```
#include <SPI.h>
#include <RF24.h>
#include <Servo.h>
RF24 radio(7, 6); // CE, CSN
const byte address[6] = "00001";
struct Data_Package {
  bool bValue; // Joystick Button
  int yValue; // Y value of joystick
  int val;    // Value of potentiometer
  bool buttonState; // State of push button
  int distance; // Distance sensed by Ultrasonic sensor };
Data_Package data;
const int directionPin1 = 12;  // Channel A
const int pwmPin1 = 3;
const int brakePin1 = 9;
const int directionPin2 = 13; // Channel B
const int pwmPin2 = 11;
const int brakePin2 = 8;
const int pingPin = 22;
const int buzzerPin = 10;
Servo myservo;
int horn = 26;
void setup() {
  radio.begin();
  radio.openReadingPipe(1, address); // Set reading pipe
  radio.setPALevel(RF24_PA_MIN);
  radio.enableAckPayload(); // Enable ackPayload feature
  radio.enableDynamicPayloads(); // Enable dynamic payload length
  radio.startListening();
  Serial.begin(9600);
  pinMode(horn, OUTPUT);
  pinMode(directionPin1, OUTPUT);
  pinMode(pwmPin1, OUTPUT);
pinMode(brakePin1, OUTPUT);
  pinMode(directionPin2, OUTPUT);
  pinMode(pwmPin2, OUTPUT);
  pinMode(brakePin2, OUTPUT);
  myservo.attach(2);
  pinMode(buzzerPin, OUTPUT); }
void loop() {
  if (radio.available()) {
    while (radio.available()) { // Read all available payloads
      radio.read(&data, sizeof(Data_Package));
      // Process received data
      myservo.write(data.val);
      Serial.print("Received data - Y value: ");
      Serial.print(data.yValue);
      Serial.print(", Potentiometer value: ");
      Serial.print(data.val);
      Serial.print(", Button state: ");
      Serial.println(data.buttonState);
      // Check if joystick button is pressed, activate horn if true
      if (data.bValue == LOW) {
        tone(horn, 6000); // Send sound signal
        delay(220);
        noTone(horn); // Stop sound
        delay(50);      }
      // Control motors based on joystick input
      if (data.yValue > 300 && data.yValue < 600) {
        // Stop motors
        analogWrite(pwmPin1, 0);
        analogWrite(pwmPin2, 0);
        digitalWrite(brakePin1, LOW);
        digitalWrite(brakePin2, LOW);
      } else if (data.yValue < 300) {
        // Move forward
```

```
digitalWrite(directionPin1, LOW);
        digitalWrite(directionPin2, LOW);
        analogWrite(pwmPin1, 100);
        analogWrite(pwmPin2, 100);
        digitalWrite(brakePin1, LOW);
        digitalWrite(brakePin2, LOW);
      } else if (data.yValue > 600) {
        // Move backward
        digitalWrite(directionPin1, HIGH);
        digitalWrite(directionPin2, HIGH);
        analogWrite(pwmPin1, 100);
        analogWrite(pwmPin2, 100);
        digitalWrite(brakePin1, LOW);
        digitalWrite(brakePin2, LOW);      }
      // Check if push button is pressed, activate emergency brake if true
      if(data.buttonState == HIGH) {
        digitalWrite(brakePin1, HIGH);
        digitalWrite(brakePin2, HIGH);      }
      // Read distance from ultrasonic sensor
      data.distance = readDistance();
      Serial.print("Distance: ");
      Serial.println(data.distance);
      // Send distance data back to transmitter as acknowledgment
      radio.writeAckPayload(1, &data, sizeof(Data_Package));
      // Activate buzzer if obstacle detected
      if (data.distance < 5) {
        digitalWrite(brakePin1, HIGH);
        digitalWrite(brakePin2, HIGH);
        tone(buzzerPin, 1000); // Send sound signal
        delay(220);
        noTone(buzzerPin); // Stop sound
        delay(100);
        digitalWrite(brakePin1, LOW);
digitalWrite(brakePin2, LOW);
        digitalWrite(directionPin1, LOW);
        digitalWrite(directionPin2, LOW);
        analogWrite(pwmPin1, 100);
        analogWrite(pwmPin2, 100);
        delay(500); }}}}
// Function to read distance from ultrasonic sensor
int readDistance() {
  long duration, inches, cm;
  pinMode(pingPin, OUTPUT);
  digitalWrite(pingPin, LOW);
  delayMicroseconds(2);
  digitalWrite(pingPin, HIGH);
  delayMicroseconds(5);
  digitalWrite(pingPin, LOW);
  pinMode(pingPin, INPUT);
  duration = pulseIn(pingPin, HIGH);
  inches = microsecondsToInches(duration);
  cm = microsecondsToCentimeters(duration);
  return inches; }
// Function to convert microseconds to inches
long microsecondsToInches(long microseconds) {
  return microseconds / 74 / 2; }
// Function to convert microseconds to centimeters
long microsecondsToCentimeters(long microseconds) {
  return microseconds / 29 / 2; }
```