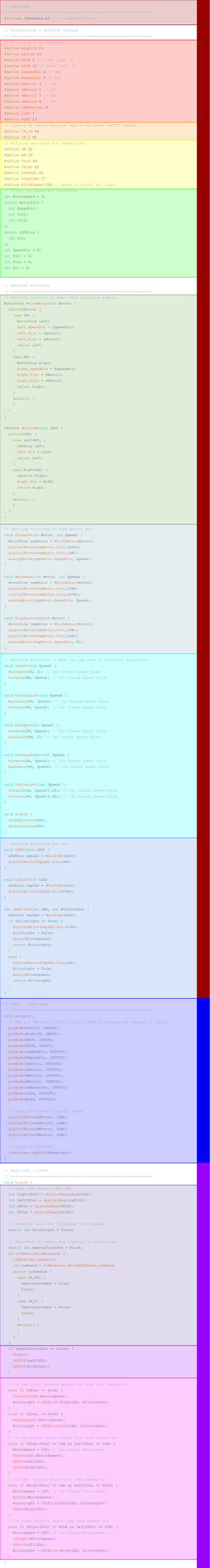

```
// LIBRARIES
// ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#include <IRremote.h> // IR remote library

// INITIALIZING & DEFINING GLOBALS
// ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
// Linking pins to variables
#define RightIR 11
#define LeftIR 10
#define ORIR 2 // outer right ir
#define OLIR 12 // outer left  ir
#define LSpeedPin 5 // ENA
#define RSpeedPin 6 // ENB
#define LMotor1 3 // IN1
#define LMotor2 4 // IN2
#define RMotor1 7 // IN3
#define RMotor2 8 // IN4
#define IRReciver 9
#define LLED 1
#define RLED 13
// Linking IR remote buttons' HEX to variables (DON'T CHANGE)
#define IR_OK 64
#define IR_0 82
// Defining variables for readability
#define LM 22
#define RM 33
#define True 44
#define False 55
#define LeftLED 66
#define RightLED 77
#define BlinkSpeed 200 // Speed of on/off for light
// Global variables and structures
int MotorSpeed = 0;
struct MotorPins {
  int SpeedPin;
  int Pin1;
  int Pin2;
};
struct LEDPins {
  int Pin;
};
int SpeedPin = 0;
int Pin1 = 0;
int Pin2 = 0;
int Pin = 0;

// DEFINING FUNCTIONS
// ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
// Defining function to make other functions simpler
MotorPins WhichMotor(int Motor) {
  switch(Motor) {
    case LM: {
      MotorPins Left;
      Left.SpeedPin = LSpeedPin;
      Left.Pin1 = LMotor1;
      Left.Pin2 = LMotor2;
      return Left;
    }
    case RM: {
      MotorPins Right;
      Right.SpeedPin = RSpeedPin;
      Right.Pin1 = RMotor1;
      Right.Pin2 = RMotor2;
      return Right;
    }
    default: {
    }
  }
}

LEDPins WhichLED(int LED) {
  switch(LED) {
    case LeftLED: {
      LEDPins Left;
      Left.Pin = LLED;
      return Left;
    }
    case RightLED: {
      LEDPins Right;
      Right.Pin = RLED;
      return Right;
    }
    default: {
    }
  }
}

// Defining functions to make motors spin
void Forward(int Motor, int Speed) {
  MotorPins tmpMotor = WhichMotor(Motor);
  digitalWrite(tmpMotor.Pin1,HIGH);
  digitalWrite(tmpMotor.Pin2,LOW);
  analogWrite(tmpMotor.SpeedPin, Speed);
}

void Backward(int Motor, int Speed) {
  MotorPins tmpMotor = WhichMotor(Motor);
  digitalWrite(tmpMotor.Pin1,LOW);
  digitalWrite(tmpMotor.Pin2,HIGH);
  analogWrite(tmpMotor.SpeedPin, Speed);
}

void StopSpinning(int Motor) {
  MotorPins tmpMotor = WhichMotor(Motor);
  digitalWrite(tmpMotor.Pin1,LOW);
  digitalWrite(tmpMotor.Pin2,LOW);
  analogWrite(tmpMotor.SpeedPin, 0);
}

// Defining functions to make the car move in different directions
void GoSoftLeft(int Speed) {
  Backward(LM, 1); // Can Change Speed Value
  Forward(RM, Speed); // Can Change Speed Value
}

void GoSharpLeft(int Speed) {
  Backward(LM, Speed); // Can Change Speed Value
  Forward(RM, Speed); // Can Change Speed Value
}

void GoRight(int Speed) {
  Forward(LM, Speed); // Can Change Speed Value
  Backward(RM, 1); // Can Change Speed Value
}

void GoSharpRight(int Speed) {
  Forward(LM, Speed); // Can Change Speed Value
  Backward(RM, Speed); // Can Change Speed Value
}

void GoStraight(int Speed) {
  Forward(LM, Speed/1.05); // Can Change Speed Value
  Forward(RM, Speed/1.05); // Can Change Speed Value
}

void Stop() {
  StopSpinning(LM);
  StopSpinning(RM);
}

// Defining functions for LEDs
void LEDOn(int LED) {
  LEDPins tmpLED = WhichLED(LED);
  digitalWrite(tmpLED.Pin,LOW);
}

void LEDOff(int LED) {
  LEDPins tmpLED = WhichLED(LED);
  digitalWrite(tmpLED.Pin,HIGH);
}

int LEDBlink(int LED, int BlinkingOn) {
  LEDPins tmpLED = WhichLED(LED);
  if (BlinkingOn == True) {
    digitalWrite(tmpLED.Pin,HIGH);
    BlinkingOn = False;
    delay(BlinkSpeed);
    return BlinkingOn;
  }
  else {
    digitalWrite(tmpLED.Pin,LOW);
    BlinkingOn = True;
    delay(BlinkSpeed);
    return BlinkingOn;
  }
}

// SETUP - RUNS ONCE
// ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
void setup() {
  // Set all the motor control pins & LEDs to outputs and sensors to inputs
  pinMode(LeftIR, INPUT);
  pinMode(RightIR, INPUT);
  pinMode(ORIR, INPUT);
  pinMode(OLIR, INPUT);
  pinMode(LSpeedPin, OUTPUT);
  pinMode(RSpeedPin, OUTPUT);
  pinMode(LMotor1, OUTPUT);
  pinMode(LMotor2, OUTPUT);
  pinMode(RMotor1, OUTPUT);
  pinMode(RMotor2, OUTPUT);
  pinMode(IRReciver, INPUT);
  pinMode(LLED, OUTPUT);
  pinMode(RLED, OUTPUT);

  // Turn off motors - Initial state
  digitalWrite(LMotor1, LOW);
  digitalWrite(LMotor2, LOW);
  digitalWrite(RMotor1, LOW);
  digitalWrite(RMotor2, LOW);

  // Start IR receiver
  IrReceiver.begin(IRReciver);
}

// MAIN CODE - LOOPS
// ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
void loop() {
  // Read line sensors (Pin IR)
  int RightIRVal = digitalRead(RightIR);
  int LeftIRVal = digitalRead(LeftIR);
  int ORVal = digitalRead(ORIR);
  int OLVal = digitalRead(OLIR);

  // Starting value for "blinking" for command
  static int BlinkingOn = False;

  // Determine if remote has signaled to start/stop
  static int RemoteTurnedOn = False;
  if (IrReceiver.decode()) {
    IrReceiver.resume();
    int command = IrReceiver.decodedIRData.command;
    switch (command) {
      case IR_OK: {
        RemoteTurnedOn = True;
        break;
      }
      case IR_0: {
        RemoteTurnedOn = False;
        break;
      }
      default: {
      }
    }
  }
  if (RemoteTurnedOn == False) {
    Stop();
    LEDOff(LeftLED);
    LEDOff(RightLED);
  }

  // If the outer sensors detect the line (and remote on)
  else if (ORVal == HIGH) {
    GoSharpRight(MotorSpeed);
    BlinkingOn = LEDBlink(RightLED, BlinkingOn);
  }
  else if (OLVal == HIGH) {
    GoSharpLeft(MotorSpeed);
    BlinkingOn = LEDBlink(LeftLED, BlinkingOn);
  }
  // If no sensors detect black line (and remote on)
  else if (RightIRVal == LOW && LeftIRVal == LOW) {
    MotorSpeed = 100; // Can Change MotorSpeed
    GoStraight(MotorSpeed);
    LEDOn(LeftLED);
    LEDOn(RightLED);
  }
  // If left detects black line (and remote on)
  else if (RightIRVal == LOW && LeftIRVal == HIGH) {
    MotorSpeed = 100; // Can Change MotorSpeed
    GoLeft(MotorSpeed);
    BlinkingOn = LEDBlink(LeftLED, BlinkingOn);
    LEDOn(RightLED);
  }
  // If right detects black line (and remote on)
  else if (RightIRVal == HIGH && LeftIRVal == LOW) {
    MotorSpeed = 100; // Can Change MotorSpeed
    GoRight(MotorSpeed);
    LEDOn(LeftLED);
    BlinkingOn = LEDBlink(RightLED, BlinkingOn);
  }
}
```

Initialization: Libraries, definitions, variables, etc. - referenced in other code

Setup: This code runs once

Loop: This code loops repeatedly

Including a library for functionality of the IR remote

Linking variables/names to the Arduino's digital pins - improves readability

Linking variables/names to the signals received from the IR remote when certain buttons are pressed

Defining variables to make the code easier to read and understand

Defining global structures and initializing global variables

Defining functions that make later functions easier to create and read/understand

Defining functions that make the motors spin

Defining functions that coordinate the motors to make the car as a whole move

Defining functions to make the LED 'headlights/turn signals' turn on and off

Setup function - runs once to 'prepare' for the loop by setting the different Arduino pins to input/output, making the motors still, and turning on the IR receiver for the remote

At the beginning of each iteration of the loop - checking if the IR sensors detect the line and if the IR receiver has received any signals from the remote

If no buttons on the remote have been pressed, or the "0" button was the most recent button press, the robot stops and the 'headlights' turn off

Decides how the robot should move based on input from the IR sensors