SENSOR FOCUSED TRAFFIC LIGHT

Allan Farai Makwembere 2108418

***To design a traffic light system that senses car movements without the use of a camera.***

# Project Description

The object is a traffic light sensor system that warns and detects cars when they move beyond a certain distance or move whilst the traffic light is on red **without the use of a camera**. This is specifically useful in busy roads or accident-prone roads that require extra monitoring.

The components used are:

* Ultrasonic distance sensor: This would be used to calculate the distance between the car and the traffic light.
* PIR sensor: assists the ultrasonic sensor. They work in tandem with each other. It only activates when the car is within a certain distance of the ultrasound sensor.
* Wires: These would link all the components together and be used for communication between the Arduino’s by way of the TX and RX ports.
* Two Arduino boards: They facilitate the entire device by providing connecting ports, communication, a programmable interface and power.
* Breadboard: This provides organization through more ports for the wires and devices so that they are not scattered around the Arduino.
* Potentiometer: This would act as a variable resistor for the ground port inside the LCD display so that it receives just the right amount of power to operate. It can however be adjusted readily to reduce current.
* Resistor: This is used to provide resistance to the current that flows into the LED. Where the potentiometer helped the ground port this resistor is specifically for the LED lights within the display.
* LCD 16 X 2: this would be the initial mode of communication which warns when a car gets past a certain distance or moves through a red traffic light
* LED: They will act as the traffic lights and the entire program will be controlled by the colour of the LED.

Graphical user interface, text, application, email

Description automatically generated

Therefore, the components would then work together to deliver a program that runs by way of the following diagram:

Graphical user interface, chart

Description automatically generated with medium confidence

# Circuitry of Device

Diagram

Description automatically generated

Diagram, schematic

Description automatically generated

For my 3 LEDs I used a resistance of 220ohms for each. The Arduino Pins used to feed into the LEDs have a voltage of 5V. The LEDs have a voltage drop of 1.8V and a current of about 25mA

Thus, the values for the resistance were obtained from the formula: Resistance = Voltage/Current.

Which would be: 5V – 1.8V = 3.2V / 25mA = 3.2 / 0.025 = 128 ohms. However, it is better to round it off to a more practical figure of 220ohms in order to also protect the Microcontroller pin as well.

For the LCD, I used a potentiometer which has a variable resistance which can be tuned. It is particularly useful for the LCD since it has a collection of LEDs within it.

For the ultrasonic sensor I used inches and centimetres for the distance.

# Program

For the program it is split up into two sections, as one contains code for the Arduino with the sensors and the other Arduino is for the LCD display and LEDs

## LCD & LED CODE

// include the library code:

#include <LiquidCrystal.h>

char monitor[3];

int seconds = 0;

int amber = 8;

int red = 10;

int green = 6;

bool amber\_on = false;

char red\_pos[9] = "carstops";

char green\_pos[9] = "car-goes";

byte row;

// initialize the library by associating any needed LCD interface pin with the arduino pin number it is connected to.

const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;

LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

void setup() {

row = 0;

pinMode(amber, OUTPUT);

pinMode (red, OUTPUT);

pinMode (green, OUTPUT);

Serial.begin(9600);

lcd.begin(16, 2);

}

void loop() {

do {

digitalWrite(red, HIGH);

Serial.write(red\_pos, 9);

delay(10000);

digitalWrite(amber, HIGH);

delay(1000);

digitalWrite(amber, LOW);

digitalWrite(red, LOW);

digitalWrite(green, HIGH);

Serial.write(green\_pos, 9);

delay(10000);

digitalWrite(green, LOW);

digitalWrite(amber, HIGH);

amber\_on = true;

delay(3000);

digitalWrite(amber, LOW);

//Here the code allows the LEDs to alternate between red, amber and green each taking a fixed amount of time before switching off.

}

while(amber\_on == true);

lcd.clear();

lcd.setCursor(0, row);

lcd.print("Trafffic");

row = ! row;

Serial.readBytes(monitor,3); //Read the serial data and store in var monitor

if (monitor == "go"){

//reads the transmitted data and checks if it allows the car to proceed which will then show on the lcd.

lcd.clear();

lcd.setCursor(0, row);

lcd.print("CarCanGo");

row = ! row;

delay(1000);}

else if (monitor == "no"){

// this shows if car cannot proceed.

lcd.clear();

lcd.setCursor(0, row);

lcd.print("CarCantGo");

row = ! row;

delay(1000);}

}

## Sensors Code

char monitor[3] = ("go");

char monitor2[3] = ("no");

char red\_pos[9];

char green\_pos[9];

int inches = 0;

int a = 0;

int b = 0;

int cm = 0;

long readUltrasonicDistance(int triggerPin, int echoPin)

{

pinMode(triggerPin, OUTPUT); // Clear the trigger

digitalWrite(triggerPin, LOW);

delayMicroseconds(2);

// Sets the trigger pin to HIGH state for 10 microseconds

digitalWrite(triggerPin, HIGH);

delayMicroseconds(10);

digitalWrite(triggerPin, LOW);

pinMode(echoPin, INPUT);

// Reads the echo pin, and returns the sound wave travel time in microseconds

return pulseIn(echoPin, HIGH);

}

void setup() {

// Begin the Serial at 9600 Baud

Serial.begin(9600);

}

void loop() {

Serial.readBytes(red\_pos, 9);

Serial.readBytes(green\_pos, 9);

Serial.println(red\_pos);

// measure the ping time in cm

cm = 0.01723 \* readUltrasonicDistance(7, 7);

// convert to inches by dividing by 2.54

inches = (cm / 2.54);

Serial.print(inches);

Serial.print("in, ");

Serial.print(cm);

Serial.println("cm");

delay(100); // Wait for 100 millisecond(s)

if (red\_pos == "carstops"){

if (inches < 5 ){

//here the car has already passed the red traffic light just as it changes and there is no use in stopping it.

Serial.write(monitor, 3);}

else if(inches > 5){

//this means the car has not gone beyond the traffic light but it is at a safe manageable distance

a = analogRead(A0);

b = map(a,0,1023,0,255);

// the analogue is stored in variable a. whilst all our data would be mapped intom b where it is read up 1024 times per second.

Serial.println(b);

if (b > 100){

//if the car moves through the sensor it has made a traffic offense and gone through a red light.

Serial.println("TrafficOffense");

Serial.write(monitor2, 3);

delay(100);

}

}

}else if (green\_pos == "car-goes"){

Serial.write(monitor, 3);

Serial.println("Car can go");

}

}

## TESTING

To compare my test data, I used several values and inputs against other values. However due to the device’s usage of several lines of code and logic, the Arduino which has limited memory (**The Arduino UNO has only 32K bytes of Flash memory and 2K bytes of SRAM)** was unable to quickly process all the code and experienced slow processing speed. Thus, it did not work as best as intended. Despite the functionality of the code and circuit design it could not display output on the LCD monitor but could display on the serial monitor.

The outputs are illustrated in the table below:

|  |  |  |
| --- | --- | --- |
| CAR/ OBJECT POSITION | TRAFFIC LIGHT | SERIAL OUTPUT |
| 1. <5inches | red | “carstops” sensors in default |
| 1. 1 inch from traffic light | Red | “car-stops” 1 inch 3cm |
| 1. 11.7 inches from traffic light   Pir sensor input | red | “car-stops” |
| 1. ----- | green | “car-stops” |
| 1. ---- | Green to amber | “car-stops” |
| 1. ---- | Red to amber | “carstops” sensors in default |

* Since the program is controlled by a continuous change in traffic lights and signals, the outputs too had to be controlled by this system.
* The first output deals with the code as it starts or as it is in default mode.
* The second output infers from this part of the code:

if (red\_pos == "carstops"){

if (inches < 5 ){

Serial.write(monitor, 3);

Serial.println(green\_pos);}

* The third output infers from this part of the code

else if(inches > 5){

a = analogRead(A0);

b = map(a,0,1023,0,255);

Serial.println(b);

if (b > 100){

Serial.println("TrafficOffense");

Serial.write(monitor2, 3);

delay(100);

}

* The fourth to last outputs deal with this part of the code:

}else if (green\_pos == "car-goes"){

Serial.write(monitor, 3);

Serial.println("Car can go");

}

Results and Findings

However, the code and device did not work as planned. The test results provided insufficient output to make the system functional. For instance, as the car approaches the traffic lights the output is fixed to the results in the serial monitor below.

Graphical user interface, text

Description automatically generated

Some of the code is ignored perhaps due to insufficient memory and the device does not produce the expected output.

The other issue is that the system does not catch or detect when a vehicle passes through at a red traffic light. Where it is supposed to say, “Traffic offense” it is merely giving the output “car-stops”.

As a result, several amendments to the code and system are needed.

These new changes will be:

1. Reducing the code in order to speed up the program.
2. Removing the LCD display in order to allocate more power and memory to the sensors and LED.
3. Amending the code to display the needed values and output in the Serial monitor.
4. Using the PIR sensor for detecting a vehicle instead of the ultrasonic sensor and not for movement.

The new design is therefore:

Diagram

Description automatically generated

The circuitry for the new device is:

Diagram

Description automatically generated with medium confidence

It still uses the same resistor values and power.

The code is still divided into two:

## CODE FOR THE TRAFFIC LIGHTS:

char pir[13];

int amber = 8;

char colour[4] = "red";

//store the red traffic light data

int red = 10;

int green = 6;

bool amber\_on = false;

void setup(){

pinMode(amber, OUTPUT);

pinMode (red, OUTPUT);

pinMode (green, OUTPUT);

Serial.begin(9600);

}

void loop(){

do {

digitalWrite(red, HIGH);

Serial.write(colour, 4);

delay(10000);

digitalWrite(amber, HIGH);

delay(1000);

digitalWrite(amber, LOW);

digitalWrite(red, LOW);

digitalWrite(green, HIGH);

delay(10000);

digitalWrite(green, LOW);

digitalWrite(amber, HIGH);

amber\_on = true;

delay(3000);

digitalWrite(amber, LOW);

}

while(amber\_on == true);

Serial.readBytes(pir, 13);

Serial.println(pir);

}

The above code retains the same functionality and structure with a few tweaks.

## CODE FOR THE SENSORS:

char pir[13] = "Car detected"; //String data

char red[4];

int inches = 0;

int cm = 0;

int infrared = 7;

long readUltrasonicDistance(int triggerPin, int echoPin)

{

pinMode(triggerPin, OUTPUT); // Clear the trigger

digitalWrite(triggerPin, LOW);

delayMicroseconds(2);

// Sets the trigger pin to HIGH state for 10 microseconds

digitalWrite(triggerPin, HIGH);

delayMicroseconds(10);

digitalWrite(triggerPin, LOW);

pinMode(echoPin, INPUT);

// Reads the echo pin, and returns the sound wave travel time in microseconds

return pulseIn(echoPin, HIGH);

}

void setup() {

// Begin the Serial at 9600 Baud

pinMode(infrared, INPUT);

Serial.begin(9600);

}

void loop() {

infrared = digitalRead(7);

if (infrared == HIGH){

Serial.println("Car detected");

Serial.write(pir, 13);

// measure the ping time in cm

cm = 0.01723 \* readUltrasonicDistance(8, 8);

// convert to inches by dividing by 2.54

inches = (cm / 2.54);

Serial.print(inches);

Serial.print("in, ");

Serial.print(cm);

Serial.println("cm");

Serial.readBytes(red, 4);

delay(100); // Wait for 100 millisecond(s)

if (inches < 5 || red == "red" ){

//checks to see how far from the traffic light the car is.

//If the object meets the above expression it has passed the red traffic light

Serial.println("TrafficOffense");

}

}

else{

Serial.println("No car detected");

}

delay(10);

}

The above code differs from the previous prototype in that it is now controlled by the PIR sensor instead of the ultrasound sensor. This means that when a car is detected by the PIR sensor, it registers this and then proceeds to activate the ultrasonic sensor which then checks the distance of the car to the traffic light. If the distance is below 5 inches it means that the car has passed a red traffic light.

## TEST DATA

The new test data for the device is as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| PIR | ULTRA-SOUND DISTANCE | TRAFFIC LIGHT | OUTPUT |
| 1. No car detected   (nothing passed though infra-red) | null | red | Funnel chart  Description automatically generated with medium confidence |
| 1. Car detected (passed something through infrared | 50inches 128 cm | red |  |
| 1. Car detected | 4inches 12cm | red |  |

Because the previous code was not working in the above sections shown in the table, I only illustrated that section since the other part that involves the green and amber traffic light was working as it should.

The first output deals with the program in its default state without a car passing through.

The second output deals with the program having a car pass through the infra-red sensor. It detects the car but because the car does not move further, it means it has stopped and has not made an offense.

The third part shows a car that is detected but proceeds to move beyond the required distance which is picked up by the ultrasonic sensor and a signal is sent which then classes the action as a “Traffic offense”.

## Conclusion.

Where the traffic light system had started off with the intention of using sensors alone to detect misbehaving drivers, the output and result has convinced me that it would be better off to use a camera as opposed to the ultrasonic and infra-red sensors I used.

My circuit consisted of the needed components according to the conventional needs like the infra-red sensor and an ultrasonic sensor. My initial prototype failed to meet the expectations due to the limitations of hardware. As a result, the prototype had to be changed to one which is more functional.

The final prototype would then detect a car using the infra-red sensor, once a car is detected it would then monitor the car with the ultra-sound sensors. This monitors it for movement beyond its traffic limitations. Should there be movement the output would then be shown in the serial monitor and a camera would then be authorized to take a picture of the car, or penalty system would be used to control the driver.

However, despite the new changes, the device no longer fulfilled its intended purpose of being less dependent on a camera. It therefore became like any ordinary traffic light.

Overall, I found the task to be helpful even though the final design was not as intended. Other prototypes of the same device just include the altering LEDs and do not include a system that monitors the vehicle whilst others are entirely dependent on a camera. As a result, the inclusion of the different sensors allowed the device to be unique compared to the others already available.

Project link

https://www.tinkercad.com/things/ffSaEJawX1s?sharecode=ImMNtRANVZco\_xKZATYT3Zu6VVQXyD-f4f1hfDLRF7s