Auto-Driver

basic requirements

Components Required:

1. **Arduino Uno**: This microcontroller will be the brain of your auto-driver system. It will process inputs, execute control algorithms, and send commands to actuators.
2. **Sensors**:
   - **Ultrasonic Sensors**: Used for detecting obstacles and measuring distances around the car.
   - **Infrared (IR) Sensors**: Can be used for line following or detecting objects.
3. **Motor Driver**: To control the motors responsible for steering and driving the car.
4. **DC Motors**: For controlling the movement of the car (forward, backward) and steering (left, right).
5. **Power Supply**: Depending on the motors and components you use, you may need a separate power supply for the motors (e.g., batteries) and one for the Arduino (e.g., USB or a separate power adapter).
6. **Chassis**: The physical structure of the car where all components will be mounted.
7. **Wires, Breadboard, and Connectors**: For connecting and interfacing all components together.

Overview of Operation:

1. **Sensors Integration**:
   - Connect ultrasonic sensors and IR sensors to the Arduino Uno. Ultrasonic sensors will help detect obstacles around the car, while IR sensors can help with line following or obstacle detection.
   - Write code to read sensor data and interpret it (e.g., detect obstacles, follow lines).

2. **Control Algorithm**:
   - Develop control algorithms based on sensor data to make decisions about steering and driving the car.
   - Use techniques like PID control for smooth and accurate control of the motors.
3. **Motor Control**:
   - Connect the motor driver to the Arduino Uno. The motor driver will provide the necessary power and control signals to the DC motors.
   - Write code to control the motors based on the output of your control algorithm. This includes forward/backward motion and steering.
4. **Testing and Calibration**:
   - Test your system in a controlled environment to ensure that it responds correctly to sensor inputs and executes desired movements.
   - Calibrate your control algorithms as needed for better performance (e.g., adjusting PID constants).
5. **Integration and Finalization**:
   - Mount all components on the car chassis securely.
   - Fine-tune your code and system parameters for optimal performance.

Additional Considerations:

- **Safety**: Implement safety features such as emergency stop buttons or fail-safe mechanisms to prevent accidents during testing.
- **Real-time Constraints**: Consider real-time constraints for sensor readings and motor control to ensure timely responses.
- **Power Management**: Ensure sufficient power supply for all components, especially motors, to avoid performance issues or damage.
- **Programming**: Use Arduino IDE or other compatible software to write, compile, and upload code to your Arduino Uno.

Code for the controller

}

```
#include <Servo.h>

// Define pin connections
const int leftIRSensorPin = A0; // Analog pin for left IR sen
const int rightIRSensorPin = A1; // Analog pin for right IR s
const int obstaclePin = 2; // Digital pin for obstacle detect
const int leftMotorPin = 3; // PWM pin for left motor speed c
const int rightMotorPin = 5; // PWM pin for right motor speed
const int echoPin = 7; // Echo pin for ultrasonic sensor
const int trigPin = 8; // Trigger pin for ultrasonic sensor
const int steeringPin = 9; // Pin for steering servo

// Define constants for sensor thresholds and motor speeds
const int obstacleThreshold = 20; // Distance threshold for o
const int leftEdgeThreshold = 500; // Threshold for detecting
const int rightEdgeThreshold = 500; // Threshold for detectin
const int midSteeringAngle = 90; // Midpoint angle for steeri
const int leftSteeringAngle = 60; // Angle for left steering
const int rightSteeringAngle = 120; // Angle for right steeri
const int baseSpeed = 150; // Base speed for motors (adjust a

// Initialize servo motor for steering
Servo steeringServo;

void setup() {
  Serial.begin(9600); // Initialize serial communication
  pinMode(leftIRSensorPin, INPUT);
  pinMode(rightIRSensorPin, INPUT);
  pinMode(obstaclePin, INPUT);
  pinMode(leftMotorPin, OUTPUT);
  pinMode(rightMotorPin, OUTPUT);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  steeringServo.attach(steeringPin); // Attach servo to pin
}

void loop() {
  int leftIRValue = analogRead(leftIRSensorPin);
```

```
  int rightIRValue = analogRead(rightIRSensorPin);
  int obstacleDistance = getObstacleDistance();

  // Debugging output
  Serial.print("Left IR: ");
  Serial.print(leftIRValue);
  Serial.print(" - Right IR: ");
  Serial.print(rightIRValue);
  Serial.print(" - Obstacle Distance: ");
  Serial.println(obstacleDistance);

  // Check for obstacles
  if (obstacleDistance < obstacleThreshold) {
    avoidObstacle(); // If obstacle detected, avoid it
  } else {
    // Lane following behavior
    if (leftIRValue > leftEdgeThreshold && rightIRValue > rig
      // Both sensors on the lane, go straight
      steeringServo.write(midSteeringAngle);
      driveForward();
    } else if (leftIRValue <= leftEdgeThreshold && rightIRVal
      // Left sensor off the lane, turn left
      steeringServo.write(leftSteeringAngle);
      driveForward();
    } else if (rightIRValue <= rightEdgeThreshold && leftIRVa
      // Right sensor off the lane, turn right
      steeringServo.write(rightSteeringAngle);
      driveForward();
    } else {
      // Both sensors off the lane, stop and analyze situatio
      steeringServo.write(midSteeringAngle);
      stopMotors();
    }
  }
}

int getObstacleDistance() {
  digitalWrite(trigPin, LOW);
```

```
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  long duration = pulseIn(echoPin, HIGH);
  int distance = duration * 0.034 / 2; // Calculate distance
  return distance;
}

void driveForward() {
  analogWrite(leftMotorPin, baseSpeed);
  analogWrite(rightMotorPin, baseSpeed);
}

void stopMotors() {
  analogWrite(leftMotorPin, 0);
  analogWrite(rightMotorPin, 0);
}

void avoidObstacle() {
  stopMotors(); // Stop motors
  delay(1000); // Wait for a moment
  analogWrite(leftMotorPin, -baseSpeed); // Reverse left moto
  analogWrite(rightMotorPin, -baseSpeed); // Reverse right mo
  delay(1000); // Reverse for 1 second
  analogWrite(leftMotorPin, baseSpeed); // Turn right (adjust
  analogWrite(rightMotorPin, 0); // Stop right motor
  delay(1000); // Turn for 1 second
  driveForward(); // Resume forward motion
}
```

Explanation of the Code:

1. **Setup Section**:
    - Initializes pins, serial communication, and attaches the servo motor.
2. **Loop Section**:

- Reads sensor values and checks for obstacles or line following conditions.
- Executes appropriate motor control functions based on sensor inputs.

3. **Functions**:
    - `getObstacleDistance()`: Calculates and returns the distance from an obstacle using the ultrasonic sensor.
    - `driveForward()`, `turnLeft()`, `turnRight()`: Control motor movements for forward, left turn, and right turn.
    - `avoidObstacle()`: Maneuvers the car to avoid obstacles by reversing and turning.