

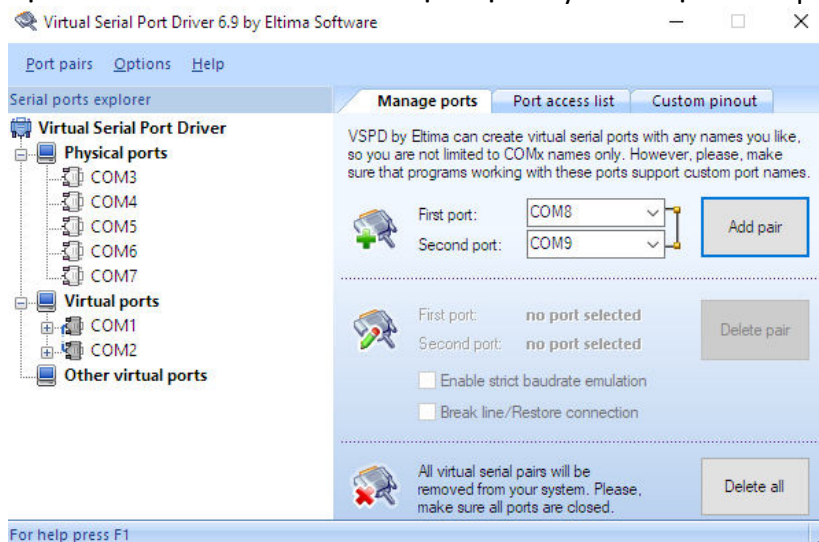
BÁO CÁO BÀI TẬP MÔN ĐO LƯỜNG VÀ ĐIỀU KHIỂN BẰNG MÁY TÍNH

Đề tài: Giao tiếp vi điều khiển và máy tính và điều khiển đèn giao thông

A. Công cụ thực hiện

A.1. Phần mềm tạo cổng COM ảo nối tiếp

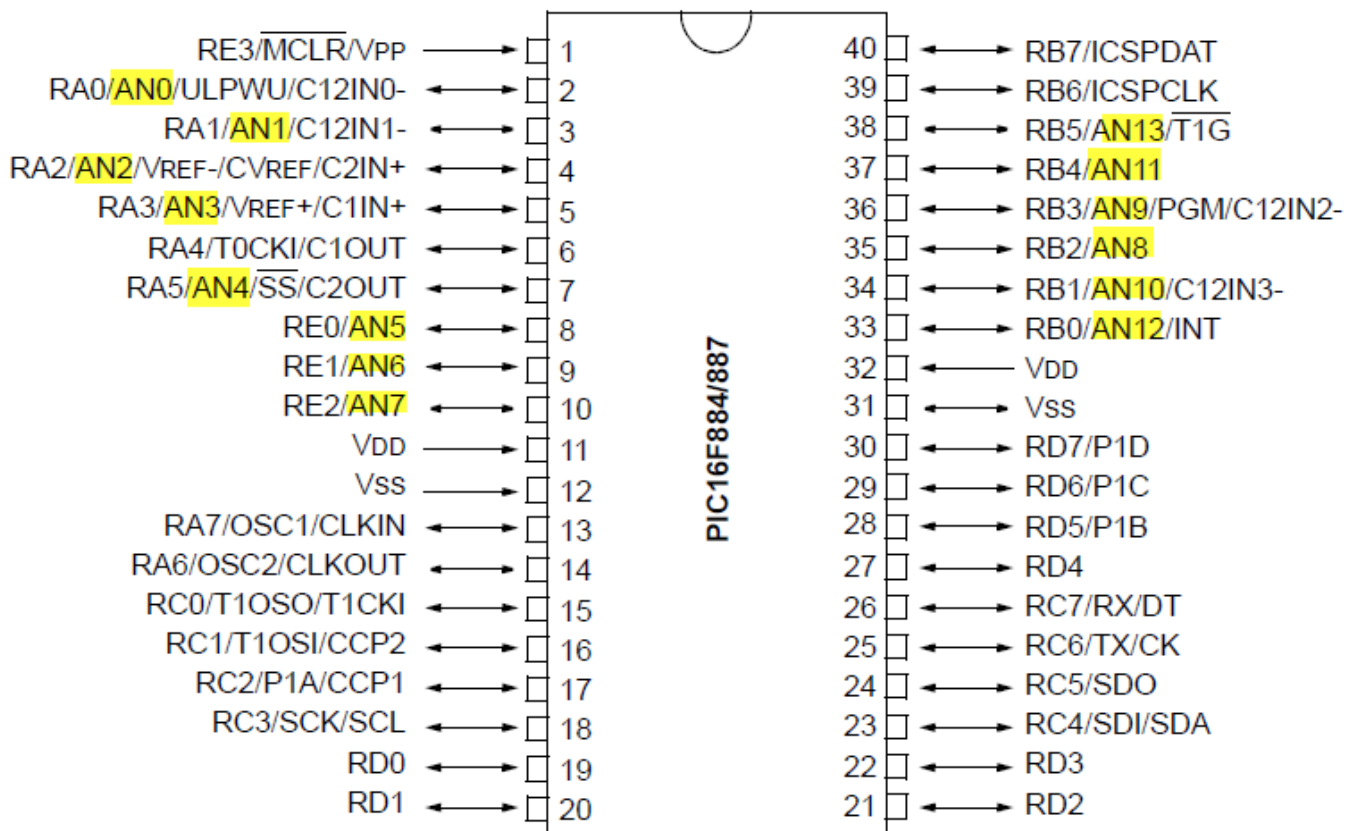
- Virtual Serial Port Driver là phần mềm hữu hiệu để tạo ra các cổng nối tiếp ảo và kết nối chúng theo cặp thông qua dây cáp null-modem ảo. Các ứng dụng trên cả hai đầu của cặp đó sẽ có thể trao đổi dữ liệu cho nhau. Khi đó, dữ liệu được ghi trên cổng đầu tiên sẽ xuất hiện ở cổng thứ hai và ngược lại.
- Tất cả các cổng nối tiếp ảo đều hoạt động chính xác như những cổng thực, mô phỏng các thiết lập của chúng. Do đó, có thể tạo ra bao nhiêu cặp cổng ảo theo ý muốn mà không cần phải sử dụng phần cứng bổ sung nào.
- Trong bài báo cáo này ta tạo ra COM1 và COM2 ảo để thực hiện truyền dữ liệu nối tiếp.



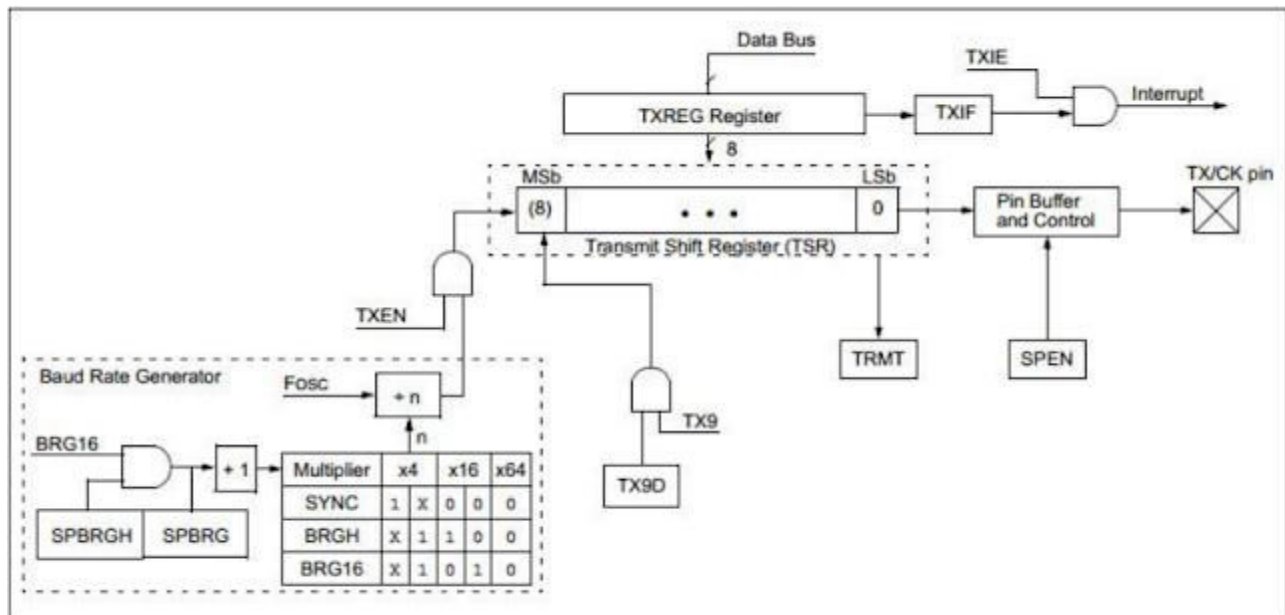
A.2. Vi điều khiển và chương trình hỗ trợ

- Trong đề tài này, nhóm sẽ sử dụng vi điều khiển PIC16F887 để thực hiện truyền nhận dữ liệu nối tiếp và chốt dữ liệu qua IC74HC595.

1. Sơ đồ chân của PIC 16F887 như sau:



- Sơ đồ khối bộ truyền UART:



Nguyên tắc hoạt động:

- Dữ liệu cần truyền được đặt vào thanh ghi TXREG, baud rate được tạo ra, khi TXEN gán bằng 1 dữ liệu từ thanh ghi TXREG đưa vào thanh ghi TSR đồng thời baud rate tác động đến TSR, đẩy dữ liệu cần truyền ra bộ đệm sau đó xuất ra chân TX.
- Bit TXIF dùng để báo trạng thái trong thanh ghi TXREG, nếu có dữ liệu trong TXREG thì TXIF =1. Nếu dữ liệu được truyền xuống thanh TSR thì TXIF = 0. Tương tự bit TRMT dùng để báo trạng thái thanh ghi TSR.

- Cho phép ngắt toàn cục: CIE = 1; PEIE = 1;
- Xử lý các phần khác của chương trình khi có ngắt xảy ra thì xử lý dữ liệu.

Thanh ghi quy định chế độ nhận:

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-x
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
bit 7							bit 0

Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

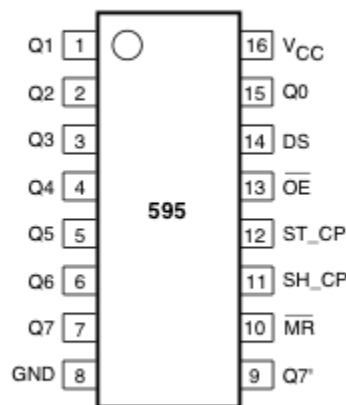
'0' = Bit is cleared

x = Bit is unknown

- SPEN: Khởi tạo cổng nối tiếp
- SPEN = 1; // Cho phép cổng nối tiếp- SPEN = 0; // Không cho phép- RX9: Cho phép nhận 9bit
- RX9 = 1; // Cho phép nhận 9bit- RX9 = 0; // Nhận 8bit- CREN: Cho phép nhận liên tục
- CREN = 1; // Cho phép- CREN = 0; // Không cho phép- ADDEN: Bit cho phép phát hiện địa chỉ (sử dụng ở chế độ truyền nhận bất đồng bộ 9 bit)
- ADDEN = 1; // Cho phép phát hiện địa chỉ , cho phép ngắt và tải bộ đệm nhận khi RSR<8> được set.- ADDEN = 0; // Không cho phép phát hiện địa chỉ , tất cả byte được nhận và bit thứ 9 dùng làm bit parity.- FERR: Bit báo lỗi frame
- FERR == 1; // Có lỗi.- FERR == 0; // Không có lỗi.- OERR: Lỗi OVERRUN
- OEER == 1 ; // Có lỗi- OEER == 0; // Không lỗi- RX9D: Lưu dữ liệu nhận của bit thứ 9
- Thanh ghi TXREG: Dùng để chứa dữ liệu truyền đi.
- Thanh ghi RCREG: Dùng để lưu dữ liệu từ ngoài vào.
- Thanh ghi SPBRG: Thiết lập baud rate của PIC

2. Đôi nét về IC74HC595

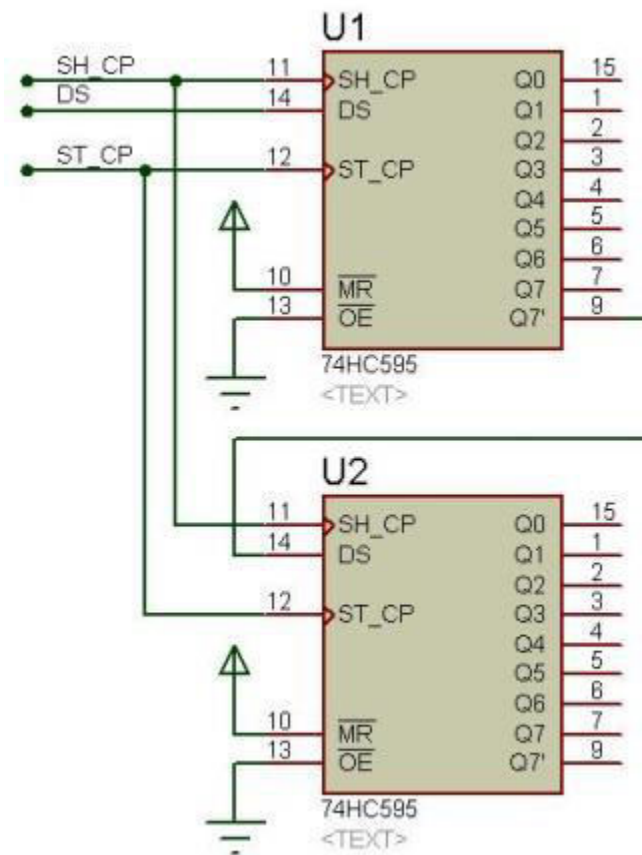
- Sơ đồ chân:



- Giải thích ý nghĩa hoạt động của một số chân quan trọng:

PINS 1-7, 15	Q0 đến Q7	các chân xuất tín hiệu, giống như các chân Digital được cài đặt là OUTPUT
PIN 8	GND	Ground, Cực âm
PIN 9	Q7"	Chân xuất ra tín hiệu Serial
PIN 10	MR	Master Reclear, nối cực dương để bật IC hoạt động
PIN 11	SH_CP	Shift register clock pin
PIN 12	ST_CP	Storage register clock pin (latch pin)
PIN 13	OE	Output enable, nối cực âm để các đèn LED có thể sáng được.
PIN 14	DS	Serial data input
PIN 16	Vcc	Cấp nguồn cho IC và LED.

- Nối nhiều IC74HC595



- Nối song song các chân SHCP và STCP. Nối tiếp chân Q7' của IC trước tới cổng DS của IC sau.

3. Phần mềm hỗ trợ

- CCS là trình biên dịch lập trình ngôn ngữ C cho Vi điều khiển PIC của hãng Microchip.
- Chương trình là sự tích hợp của 3 trình biên dịch riêng biệt cho 3 dòng PIC khác nhau đó là:
 - PCB cho dòng PIC 12-bit opcodes
 - PCM cho dòng PIC 14-bit opcodes

- PCH cho dòng PIC 16 và 18-bit

- Tất cả 3 trình biên dịch này được tích hợp lại vào trong một chương trình bao gồm cả trình soạn thảo và biên dịch là CCS, phiên bản mới nhất là PCWH Compiler Ver 3.227.

- Giống như nhiều trình biên dịch C khác cho PIC, CCS giúp cho người sử dụng nắm bắt nhanh được vi điều khiển PIC và sử dụng PIC trong các dự án. Các chương trình điều khiển sẽ được thực hiện nhanh chóng và đạt hiệu quả cao thông qua việc sử dụng ngôn ngữ lập trình cấp cao – Ngôn ngữ C. Tài liệu hướng dẫn sử dụng có rất nhiều, nhưng chi tiết nhất chính là bản Help đi kèm theo phần mềm (tài liệu Tiếng Anh). Trong bản trợ giúp nhà sản xuất đã mô tả rất nhiều về hằng, biến, chỉ thị tiền xử lý, cấu trúc các câu lệnh trong chương trình, các hàm tạo sẵn cho người sử dụng...

B. Chương trình giao tiếp vi điều khiển và C# (PC)

B.1. Form C# giao tiếp vi điều khiển

The screenshot shows a Windows application window titled "Form1" with standard minimize, maximize, and close buttons. The window is divided into three main sections:

- Settings:** Contains five dropdown menus for configuring serial communication:
 - Ports: *COM1*
 - Baud Rate: *9600*
 - Data Bits: *8*
 - Parity: *None*
 - Stop Bit: *1*
- Time Adjustments:** Contains four steps for timing adjustments, each with a text input field and a label:
 - Step 1: *45* Red_A - Green_B ON
 - Step 2: *05* Red_A - Yellow_B ON
 - Step 3: *45* Green_A - Red_B ON
 - Step 4: *05* Yellow_A - Red_B ON
 Below these steps is a large button labeled **Connect**.
- Monitoring:** Features a 2x3 grid of gray rectangular boxes. To the left of the grid are large, bold letters **A** and **B**, indicating the rows of the monitoring display.

At the bottom of the window, there is a status bar labeled "Connection Status" with a small icon on the right.

- Code lập trình giao diện trên

```
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 Serial Ports
using System.IO;
using System.IO.Ports;
using System.Xml;
// Using Thread
using System.Threading;

namespace TrafficLight
{
    public partial class Form1 : Form
    {
        //SerialPort Port = new SerialPort(); // SerialPort Selected
        string InputData = String.Empty;
        public Form1()
        {
            InitializeComponent();
            // Serial Port Connection Configurations
            tmrPort1.Interval = 1000;
            tmrPort1.Enabled = true;
            Port1.ReadTimeout = 1000;
            //Port1.DataReceived += new SerialDataReceivedEventHandler(DataReceive);
            // BaudRate Settings
            string[] BaudRate = { "1200", "2400", "4800", "9600", "19200", "38400", "57600", "115200" };
            cbBaudRate.Items.AddRange(BaudRate);

            // DataBits Settings
            string[] Databits = { "6", "7", "8" };
            cbDataBits.Items.AddRange(Databits);

            // Parity Settings
            string[] Parity = { "None", "Odd", "Even" };
            cbParity.Items.AddRange(Parity);

            // Stop bit Settings
            string[] stopbit = { "1", "1.5", "2" };
            cbStopBit.Items.AddRange(stopbit);
        }

        string[] ports = SerialPort.GetPortNames();
        int intlen = 0;

        private void tmrPort1_Tick(object sender, EventArgs e)
        {
            if (intlen != ports.Length)
            {
                intlen = ports.Length;
                cbb_COMPort.Items.Clear();
                for (int i = 0; i < intlen; i++)
                {
                    cbb_COMPort.Items.Add(ports[i]);
                }
            }
        }
    }
}
```

```

private void Form1_Load(object sender, EventArgs e)
{
    // Connection Default
    cbBaudRate.SelectedIndex = 3;
    cbDataBits.SelectedIndex = 2;
    cbParity.SelectedIndex = 0;
    cbStopBit.SelectedIndex = 0;
    Port1.Close();
    Control.CheckForIllegalCrossThreadCalls = false;
}

private void cbb_COMPort_SelectedIndexChanged(object sender, EventArgs e)
{
    if (Port1.IsOpen)
    {
        Port1.Close();
    }
    Port1.PortName = cbb_COMPort.Text;
}

private void cbBaudRate_SelectedIndexChanged(object sender, EventArgs e)
{
    if (Port1.IsOpen)
    {
        Port1.Close();
    }
    Port1.BaudRate = Convert.ToInt32(cbBaudRate.Text);
}

private void cbDataBits_SelectedIndexChanged(object sender, EventArgs e)
{
    if (Port1.IsOpen)
    {
        Port1.Close();
    }
    Port1.DataBits = Convert.ToInt16(cbDataBits.Text);
}

private void cbParity_SelectedIndexChanged(object sender, EventArgs e)
{
    if (Port1.IsOpen)
    {
        Port1.Close();
    }
    switch (cbParity.SelectedIndex.ToString())
    {
        case "Odd":
            Port1.Parity = Parity.Odd;
            break;
        case "Even":
            Port1.Parity = Parity.Even;
            break;
        default:
            Port1.Parity = Parity.None;
            break;
    }
}

private void cbStopBit_SelectedIndexChanged(object sender, EventArgs e)
{
    if (Port1.IsOpen)
    {

```



```

        Port1.Close();
    }
    switch (cbStopBit.SelectedIndex.ToString())
    {
        case "1":
            Port1.StopBits = StopBits.One;
            break;
        case "1.5":
            Port1.StopBits = StopBits.OnePointFive;
            break;
        default:
            Port1.StopBits = StopBits.Two;
            break;
    }
}

private void btnConnect_Click(object sender, EventArgs e)
{
    string data = tbStep1.Text.Trim() + tbStep2.Text.Trim() + tbStep3.Text.Trim() +
tbStep4.Text.Trim() + "~";
    if (btnConnect.Text == "Connect")
    {
        btnConnect.Text = "Disconnect";
        try
        {
            Port1.Open();
            // statusStrip Display
            statusLabel.Text = "Connecting to " + cbb_COMPort.SelectedItem.ToString();
            Port1.Write(data);
        }
        catch (Exception ex)
        {
            statusLabel.Text = "Connection FAILED";
        }
    }
    else
    {
        btnConnect.Text = "Connect";
        Port1.Close();
        // statusStrip Display
        statusLabel.Text = "Disconnection Complete";
    }
}

private void Port1_DataReceived(object sender, SerialDataReceivedEventArgs e)
{
    try
    {
        InputData = Port1.ReadLine();
        Port1.Close(); // Reset UART
        LightDisplay();
        Port1.Open(); // Restart UART
    }
    catch (System.TimeoutException) { }
}

private void LightDisplay()
{
    if (InputData == "RedA_GreenB")
    {
        RedA.BackColor = Color.Red;
        GreenA.BackColor = Color.Gray;
        YellowA.BackColor = Color.Gray;
        RedB.BackColor = Color.Gray;
    }
}

```

```
        GreenB.BackColor = Color.Green;
        YellowB.BackColor = Color.Gray;
    }
    if (InputData == "RedA_YellowB")
    {
        RedA.BackColor = Color.Red;
        GreenA.BackColor = Color.Gray;
        YellowA.BackColor = Color.Gray;
        RedB.BackColor = Color.Gray;
        GreenB.BackColor = Color.Gray;
        YellowB.BackColor = Color.Yellow;
    }
```

```
    if (InputData == "GreenA_RedB")
    {
        RedA.BackColor = Color.Gray;
        GreenA.BackColor = Color.Green;
        YellowA.BackColor = Color.Gray;
        RedB.BackColor = Color.Red;
        GreenB.BackColor = Color.Gray;
        YellowB.BackColor = Color.Gray;
    }
```

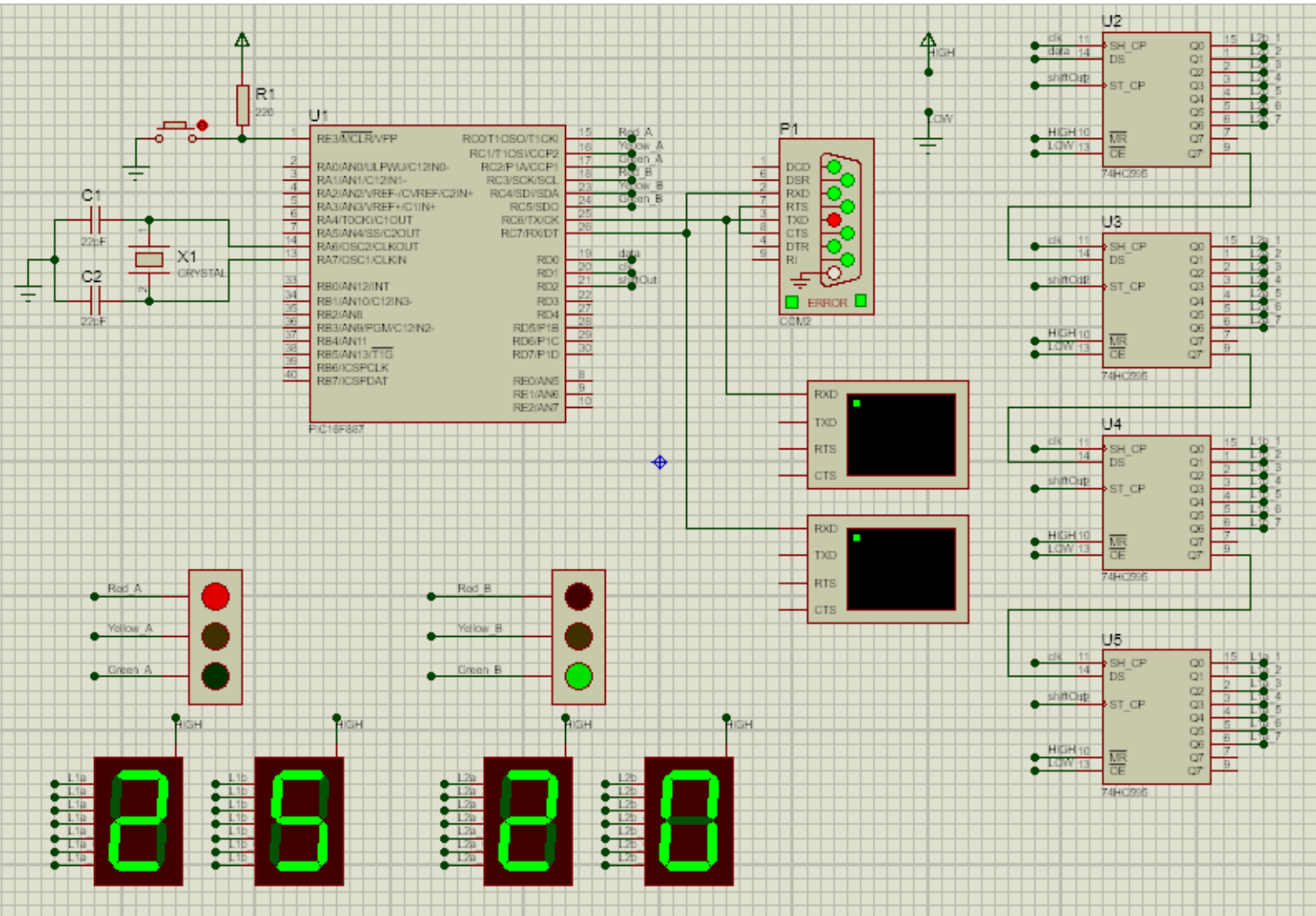
```
    if (InputData == "YellowA_RedB")
    {
        RedA.BackColor = Color.Gray;
        GreenA.BackColor = Color.Gray;
        YellowA.BackColor = Color.Yellow;
        RedB.BackColor = Color.Red;
        GreenB.BackColor = Color.Gray;
        YellowB.BackColor = Color.Gray;
    }
```

```
}
```


```
}
```

```
}
```

B.2. *Vẽ mạch mô phỏng trên Proteus*



- Ta thiết lập cổng COMPIM như sau:

 Edit Component

Part Reference:	<input type="text" value="P1"/>	Hidden:	<input type="checkbox"/>
Part Value:	<input type="text" value="COM2"/>	Hidden:	<input type="checkbox"/>
Element:	<div><div></div></div> <div>New</div>		
VSM Model:	<input type="text" value="COMPIM.DLL"/>	Hide All	<div></div>
Physical port:	<input type="text" value="COM2"/>	Hide All	<div></div>
Physical Baud Rate:	<input type="text" value="9600"/>	Hide All	<div></div>
Physical Data Bits:	<input type="text" value="8"/>	Hide All	<div></div>
Physical Parity:	<input type="text" value="NONE"/>	Hide All	<div></div>
Virtual Baud Rate:	<input type="text" value="9600"/>	Hide All	<div></div>
Virtual Data Bits:	<input type="text" value="8"/>	Hide All	<div></div>
Virtual Parity:	<input type="text" value="NONE"/>	Hide All	<div></div>
Advanced Properties:			
Physical Stop Bits	<div><div></div><div>1</div><div></div></div>	Hide All	<div></div>

- 2 khối VIRTUAL TERMINAL có chức năng đọc dữ liệu truyền nhận vào vi điều khiển.

B.3. Soạn thảo code trên phần mềm hỗ trợ CCS Compiler

- Trước tiên ta khai báo thư viện và vi điều khiển đang sử dụng. Đồng thời khai báo chức năng và gán tên cho các chân vi điều khiển PIC16F887.

```
#include <16f887.h>
#include <stdio.h>
#fuses NOMCLR, INTRC_IO, NOBROWNOUT, NOLVP
#use delay(clock=8M)
#use rs232(UART1, baud=9600, xmit=PIN_C6, rcv=PIN_C7)
#define data    PIN_D0 // D0 xuất đưa dữ liệu vào ic595
#define clk     PIN_D1 // Xung Clk có cạnh lên ở ic595 sẽ đẩy bit vào
#define shiftOut PIN_D2 // D2 có xung clock sẽ đẩy dữ liệu ra các cổng output của ic595 từng bit một
#define Red_A   PIN_C0
#define Yellow_A PIN_C1
#define Green_A  PIN_C2
#define Red_B   PIN_C3
#define Yellow_B PIN_C4
#define Green_B  PIN_C5
```

- Hàm phục vụ ngắt khi có dữ liệu trên đường truyền nối tiếp và khai báo Array lưu trữ dữ liệu. Trong chuỗi dữ liệu truyền, ở việc truyền từ PC (Form C#) sẽ thêm ký tự '~' vào cuối chuỗi để thông báo kết thúc chuỗi dữ liệu truyền và bật cờ ngắt lên 1 để chương trình thực hiện xuất dữ liệu lên LED 7 đoạn.

```
char buffer[8], c;
int index = 0;
int1 flag_data=0;//cơ chế kiểm tra xem đã nhận được ký tự kết thúc chuỗi nhận được
unsigned int8 i;

#int_rda
void _Interrupt_RDA(void)
{
    c=getchar();//gán tạm giá trị nhận được vào biến c
    if(c=='~')
    {
        flag_data=1;//kết thúc quá trình truyền nhận dữ liệu với máy tính
        //index=0;//reset hệ số mảng về 0
    }
    else
    {
        buffer[index]=c;//nhập dữ liệu nhận được vào các phần tử mảng
        index++;//tăng hệ số mảng lên 1
    }
}
```

- Hàm main thực hiện việc tính toán để biến đổi dữ liệu từ kiểu char sang int8 và xuất ra LED 7 đoạn (chốt bit). Khi bước vào hàm main, luồng xử lý sẽ không được thực hiện (flag_data chưa lên mức 1) đến khi chuỗi dữ liệu trên đường truyền đã được truyền xong, buffer[8] sẽ lưu trữ dữ liệu đó.

- Cứ mỗi giây đếm lùi, vi điều khiển sẽ gửi tình trạng đèn sáng hiện tại về PC (Form C#) thông qua các chuỗi dữ liệu "RedA_GreenB", "RedA_YellowB", "GreenA_RedB", "YellowA_RedB" và chương trình C# sẽ xử lý thông tin để đổi màu textBox đối với mỗi trường hợp chuỗi dữ liệu khác nhau.

```
const unsigned char led7seg[10] = {0XC0,0XF9,0XA4,0XB0,0X99,0X92,0X82,0XF8,0X80,0X90};

unsigned int8 L1a, L1b, L2a, L2b;

// Day du lieu ra cong DS cua ic595
void Out_byte (unsigned int8 Out)
{
    unsigned int8 i = 0;
    #bit bitData = Out.7; // Day du lieu tu bit cao ra ic595
    for (i = 0; i < 8; i++)
    {
        output_bit(data, bitData);
        output_low(clk); output_high(clk); // Tao canh len xung clock
        Out = Out << 1;
    }
}

void OutToLed()
{
    Out_byte(L1a);
    Out_byte(L1b);
    Out_byte(L2a);
    Out_byte(L2b);
    output_low(shiftOut); output_high(shiftOut); // Day cac bit ngo ra cua ic595 ra cac LED (xung len cua ST_CP)
}

// Giai ma LED 7 doan
void LED_decoder1(unsigned int x)
{
    L1a = led7seg[x/10]; // Hang chuc. VD: 26s thi L1a = 2 ung voi ma trong chuoi led7seg
    L1b = led7seg[x%10]; // Hang don vi. VD: 26/10 du 6 thi 6 ung voi ma trong chuoi led7seg
}

void LED_decoder2(unsigned int x)
{
    L2a = led7seg[x/10]; // Hang chuc. VD: 26s thi L1a = 2 ung voi ma trong chuoi led7seg
    L2b = led7seg[x%10]; // Hang don vi. VD: 26/10 du 6 thi 6 ung voi ma trong chuoi led7seg
}

void main()
{
    beginLoop:
    setup_oscillator(OSC_8MHZ);          // Set internal oscillator to 8MHz
    unsigned int i = 0;
    //set_tris_C(0); // PortC: Output
```

```

//set_tris_D(0); // PortD: Output
int8 step1, step2, step3, step4;
ENABLE_INTERRUPTS(INT_RDA);//cho phep ngat uart(ngat truyen nhan du lieu)
ENABLE_INTERRUPTS(GLOBAL);//cho phep ngat toan cuc
while(TRUE)
{
    if (flag_data == 1)
    {
        // Tinh toan thoi gian cac step
        step1 = (buffer[0]-48)*10 + (buffer[1]-48);
        step2 = (buffer[2]-48)*10 + (buffer[3]-48);
        step3 = (buffer[4]-48)*10 + (buffer[5]-48);
        step4 = (buffer[6]-48)*10 + (buffer[7]-48);
        // Step1: Red_A & Green_B ON
        for (i = 0; i < step1 ; i++)
        {
            printf("RedA_GreenB\n\r");
            output_high(Red_A); output_low(Yellow_A); output_low(Green_A);
            output_high(Green_B); output_low(Yellow_B); output_low(Red_B);
            LED_decoder1(step1 +step2 - i); LED_decoder2(step1 - i);
            OutToLed();
            delay_ms(1000);
        }

        // Step2: Red_A & Yellow_B ON
        for (i = 0; i < step2; i++)
        {
            printf("RedA_YellowB\n\r");
            output_high(Red_A); output_low(Yellow_A); output_low(Green_A);
            output_high(Yellow_B); output_low(Green_B); output_low(Red_B);
            LED_decoder1(step2 - i); LED_decoder2(step2 - i);
            OutToLed();
            delay_ms(1000);
        }

        // Step3: Green_A & Red_B ON
        for (i = 0; i < step3; i++)
        {
            printf("GreenA_RedB\n\r");
            output_high(Green_A); output_low(Yellow_A); output_low(Red_A);
            output_high(Red_B); output_low(Yellow_B); output_low(Green_B);
            LED_decoder1(step3 + step4 - i); LED_decoder2(step3 - i);
            OutToLed();
            delay_ms(1000);
        }

        // Step4: Yellow_A & Red_B ON
        for (i = 0; i < step4; i++)
        {

```

```

printf("YellowA_RedB\n\r");
output_high(Yellow_A); output_low(Red_A); output_low(Green_A);
output_high(Red_B); output_low(Yellow_B); output_low(Green_B);
LED_decoder1(step4 - i); LED_decoder2(step4 - i);
OutToLed();
delay_ms(1000);
}
}
else goto beginLoop;
}

```

C. Kết quả thực hiện

- Vi điều khiển nhận chuỗi ký tự thời gian từ C# và xử lý dữ liệu thành công để xuất ra LED 7 đoạn
- PC nhận chuỗi dữ liệu trạng thái đèn giao thông thành công và biểu diễn sự thay đổi màu sắc của textBox như trên mô phỏng Proteus.

