CHƯƠNG 3

XÂY DỰNG GIAO DIỆN TRỰC QUAN VÀ ỨNG DỤNG TRONG ĐIỀU KHIỂN

GVHD: TS. GVC LÊ THỊ THÚY NGA

BỘ MÔN: ĐIỀU KHIỂN HỌC

NỘI DUNG

3.1 Thiết kế giao diện

3.2 Các điều khiển cơ bản

3.3 Các điều khiển phân nhóm

3.4 Giao tiếp ngoại vi

3.5 Một số ứng dụng giao diện trực quan trong kỹ thuật điều khiển

3.4 GIAO TIẾP THIẾT BỊ NGOẠI VI VÀ ỨNG DỤNG TRONG ĐIỀU KHIỂN

- Các thư viện sử dụng kết nối vào/ra thường được sử dụng trong Windows Form:

```
using System.IO;//Thư viện xuất nhập
using System.IO.Ports;// Thư viện dùng cho cổng COM
using System.Xml;
using ZedGraph;//Thư viện dụng cho vẽ đồ thị.
```

- Thuộc tính thông dụng của **SerialPort**:

BaseStream	Gets the underlying Stream object for a SerialPort object.
BaudRate	Gets or sets the serial baud rate.
BreakState	Gets or sets the break signal state.
BytesToRead	Gets the number of bytes of data in the receive buffer.
BytesToWrite	Gets the number of bytes of data in the send buffer.
CanRaiseEvents	Gets a value indicating whether the component can raise an event. (Inherited from Component)
CDHolding	Gets the state of the Carrier Detect line for the port.
Container	Gets the IContainer that contains the Component. (Inherited from Component)
CtsHolding	Gets the state of the Clear-to-Send line.
DataBits	Gets or sets the standard length of data bits per byte.

DesignMode	Gets a value that indicates whether the Component is currently in design mode. (Inherited from Component)
DiscardNull	Gets or sets a value indicating whether null bytes are ignored when transmitted between the port and the receive buffer.
DsrHolding	Gets the state of the Data Set Ready (DSR) signal.
DtrEnable	Gets or sets a value that enables the Data Terminal Ready (DTR) signal during serial communication.
Encoding	Gets or sets the byte encoding for pre- and post-transmission conversion of text.
Events	Gets the list of event handlers that are attached to this Component. (Inherited from Component)
Handshake	Gets or sets the handshaking protocol for serial port transmission of data using a value from Handshake.
IsOpen	Gets a value indicating the open or closed status of the SerialPort object.
NewLine	Gets or sets the value used to interpret the end of a call to the ReadLine() and WriteLine(String) methods.
Parity	Gets or sets the parity-checking protocol.

- Thuộc tính thông dụng của **SerialPort**:

ParityReplace	Gets or sets the byte that replaces invalid bytes in a data stream when a parity error occurs.
PortName	Gets or sets the port for communications, including but not limited to all available COM ports.
ReadBufferSize	Gets or sets the size of the SerialPort input buffer.
ReadTimeout	Gets or sets the number of milliseconds before a time-out occurs when a read operation does not finish.
ReceivedBytesThreshold	Gets or sets the number of bytes in the internal input buffer before a DataReceived event occurs.
RtsEnable	Gets or sets a value indicating whether the Request to Send (RTS) signal is enabled during serial communication.
Site	Gets or sets the ISite of the Component. (Inherited from Component)
StopBits	Gets or sets the standard number of stopbits per byte.
WriteBufferSize	Gets or sets the size of the serial port output buffer.
WriteTimeout	Gets or sets the number of milliseconds before a time-out occurs when a write operation does not finish.

- Phương thức thông dụng của **SerialPort**:

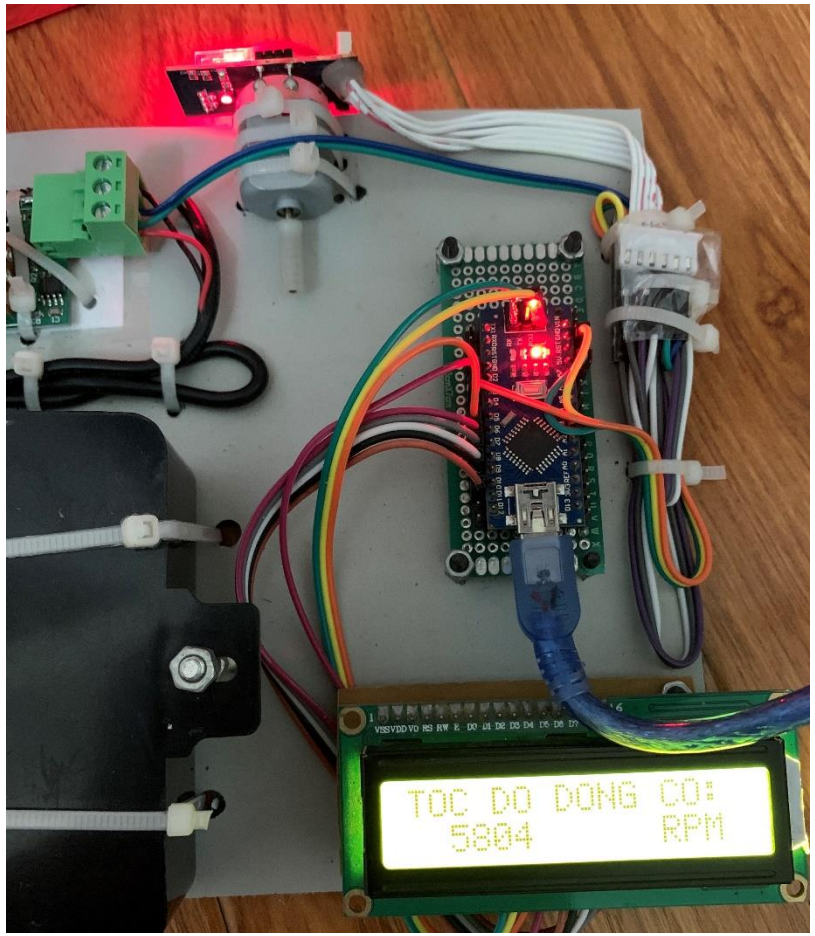
Close()	Closes the port connection, sets the IsOpen property to false, and disposes of the internal Stream object.
CreateObjRef(Type)	Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object. (Inherited from MarshalByRefObject)
DiscardInBuffer()	Discards data from the serial driver's receive buffer.
DiscardOutBuffer()	Discards data from the serial driver's transmit buffer.
Dispose()	Releases all resources used by the Component. (Inherited from Component)
Dispose(Boolean)	Releases the unmanaged resources used by the SerialPort and optionally releases the managed resources.
Equals(Object)	Determines whether the specified object is equal to the current object. (Inherited from Object)
GetHashCode()	Serves as the default hash function. (Inherited from Object)

GetLifetimeService()	Retrieves the current lifetime service object that controls the lifetime policy for this instance. (Inherited from MarshalByRefObject)
GetPortNames()	Gets an array of serial port names for the current computer.
GetService(Type)	Returns an object that represents a service provided by the Component or by its Container. (Inherited from Component)
GetType()	Gets the Type of the current instance. (Inherited from Object)
InitializeLifetimeService()	Obtains a lifetime service object to control the lifetime policy for this instance. (Inherited from MarshalByRefObject)
MemberwiseClone()	Creates a shallow copy of the current Object. (Inherited from Object)
MemberwiseClone(Boolean)	Creates a shallow copy of the current MarshalByRefObject object. (Inherited from MarshalByRefObject)
Open()	Opens a new serial port connection.
Read(Byte[], Int32, Int32)	Reads a number of bytes from the SerialPort input buffer and writes those bytes into a byte array at the specified offset.
Read(Char[], Int32, Int32)	Reads a number of characters from the SerialPort input buffer and writes them into an array of characters at a given offset.

ReadByte()	Synchronously reads one byte from the SerialPort input buffer.
ReadChar()	Synchronously reads one character from the SerialPort input buffer.
ReadExisting()	Reads all immediately available bytes, based on the encoding, in both the stream and the input buffer of the SerialPort object.
ReadLine()	Reads up to the NewLine value in the input buffer.
ReadTo(String)	Reads a string up to the specified value in the input buffer.
ToString()	Returns a String containing the name of the Component, if any. This method should not be overridden. (Inherited from Component)
Write(Byte[], Int32, Int32)	Writes a specified number of bytes to the serial port using data from a buffer.
Write(Char[], Int32, Int32)	Writes a specified number of characters to the serial port using data from a buffer.
Write(String)	Writes the specified string to the serial port.
WriteLine(String)	Writes the specified string and the NewLine value to the output buffer.

3.4 GIAO TIẾP THIẾT BỊ NGOẠI VI VÀ ỨNG DỤNG TRONG ĐIỀU KHIỂN

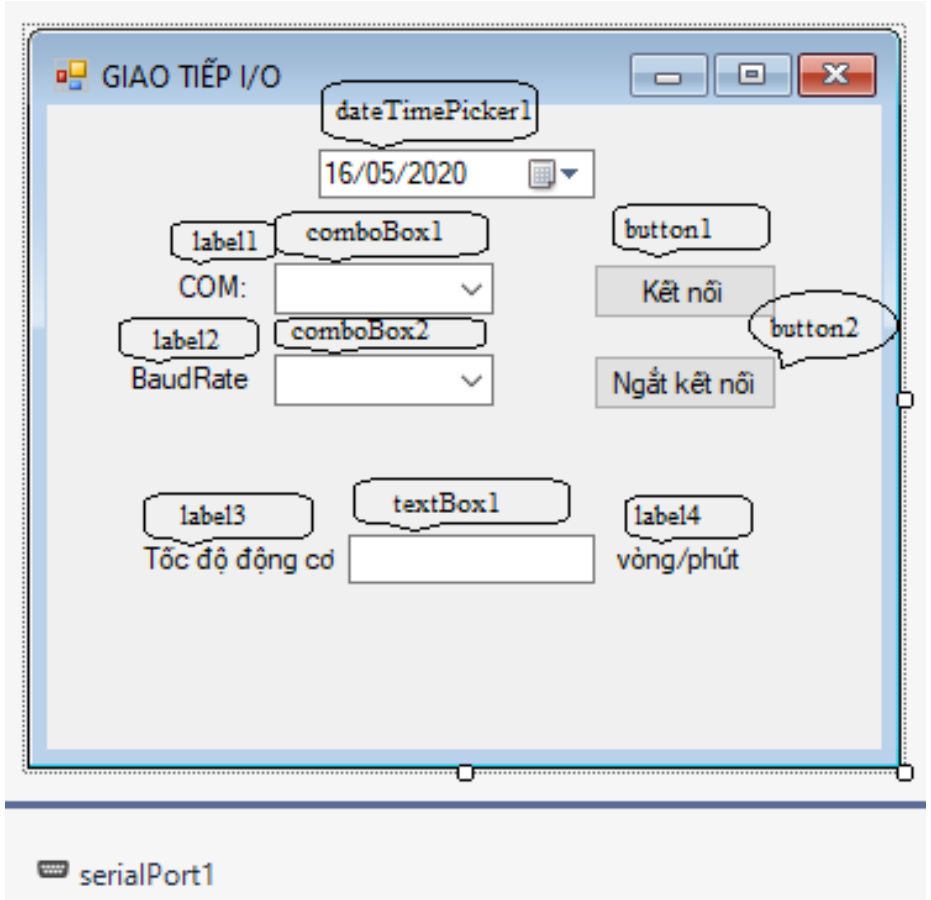
❖ **Ví dụ áp dụng 1**: Ứng dụng Windows Form xây dựng giao diện có chức năng giao tiếp cổng COM và nhận dữ liệu là tốc độ quay của động cơ điện DC.



- Giao diện có 2 chức năng sau:
  - *Nhận được các cổng COM của PC.*
  - *Nhận được dữ liệu tốc độ từ thiết bị ngoại vi đẩy lên thông qua kết nối IO.*

3.4 GIAO TIẾP THIẾT BỊ NGOẠI VI VÀ ỨNG DỤNG TRONG ĐIỀU KHIỂN

❖Giao diện có được thiết kế như hình bên:



3.4 GIAO TIẾP THIẾT BỊ NGOẠI VI VÀ ỨNG DỤNG TRONG ĐIỀU KHIỂN

- Chương trình thực hiện:

```
public Form1()
    {
        InitializeComponent();
        string[] ports = SerialPort.GetPortNames();// Khai báo và khởi tạo một đối tượng có tên ports thuộc SerialPort.
        comboBox1.Items.AddRange(ports); //Thêm các ports mà máy tính nhận được.
        string[] baudrate = { "9600" };//Khai báo và khởi tạo giá trị biến có tên baudrate thuộc kiểu mảng để biểu diễn tốc độ truyền dữ liệu.
        comboBox2.Items.AddRange(baudrate);// Thêm các baudrate mà máy tính nhận được.
    }
```

3.4 GIAO TIẾP THIẾT BỊ NGOẠI VI VÀ ỨNG DỤNG TRONG ĐIỀU KHIỂN

- Chương trình thực hiện:

```
private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
    {
       if (serialPort1.IsOpen)   //Kiểm tra điều kiện xem cổng COM có mở không?
       {
       }
       comboBox2.SelectedIndex = 0;
    }
```

3.4 GIAO TIẾP THIẾT BỊ NGOẠI VI VÀ ỨNG DỤNG TRONG ĐIỀU KHIỂN

- Chương trình thực hiện:

```
private void serial(object sender, SerialDataReceivedEventArgs e)
    {
        string tocdo = serialPort1.ReadLine().ToString();
        try
        {
            BeginInvoke(new Action(() =>
                {
                    textBox1.Text = tocdo.ToString();
                }));
        }
        catch (Exception)
        {
        }
    }
```

3.4 GIAO TIẾP THIẾT BỊ NGOẠI VI VÀ ỨNG DỤNG TRONG ĐIỀU KHIỂN

- Chương trình thực hiện:

```
private void button1_Click(object sender, EventArgs e)
        {
            if (!serialPort1.IsOpen)
            {
                serialPort1.PortName = comboBox1.Text;
                try
                {
                    serialPort1.Open();
                    button1.Enabled = false;
                    comboBox1.Enabled = false;
                    comboBox2.Enabled = false;
                    button1.ForeColor = Color.Green;
                    button2.Enabled = true;
                    button2.ForeColor = Color.Green;
                }
                catch
                {
                    MessageBox.Show("Không kết nối được",
"Thử lại", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                }
            }
```

3.4 GIAO TIẾP THIẾT BỊ NGOẠI VI VÀ ỨNG DỤNG TRONG ĐIỀU KHIỂN

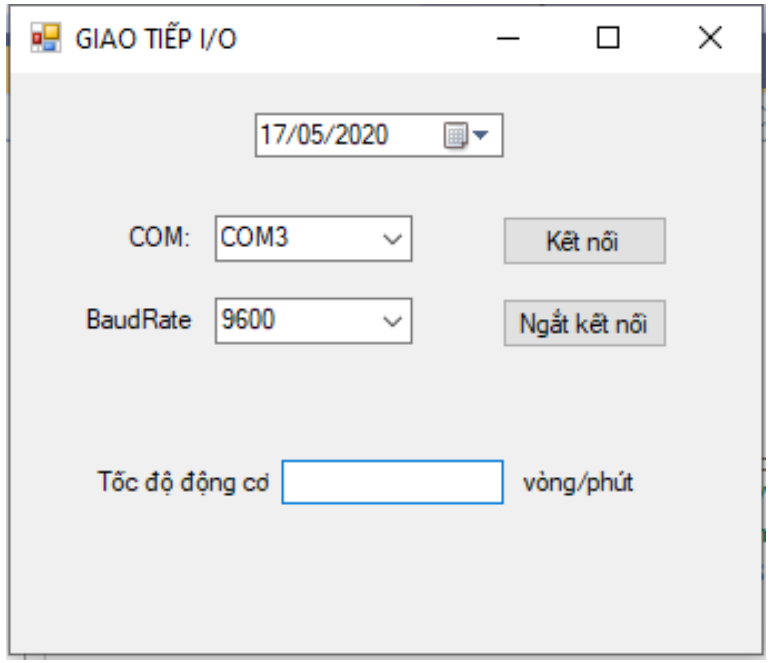
- Chương trình thực hiện:

```
private void button2_Click(object sender, EventArgs e)
        {
            serialPort1.Close();
            button1.Enabled = true;
            button2.Enabled = false;
            textBox1.Text = "";
}
```

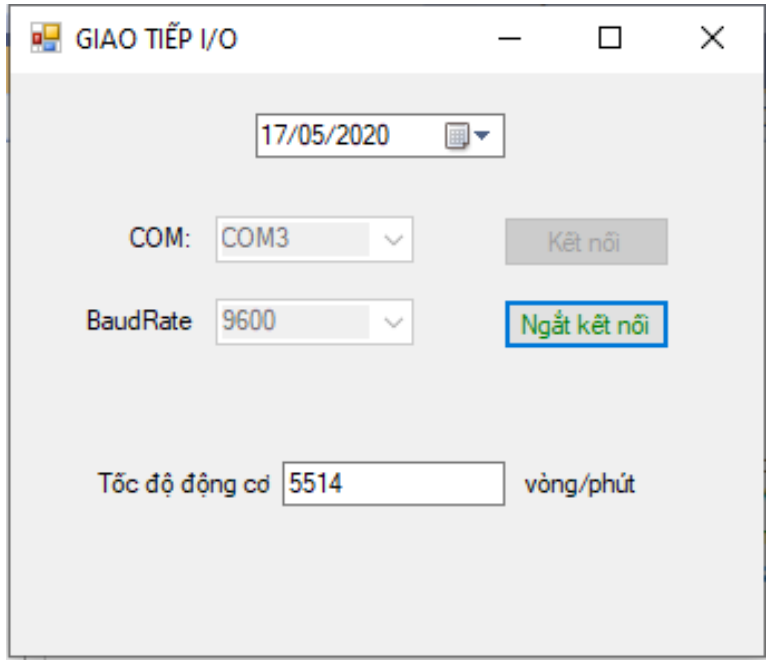
```
private void Form1_Load(object sender, EventArgs e)
        {
            comboBox1.DataSource = SerialPort.GetPortNames();
        }
```

3.4 GIAO TIẾP THIẾT BỊ NGOẠI VI VÀ ỨNG DỤNG TRONG ĐIỀU KHIỂN

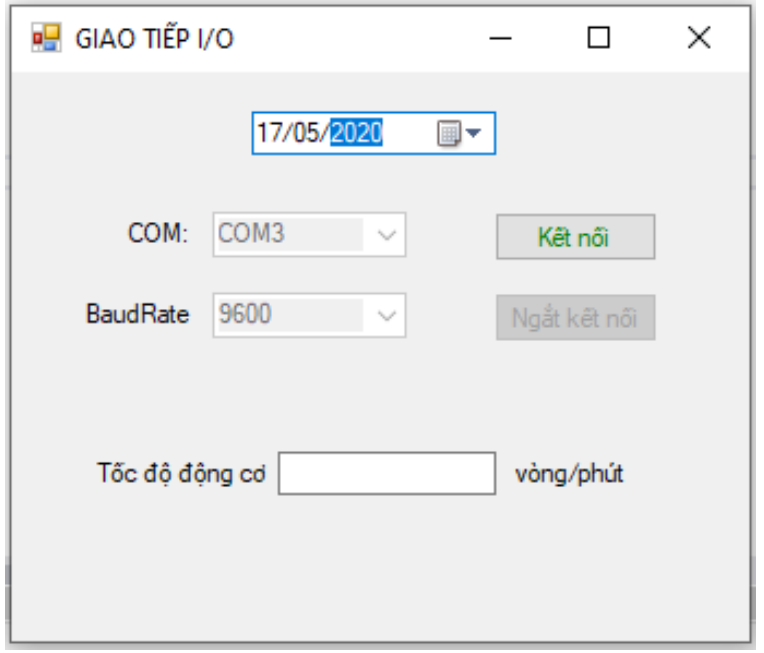
- Kết quả:



Giao diện khởi động



Giao diện Kết nối ngoại vi



Giao diện Ngắt kết nối ngoại vi

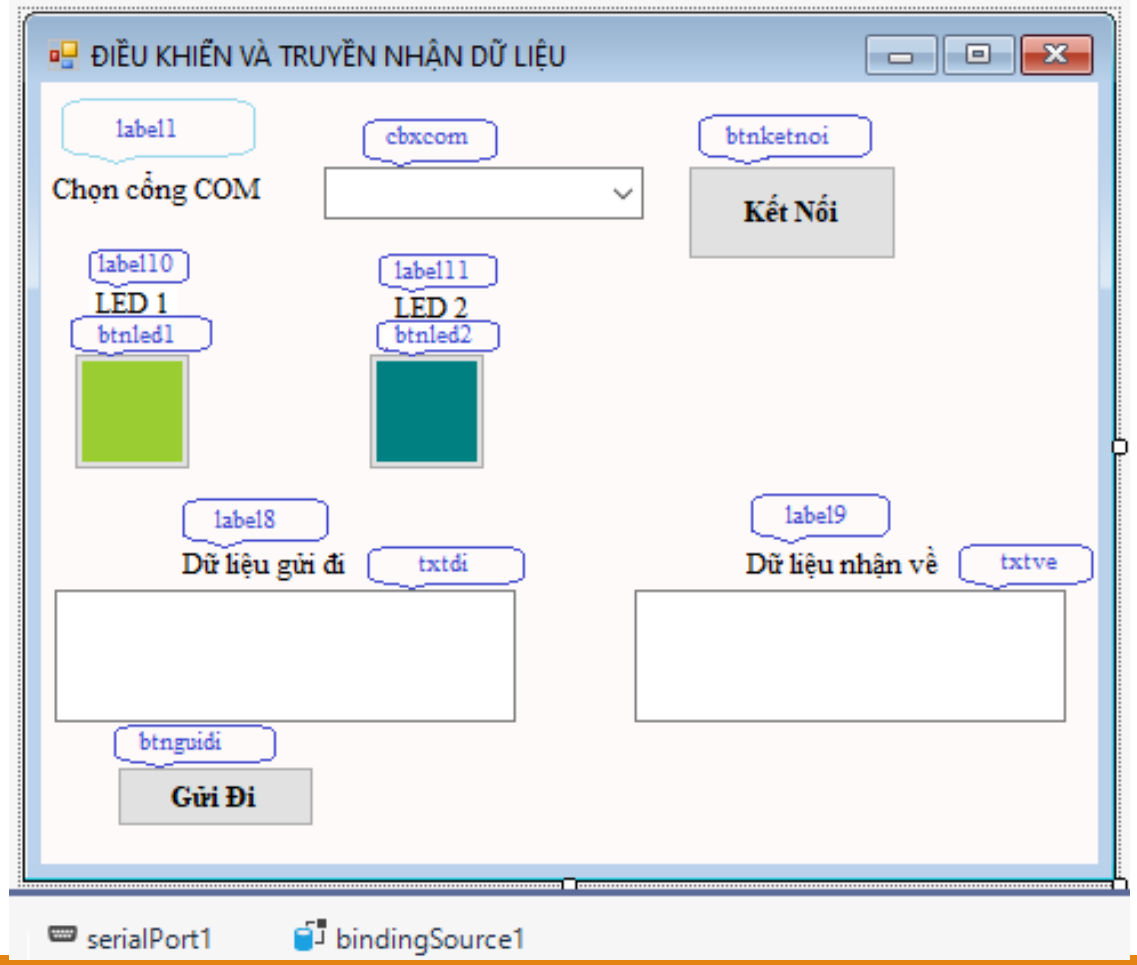
3.4 GIAO TIẾP THIẾT BỊ NGOẠI VI VÀ ỨNG DỤNG TRONG ĐIỀU KHIỂN

❖ **Ví dụ áp dụng 2**: Ứng dụng Windows Form xây dựng giao diện có chức năng giao tiếp cổng COM, nhận dữ liệu là khoảng cách được đo từ cảm biến siêu âm, truyền dữ liệu và điều khiển đèn LED .

- Giao diện có 4 chức năng sau:
  - *Nhận được các cổng COM của PC.*
  - *Nhận được dữ liệu tốc độ từ thiết bị ngoại vi đẩy lên thông qua kết nối IO.*
  - *Truyền dữ liệu là chuỗi từ trên Form xuống thiết bị ngoại vi.*
  - *Điều khiển Bật/Tắt đèn LED trên thiết bị ngoại vi thông qua Form.*

3.4 GIAO TIẾP THIẾT BỊ NGOẠI VI VÀ ỨNG DỤNG TRONG ĐIỀU KHIỂN

- Giao diện được thiết kế như hình bên:



3.4 GIAO TIẾP THIẾT BỊ NGOẠI VI VÀ ỨNG DỤNG TRONG ĐIỀU KHIỂN

- Chương trình thực hiện:

```
public Form1()
    {
        InitializeComponent();
    }
private void Form1_Load(object sender, EventArgs e)
    {
        //serialPort1.Open();
        string[] ComList = SerialPort.GetPortNames();
        Array.Sort(ComList);
        cbxcom.Items.AddRange(ComList);
    }
private void button2_Click(object sender, EventArgs e)
    {
        serialPort1.Write(txtdi.Text);
    }
private void Form1_FormClosing(object sender, FormClosingEventArgs e)
    {
        serialPort1.Close();
    }
private void serialPort1_DataReceived(object sender,
SerialDataReceivedEventArgs e)
    {
        string data = serialPort1.ReadExisting();

        Invoke(new MethodInvoker(() => txtve.Text =" khoảng cách :
"+"\n"+ data + "cm"));
    }
```

3.4 GIAO TIẾP THIẾT BỊ NGOẠI VI VÀ ỨNG DỤNG TRONG ĐIỀU KHIỂN

- Chương trình thực hiện:

```
private void btnketnoi_Click(object sender, EventArgs e)
    {
      if (cbxcom.Text == "")
      {
         MessageBox.Show("Vui lòng chọn cổng COM ");
      }
      //serialPort1.PortName = cbxcom.Text;
      if (serialPort1.IsOpen)
      {
         serialPort1.Close();
         btnketnoi.Text = " Kết Nối";
      }
      else
      {
         serialPort1.PortName = cbxcom.Text;
         serialPort1.Open();
         btnketnoi.Text = " Ngắt Kết Nối ";
      }
    }
```

```
private void btnled1_Click(object sender, EventArgs e)
    {
      i++;
      if (i%2==0)
      {
          serialPort1.Write("Bat led 1");
         btnled1.BackColor = Color.Blue;
         btnled1.Text = "Bật";
      }
      else if(i % 2 == 1)
      {
          serialPort1.Write("Tat led 1");
         btnled1.BackColor = Color.White;
         btnled1.Text = "Tắt";
      }
```

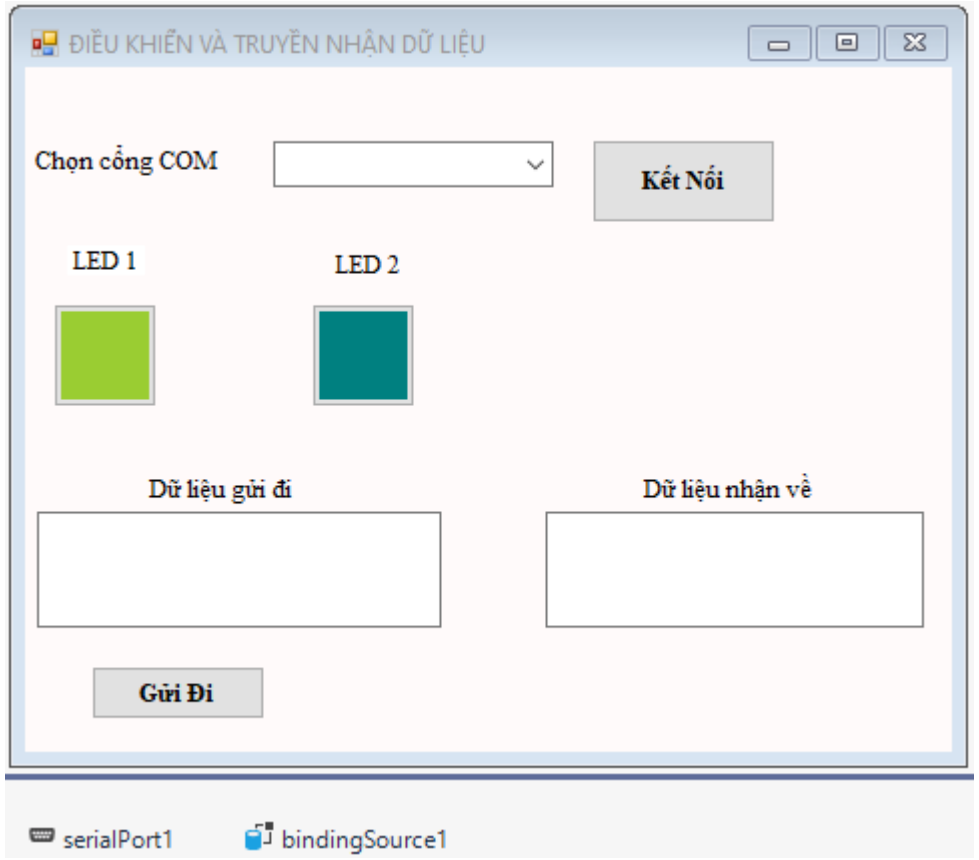
3.4 GIAO TIẾP THIẾT BỊ NGOẠI VI VÀ ỨNG DỤNG TRONG ĐIỀU KHIỂN

- Chương trình thực hiện:

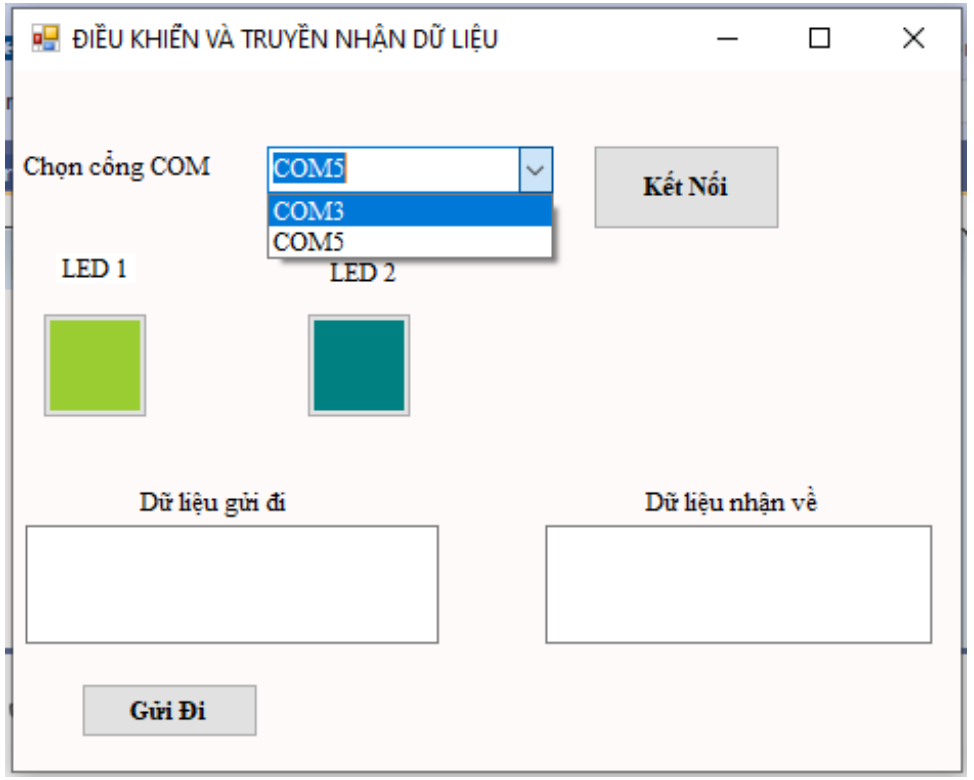
```
private void btnled2_Click(object sender, EventArgs e)
    {
        i++;
        if (i % 2 == 0)
        {
            serialPort1.Write("Bat led 2");
            btnled2.BackColor = Color.Blue;
            btnled2.Text = "Bật";
        }
        else if (i % 2 == 1)
        {
            serialPort1.Write("Tat led 2");
            btnled2.BackColor = Color.Black;
            btnled2.Text = "Tắt";
        }
    }
```

3.4 GIAO TIẾP THIẾT BỊ NGOẠI VI VÀ ỨNG DỤNG TRONG ĐIỀU KHIỂN

- Kết quả:



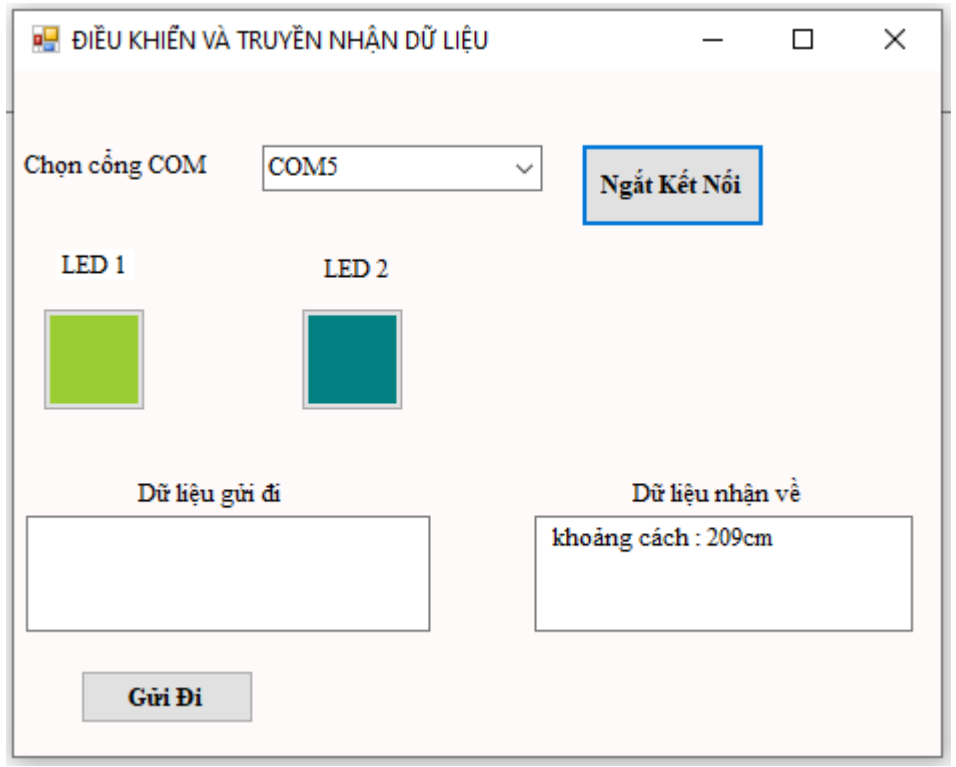
Giao diện khởi động



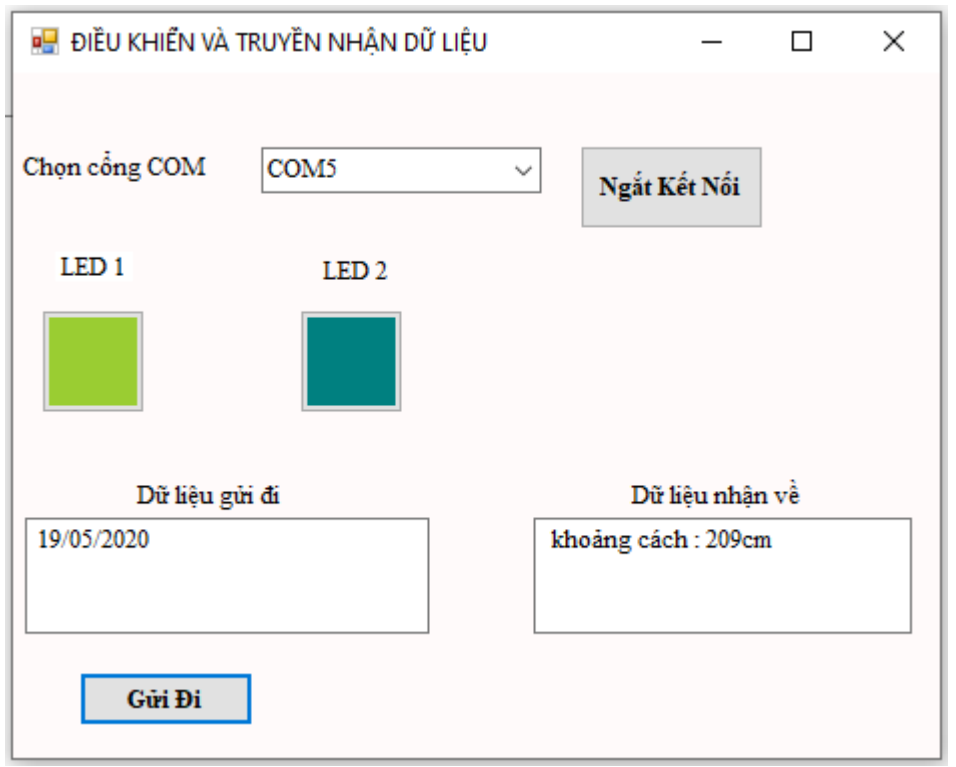
Giao diện Chọn cổng kết nối ngoại vi

3.4 GIAO TIẾP THIẾT BỊ NGOẠI VI VÀ ỨNG DỤNG TRONG ĐIỀU KHIỂN

- Kết quả:



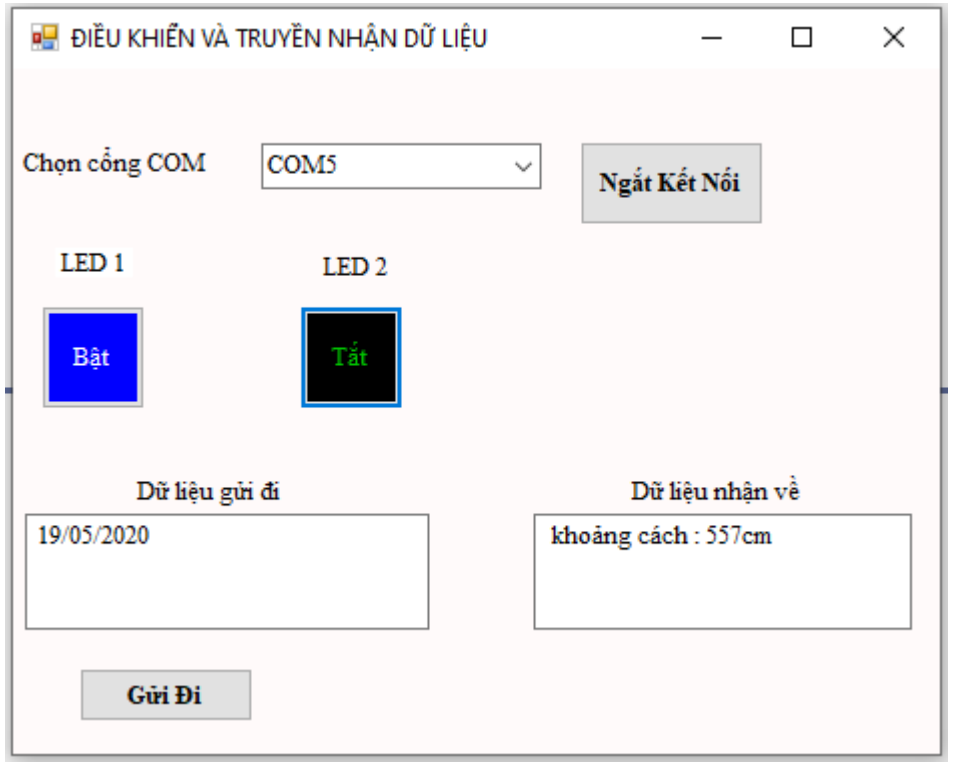
Giao diện Kết nối ngoại vi cổng COM5 nhận dữ liệu là khoảng cách từ cảm biến siêu âm đưa về



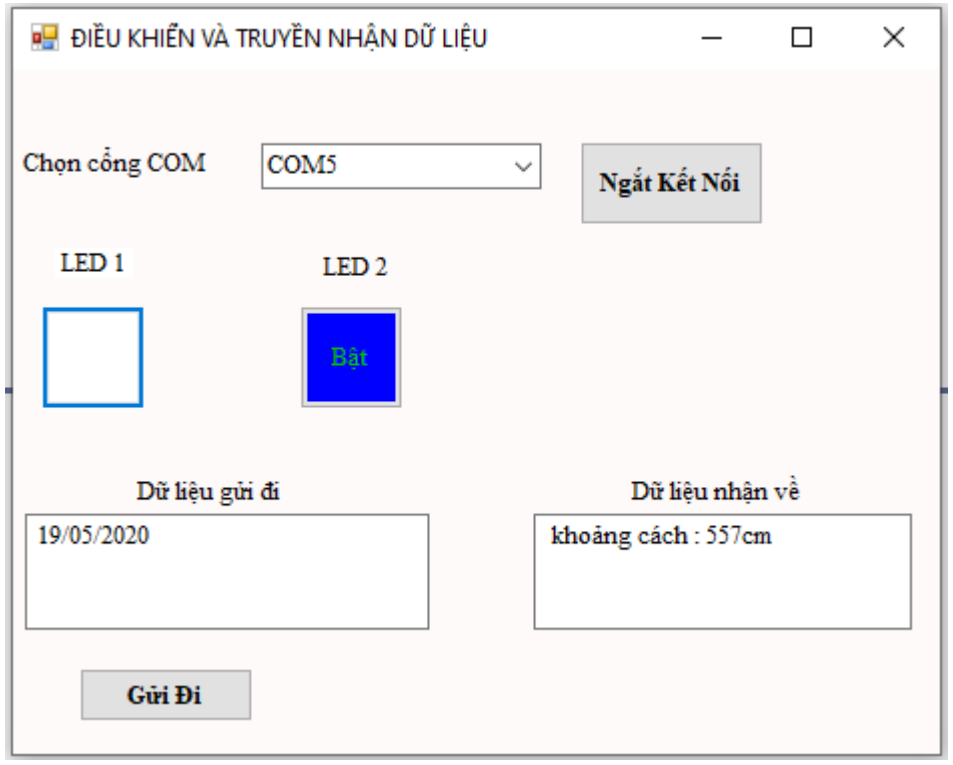
Giao diện Gửi Đi dữ liệu xuống thiết bị ngoại vi thông qua cổng COM5

3.4 GIAO TIẾP THIẾT BỊ NGOẠI VI VÀ ỨNG DỤNG TRONG ĐIỀU KHIỂN

- Kết quả:



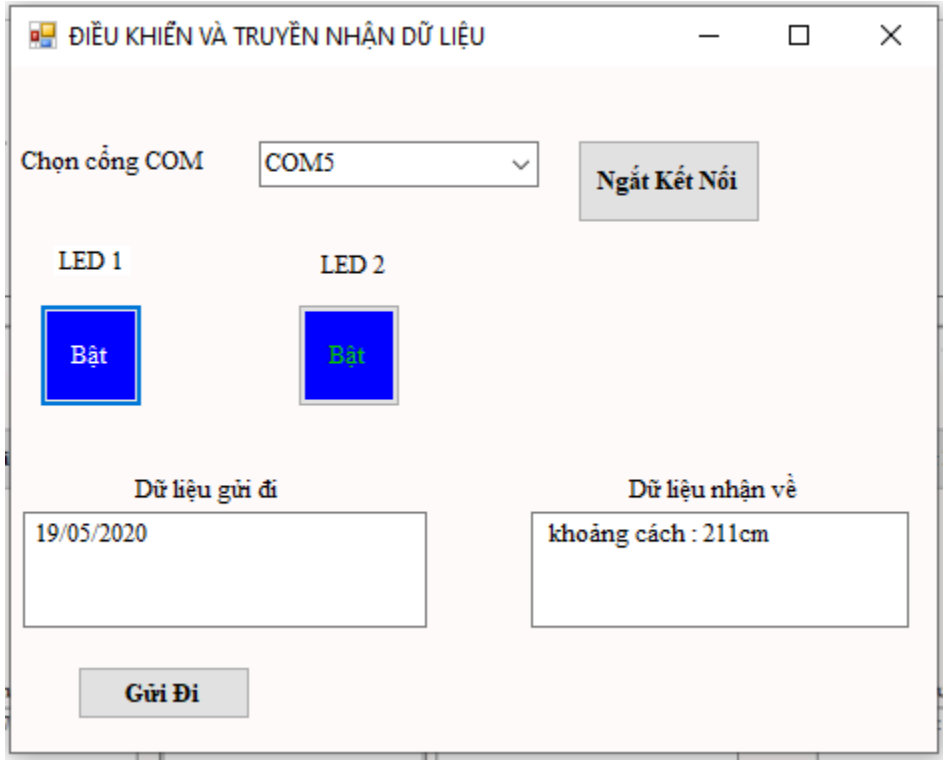
Giao diện Điều khiển LED 1 Bật, LED 2 Tắt



Giao diện Điều khiển LED 1 Tắt, LED 2 Bật

3.4 GIAO TIẾP THIẾT BỊ NGOẠI VI VÀ ỨNG DỤNG TRONG ĐIỀU KHIỂN

- Kết quả:



Giao diện Điều khiển LED 1 Bật, LED 2 Bật