APLICAÇÃO PARA CONTROLE DE FLUXO E MENSAGERIA ENTRE DISPOSITIVOS IOT

Aluno: Silvio Greuel

Orientador: Dalton Solano dos Reis

FURB
UNIVERSIDADE DE BLUMENAU

Roteiro

- Introdução
- Objetivos
- Fundamentação teórica
- Trabalhos correlatos
- Especificação
- Implementação
- Operacionalidade
- Resultados e discussões
- Conclusões
- Sugestões

Introdução

Surgiu de algumas necessidades:

- Gestão dos fluxos de mensagens entre dispositivos IoT
- Ser centralizado
- Não precisar ter acesso físico
- Ser multiplataforma
- Código aberto

Objetivos

- Realizar a gestão dos fluxos de mensageria de dispositivos IoT de modo centralizado sem a necessidade de efetuar alterações no firmware
- Ser multiplataforma, explorando APIs nativas dos dispositivos
- Explorar o uso da aplicação utilizando uma bancada de homologação, contendo sensores e atuadores
- Apresentar soluções para casos de uso no ambiente agrário
- Apresentar soluções para casos de uso no ambiente industrial

Fundamentação Teórica
Internet das Coisas (IoT)

- Objetos acessíveis na rede
- Dados sobre o ambiente físico – sensores
- Informação – processamento
- Ações sobre o ambiente físico – atuadores
- Expansão da internet

Fundamentação Teórica
Progressive Web App (PWA)

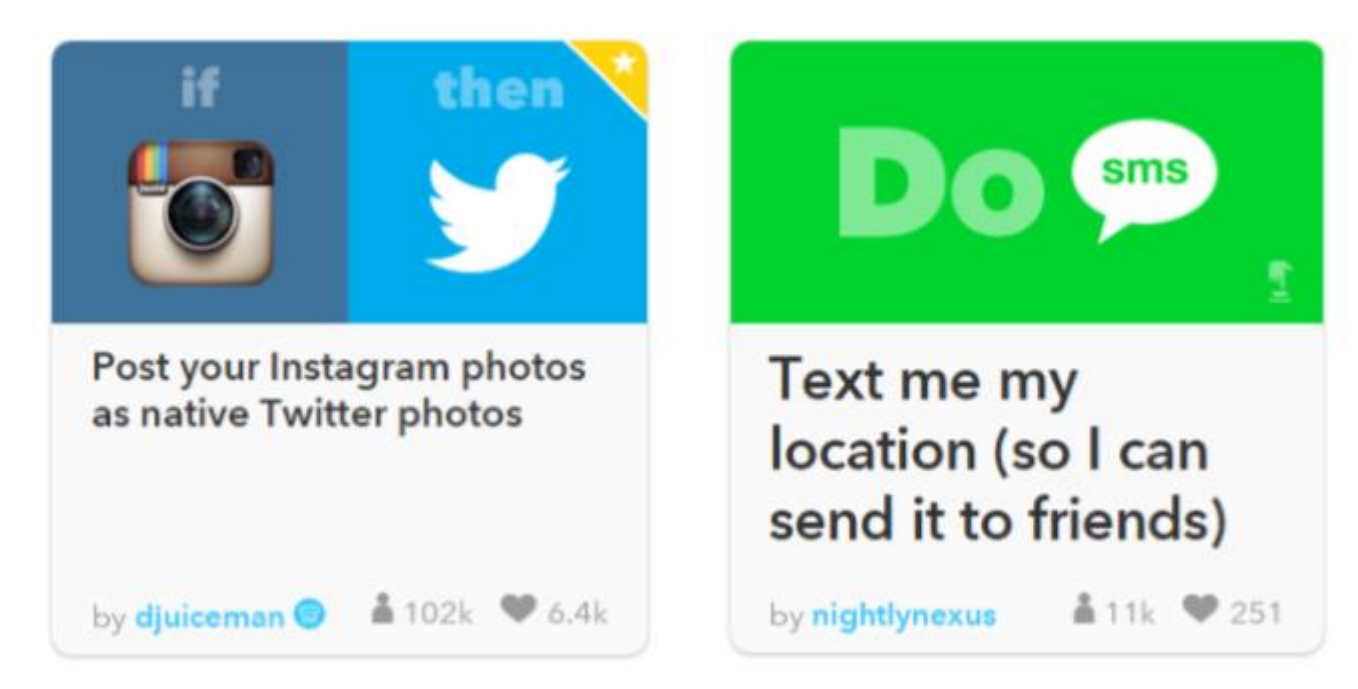
- Aplicações web com capacidades similares a aplicações nativa
- Progressiva - qualquer navegador
- Responsiva - qualquer dispositivo
- Independência da rede - acesso off-line
- Detectável - o SO reconhece a aplicação
- Instalável - o SO permite sua instalação
- Compartilhável - acessível via url

Trabalhos Correlatos

- Vorapojpisut (2015)
  - Aborda definição do controle de fluxos
  - Define gatilhos
  - Define ações
- ADLINK (2017)
  - Aborda protocolos para IoT
  - Levanta características dos protocolos
- Fransson e Driaguine (2017)
  - Aborda aplicações mutliplataforma
  - Compara aplicações nativas vs pwa

Trabalhos Correlatos Vorapojpisut (2015)

- Analisa o modelo “Se Isso, Então Aquilo”



Trabalhos Correlatos Vorapojpisut (2015)

Define uma arquitetura de gatilho e ação

- Gatilhos – se isso
  - Reagem a eventos de aplicações externas
  - Executam ações
- Ações – então aquilo
  - Realizam processamento
  - Interagem com aplicações externas

Trabalhos Correlatos

ADLINK (2017).

Levanta e compara características entre protocolos IoT

- MQTT
  - Arquitetura centralizada
  - Roteamento de mensagens via tópicos
- AMQP
  - Arquitetura centralizada
  - Roteamento de mensagens via roteadores
  - Interoperabilidade
- REST (HTTP)
  - Arquitetura cliente/servidor

Trabalhos Correlatos

Fransson e Driaguine (2017).

Analisa a viabilidade de aplicações PWA

- Comparou a performance entre PWAs e Aplicações Nativas
  - Não foi notado grande perda de performance na renderização
  - Não foi notado grande perda de performance no acesso a APIs nativas (câmera)

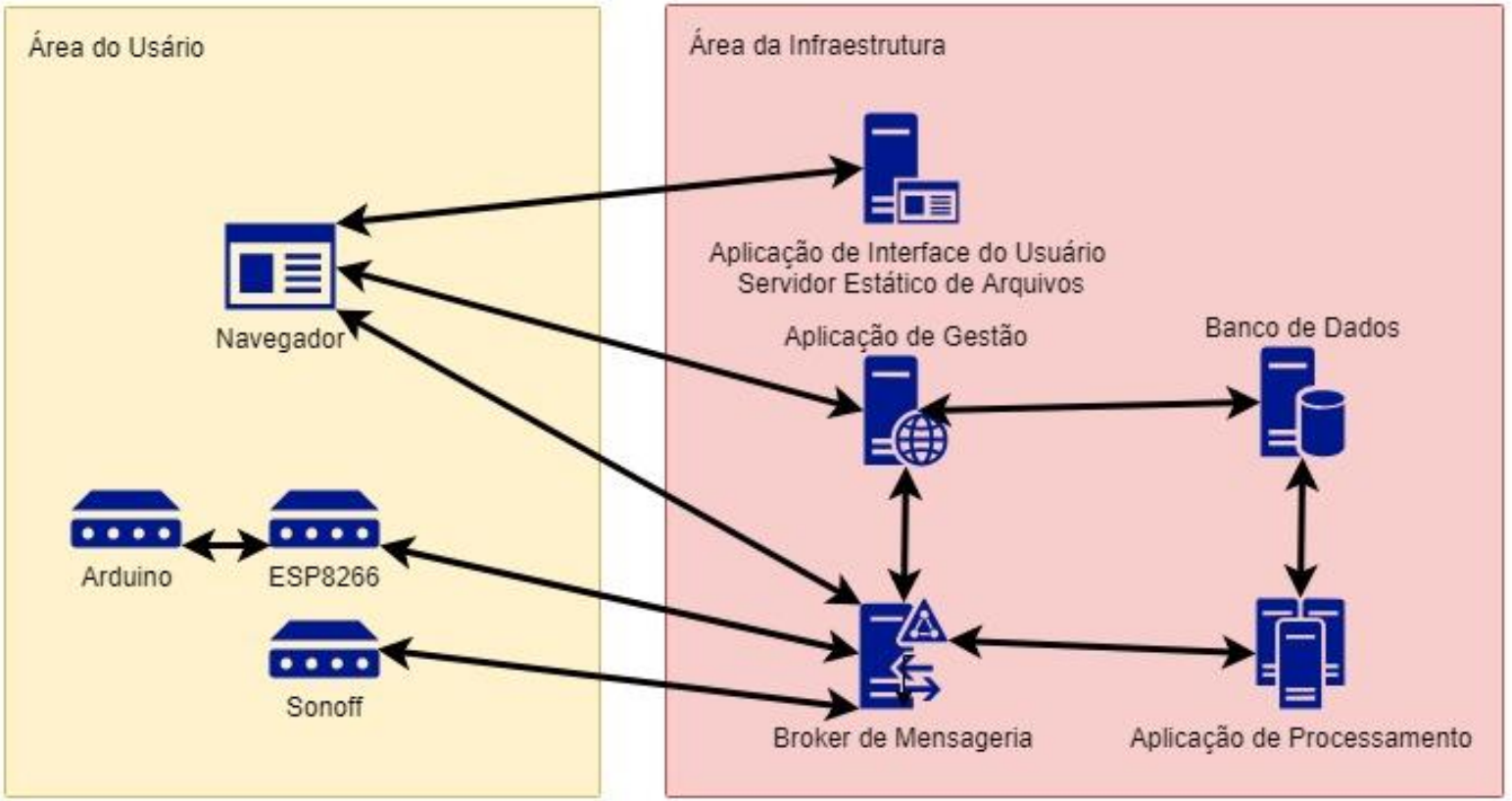
Trabalhos Correlatos

Fransson e Driaguine (2017).

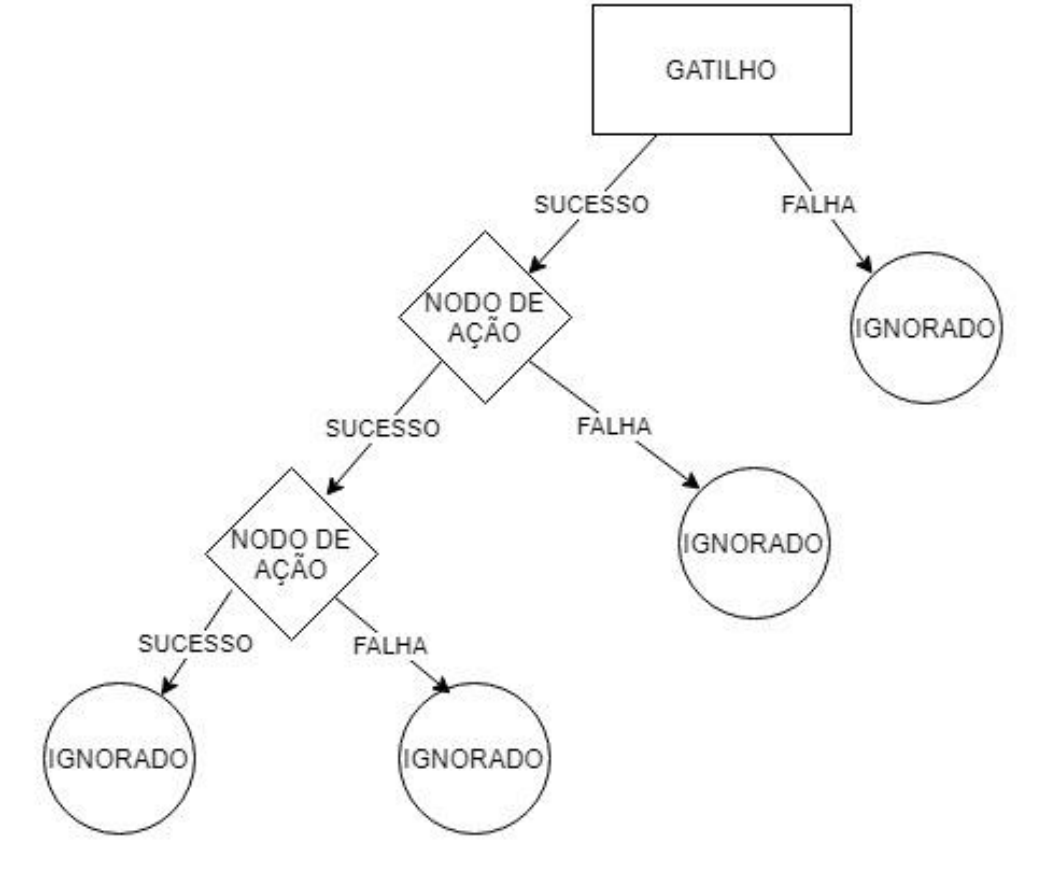
Apresenta algumas APIs fornecidas pelos navegadores

- Push messages
  - Capacidade de enviar e receber mensagens nativas
- Bluetooth
  - Capacidade de mandar mensagens via bluetooth
- Off-line mode
  - Capacidade de estar disponível off-line
- Vibration
  - Capacidade de fazer o dispositivo vibrar
- Full screen
  - Capacidade de ser apresentado sem a barra de navegação

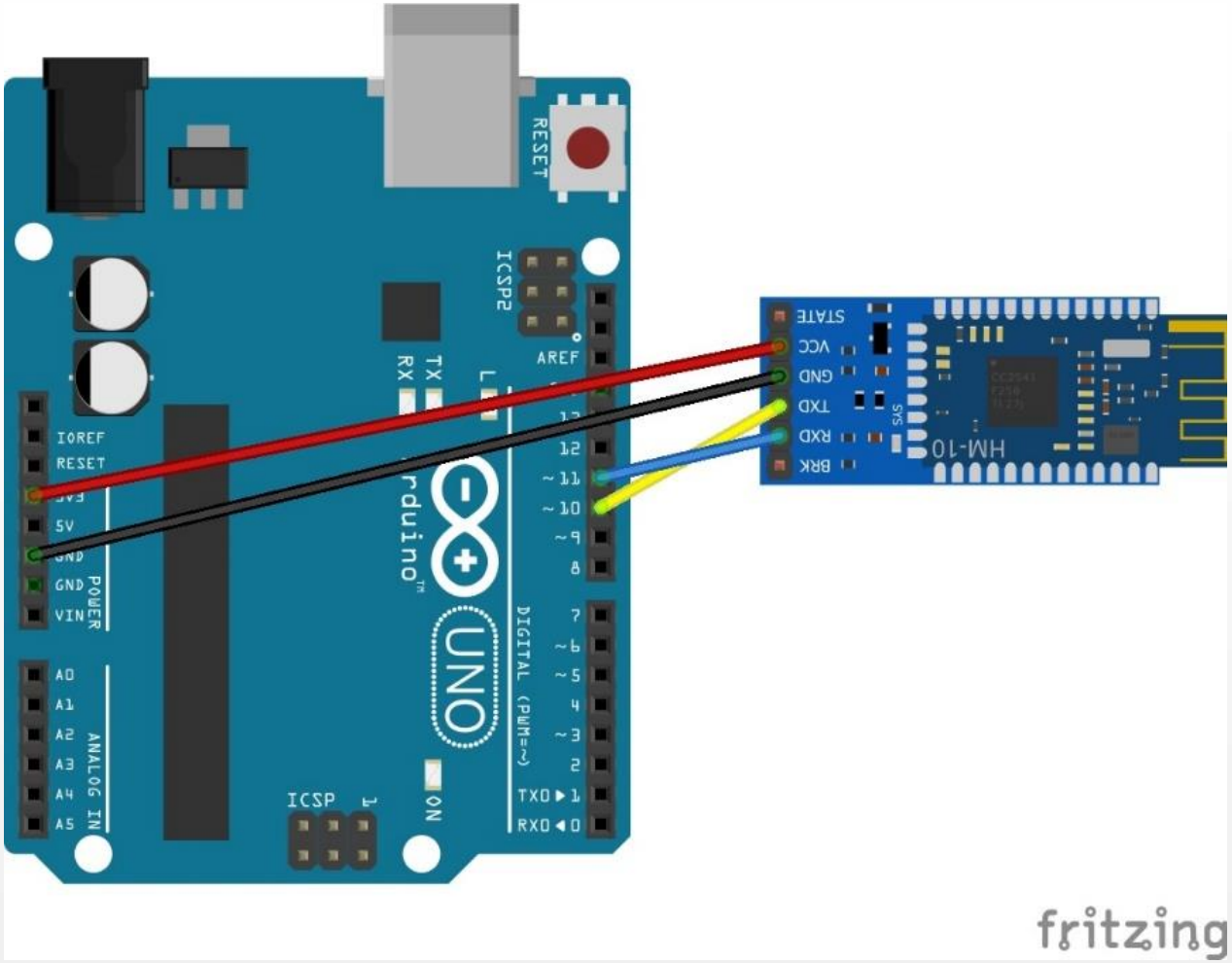
Especificação
Arquitetura



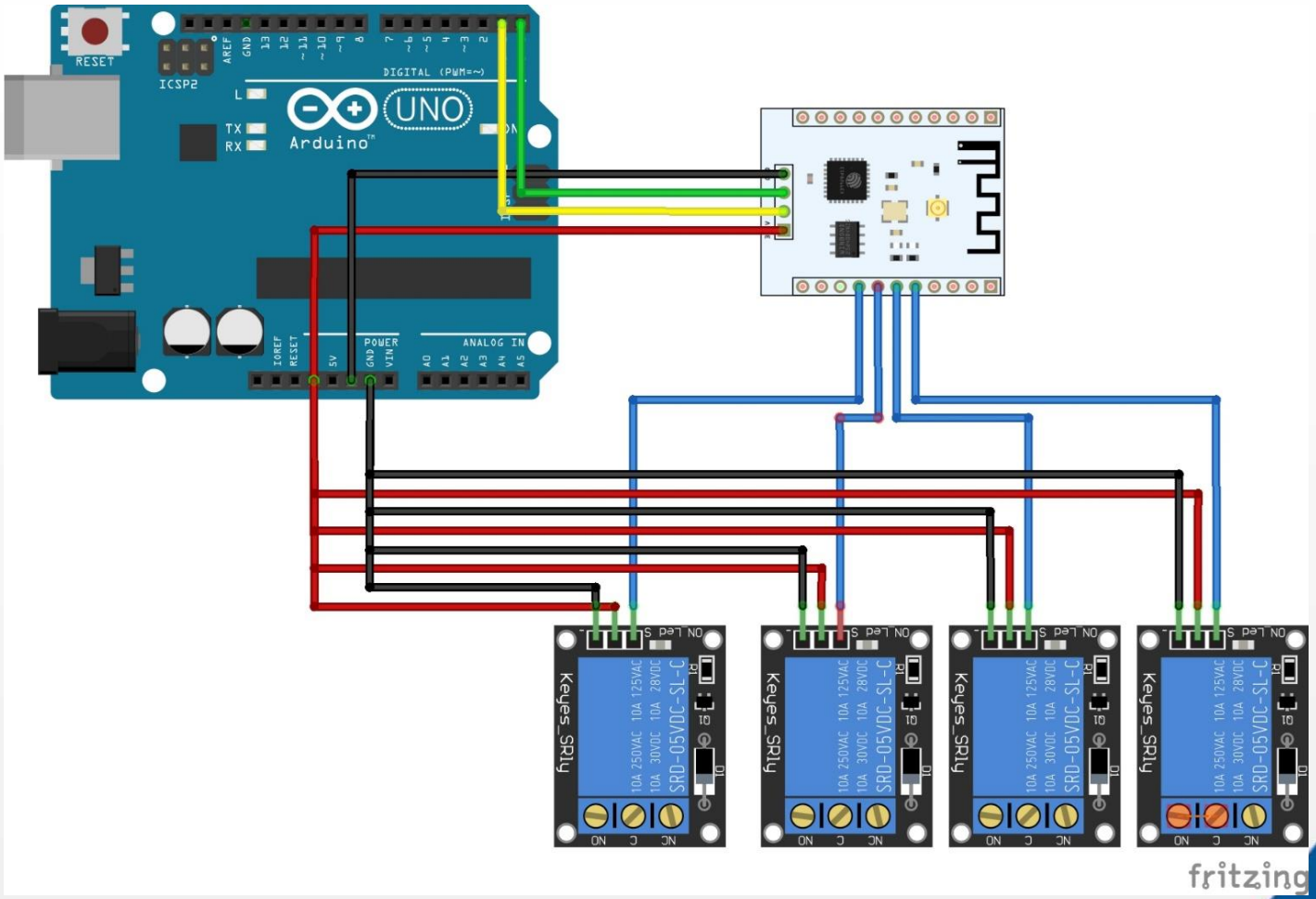
Especificação
Fluxo



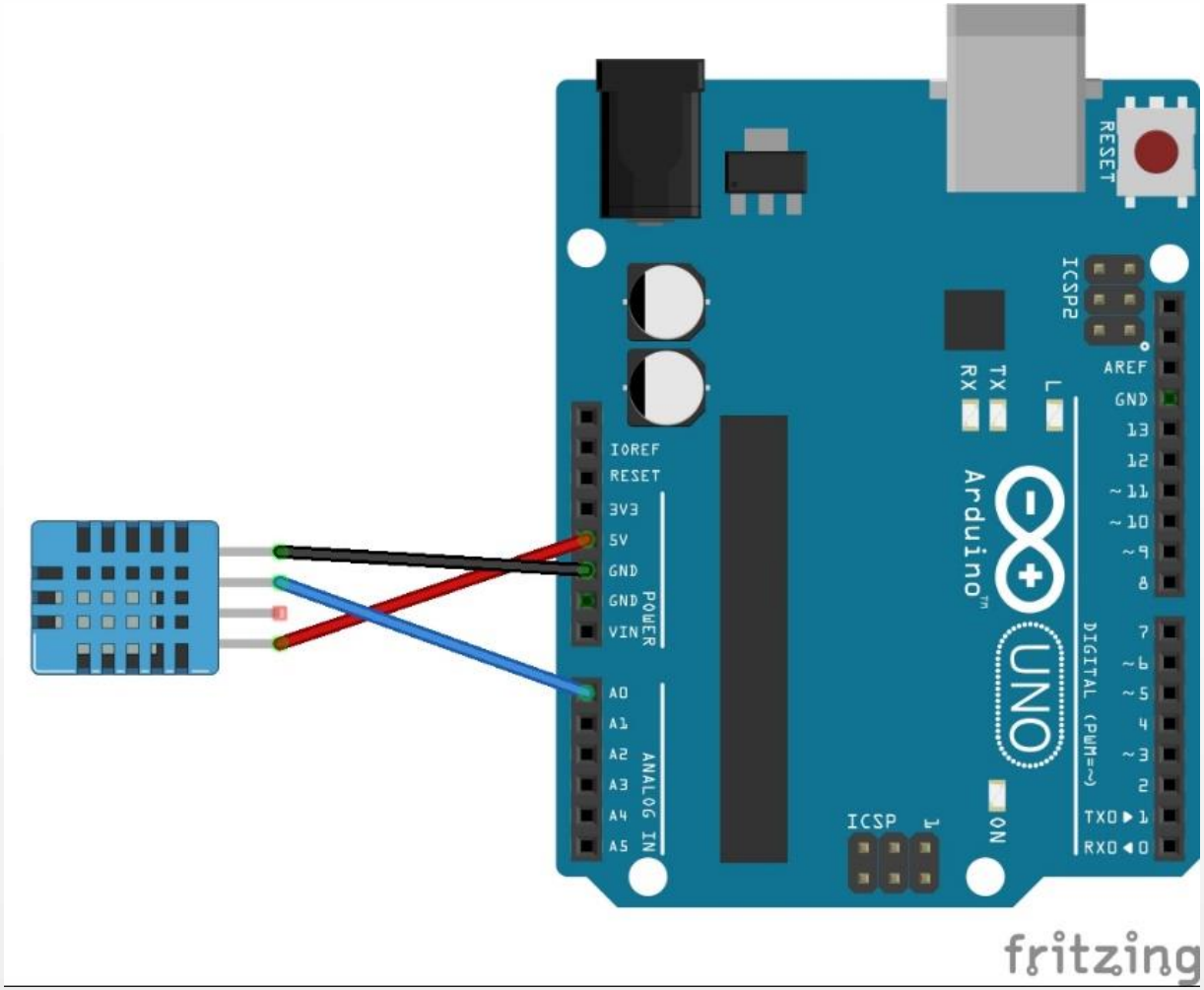
Especificação
Circuito Arduino/HM10 (BLE)



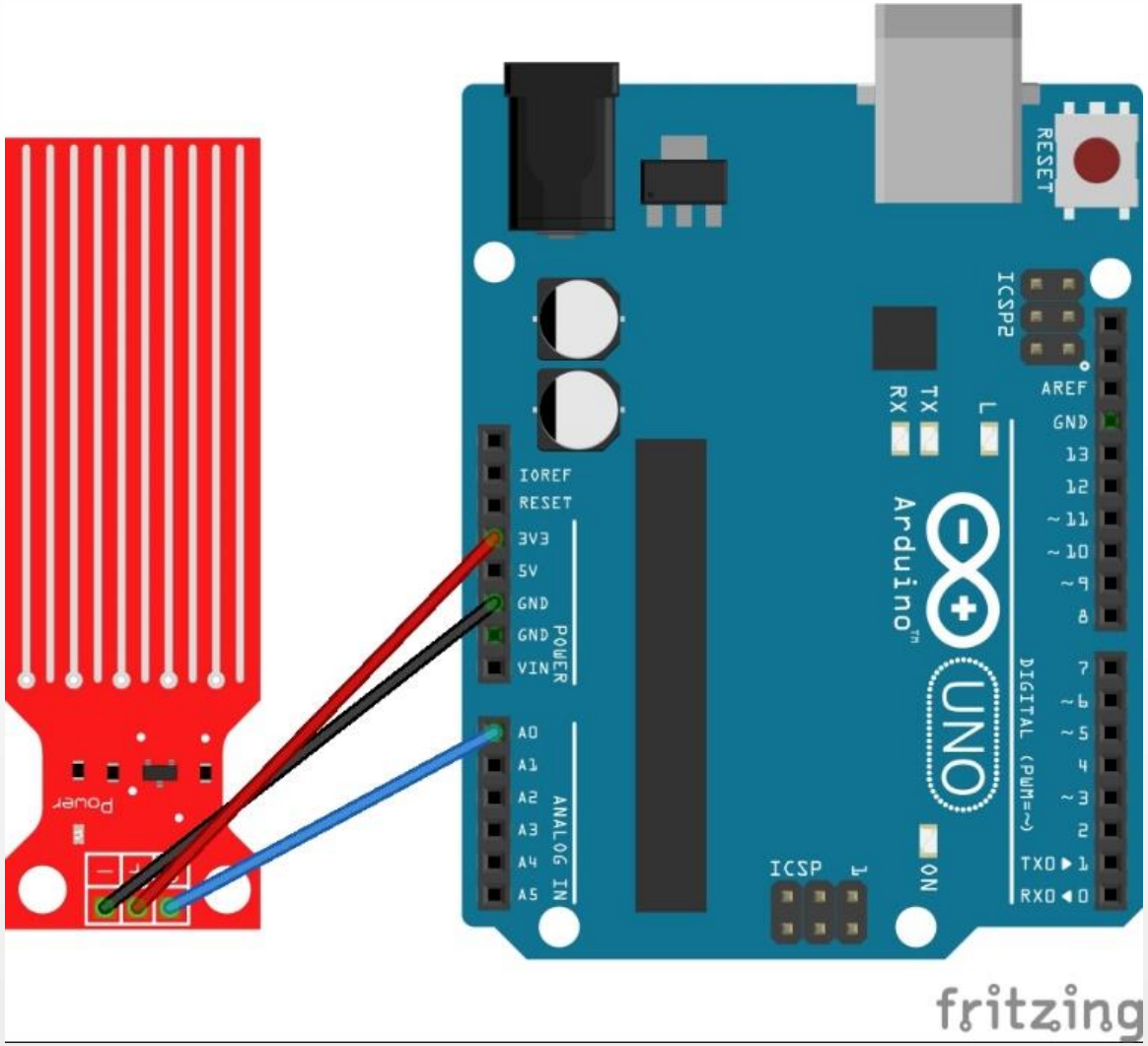
Especificação
Circuito Arduino/ESP8266



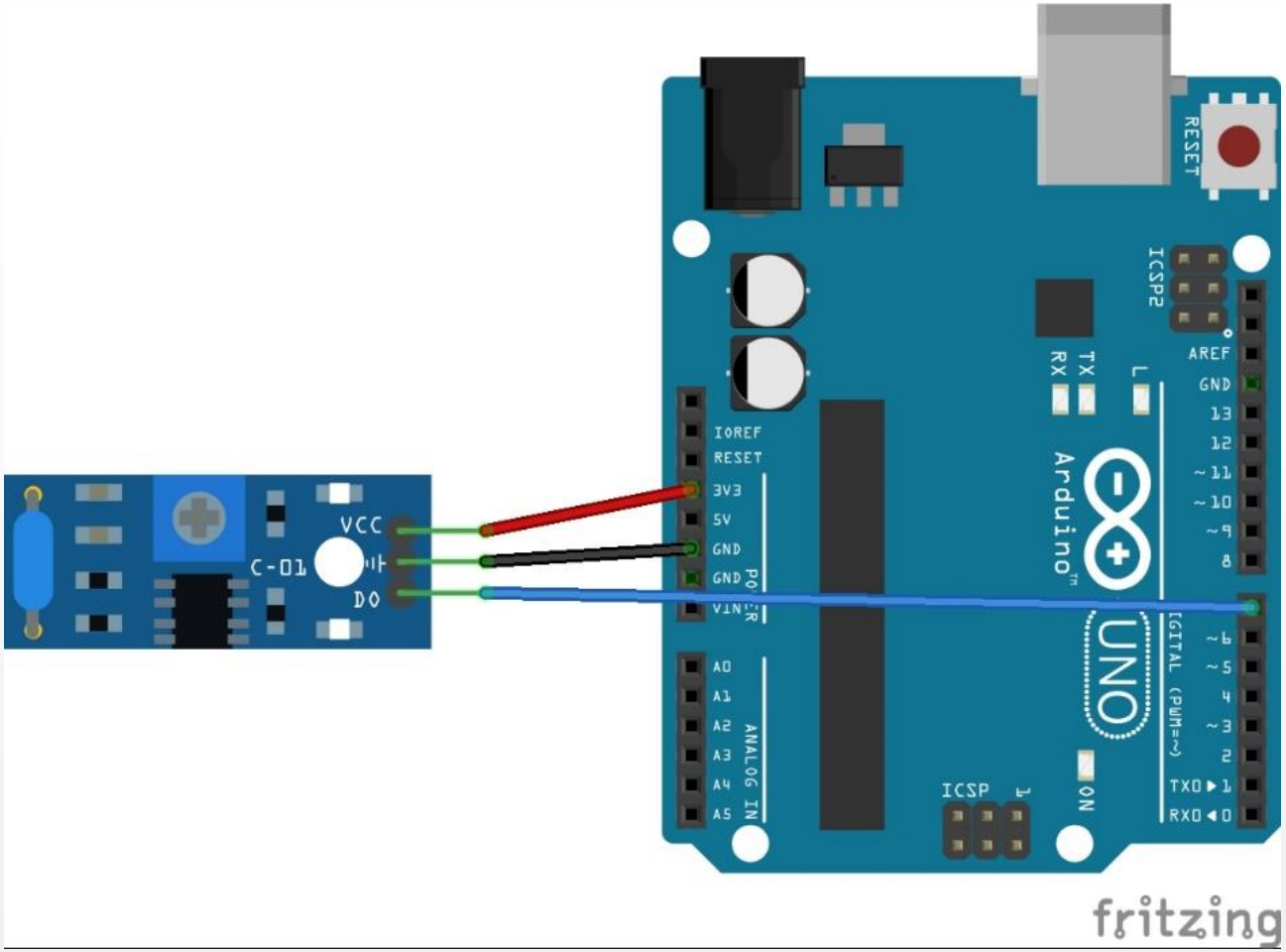
Especificação
Circuito Arduino/DHT11



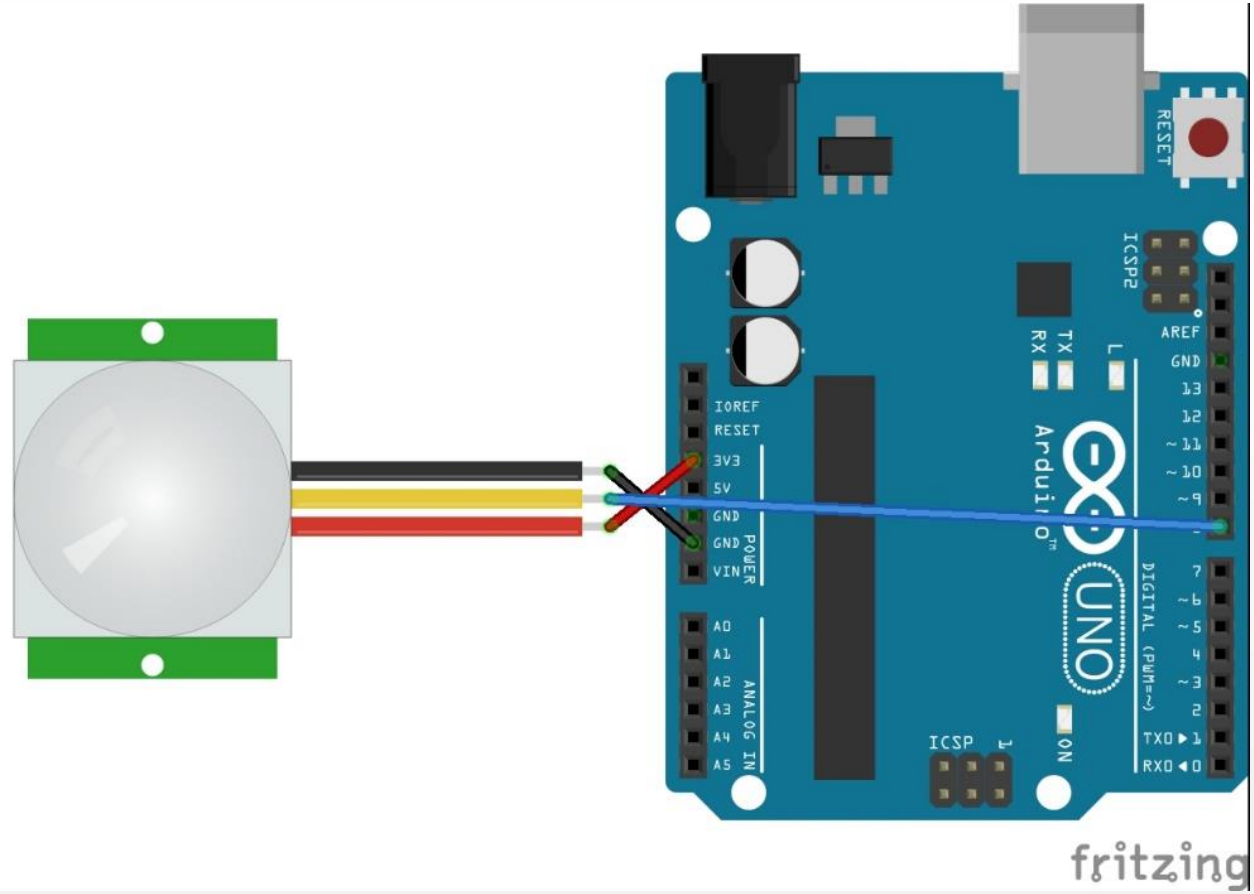
Especificação
Circuito Arduino/Nível Água



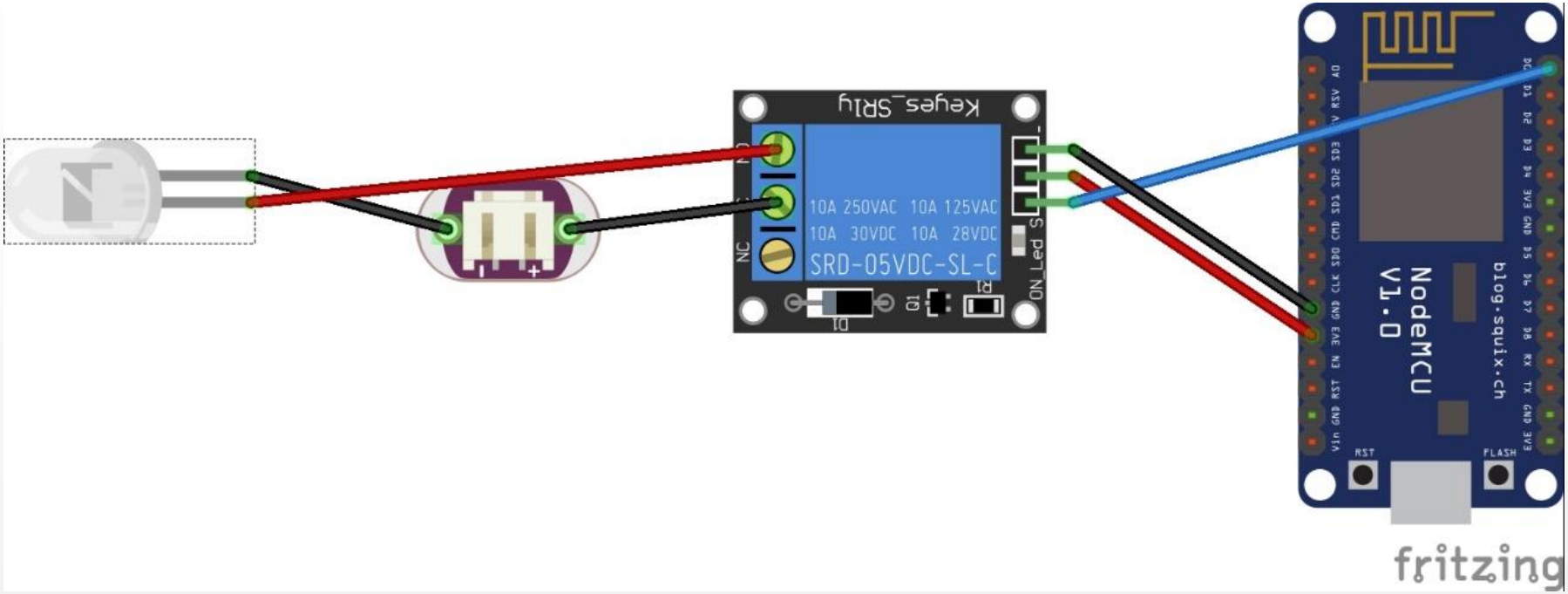
Especificação
Circuito Arduino/SW-410



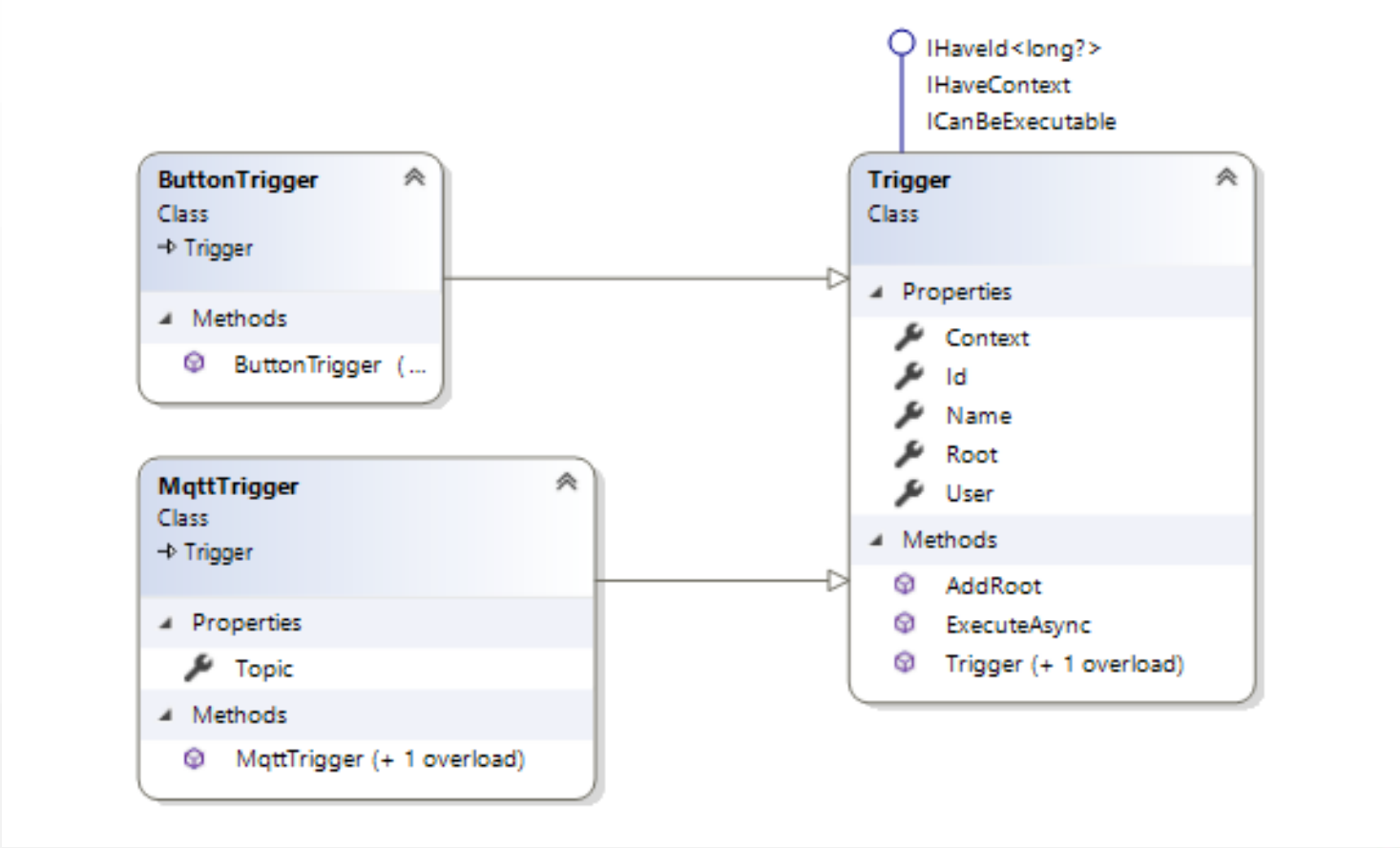
Especificação
Circuito Arduino/PIR



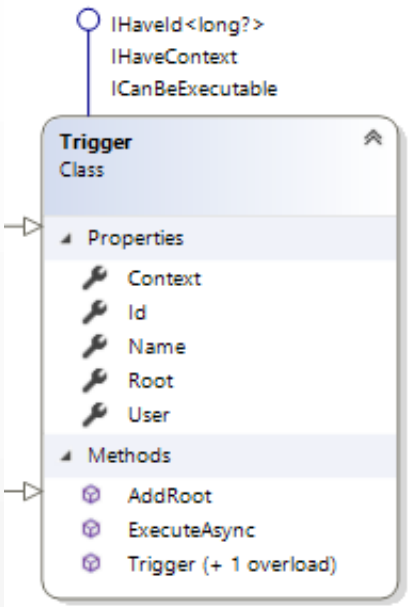
Especificação
Circuito Exemplo/Sonoff



Implementação
Domínio Gatilho

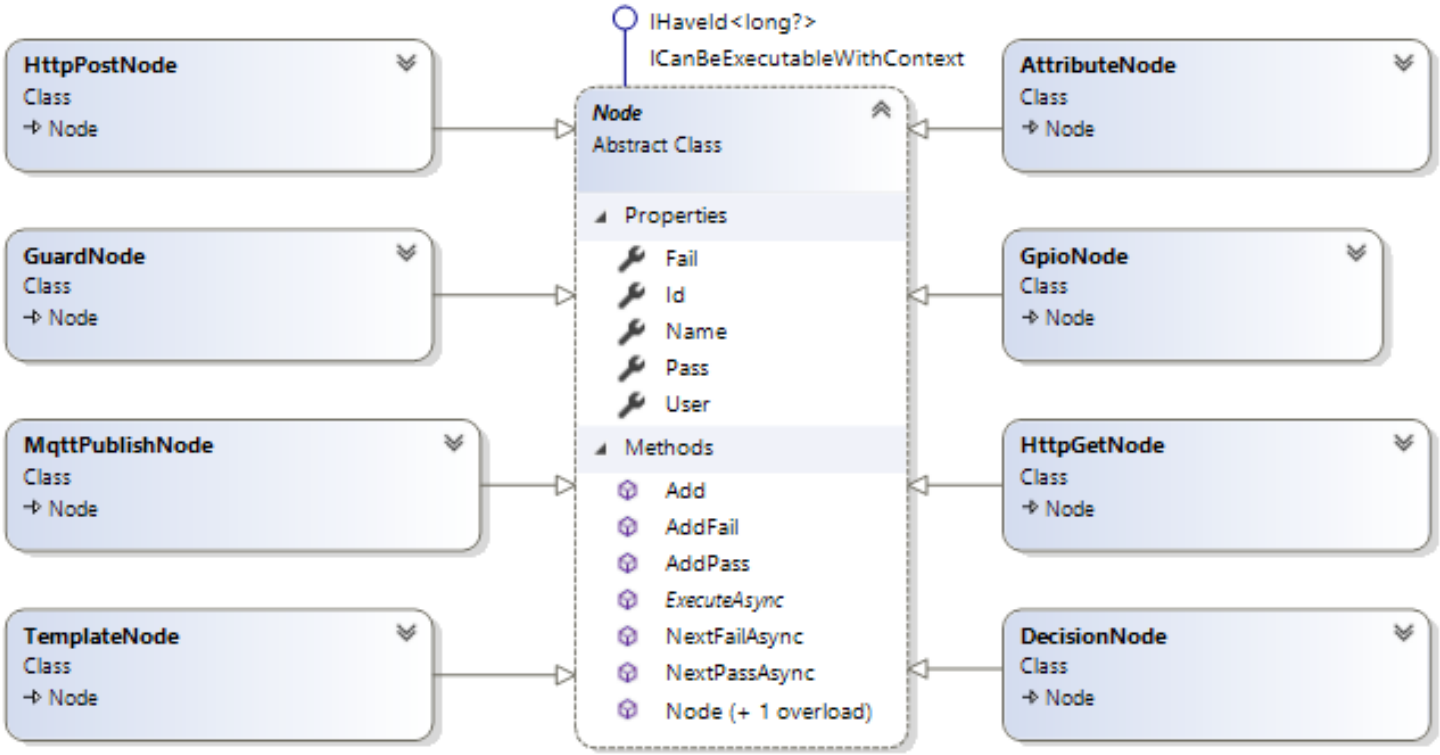


Implementação Domínio Gatilho

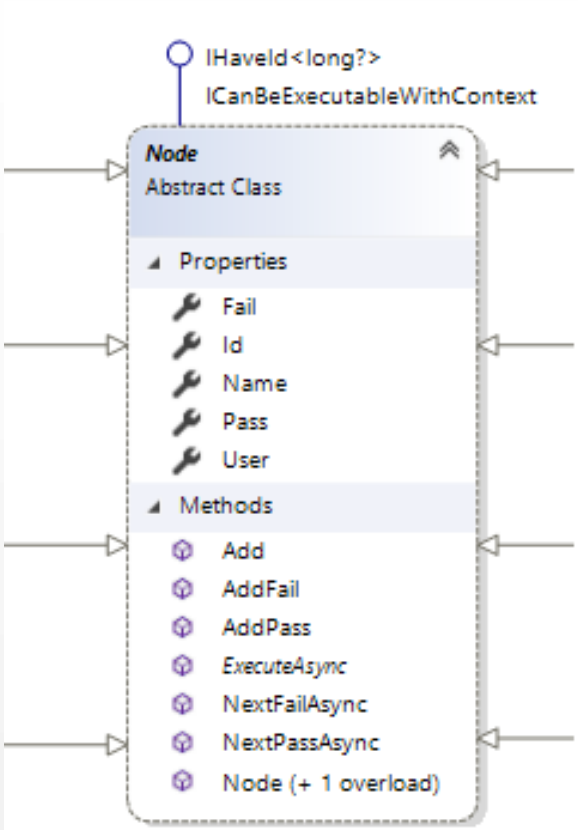



```
public class Trigger : IHaveId<long?>, IHaveContext, ICanBeExecutable
{
    public long? Id { get; set; }
    public string Name { get; set; }
    public virtual Node Root { get; set; }
    public virtual User User { get; set; }
    public virtual IDictionary<string, dynamic> Context { get; set; }

    public Trigger() { }

    public Trigger(User user, string name)
    {
        this.Name = name;
        this.User = user;
    }

    public async Task ExecuteAsync()
    {
        if (Root == null) { return; }
        await Root.ExecuteAsync(Context);
    }

    public void AddRoot(Node root)
    {
        Root = root;
    }
}
```

Implementação
Domínio Nodo

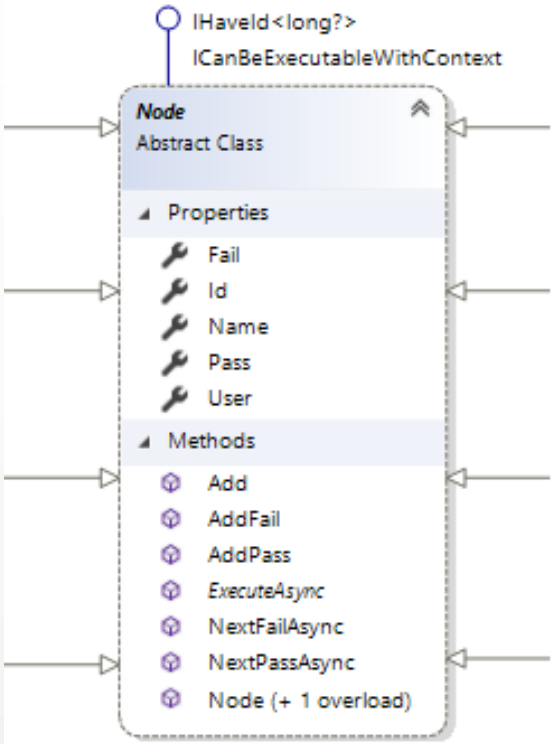


Implementação Domínio Nodo




```
public long? Id { get; set; }
public string Name { get; set; }
public virtual User User { get; set; }
public virtual Node Pass { get; set; }
public virtual Node Fail { get; set; }

public Node() { }

public Node(User user, string name)
{
    this.Name = name;
    this.User = user;
}

public void AddPass(Node pass)
{
    this.Pass = pass;
}

public void AddFail(Node fail)
{
    this.Fail = fail;
}

public void Add(Node pass, Node fail)
{
    AddPass(pass);
    AddFail(fail);
}
```

Implementação Domínio Nodo




```
public async Task NextPassAsync(IDictionary<string, dynamic> context)
{
    try
    {
        if (Pass == null) return;
        await Pass.ExecuteAsync(context);
    }
    catch (Exception e)
    {
        Log.Logger.Information($"EXCEPTION {e.Message}");
        context["exception"] = e.Message;
    }
}

public async Task NextFailAsync(IDictionary<string, dynamic> context)
{
    try
    {
        if (Fail == null) return;
        await Fail.ExecuteAsync(context);
    }
    catch (Exception e)
    {
        Log.Logger.Information($"EXCEPTION {e.Message}");
        context["exception"] = e.Message;
    }
}

public abstract Task ExecuteAsync(IDictionary<string, dynamic> context);
```

Implementação Domínio Nodo

```
public class HttpGetNode : Node
{
    public string Url { get; set; }
    public string Field { get; set; }

    public HttpGetNode() { }

    public HttpGetNode(User user, string name, string url, string field) : base(user, name)
    {
        this.Url = url;
        this.Field = field;
    }

    public override async Task ExecuteAsync(IDictionary<string, dynamic> context)
    {
        var url = string.Format(new TemplateFormatProvider(), Url, context);
        var http = new HttpClient();
        var request = new HttpRequestMessage(HttpMethod.Get, url);
        var response = await http.SendAsync(request);
        context[Field] = await response.Content.ReadAsStringAsync();

        Log.Logger.Information($"GET {url} = {Field}:{context[Field]}");
        await NextPassAsync(context);
    }
}
```

Implementação
PWA App Manifest

```
{
  "name": "asdf-flow-pwa",
  "short_name": "asdf-flow-pwa",
  "icons": [
    {
      "src": "/img/icons/android-chrome-192x192.png",
      "sizes": "192x192",
      "type": "image/png"
    },
    {
      "src": "/img/icons/android-chrome-512x512.png",
      "sizes": "512x512",
      "type": "image/png"
    }
  ],
  "start_url": "/index.html",
  "display": "standalone",
  "background_color": "#000000",
  "theme_color": "#4DBA87"
}
```

Implementação PWA Bluetooth

```
connectToDevice: function (device) {
  return (device ? Promise.resolve(device) : this.requestBluetoothDevice())
    .then((device) => this.connectDeviceAndCacheCharacteristic(device))
    .then((characteristic) => this.startNotifications(characteristic))
    .catch((error) => {
      this.log(error);
      return Promise.reject(error);
    });
},
```

Implementação PWA Bluetooth

```
requestBluetoothDevice: function() {
  this.log('Requesting bluetooth device...');

  return navigator.bluetooth.requestDevice({
    filters: [{services: [this.serviceUuid]}],
  }).then((device) => {
    this.log('"' + device.name + '" bluetooth device selected');

    this.device = device;
    this.device.addEventListener('gattserverdisconnected', this.handleDisconnection.bind(this));

    return this.device;
  });
},
```

Implementação PWA Bluetooth

```
connectDeviceAndCacheCharacteristic: function(device) {
  if (device.gatt.connected && this.characteristic) {
    return Promise.resolve(this.characteristic);
  }

  this.log('Connecting to GATT server...');
  You, 16 days ago • Adding bluetooth
  return device.gatt.connect().
      then((server) => {
        this.log('GATT server connected', 'Getting service...');

        return server.getPrimaryService(this.serviceUuid);
      }).
      then((service) => {
        this.log('Service found', 'Getting characteristic...');

        return service.getCharacteristic(this.characteristicUuid);
      }).
      then((characteristic) => {
        this.log('Characteristic found');

        this.characteristic = characteristic; // Remember characteristic.

        return this.characteristic;
    });
},
```

Implementação PWA Bluetooth

```
startNotifications: function(characteristic) {      You, 16 days ago • A
  this.log('Starting notifications...');

  return characteristic.startNotifications().
      then(() => {
        this.log('Notifications started');

        characteristic.addEventListener('characteristicvaluechanged',
            this.handleCharacteristicValueChanged.bind(this));
      });
},
```

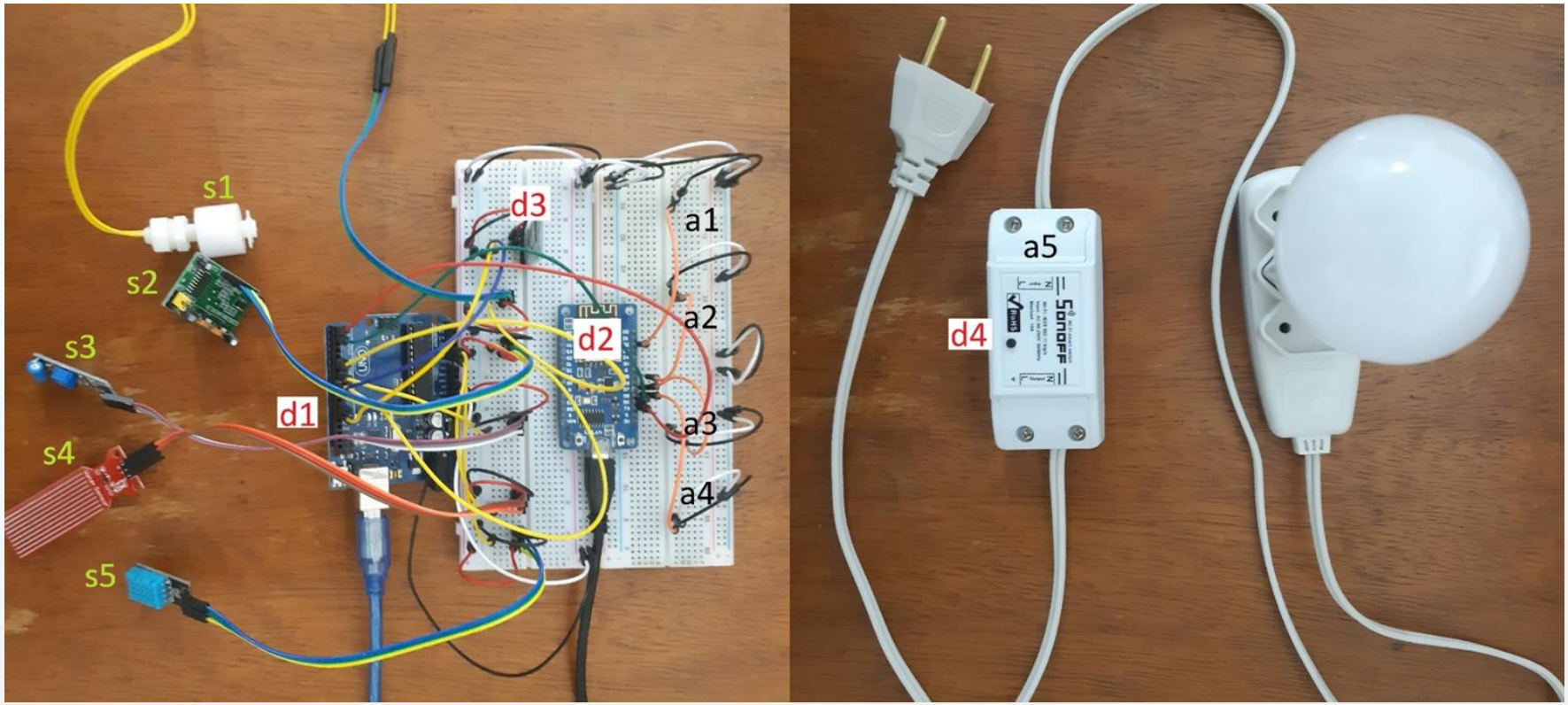
Implementação PWA NFC

```
write: function(message) {
  alert('Writing')
  navigator
    .nfc
    .push(message)
    .then(() => {
      alert("Message pushed.");
    })
    .catch((error) => {
      alert("Push failed :-( try again.");
    });
},
read: function() {
  alert('Reading')
  navigator
    .nfc
    .watch((message) => {
      alert('Message received')
      this.processMessage(message)
    })
    .then(() => alert("Added a watch."))
    .catch(err => alert("Adding watch failed: " + err.name))
},
```

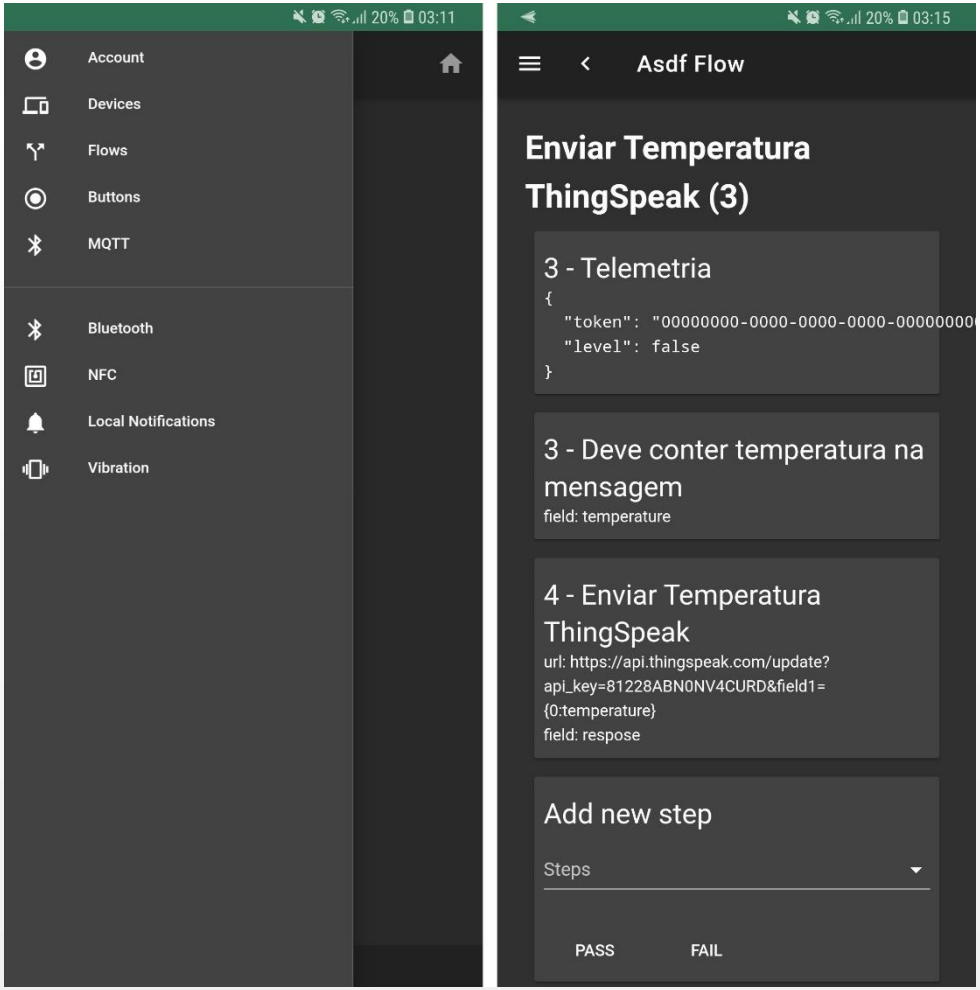
Implementação PWA Notificação

```
notify: function(message) {
  Notification.requestPermission(permission => {
    this.permission = permission
    if (this.permission != 'granted') {return;}

      this.messages.push(message)

      const options = {
        body: message,
        icon: 'img/icons/mstile-150x150.png',
        vibrate: [100, 50, 100],
        data: {
          dateOfArrival: Date.now(),
          primaryKey: 1
        },
      }
      navigator
        .serviceWorker
        .getRegistration()
        .then(reg => {
          reg.showNotification(message, options)
        })
    })
}
```

Implementação
IoT Bancada



Operacionalidade PWA



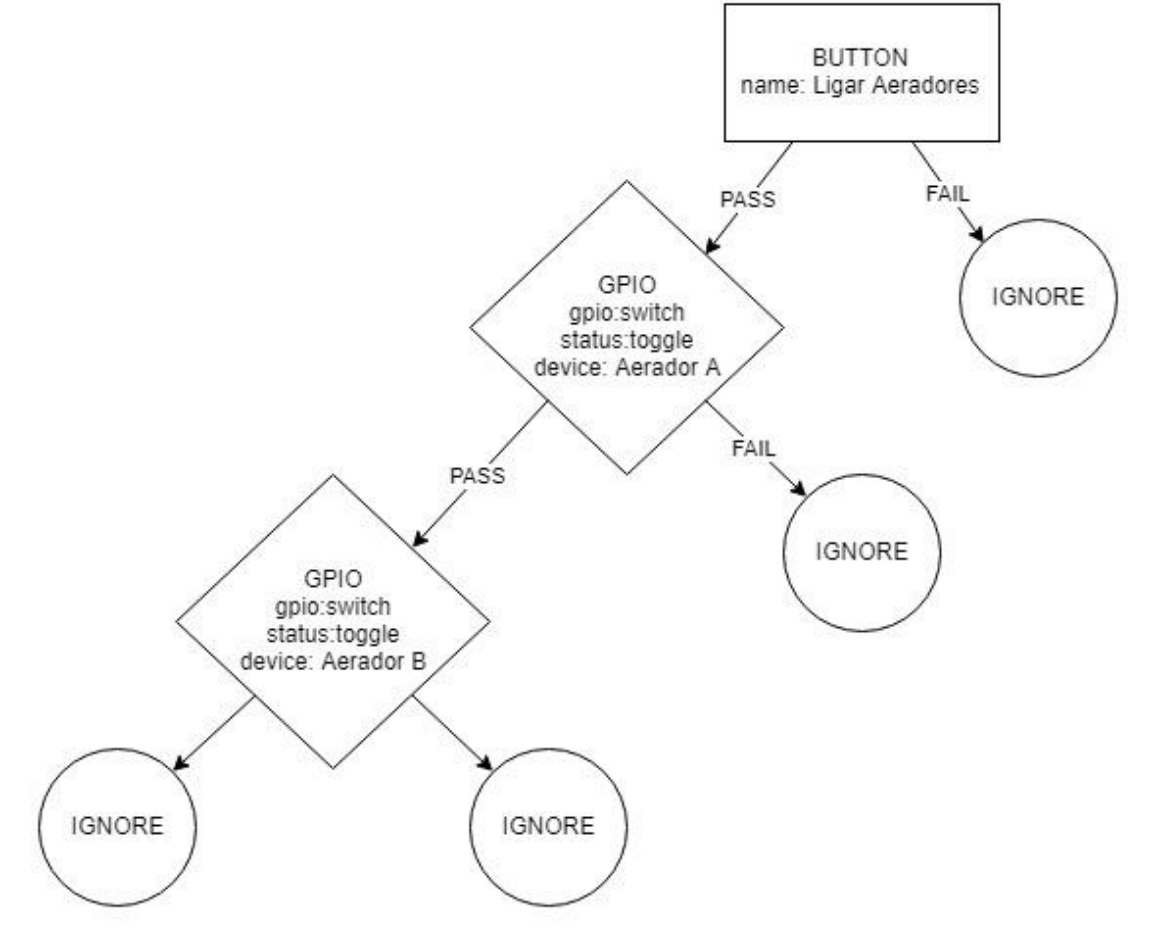
Resultados e Discussões

- Levantados três casos de uso
  - Piscicultura
  - Plantio de Arroz Irrigado
  - Fiação

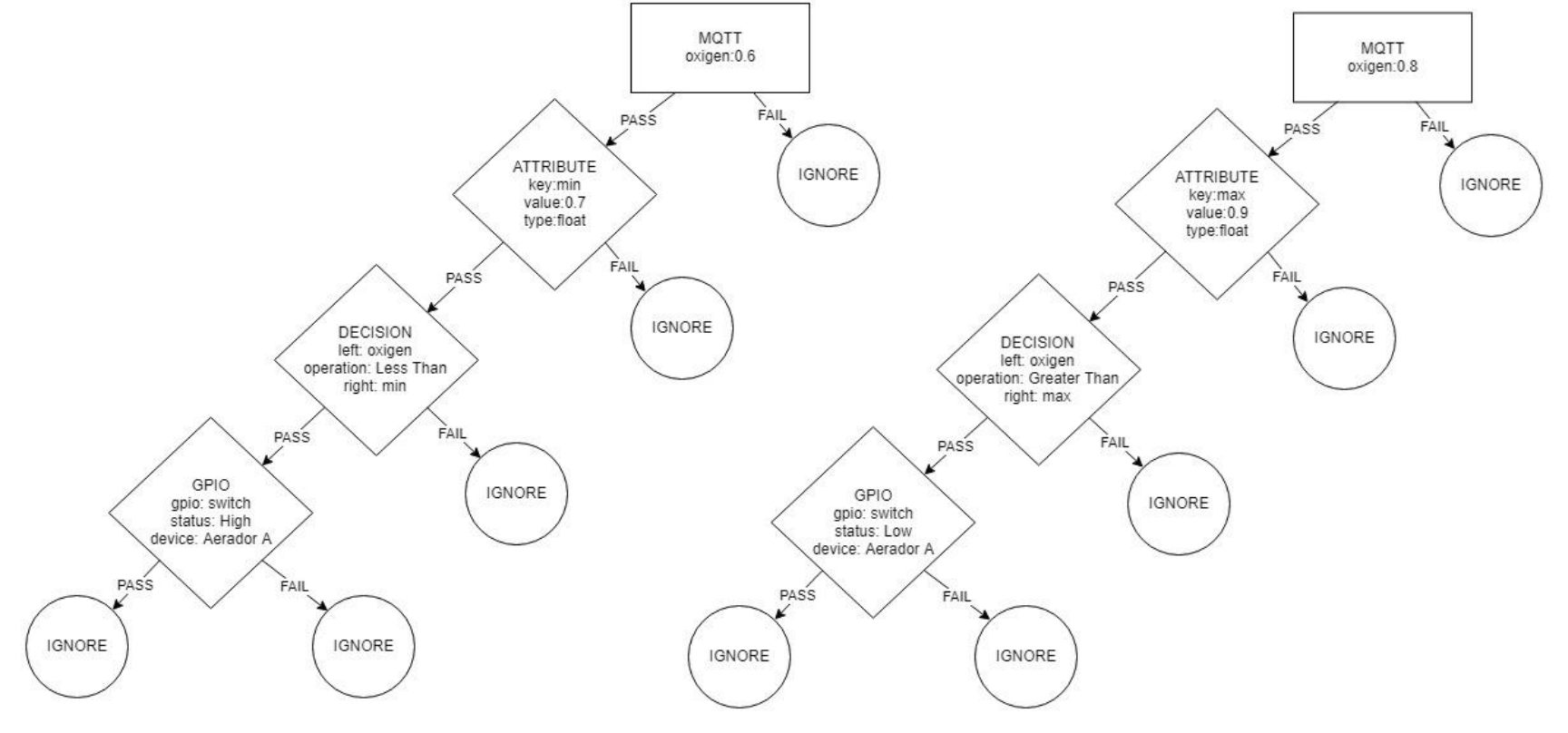
Resultados e Discussões
Solução Piscicultura

- Problema
  - Locomoção até a propriedade
  - Ligar manualmente aeradores distintos
- Desejo
  - Ligar os aeradores remotamente

Resultados e Discussões
Solução Piscicultura



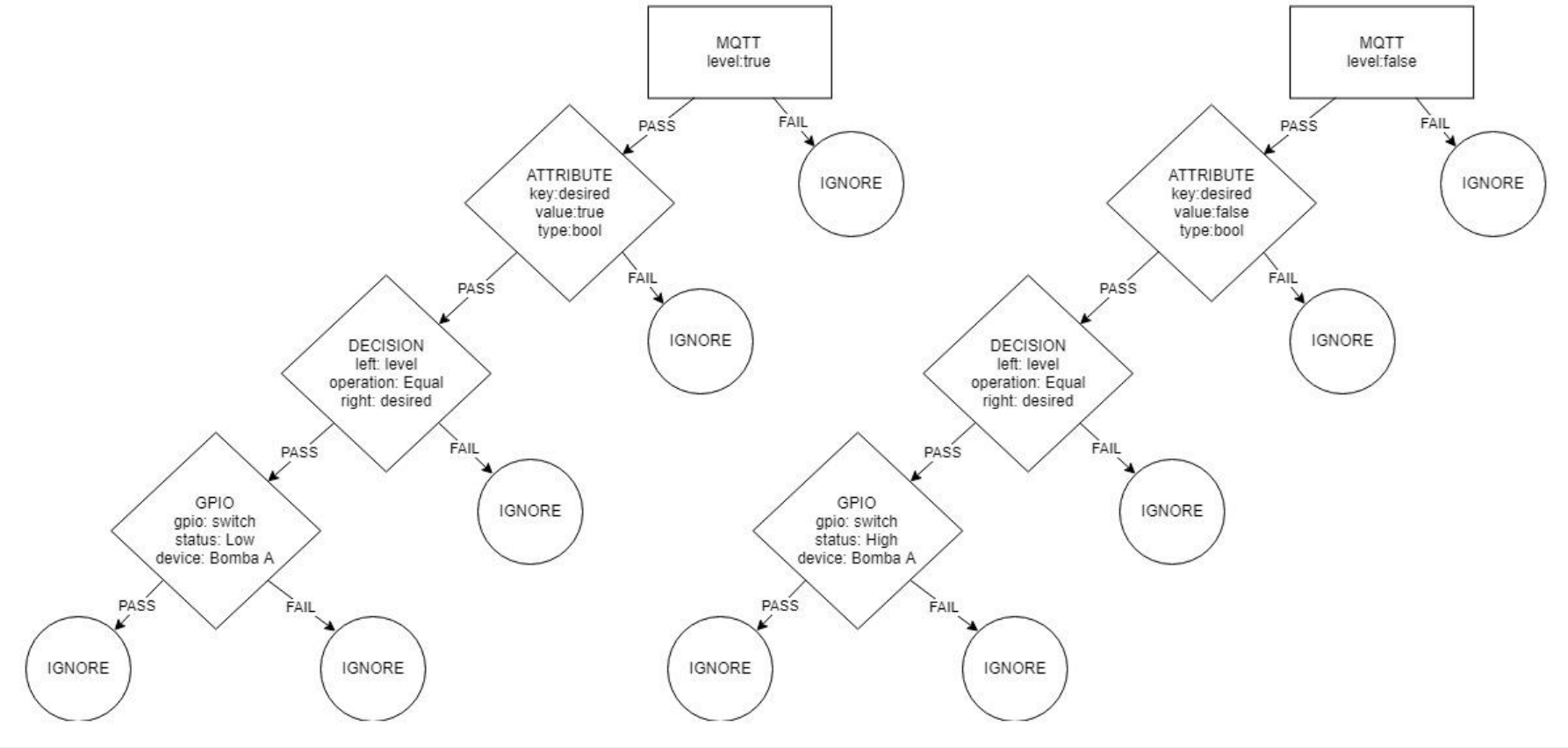
Resultados e Discussões
Solução Piscicultura



Resultados e Discussões
Solução Arroz Irrigado

- Problema
  - Manter lamina de água no nível correto
  - Necessário vigia humana
  - Locomoção até a bomba de água
- Desejo
  - Ligar bomba somente quando necessário
  - Ligar bomba remotamente

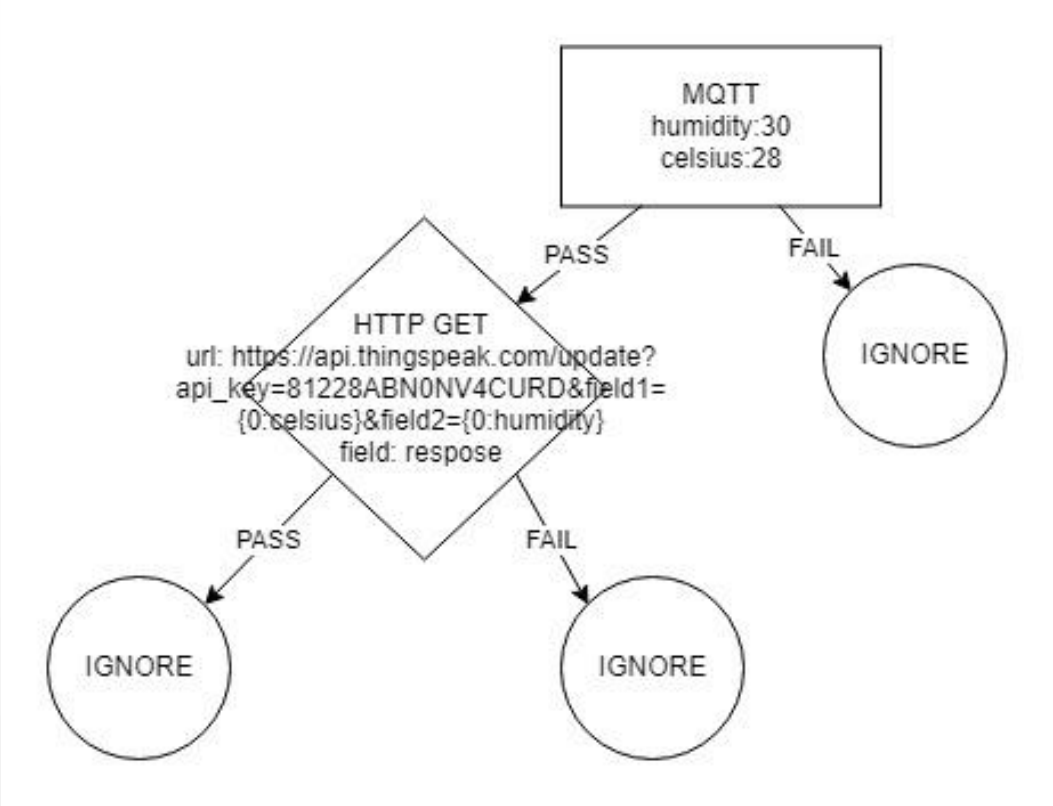
Resultados e Discussões
Solução Arroz Irrigado



Resultados e Discussões
Solução Fiação

- Problema
  - Suspeita da eficiência da solução atual
  - Solução atual não permite a coleta de dados
- Desejo
  - Coletar dados permitindo sua visualização posterior

Resultados e Discussões
Solução Fiação



Conclusões

- O controle de fluxos e decisões, se tornou capaz de contemplar procedimentos de coleta e tomada de decisão
- A aplicação também se mostrou apta, ao realizar integração com uma plataforma externa
- Viabilidade do uso de PWAs como substituto de uma aplicação nativa

Sugestões

- Alterar a arquitetura para que a execução do fluxo possa ocorrer em uma rede local ou internamente nos dispositivos
- Desenvolver novos gatilhos e novos nodos de ação, de forma a atender novos casos de uso
- Definir de maneira mais detalha o protocolo de mensagem, adicionando novos tipos de interesse ao IoT, por exemplo um tipo gpio, definindo os estados possíveis deste tipo

Sugestões

- Desenvolver um firmware universal compatível com o protocolo de mensagem
- Adicionar a possibilidade de realizar o envio do firmware para um dispositivo IoT utilizando a API WebUSB