Protótipo de plugue para tomada elétrica controlado remotamente

Aluno(a): Dyego Alekssander Maas

Orientador: Francisco Adell Péricas

FURB
UNIVERSIDADE DE BLUMENAU

Roteiro

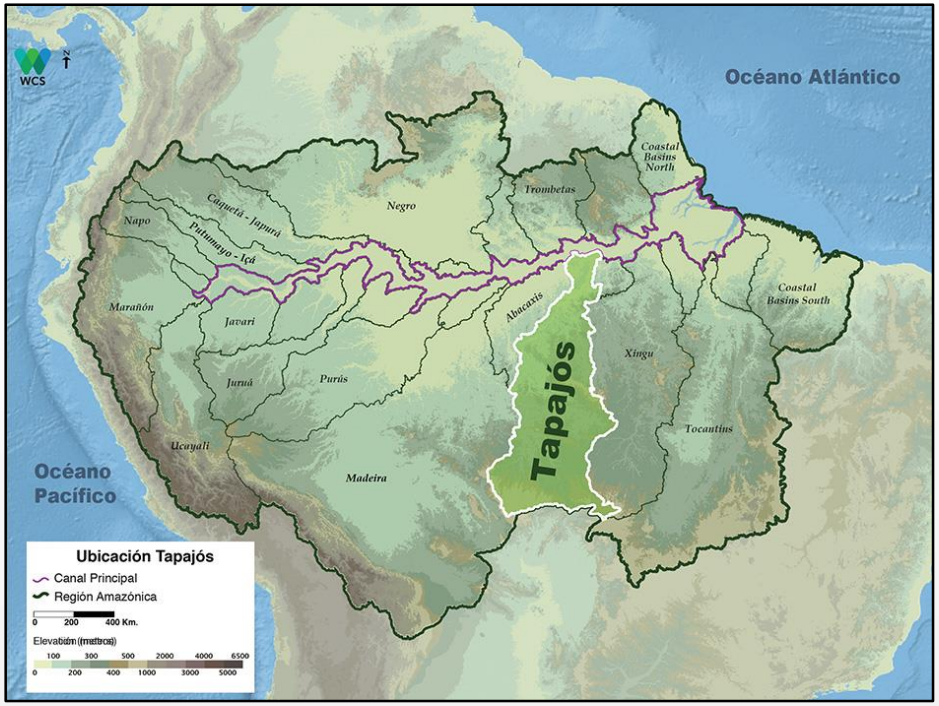
- Introdução
- Objetivos
- Fundamentação teórica
- **Desenvolvimento**
- Conclusões
- Sugestões de continuidade
- Demonstração

Introdução

Desperdício de 53TWh em 2015

40 novas usinas hidrelétricas na Bacia do Rio Tapajós

Consumo residencial de 9,6%



FURB
UNIVERSIDADE DE BLUMENAU

Objetivo Geral

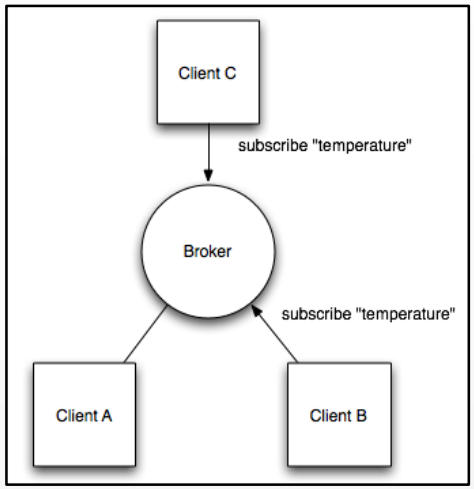
Desenvolver um plugue para tomada elétrica que possa, através do emprego de sensores e atuadores, ser controlado remotamente e que permita o monitoramento do consumo elétrico dos equipamentos conectados.

Objetivos Específicos

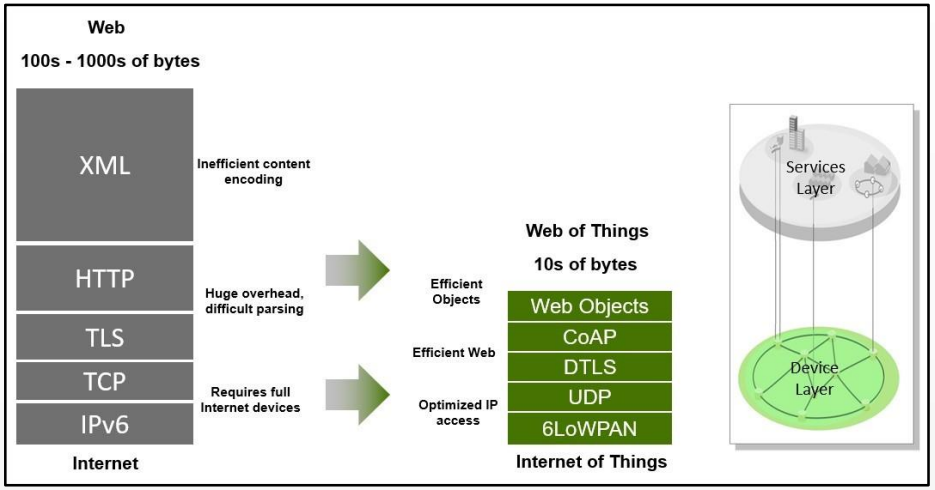
- Desenvolver um protótipo de plugue para tomada elétrica controlado remotamente;
- Permitir ativar e desativar, via aplicativo móvel, o fornecimento de energia elétrica para os aparelhos conectados ao plugue;
- Disponibilizar em um aplicativo móvel relatórios do consumo de energia elétrica dos aparelhos conectados ao plugue.

Internet das Coisas

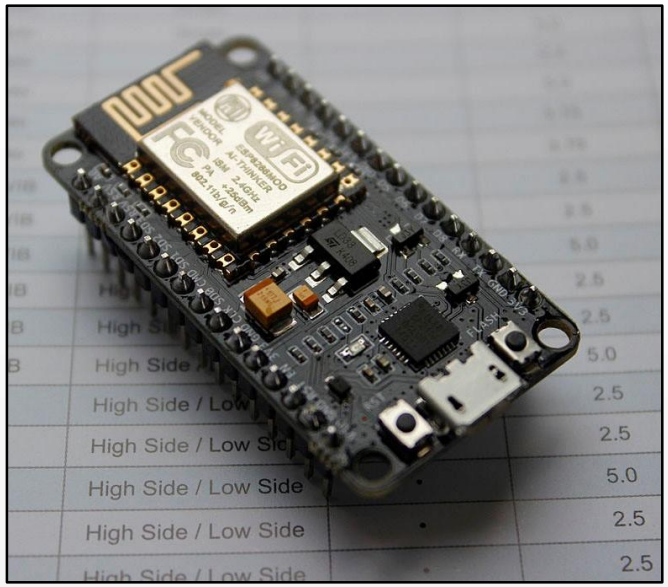
MQTT



CoAP

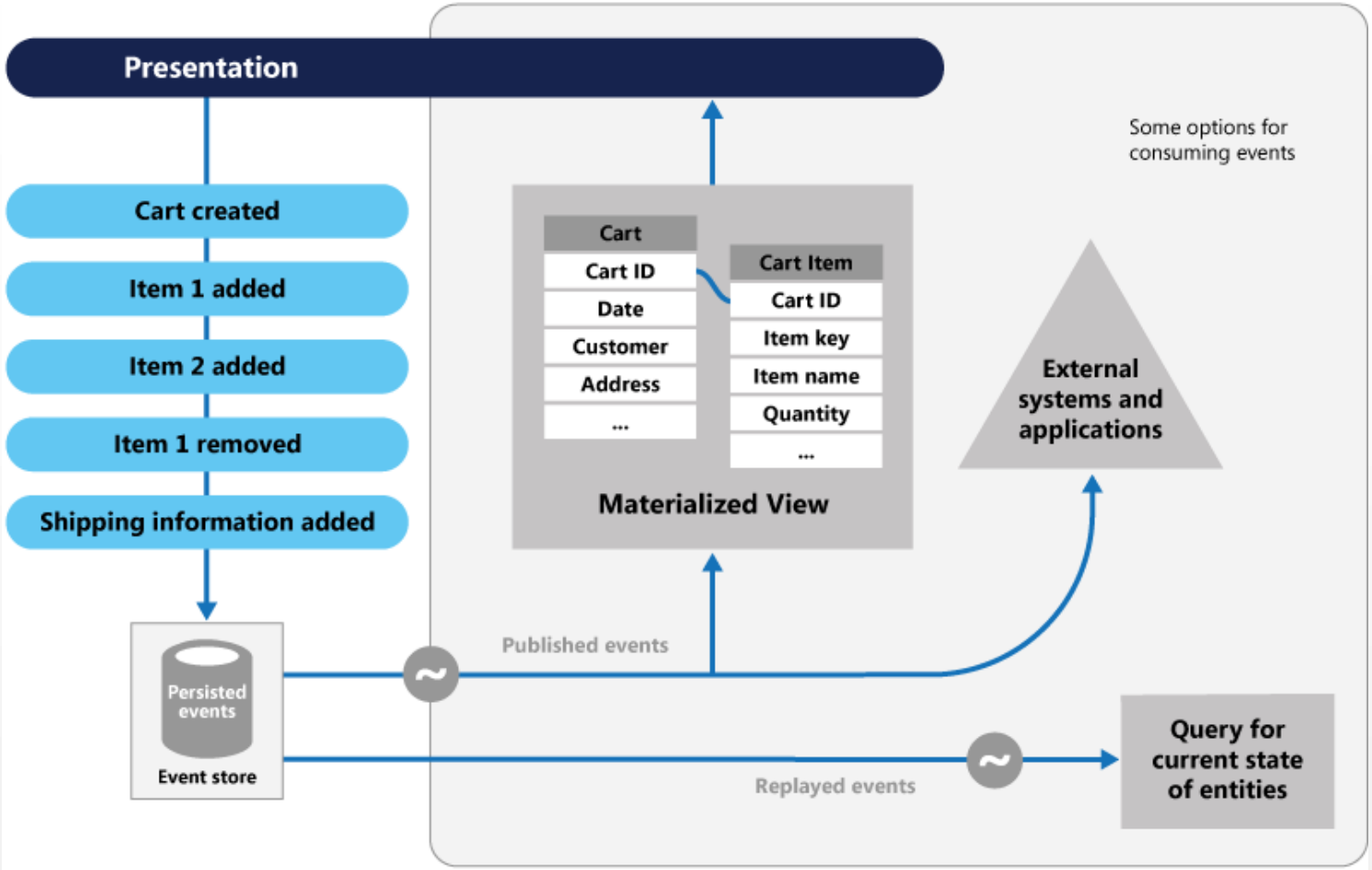


NodeMCU – ESP8266



16 GPIOs
1 ADC
32bit 80Mhz
Wi-Fi
Baixo custo

Event Sourcing



Trabalhos Correlatos

Sonoff POW

- Controla equipamentos;
- Mede consumo;
- Baixo custo;
- Invasivo.

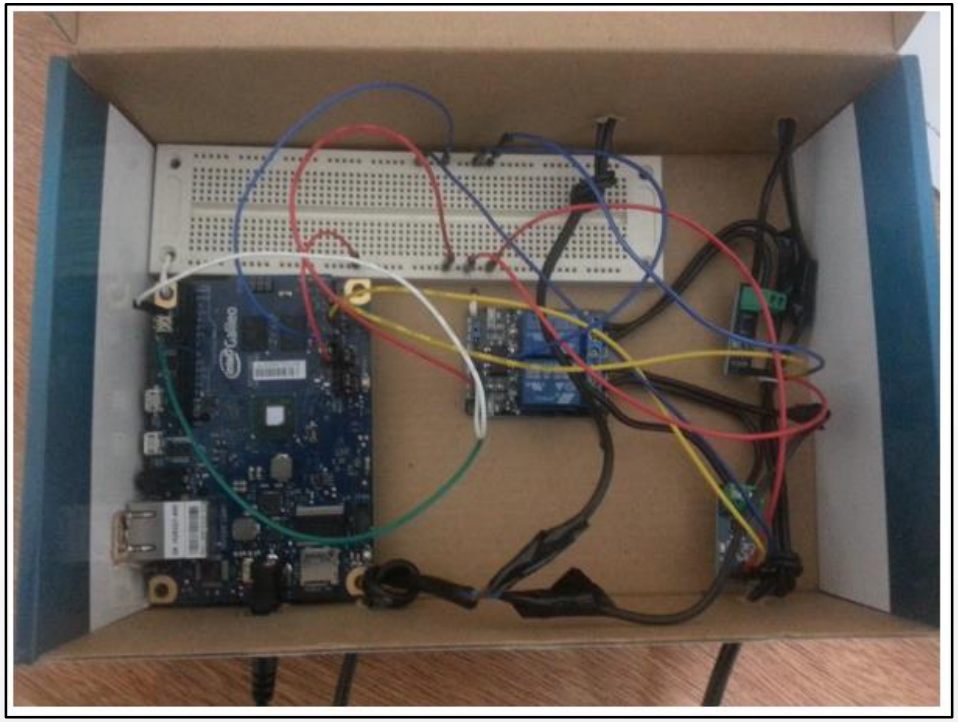
Trabalhos Correlatos

TP-LINK HS110

- Controla equipamentos;
- Indisponível no padrão nacional tomadas.

Trabalhos Correlatos

Waka (2015)



- Controla equipamentos;
- Mede consumo.

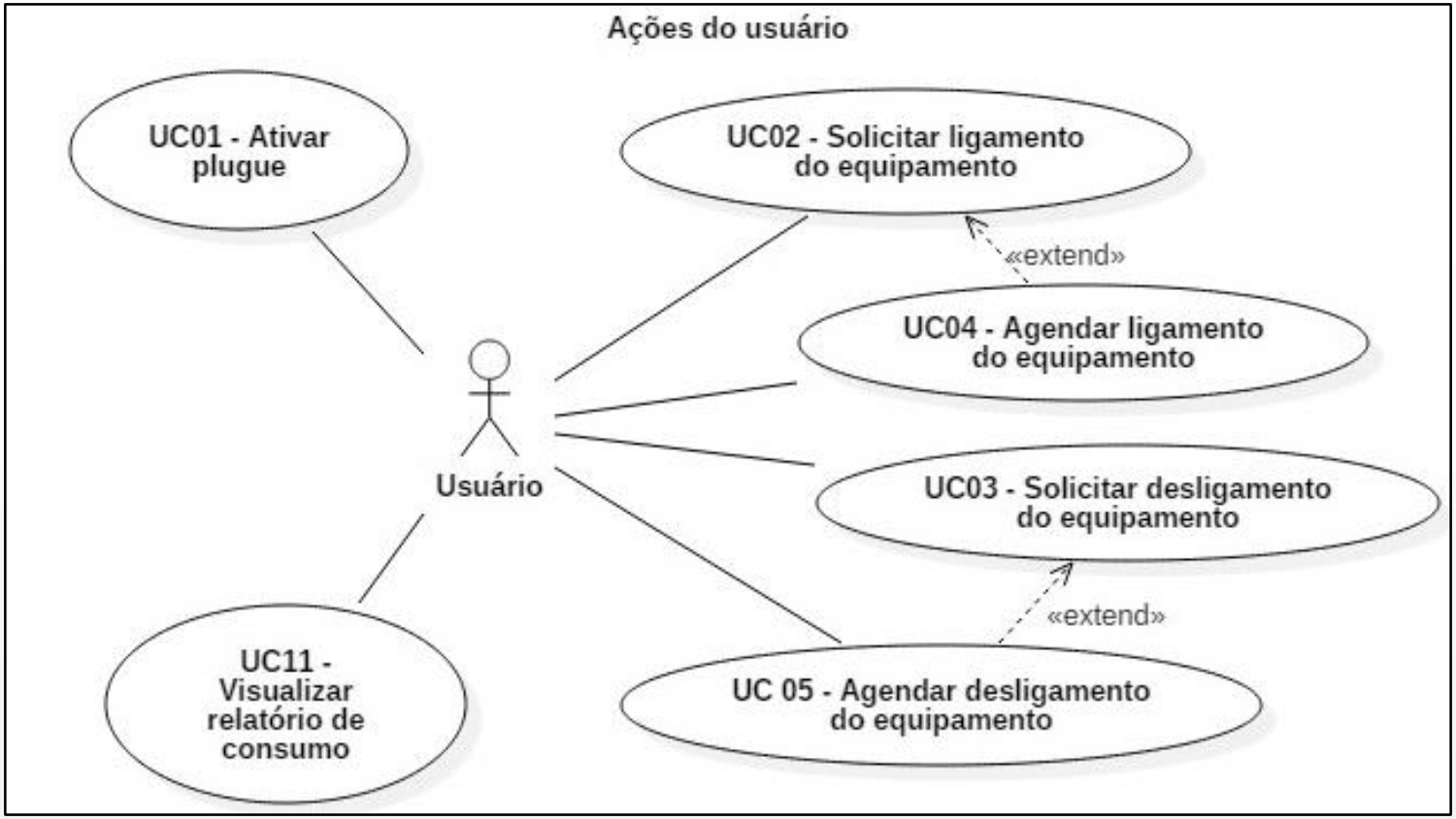
Requisitos funcionais

Requisito	Descrição
RF1	Permitir ligar/desligar um equipamento
RF2	Permitir agendar as ações de ligar/desligar
RF3	Disponibilizar relatório de consumo
RF4	Permitir gerenciar múltiplos plugues

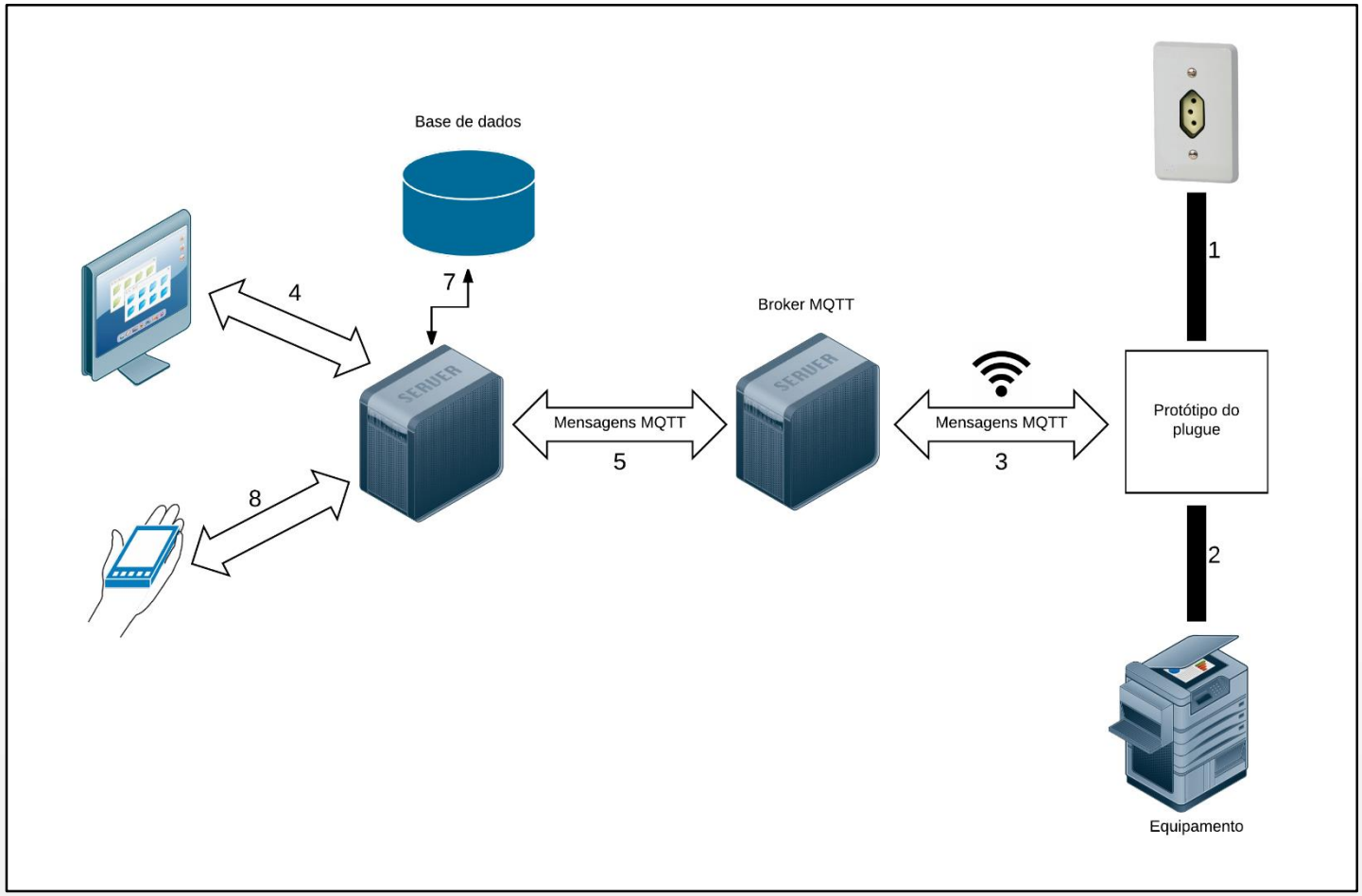
Requisitos não-funcionais

Requisito	Descrição
RNF1	Consumir menos de 1W
RNF2	Apresentar compatibilidade com tomadas no padrão nacional
RNF3	Utilizar conectividade Wi-Fi e permitir controle via aplicativo móvel
RNF4	Utilizar o SoC ESP8266
RNF5	Utilizar o protocolo MQTT
RNF6	Utilizar IDE Visual Studio 2017 com *plugin* Visual Micro para o *firmware*
RNF7	Utilizar o padrão arquitetural Event Sourcing para salvar os eventos do plugue

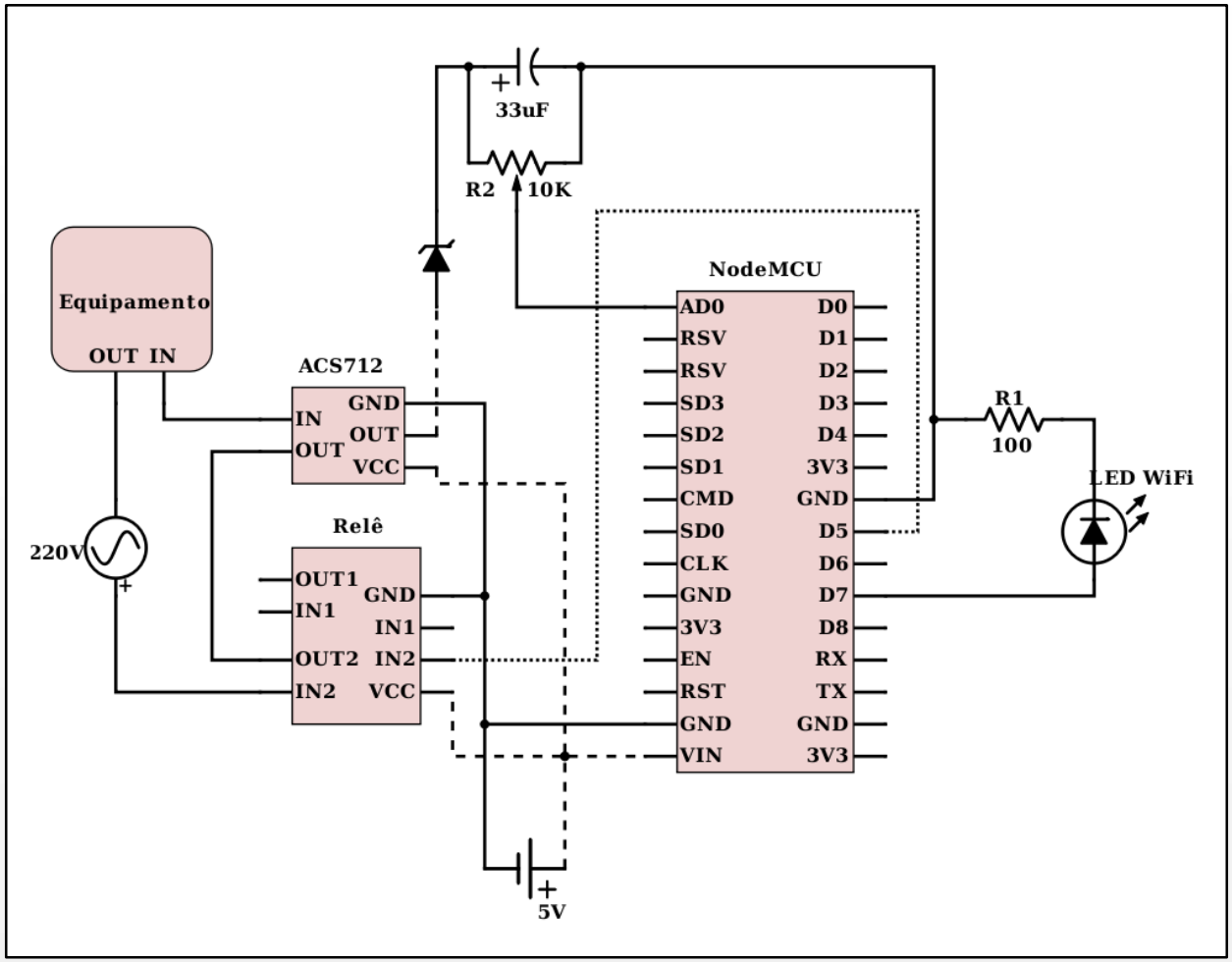
Casos de Uso


Ações do usuário
UC01 - Ativar plugue
UC02 - Solicitar ligamento do equipamento
«extend»
UC04 - Agendar ligamento do equipamento
Usuário
UC03 - Solicitar desligamento do equipamento
«extend»
UC11 - Visualizar relatório de consumo
UC 05 - Agendar desligamento do equipamento

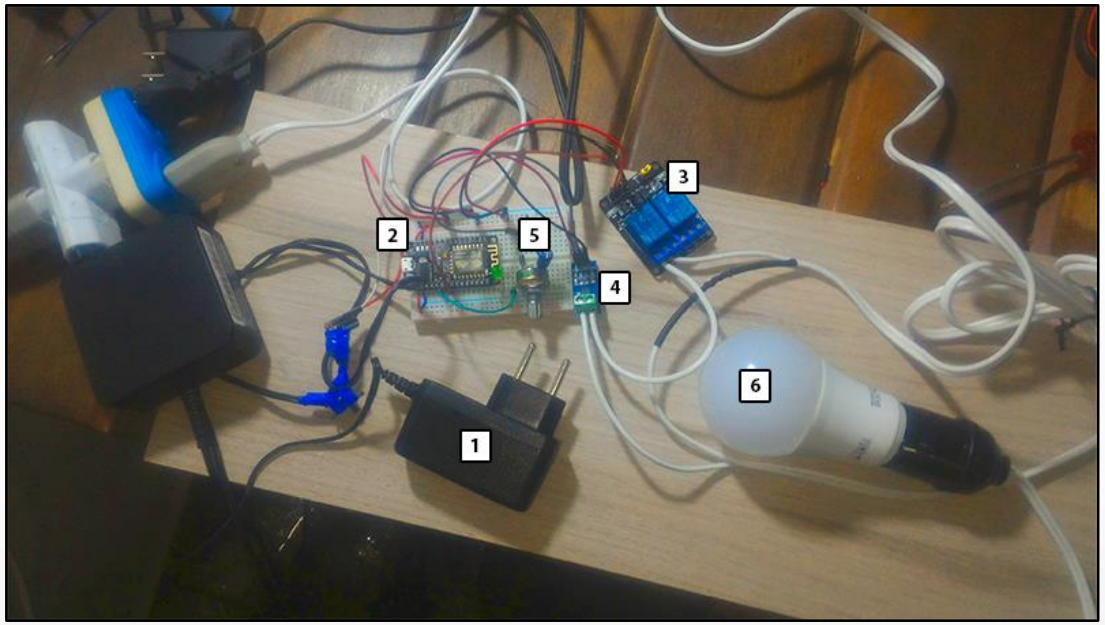
Arquitetura

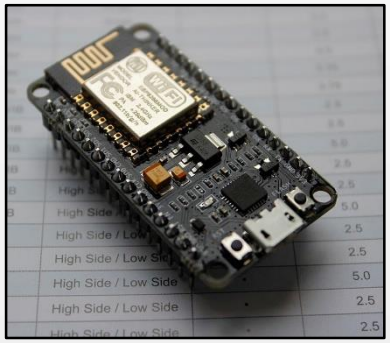


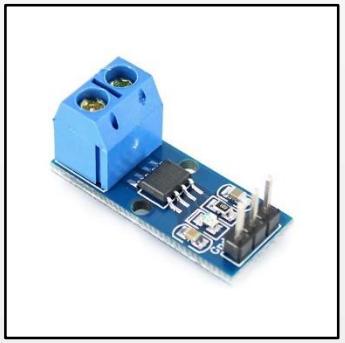
Protótipo do Hardware



Hardware

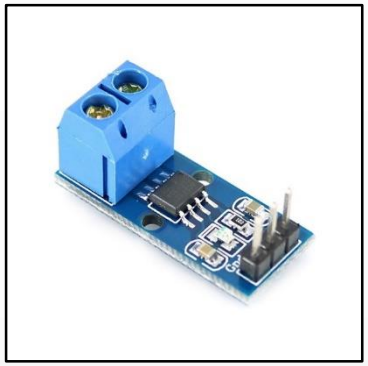




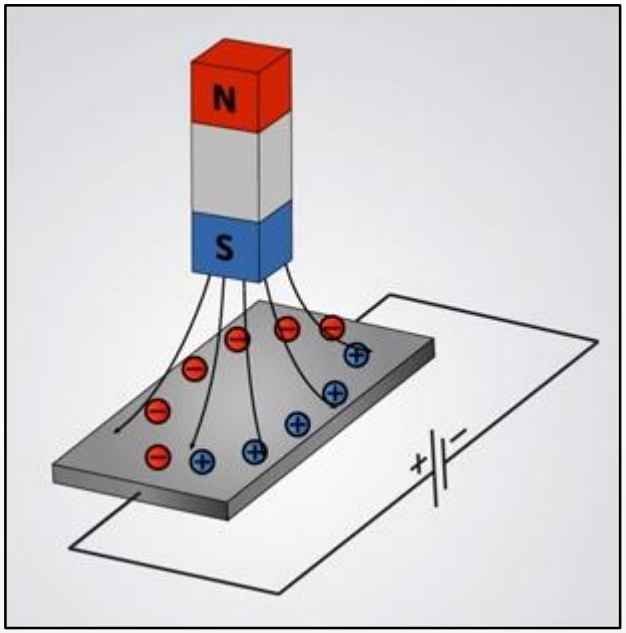


Sensor de corrente

ACS712



Efeito Hall



-30A a 30A → 0V a 5V → 0V a 3,3V * → 0 a 1023 → Cálculo de potência → Publicação

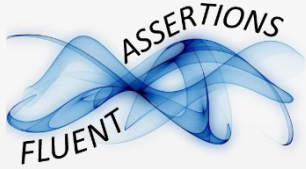
Ferramentas utilizadas

+

PostgreSQL

+

+

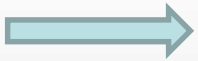


Estrutura das mensagens MQTT

ID do plugue	Divisor	Conteúdo
2c6c-54ea-485a-9e0b-c74a7a0fb334	|	turn-on
2c6c-54ea-485a-9e0b-c74a7a0fb334	|	1.5|220|330

Tabela de Eventos

mt_events
seq_id: uuid
id: uuid
stream_id: uuid
version: int
data: jsonb
type: varchar
timestamp: timestamp
tentant_id: varchar
mt_dotnet_type: varchar



```
{
    "Current": 4.64,
    "Voltage": 220,
    "ConsumptionInWatts": 1021.77
}
```

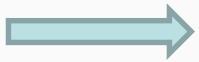
Eventos

Eventos
PlugActivated
PlugDeativated
PlugRenamed
PlugTurnedOn
PlugTurnedOn
OperationScheduled
ConsumptionReadingReceived

```
public class Plug
{
    public Guid Id { get; set; }
    public PlugState CurrentState { get; set; }
    public string Name { get; set; }
    public double LastConsumptionInWatts { get; set; }
    public bool Active { get; set; }
    public bool IsOn() => CurrentState == PlugState.On;

    public void Apply(PlugActivated activation)
    {
        Id = activation.PlugId;
        Name = activation.PlugName;
        Active = true;
    }

    public void Apply(PlugTurnedOn plugTurnedOn)
    {
        CurrentState = PlugState.On;
    }

    //...

    public void Apply(PlugRenamed plugRenamed)
    {
        Name = plugRenamed.NewName;
    }
}
```

Visões Materializadas

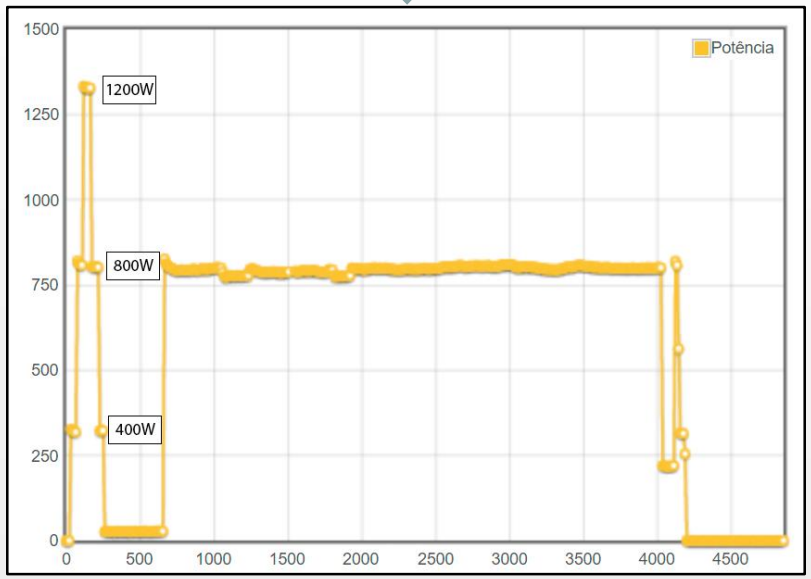
mt_doc_plug
id: uuid
data: jsonb
mt_last_modified: timestamp
mt_version: uuid
mt_dotnet_type: varchar




```
{
    "Id": "90b94d31-61df-44fe-8f6a-d2df9ebdf6ec",
    "Name": "Pinheiro de Natal",
    "Active": false,
    "CurrentState": 0,
    "LastConsumptionInWatts": 0
}
```

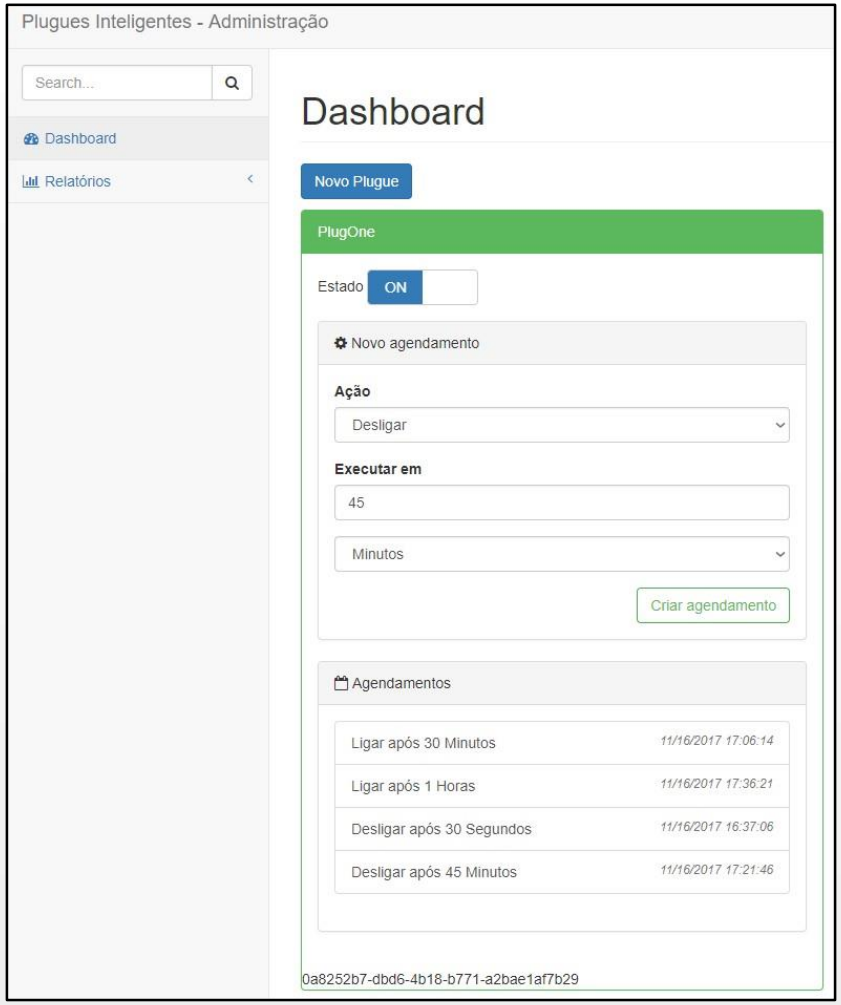
Relatório de Consumo

```
public IEnumerable<ConsumptionInTime> GetConsumptionReport(Guid plugId, DateTime? startTime = null)
{
    using (var session = _documentStore.LightweightSession())
    {
        var consumption = session.Events
            .FetchStream(plugId, timestamp:startTime)
            .Where(e =>
                e.StreamId == plugId &&
                e.Data.GetType() == typeof(ConsumptionReadingReceived))
            .OrderBy(e => e.Timestamp)
            .Select(e =>
            {
                var reading = (ConsumptionReadingReceived) e.Data;
                return new ConsumptionInTime(reading.ConsumptionInWatts, reading.Current, e.Timestamp);
            });
        return consumption;
    }
}
```

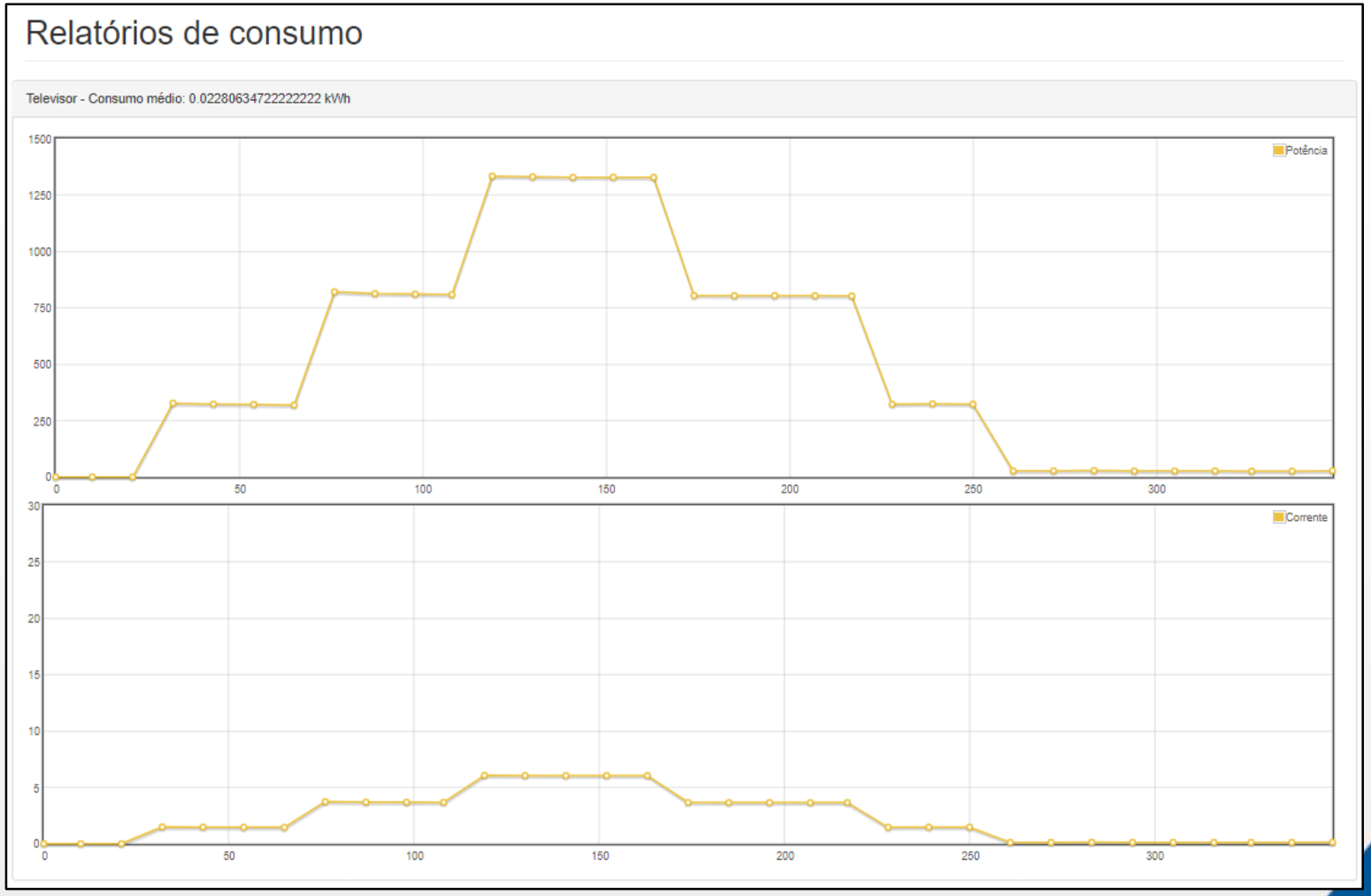


FURB
UNIVERSIDADE DE BLUMENAU

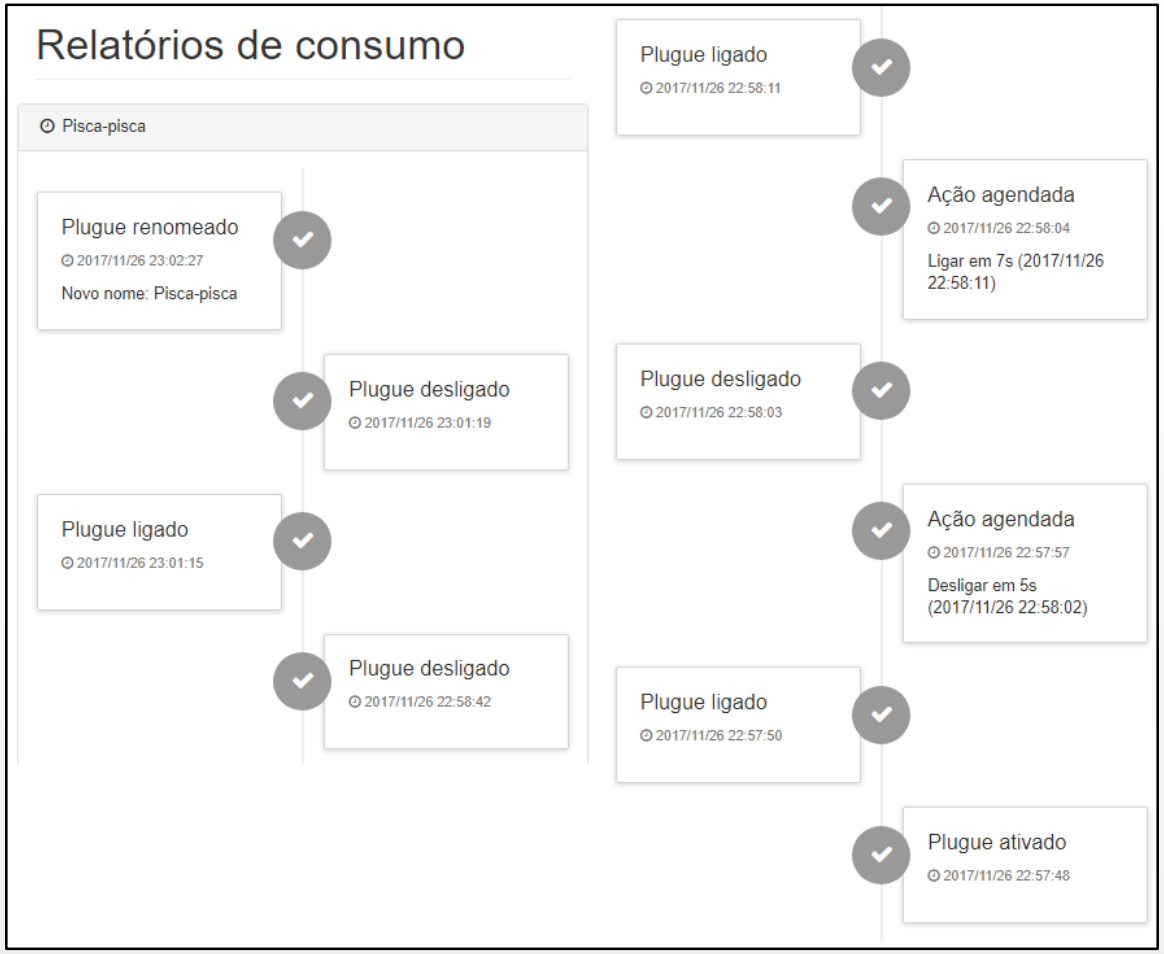
Operacionalidade da Implementação



Operacionalidade da Implementação



Operacionalidade da Implementação



Resultados e Discussões

Características mais relevantes	Trabalhos correlatos			Maas (2017)
	Sonoff POW	TP-Link HS110	Waka (2015)	Plugue Inteligente
Portal cativo para informar credenciais de rede	Sim	Sim	Não	Não
Interface de usuário	Android/iOS	Android/iOS	Navegador Web	Navegador Web
Histórico de consumo	Sim	Não	Sim	Sim
Permite agendamento	Sim	Sim	Não	Sim
Histórico de atividades do plugue/tomada	Sim	Sim	Não	Sim
Desenho intrusivo	Sim	Não	Não	Não
Aplicação controla múltiplos dispositivos	Sim	Sim	Não	Sim
Custo	US$ 10,50 (R$35,00)	US$ 39,99 (R$ 133,00)	R$ 443,90 (protótipo)	R$ 148,39 (protótipo)

Conclusões

- Um dos objetivos específicos foi atendido parcialmente, com o portal WEB substituindo o aplicativo móvel previsto;
- O requisito RNF1 não foi atendido;
- O protótipo apresentou diferencias em relação aos correlatos, como históricos de consumo e de atividade, além de um desenho não-intrusivo;
- O uso do padrão *Event Sourcing* e da biblioteca Marten simplificaram o processo de desenvolvimento.

Sugestões de continuidade - I

- Adicionar um portal cativo para informar credenciais de rede;
- Empregar técnicas de aprendizado de máquina, permitindo ao plugue adaptar-se à rotina do usuário;
- Acrescentar novas formas de agendamento;
- Persistir os agendamentos na EPROM do ESP8266;
- Reimplementar a interface de usuário para dispositivos móveis, como um Progressive Web App ou aplicativo nativo.

Sugestões de continuidade - II

- Reduzir o consumo do plugue, implementado estratégias de hibernação do ESP8266;
- Utilizar a variante do ACS712 que trabalha na faixa de -20A a 20A;
- Utilizar um sensor de tensão, visando melhorar a acuracidade do cálculo de potência.

Sugestões de continuidade - III

- Criar um HUB para centralizar o controle de múltiplos plugues;
- Implementar recursos de segurança na comunicação com o *broker* MQTT;
- Implementar uma interface de harware para conectar módulos de sensores ao plugue, de modo que possam conferir novos comportamentos condicionados por dados sensoriais.

Demonstração