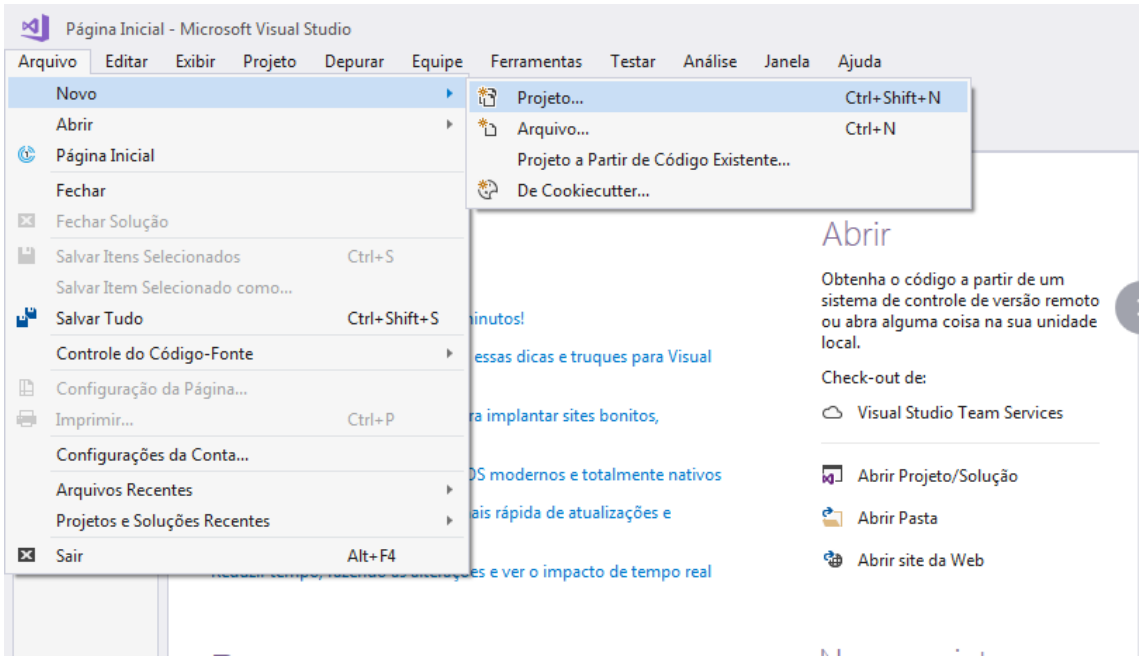
CONTROLANDO ARDUINO USANDO C#

Para prosseguir em nosso estudo, vamos usar o ambiente Visual Studio e linguagem de programação C#.

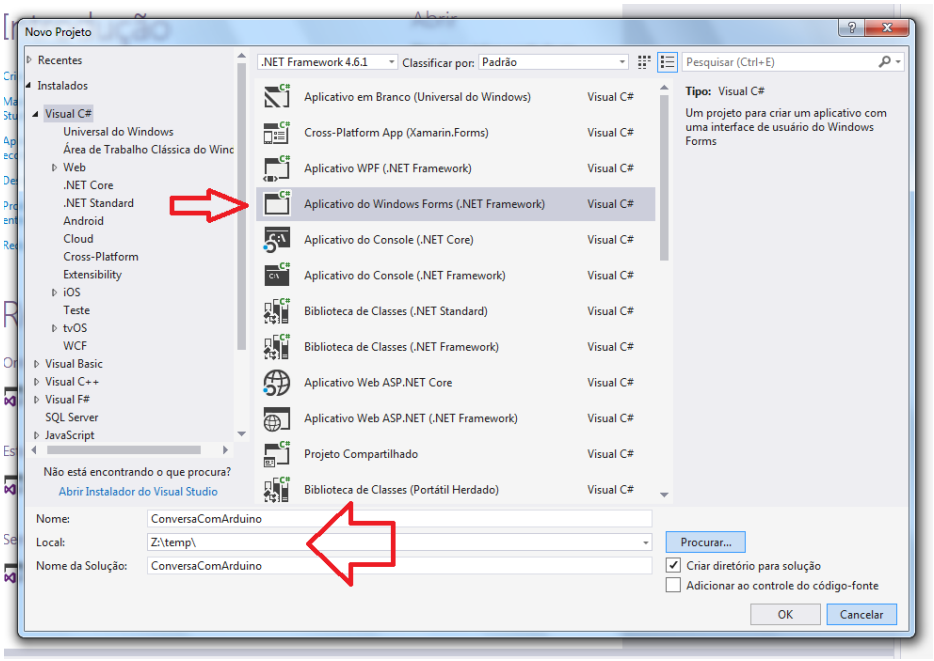
Testando a comunicação

Primeiro vamos criar uma aplicação que envia dados para o Arduino usando a porta Serial. Esta aplicação poderá ser usada como base para qualquer aplicação de comunicação com o Arduino ou qualquer outro dispositivo conectado a uma porta serial do computador.

Ao Iniciar o Visual Studio será exibida sua tela inicial e para iniciar um novo projeto deve-se acessar o menu *ARQUIVO > Novo > Projeto*. Como vamos trabalhar com a linguagem C#, deve-se selecionar a opção *Visual C#* no menu lateral. Agora vamos iniciar o passo a passo pra criar nossa aplicação:

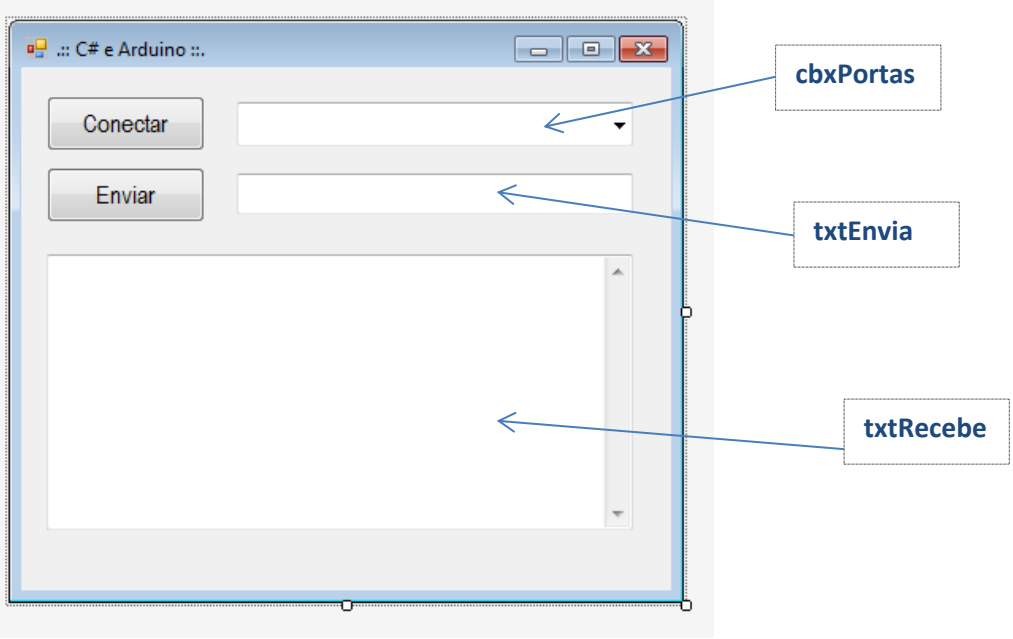


O primeiro passo é iniciar um novo projeto *Aplicativo do Windows Forms* em C#. Altere o nome e local de salvamento do projeto.

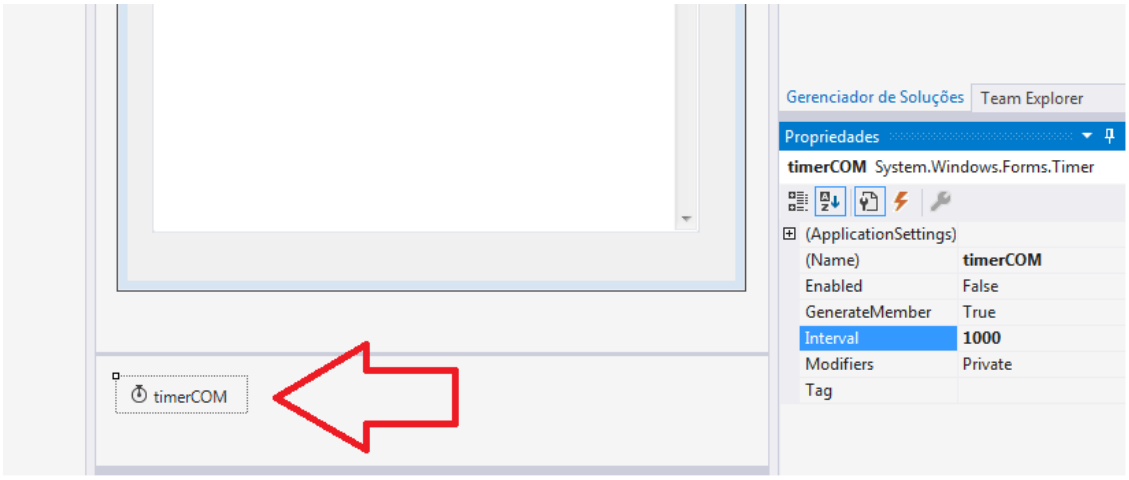


Agora vamos inserir os seguintes componentes no Formulário:

Componente	Name	Text	Outros
Button	btnConectar	Conectar	
ComboBox	cbxPortas		
Button	btnEnviar	Enviar	
TextBox	txtEnvia		
TextBox	txtRecebe		Multiline = true ScrollBar = Vertical



O próximo passo é inserir um componente timer que será responsável pela atualização das portas COM disponíveis no PC. Selecione o componente **timer** e clique dentro do Form. Será exibido logo abaixo o componente timer1. Troque a propriedade *Name* para “timerCOM” e *Interval* para 1000, conforme exibido a seguir:



Por último vamos inserir o componente de comunicação serial, o SerialPort. Selecione o componente SerialPort e depois clique dentro do Form. Será exibido este componente ao lado do timerCOM: Altere o nome do componente para spSerial.

Com os componentes inseridos no Form, vamos para a codificação.

Antes de conectar a porta Serial, é necessário verificar as portas COMs disponíveis para uso, e qual a porta o usuário deseja conectar. Para isso vamos atualizar a cada segundo a ComboBox com as portas disponíveis (por isso o timerCOM tem interval 1000). Vamos criar um método privado dentro da classe **frmArduino**, que será chamado de atualizaListaCOMs. Clique

com o botão direito no Form e selecione a opção View code. Insira o método atualizaListaCOMs(), conforme exibido no código a seguir:

```
private void atualizaListaCOMs()
{
    int i = 0;
    bool quantDiferente; //flag para sinalizar se
                         //a quantidade de portas mudou
    quantDiferente = false;

    //se a quantidade de portas mudou
    if (cbxPortas.Items.Count == SerialPort.GetPortNames().Length)
    {
        foreach (string s in SerialPort.GetPortNames())
        {
            if (!cbxPortas.Items[i++].Equals(s))
            {
                quantDiferente = true;
                break;  // escapa do foreach
            }
        }
    }
    else
    {
        quantDiferente = true;
    }

    //Se não foi detectado diferença
    if (!quantDiferente)
    {
        return;                     //retorna
    }

    //limpa comboBox
    cbxPortas.Items.Clear();

    //adiciona todas as COM diponíveis na lista
    foreach (string s in SerialPort.GetPortNames())
    {
        cbxPortas.Items.Add(s);
    }
    //seleciona a primeira posição da lista
    cbxPortas.SelectedIndex = 0;
}
```

Nos includes do arquivo, adicione o seguinte using:

```
using System.IO.Ports;
```

Rode a aplicação e veja se alguma porta aparece no combo. Caso o arduino esteja conectado ao seu computador, a porta deve aparecer automaticamente. Para que isso aconteça, vamos usar o timerCOM que está configurado para gerar um evento a cada segundo (interval em 1000). Inicialmente deve-se habilitar o timer logo após a inicialização do Form e colocar o método de atualização dentro do evento timerCOM_tick, conforme exibido a seguir:

```
public partial class frmArduino : Form
    {
        public frmArduino()
        {
            InitializeComponent();
            timerCOM.Enabled = true;
        }

        private void timerCOM_Tick(object sender, EventArgs e)
        {
            atualizaListaCOMs();
     }

     ...
```

Obs.: Para gerar o evento timerCOM_tick basta dar duplo clique no componente timerCOM na aba design.

Agora já se pode escolher em qual porta a aplicação vai conectar. O evento click do btConectar será usado para fazer a conexão. Para criar esse evento, selecione a aba de design do Form e dê um duplo clique no botão conectar. Será gerado o evento e agora deve-se inserir o código para conexão. O botão conectar também servirá para desconectar quando a porta já estiver conectada, confira o código a seguir:

```
private void btConectar_Click(object sender, EventArgs e)
{
   if (! spSerial.IsOpen) // Porta fechada
   {
     try
     {
        spSerial.PortName = cbxPortas.Items[cbxPortas.SelectedIndex].ToString();
        spSerial.Open();
```

```
        }
        catch
        {
            return;
        }
        if (spSerial.IsOpen) // abriu!!
        {
            btnConectar.Text = "Desconectar";
            cbxPortas.Enabled = false;
        }
    }
    else
    {
        try
        {
          spSerial.Close();
          cbxPortas.Enabled = true;
          btnConectar.Text = "Conectar";
        }
        catch
        {
          return;
        }

    }
}
```

Por segurança, é necessário colocar uma proteção para que o programa não seja fechado e deixe a porta COM aberta, dessa forma impedindo que outros programas possam usá-la. Para isso vamos fechar a porta dentro do evento frmArduino_FormClosed:

```
private void frmArduino_FormClosed(object sender, FormClosedEventArgs e)
{
        if (spSerial.IsOpen)  // se porta aberta
              spSerial.Close();         //fecha a port
}
```

PRONTO! Todo o processo para conexão e fechamento da porta serial já está feito!!

O próximo passo é fazer o programa enviar para o Arduino o que for digitado dentro do textBoxEnviar. Para isso, no evento Click do botão enviar, vamos inserir o seguinte código:

```
private void btnEnviar_Click(object sender, EventArgs e)
{
      if (spSerial.IsOpen)          //porta está aberta
            spSerial.Write(txtEnvia.Text);  //envia o texto digitado no textbox
}
```

Agora vamos receber dados do Arduino!!!

A recepção de dados requer um pouco mais de atenção. Inicialmente deve-se criar um evento spSerial_DataReceived e uma variável global do tipo String. O processo de recepção acontece em uma Thread diferente da atualização dos componentes. A atualização do txtRecebe deve ser feita fora do evento de recepção da serial. Para isso criamos uma função trataDadoRecebido.

```
public partial class frmArduino : Form
{

    string RxString; // declarada como atributo da classe

    public frmArduino()

    ...


   private void trataDadoRecebido(object sender, EventArgs e)
    {
        txtRecebe.AppendText(RxString);
    }

    private void spSerial_DataReceived(object sender, SerialDataReceivedEventArgs e)
    {
        //le o dado disponível na serial
        RxString = spSerial.ReadExisting();

        //chama outra thread para escrever  o dado no text box
        this.Invoke(new EventHandler(trataDadoRecebido));
    }
```

Para testar a aplicação junto ao Arduino, vamos fazer o upload do seguinte sketch:

```
void setup()
{
  Serial.begin(9600);  //inicia comunicação serial com 9600
}

void loop()
```

```
{
  if(Serial.available())        //se algum dado disponível
  {
    char c = Serial.read();    //le o byte disponível
    Serial.write(c);           //retorna o que foi lido
  }
}
```

Nesse programa o Arduino simplesmente retornará o dado que ele receber. Dessa forma, quando enviarmos dados pelo programa, estes serão exibidos no computador por meio do txtRecebe.

Agora que a aplicação está completa, ou seja, já conseguimos enviar e receber dados!!!

Acendendo e apagando LED

Vamos aproveitar a nossa tela para criar uma aplicação mais funcional!!

Vamos controlar o acender e apagar de um LED no arduino através da nossa aplicação em C#. Para isso, vamos estabelecer um protocolo de comunicação entre a nossa aplicação e o Arduino para que a aplicação possa informar um código para acender o LED e um outro código para apagar o LED, assim, basta o Arduino interpretar esse nosso protocolo de comunicação e agir de acordo com o solicitado. Nosso protocolo então fica assim:

CODIGO PASSADO	AÇÃO A SER REALIZADA
L	Acender o LED (Ligado)
D	Apagar o LED (Desligado)

Código para Upload no Arduino para interpretar o protocolo acima:

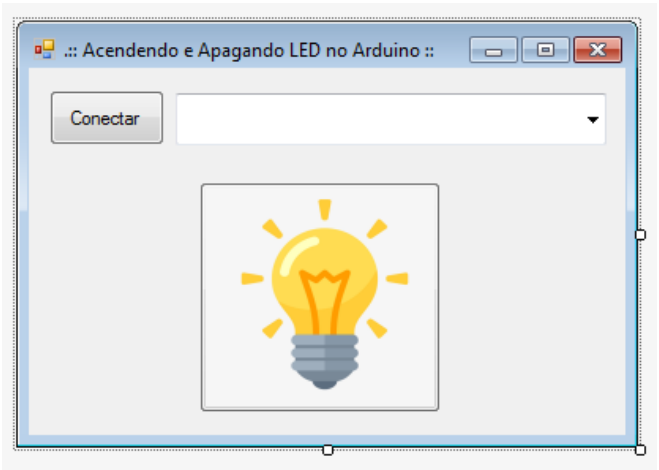
```
int valor_AD = 0;
int valor_leituras = 0;
byte i = 0;

void setup() {
  Serial.begin(9600);    //configura comunicação serial com 9600 bps
  pinMode(LED,OUTPUT);   //configura pino do led como saída
}
```

```
void loop() {
   if (Serial.available()) //se byte pronto para leitura
   {
    switch(Serial.read())      //verifica qual caracter recebido
    {
      case 'L':                  //caso 'L' → Ligado
              digitalWrite(LED,!digitalRead(LED)); //inverte estado do LED
              break;

      case 'D':     // Caso 'D' → Desligado
              Serial.println(valor_AD);
              break;
    }
  }

  //MEDIA DE LEITURAS
  valor_leituras += analogRead(A0);
  i++;

  if(i==16)
  {
    i = 0;
    valor_AD = valor_leituras>>4;
    valor_leituras = 0;
  }

}
```

Execute a aplicação em C#, conectando a porta na qual o Arduino está ligado e envie o caractere 'L'. Verifique o resultado no LED conectado ao pino 13 da placa Arduino. Logo depois, envie o caractere 'D' e verifique o resutado.

Com base nesse protocolo, altere o formulário para controlar o acender e apagar do LED apenas com um botão, alternando o caractere do protocolo a ser enviado!

DICA: Crie uma classe em C# para representar o protocolo de comunicação com o Arduino para trabalho com LED!!



-- FIM.