Integração entre PSoC e Smartphone

Victor São Paulo Ruela
Universidade Federal de Minas Gerais

May 10, 2015

Sumário

1 Introdução

O projeto consiste na integração de dados entre um system-on-ship (PSoC) e um aplicativo Android, utilizando para isso um Web Service na plataforma .NET. Através do aplicativo, o usuário consegue visualizar a temperatura do PSoC pressionando um botão. Também foi implementado um cliente Web para consumir o Web Service. Este cliente possui um botão para requisitar a leitura da temperatura do PSoC e a partir dessas leituras, constrói uma tabela contendo todos os valores lidos. Portanto, a seguinte arquitetura de integração foi implementada:

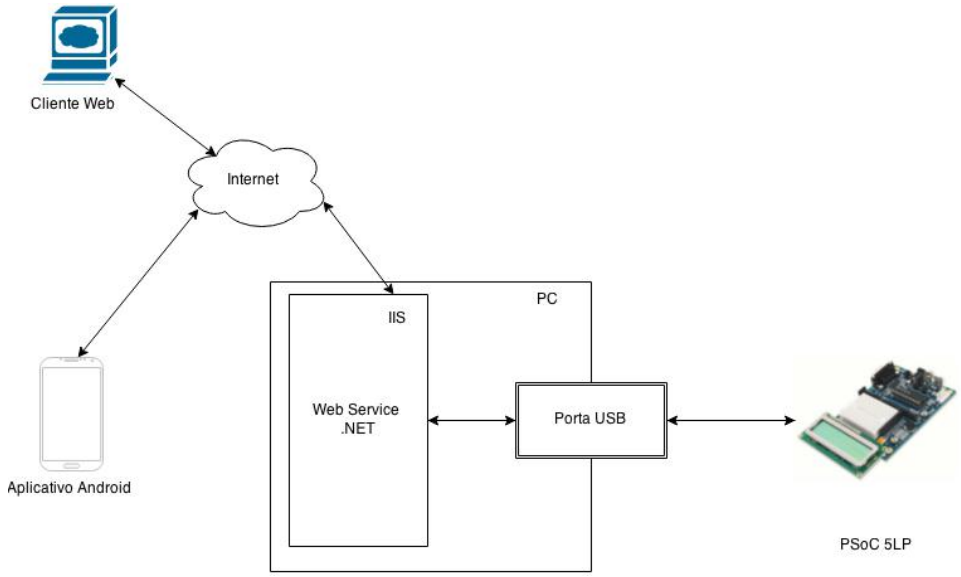

Figure 1: Arquitetura da integração

2 Implementação

2.1 Aplicação do PSoC

A aplicação foi feita no PSoC Creator. A criação de uma aplicação para o PSoC é feita através de diagrama de blocos e em C. Inicialmente, colocamos as instâncias dos componentes que serão utilizados em uma página do arquivo **TopDesig.cysch**, que é o padrão de qualquer projeto no PSoC Cretor. Depois precisamos inicializar e interligar os diversos componentes através das APIs de cada componente, em linguagem C. Para este projeto, foi foram necessários os componentes **DieTemp** e **USBUART**. O primeiro permite a leitura da temperatura do PSoC enquanto o segundo habilita o uso da porta USB da placa. Ambos foram usados com a configuração padrão.

O código em C utiliza as funções da API de cada componente. A lógica de funcionamento da aplicação é bem simples:

1. Inicializar o componente **USBUART**
2. Aguardar enumeração da porta USB
3. Aguardar mensagem de requisiçao de leitura
4. Obter temperatura através do bloco **DieTemp**
5. Enviar para a porta USB

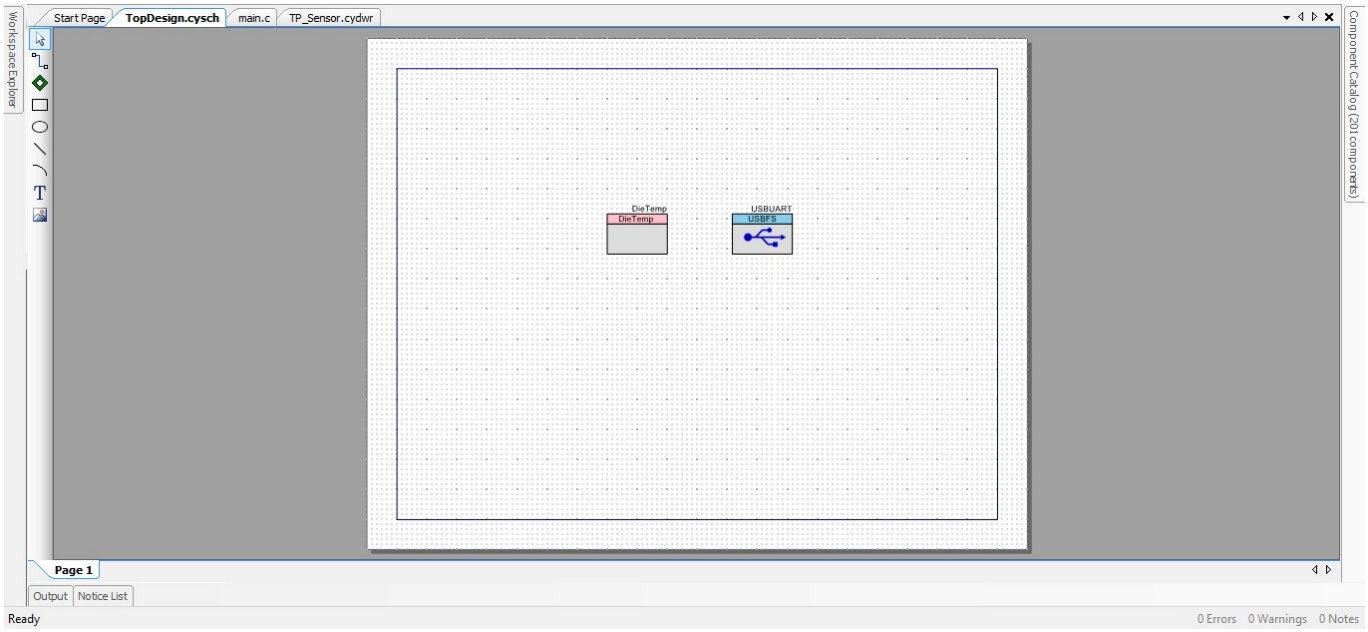

Figure 2: Diagram de blocos no PSoC

O PSoC fica em um loop infinito aguardando novas mensagens de requisição de leitura. O bloco USBUART funciona como uma porta serial COM, facilitando a leitura e escrita de dados pela aplicação **.NET** do Web Service.

2.2 Web Service

O Web Service está implementado na platforma .NET versão 4.0, usando a linguagem C#. Ele será publicado em um servidor web local através do **IIS - Internet Information Services**.Foi necessário realizar uma configuração do roteador: redirecionamento da porta 80 ao endereço de IP local da máquina conectada ao PSoC, para tornar o Web Service visível a todos os computadores e dispositivos conectados à rede. A publicação de Web Service foi facilitada com o uso do **MS Visual Studio 2012**, o qual possui uma ferramenta muito útil para a publicação de aplicativos Web. Isso reduziu bastante o tempo necessário para desenvolver a aplicação, uma vez que não foram encontrados muitos problemas relacionados à configuração do IIS. O Web Service foi configurado para funcionar com o protocolo SOAP e os métodos GET e POST do http.

2.2.1 Estrutura da aplicação

Decidi por implementar o Web Service usando ASMX, porque seu uso ele é mais simples e fácil de configurar, e atendia a todos os requisitos do projeto: uso do HTTP(SOAP) e do IIS. O WCF também poderia ser usado, mas é uma ferramenta mais avançada e poderosa. Como a aplicação desenvolvida é simples, seu uso se mostrou desnecessário. O Web Service fornece dois serviços:

ReadPSoC Serviço responsável pela comunicação com o PSoC através da porta USB, consumido principalmente pelo aplicativo Android. Inicialmente, ele obtém uma lista de valores de temperatura através da variável Session, que contém os dados de sessão para um determinado usuário. Feito isso, começa a troca de dados propriamente dita:

1. Criação do objeto SerialPort:

```
SerialPort s = new SerialPort("COM3", 9600);
```

O parâmetro COM3 é o número da porta COM e 9600 é a baudrate da porta, definida na configuração do bloco USBUART.

2. Abertura da porta serial:

```
s.Open();
```

3. Escrita e leitura na porta serial:

```
s.WriteLine("OK");
line = s.ReadLine();
```

O dado lido também é adicionado à lista descrita acima, para ficar disponível ao usuário da aplicação Web.

4. Fechamento da porta serial:

```
s.Close();
```

As funções acima fazem parte da biblioteca**System.IO.Ports** do .NET. O código possui tratamento de exceções para evitar que a leitura falhe e comprometa a integração do sistema. A principal exceção gerada ocorre quando tentamos ler com o PSoC desconectado do computador. Nesse caso, o Web Service envia a mensagem "Error opening port", indicando a falha. O serviço retorna uma string, contendo a temperatura lida e uma *time stamp* indicando o momento em que a leitura foi feita.

getReadings Serviço usado pela aplicação Web, que será explicada com mais detalhes à frente. Ela retorna uma lista de strings contendo as temperaturas lidas salvas na variável Session do usuário.

Ambos os servicos são definidos como **[WebMethod(EnableSession=true)]**. Esse parâmetro permite salvar e recuperar valores do usuário que está acessando página Web.

2.3 Aplicativo Android

O aplicativo foi implementado em Java, usando as bibliotecas do pacote Android SDK tools.

2.3.1 Interface gráfica

A interface gráfica é bem simples, e consiste de um **EditText** para o usuário digitar o IP, um **TextView** para exibir a temperatura e um **Button** para requisitar um serviço do Web Service. A interface gráfica será exibida na seção dos resultados.

2.3.2 Estrutura da aplicação

O aplicativo foi implementado usando duas classes: **MainActivity** e **XMLResponseHandler**.

MainActivity Consiste na atividade principal (thread primária). É responsável por atualizar o IP digitado e realizar as ações necessárias quando o botão é pressionado. Ao ser pressionado, ele cria uma thread secundária que inicia a comunicação com o Web Service. Esta é considerada uma boa prática em aplicativos Android, pois evita que a interface gráfica fique congelada enquanto a comunicação está sendo processada. A thread cria uma instancia da classe **XMLResponseHandler** para traduzir a mensagem XML recebida pela requisição HTML.

XMLResponseHandler Classe utilizada para processar a mensagem recebida em XML. Ela utiliza a API **XMLPULL V1** para processar a XML e obter a informação desejada, no caso a temperatura do PSoC. Seu funcionamento é simples e direto, e está bem descrito em: `http://www.xmlpull.org/`. A informação extraída é então salva na variável associada ao **TextView** que exibe a temperatura na UI.

2.4 Clente Web

O cliente Web foi implementado usando Web Forms ASP.NET. A página foi desenvolvida em HTML e contém uma breve descrição de sua funcionalidade, um botão para requisitar a leitura do PSoC e uma tabela mostrando os valores recentemente lidos pelo usuário. Ao requisitar o botão, o seguinte script em C# é executado:

```
protected void btnRead_Click(object sender, EventArgs e)
{
        // Criar instancia do objeto ReadPSoCService
        ReadPSoCService.Service1SoapClient client =
                        new ReadPSoCService.Service1SoapClient();

        // Realiza leitura do PSoC
        string result = client.ReadPSoC();
        // Exibe o resultado na pagina
        lblResult.Text = result;

        // Associa a variavel com os valores da tabela a lista
           retornada pelo servico
        gvReadings.DataSource = client.getReadings();
        gvReadings.DataBind();

        gvReadings.HeaderRow.Cells[0].Text = "Recent Temperature
           Readings";
}
```

Inicialmente, precisamos adicionar uma referência ao serviço fornecido pelo Web Service ao projeto. Feito isso, criamos uma instância de um objeto representando o serviço acima. Logo, podemos acessar ambos os serviços definidos anteriormente (**ReadPSoC** e **getReadings**. **ReadPOsC** é usado para fazer a leitura do PSoC quando clica-se no botão e **getReadings** é necessária para montar a tabela com os valores lidos, pois ela acessa a variável contendo os dados de sessão do ususário.

3 Resultados

3.1 Cliente Web

O cliente Web foi testado em diversos computadores na rede local e seu funcionamento foi como o esperado em todos os casos. A página web foi acessada de um computador com sistema operacional Windows, outro com iOS e de smartphones, funcionando corretamente em todos.

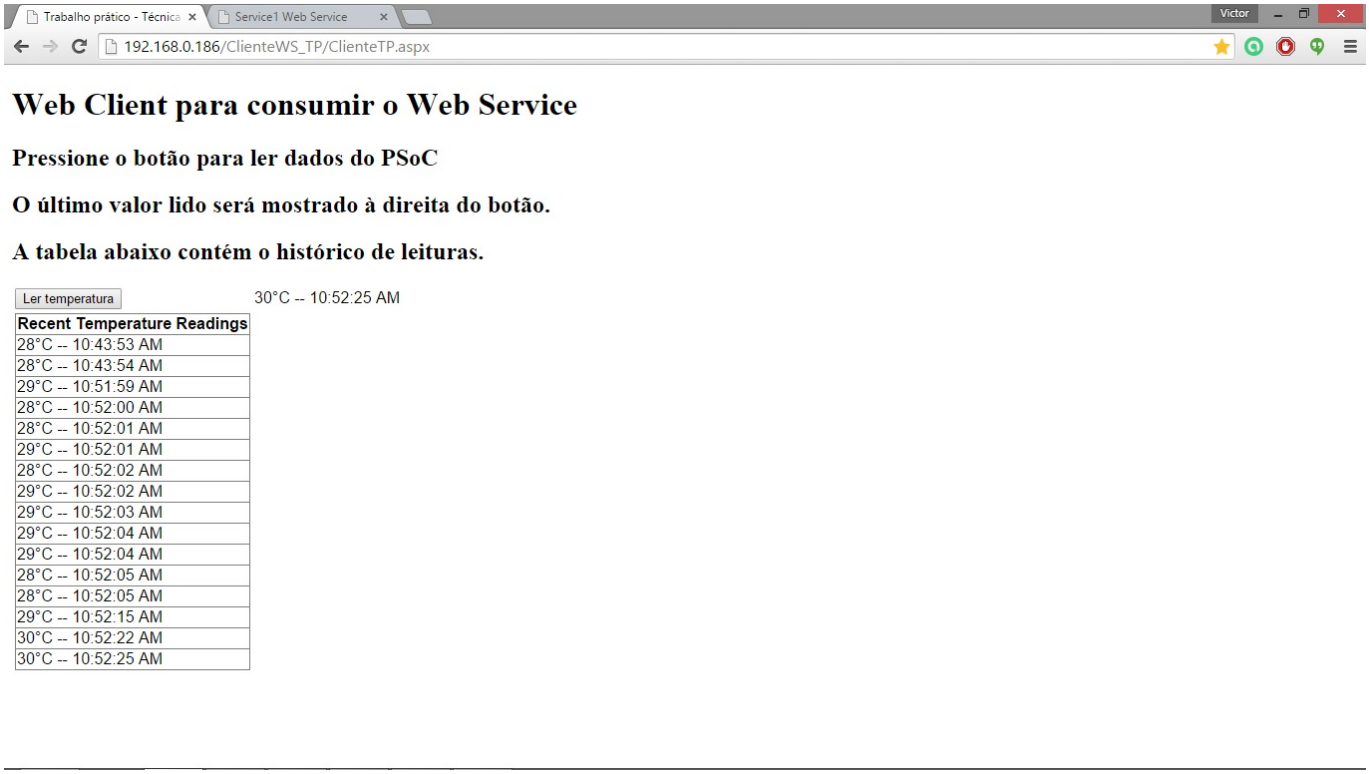

Figure 3: Funcionamento do cliente Web

3.2 Aplicativo Android

O aplicativo Android foi testado no emulador de dispositivos Android e em um Nexus 4. Ambos funcionaram perfeitamente.

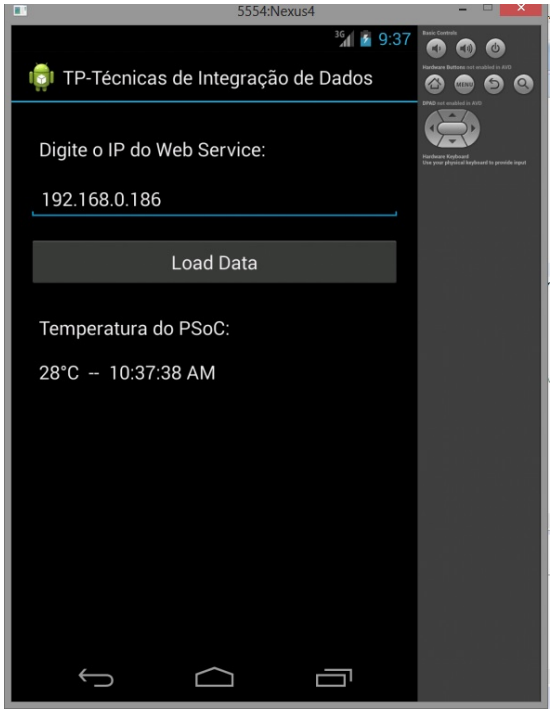

Figure 4: Screenshot do emulador de Dispositivos Android

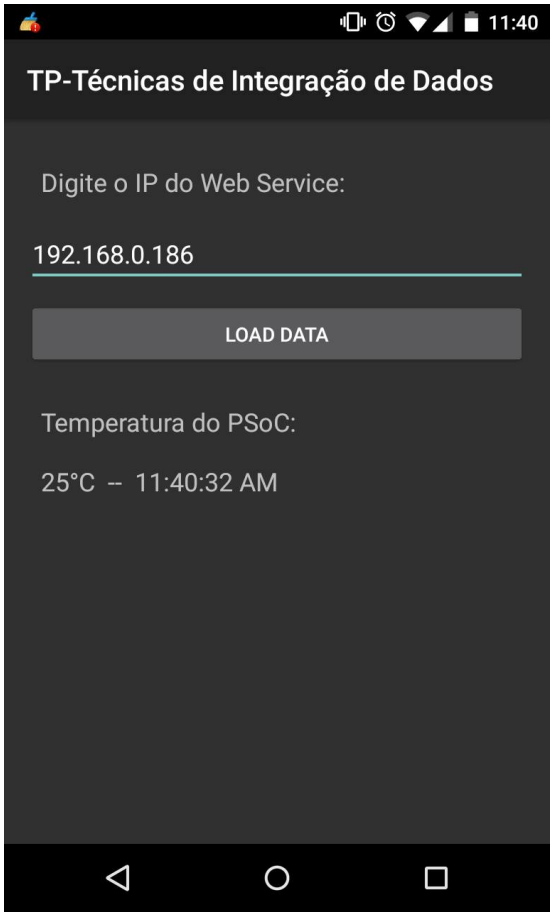

Figure 5: Screenshot de um Nexus 4

4 Conclusão