**Advanced bus system (CM2)**

**Authors:**

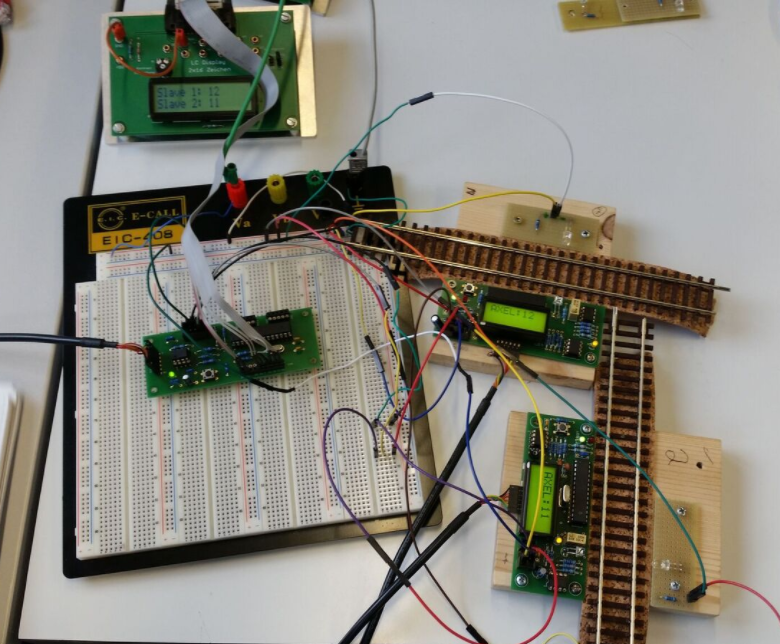
Shahab Shafieihajiabady

Atif Saeed

Ievgenii Nudga

January 2018

horizontal line



# Introduction

Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the [Arduino programming language](https://www.arduino.cc/en/Reference/HomePage) (based on [Wiring](http://wiring.org.co/)), and [the Arduino Software (IDE)](https://www.arduino.cc/en/Main/Software), based on [Processing](https://processing.org/).

Over the years Arduino has been the brain of thousands of projects, from everyday objects to complex scientific instruments. A worldwide community of makers - students, hobbyists, artists, programmers, and professionals - has gathered around this open-source platform, their contributions have added up to an incredible amount of accessible knowledge that can be of great help to novices and experts alike.

## Steps

1. Program and test the Axis-Counter (Display on the LCD turned by 180 °!)

2. Program and test the Master-Controller

3. Install all Axis-Counters (Adr. from 21 to 30) and connect them to the bus

4. Connect a Master-Controller and some Monitor-Controllers to the bus

5. Consider a procedure for the function of the Block-Control

6. Test the bus and the function of the Block-Control

7. Send data via the USB-port to the PCs 8. Program the PCs to display the track-plan and the position of the trains

### Master Board and Slave

In this project two boards are programmed to communicate with one another in a Master Writer/Slave Receiver configuration via the I2C Synchronous Serial protocol. Several functions of Arduino Wire Library are used to accomplish this. Arduino 1, the Master, is programmed to send 6 bytes of data every half second to a uniquely addressed Slave. Once that message is received, it can then be viewed in the Slave board's serial monitor window opened on the USB connected computer running the Arduino Software (IDE).

The I2C protocol involves using two lines to send and receive data: a serial clock pin (SCL) that the Arduino or Master board pulses at a regular interval, and a serial data pin (SDA) over which data is sent between the two devices. As the clock line changes from low to high (known as the rising edge of the clock pulse), a single bit of information - that will form in sequence the address of a specific device and a command or data - is transferred from the board to the I2Cdevice over the SDA line. When this information is sent - bit after bit -, the called upon device executes the request and transmits it's data back - if required - to the board over the same line using the clock signal still generated by the Master on SCL as timing. The initial eight bits (i.e. eight clock pulses) from the Master to Slaves contain the address of the device the Master wants data from. The bits after contain the memory address on the Slave that the Master wants to read data from or write data to, and the data to be written, if any.

Each Slave device has to have its own unique address and both master and slave devices need to take turns communicating over a the same data line line. In this way, it's possible for your Arduino or Master boards to communicate with many device or other boards using just two pins of your microcontroller, using each device's unique address.

### 

### 

### 

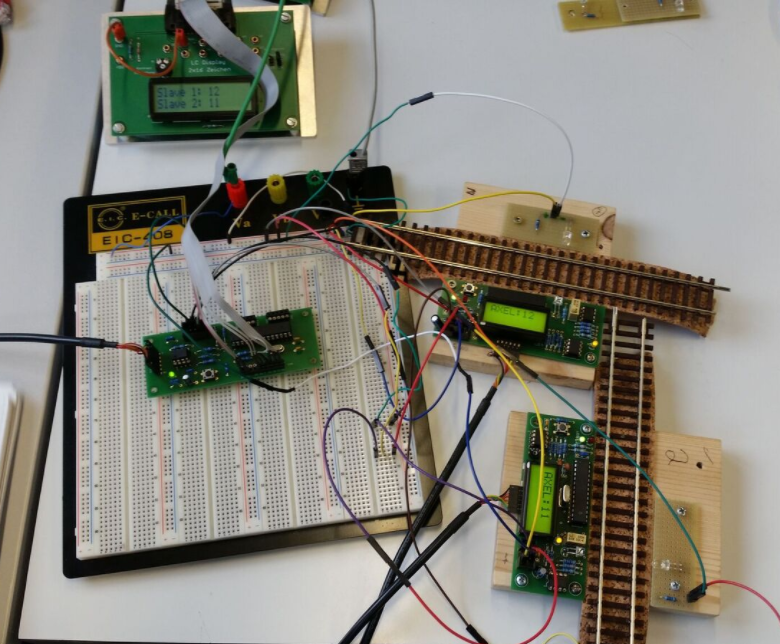
### 

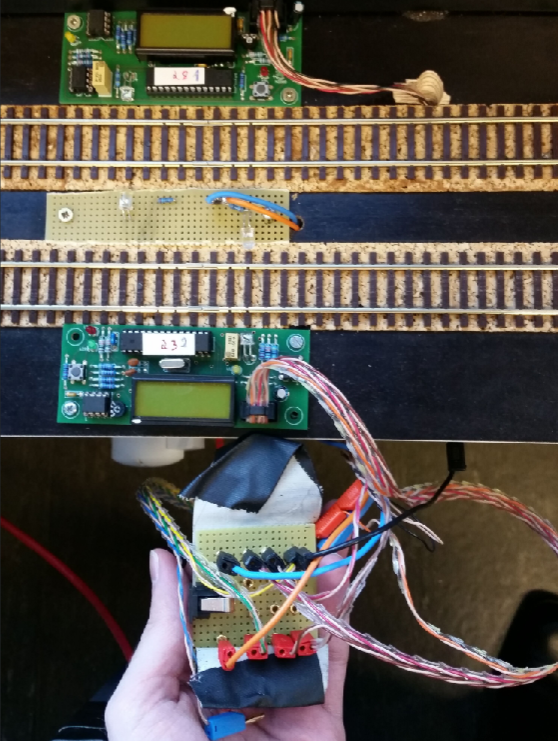
### 

### 

### Monitor

The LCD Monitor was used to show the output of Master-controller. The position of the train will be sent to master from slaves and afterwards, It will be shown on the monitor. When train passes a Slave Boards, a signal containing the number of slaves will be sent to the master. When the train enters the next station, the number of slaves will be counted and sent to the master. Master will compare the two numbers and if they are equal, a signal from master will be sent to reset the first slave.





### 

### Codes

***Normal Slave:***

#include <Wire.h>

#include <LiquidCrystal.h>

#define NODE\_ADDRESS 23 // Change this unique address for each I2C slave node

#define PAYLOAD\_SIZE 2 // Number of bytes expected to be received by the master I2C node

byte nodePayload[PAYLOAD\_SIZE];

LiquidCrystal lcd(7, 8, 9, 10, 11, 12, 13);

int ledPin = 3;

int comparatorPin = 5;

int counter=0;

int increment=0;

unsigned long changeTime;

unsigned long currentTime;

int delta = 0;

int currentState=0;

int nextState=0;

int waste;

int turnGreen = 0;

void setup()

{

// put your setup code here, to run once:

pinMode(ledPin,OUTPUT);

digitalWrite(ledPin, HIGH);

lcd.begin(8, 2); //initialize the lcd

lcd.setCursor(0,0);

//lcd.print(0);

digitalWrite(ledPin, HIGH);

Serial.begin(9600);

lcd.print("COUNT:");

lcd.setCursor(0,1);

lcd.print("RECVD:");

Wire.begin(NODE\_ADDRESS); // Join I2C network

Wire.onRequest(requestEvent); // Request attention of master node

Wire.onReceive(receiveEvent); // receive value from master

}

void loop()

{

// put your main code here, to run repeatedly:

if( digitalRead(comparatorPin) == LOW )

{

nextState =0;

currentTime = millis();

delta = currentTime - changeTime;

// lcd.setCursor(0,1);

//lcd.print(delta);

}

else

{

currentState = 1;

if(currentState != nextState){

changeTime = millis();

counter = counter + 1;

//nodePayload[0] = NODE\_ADDRESS; // Send Node address back.

nodePayload[1] = counter; // send counter value to master

digitalWrite(ledPin, LOW); // switch on Red LED

lcd.setCursor(6,0);

lcd.print(counter);

}

nextState = currentState;

}

}

void requestEvent()

{

Wire.write(counter);

Serial.print("Sensor value: "); // for debugging purposes.

Serial.println(counter); // for debugging purposes.

}

void receiveEvent(int howMany)

{

while (1 < Wire.available()) {waste = Wire.read(); }

counter = 0;

//lcd.clear();

lcd.setCursor(6,0);

lcd.print(" ");

lcd.print(counter);

turnGreen = Wire.read(); // receive number for turn on green LED

digitalWrite(ledPin, HIGH); // switch on Green LED

}

***Reversed Slave***

#include <Wire.h>

#include <LiquidCrystal.h>

#define NODE\_ADDRESS 28// Change this unique address for each I2C slave node

#define PAYLOAD\_SIZE 2 // Number of bytes expected to be received by the master I2C node

byte nodePayload[PAYLOAD\_SIZE];

LiquidCrystal lcd(7, 8, 9, 10, 11, 12, 13);

int ledPin = 3;

int comparatorPin = 5;

int counter=0;

int increment=0;

unsigned long changeTime;

unsigned long currentTime;

int delta = 0;

int currentState=0;

int nextState=0;

int waste;

int turnGreen = 0;

byte one[8] = { B00000, B01110, B00100, B00100, B00100, B00100, B00110, B00100 };

byte two[8] = { B00000, B11111, B00010, B00100, B01000, B10000, B10000, B01110 };

byte three[8] = { B00000, B01110, B10001, B10000, B01000, B00100, B01000, B11111 };

byte four[8] = { B00000, B01000, B01000, B11111, B01001, B01010, B01100, B01000 };

byte five[8] = { B00000, B01110, B10001, B10000, B10000, B01111, B00001, B11111 };

byte seven[8] = {B00000, B00010, B00010, B00010, B00100, B01000, B10000, B11111 };

void printFlip(int counter);

void setup()

{

// put your setup code here, to run once:

pinMode(ledPin,OUTPUT);

digitalWrite(ledPin, HIGH);

lcd.begin(8, 2); //initialize the lcd

//lcd.setCursor(0,0);

//lcd.print(0);

digitalWrite(ledPin, HIGH);

Serial.begin(9600);

lcd.setCursor(6,1);

lcd.print(":X");

lcd.setCursor(6,0);

lcd.print(":H");

/\*

lcd.print("COUNT:");

lcd.setCursor(0,1);

lcd.print("RECVD:");

\*/

Wire.begin(NODE\_ADDRESS); // Join I2C network

Wire.onRequest(requestEvent); // Request attention of master node

Wire.onReceive(receiveEvent); // receive value from master

// add to setup

lcd.createChar(0, one);

lcd.createChar(1, two);

lcd.createChar(2, three);

lcd.createChar(3, four);

lcd.createChar(4, five);

lcd.createChar(5, seven);

}

void printFlip(int counter){

//break into tens and unit

int tens = counter/10;

int unit = counter%10;

//print tens

lcd.setCursor(5,1);

if (tens == 1) lcd.write(byte(0));

else if (tens == 2) lcd.write(byte(1));

else if (tens == 3) lcd.write(byte(2));

else if (tens == 4) lcd.write(byte(3));

else if (tens == 5) lcd.write(byte(4));

else if (tens == 7) lcd.write(byte(5));

else if (tens == 6) lcd.print("9");

else if (tens == 9) lcd.print("6");

else if (tens == 0) lcd.print("");

else lcd.print(tens);

//print unit

lcd.setCursor(4,1);

if (unit == 1) lcd.write(byte(0));

else if (unit == 2) lcd.write(byte(1));

else if (unit == 3) lcd.write(byte(2));

else if (unit == 4) lcd.write(byte(3));

else if (unit == 5) lcd.write(byte(4));

else if (unit == 7) lcd.write(byte(5));

else if (unit == 6) lcd.print("9");

else if (unit == 9) lcd.print("6");

else lcd.print(unit);

}

void loop()

{

// put your main code here, to run repeatedly:

if( digitalRead(comparatorPin) == LOW )

{

nextState =0;

currentTime = millis();

delta = currentTime - changeTime;

// lcd.setCursor(0,1);

//lcd.print(delta);

}

else

{

currentState = 1;

if(currentState != nextState){

changeTime = millis();

counter = counter + 1;

//nodePayload[0] = NODE\_ADDRESS; // Send Node address back.

nodePayload[1] = counter; // send counter value to master

digitalWrite(ledPin, LOW); // switch on Red LED

printFlip(counter);

}

nextState = currentState;

}

}

void requestEvent()

{

Wire.write(counter);

Serial.print("Sensor value: "); // for debugging purposes.

Serial.println(counter);

}

void receiveEvent(int howMany)

{

while (1 < Wire.available()) {waste = Wire.read(); }

counter = 0;

//lcd.clear();

//printFlip(counter);

lcd.setCursor(4,1);

lcd.print("0");

lcd.setCursor(5,1);

lcd.print(" ");

turnGreen = Wire.read(); // receive number for turn on green LED

digitalWrite(ledPin, HIGH); // switch on Green LED

}

***Master***

#include <Wire.h>

#include <LiquidCrystal.h>

#define LED\_PIN 3

#define INPUT\_PIN PD5

LiquidCrystal lcd(3,4,6,5,8,7);

//LiquidCrystal lcd(3,4,5,6,7,8);

int axelCount1;

int axelCount2;

byte on\_inv[8] = {

B01110,

B00100,

B00100,

B00100,

B00100,

B00110,

B00100,

};

byte tw\_inv[8] = {

B11111,

B00010,

B00100,

B01000,

B10000,

B10001,

B01110,

};

byte th\_inv[8] = {

B01110,

B10001,

B10000,

B01000,

B00100,

B01000,

B11111,

};

byte fo\_inv[8] = {

B01000,

B01000,

B11111,

B01001,

B01010,

B01100,

B01000,

};

byte fi\_inv[8] = {

B01110,

B10001,

B10000,

B10000,

B01111,

B00001,

B11111,

};

byte se\_inv[8] = {

B00010,

B00010,

B00010,

B00100,

B01000,

B10000,

B11111,

};

void setup() {

// put your setup code here, to run once:

Wire.begin(); // join i2c bus (address optional for master)

Wire.onReceive(receiveEvent);

Serial.begin(9600);

lcd.begin(16, 2);

char character = 'a';

lcd.createChar(1, on\_inv);

lcd.createChar(2, tw\_inv);

lcd.createChar(3, th\_inv);

lcd.createChar(4, fo\_inv);

lcd.createChar(5, fi\_inv);

lcd.createChar(7, se\_inv);

}

void loop() {

//Serial.println(character);

Wire.requestFrom(23, 1);

delay(100);

while (Wire.available()) {

axelCount1 = Wire.read();

//Serial.println("axelCount1"+axelCount1);

}

Wire.requestFrom(28, 1);

delay(100);

while (Wire.available()) {

axelCount2 = Wire.read();

// Serial.println("axelCount2"+axelCount2);

}

if((axelCount1 == axelCount2)&&(axelCount1!=0))

{

Wire.beginTransmission(28);

delay(100);

Wire.write(2);

delay(100);

Wire.endTransmission();

Serial.println("equal");

}

lcd.setCursor(0,1);

lcd.print("Slave 23: ");

lcd.print(axelCount1);

// switch (axelCount1){

// case 0:

// lcd.print(0);

// break;

// case 1:

// lcd.write(byte(1));

// break;

// case 2:

// lcd.write(byte(2));

// break;

// case 3:

// lcd.write(byte(3));

// break;

// case 4:

// lcd.write(byte(4));

// break;

// case 5:

// lcd.write(byte(5));

// break;

// case 6:

// lcd.print(9);

// break;

// case 7:

// lcd.write(byte(7));

// break;

// case 8:

// lcd.print(8);

// break;

// case 9:

// lcd.print(6);

// break;

// default:

// lcd.print("");

//

// break;

//

// }

//

// lcd.print(axelCount1);

//lcd.print("test");

lcd.setCursor(0,0);

lcd.print("Slave 28: ");

lcd.print(axelCount2);

delay(50);

}

void receiveEvent(int howMany) {

// while (Wire.available()) { // loop through all but the last

// char c = Wire.read(); // receive byte as a character

// Serial.print(" " + c); // print the character

//}

}

***Monitor***

#include <Wire.h>

#include <LiquidCrystal.h>

#define NODE\_ADDRESS 33 // Change this unique address for each I2C slave node

#define PAYLOAD\_SIZE 2 // Number of bytes expected to be received by the master I2C node

byte nodePayload[PAYLOAD\_SIZE];

LiquidCrystal lcd(3, 4, 6, 5, 8, 7);

int cursors[] = {1, 3 ,5, 7, 9, 1, 3, 5, 7, 10};

int ledPin = 3;

int comparatorPin = 5;

int counter=0;

int increment=0;

unsigned long changeTime;

unsigned long currentTime;

int delta = 0;

int currentState=0;

int nextState=0;

int waste;

int turnGreen = 0;

void setup()

{

// put your setup code here, to run once:

pinMode(ledPin,OUTPUT);

digitalWrite(ledPin, HIGH);

lcd.begin(16, 2); //initialize the lcd

lcd.setCursor(0,0);

lcd.print("1 2 3 4 5");

lcd.setCursor(0,1);

lcd.print("6 7 8 9 10");

Wire.begin(NODE\_ADDRESS); // Join I2C network

Wire.onRequest(requestEvent); // Request attention of master node

Wire.onReceive(receiveEvent); // receive value from master

}

void loop()

{

}

void requestEvent()

{

Wire.write(counter);

Serial.print("Sensor value: "); // for debugging purposes.

Serial.println(counter); // for debugging purposes.

}

void receiveEvent(int howMany)

{

while (1 < Wire.available()) {waste = Wire.read(); }

int slaveAddress = Wire.read();

Serial.begin(9600);

Serial.println("WE ARE RECEIVING!!!");

Serial.println(slaveAddress);

if (slaveAddress > 30){

slaveAddress -= 20;

slaveAddress = slaveAddress % 20;

Serial.println(slaveAddress);

if (slaveAddress<=5) {

lcd.setCursor(cursors[slaveAddress-1],0);

lcd.print(" ");

}

else {

lcd.setCursor(cursors[slaveAddress-1],1);

lcd.print(" ");

}

}

else

{

slaveAddress = slaveAddress % 20;

Serial.println(slaveAddress);

if (slaveAddress<=5) {

lcd.setCursor(cursors[slaveAddress-1],0);

lcd.print("X");

}

else {

lcd.setCursor(cursors[slaveAddress-1],1);

lcd.print("X");

}

}

}

**C# Code**

**SERIAL :**

**Form1.cs :**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.IO;

using System.IO.Ports;

namespace Serial

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

getAvailablePorts();

button3Read.Enabled = false;

}

private void Form1\_Load(object sender, EventArgs e) { }

private void getAvailablePorts()

{

string[] ports = SerialPort.GetPortNames();

comboBox1AvailablePorts.Items.AddRange(ports);

}

private void label2\_Click(object sender, EventArgs e) { }

private void textBox1\_TextChanged(object sender, EventArgs e) { }

private void textBox2\_TextChanged(object sender, EventArgs e) { }

private void label1\_Click(object sender, EventArgs e) { }

// What ?? to do when Start Button will be pressed. i.e. open the serial port for transmission or reception

private void button1\_Click(object sender, EventArgs e)

{

{

try

{

if (comboBox1AvailablePorts.Text == "")

{

textBox1PortAndBaudInfo.Text = "Please Select Port";

}

else if (comboBox2BaudRate.Text == "")

{

textBox1PortAndBaudInfo.Text = "Please select Baud Rate";

}

else

{

SerialPort1.PortName = comboBox1AvailablePorts.Text;

SerialPort1.BaudRate = Convert.ToInt32(comboBox2BaudRate.Text);

SerialPort1.Open();

textBox1PortAndBaudInfo.Text = "Open";

button3Read.Enabled = true;

}

}

catch (UnauthorizedAccessException)

{

textBox1PortAndBaudInfo.Text = "Error";

}

}

}

// what ?? to do after stop button will be pressed. i.e. close the serial port, no more serial reception or transmission

private void button2\_Click(object sender, EventArgs e)

{

SerialPort1.Close();

textBox1PortAndBaudInfo.Text = "Close";

button3Read.Enabled = false;

}

private void button3\_Click(object sender, EventArgs e)

{

int j = 0;

try

{

while (j < 24)

{

textBox2ReadValue.Text += SerialPort1.ReadLine() + "\n";

j++;

}

}

catch (TimeoutException)

{

{

textBox2ReadValue.Text = "Timeout";

}

}

}

private void comboBox2BaudRate\_SelectedIndexChanged(object sender, EventArgs e)

{

}

}

}

**Form1.Designer.cs :**

namespace Serial

{

partial class Form1

{

/// <summary>

/// Required designer variable.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Clean up any resources being used.

/// </summary>

/// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

private void InitializeComponent()

{

this.components = new System.ComponentModel.Container();

this.button1Start = new System.Windows.Forms.Button();

this.button2Stop = new System.Windows.Forms.Button();

this.button3Read = new System.Windows.Forms.Button();

this.comboBox1AvailablePorts = new System.Windows.Forms.ComboBox();

this.comboBox2BaudRate = new System.Windows.Forms.ComboBox();

this.SerialPort1 = new System.IO.Ports.SerialPort(this.components);

this.label1AvailablePorts = new System.Windows.Forms.Label();

this.label2BaudRate = new System.Windows.Forms.Label();

this.textBox1PortAndBaudInfo = new System.Windows.Forms.TextBox();

this.textBox2ReadValue = new System.Windows.Forms.TextBox();

this.SuspendLayout();

//

// button1Start

//

this.button1Start.BackColor = System.Drawing.Color.LightGreen;

this.button1Start.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));

this.button1Start.Location = new System.Drawing.Point(12, 145);

this.button1Start.Name = "button1Start";

this.button1Start.Size = new System.Drawing.Size(75, 23);

this.button1Start.TabIndex = 0;

this.button1Start.Text = "START";

this.button1Start.UseVisualStyleBackColor = false;

this.button1Start.Click += new System.EventHandler(this.button1\_Click);

//

// button2Stop

//

this.button2Stop.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(255)))), ((int)(((byte)(128)))), ((int)(((byte)(128)))));

this.button2Stop.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));

this.button2Stop.Location = new System.Drawing.Point(12, 105);

this.button2Stop.Name = "button2Stop";

this.button2Stop.Size = new System.Drawing.Size(75, 23);

this.button2Stop.TabIndex = 1;

this.button2Stop.Text = "STOP";

this.button2Stop.UseVisualStyleBackColor = false;

this.button2Stop.Click += new System.EventHandler(this.button2\_Click);

//

// button3Read

//

this.button3Read.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(128)))), ((int)(((byte)(128)))), ((int)(((byte)(255)))));

this.button3Read.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));

this.button3Read.Location = new System.Drawing.Point(12, 187);

this.button3Read.Name = "button3Read";

this.button3Read.Size = new System.Drawing.Size(75, 23);

this.button3Read.TabIndex = 2;

this.button3Read.Text = "READ";

this.button3Read.UseVisualStyleBackColor = false;

this.button3Read.Click += new System.EventHandler(this.button3\_Click);

//

// comboBox1AvailablePorts

//

this.comboBox1AvailablePorts.FormattingEnabled = true;

this.comboBox1AvailablePorts.Location = new System.Drawing.Point(12, 25);

this.comboBox1AvailablePorts.Name = "comboBox1AvailablePorts";

this.comboBox1AvailablePorts.Size = new System.Drawing.Size(121, 21);

this.comboBox1AvailablePorts.TabIndex = 3;

this.comboBox1AvailablePorts.SelectedIndexChanged += new System.EventHandler(this.Form1\_Load);

//

// comboBox2BaudRate

//

this.comboBox2BaudRate.FormattingEnabled = true;

this.comboBox2BaudRate.Items.AddRange(new object[] {

"9600",

"11500"});

this.comboBox2BaudRate.Location = new System.Drawing.Point(292, 25);

this.comboBox2BaudRate.Name = "comboBox2BaudRate";

this.comboBox2BaudRate.Size = new System.Drawing.Size(121, 21);

this.comboBox2BaudRate.TabIndex = 4;

this.comboBox2BaudRate.SelectedIndexChanged += new System.EventHandler(this.comboBox2BaudRate\_SelectedIndexChanged);

//

// label1AvailablePorts

//

this.label1AvailablePorts.AutoSize = true;

this.label1AvailablePorts.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));

this.label1AvailablePorts.Location = new System.Drawing.Point(12, 7);

this.label1AvailablePorts.Name = "label1AvailablePorts";

this.label1AvailablePorts.Size = new System.Drawing.Size(119, 13);

this.label1AvailablePorts.TabIndex = 5;

this.label1AvailablePorts.Text = "AVAILABLE PORTS";

this.label1AvailablePorts.Click += new System.EventHandler(this.label1\_Click);

//

// label2BaudRate

//

this.label2BaudRate.AutoSize = true;

this.label2BaudRate.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));

this.label2BaudRate.Location = new System.Drawing.Point(289, 7);

this.label2BaudRate.Name = "label2BaudRate";

this.label2BaudRate.Size = new System.Drawing.Size(78, 13);

this.label2BaudRate.TabIndex = 6;

this.label2BaudRate.Text = "BAUD RATE";

this.label2BaudRate.Click += new System.EventHandler(this.label2\_Click);

//

// textBox1PortAndBaudInfo

//

this.textBox1PortAndBaudInfo.BackColor = System.Drawing.SystemColors.ActiveCaption;

this.textBox1PortAndBaudInfo.ForeColor = System.Drawing.Color.FromArgb(((int)(((byte)(192)))), ((int)(((byte)(0)))), ((int)(((byte)(0)))));

this.textBox1PortAndBaudInfo.Location = new System.Drawing.Point(12, 52);

this.textBox1PortAndBaudInfo.Multiline = true;

this.textBox1PortAndBaudInfo.Name = "textBox1PortAndBaudInfo";

this.textBox1PortAndBaudInfo.ReadOnly = true;

this.textBox1PortAndBaudInfo.Size = new System.Drawing.Size(401, 47);

this.textBox1PortAndBaudInfo.TabIndex = 7;

this.textBox1PortAndBaudInfo.TextChanged += new System.EventHandler(this.textBox1\_TextChanged);

//

// textBox2ReadValue

//

this.textBox2ReadValue.BackColor = System.Drawing.SystemColors.ActiveCaption;

this.textBox2ReadValue.ForeColor = System.Drawing.Color.White;

this.textBox2ReadValue.Location = new System.Drawing.Point(12, 229);

this.textBox2ReadValue.Multiline = true;

this.textBox2ReadValue.Name = "textBox2ReadValue";

this.textBox2ReadValue.ReadOnly = true;

this.textBox2ReadValue.Size = new System.Drawing.Size(301, 236);

this.textBox2ReadValue.TabIndex = 8;

this.textBox2ReadValue.TextChanged += new System.EventHandler(this.textBox2\_TextChanged);

//

// Form1

//

this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.ClientSize = new System.Drawing.Size(718, 477);

this.Controls.Add(this.textBox2ReadValue);

this.Controls.Add(this.textBox1PortAndBaudInfo);

this.Controls.Add(this.label2BaudRate);

this.Controls.Add(this.label1AvailablePorts);

this.Controls.Add(this.comboBox2BaudRate);

this.Controls.Add(this.comboBox1AvailablePorts);

this.Controls.Add(this.button3Read);

this.Controls.Add(this.button2Stop);

this.Controls.Add(this.button1Start);

this.Name = "Form1";

this.Text = "Form1";

this.Load += new System.EventHandler(this.Form1\_Load);

this.ResumeLayout(false);

this.PerformLayout();

}

#endregion

private System.Windows.Forms.Button button1Start;

private System.Windows.Forms.Button button2Stop;

private System.Windows.Forms.Button button3Read;

private System.Windows.Forms.ComboBox comboBox1AvailablePorts;

private System.Windows.Forms.ComboBox comboBox2BaudRate;

private System.IO.Ports.SerialPort SerialPort1;

private System.Windows.Forms.Label label1AvailablePorts;

private System.Windows.Forms.Label label2BaudRate;

private System.Windows.Forms.TextBox textBox1PortAndBaudInfo;

private System.Windows.Forms.TextBox textBox2ReadValue;

}

}

**TRACK :**

**Form1.cs:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.IO.Ports;

namespace Tracks

{

public partial class Form1 : Form

{

private string slaveID;

private static String[] slaveAddressLowerPart = new String[] {"21","23","27","29"};

private static String[] slaveAddressUpperPart = new String[] {"22","24","28","30","26","25"};

Dictionary<String, int[]> slaveIDAndPosition = new Dictionary<String, int[]>();

private const int lineSpace = 30 ;

private int widthOfSlaveIndicator;

private int heightOfSlaveIndicator; // taking only half of its width

private int positionXOfSlaveIndicator; // position adjustment to the vertical line

private int positionYOfSlaveIndicator; // lineSpace/2+lineSpace/4 is nothing but adjustment to the Horizontal line

public Form1()

{

InitializeComponent();

getAvailablePorts();

slaveID = " ";

widthOfSlaveIndicator = lineSpace;

heightOfSlaveIndicator = (int)(0.5 \* widthOfSlaveIndicator); // taking only half of its width

positionXOfSlaveIndicator = lineSpace \* 31; // position adjustment to the vertical line

positionYOfSlaveIndicator = (lineSpace \* 5) + lineSpace / 2 + lineSpace / 4; // lineSpace/2+lineSpace/4 is nothing but adjustment to the Horizontal line

button1Starten.Enabled = true;

button2Stop.Enabled = false;

// connect the paint event of the picture box to the event handler method.

pictureBox1.Paint += new System.Windows.Forms.PaintEventHandler(this.pictureBox1\_Paint);

// Add the pictureBox control to the Form.

this.Controls.Add(pictureBox1);

}

// method theat finds the available Ports

private void getAvailablePorts()

{

string[] ports = SerialPort.GetPortNames();

comboBox1AvailablePorts.Items.AddRange(ports);

//String[] baud = { "9600", "11500"};

//comboBox2BaudRates.Items.AddRange(baud);

comboBox2BaudRates.Items.Add("9600");

}

private void pictureBox1\_Paint(object sender, PaintEventArgs e)

{

// creating pens with different colors

System.Drawing.Pen myPenBlack10 = new System.Drawing.Pen(Color.Black, 10.0f);

System.Drawing.Pen myPenGreen10 = new System.Drawing.Pen(Color.Green, 10.0f);

System.Drawing.Pen myPenRed10 = new System.Drawing.Pen(Color.Red, 10.0f);

System.Drawing.Pen myPenBlack4 = new System.Drawing.Pen(Color.Black, 4.0f);

System.Drawing.Pen myPenLightBlue1 = new System.Drawing.Pen(Color.LightBlue, 1.0f);

System.Drawing.Pen myPenBlack1 = new System.Drawing.Pen(Color.Black, 1.0f);

// Drawing Fonts

System.Drawing.Font drawFont = new System.Drawing.Font("Arial", 11);

System.Drawing.SolidBrush myBrushYellow = new System.Drawing.SolidBrush(Color.Yellow);

SolidBrush drawBrush = new SolidBrush(Color.Black);

// Create a local version of the graphics object for the pictureBox

Graphics g = e.Graphics;

// Draw Grid

int gridWidth = this.ClientSize.Width;

int gridHeight = this.ClientSize.Height;

int noOfVerticalLines = (int)(gridWidth / lineSpace);

int noOfHorizontalLines = (int)(gridHeight / lineSpace);

for (int i = 0; i < noOfVerticalLines; i++) { g.DrawLine(myPenLightBlue1, i \* lineSpace, 0, i \* lineSpace, gridHeight); }

for (int j = 0; j < noOfHorizontalLines; j++) { g.DrawLine(myPenLightBlue1, 0, j \* lineSpace, gridWidth, j \* lineSpace); }

int stationNumberLowerPart = 4;

int StationNumberUpperPart = 6;

int[] positionY = createTrainStations(g, lineSpace, stationNumberLowerPart, StationNumberUpperPart, myPenBlack1);

int lowerPartStationPositionY = positionY[0];

int upperPartStationPositionY = positionY[1];

createPlatForms(g, myPenGreen10, lowerPartStationPositionY, upperPartStationPositionY, lineSpace);

// draw border of the platform

int x1Border = 9 \* lineSpace;

int y1Border = upperPartStationPositionY + (int)(1.6 \* lineSpace);

int x2Border = 9 \* lineSpace; ;

int y2Border = y1Border + (int)(0.8 \* lineSpace);

int borderDistance = 5 \* lineSpace;

int numberOfBorder = 5;

drawBorderOfPlatForm(g, myPenBlack4, x1Border, y1Border, x2Border, y2Border, borderDistance, numberOfBorder);

drawBorderOfPlatForm(g, myPenBlack4, x1Border, y1Border + 2 \* lineSpace, x2Border, y2Border + 2 \* lineSpace, borderDistance, numberOfBorder);

// Draw a String in a box on the PictureBox For Slave(Train) ID

g.FillRectangle(myBrushYellow, positionXOfSlaveIndicator, positionYOfSlaveIndicator, widthOfSlaveIndicator, heightOfSlaveIndicator);

g.DrawRectangle(myPenBlack1, positionXOfSlaveIndicator, positionYOfSlaveIndicator, widthOfSlaveIndicator, heightOfSlaveIndicator);

g.DrawString(slaveID, drawFont, drawBrush, positionXOfSlaveIndicator+4, positionYOfSlaveIndicator);

}

// Creating Train Stations

private int[] createTrainStations(Graphics g, int lineSpace, int SNLP, int SNUP, Pen pen)

{

int[] position = { 0, 0 };

int numLower = SNLP;// station number Lower Part

int numUpper = SNUP; // station number Upper Part

int howManyRowShiftFromOneBoxToAnother = 5;

// Draw Station Number i.e. 1 to 10

System.Drawing.Font drawFont = new System.Drawing.Font("Arial", 12);

System.Drawing.SolidBrush myBrushLightPink = new System.Drawing.SolidBrush(Color.LightPink);

SolidBrush drawBrushPlatForm = new SolidBrush(Color.Black);

// For Stations 1 to 4

int stationOnePositionX = 11 \* lineSpace;

int stationPositionX = 0;

int stationOnePositionY = 7 \* lineSpace;

position[0] = stationOnePositionY;

int stationNo = 1;

for (int l = 0; l < numLower; l++)

{

stationPositionX = l \* lineSpace \* howManyRowShiftFromOneBoxToAnother + stationOnePositionX;

g.FillRectangle(myBrushLightPink, stationPositionX, stationOnePositionY, lineSpace, lineSpace);

g.DrawRectangle(pen, stationPositionX, stationOnePositionY, lineSpace, lineSpace);

g.DrawString(Convert.ToString(stationNo++), drawFont, drawBrushPlatForm, stationPositionX+5 , stationOnePositionY+4 );

int [] positionLowerSlave = new int [] {stationPositionX,stationOnePositionY-(lineSpace+lineSpace/4)};

if (!slaveIDAndPosition.ContainsKey(slaveAddressLowerPart[l])) { slaveIDAndPosition.Add(slaveAddressLowerPart[l], positionLowerSlave); }

}

// For Stations 5 to 10

int stationTenPositionX = 6 \* lineSpace;

int positionXOfStation = 0;

int stationTenPositionY = 2 \* lineSpace;

position[1] = stationTenPositionY;

stationNo = numLower + numUpper; // total sum of stations and also starting string name of the upper part of our picture Box1

for (int m = 0; m < numUpper; m++)

{

positionXOfStation = (m \* lineSpace \* howManyRowShiftFromOneBoxToAnother) + stationTenPositionX;

g.FillRectangle(myBrushLightPink, positionXOfStation, stationTenPositionY, lineSpace, lineSpace);

g.DrawRectangle(pen, positionXOfStation, stationTenPositionY, lineSpace, lineSpace);

g.DrawString(Convert.ToString(stationNo--), drawFont, drawBrushPlatForm, positionXOfStation+5 , stationTenPositionY+4);

int[] positionUpperSlave = { positionXOfStation, stationTenPositionY+(lineSpace+lineSpace/2+lineSpace/4)};

if (!slaveIDAndPosition.ContainsKey(slaveAddressUpperPart[m])) { slaveIDAndPosition.Add(slaveAddressUpperPart[m], positionUpperSlave); }

}

return position;

}

// Creating Platforms in our GUI

private void createPlatForms(Graphics g, Pen pen, int stationOnePositionY, int stationTenPositionY, int lineSpace)

{

// creating platform

int platFormOneColumnShift = 8;

int platFormOneRowShift = stationOnePositionY - (1 \* lineSpace);

int lengthOfLowerPlatform = 22 \* lineSpace;

int platFormOnePositionX1 = lineSpace \* platFormOneColumnShift;

// here variable name is platFormOnePosition, "one" since during the drawing process we start from station one

int platFormOnePositionY = platFormOneRowShift;

int platFormOnePositionX2 = platFormOnePositionX1 + lengthOfLowerPlatform;

// black track

System.Drawing.Pen myPenBlack10 = new System.Drawing.Pen(Color.Black, 10.0f);

int blackX1 = platFormOnePositionX1;

int blackY1 = platFormOnePositionY;

int blackX2 = platFormOnePositionX1 - 4 \* lineSpace;

int blackY2 = platFormOnePositionY;

drawPlatform(g, myPenBlack10, blackX1, blackY1, blackX2, blackY2);

blackX1 = blackX2;

blackY1 = blackY2;

blackX2 = platFormOnePositionX2 + 4 \* lineSpace;

blackY2 = platFormOnePositionY;

drawPlatform(g, myPenBlack10, blackX1, blackY1, blackX2, blackY2);

blackX1 = blackX2;

blackY1 = blackY2;

blackX2 = blackX1 + (int)(0.5 \* lineSpace);

blackY2 = blackY2 + (int)(0.5 \* lineSpace);

drawPlatform(g, myPenBlack10, blackX1, blackY1, blackX2, blackY2);

blackX1 = blackX2;

blackY1 = blackY2;

blackY2 = blackY2 + lineSpace;

drawPlatform(g, myPenBlack10, blackX1, blackY1, blackX2, blackY2);

blackX1 = platFormOnePositionX1 + lineSpace;

blackY1 = platFormOnePositionY;

blackX2 = platFormOnePositionX1 + 3 \* lineSpace;

blackY2 = platFormOnePositionY - 2 \* lineSpace;

drawPlatform(g, myPenBlack10, blackX1, blackY1, blackX2, blackY2);

blackX1 = platFormOnePositionX2 - lineSpace;

blackY1 = platFormOnePositionY;

blackX2 = platFormOnePositionX2 - 3 \* lineSpace;

blackY2 = platFormOnePositionY - 2 \* lineSpace;

drawPlatform(g, myPenBlack10, blackX1, blackY1, blackX2, blackY2);

//platform 1 to 4

drawPlatform(g, pen, platFormOnePositionX1, platFormOnePositionY, platFormOnePositionX2, platFormOnePositionY);

//platform 5 to 10

int platFormTenColumnShift = 4;

int platFormTenRowShift = stationTenPositionY + (2 \* lineSpace); ;

int lengthOfUpperPlatform = 30 \* lineSpace;

int platFormTenPositionY = platFormTenRowShift;

int platFormTenPositionX1 = lineSpace \* platFormTenColumnShift;

int platFormTenPositionX2 = platFormTenPositionX1 + lengthOfUpperPlatform;

drawPlatform(g, pen, platFormTenPositionX1, platFormTenPositionY, platFormTenPositionX2, platFormTenPositionY);

// rest Part Of Tracks : Left Side

int x1Temp = platFormTenPositionX1 - (2 \* lineSpace);

int y1Temp = platFormTenPositionY + (2 \* lineSpace);

int x2Temp = platFormTenPositionX1;

int y2Temp = platFormTenPositionY;

drawPlatform(g, pen, x1Temp, y1Temp, x2Temp, y2Temp);

x2Temp = x1Temp;

y2Temp = y1Temp + (2 \* lineSpace);

drawPlatform(g, pen, x1Temp, y1Temp, x2Temp, y2Temp);

x1Temp = x2Temp;

y1Temp = y2Temp;

x2Temp = platFormTenPositionX1;

y2Temp = platFormTenPositionY + (6 \* lineSpace);

drawPlatform(g, pen, x1Temp, y1Temp, x2Temp, y2Temp);

x1Temp = x2Temp;

y1Temp = y2Temp;

x2Temp = x2Temp + (1 \* lineSpace);

drawPlatform(g, pen, x1Temp, y1Temp, x2Temp, y2Temp);

x1Temp = x2Temp;

y1Temp = y2Temp;

x2Temp = platFormOnePositionX1;

y2Temp = platFormOnePositionY;

drawPlatform(g, pen, x1Temp, y1Temp, x2Temp, y2Temp);

// rest Part Of Tracks : Right Side

x1Temp = platFormTenPositionX2;

y1Temp = platFormTenPositionY;

x2Temp = platFormTenPositionX2 + (2 \* lineSpace);

y2Temp = platFormTenPositionY + (2 \* lineSpace);

drawPlatform(g, pen, x1Temp, y1Temp, x2Temp, y2Temp);

x1Temp = x2Temp;

y1Temp = y2Temp;

x2Temp = x1Temp;

y2Temp = y2Temp + (2 \* lineSpace);

drawPlatform(g, pen, x1Temp, y1Temp, x2Temp, y2Temp);

x1Temp = x2Temp;

y1Temp = y2Temp;

x2Temp = x2Temp - (2 \* lineSpace);

y2Temp = y2Temp + (2 \* lineSpace);

drawPlatform(g, pen, x1Temp, y1Temp, x2Temp, y2Temp);

x1Temp = x2Temp;

y1Temp = y2Temp;

x2Temp = x2Temp - (1 \* lineSpace);

drawPlatform(g, pen, x1Temp, y1Temp, x2Temp, y2Temp);

x1Temp = x2Temp;

y1Temp = y2Temp;

x2Temp = platFormOnePositionX2;

y2Temp = platFormOnePositionY;

drawPlatform(g, pen, x1Temp, y1Temp, x2Temp, y2Temp);

}

private void drawPlatform(Graphics g, Pen pen, int x1, int y1, int x2, int y2)

{

g.DrawLine(pen, x1, y1, x2, y2);

}

private void drawBorderOfPlatForm(Graphics g, Pen pen, int x1, int y1, int x2, int y2, int borderDistance, int number)

{

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

{

drawPlatform(g, pen, x1, y1, x2, y2);

x1 = x1 + borderDistance;

x2 = x2 + borderDistance;

}

}

// This Method selects The position of Slave Id according to its Name, i.e SlaveID which is Key in our Dictonary :

private void serialPort1\_DataReceived(object sender, System.IO.Ports.SerialDataReceivedEventArgs e)

{

String input = this.serialPort1.ReadLine();

int inputIntoINT = Int32.Parse(input);

String ID = inputIntoINT.ToString();

this.slaveID = ID;

int[] tempPosition = { 0, 0 };

if (this.slaveIDAndPosition.ContainsKey(ID))

{

tempPosition = this.slaveIDAndPosition[ID];

}

this.positionXOfSlaveIndicator = tempPosition[0];

this.positionYOfSlaveIndicator = tempPosition[1];

this.pictureBox1.Invalidate();

}

// Here below in this method, we will specified what ?? to do when someone clicks the START button

private void button1Starten\_Click(object sender, EventArgs e)

{

try

{

if (comboBox1AvailablePorts.Text == "")

{

MessageBox.Show(" Please Select Ports ");

}

else if (comboBox2BaudRates.Text == "")

{

MessageBox.Show(" Please Select Baud Rate Setting:");

}

else

{

serialPort1.PortName = comboBox1AvailablePorts.Text;

serialPort1.BaudRate = Convert.ToInt32(comboBox2BaudRates.Text);

serialPort1.Open();

button2Stop.Enabled = true;

button1Starten.Enabled = false;

}

}

catch (UnauthorizedAccessException)

{

MessageBox.Show(" Something Went Wrong Please Try Again ");

}

}

// Here below in this method, we will specified what ?? to do when someone clicks the STOP button

private void button2Stop\_Click(object sender, EventArgs e)

{

this.serialPort1.Close(); // stops the serial communication by closing the port

this.button1Starten.Enabled = true;

this.button2Stop.Enabled = false;

}

// Here below in this method, we will specified what ?? to do when someone clicks the INFO button

private void button4Info\_Click(object sender, EventArgs e)

{

MessageBox.Show(" You Can See Here The Transition of Train From One Trainstation to Other ");

}

}

}

**Form1.Designer.cs :**

namespace Tracks

{

partial class Form1

{

/// <summary>

/// Required designer variable.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Clean up any resources being used.

/// </summary>

/// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

private void InitializeComponent()

{

this.components = new System.ComponentModel.Container();

this.button1Starten = new System.Windows.Forms.Button();

this.button3Weiter = new System.Windows.Forms.Button();

this.pictureBox1 = new System.Windows.Forms.PictureBox();

this.button4Info = new System.Windows.Forms.Button();

this.textBox1GleisPlan = new System.Windows.Forms.TextBox();

this.textBox2RaumNr = new System.Windows.Forms.TextBox();

this.dateTimePicker1 = new System.Windows.Forms.DateTimePicker();

this.panel1 = new System.Windows.Forms.Panel();

this.label2BaudRates = new System.Windows.Forms.Label();

this.label1Ports = new System.Windows.Forms.Label();

this.comboBox2BaudRates = new System.Windows.Forms.ComboBox();

this.comboBox1AvailablePorts = new System.Windows.Forms.ComboBox();

this.button2Stop = new System.Windows.Forms.Button();

this.serialPort1 = new System.IO.Ports.SerialPort(this.components);

((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).BeginInit();

this.panel1.SuspendLayout();

this.SuspendLayout();

//

// button1Starten

//

this.button1Starten.Font = new System.Drawing.Font("Arial Black", 8.25F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));

this.button1Starten.ForeColor = System.Drawing.Color.FromArgb(((int)(((byte)(0)))), ((int)(((byte)(192)))), ((int)(((byte)(0)))));

this.button1Starten.Location = new System.Drawing.Point(3, 32);

this.button1Starten.Name = "button1Starten";

this.button1Starten.Size = new System.Drawing.Size(75, 23);

this.button1Starten.TabIndex = 0;

this.button1Starten.Text = "STARTEN";

this.button1Starten.UseVisualStyleBackColor = true;

this.button1Starten.Click += new System.EventHandler(this.button1Starten\_Click);

//

// button3Weiter

//

this.button3Weiter.Font = new System.Drawing.Font("Arial Black", 8.25F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));

this.button3Weiter.Location = new System.Drawing.Point(164, 31);

this.button3Weiter.Name = "button3Weiter";

this.button3Weiter.Size = new System.Drawing.Size(75, 23);

this.button3Weiter.TabIndex = 1;

this.button3Weiter.Text = "WEITER";

this.button3Weiter.UseVisualStyleBackColor = true;

//

// pictureBox1

//

this.pictureBox1.BackColor = System.Drawing.Color.White;

this.pictureBox1.BorderStyle = System.Windows.Forms.BorderStyle.Fixed3D;

this.pictureBox1.Dock = System.Windows.Forms.DockStyle.Bottom;

this.pictureBox1.Location = new System.Drawing.Point(0, 60);

this.pictureBox1.Name = "pictureBox1";

this.pictureBox1.Size = new System.Drawing.Size(1200, 562);

this.pictureBox1.TabIndex = 2;

this.pictureBox1.TabStop = false;

//

// button4Info

//

this.button4Info.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));

this.button4Info.ForeColor = System.Drawing.Color.FromArgb(((int)(((byte)(0)))), ((int)(((byte)(0)))), ((int)(((byte)(192)))));

this.button4Info.Location = new System.Drawing.Point(244, 31);

this.button4Info.Margin = new System.Windows.Forms.Padding(2, 2, 2, 2);

this.button4Info.Name = "button4Info";

this.button4Info.Size = new System.Drawing.Size(75, 23);

this.button4Info.TabIndex = 3;

this.button4Info.Text = "INFO";

this.button4Info.UseVisualStyleBackColor = true;

this.button4Info.Click += new System.EventHandler(this.button4Info\_Click);

//

// textBox1GleisPlan

//

this.textBox1GleisPlan.BackColor = System.Drawing.SystemColors.Window;

this.textBox1GleisPlan.Location = new System.Drawing.Point(822, 32);

this.textBox1GleisPlan.Margin = new System.Windows.Forms.Padding(2, 2, 2, 2);

this.textBox1GleisPlan.Name = "textBox1GleisPlan";

this.textBox1GleisPlan.ReadOnly = true;

this.textBox1GleisPlan.Size = new System.Drawing.Size(76, 20);

this.textBox1GleisPlan.TabIndex = 4;

this.textBox1GleisPlan.Text = " Gleis Plan";

this.textBox1GleisPlan.TextAlign = System.Windows.Forms.HorizontalAlignment.Center;

//

// textBox2RaumNr

//

this.textBox2RaumNr.BackColor = System.Drawing.SystemColors.Window;

this.textBox2RaumNr.Location = new System.Drawing.Point(902, 32);

this.textBox2RaumNr.Margin = new System.Windows.Forms.Padding(2, 2, 2, 2);

this.textBox2RaumNr.Name = "textBox2RaumNr";

this.textBox2RaumNr.ReadOnly = true;

this.textBox2RaumNr.Size = new System.Drawing.Size(76, 20);

this.textBox2RaumNr.TabIndex = 5;

this.textBox2RaumNr.Text = " Raum 804";

this.textBox2RaumNr.TextAlign = System.Windows.Forms.HorizontalAlignment.Center;

//

// dateTimePicker1

//

this.dateTimePicker1.Location = new System.Drawing.Point(982, 32);

this.dateTimePicker1.Margin = new System.Windows.Forms.Padding(2, 2, 2, 2);

this.dateTimePicker1.Name = "dateTimePicker1";

this.dateTimePicker1.Size = new System.Drawing.Size(192, 20);

this.dateTimePicker1.TabIndex = 6;

//

// panel1

//

this.panel1.Controls.Add(this.label2BaudRates);

this.panel1.Controls.Add(this.label1Ports);

this.panel1.Controls.Add(this.comboBox2BaudRates);

this.panel1.Controls.Add(this.comboBox1AvailablePorts);

this.panel1.Controls.Add(this.button2Stop);

this.panel1.Controls.Add(this.button1Starten);

this.panel1.Controls.Add(this.dateTimePicker1);

this.panel1.Controls.Add(this.button3Weiter);

this.panel1.Controls.Add(this.textBox2RaumNr);

this.panel1.Controls.Add(this.button4Info);

this.panel1.Controls.Add(this.textBox1GleisPlan);

this.panel1.Dock = System.Windows.Forms.DockStyle.Top;

this.panel1.Location = new System.Drawing.Point(0, 0);

this.panel1.Margin = new System.Windows.Forms.Padding(2, 2, 2, 2);

this.panel1.Name = "panel1";

this.panel1.Size = new System.Drawing.Size(1200, 60);

this.panel1.TabIndex = 7;

//

// label2BaudRates

//

this.label2BaudRates.AutoSize = true;

this.label2BaudRates.Font = new System.Drawing.Font("Bahnschrift SemiBold", 8F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));

this.label2BaudRates.Location = new System.Drawing.Point(422, 16);

this.label2BaudRates.Name = "label2BaudRates";

this.label2BaudRates.Size = new System.Drawing.Size(62, 13);

this.label2BaudRates.TabIndex = 11;

this.label2BaudRates.Text = "Baud Rates";

//

// label1Ports

//

this.label1Ports.AutoSize = true;

this.label1Ports.Font = new System.Drawing.Font("Bahnschrift SemiBold", 8F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));

this.label1Ports.Location = new System.Drawing.Point(345, 16);

this.label1Ports.Name = "label1Ports";

this.label1Ports.Size = new System.Drawing.Size(34, 13);

this.label1Ports.TabIndex = 10;

this.label1Ports.Text = "Ports";

//

// comboBox2BaudRates

//

this.comboBox2BaudRates.FormattingEnabled = true;

this.comboBox2BaudRates.Location = new System.Drawing.Point(405, 32);

this.comboBox2BaudRates.Name = "comboBox2BaudRates";

this.comboBox2BaudRates.Size = new System.Drawing.Size(90, 21);

this.comboBox2BaudRates.TabIndex = 9;

//

// comboBox1AvailablePorts

//

this.comboBox1AvailablePorts.FormattingEnabled = true;

this.comboBox1AvailablePorts.Location = new System.Drawing.Point(324, 32);

this.comboBox1AvailablePorts.Name = "comboBox1AvailablePorts";

this.comboBox1AvailablePorts.Size = new System.Drawing.Size(75, 21);

this.comboBox1AvailablePorts.TabIndex = 8;

//

// button2Stop

//

this.button2Stop.Font = new System.Drawing.Font("Arial Black", 8.25F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));

this.button2Stop.ForeColor = System.Drawing.Color.FromArgb(((int)(((byte)(192)))), ((int)(((byte)(0)))), ((int)(((byte)(0)))));

this.button2Stop.Location = new System.Drawing.Point(83, 32);

this.button2Stop.Margin = new System.Windows.Forms.Padding(2, 2, 2, 2);

this.button2Stop.Name = "button2Stop";

this.button2Stop.Size = new System.Drawing.Size(75, 23);

this.button2Stop.TabIndex = 7;

this.button2Stop.Text = "STOP";

this.button2Stop.UseVisualStyleBackColor = true;

this.button2Stop.Click += new System.EventHandler(this.button2Stop\_Click);

//

// serialPort1

//

this.serialPort1.DataReceived += new System.IO.Ports.SerialDataReceivedEventHandler(this.serialPort1\_DataReceived);

//

// Form1

//

this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.BackColor = System.Drawing.SystemColors.GradientInactiveCaption;

this.ClientSize = new System.Drawing.Size(1200, 622);

this.Controls.Add(this.panel1);

this.Controls.Add(this.pictureBox1);

this.Name = "Form1";

this.Text = "Gleisplan";

((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).EndInit();

this.panel1.ResumeLayout(false);

this.panel1.PerformLayout();

this.ResumeLayout(false);

}

#endregion

private System.Windows.Forms.Button button1Starten;

private System.Windows.Forms.Button button2Stop;

private System.Windows.Forms.Button button3Weiter;

private System.Windows.Forms.Button button4Info;

private System.Windows.Forms.PictureBox pictureBox1;

private System.Windows.Forms.TextBox textBox1GleisPlan;

private System.Windows.Forms.TextBox textBox2RaumNr;

private System.Windows.Forms.DateTimePicker dateTimePicker1;

private System.Windows.Forms.Panel panel1;

public System.IO.Ports.SerialPort serialPort1;

private System.Windows.Forms.ComboBox comboBox2BaudRates;

private System.Windows.Forms.ComboBox comboBox1AvailablePorts;

private System.Windows.Forms.Label label2BaudRates;

private System.Windows.Forms.Label label1Ports;

}

}

**Track Plan:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.IO.Ports;

namespace Tracks

{

public partial class Form1 : Form

{

private string slaveID;

private static String[] slaveAddressLowerPart = new String[] {"21","23","27","29"};

private static String[] slaveAddressUpperPart = new String[] {"22","24","28","30","26","25"};

Dictionary<String, int[]> slaveIDAndPosition = new Dictionary<String, int[]>();

private const int lineSpace = 30 ;

private int widthOfSlaveIndicator;

private int heightOfSlaveIndicator; // taking only half of its width

private int positionXOfSlaveIndicator; // position adjustment to the vertical line

private int positionYOfSlaveIndicator; // lineSpace/2+lineSpace/4 is nothing but adjustment to the Horizontal line

public Form1()

{

InitializeComponent();

getAvailablePorts();

slaveID = " ";

widthOfSlaveIndicator = lineSpace;

heightOfSlaveIndicator = (int)(0.5 \* widthOfSlaveIndicator); // taking only half of its width

positionXOfSlaveIndicator = lineSpace \* 31; // position adjustment to the vertical line

positionYOfSlaveIndicator = (lineSpace \* 5) + lineSpace / 2 + lineSpace / 4; // lineSpace/2+lineSpace/4 is nothing but adjustment to the Horizontal line

button1Starten.Enabled = true;

button2Stop.Enabled = false;

// connect the paint event of the picture box to the event handler method.

pictureBox1.Paint += new System.Windows.Forms.PaintEventHandler(this.pictureBox1\_Paint);

// Add the pictureBox control to the Form.

this.Controls.Add(pictureBox1);

}

// method theat finds the available Ports

private void getAvailablePorts()

{

string[] ports = SerialPort.GetPortNames();

comboBox1AvailablePorts.Items.AddRange(ports);

//String[] baud = { "9600", "11500"};

//comboBox2BaudRates.Items.AddRange(baud);

comboBox2BaudRates.Items.Add("9600");

}

private void pictureBox1\_Paint(object sender, PaintEventArgs e)

{

// creating pens with different colors

System.Drawing.Pen myPenBlack10 = new System.Drawing.Pen(Color.Black, 10.0f);

System.Drawing.Pen myPenGreen10 = new System.Drawing.Pen(Color.Green, 10.0f);

System.Drawing.Pen myPenRed10 = new System.Drawing.Pen(Color.Red, 10.0f);

System.Drawing.Pen myPenBlack4 = new System.Drawing.Pen(Color.Black, 4.0f);

System.Drawing.Pen myPenLightBlue1 = new System.Drawing.Pen(Color.LightBlue, 1.0f);

System.Drawing.Pen myPenBlack1 = new System.Drawing.Pen(Color.Black, 1.0f);

// Drawing Fonts

System.Drawing.Font drawFont = new System.Drawing.Font("Arial", 11);

System.Drawing.SolidBrush myBrushYellow = new System.Drawing.SolidBrush(Color.Yellow);

SolidBrush drawBrush = new SolidBrush(Color.Black);

// Create a local version of the graphics object for the pictureBox

Graphics g = e.Graphics;

// Draw Grid

int gridWidth = this.ClientSize.Width;

int gridHeight = this.ClientSize.Height;

int noOfVerticalLines = (int)(gridWidth / lineSpace);

int noOfHorizontalLines = (int)(gridHeight / lineSpace);

for (int i = 0; i < noOfVerticalLines; i++) { g.DrawLine(myPenLightBlue1, i \* lineSpace, 0, i \* lineSpace, gridHeight); }

for (int j = 0; j < noOfHorizontalLines; j++) { g.DrawLine(myPenLightBlue1, 0, j \* lineSpace, gridWidth, j \* lineSpace); }

int stationNumberLowerPart = 4;

int StationNumberUpperPart = 6;

int[] positionY = createTrainStations(g, lineSpace, stationNumberLowerPart, StationNumberUpperPart, myPenBlack1);

int lowerPartStationPositionY = positionY[0];

int upperPartStationPositionY = positionY[1];

createPlatForms(g, myPenGreen10, lowerPartStationPositionY, upperPartStationPositionY, lineSpace);

// draw border of the platform

int x1Border = 9 \* lineSpace;

int y1Border = upperPartStationPositionY + (int)(1.6 \* lineSpace);

int x2Border = 9 \* lineSpace; ;

int y2Border = y1Border + (int)(0.8 \* lineSpace);

int borderDistance = 5 \* lineSpace;

int numberOfBorder = 5;

drawBorderOfPlatForm(g, myPenBlack4, x1Border, y1Border, x2Border, y2Border, borderDistance, numberOfBorder);

drawBorderOfPlatForm(g, myPenBlack4, x1Border, y1Border + 2 \* lineSpace, x2Border, y2Border + 2 \* lineSpace, borderDistance, numberOfBorder);

// Draw a String in a box on the PictureBox For Slave(Train) ID

g.FillRectangle(myBrushYellow, positionXOfSlaveIndicator, positionYOfSlaveIndicator, widthOfSlaveIndicator, heightOfSlaveIndicator);

g.DrawRectangle(myPenBlack1, positionXOfSlaveIndicator, positionYOfSlaveIndicator, widthOfSlaveIndicator, heightOfSlaveIndicator);

g.DrawString(slaveID, drawFont, drawBrush, positionXOfSlaveIndicator+4, positionYOfSlaveIndicator);

}

// Creating Train Stations

private int[] createTrainStations(Graphics g, int lineSpace, int SNLP, int SNUP, Pen pen)

{

int[] position = { 0, 0 };

int numLower = SNLP;// station number Lower Part

int numUpper = SNUP; // station number Upper Part

int howManyRowShiftFromOneBoxToAnother = 5;

// Draw Station Number i.e. 1 to 10

System.Drawing.Font drawFont = new System.Drawing.Font("Arial", 12);

System.Drawing.SolidBrush myBrushLightPink = new System.Drawing.SolidBrush(Color.LightPink);

SolidBrush drawBrushPlatForm = new SolidBrush(Color.Black);

// For Stations 1 to 4

int stationOnePositionX = 11 \* lineSpace;

int stationPositionX = 0;

int stationOnePositionY = 7 \* lineSpace;

position[0] = stationOnePositionY;

int stationNo = 1;

for (int l = 0; l < numLower; l++)

{

stationPositionX = l \* lineSpace \* howManyRowShiftFromOneBoxToAnother + stationOnePositionX;

g.FillRectangle(myBrushLightPink, stationPositionX, stationOnePositionY, lineSpace, lineSpace);

g.DrawRectangle(pen, stationPositionX, stationOnePositionY, lineSpace, lineSpace);

g.DrawString(Convert.ToString(stationNo++), drawFont, drawBrushPlatForm, stationPositionX+5 , stationOnePositionY+4 );

int [] positionLowerSlave = new int [] {stationPositionX,stationOnePositionY-(lineSpace+lineSpace/4)};

if (!slaveIDAndPosition.ContainsKey(slaveAddressLowerPart[l])) { slaveIDAndPosition.Add(slaveAddressLowerPart[l], positionLowerSlave); }

}

// For Stations 5 to 10

int stationTenPositionX = 6 \* lineSpace;

int positionXOfStation = 0;

int stationTenPositionY = 2 \* lineSpace;

position[1] = stationTenPositionY;

stationNo = numLower + numUpper; // total sum of stations and also starting string name of the upper part of our picture Box1

for (int m = 0; m < numUpper; m++)

{

positionXOfStation = (m \* lineSpace \* howManyRowShiftFromOneBoxToAnother) + stationTenPositionX;

g.FillRectangle(myBrushLightPink, positionXOfStation, stationTenPositionY, lineSpace, lineSpace);

g.DrawRectangle(pen, positionXOfStation, stationTenPositionY, lineSpace, lineSpace);

g.DrawString(Convert.ToString(stationNo--), drawFont, drawBrushPlatForm, positionXOfStation+5 , stationTenPositionY+4);

int[] positionUpperSlave = { positionXOfStation, stationTenPositionY+(lineSpace+lineSpace/2+lineSpace/4)};

if (!slaveIDAndPosition.ContainsKey(slaveAddressUpperPart[m])) { slaveIDAndPosition.Add(slaveAddressUpperPart[m], positionUpperSlave); }

}

return position;

}

// Creating Platforms in our GUI

private void createPlatForms(Graphics g, Pen pen, int stationOnePositionY, int stationTenPositionY, int lineSpace)

{

// creating platform

int platFormOneColumnShift = 8;

int platFormOneRowShift = stationOnePositionY - (1 \* lineSpace);

int lengthOfLowerPlatform = 22 \* lineSpace;

int platFormOnePositionX1 = lineSpace \* platFormOneColumnShift;

// here variable name is platFormOnePosition, "one" since during the drawing process we start from station one

int platFormOnePositionY = platFormOneRowShift;

int platFormOnePositionX2 = platFormOnePositionX1 + lengthOfLowerPlatform;

// black track

System.Drawing.Pen myPenBlack10 = new System.Drawing.Pen(Color.Black, 10.0f);

int blackX1 = platFormOnePositionX1;

int blackY1 = platFormOnePositionY;

int blackX2 = platFormOnePositionX1 - 4 \* lineSpace;

int blackY2 = platFormOnePositionY;

drawPlatform(g, myPenBlack10, blackX1, blackY1, blackX2, blackY2);

blackX1 = blackX2;

blackY1 = blackY2;

blackX2 = platFormOnePositionX2 + 4 \* lineSpace;

blackY2 = platFormOnePositionY;

drawPlatform(g, myPenBlack10, blackX1, blackY1, blackX2, blackY2);

blackX1 = blackX2;

blackY1 = blackY2;

blackX2 = blackX1 + (int)(0.5 \* lineSpace);

blackY2 = blackY2 + (int)(0.5 \* lineSpace);

drawPlatform(g, myPenBlack10, blackX1, blackY1, blackX2, blackY2);

blackX1 = blackX2;

blackY1 = blackY2;

blackY2 = blackY2 + lineSpace;

drawPlatform(g, myPenBlack10, blackX1, blackY1, blackX2, blackY2);

blackX1 = platFormOnePositionX1 + lineSpace;

blackY1 = platFormOnePositionY;

blackX2 = platFormOnePositionX1 + 3 \* lineSpace;

blackY2 = platFormOnePositionY - 2 \* lineSpace;

drawPlatform(g, myPenBlack10, blackX1, blackY1, blackX2, blackY2);

blackX1 = platFormOnePositionX2 - lineSpace;

blackY1 = platFormOnePositionY;

blackX2 = platFormOnePositionX2 - 3 \* lineSpace;

blackY2 = platFormOnePositionY - 2 \* lineSpace;

drawPlatform(g, myPenBlack10, blackX1, blackY1, blackX2, blackY2);

//platform 1 to 4

drawPlatform(g, pen, platFormOnePositionX1, platFormOnePositionY, platFormOnePositionX2, platFormOnePositionY);

//platform 5 to 10

int platFormTenColumnShift = 4;

int platFormTenRowShift = stationTenPositionY + (2 \* lineSpace); ;

int lengthOfUpperPlatform = 30 \* lineSpace;

int platFormTenPositionY = platFormTenRowShift;

int platFormTenPositionX1 = lineSpace \* platFormTenColumnShift;

int platFormTenPositionX2 = platFormTenPositionX1 + lengthOfUpperPlatform;

drawPlatform(g, pen, platFormTenPositionX1, platFormTenPositionY, platFormTenPositionX2, platFormTenPositionY);

// rest Part Of Tracks : Left Side

int x1Temp = platFormTenPositionX1 - (2 \* lineSpace);

int y1Temp = platFormTenPositionY + (2 \* lineSpace);

int x2Temp = platFormTenPositionX1;

int y2Temp = platFormTenPositionY;

drawPlatform(g, pen, x1Temp, y1Temp, x2Temp, y2Temp);

x2Temp = x1Temp;

y2Temp = y1Temp + (2 \* lineSpace);

drawPlatform(g, pen, x1Temp, y1Temp, x2Temp, y2Temp);

x1Temp = x2Temp;

y1Temp = y2Temp;

x2Temp = platFormTenPositionX1;

y2Temp = platFormTenPositionY + (6 \* lineSpace);

drawPlatform(g, pen, x1Temp, y1Temp, x2Temp, y2Temp);

x1Temp = x2Temp;

y1Temp = y2Temp;

x2Temp = x2Temp + (1 \* lineSpace);

drawPlatform(g, pen, x1Temp, y1Temp, x2Temp, y2Temp);

x1Temp = x2Temp;

y1Temp = y2Temp;

x2Temp = platFormOnePositionX1;

y2Temp = platFormOnePositionY;

drawPlatform(g, pen, x1Temp, y1Temp, x2Temp, y2Temp);

// rest Part Of Tracks : Right Side

x1Temp = platFormTenPositionX2;

y1Temp = platFormTenPositionY;

x2Temp = platFormTenPositionX2 + (2 \* lineSpace);

y2Temp = platFormTenPositionY + (2 \* lineSpace);

drawPlatform(g, pen, x1Temp, y1Temp, x2Temp, y2Temp);

x1Temp = x2Temp;

y1Temp = y2Temp;

x2Temp = x1Temp;

y2Temp = y2Temp + (2 \* lineSpace);

drawPlatform(g, pen, x1Temp, y1Temp, x2Temp, y2Temp);

x1Temp = x2Temp;

y1Temp = y2Temp;

x2Temp = x2Temp - (2 \* lineSpace);

y2Temp = y2Temp + (2 \* lineSpace);

drawPlatform(g, pen, x1Temp, y1Temp, x2Temp, y2Temp);

x1Temp = x2Temp;

y1Temp = y2Temp;

x2Temp = x2Temp - (1 \* lineSpace);

drawPlatform(g, pen, x1Temp, y1Temp, x2Temp, y2Temp);

x1Temp = x2Temp;

y1Temp = y2Temp;

x2Temp = platFormOnePositionX2;

y2Temp = platFormOnePositionY;

drawPlatform(g, pen, x1Temp, y1Temp, x2Temp, y2Temp);

}

private void drawPlatform(Graphics g, Pen pen, int x1, int y1, int x2, int y2)

{

g.DrawLine(pen, x1, y1, x2, y2);

}

private void drawBorderOfPlatForm(Graphics g, Pen pen, int x1, int y1, int x2, int y2, int borderDistance, int number)

{

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

{

drawPlatform(g, pen, x1, y1, x2, y2);

x1 = x1 + borderDistance;

x2 = x2 + borderDistance;

}

}

// This Method selects The position of Slave Id according to its Name, i.e SlaveID which is Key in our Dictonary :

private void serialPort1\_DataReceived(object sender, System.IO.Ports.SerialDataReceivedEventArgs e)

{

String input = this.serialPort1.ReadLine();

int inputIntoINT = Int32.Parse(input);

String ID = inputIntoINT.ToString();

this.slaveID = ID;

int[] tempPosition = { 0, 0 };

if (this.slaveIDAndPosition.ContainsKey(ID))

{

tempPosition = this.slaveIDAndPosition[ID];

}

this.positionXOfSlaveIndicator = tempPosition[0];

this.positionYOfSlaveIndicator = tempPosition[1];

this.pictureBox1.Invalidate();

}

// Here below in this method, we will specified what ?? to do when someone clicks the START button

private void button1Starten\_Click(object sender, EventArgs e)

{

try

{

if (comboBox1AvailablePorts.Text == "")

{

MessageBox.Show(" Please Select Ports ");

}

else if (comboBox2BaudRates.Text == "")

{

MessageBox.Show(" Please Select Baud Rate Setting:");

}

else

{

serialPort1.PortName = comboBox1AvailablePorts.Text;

serialPort1.BaudRate = Convert.ToInt32(comboBox2BaudRates.Text);

serialPort1.Open();

button2Stop.Enabled = true;

button1Starten.Enabled = false;

}

}

catch (UnauthorizedAccessException)

{

MessageBox.Show(" Something Went Wrong Please Try Again ");

}

}

// Here below in this method, we will specified what ?? to do when someone clicks the STOP button

private void button2Stop\_Click(object sender, EventArgs e)

{

this.serialPort1.Close(); // stops the serial communication by closing the port

this.button1Starten.Enabled = true;

this.button2Stop.Enabled = false;

}

// Here below in this method, we will specified what ?? to do when someone clicks the INFO button

private void button4Info\_Click(object sender, EventArgs e)

{

MessageBox.Show(" You Can See Here The Transition of Train From One Trainstation to Other ");

}

}

}

**Conclusions :**

To sum up, we got to achieve another portion of knowledge and skills during our work with project, especially with having the possibility to realize the control of HW (Arduino) through SW (Arduino and Visual Studio).