \* 1000; // Augmente l'intervalle de 30 minutes

Serial.print("Intervalle augmenté à : ");

Serial.println(motorInterval / 1000 / 60 / 60);

}

if (digitalRead(BTN\_DEC\_PIN) == LOW && (currentMillis - lastDebounceTimeDec) > debounceDelay) {

lastDebounceTimeDec = currentMillis;

if (motorInterval > 30 \* 60 \* 1000) { // Empêche l'intervalle de devenir négatif

motorInterval -= 30 \* 60 \* 1000; // Diminue l'intervalle de 30 minutes

Serial.print("Intervalle réduit à : ");

Serial.println(motorInterval / 1000 / 60 / 60);

}

}

}  
  
  
AVEC BOUTON + ETHERNET  
  
#include <SPI.h>

#include <UIPEthernet.h>

#include <MFRC522.h>

#include <Servo.h>

#define SS\_PIN 10

#define RST\_PIN 9

#define MOTOR\_STEP\_PIN 3

#define MOTOR\_DIR\_PIN 4

#define SERVO\_PIN 6

#define PIR\_PIN 2

#define BTN\_INC\_PIN 5

#define BTN\_DEC\_PIN 7

MFRC522 rfid(SS\_PIN, RST\_PIN);

Servo servo;

EthernetServer server(80);

const int stepsPerRevolution = 2048;

int stepperSpeed = 15;

int angleOuvert = 90;

int angleFerme = 0;

bool estOuvert = false;

unsigned long lastMotorTime = 0;

unsigned long motorInterval = 8 \* 60 \* 60 \* 1000;

unsigned long motorDuration = 10000;

unsigned long lastDebounceTimeInc = 0;

unsigned long lastDebounceTimeDec = 0;

unsigned long debounceDelay = 50;

void setup() {

Serial.begin(9600);

SPI.begin();

rfid.PCD\_Init();

servo.attach(SERVO\_PIN);

servo.write(angleFerme);

pinMode(PIR\_PIN, INPUT);

pinMode(BTN\_INC\_PIN, INPUT\_PULLUP);

pinMode(BTN\_DEC\_PIN, INPUT\_PULLUP);

uint8\_t mac[6] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};

Ethernet.begin(mac);

server.begin();

Serial.print("Serveur web démarré à l'adresse IP : ");

Serial.println(Ethernet.localIP());

}

void loop() {

EthernetClient client = server.available();

if (client) {

String request = "";

while (client.connected()) {

if (client.available()) {

char c = client.read();

request += c;

if (c == '\n') {

break;

}

}

}

if (request.indexOf("GET /on") >= 0) {

digitalWrite(MOTOR\_DIR\_PIN, HIGH);

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

digitalWrite(MOTOR\_STEP\_PIN, HIGH);

delay(1000 / stepperSpeed);

digitalWrite(MOTOR\_STEP\_PIN, LOW);

delay(1000 / stepperSpeed);

}

delay(motorDuration);

lastMotorTime = millis();

}

if (request.indexOf("GET /increase") >= 0 && (millis() - lastDebounceTimeInc) > debounceDelay) {

lastDebounceTimeInc = millis();

motorInterval += 30 \* 60 \* 1000;

}

if (request.indexOf("GET /decrease") >= 0 && (millis() - lastDebounceTimeDec) > debounceDelay) {

lastDebounceTimeDec = millis();

if (motorInterval > 30 \* 60 \* 1000) {

motorInterval -= 30 \* 60 \* 1000;

}

}

client.println("HTTP/1.1 200 OK");

client.println("Content-Type: text/html");

client.println();

client.println("<html>");

client.println("<head><title>Contrôle du moteur pas à pas</title></head>");

client.println("<body>");

client.println("<h1>Contrôle du moteur pas à pas</h1>");

client.println("<p><a href=\"/on\">Activer le moteur</a></p>");

client.println("<p><a href=\"/increase\">Augmenter l'intervalle de 30 minutes</a></p>");

client.println("<p><a href=\"/decrease\">Diminuer l'intervalle de 30 minutes</a></p>");

client.println("<p>Intervalle actuel : ");

client.print(motorInterval / 1000 / 60 / 60);

client.println(" heures</p>");

client.println("</body>");

client.println("</html>");

client.stop();

}

if (millis() - lastMotorTime >= motorInterval && !estOuvert) {

digitalWrite(MOTOR\_DIR\_PIN, HIGH);

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

digitalWrite(MOTOR\_STEP\_PIN, HIGH);

delay(1000 / stepperSpeed);

digitalWrite(MOTOR\_STEP\_PIN, LOW);

delay(1000 / stepperSpeed);

}

delay(motorDuration);

lastMotorTime = millis();

}

if (rfid.PICC\_IsNewCardPresent()) {

if (rfid.PICC\_ReadCardSerial()) {

Serial.println("Puce RFID détectée !");

if (!estOuvert) {

servo.write(angleOuvert);

estOuvert = true;

Serial.println("Couvercle ouvert.");

}

delay(1000);

}

}

if (estOuvert && digitalRead(PIR\_PIN) == LOW) {

servo.write(angleFerme);

estOuvert = false;

Serial.println("Couvercle fermé.");

::contentReference[oaicite:43]{index=43}  
  
  
SENSIBLEMENT AVEC TOUTE  
  
#include <SPI.h>

#include <UIPEthernet.h>

#include <MFRC522.h>

#include <Servo.h>

#include <Wire.h>

#include <RTClib.h>

#define SS\_PIN 10

#define RST\_PIN 9

#define MOTOR\_STEP\_PIN 3

#define MOTOR\_DIR\_PIN 4

#define SERVO\_PIN 6

#define PIR\_PIN 2

#define BTN\_INC\_PIN 5

#define BTN\_DEC\_PIN 7

MFRC522 rfid(SS\_PIN, RST\_PIN);

Servo servo;

RTC\_DS3231 rtc;

EthernetServer server(80);

const int stepsPerRevolution = 2048;

int stepperSpeed = 15;

int angleOuvert = 90;

int angleFerme = 0;

bool estOuvert = false;

unsigned long lastMotorTime = 0;

unsigned long motorInterval = 8UL \* 60 \* 60 \* 1000; // 8 hours in ms

unsigned long motorDuration = 10000; // 10 seconds

unsigned long lastDebounceTimeInc = 0;

unsigned long lastDebounceTimeDec = 0;

unsigned long debounceDelay = 50;

void setup() {

Serial.begin(9600);

SPI.begin();

rfid.PCD\_Init();

servo.attach(SERVO\_PIN);

servo.write(angleFerme);

pinMode(PIR\_PIN, INPUT);

pinMode(BTN\_INC\_PIN, INPUT\_PULLUP);

pinMode(BTN\_DEC\_PIN, INPUT\_PULLUP);

pinMode(MOTOR\_STEP\_PIN, OUTPUT);

pinMode(MOTOR\_DIR\_PIN, OUTPUT);

// Ethernet initialization

uint8\_t mac[6] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};

Ethernet.begin(mac);

server.begin();

Serial.print("Serveur web démarré à l'adresse IP : ");

Serial.println(Ethernet.localIP());

// RTC initialization

Wire.begin();

if (!rtc.begin()) {

Serial.println("Impossible de trouver le module RTC");

while (1);

}

if (rtc.lostPower()) {

Serial.println("L'heure n'est pas définie, initialisation...");

rtc.adjust(DateTime(F(\_\_DATE\_\_), F(\_\_TIME\_\_)));

}

}

void activerMoteur() {

Serial.println("Activation du moteur pas à pas...");

digitalWrite(MOTOR\_DIR\_PIN, HIGH);

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

digitalWrite(MOTOR\_STEP\_PIN, HIGH);

delay(1000 / stepperSpeed);

digitalWrite(MOTOR\_STEP\_PIN, LOW);

delay(1000 / stepperSpeed);

}

delay(motorDuration);

lastMotorTime = millis();

Serial.println("Moteur désactivé.");

}

void loop() {

DateTime now = rtc.now();

// Web server

EthernetClient client = server.available();

if (client) {

String request = "";

while (client.connected()) {

if (client.available()) {

char c = client.read();

request += c;

if (c == '\n') break;

}

}

if (request.indexOf("GET /on") >= 0) {

activerMoteur();

}

if (request.indexOf("GET /increase") >= 0 && (millis() - lastDebounceTimeInc) > debounceDelay) {

lastDebounceTimeInc = millis();

motorInterval += 30UL \* 60 \* 1000;

}

if (request.indexOf("GET /decrease") >= 0 && (millis() - lastDebounceTimeDec) > debounceDelay) {

lastDebounceTimeDec = millis();

if (motorInterval > 30UL \* 60 \* 1000) {

motorInterval -= 30UL \* 60 \* 1000;

}

}

client.println("HTTP/1.1 200 OK");

client.println("Content-Type: text/html");

client.println();

client.println("<html><head><title>Contrôle</title></head><body>");

client.println("<h1>Contrôle du moteur pas à pas</h1>");

client.println("<p><a href=\"/on\">Activer le moteur</a></p>");

client.println("<p><a href=\"/increase\">Augmenter l'intervalle de 30 min</a></p>");

client.println("<p><a href=\"/decrease\">Diminuer l'intervalle de 30 min</a></p>");

client.print("<p>Intervalle actuel : ");

client.print(motorInterval / 1000 / 60 / 60);

client.println(" heures</p>");

client.println("</body></html>");

client.stop();

}

// Motor auto-activation

if ((millis() - lastMotorTime >= motorInterval) && !estOuvert) {

activerMoteur();

}

// RFID detection

if (rfid.PICC\_IsNewCardPresent() && rfid.PICC\_ReadCardSerial()) {

Serial.println("Puce RFID détectée !");

if (!estOuvert) {

servo.write(angleOuvert);

estOuvert = true;

Serial.println("Couvercle ouvert.");

}

delay(1000);

}

// PIR detection

if (estOuvert && digitalRead(PIR\_PIN) == LOW) {

servo.write(angleFerme);

estOuvert = false;

Serial.println("Couvercle fermé.");

delay(1000);

}

// Button Increase

if (digitalRead(BTN\_INC\_PIN) == LOW && (millis() - lastDebounceTimeInc) > debounceDelay) {

lastDebounceTimeInc = millis();

motorInterval += 30UL \* 60 \* 1000;

Serial.print("Intervalle augmenté à : ");

Serial.println(motorInterval / 1000 / 60 / 60);

}

// Button Decrease

if (digitalRead(BTN\_DEC\_PIN) == LOW && (millis() - lastDebounceTimeDec) > debounceDelay) {

lastDebounceTimeDec = millis();

if (motorInterval > 30UL \* 60 \* 1000) {

motorInterval -= 30UL \* 60 \* 1000;

Serial.print("Intervalle réduit à : ");

Serial.println(motorInterval / 1000 / 60 / 60);

}

}

}

#SENSIBLEMENT AVEC TOUTE 2  
#include <SPI.h>

#include <UIPEthernet.h>

#include <MFRC522.h>

#include <Servo.h>

#include <Wire.h>

#include <RTClib.h>

// Définition des broches

#define RFID\_SDA\_PIN 10 // SS (CS) pour le module RFID

#define RFID\_RST\_PIN 9 // RST pour le module RFID

#define MOTOR\_STEP\_PIN 3 // IN1 pour le moteur pas à pas

#define MOTOR\_DIR\_PIN 4 // IN2 pour le moteur pas à pas

#define MOTOR\_IN3\_PIN 5 // IN3 pour le moteur pas à pas

#define MOTOR\_IN4\_PIN 6 // IN4 pour le moteur pas à pas (partagé avec servo)

#define SERVO\_PIN 7 // Signal pour le servo moteur (déplacé à D7)

#define PIR\_PIN 2 // OUT pour le capteur PIR

#define BTN\_INC\_PIN A0 // Bouton +

#define BTN\_DEC\_PIN A1 // Bouton -

// Initialisation des modules

MFRC522 rfid(RFID\_SDA\_PIN, RFID\_RST\_PIN);

Servo servo;

RTC\_DS1307 rtc;

EthernetServer server(80);

const int stepsPerRevolution = 2048; // Nombre de pas pour une révolution complète du moteur

int stepperSpeed = 15; // Vitesse du moteur en tours par minute

int angleOuvert = 90; // Angle pour ouvrir le servo

int angleFerme = 0; // Angle pour fermer le servo

bool estOuvert = false;

unsigned long lastMotorTime = 0;

unsigned long motorInterval = 8UL \* 60 \* 60 \* 1000; // 8 heures en ms

unsigned long motorDuration = 10000; // 10 secondes

unsigned long lastDebounceTimeInc = 0;

unsigned long lastDebounceTimeDec = 0;

unsigned long debounceDelay = 50;

void setup() {

Serial.begin(9600);

SPI.begin();

// Initialisation RFID

rfid.PCD\_Init();

// Initialisation du servo

servo.attach(SERVO\_PIN); // Utilise D7 pour le servo

servo.write(angleFerme); // Position fermée du servo

// Configuration des broches

pinMode(PIR\_PIN, INPUT); // Capteur PIR

pinMode(BTN\_INC\_PIN, INPUT\_PULLUP); // Bouton +

pinMode(BTN\_DEC\_PIN, INPUT\_PULLUP); // Bouton -

pinMode(MOTOR\_STEP\_PIN, OUTPUT); // IN1 pour le moteur

pinMode(MOTOR\_DIR\_PIN, OUTPUT); // IN2 pour le moteur

pinMode(MOTOR\_IN3\_PIN, OUTPUT); // IN3 pour le moteur

pinMode(MOTOR\_IN4\_PIN, OUTPUT); // IN4 pour le moteur (partagé avec le servo)

// Initialisation du serveur Ethernet

uint8\_t mac[6] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};

Ethernet.begin(mac);

server.begin();

Serial.print("Serveur web démarré à l'adresse IP : ");

Serial.println(Ethernet.localIP());

// Initialisation du module RTC

Wire.begin();

if (!rtc.begin()) {

Serial.println("Impossible de trouver le module RTC");

while (1);

}

if (rtc.lostPower()) {

Serial.println("L'heure n'est pas définie, initialisation...");

rtc.adjust(DateTime(F(\_\_DATE\_\_), F(\_\_TIME\_\_))); // Réinitialiser l'heure à la compilation

}

}

void loop() {

DateTime now = rtc.now();

// Serveur web

EthernetClient client = server.available();

if (client) {

String request = "";

while (client.connected()) {

if (client.available()) {

char c = client.read();

request += c;

if (c == '\n') break;

}

}

// Commande d'activation du moteur

if (request.indexOf("GET /on") >= 0) {

activerMoteur();

}

// Commande d'augmentation de l'intervalle

if (request.indexOf("GET /increase") >= 0 && (millis() - lastDebounceTimeInc) > debounceDelay) {

lastDebounceTimeInc = millis();

motorInterval += 30UL \* 60 \* 1000;

}

// Commande de réduction de l'intervalle

if (request.indexOf("GET /decrease") >= 0 && (millis() - lastDebounceTimeDec) > debounceDelay) {

lastDebounceTimeDec = millis();

if (motorInterval > 30UL \* 60 \* 1000) {

motorInterval -= 30UL \* 60 \* 1000;

}

}

// Réponse HTTP

client.println("HTTP/1.1 200 OK");

client.println("Content-Type: text/html");

client.println();

client.println("<html><head><title>Contrôle</title></head><body>");

client.println("<h1>Contrôle du moteur pas à pas</h1>");

client.println("<p><a href=\"/on\">Activer le moteur</a></p>");

client.println("<p><a href=\"/increase\">Augmenter l'intervalle de 30 min</a></p>");

client.println("<p><a href=\"/decrease\">Diminuer l'intervalle de 30 min</a></p>");

client.print("<p>Intervalle actuel : ");

client.print(motorInterval / 1000 / 60 / 60);

client.println(" heures</p>");

client.println("</body></html>");

client.stop();

}

// Activation automatique du moteur

if ((millis() - lastMotorTime >= motorInterval) && !estOuvert) {

activerMoteur();

}

// Détection RFID

if (rfid.PICC\_IsNewCardPresent() && rfid.PICC\_ReadCardSerial()) {

Serial.println("Puce RFID détectée !");

if (!estOuvert) {

servo.write(angleOuvert); // Ouvre le servo

estOuvert = true;

Serial.println("Couvercle ouvert.");

}

delay(1000);

}

// Détection PIR

if (estOuvert && digitalRead(PIR\_PIN) == LOW) {

servo.write(angleFerme); // Ferme le servo

estOuvert = false;

Serial.println("Couvercle fermé.");

delay(1000);

}

// Bouton Augmenter

if (digitalRead(BTN\_INC\_PIN) == LOW && (millis() - lastDebounceTimeInc) > debounceDelay) {

lastDebounceTimeInc = millis();

motorInterval += 30UL \* 60 \* 1000;

Serial.print("Intervalle augmenté à : ");

Serial.println(motorInterval / 1000 / 60 / 60);

}

// Bouton Diminuer

if (digitalRead(BTN\_DEC\_PIN) == LOW && (millis() - lastDebounceTimeDec) > debounceDelay) {

lastDebounceTimeDec = millis();

if (motorInterval > 30UL \* 60 \* 1000) {

motorInterval -= 30UL \* 60 \* 1000;

Serial.print("Intervalle réduit à : ");

Serial.println(motorInterval / 1000 / 60 / 60);

}

}

}

void activerMoteur() {

Serial.println("Activation du moteur pas à pas...");

digitalWrite(MOTOR\_DIR\_PIN, HIGH);

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

digitalWrite(MOTOR\_STEP\_PIN, HIGH);

delay(1000 / stepperSpeed);

digitalWrite(MOTOR\_STEP\_PIN, LOW);

delay(1000 / stepperSpeed);

}

delay(motorDuration);

lastMotorTime = millis();

Serial.println("Moteur désactivé.");

}