Relatório do Projeto 1 – Interface Multifunções

Gabriel Cortez

Heloísa Passos

João Vitor Pinheiro

Introdução

A atividade proposta teve como objetivo o desenvolvimento de um aplicativo em Windows Forms na linguagem C#, visando à integração entre software e hardware. Os alunos, organizados em equipes de no máximo quatro alunos, foram desafiados a criar uma interface capaz de ler um sensor de temperatura e um potenciômetro, bem como controlar um LED por meio de um botão de liga/desliga. Com isso, buscou-se consolidar o conhecimento adquirido na programação, eletrônica e comunicação serial.

A relevância deste projeto está na aplicação prática de conceitos-chaves de interfaces industriais, reforçando a compreensão da comunicação entre sistemas embarcados (como o Arduino) e aplicações desktop. Ao simular um ambiente real de automação, os alunos puderam desenvolver suas habilidades práticas e o trabalho em equipe, se aproximando do ambiente de trabalho encontrado na indústria.

Durante a execução da atividade, buscou-se não apenas cumprir as funcionalidades do sistema, mas também garantir uma interface com uma boa interação ao usuário, e uma comunicação eficiente e estável entre os dispositivos físicos e a porta serial do computador.

O que foi utilizado

Para o desenvolvimento do projeto, foi utilizado o ambiente de desenvolvimento Visual Studio, com a linguagem de programação C# e a biblioteca Windows Forms para a criação da interface gráfica. A comunicação com os sensores e atuadores foi realizada através de um microcontrolador Arduino, conectado ao computador via porta COM, utilizando a classe `SerialPort` do .NET Framework.

Como hardware, utilizou-se um sensor de temperatura (DHT11), um potenciômetro para simular variação de tensão, um LED para controle de saída, e uma placa Arduino UNO para leitura e envio dos dados. O Arduino foi programado com o código responsável por ler os valores dos sensores e receber comandos para acionar o LED, comunicando-se com o aplicativo C# por meio de mensagens no protocolo serial.

Além disso, no lado do aplicativo, foram criados elementos visuais como *labels*, *textboxes* e *buttons* para exibir os dados lidos e permitir o controle do LED. A lógica de leitura contínua dos dados da porta serial foi implementada com uso de *event handlers* e *threads*, garantindo uma atualização em tempo real sem travar a interface gráfica.

Código

Alguns trechos do código do Arduino para referência do que foi feito, com a explicação do que faz cada linha na frente:

```
const int pino_A0 = A0;  // Define o pino analógico A0
const int ledPin = 13;   // Pino do LED (ajuste conforme necessário)
bool ledEstado = false;  // Estado inicial do LED

// Define o pino onde o DHT11 está conectado
#define DHTPIN 4

// Define o tipo de sensor
#define DHTTYPE DHT11

// Inicializa o sensor DHT
DHT dht(DHTPIN, DHTTYPE);
```

```
 // Lê a temperatura
  float t = dht.readTemperature();

// Verifica se a leitura falhou
  if (isnan(t)) {
    Serial.println("Falha na leitura do sensor DHT11!");
    return;
```

```
// Envia os dados para o C#
  Serial.print(t);  // Envia a temperatura
  Serial.print(","); // Delimitador para separar os dados
  Serial.println(tensao_A0, 2);  // Envia a tensão do A0 (com 2 casas decimais)
```

```
//Envia os dados do arduino e mostra na interface do aplicativo

private void serialPort1_DataReceived(object sender, SerialDataReceivedEventArgs e)
{
   if (!serialPort1.IsOpen) return;

   try
   {
      string dados = serialPort1.ReadExisting();
      bufferRecebido += dados;

      if (bufferRecebido.Contains("\n"))
      {
         this.Invoke(new Action(() =>
         {
            string[] linhas = bufferRecebido.Split(new[] { '\r', '\n' },
StringSplitOptions.RemoveEmptyEntries);

            if (linhas.Length > 0)
            {
               string ultimaLinha = linhas[linhas.Length - 1];
               bufferRecebido = ""; // limpar buffer após leitura

               string[] partes = ultimaLinha.Split(',');

               if (partes.Length == 2 &&
                  double.TryParse(partes[0], System.Globalization.NumberStyles.Float,
System.Globalization.CultureInfo.InvariantCulture, out double temperatura) &&
                  double.TryParse(partes[1], System.Globalization.NumberStyles.Float,
System.Globalization.CultureInfo.InvariantCulture, out double tensao))
               {
                  thermControl1.UpdateControl((int)temperatura);
                  aGauge1.Value = (int)(tensao * 20);
               }
            }
         }));
      }
   }
   catch (Exception ex)
   {

      Console.WriteLine("Erro na leitura da porta serial: " + ex.Message);
   }
}
```

Conclusão

A realização da atividade permitiu vivenciar o desenvolvimento completo de um sistema de interface homem-máquina, desde a leitura de sensores até o controle de atuadores, utilizando comunicação serial. Essa experiência foi fundamental para fortalecer o entendimento sobre a integração entre software e hardware, especialmente no contexto industrial e de automação.

Por fim, o projeto demonstrou como ferramentas acessíveis e amplamente utilizadas, como o Arduino e o Visual Studio, podem ser combinadas para criar aplicações funcionais e relevantes. O trabalho em grupo e a divisão de tarefas contribuíram para a fluidez do desenvolvimento, além de estimular a colaboração e a troca de conhecimento entre os integrantes.