# Sistema de monitoramento de solo e lavoura

Aluno(a): Johnny Jarbas Hertel

Orientador: Miguel Alexandre Wisintainer

FURB
UNIVERSIDADE DE BLUMENAU

# Roteiro

- Introdução
- Objetivos
- Fundamentação teórica
- Trabalhos correlatos
- Requisitos funcionais e não funcionais
- Especificação e implementação
- Resultados e discussões
- Conclusões
- Extensões
- Demonstração

# Introdução

- Desde o final da segunda guerra teve início um declínio do conjunto de técnicas de cultivo
- Desperdício de água
- Agricultura moderna é insustentável
- Dificuldade para produzir alimentos
- Incompatibilidade com a disponibilidade de recursos naturais

# Objetivo Geral

Desenvolver uma rede de dispositivos capazes de monitorar informações de umidade do solo de uma plantação de morangos e uma aplicação que coleta estes dados e transforma em informações ao agricultor

# Objetivos Específicos

a) Desenvolver uma rede de dispositivos utilizando o módulo ESP8299 Thing capaz de realizar a captura da umidade do solo numa plantação de morangos

b) Transmitir os dados coletados para uma aplicação que vai consumi-los para apresentação ao agricultor

c) Desenvolver uma aplicação para consumir os dados e transformá-los em informações para que o agricultor possa tomar decisões, como identificar a área com necessidade de irrigação.

# Fundamentação Teórica

- A falta de água no planeta
- 40% da população vive sob situação de estresse hídrico
- Previsão de que em 2050 45% da população mundial não terá acesso a quantidade mínima de água
- Projeção de 9 bilhões de pessoas em 2050
- Desafios da agricultura são enormes

# Fundamentação Teórica

- Como alimentar tantas pessoas com uso sustentável de recursos naturais
- Necessidade de inovação agropecuária
- Internet das coisas
- Microcontroladores
- Esp8266 Thing
- Sensores
- Painel solar

# Trabalhos Correlatos

| Características | Aplicação web para monitoramento dos dados coletados por rede de sensores sem fio em ambiente agrícola | Sistema Irriga | Trabalho Proposto |
|---|---|---|---|
| **Faz leitura de umidade solo** | Sim | Sim | Sim |
| **Possui uma rede de dispositivos para monitorar a lavoura** | Sim | Sim | Sim |
| **Possui uma aplicação que transmite as informações para o agricultor** | Sim | Sim | Sim |
| **Informa o agricultor a necessidade de irrigar a lavoura** | Sim | Sim | Sim |
| **Calcula o volume de água necessário para irrigação** | Não | Sim | Não |
| **Necessário licença** | Não | Sim | Não |
| **Baixa custo** | Não | Não | Sim |

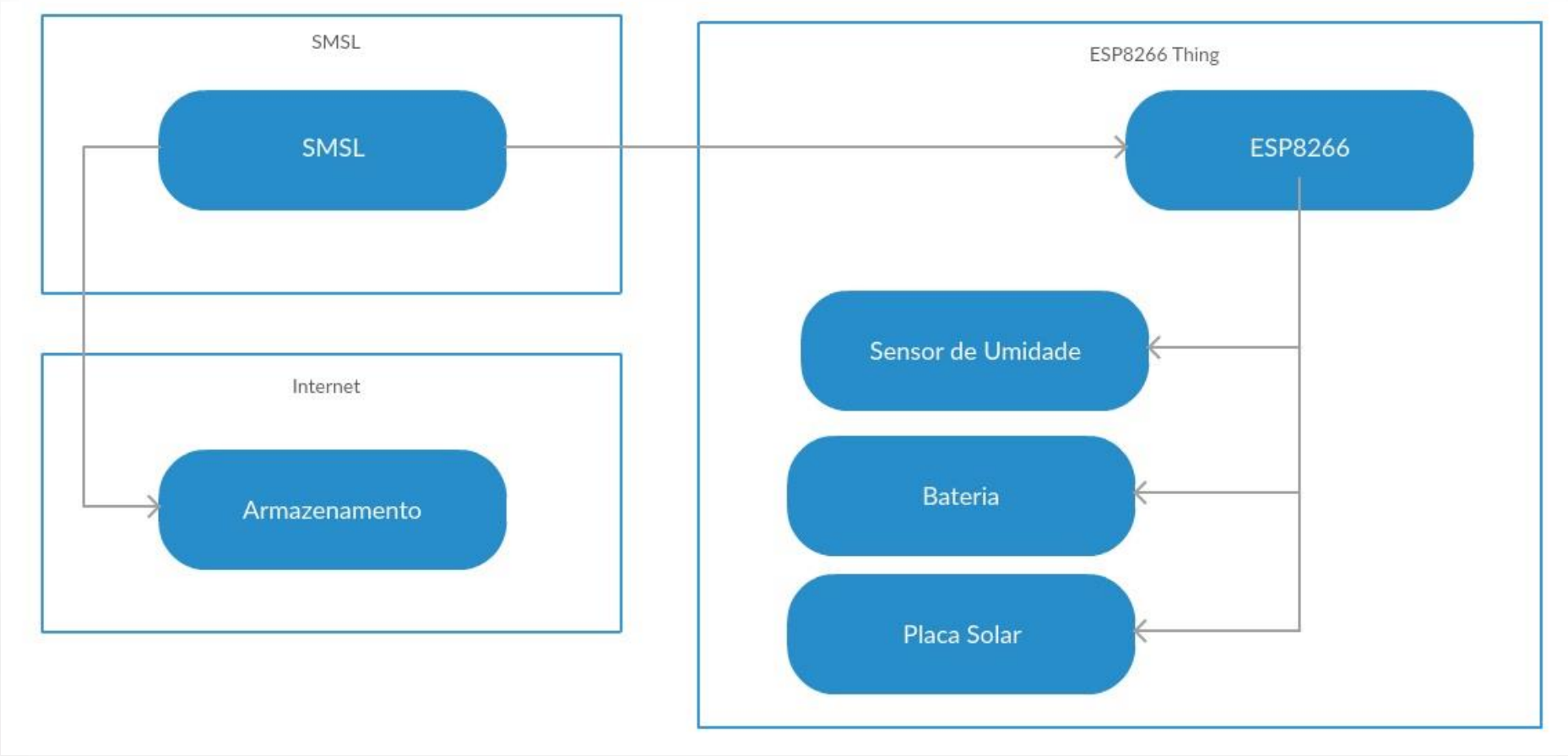
# Requisitos

- **RF01**: Os dispositivos devem ser capazes de fazer leitura da umidade do solo
- RF02: O módulo ESP8266 deve armazenar em sua memória as leituras do solo
- RF03: A aplicação deve demonstrar ao agricultor quando é necessário a irrigação da plantação
- RF04: Os dispositivos devem ser capazes de transmitir as últimas leituras do solo para aplicação em caso de falta de conexão Wi-Fi

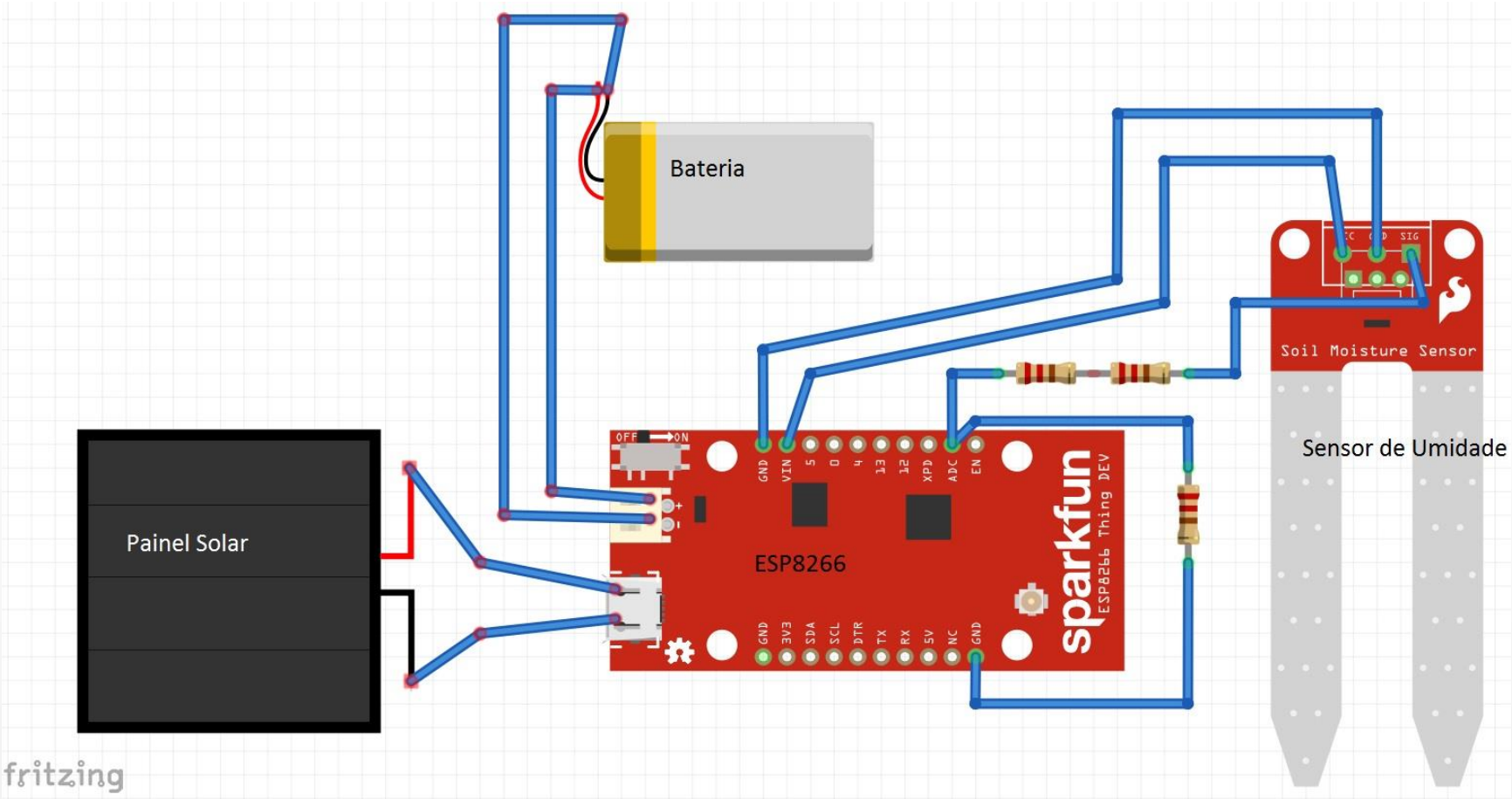
# Requisitos

- **RNF01**: A comunicação dos dispositivos com a aplicação deve ser via Wi-Fi
- RNF02: Os dispositivos devem ser programados na IDE do Arduino
- RNF03: A aplicação que exibe as informações ao agricultor deve ser programada em Hypertext Preprocessor (PHP)
- RNF04: A aplicação deve armazenar os dados no banco de dados MySQL
- RNF05: Para construção dos dispositivos utilizar a placa ESP8266 Thing

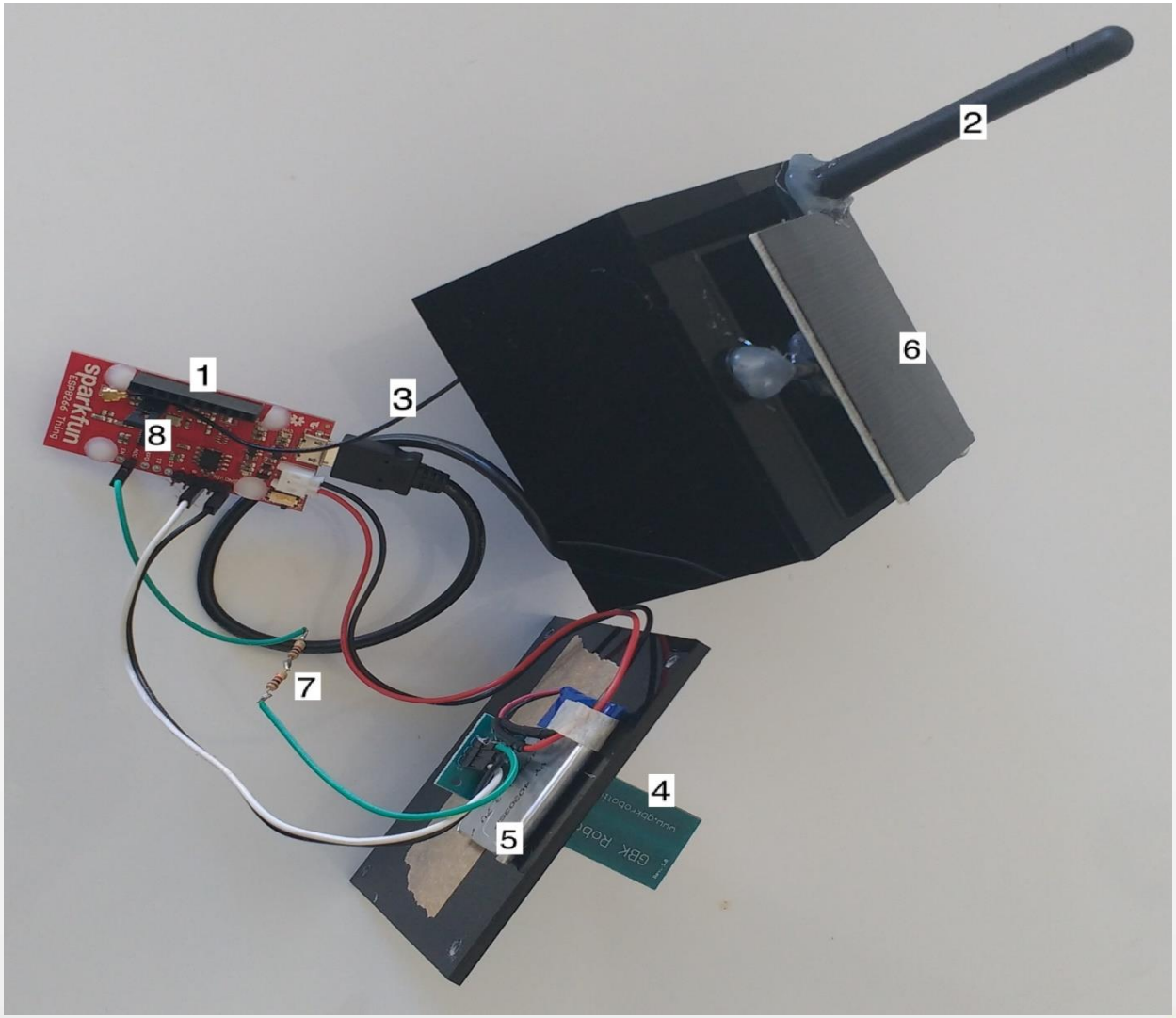
# Diagrama de Arquitetura



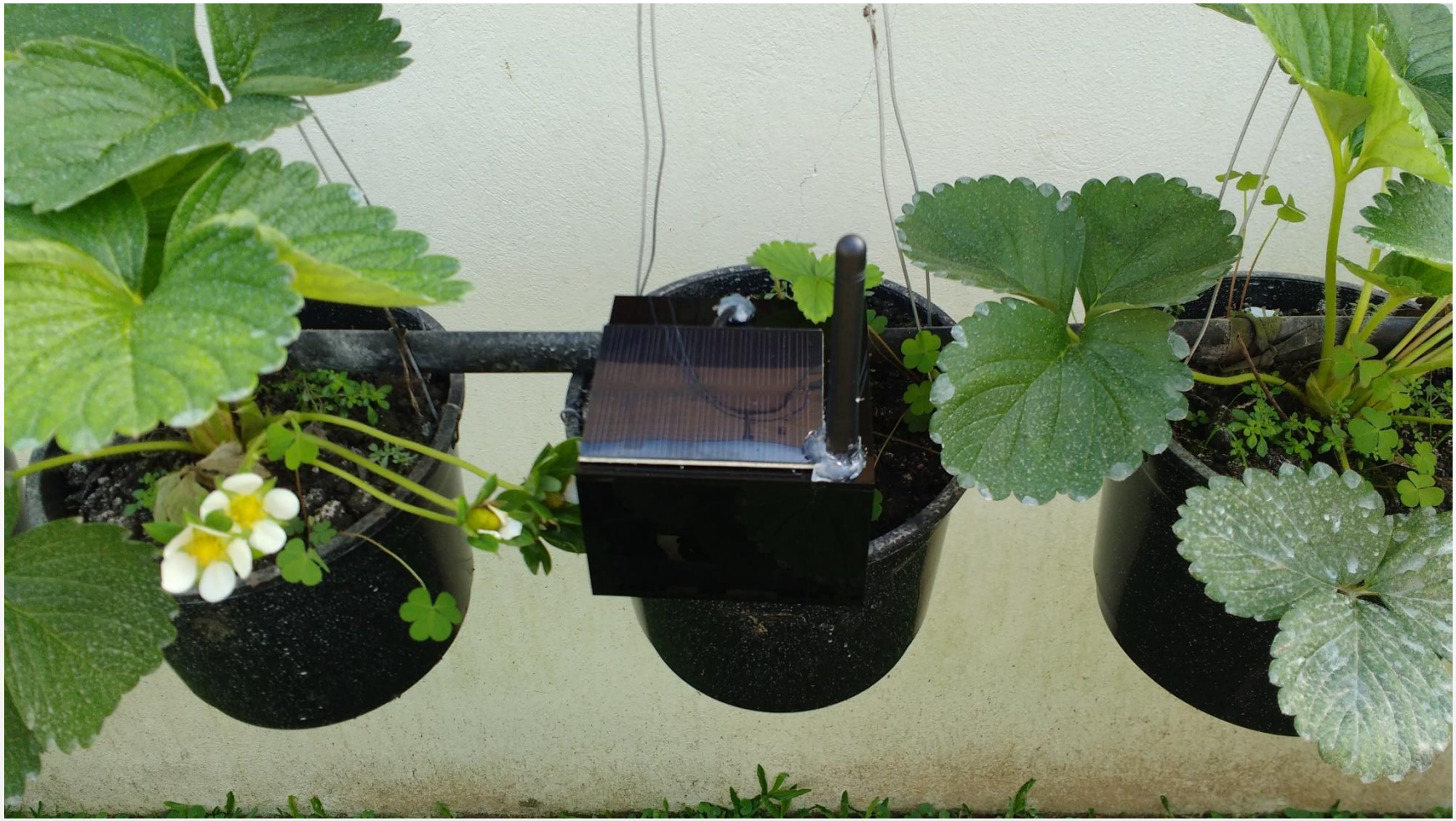
# Diagrama Esquemático



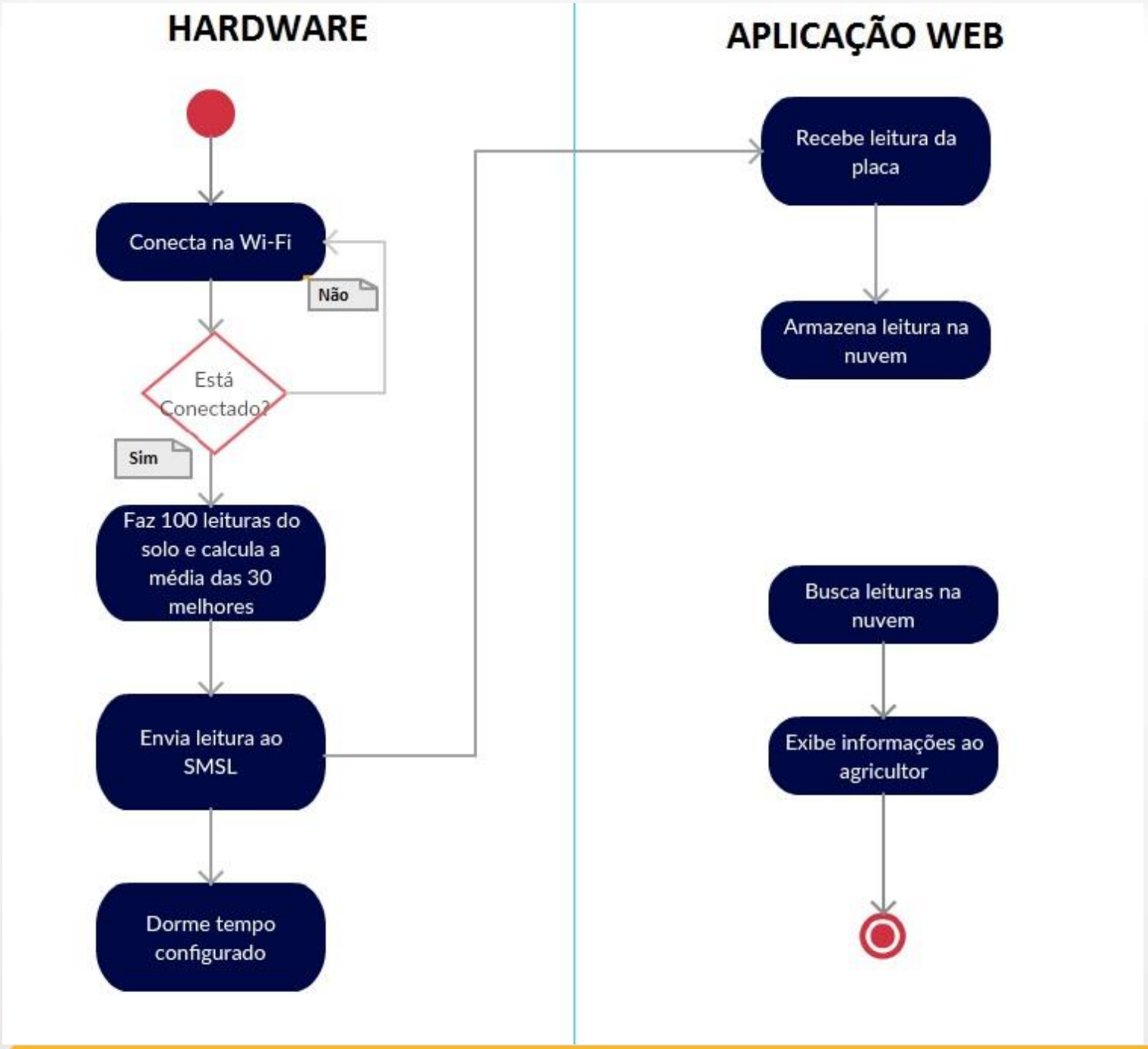
# Hardware Montado



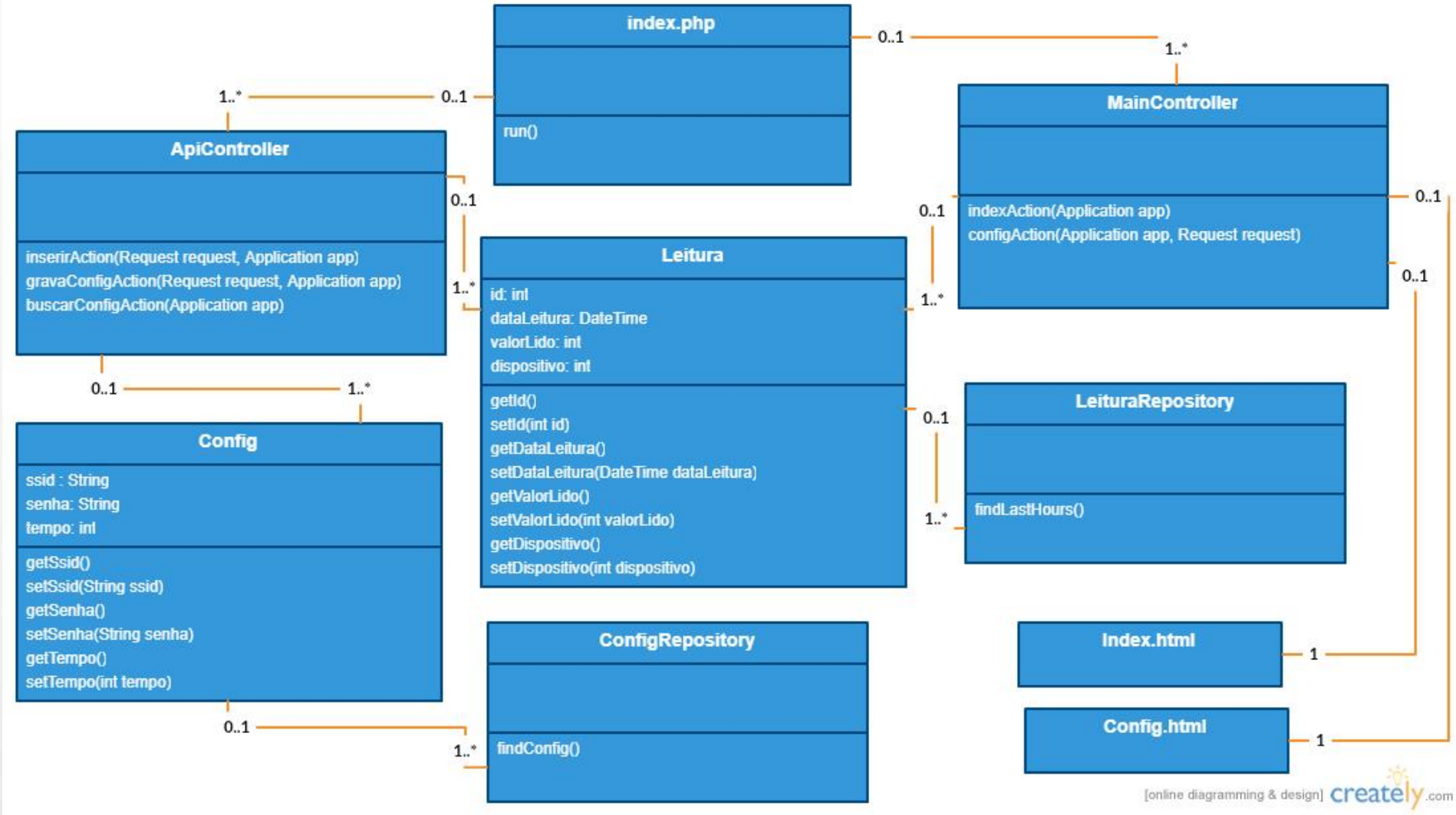
# Hardware em Campo



# Diagrama de Atividades


HARDWARE
APLICAÇÃO WEB
Conecta na Wi-Fi
Não
Está Conectado?
Sim
Faz 100 leituras do solo e calcula a média das 30 melhores
Envia leitura ao SMSL
Dorme tempo configurado
Recebe leitura da placa
Armazena leitura na nuvem
Busca leituras na nuvem
Exibe informações ao agricultor

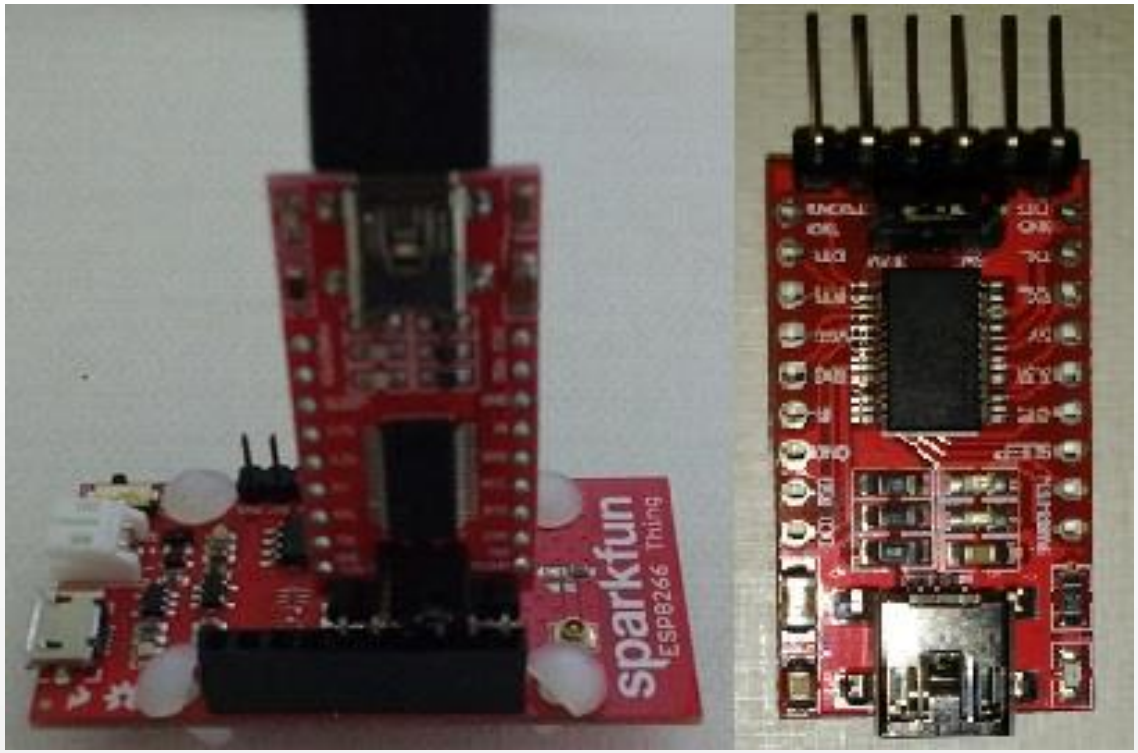
# Diagrama de Classes da Aplicação


index.php
run()
0..1
1..*
1..*
0..1
MainController
indexAction(Application app)
configAction(Application app, Request request)
0..1
0..1
0..1
0..1
ApiController
inserirAction(Request request, Application app)
gravaConfigAction(Request request, Application app)
buscarConfigAction(Application app)
0..1
1..*
Leitura
id: int
dataLeitura: DateTime
valorLido: int
dispositivo: int
getId()
setId(int id)
getDataLeitura()
setDataLeitura(DateTime dataLeitura)
getValorLido()
setValorLido(int valorLido)
getDispositivo()
setDispositivo(int dispositivo)
0..1
1..*
0..1
1..*
LeituraRepository
findLastHours()
Config
ssid : String
senha: String
tempo: int
getSsid()
setSsid(String ssid)
getSenha()
setSenha(String senha)
getTempo()
setTempo(int tempo)
0..1
1..*
ConfigRepository
findConfig()
Index.html
1
Config.html
1
[online diagramming & design] creately.com

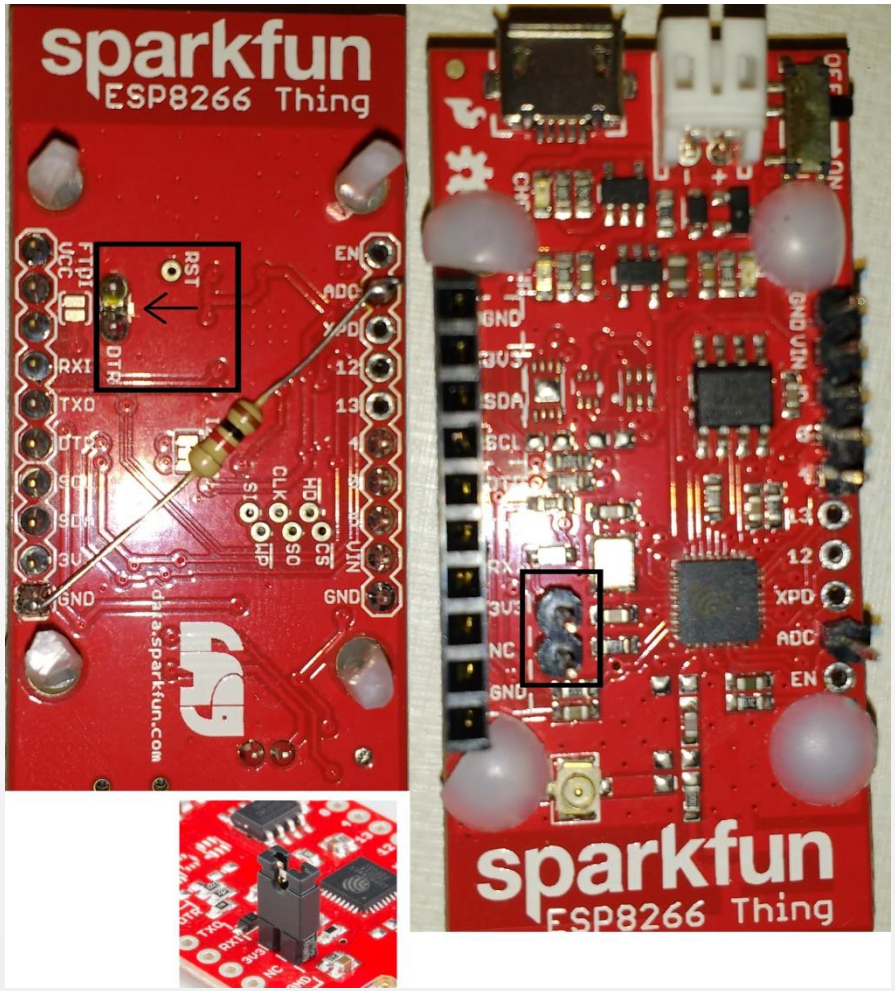
# Implementação

- Trabalho desenvolvido na arquitetura cliente-servidor
- Cliente é responsável por coletar os dados do solo e transmitir ao servidor
- Servidor processa e armazena os dados e exibe as informações ao agricultor

# Conversor USB/Serial



# Jumper


sparkfun
ESP8266 Thing
RST
DTR
EN
ADC
XPD
12
13
RXI
TXO
GND
sparkfun
ESP8266 Thing

# Divisor de Tensão

Arduino e Cia

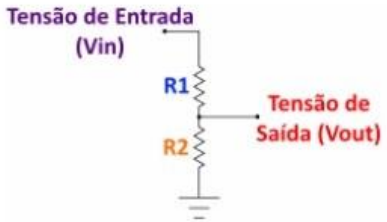
## Calculadora Online - Divisor de tensão com resistores

### Calculador de resistências para divisor de tensão

Formulário para cálculo dos resistores utilizados em um divisor de tensão. Também pode ser utilizado para calcular a Tensão de Entrada, R1 ou R2.

Como usar o calculador :

- Preencha os campos **Tensão de entrada, R1 e R2** para obter a Tensão de Saída
- Preencha os campos **Tensão de entrada, R2 e Tensão de saída** para obter o valor de R1
- Preencha os campos **Tensão de entrada, R1 e Tensão de saída** para obter o valor de R2
- Preencha os campos **R1, R2 e tensão de saída** para obter a Tensão de entrada




Digite os parâmetros desejados e clique em **CALCULAR**

Para limpar os campos do formulário, clique em **LIMPAR**

| | | | |
|---|---|---|---|
| Tensão de entrada | 5 | Volts | Calcular |
| R1 | 2000 | ohms | Limpar |
| R2 | 1000 | ohms | |
| Tensão de Saída | 1 | Volts | |

FURB
UNIVERSIDADE DE BLUMENAU

# Algoritmo de conexão inicial

```
40 void setup() {
41   pinMode(pino_alimentacao_sensor, OUTPUT);
42   Serial.begin(9600);
43   os_timer_setfn(&mTimer, tCallback, NULL);
44   os_timer_arm(&mTimer, 1000, true);
45   //Abre o sistema de arquivos (mount)
46   openFS();
47   bool conectou = false;
48   if (SPIFFS.exists(configFile)) {
49     readConfigFile();
50
51     int str_len = ssidConfigFile.length();
52     char ssid_array[str_len];
53     memset(ssid_array, '\0', sizeof(ssid_array));
54     ssidConfigFile.toCharArray(ssid_array, str_len);
55
56     str_len = passConfigFile.length();
57     char pass_array[str_len];
58     memset(pass_array, '\0', sizeof(pass_array));
59     passConfigFile.toCharArray(pass_array, str_len);
60
61     if (!conectaWiFi(ssid_array, pass_array)) {
62       conectou = conectaWiFi(SSID_REDE, SENHA_REDE);
63       while (!conectou) {
64         conectou = conectaWiFi(SSID_REDE, SENHA_REDE);
65       }
66     }
67   } else {
68     conectou = conectaWiFi(SSID_REDE, SENHA_REDE);
69     while (!conectou) {
70       conectou = conectaWiFi(SSID_REDE, SENHA_REDE);
71     }
72   }
73 }
```

# Conexão com Wi-Fi

- Atualiza o arquivo de configuração
- Atualiza o relógio do ESP8266

# Algoritmo Principal

```
75 void loop() {
76
77   // só manda energia para o sensor quando vai utilizar ele, para ser mais eficiente.
78   digitalWrite(pino_alimentacao_sensor, HIGH);
79
80   // faz 100 leituras da tensão que o pino analógico esta recebendo. Esta tensão pode
81   // sendo 0v considerado sem umidade e 1v totalmente umido.
82   int reads[100];
83   for (int i = 0; i < 100; i++) {
84     reads[i] = analogRead(pino_sinal_analogico);
85   }
86
87   // ordena o array de leituras
88   for (int i = 0; i < 100; i++) {
89     for (int j = 1; j < (100 - i); j++) {
90       if (reads[j - 1] > reads[j]) {
91         int temp = reads[j - 1];
92         reads[j - 1] = reads[j];
93         reads[j] = temp;
94       }
95     }
96   }
97
98   // descarta as 35 maiores e as 35 menores leituras
99   // soma as 30 melhores leituras e divide pela quantidade de leituras para ter o val
100  int analogValue = 0;
101  for (int i = 35; i < 65; i++) {
102    analogValue += reads[i];
103  }
104  analogValue = analogValue / 30;
```

```
106   char leituraSolo[90];
107   sprintf(leituraSolo, "{\"valorLido\": %d,\"dispositivo\": %d, \"dataLeitura\'
108   if (SPIFFS.exists(leiturasFile)) {
109     writeFile(leiturasFile, leituraSolo);
110   } else {
111     createFile(leiturasFile);
112     writeFile(leiturasFile, leituraSolo);
113   }
114
115   readConfigFile();
116
117   int str_len = ssidConfigFile.length();
118   char ssid_array[str_len];
119   memset(ssid_array, '\0', sizeof(ssid_array));
120   ssidConfigFile.toCharArray(ssid_array, str_len);
121
122   str_len = passConfigFile.length();
123   char pass_array[str_len];
124   memset(pass_array, '\0', sizeof(pass_array));
125   passConfigFile.toCharArray(pass_array, str_len);
126
127   if (conectaWiFi(ssid_array, pass_array)) {
128     enviaInformacao();
129   }
130
131   // desliga alimentação do pino
132   digitalWrite(pino_alimentacao_sensor, LOW);
133
134   delay(tempoConfigFile.toInt());
135 }
```

# Aplicação Web

```php
<?php

require "../setup.php";

$app->get('/', "App\\Controller\\MainController::indexAction")->bind( routeName: 'index');
$app->get('/config', "App\\Controller\\MainController::configAction")->bind( routeName: "config_view");
$app->post( pattern: '/inserir',  to: "App\\Controller\\ApiController::inserirAction");
$app->post( pattern: '/gravar',  to: "App\\Controller\\ApiController::gravaConfigAction")->bind( routeName: 'gravar_config');
$app->get('/buscar', "App\\Controller\\ApiController::buscarConfigAction")->bind( routeName: 'buscar_config');

$app->run();
```

# ApiController

- inserirAction
- gravaConfigAction
- buscarConfigAction

# MainController

- Traz os HTML’s
- indexAction
- configAction

# Operacionalidade da Implementação

CONFIGURAÇÃO

# Sistema de Monitoramento de Solo e Lavoura


| Baixa | Moderada | Alta |
|---|---|---|


Informações da rede Wi-Fi

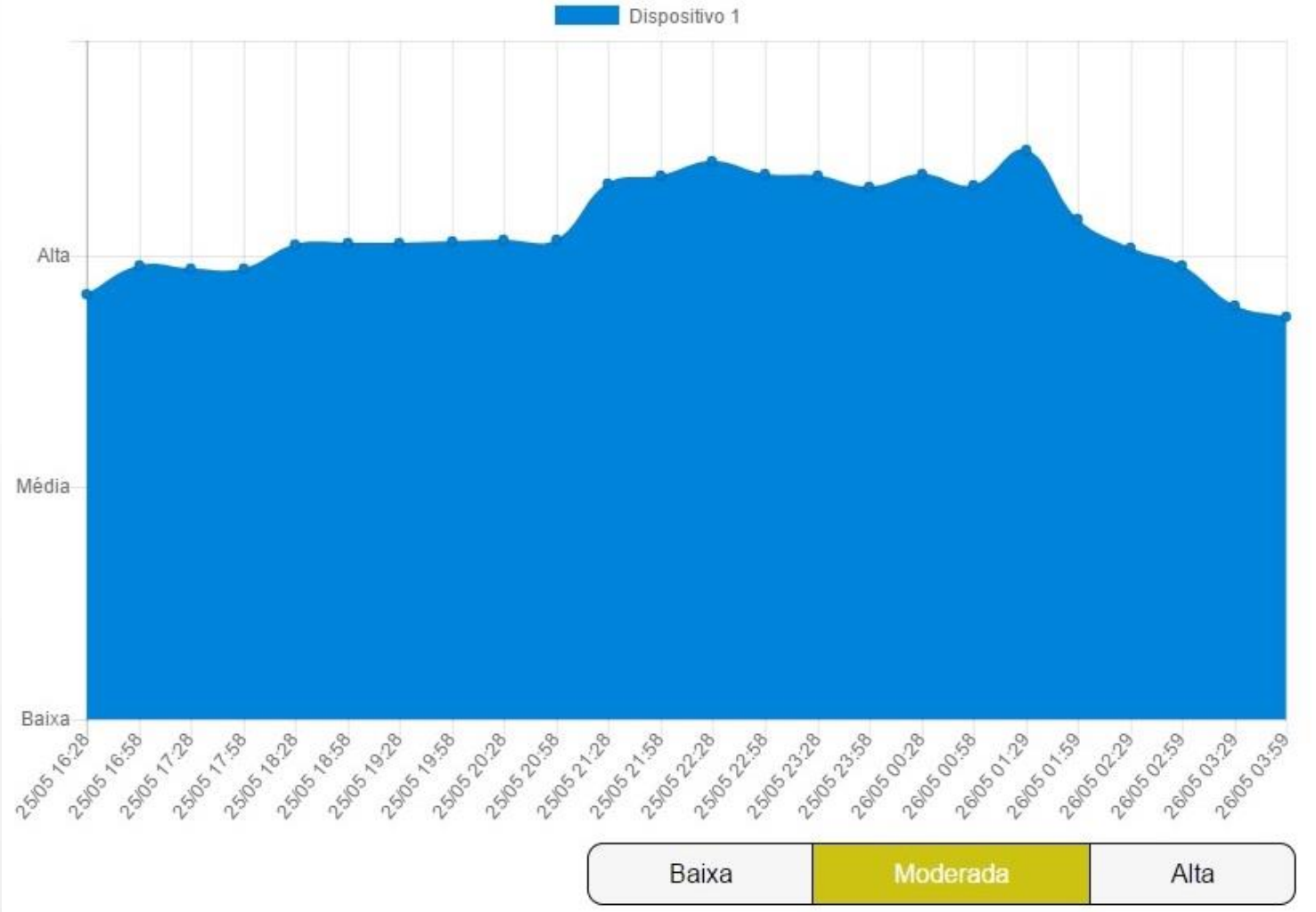
| | |
|---|---|
| SSID Rede | WiFi Johnny |
| Senha Rede | |
| Tempo Leitura (ms) | 1800000 |

Gravar

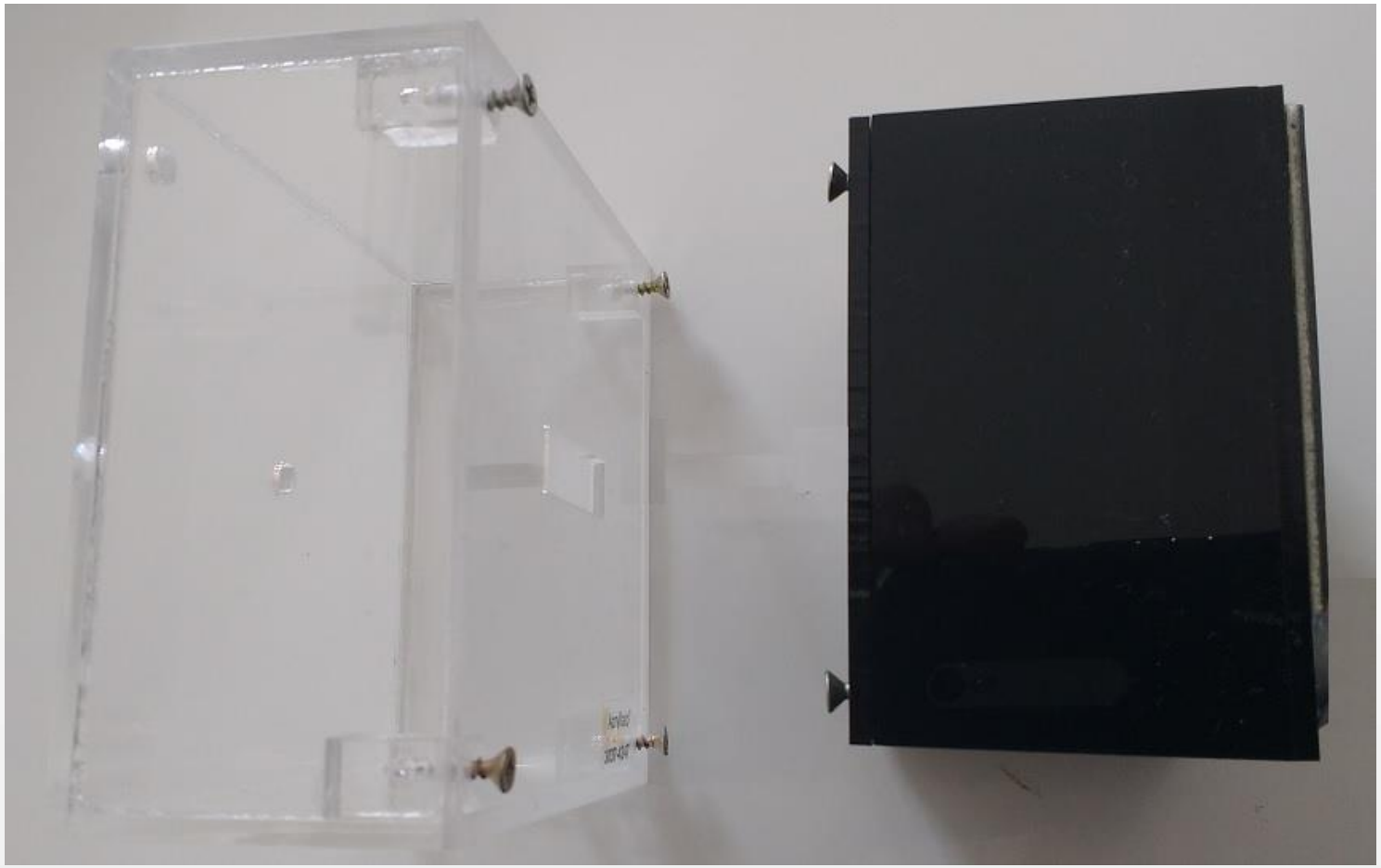
# Resultados e Discussões

- Atingiu os objetivos propostos
- Interface web não foi preparada para visualização em Smartphones
- Necessário prever a falta de conexão Wi-Fi
- Necessário criar algoritmo para primeira conexão com a Wi-Fi
- Dificuldade com sensor que foi substituído

# Resultados e Discussões



# Acondicionamento do Hardware



# Conclusões e Sugestões

- Conforme demonstrado, foi concebido um protótipo e uma aplicação que demonstram a viabilidade do projeto
- Ferramentas utilizadas para especificação e desenvolvimento atenderam as necessidades
- Avaliar a possibilidade de utilizar a biblioteca Deep Sleep

# Conclusões e Sugestões

- Incluir possibilidade de ativar irrigadores a partir da aplicação web
- Incluir possibilidade de notificação ao agricultor através de e-mail/SMS
- Calcular volume de água que deve ser aplicado na plantação
- Possibilidade de exibir a previsão do tempo na aplicação web

# Demonstração