**Documentatie**

-Sistem de monitorizare a parcarii unei masini-

Realizat de:

Florea Ionut

Stahie Alexandru

1. **Scurta descriere:**

Proiectul se bazeaza pe legarea a 5 senzori ultrasonici in paralel pe o bara de lemn (aceasta reprezentand bara de protectie din spate a masinii), la distante egale unul de altul. Acestia sunt conectati la o placa de dezvoltare Arduino Uno, cu ajutorul careia vom trimite datele receptionale de la ei prin comunicatie seriala la o aplicatie ce va rula pe laptop si care, va afisa valorile returnate de senzori impreuna cu o interfata grafica pentru utilizotor mai usor de observat in timpul unei parcari.



Figure 1: Senzor ultrasonic HC-SR04

1. **Componente folosite:**
2. 5 senzori ultrasonici HC-SR04 (Figure 1):

* Caracteristici tehnice:
  + Tensiune de alimentare: 5V;
  + Curent consumat: 15Ma;
  + Distanta de functionare: 2cm-4m;
  + Unghiul de masurare: 15o;
  + Eroare de doar 3mm;
  + Durata semnal input 10us;
  + Dimensiuni: 45mm x 20mm x 15mm;
* Avantajele acestui senzor: este compatibil cu Arduino, usor de folosit, necesita doar pini I/O digitali si are o imunitate mai mare la zgomot.

1. Placa de dezvoltare Arduino Uno(Figure 2):

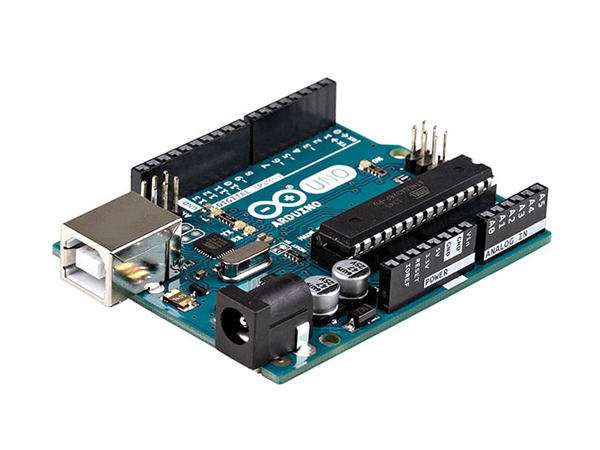
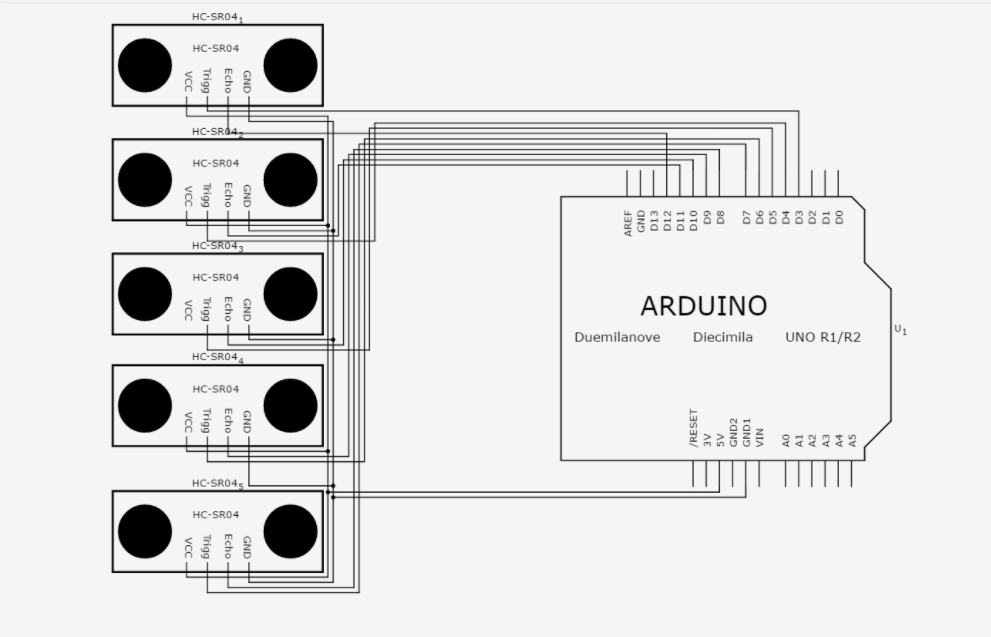


Figure 2:Placa Arduino Uno

1. Cablu USB pentru comunicatia seriala dintre laptop si placa Arduino
2. **Schema de circuit:**



1. **Programul Arduino:**

#include <NewPing.h>

#define SONAR\_NUM 5 // Number of sensors.

#define MAX\_DISTANCE 150 // Maximum distance (in cm) to ping.

#define PING\_INTERVAL 40 // Milliseconds between sensor pings (29ms is about the min to avoid cross-sensor echo).

unsigned long pingTimer[SONAR\_NUM]; // Holds the times when the next ping should happen for each sensor.

unsigned int cm[SONAR\_NUM]; // Where the ping distances are stored.

uint8\_t currentSensor = 0; // Keeps track of which sensor is active.

NewPing sonar[SONAR\_NUM] = { // Sensor object array.

NewPing(3, 12, MAX\_DISTANCE), // Each sensor's trigger pin, echo pin, and max distance to ping.

NewPing(4, 11, MAX\_DISTANCE),

NewPing(5, 10, MAX\_DISTANCE),

NewPing(6, 9, MAX\_DISTANCE),

NewPing(7, 8, MAX\_DISTANCE)

};

void setup() {

Serial.begin(9600);

while (true) {

Serial.println("ACK,");

delay(250);

if (first\_step()) {

break;

}

}

setPingTime();

}

void loop() {

for (uint8\_t i = 0; i < SONAR\_NUM; i++) { // Loop through all the sensors.

if (millis() >= pingTimer[i]) { // Is it this sensor's time to ping?

pingTimer[i] += PING\_INTERVAL \* SONAR\_NUM; // Set next time this sensor will be pinged.

if (i == 0 && currentSensor == SONAR\_NUM - 1)

oneSensorCycle(); // Sensor ping cycle complete, do something with the results.

sonar[currentSensor].timer\_stop(); // Make sure previous timer is canceled before starting a new ping (insurance).

currentSensor = i; // Sensor being accessed.

cm[currentSensor] = 0; // Make distance zero in case there's no ping echo for this sensor.

sonar[currentSensor].ping\_timer(echoCheck); // Do the ping (processing continues, interrupt will call echoCheck to look for echo).

}

}

// Other code that \*DOESN'T\* analyze ping results can go here.

if (Serial.available() > 0) {

if (stop\_action()) {

for (uint8\_t i = 0; i < SONAR\_NUM; i++) {

sonar[i].timer\_stop();

}

while (true) {

if (Serial.available() > 0) {

if (start\_action()) {

setPingTime();

break;

}

delay(50);

}

}

}

}

}

void echoCheck() { // If ping received, set the sensor distance to array.

if (sonar[currentSensor].check\_timer())

cm[currentSensor] = sonar[currentSensor].ping\_result / US\_ROUNDTRIP\_CM;

cm[currentSensor] = cm[currentSensor] + 1;

}

void oneSensorCycle() { // Sensor ping cycle complete, do something with the results.

// The following code would be replaced with your code that does something with the ping results.

Serial.print("START,");

for (uint8\_t i = 0; i < SONAR\_NUM; i++) {

Serial.print(cm[i]);

Serial.print(",");

}

Serial.println("END");

}

bool first\_step() {

if (Serial.available() > 0) {

String data = String(Serial.readString());

data.trim();

int j = 0;

for (int i = 0; i < data.length(); i++) {

if (data[i] == ',') {

if (data.substring(j, i) == "ACK" && i - j <= 3) {

return true;

}

j = i + 1;

}

}

}

return false;

}

bool stop\_action() {

String data = String(Serial.readString());

data.trim();

int j = 0;

for (int i = 0; i < data.length(); i++) {

if (data[i] == ',') {

if (data.substring(j, i) == "STOP" && i - j <= 4) {

return true;

}

j = i + 1;

}

}

return false;

}

bool start\_action() {

String data = String(Serial.readString());

data.trim();

int j = 0;

for (int i = 0; i < data.length(); i++) {

if (data[i] == ',') {

if (data.substring(j, i) == "START" && i - j <= 5) {

return true;

}

j = i + 1;

}

}

return false;

}

void setPingTime() {

pingTimer[0] = millis() + 75; // First ping starts at 75ms, gives time for the Arduino to chill before starting.

for (uint8\_t i = 1; i < SONAR\_NUM; i++) // Set the starting time for each sensor.

pingTimer[i] = pingTimer[i - 1] + PING\_INTERVAL;

}

1. Initializari:

Vom define o serie de variabile ce ne vor ajuta in program cum ar fi:

* #define SONAR\_NUM 5 //numarul de senzori,
* #define MAX\_DISTANCE 150 //distanta maxima de masurare in cm
* #define PING\_INTERVAL 40

//intervalul in milisecunde pentru captarea datelor (minimul ar trebui sa fie 29ms pentru a evita interferenta senzorilor)

* unsigned long pingTimer[SONAR\_NUM]

//un vector in care memoram momentele in care fiecare senzor ar trebui sa capteze date

* unsigned int cm[SONAR\_NUM];

//un vector cu datele masurate de fiecare senzor

* uint8\_t currentSensor = 0;

//o variaibla pentru a tine minte ce senzor este activ

De asemenea vom define un obiect de tip NewPing (libraria folosita) numit sonar unde vom seta pentru fiecare senzor pinii echo, trigger si distanta maxima de masurare.

NewPing sonar[SONAR\_NUM] = {

NewPing(3, 12, MAX\_DISTANCE)

NewPing(4, 11, MAX\_DISTANCE),

NewPing(5, 10, MAX\_DISTANCE),

NewPing(6, 9, MAX\_DISTANCE),

NewPing(7, 8, MAX\_DISTANCE)

};

1. Functia void setup():

In aceasta functie vom sincroniza comunicatia seriala intre programul arduino de pe placuta si programul Windows Form ce ruleaza pe laptop. Sincronizarea are loc prin trimiterea de mesaje “ACK,” (acknowledge me) de la placa la laptop cu ajutorul functiei de print: Serial.println. Acest lucru se va produce pana ce programul de pe laptop va trimite inapoi la placa, un string cu valoarea: “,ACK”, verificat de functia “void first\_step()”. In acest moment sincronizarea dintre laptop si placa pentru comunicatia seriala este gata. In final setandu-se pentu fiecare senzor momentul sau de captare a datelor cu ajutorul functiei setPingTime().

1. Functia void first\_step():

Asa cum am descris si mai sus, aceasta functie are rolul de a primi mesajul de la programul de pe laptop si de a verifica ca acesta este “,ACK”.

1. Functia void echoCheck():

In aceasta functie vom pune valorile tuturor senzorilor in vectorul cm, urmand ca valoarea acestuia sa fie trimis la programul de pe laptop.

1. Functia void oneSensorCycle():

Aici are loc trimiterea valorilor din vectorul cm prin comunicatia seriala sub forma “START, val1, val2, val3, val4, val5, END”

1. Functiile bool stop\_action() si bool start\_action():

De aici putem porni sau opri trimiterea datelor de pe laptop, trimitand mesaje ca “START” sa “STOP”. In caz ca am vrea sa oprim aplicatia.

1. Functia void setPingTime()

Prima masurare va avea loc dupa 75 ms de la sincronizarea prin seriala, dand timp placii arduino pentru o comunicatie buna. Asadar aici se seteaza timpul de start pentru masurare a fiecarui senzor, in functie de intervalul declarant initial : PING\_INTERVAL;

1. Functia “void loop()”

Aceasta functie este cea principala, aici verifiand pentru fiecare senzor daca este timpul sau de a capta date, atunci se va updata noul moment in care senzorul va fi activ, urmand ca dupa un ciclu complet in care toti senzorii capteaza date, acestea sa fie introduse in vectorul cm si trimise la laptop prin comunicatia seriala cu ajutorul functiilor oneSensorCycle() si echoCheck().

In aceasta functie se verifica si daca functiile “stop\_action()” sau “stat\_action()” vor fi apelate iar in cazul in care raspunsul este unul pozitiv, timerul se va opri sau porni, iar intervalul de masurare pentru fiecare senzor va fi recalculat.

1. **Programul Visual Studio Windows Form:**

Acesta aplicatie consta in doua etape, fiecare reprezentata de cate un formular.

1. Etapa 1:

public partial class Connection : Form

{

public Connection()

{

InitializeComponent();

}

private void Connect\_Click(object sender, EventArgs e)

{

Status stat = new Status();

stat.Show();

this.Hide();

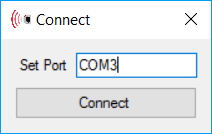
stat.connectionReference = this;

stat.Connect(textBox1.Text.ToString());

}

}

In aceasta etapa vom completa portul prin intermediul caruia va avea loc comunicatia seriala dintre arduino si laptop. Dupa ce conexiunea are loc, aceasta fereastra va disparea, aparand formularul celei de-a doua etape.



1. Etapa 2:

public partial class Status : Form

{

#region DrawingParameters

const int width = 15;

const int height = 30;

const int heightOff = 4;

int xPosition = 0;

int yPosition = 0;

const int xOffset = 20;

const int yOffset = 2;

#endregion

public delegate void SensorUpdate(string[]rd2);

public Status()

{

InitializeComponent();

}

public void Connect(string portName)

{

try

{

this.BackColor = System.Drawing.Color.LightGray;

serialPort.PortName = portName;

serialPort.Open();

errorText.Text = "Port " + portName + " successfully opened.";

start\_stop.Text = "STOP";

start\_stop.Visible = true;

tableLayoutPanel1.Visible = true;

Sensor\_1.Visible = true;

Sensor\_2.Visible = true;

Sensor\_3.Visible = true;

Sensor\_4.Visible = true;

Sensor\_5.Visible = true;

pictureBox1.Visible = true;

xPosition = tableLayoutPanel1.Location.X + Sensor\_1.Location.X + tableLayoutPanel1.Size.Width + xOffset;

yPosition = tableLayoutPanel1.Location.Y + Sensor\_1.Location.Y + Sensor\_1.Size.Height / 2 - height / 2;

for (int i = 0; i < 5; i++)

{

DrawIt(xPosition + xOffset \* 0, yPosition + yOffset \* 0 + 41 \* i, width, height - heightOff \* 0, Pens.Red);

DrawIt(xPosition + xOffset \* 1, yPosition + yOffset \* 1 + 41 \* i, width, height - heightOff \* 1, Pens.Red);

DrawIt(xPosition + xOffset \* 2, yPosition + yOffset \* 2 + 41 \* i, width, height - heightOff \* 2, Pens.Orange);

DrawIt(xPosition + xOffset \* 3, yPosition + yOffset \* 3 + 41 \* i, width, height - heightOff \* 3, Pens.Orange);

DrawIt(xPosition + xOffset \* 4, yPosition + yOffset \* 4 + 41 \* i, width, height - heightOff \* 4, Pens.Green);

DrawIt(xPosition + xOffset \* 5, yPosition + yOffset \* 5 + 41 \* i, width, height - heightOff \* 5, Pens.Green);

}

}

catch

{

errorText.Text = "Error. Unable to connect.\nCheck connection";

}

}

private void Back\_Click(object sender, EventArgs e)

{

try

{

serialPort.Close();

connectionReference.Show();

this.Dispose();

}

catch

{

errorText.Text = "Error. Unable to close connection";

}

}

private void Status\_FormClosing(object sender, FormClosingEventArgs e)

{

switch(e.CloseReason)

{

case CloseReason.UserClosing:

{

try

{

serialPort.Close();

connectionReference.Show();

this.Dispose();

}

catch

{

errorText.Text = "Error. Unable to close connection";

}

break;

}

}

}

private void serialPort\_DataReceived(object sender, System.IO.Ports.SerialDataReceivedEventArgs e)

{

string read;

try

{

read = serialPort.ReadLine();

}

catch (Exception ex) {

string filePath = @"..\Error.txt";

using (StreamWriter writer = new StreamWriter(filePath, true))

{

writer.WriteLine("Message :" + ex.Message + "<br/>" + Environment.NewLine + "StackTrace :" + ex.StackTrace +

"" + Environment.NewLine + "Date :" + DateTime.Now.ToString());

writer.WriteLine(Environment.NewLine + "-----------------------------------------------------------------------------" + Environment.NewLine);

}

return;

}

switch (state)

{

case 0:

{

if (read.Equals("ACK,\r"))

{

state += 1;

serialPort.Write("ACK,");

serialPort.DiscardInBuffer();

}

break;

}

case 1:

{

string[] rd2 = read.Split(',');

if (rd2[0] == "START" && rd2[6] == "END\r")

{

setSensorData(rd2);

}

break;

}

default:

{

break;

}

}

}

private void setSensorData(string[] rd2)

{

if (Sensor\_1.InvokeRequired)

{

SensorUpdate s = new SensorUpdate(setSensorData);

this.Invoke(s, new object[] { rd2 });

}

else

{

int[] sensor = new int[5];

Sensor\_1.Text = rd2[1] + " cm";

Sensor\_2.Text = rd2[2] + " cm";

Sensor\_3.Text = rd2[3] + " cm";

Sensor\_4.Text = rd2[4] + " cm";

Sensor\_5.Text = rd2[5] + " cm";

for(int i = 0; i < 5; i++)

{

Int32.TryParse(rd2[i+1], out sensor[i]);

}

#region DrawingSensorData

for (int i = 0; i < 5; i++)

{

if (sensor[i] <= 60 && sensor[i] >= 50)

{

FillIt(xPosition + xOffset \* 5, yPosition + yOffset \* 5 + 41 \* i, width, height - heightOff \* 5, Color.Green);

Unfill(xPosition + xOffset \* 4 + 1, yPosition + yOffset \* 4 + 41 \* i + 1, width - 1, height - heightOff \* 4 - 1);

Unfill(xPosition + xOffset \* 3 + 1, yPosition + yOffset \* 3 + 41 \* i + 1, width - 1, height - heightOff \* 3 - 1);

Unfill(xPosition + xOffset \* 2 + 1, yPosition + yOffset \* 2 + 41 \* i + 1, width - 1, height - heightOff \* 2 - 1);

Unfill(xPosition + xOffset \* 1 + 1, yPosition + yOffset \* 1 + 41 \* i + 1, width - 1, height - heightOff \* 1 - 1);

Unfill(xPosition + xOffset \* 0 + 1, yPosition + yOffset \* 0 + 41 \* i + 1, width - 1, height - heightOff \* 0 - 1);

}

else if (sensor[i] <= 50 && sensor[i] > 40)

{

FillIt(xPosition + xOffset \* 5, yPosition + yOffset \* 5 + 41 \* i, width, height - heightOff \* 5, Color.Green);

FillIt(xPosition + xOffset \* 4, yPosition + yOffset \* 4 + 41 \* i, width, height - heightOff \* 4, Color.Green);

Unfill(xPosition + xOffset \* 3 + 1, yPosition + yOffset \* 3 + 41 \* i + 1, width - 1, height - heightOff \* 3 - 1);

Unfill(xPosition + xOffset \* 2 + 1, yPosition + yOffset \* 2 + 41 \* i + 1, width - 1, height - heightOff \* 2 - 1);

Unfill(xPosition + xOffset \* 1 + 1, yPosition + yOffset \* 1 + 41 \* i + 1, width - 1, height - heightOff \* 1 - 1);

Unfill(xPosition + xOffset \* 0 + 1, yPosition + yOffset \* 0 + 41 \* i + 1, width - 1, height - heightOff \* 0 - 1);

}

else if (sensor[i] <= 40 && sensor[i] > 30)

{

FillIt(xPosition + xOffset \* 5, yPosition + yOffset \* 5 + 41 \* i, width, height - heightOff \* 5, Color.Green);

FillIt(xPosition + xOffset \* 4, yPosition + yOffset \* 4 + 41 \* i, width, height - heightOff \* 4, Color.Green);

FillIt(xPosition + xOffset \* 3, yPosition + yOffset \* 3 + 41 \* i, width, height - heightOff \* 3, Color.Orange);

Unfill(xPosition + xOffset \* 2 + 1, yPosition + yOffset \* 2 + 41 \* i + 1, width - 1, height - heightOff \* 2 - 1);

Unfill(xPosition + xOffset \* 1 + 1, yPosition + yOffset \* 1 + 41 \* i + 1, width - 1, height - heightOff \* 1 - 1);

Unfill(xPosition + xOffset \* 0 + 1, yPosition + yOffset \* 0 + 41 \* i + 1, width - 1, height - heightOff \* 0 - 1);

}

else if (sensor[i] <= 30 && sensor[i] > 20)

{

FillIt(xPosition + xOffset \* 5, yPosition + yOffset \* 5 + 41 \* i, width, height - heightOff \* 5, Color.Green);

FillIt(xPosition + xOffset \* 4, yPosition + yOffset \* 4 + 41 \* i, width, height - heightOff \* 4, Color.Green);

FillIt(xPosition + xOffset \* 3, yPosition + yOffset \* 3 + 41 \* i, width, height - heightOff \* 3, Color.Orange);

FillIt(xPosition + xOffset \* 2, yPosition + yOffset \* 2 + 41 \* i, width, height - heightOff \* 2, Color.Orange);

Unfill(xPosition + xOffset \* 1 + 1, yPosition + yOffset \* 1 + 41 \* i + 1, width - 1, height - heightOff \* 1 - 1);

Unfill(xPosition + xOffset \* 0 + 1, yPosition + yOffset \* 0 + 41 \* i + 1, width - 1, height - heightOff \* 0 - 1);

}

else if (sensor[i] <= 20 && sensor[i] > 10)

{

FillIt(xPosition + xOffset \* 5, yPosition + yOffset \* 5 + 41 \* i, width, height - heightOff \* 5, Color.Green);

FillIt(xPosition + xOffset \* 4, yPosition + yOffset \* 4 + 41 \* i, width, height - heightOff \* 4, Color.Green);

FillIt(xPosition + xOffset \* 3, yPosition + yOffset \* 3 + 41 \* i, width, height - heightOff \* 3, Color.Orange);

FillIt(xPosition + xOffset \* 2, yPosition + yOffset \* 2 + 41 \* i, width, height - heightOff \* 2, Color.Orange);

FillIt(xPosition + xOffset \* 1, yPosition + yOffset \* 1 + 41 \* i, width, height - heightOff \* 1, Color.Red);

Unfill(xPosition + xOffset \* 0 + 1, yPosition + yOffset \* 0 + 41 \* i + 1, width - 1, height - heightOff \* 0 - 1);

}

else if (sensor[i] <= 10 && sensor[i] >= 0)

{

FillIt(xPosition + xOffset \* 5, yPosition + yOffset \* 5 + 41 \* i, width, height - heightOff \* 5, Color.Green);

FillIt(xPosition + xOffset \* 4, yPosition + yOffset \* 4 + 41 \* i, width, height - heightOff \* 4, Color.Green);

FillIt(xPosition + xOffset \* 3, yPosition + yOffset \* 3 + 41 \* i, width, height - heightOff \* 3, Color.Orange);

FillIt(xPosition + xOffset \* 2, yPosition + yOffset \* 2 + 41 \* i, width, height - heightOff \* 2, Color.Orange);

FillIt(xPosition + xOffset \* 1, yPosition + yOffset \* 1 + 41 \* i, width, height - heightOff \* 1, Color.Red);

FillIt(xPosition + xOffset \* 0, yPosition + yOffset \* 0 + 41 \* i, width, height - heightOff \* 0, Color.Red);

}

else

{

Unfill(xPosition + xOffset \* 5 + 1, yPosition + yOffset \* 5 + 41 \* i + 1, width - 1, height - heightOff \* 5 - 1);

Unfill(xPosition + xOffset \* 4 + 1, yPosition + yOffset \* 4 + 41 \* i + 1, width - 1, height - heightOff \* 4 - 1);

Unfill(xPosition + xOffset \* 3 + 1, yPosition + yOffset \* 3 + 41 \* i + 1, width - 1, height - heightOff \* 3 - 1);

Unfill(xPosition + xOffset \* 2 + 1, yPosition + yOffset \* 2 + 41 \* i + 1, width - 1, height - heightOff \* 2 - 1);

Unfill(xPosition + xOffset \* 1 + 1, yPosition + yOffset \* 1 + 41 \* i + 1, width - 1, height - heightOff \* 1 - 1);

Unfill(xPosition + xOffset \* 0 + 1, yPosition + yOffset \* 0 + 41 \* i + 1, width - 1, height - heightOff \* 0 - 1);

}

}

#endregion

}

}

private void serialPort\_ErrorReceived(object sender, System.IO.Ports.SerialErrorReceivedEventArgs e)

{

start\_stop.Visible = false;

tableLayoutPanel1.Visible = false;

Sensor\_1.Visible = false;

Sensor\_2.Visible = false;

Sensor\_3.Visible = false;

Sensor\_4.Visible = false;

Sensor\_5.Visible = false;

pictureBox1.Visible = false;

errorText.Text = "Connection Error. Closing Serial Port: "+ serialPort.PortName+"\nERROR:\n";

errorText.Text += e.ToString();

try

{

serialPort.Close();

}

catch { return; }

}

private void start\_stop\_Click(object sender, EventArgs e)

{

if (start\_stop.Text.Equals("START"))

{

start\_stop.Text = "STOP";

serialPort.Write("START,");

}

else if (start\_stop.Text.Equals("STOP"))

{

start\_stop.Text = "START";

serialPort.Write("STOP,");

}

}

private void DrawIt(int x, int y, int width, int height, Pen p)

{

System.Drawing.Graphics graphics = this.CreateGraphics();

graphics.DrawRectangle(p, new Rectangle(x, y, width, height));

graphics.Dispose();

}

private void FillIt(int x, int y, int width, int height, Color c)

{

System.Drawing.SolidBrush myBrush = new System.Drawing.SolidBrush(c);

System.Drawing.Graphics graphics = this.CreateGraphics();

graphics.FillRectangle(myBrush, new Rectangle(x, y, width, height));

graphics.Dispose();

myBrush.Dispose();

}

private void Unfill(int x, int y, int width, int height)

{

System.Drawing.SolidBrush myBrush = new System.Drawing.SolidBrush(Color.LightGray);

System.Drawing.Graphics graphics = this.CreateGraphics();

graphics.FillRectangle(myBrush, new Rectangle(x, y, width, height));

graphics.Dispose();

myBrush.Dispose();

}

1. Initializari:

In regiunea DrawingParameters vom initializa parametrii pentru desenul patratelor ce vor fi umplute mai tarziu in functie de valorile returnate de senzorii ultrasonici. De asemenea am instantia un delegat care care anunta cand vom primi valorile senzorilor de la placa in loc de alt mesaj.

1. Functia Connect (string portName):

Parametrul portName fiind cel introdus in Etapa 1 si transmis mai departe aici pentru a avea loc coneciunea prin seriala. Asadar daca nu sunt erori sau exceptii returnate iar portul introdus este correct atunci comunicatia prin seriala se va stabili iar valorile senzorilor vor deveni vizibile. Tot aici, utilizand libraria System.Drawing, vom desena patratele caracteristice pentru fiecare senzor, care vor reprezenta o parte a interfetei grafice cu utilizatorul.

1. Functia Back\_Click si Status\_FormClosing:

Acestea reprezinta funcii pentru butoanele de “Back” si “X”, in care vom face fereastra de la Etapa 1 din nou vizibila, inchizand Etapa 2.

1. Functia serialPort\_DataReceived:

Aici citim informatiile de pe seriala trimise de la placa arduino. In cazul in care mesajul primit este “ACK,”, atunci vom transmite inapoi spre placa arduino mesajul “ACK,” pentru a incepe transmiterea datelor de la senzori spre programul de pe laptop.

In cazul in care placa arduino a inceput transmiterea de date de la cei 5 senzori, vom imparti mesajul de forma “START,val1,val2,val3,val4,val5,END” si vom lua valoarea fiecarui senzor pentru a o pune intr-un vector de stringuri.

1. Functia setSensorData(string[] rd2):

Functia primeste ca parametru un vector de stringuri ce contine valorile celor 5 senzori, urmand sa afisam aceste valori pe formularul etapei 2. In continuare vom updata si interfata grafica, umpland sau golind patratele caracteristice fiecarui senzor in functie de distanta aratata pana la obiectul respectiv identificat.

1. Functia serialPort\_ErrorReceived():

Asa cum precizeaza si numele, daca apare o eroare in timpul transmisiei, aceasta functie va ascunde toate informatiile de pe formularul etapei 2 si va arata doar un mesaj de eroare pentru portul introdus.

1. Functia start\_stop\_Click():

Asa cum am descris si pentru programul de arduino, putem opri transmiterea si o putem porni prin apasarea acestor doua butoane. In momentul apasarii se va trimite prin seriala mesajul “START” sau “STOP”, in functie de butonul apasat, acest mesaj fiind si analizat mai departe de catre placa arduino.

1. Functiile DrawIt, FillIt, Unfill:

Sunt functii folosite pentru desenarea, umplerea sau golirea patratelor din interfata grafica.