**Traffic Light Simulation**

1. **Abstract:**

This project uses Ultrasonic Sensor HC-SR04. There are two groups of red, yellow and green lights that represents the traffic lights on a road intersection. The project is made to determine new ways of improving the traffic in we live today.

**Introduction to Sensor :**

The HC-SR04 ultrasonic sensor uses sonar signals to determine distance to an object. It offers excellent range accuracy and stable readings in an easy-to-use package. Its operation is not affected by sunlight or black material. However, acoustically soft materials like cloth can be difficult to detect. It uses sonar to detect objects at a distance of 2 cm to 4 meters.  This sensor is widely used in robotics to build robots that move and should divert or avoid obstacles.

Moreover, it is easy to control it through the Arduino, because it only has 4 pins described below.

* **VCC** – 5V (ranging from 4.5 V to 5.5 V)
* **Trig** – Sensor input (trigger)
* **Echo** – Output Sensor (Echo)
* **GND** – Ground



1. **Project Scope:**

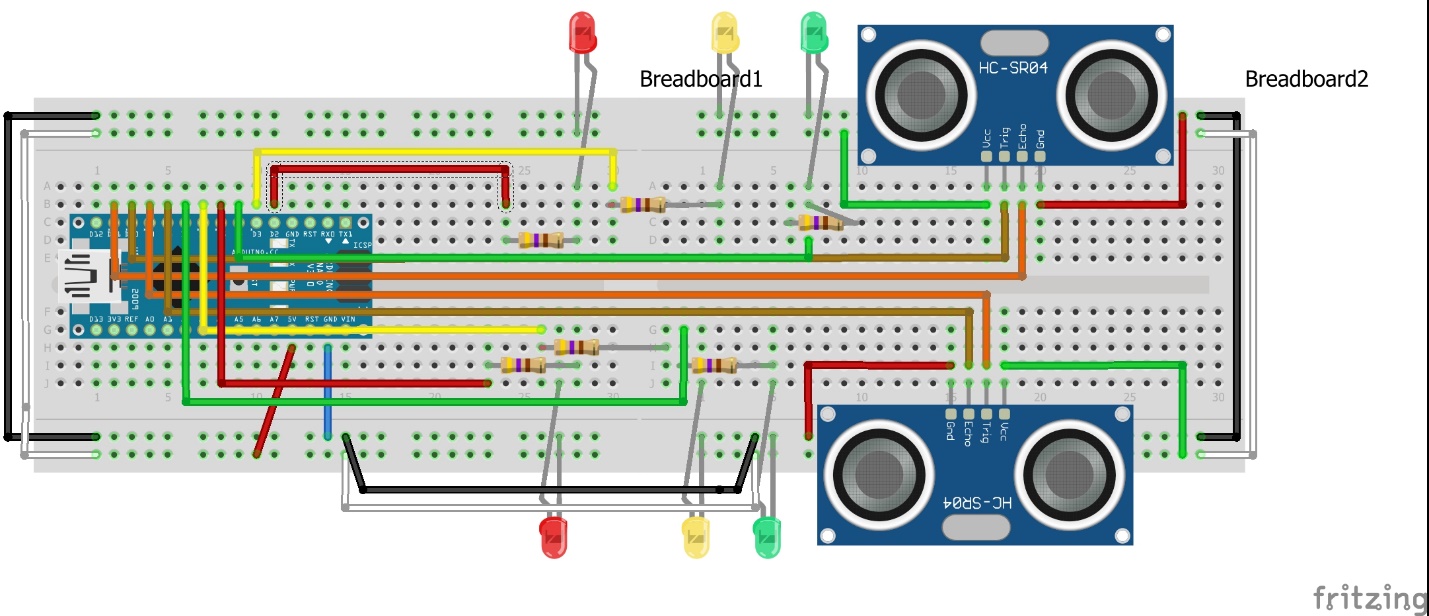
This device will register traffic data as well as a simulation for improving the way traffic lights work today to be faster and smarter.

1. **Components Required:**

* Two HC- SR04
* Six resistors of 470 ohms
* Jumper wires;
* Six LEDs of different colors;
* Two Breadboard
* Toy Cars
* Drawing of road

1. **Connections:**

* Digital ports 10, 11 ,8 and 9 of the resistors connected Arduino, which in turn are connected the legs of the LEDs positive;
* Cathode pin of the LED in the GND (ground) of the Arduino;
* VCC pin of the ultrasonic sensor HC-SRO4 in Arduino 5V;
* TRIG pin sensor HC-SRO4 digital port on the Arduino 6;
* ECHO pin sensor HC-SRO4 the digital port 7 of the Arduino;
* GND pin HC-sensor SRO4 in Arduino GND.



1. **Working:**

A short ultrasonic pulse is transmitted at the time 0, reflected by an object. The sensor receives this signal and converts it to an electric signal. The next pulse can be transmitted when the echo is faded away. This time period is called cycle period. The recommended cycle period should not be less than 50ms. If a 10μs width trigger pulse is sent to the signal pin, the Ultrasonic module will output eight 40kHz ultrasonic signal and detect the echo back. The measured distance is proportional to the echo pulse width and can be calculated by the formula above. If no obstacle is detected, the output pin will give a 38ms high level signal.

* If distance is greater than 20 cm, then LED Green will be turned ON.
* If distance is less than & equal to 20 cm and greater than & equal to 10, then LED Yellow will be turned ON.
* If distance is less than 10 cm, then LED Red will be turned ON.

1. **Source Code:**

#include "Ultrasonic.h"

Ultrasonic ultrasonic(8,9);

//-------------------------------------------------------------------------------

//-------------------------------------------------------------------------------

// The idea of this project is to control the traffic lights of a 4 way junction

// with integrated range finders for detecting the presence of cars to allow them to cross.

int red1 = 2; // red LED 1 - pin 2

int yellow1 = 3; // yellow LED 1 - pin 3

int green1 = 4; // greed LED 1 - pin 4

int red2 = 5; // red LED 2 - pin 5

int yellow2 = 6; // yellow LED 2 - pin 6

int green2 = 7; // green LED 2 - pin 7

unsigned long PreviousMillis = 0; // previous millisecond

unsigned long CurrentMillis; // current millisecond

int atraso; // delay

int state; // state for FSM

const int trigPin = 9; // ping 1

const int echoPin = 8; // ping 2

const int trigPin2 = 11; // ping 1

const int echoPin2 = 10; // ping 2

int cross=0; // presence of cars 1

int cross2=0; // presence of cars 2

long duration, cm, cm2; // range calculation variables

long microsec = 0;

float distanceCM = 0;

void setup ()

{

pinMode(red1, OUTPUT); //all LED pins are set to "output"

pinMode(yellow1, OUTPUT);

pinMode(green1, OUTPUT);

pinMode(red2, OUTPUT);

pinMode(yellow2, OUTPUT);

pinMode(green2, OUTPUT);

state=1; // initiate the FSM in the first state

Serial.begin(9600); // initialize serial communication

}

void loop ()

{

long duration, inches, cm;

CurrentMillis = millis(); // updates current millisecond

atraso=CurrentMillis-PreviousMillis; // calculates the delay

ping(); // runs the first ping function

ping2(); // runs the second ping function

next\_state\_FSM(); // runs the FSM function (next state)

output\_FSM(); // runs the FSM function (output)

//notice the loop is almost empty due to the implementation of an FSM

duration = pulseIn(echoPin, HIGH);

// convert the time into a distance

inches = microsecondsToInches(duration);

cm = microsecondsToCentimeters(duration);

delay(1000);

}

void ping(){

// The PING))) is triggered by a HIGH pulse of 2 or more microseconds.

// Give a short LOW pulse beforehand to ensure a clean HIGH pulse:

pinMode(trigPin, OUTPUT);

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

digitalWrite(trigPin, HIGH);

delayMicroseconds(5);

digitalWrite(trigPin, LOW);

// The same pin is used to read the signal from the PING))): a HIGH

// pulse whose duration is the time (in microseconds) from the sending

// of the ping to the reception of its echo off of an object.

pinMode(echoPin, INPUT);

duration = pulseIn(echoPin, HIGH);

// convert the time into a distance

cm = microsecondsToCentimeters(duration);

if(cm<10){ //in reallity it would be aproximatly 100cm but 10cm is more practical for testing the circuit on a small scale

cross=1; //if the cars is close enough the "cross" variable will change to 1

}else{

cross=0; //"cross" will be 0 when there is no presence of cars

}

}

void ping2(){

// The PING))) is triggered by a HIGH pulse of 2 or more microseconds.

// Give a short LOW pulse beforehand to ensure a clean HIGH pulse:

pinMode(trigPin2, OUTPUT);

digitalWrite(trigPin2, LOW);

delayMicroseconds(2);

digitalWrite(trigPin2, HIGH);

delayMicroseconds(5);

digitalWrite(trigPin2, LOW);

// The same pin is used to read the signal from the PING))): a HIGH

// pulse whose duration is the time (in microseconds) from the sending

// of the ping to the reception of its echo off of an object.

pinMode(echoPin2, INPUT);

duration = pulseIn(echoPin2, HIGH);

// convert the time into a distance

cm2 = microsecondsToCentimeters(duration);

if(cm2<10){ //in reallity it would be aproximatly 100cm but 10cm is more practical for testing the circuit on a small scale

cross2=1; //if the cars is close enough the "cross2" variable will change to 1

}else{

cross2=0; //"cross2" will be 0 when there is no presence of cars

}

if (cm<30 && cm>2) //Limit the range of when displaying the correct distance in cm

{

Serial.print(cm);

Serial.print("cm, ");

}

else

{

Serial.print("Sensor 1 ERROR! Not in proper range, ");

}

if (cm2<30 && cm2>2)

{

Serial.print(cm2);

Serial.print("cm2");

}

else

{

Serial.print("Sensor 2 ERROR! Not in proper range");

}

Serial.println();

}

void output\_FSM(){

switch (state){

case 1: //LED outputs are defined for state 1

digitalWrite (red1, LOW);

digitalWrite (yellow1, LOW);

digitalWrite (green1, HIGH);

digitalWrite (red2, HIGH);

digitalWrite (yellow2, LOW);

digitalWrite (green2, LOW);

break;

case 2: //LED outputs are defined for state 2

digitalWrite (red1, LOW);

digitalWrite (yellow1, HIGH);

digitalWrite (green1, LOW);

digitalWrite (red2, HIGH);

digitalWrite (yellow2, LOW);

digitalWrite (green2, LOW);

break;

case 3: //LED outputs are defined for state 3

digitalWrite (red1, HIGH);

digitalWrite (yellow1, LOW);

digitalWrite (green1, LOW);

digitalWrite (red2, HIGH);

digitalWrite (yellow2, LOW);

digitalWrite (green2, LOW);

break;

case 4: //LED outputs are defined for state 4

digitalWrite (red1, HIGH);

digitalWrite (yellow1, LOW);

digitalWrite (green1, LOW);

digitalWrite (red2, LOW);

digitalWrite (yellow2, LOW);

digitalWrite (green2, HIGH);

break;

case 5: //LED outputs are defined for state 5

digitalWrite (red1, HIGH);

digitalWrite (yellow1, LOW);

digitalWrite (green1, LOW);

digitalWrite (red2, LOW);

digitalWrite (yellow2, HIGH);

digitalWrite (green2, LOW);

break;

case 6: //LED outputs are defined for state 6

digitalWrite (red1, HIGH);

digitalWrite (yellow1, LOW);

digitalWrite (green1, LOW);

digitalWrite (red2, HIGH);

digitalWrite (yellow2, LOW);

digitalWrite (green2, LOW);

break;

case 7: //LED outputs are defined for state 7

digitalWrite (red1, LOW);

digitalWrite (yellow1, LOW);

digitalWrite (green1, HIGH);

digitalWrite (red2, HIGH);

digitalWrite (yellow2, LOW);

digitalWrite (green2, LOW);

break;

case 8: //LED outputs are defined for state 8

digitalWrite (red1, HIGH);

digitalWrite (yellow1, LOW);

digitalWrite (green1, LOW);

digitalWrite (red2, LOW);

digitalWrite (yellow2, LOW);

digitalWrite (green2, HIGH);

break;

}

}

void next\_state\_FSM(){ //this function defines when to change state and which state to change to based on the delay value and the presence of cars

switch (state){

case 1:

if(atraso>10000){

state=2;

PreviousMillis=CurrentMillis;

}else{

if(cross==1){

state=7; // does not reset previous millis so the "timer" can keep counting

}

}

break;

case 2:

if(atraso>4000){

state=3;

PreviousMillis=CurrentMillis;

}

break;

case 3:

if(atraso>4000){

state=4;

PreviousMillis=CurrentMillis;

}

break;

case 4:

if(atraso>10000){

state=5;

PreviousMillis=CurrentMillis;

}else{

if(cross2==1){

state=8;

}

}

break;

case 5:

if(atraso>4000){

state=6;

PreviousMillis=CurrentMillis;

}

break;

case 6:

if(atraso>4000){

state=1;

PreviousMillis=CurrentMillis;

}

break;

case 7: // auxiliar state to make the green light go out sooner

if(cross==0){

state=1; // does not reset previous millis so the "timer" can keep counting

}else{

if(atraso>1000){

state=2;

PreviousMillis=CurrentMillis;

}

}

break;

case 8: // auxiliar state to make the green light go out sooner

if(cross2==0){

state=4; // does not reset previous millis so the "timer" can keep counting

}else{

if(atraso>1000){

state=5;

PreviousMillis=CurrentMillis;

}

}

break;

}

}

long microsecondsToCentimeters(long microseconds) {

// The speed of sound is 340 m/s or 29 microseconds per centimeter.

// The ping travels out and back, so to find the distance of the

// object we take half of the distance travelled.

return microseconds / 29 / 2;

}

//-------------------------------------------------------------------------------

//-------------------------------------------------------------------------------

long microsecondsToInches(long microseconds)

{

// According to Parallax's datasheet for the PING))), there are

// 73.746 microseconds per inch (i.e. sound travels at 1130 feet per

// second). This gives the distance travelled by the ping, outbound

// and return, so we divide by 2 to get the distance of the obstacle.

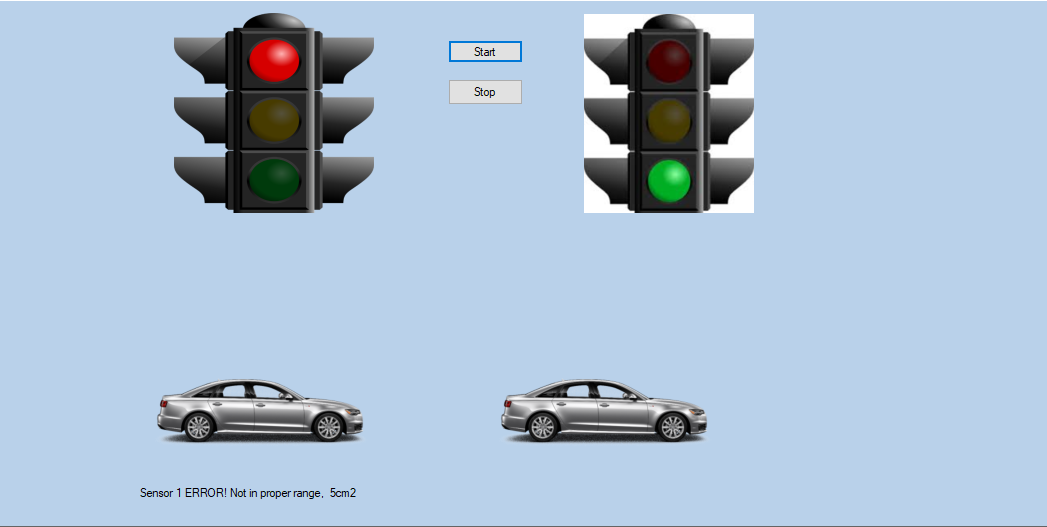
return microseconds / 74 / 2;

}

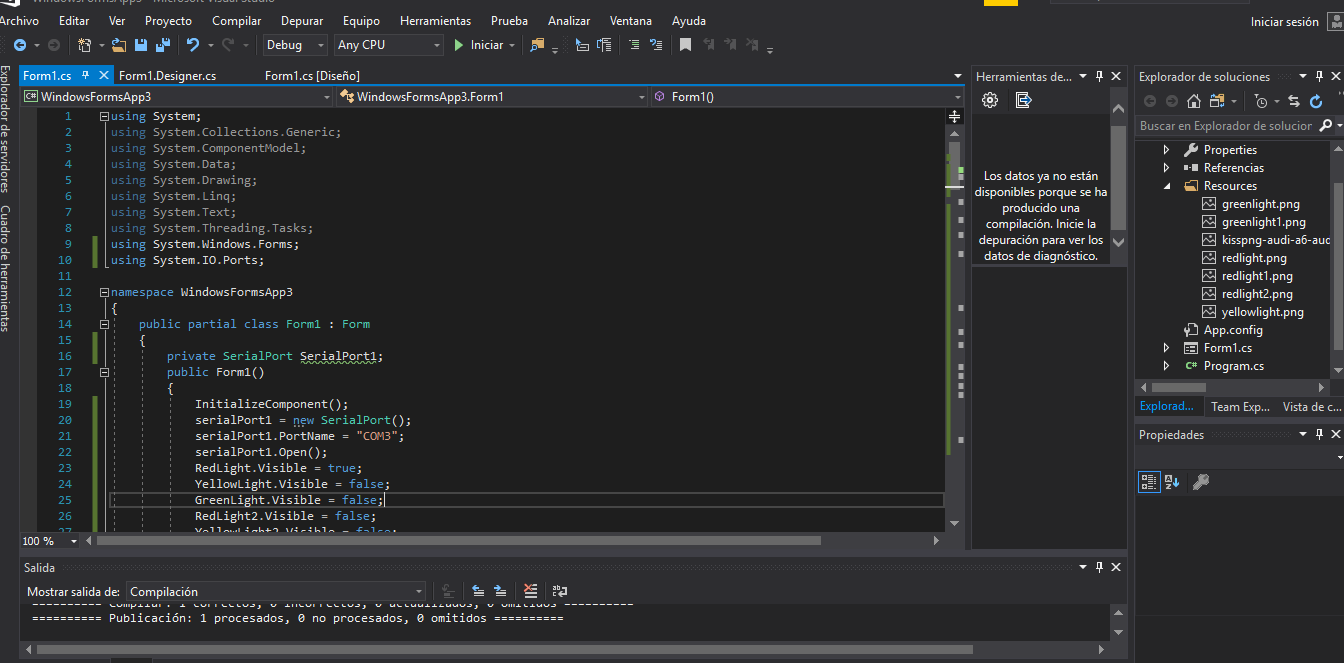
//-------------------------------------------------------------------------------

//-------------------------------------------------------------------------------

1. **Project Shots:**
2. Main Screen of traffic lights in Visual Studio



1. Interface of *Port Settings.*



1. Console Output:

