#include <math.h>

#include <SPI.h>

#include <SD.h>

//SD card

int chipSelect = 10; // Pin number for the SD card chip select

File dataFile; // File object to handle read/write operations on the SD card

//alarm

int alarm = A5; // Pin number where the alarm or buzzer is connected

// infrared sensors

int leftinfraredSensor = 9; // Pin number for the left infrared sensor

int rightinfraredSensor = 3; // Pin number for the right infrared sensor

int middleinfraredSensor= A4; // Pin number for the middle infrared sensor

int leftSensorvalue; // Variable to store the left sensor value

int rightSensorvalue; // Variable to store the right sensor value

int middleSensorvalue; // Variable to store the middle sensor value

// Motor control pins

const int motorLeftPWM = 5; // Pin for PWM control of the left motor

const int motorRightPWM = 6; // Pin for PWM control of the right motor

const int motorLeftDir1 = 2; // Pin for direction control of the left motor

const int motorLeftDir2 = 4; // Pin for direction control of the left motor

const int motorRightDir1 = 7; // Pin for direction control of the right motor

const int motorRightDir2 = 8; // Pin for direction control of the right motor

// Ultrasonic sensor pins

const int triggerPin = A3; // Pin to trigger all three ultrasonic sensor

const int echoPinFront = A0; // Pin to receive the echo from the front ultrasonic sensor

const int echoPinLeft = A1; // Pin to receive the echo from the left ultrasonic sensor

const int echoPinRight = A2; // Pin to receive the echo from the right ultrasonic sensor

// Parameters

int safeDistance = 40; // safe distance in cm

float distanceFront; // Variable to store the distance of obstacle detected by front ultrasonic sensor

float distanceLeft; // Variable to store the distance of obstacle detected by left ultrasonic sensor

float distanceRight; // Variable to store the distance of obstacle detected by right ultrasonic sensor

float d\_sensor;

int criticaldistance= 11; // Define critical distance where immediate action is needed

// Function prototypes

void moveForward(); // Function to move the robot forward

void turnLeft(); // Function to turn the robot to the left

void turnRight(); // Function to turn the robot to the right

void stopMotors(); // Function to stop all motors

float getDistance(int echoPin); // Function to calculate distance from ultrasonic sensor data

void setup() {

Serial.begin(9600); // Begin serial communication at 9600 baud rate

// Set pin modes

// SD card

pinMode(chipSelect, OUTPUT); // Set chip select pin as output for SD card

// Initialize SD card

Serial.print("Initializing SD card...");

if (!SD.begin(chipSelect)) {

Serial.println("SD card initialization failed.");

return;

}

Serial.println("SD card initialized.");

//alarm

pinMode(alarm, OUTPUT); // Set alarm pin as output

//infrared sensor

// Set infrared sensor pins as input

pinMode(leftinfraredSensor, INPUT);

pinMode(rightinfraredSensor, INPUT);

pinMode(middleinfraredSensor, INPUT);

// Set motor control pins as output

pinMode(motorLeftPWM, OUTPUT);

pinMode(motorRightPWM, OUTPUT);

pinMode(motorLeftDir1, OUTPUT);

pinMode(motorLeftDir2, OUTPUT);

pinMode(motorRightDir1, OUTPUT);

pinMode(motorRightDir2, OUTPUT);

// Ultrasonic sensor pin setup

pinMode(triggerPin, OUTPUT); // Set trigger pin as output

pinMode(echoPinFront, INPUT); // Set front echo pin as input

pinMode(echoPinLeft, INPUT); // Set left echo pin as input

pinMode(echoPinRight, INPUT); // Set right echo pin as input

// Initialize distance variables to a high value to assume no obstacle

distanceFront = 200;

distanceLeft = 200;

distanceRight = 200;

}

void loop(){

SDcardopenfile(); // Open the SD card file to log data

// Measure distances from all three ultrasonic sensors

distanceFront = getDistance(echoPinFront); // Get the distance from the front sensor

distanceLeft = getDistance(echoPinLeft); // Get the distance from the left sensor

distanceRight = getDistance(echoPinRight); // Get the distance from the right sensor

infraredsensor(); // Read values from infrared sensors

// Check if the path ahead of the robot is clear

if (distanceFront > safeDistance) {

// If the front is clear, but the left side is too close to an obstacle

if (distanceLeft <= safeDistance && leftSensorvalue == 0 ) {

moveForward(); // Continue moving forward

turnRightslightly(); // Make a slight turn to the right

Serial.println(" Turn right slightly ");

dataFile.println(" Turn right slightly ");

// If the left sensor reads a critical distance and the infrared sensor does not detect an object

} if (distanceLeft <= criticaldistance && leftSensorvalue == 1 ) {

//anomaly detected stop robot and sound alarm

stopMotors(); // Stop the robot

alarmtone(); // Sound the alarm

Serial.println(" Anomaly detected: left distance data not verified by left infraredsensor data ");

dataFile.println(" Anomaly detected: left distance data not verified by left infraredsensor data ");

}else if (distanceRight <= safeDistance && rightSensorvalue == 0) {

// Turn left slightly if an obstacle is close on the right

moveForward(); // Continue moving forward

turnLeftslightly(); // Make a slight turn to the left

Serial.println(" Turn left slightly ");

dataFile.println(" Turn left slightly ");

} if (distanceRight <= criticaldistance && rightSensorvalue == 1 ) {

// If the right sensor reads a critical distance and the infrared sensor does not detect an object

//anomaly detected stop robot and sound alarm

stopMotors();

alarmtone();

Serial.println(" Anomaly detected: Right distance data not verified by right infraredsensor data ");

dataFile.println(" Anomaly detected: Right distance data not verified by right infraredsensorr data ");

}else {

// Move forward if both sides are safe

moveForward();

Serial.println(" Move forward ");

dataFile.println(" Move forward ");

}

}

else if ((distanceRight <= safeDistance && rightSensorvalue == 0) && (distanceLeft <= safeDistance && leftSensorvalue == 0 ) && (distanceFront <= safeDistance && middleSensorvalue == 0 )) {

// If all paths are blocked or too close, move backwards

moveBackward();

Serial.println(" move backwards ");

dataFile.println(" move backwards ");

}else if((distanceRight <= criticaldistance && rightSensorvalue == 1) && (distanceLeft <= criticaldistance && leftSensorvalue == 1 ) && (distanceFront <= criticaldistance && middleSensorvalue == 1 )) {

// If sensors read critical distances and the infrared sensors does not detect objects on all sides

//anomaly detected stop robot and sound alarm

stopMotors();

alarmtone();

Serial.println(" Anomaly detected: All distance data not verified by all infraredsensor data ");

dataFile.println(" Anomaly detected: All distance data not verified by all infraredsensor data ");

}

else {

// If the path directly ahead is not clear, stop the motors

stopMotors();

Serial.println(" Stop motors ");

dataFile.println(" Stop motors ");

if (distanceFront <= criticaldistance && middleSensorvalue == 1 ) {

// If the front sensor reads a critical distance and the infrared sensor does not detect an object

//anomaly detected stop robot and sound alarm

stopMotors();

alarmtone();

Serial.println(" Anomaly detected: left distance data not verified by left infraredsensor data ");

dataFile.println(" Anomaly detected: left distance data not verified by left infraredsensor data ");

}

// Decision-making for turning

if (distanceRight > safeDistance && rightSensorvalue == 1) {

// Turn right if the right side is safe

turnRight();

Serial.println(" Turn right ");

dataFile.println(" Turn right ");

} else if (distanceRight > (criticaldistance + 12) && rightSensorvalue == 0 ) {

// If the right distance sensor does not detect an object in a distance greater than criticaldistance by 12cm,

// and the right infrared sensor does

//anomaly detected stop robot and sound alarm

stopMotors();

alarmtone();

Serial.println(" Anomaly detected: Right distance data not verified by right infraredsensor data ");

dataFile.println(" Anomaly detected: Right distance data not verified by right infraredsensor data ");

}else if (distanceLeft > safeDistance) {

// Turn left if the left side is safe

turnLeft();

Serial.println(" Turn left ");

dataFile.println(" Turn left ");

}else if (distanceLeft > (criticaldistance + 12) && leftSensorvalue == 0 ) {

// If the left distance sensor does not detect an object in a distance greater than criticaldistance by 12cm,

// and the left infrared sensor does

//anomaly detected stop robot and sound alarm

stopMotors();

alarmtone();

Serial.println(" Anomaly detected: Left distance data not verified by left infraredsensor data ");

dataFile.println(" Anomaly detected: Left distance data not verified by left infraredsensor data ");

}

}

debugging(); // Call the debugging function to print sensor values and states for monitoring purposes

}

// Function to move the robot forward

void moveForward() {

digitalWrite(motorLeftDir1, LOW);

digitalWrite(motorLeftDir2, HIGH);

digitalWrite(motorRightDir1, LOW);

digitalWrite(motorRightDir2, HIGH);

// Set the speed of both motors to 100

analogWrite(motorLeftPWM, 100); // adjust speed

analogWrite(motorRightPWM, 100);

}

// Function to move the robot backward

void moveBackward(){

// Set the motor direction to backward by inverting the forward logic

digitalWrite(motorLeftDir1, HIGH);

digitalWrite(motorLeftDir2, LOW);

digitalWrite(motorRightDir1, HIGH);

digitalWrite(motorRightDir2, LOW);

// Set the speed of both motors to 100

analogWrite(motorLeftPWM, 100); // adjust speed

analogWrite(motorRightPWM, 100);

}

// Function to turn the robot to the left

void turnLeft() {

// To turn left, the right motor moves forward and the left motor is stopped

digitalWrite(motorLeftDir1, LOW);

digitalWrite(motorLeftDir2, HIGH);

digitalWrite(motorRightDir1, LOW);

digitalWrite(motorRightDir2, LOW);

analogWrite(motorRightPWM, 100); // Speed of right motor

}

// Function to turn the robot to the right

void turnRight() {

// To turn right, the left motor moves forward and the right motor is stopped

digitalWrite(motorLeftDir1, LOW);

digitalWrite(motorLeftDir2, LOW);

digitalWrite(motorRightDir1, LOW);

digitalWrite(motorRightDir2, HIGH);

analogWrite(motorLeftPWM, 100); // Speed of left motor

}

// Function to slightly turn the robot to the left

void turnLeftslightly() {

// Similar to turnLeft but includes a brief delay to make the turn slight

digitalWrite(motorLeftDir1, LOW);

digitalWrite(motorLeftDir2, HIGH);

digitalWrite(motorRightDir1, LOW);

digitalWrite(motorRightDir2, LOW);

analogWrite(motorRightPWM, 100);

delay(500); // Delay to reduce the angle of the turn

}

// Function to slightly turn the robot to the right

void turnRightslightly() {

// Similar to turnRight but includes a brief delay to make the turn slight

digitalWrite(motorLeftDir1, LOW);

digitalWrite(motorLeftDir2, LOW);

digitalWrite(motorRightDir1, LOW);

digitalWrite(motorRightDir2, HIGH);

analogWrite(motorLeftPWM, 100);

delay(500); // Delay to reduce the angle of the turn

}

// Function to stop all motors

void stopMotors() {

// Set all motor terminals to LOW to stop the motors

digitalWrite(motorLeftDir1, LOW);

digitalWrite(motorLeftDir2, LOW);

digitalWrite(motorRightDir1, LOW);

digitalWrite(motorRightDir2, LOW);

delay(500); // Delay to ensure motors have time to stop

}

float getDistance(int echoPin) {

// Send out an ultrasonic pulse

digitalWrite(triggerPin, LOW);

delayMicroseconds(2);

digitalWrite(triggerPin, HIGH);

delayMicroseconds(10);

digitalWrite(triggerPin, LOW);

// Measure the duration it takes for the echo to return

float duration = pulseIn(echoPin, HIGH);

float distance = duration \* 0.034 / 2; // Speed of sound at 20°C is approximately 343 m/s

// Check if the distance is within an expected range, otherwise set to a default value

if(distance == 0 || distance > safeDistance) {

Serial.println("Sensor: Invalid distance");

d\_sensor = 200; // Default or safe value

} else {

d\_sensor= distance;

}

return d\_sensor; // Return the processed distance

}

// Open a file on the SD card for writing

void SDcardopenfile(){

dataFile = SD.open("data.txt", FILE\_WRITE);

// Check if the file opened successfully

if (dataFile) {

Serial.println("File opened for writing.");

} else {

Serial.println("Error opening data.txt for writing.");

}

}

// Function to activate an alarm tone

void alarmtone(){

//turn on the buzzer at 1535 frequency for 500 milliseconds

tone(alarm,400,500);

//add another 500 milliseconds of silence

delay(500);

}

// Function to read values from infrared sensors and update global variables

void infraredsensor(){

leftSensorvalue = digitalRead(leftinfraredSensor);

rightSensorvalue = digitalRead(rightinfraredSensor);

middleSensorvalue = digitalRead(middleinfraredSensor);

}

// Function for debugging and monitoring sensor values via serial communication

void debugging()

{

Serial.println(" "); // Print empty lines for separation

// Print the distance values from all three ultrasonic sensors

Serial.print("Front distance value: ");

Serial.print(distanceFront);

Serial.print("|| Left distance value: ");

Serial.print(distanceLeft);

Serial.print("|| Right distance value: ");

Serial.println(distanceRight);

// Print the light values from all three infrared sensors

Serial.print("middle\_light\_value = ");

Serial.print(middleSensorvalue);

Serial.print("|| left\_light\_value = ");

Serial.print(leftSensorvalue);

Serial.print("|| right\_light\_value = ");

Serial.print(rightSensorvalue);

Serial.println("");

Serial.println("");

Serial.println("");

// Write the same sensor values to the data.txt file on the SD card

dataFile.println(" "); // Print empty lines for separation

dataFile.print("Front distance value: ");

dataFile.print(distanceFront);

dataFile.print("|| Left distance value: ");

dataFile.print(distanceLeft);

dataFile.print("|| Right distance value: ");

dataFile.println(distanceRight);

dataFile.print("middle\_light\_value = ");

dataFile.print(middleSensorvalue);

dataFile.print("|| left\_light\_value = ");

dataFile.print(leftSensorvalue);

dataFile.print("|| right\_light\_value = ");

dataFile.print(rightSensorvalue);

dataFile.println("");

dataFile.println("");

dataFile.println("");

dataFile.close(); // Close the file after writing to save the data

}