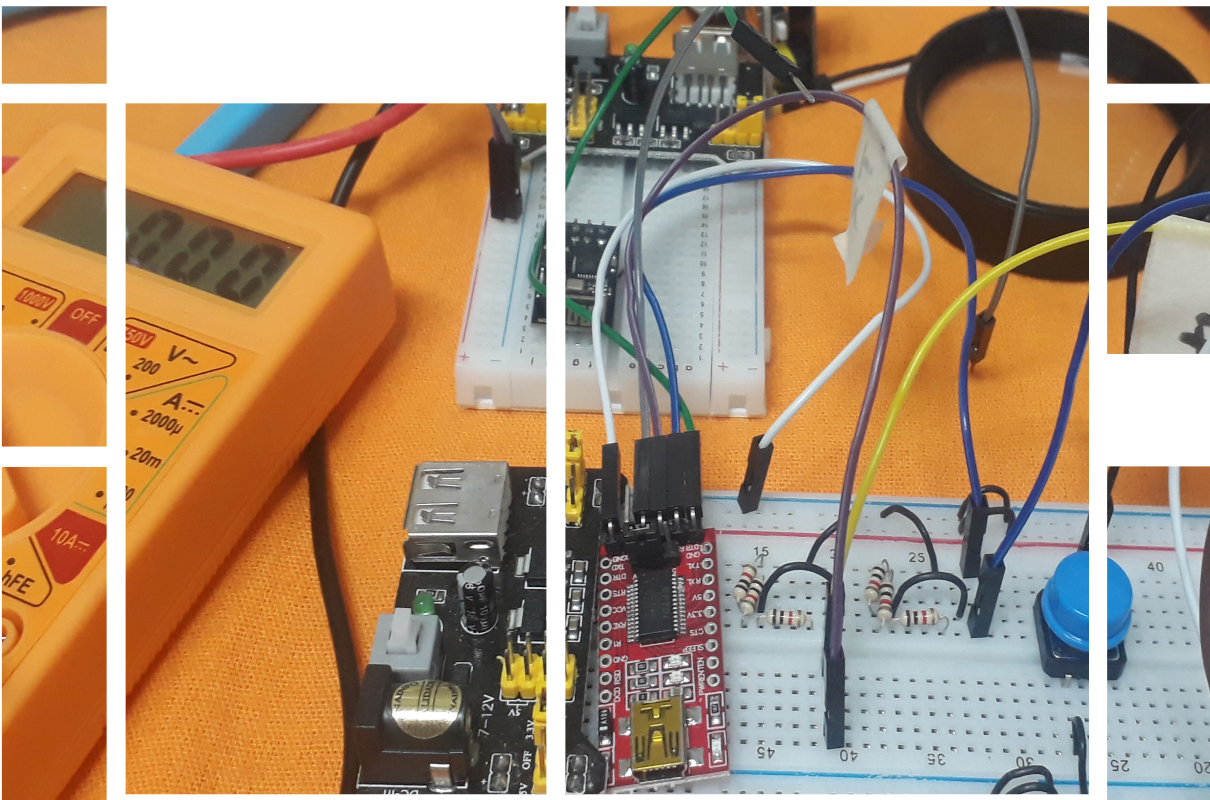
BOLETIM DE PESQUISA E DESENVOLVIMENTO
25

ISSN 1981-1004
Dezembro / 2023

IoT na Fitopatologia

Proposta de Rede *Wi-fi* P2P com ESP8266 para Monitorar Microclimas em Áreas Próximas



Embrapa

ISSN 1981-1004
Dezembro / 2023

Empresa Brasileira de Pesquisa Agropecuária
Embrapa Uva e Vinho
Ministério da Agricultura e Pecuária

BOLETIM DE PESQUISA E DESENVOLVIMENTO 25

IoT na Fitopatologia

Proposta de Rede *Wi-fi* P2P com ESP8266 para Monitorar Microclimas em Áreas Próximas

Fabio Rossi Cavalcanti

Embrapa Uva e Vinho
Bento Gonçalves, RS
2023

Embrapa Uva e Vinho
Rua Livramento, 515 - Caixa Postal 130
95701-008 Bento Gonçalves, RS
Fone: (0xx) 54 3455-8000

www.embrapa.br/uva-e-vinho
www.embrapa.br/fale-conosco/sac

1ª edição
Publicação digital (2023): PDF

Dados Internacionais de Catalogação na Publicação (CIP)
Embrapa Uva e Vinho

IoT na Fitopatologia: proposta de rede wi-fi P2P com ESP8266 para monitorar microclimas em áreas próximas / Fábio Rossi Cavalcanti – Bento Gonçalves: Embrapa Uva e Vinho, dez. 2023.

PDF (23 p.) -- (Boletim de Pesquisa e Desenvolvimento / Embrapa Uva e Vinho, ISSN 1981-1004; 25).

1. Peer-to-peer. 2. Agricultura 4.0. 3. Arduino. 4. Agricultura de Precisão. 5. NodeMCU. I. Cavalcanti, Fábio Rossi. II. Embrapa Uva e Vinho. III. Série.

CDD (21. ed.) 681.763

Rochelle Martins Alvorcem (CRB-10/1810)

Sumário

IoT na Fitopatologia

Proposta de Rede *Wi-fi* P2P com ESP8266 para Monitorar Microclimas em Áreas Próximas

Fabio Rossi Cavalcanti[1]

Resumo – Atualmente, a Agricultura de Precisão traz modernização e substituições tecnológicas associadas às cadeias produtivas do alimento. O uso de sensoriamento remoto e Sistema de Coordenadas Geográficas (GIS), do Sistema de Posicionamento Global (GPS), sensores, atuadores, controladores e processadores (de baixo custo), juntamente com novas abordagens em *Application Programming Interface* (APIs), tecnologias de armazenamento em massa, servidores em nuvem, dispositivos de internet das coisas (IoT) e recente oferta de soluções em conectividade rural, como Lora, SigFox, NB-IoT, trazem a possibilidade de ampla adoção de tecnologias mais leves e baratas destinadas ao manejo agrícola. O ESP8266 é uma família de microcontroladores que possui comunicação por *wi-fi* residente e baseado no dispositivo homônimo. Vários dispositivos estão disponíveis, como os ESP-01, ESP-07, ESP-12e e o ESP-32. A partir do ESP-07, o dispositivo pode vir com uma placa do tipo NodeMCU, que é uma plataforma de IoT, de baixo custo e *open source* totalmente compatível com o ambiente Arduíno que, por sua vez, é uma plataforma livre de prototipagem eletrônica programável com uma linguagem homônima muito semelhante a C/C++. No contexto acima, o objetivo do presente trabalho foi o de apresentar proposição de uma rede do tipo *peer-to-peer* (P2P) com dispositivos ESP8266 (ESP-1 e ESP-12e) funcional, mas passível de desenvolvimentos, baseada na recepção de um sinal simples de *wi-fi* em roteador central (provedor) por repetição desse sinal para captação de dados de sensores também construídos com ESP8266.

Termos para indexação: Peer-to-peer, agricultura 4.0, Arduino, NodeMCU, agricultura de precisão.

---

[1] Engenheiro-agrônomo, doutor em Fitopatologia, pesquisador da Embrapa Uva e Vinho, Bento Gonçalves, RS

IoT in Plant Pathology

P2P *Wi-fi* Network with ESP8266 to Monitor Microclimates in Nearby Areas

Abstract – In fact, Precision Agriculture brings modernization and technological replacements associated with food production chains. The use of remote sensing and Geographic Coordinate Systems (GIS), Global Positioning System (GPS), sensors, controllers, and processors (at low cost), along with new approaches in Application Programming Interface (APIs), mass storage technologies, cloud servers, Internet of Things (IoT) devices, and the recent offer of rural connectivity solutions (e.g., Lora, SigFox, NB-IoT) make possible the widespread adoption of lighter and cheaper technologies aimed at agricultural management. The ESP8266 is a family of microcontrollers that has resident wi-fi communication and is based on the homonymous device. Several of those are available such as ESP-01, ESP-07, ESP-12e, and ESP-32. From ESP-07, the device can come with a NodeMCU type board, which is a low-cost, fully open-source IoT platform that is fully compatible with the Arduino, which is, in turn, a free programmable electronic prototyping platform with a homonymous language very similar to C/C++. In the above context, the aim of this work was to propose a peer-to-peer (P2P) network with ESP8266 devices (ESP-1 and ESP-12e) that is functional but open to developments, based on the reception of a simple wi-fi signal in a central router (provider) by repeating this signal for data capture from sensors also built with ESP8266.

Index terms: Peer-to-peer, agriculture 4.0, Arduino, NodeMCU, precision agriculture.

Introdução

Paralelamente à Agricultura Regenerativa, uso de bioestimulantes e biopesticidas e rastreamento de alimentos, a Agricultura de Precisão (AP) está atualmente sendo tratada como uma tendência para o futuro. No entanto, considerar que a AP passe por uma atualização tecnológica de estratégias convencionais de agronomia é o primeiro passo para aplicar a modernização associada à produção agrícola proposta de modo bem sucedido, tanto em um determinado ponto da cadeia, quanto em toda a cadeia produtiva. A Agricultura de Precisão não depende tanto de revoluções tecnológicas disruptivas quanto pode parecer. Mesmo assim, não se pode negar que *upgrades* tecnológicos, fartamente disponíveis hoje a preços bastante módicos e atraentes, estão possibilitando grande oferta de soluções tecnológicas sofisticadas para a agricultura, principalmente entre *startups* e empresas jovens (FAO, 2020). Essas soluções costumam usar grandes pacotes de dados, gestão da variabilidade espacial e temporal, redução de custos e aumento de rendimento e redução de impactos ambientais (Ahmad; Nabi, 2021).

O uso de sensoriamento remoto e Sistema de Coordenadas Geográficas (GIS), Sistema de Posicionamento Global (GPS), sensores, atuadores, controladores e processadores de baixo custo, juntamente com novas abordagens em *software* apropriado, cada vez mais na forma de *Application Programming Interface* (APIs), tecnologias de armazenamento em massa, servidores em nuvem, dispositivos de internet das coisas (IoT) e recente oferta de soluções em conectividade rural, como, por exemplo, Lora, SigFox, NB-IoT e outras, trazem a possibilidade de ampla adoção de tecnologias mais leves destinadas ao manejo agrícola (Chakraborty et al., 2020). Dentre as aplicações já correntes, cabe citar levantamentos de solos com deficiências nutricionais, quantificação precisa do uso da água de irrigação, racionalização na aplicação de fertilizantes e pesticidas, com consequente redução de impactos ambientais. O barateamento de sensores permitiu um significativo incremento do monitoramento em pós-colheita do produto. Tratores autônomos, robôs terrestres e veículos aéreos não tripulados (Vant) podem ser usados para fins de captação de imagens, vigilância e pulverização (Ahmad; Nabi, 2021).

Por isso, a AP pressupõe aquisição certa de dados, intervenção no momento certo, no lugar certo, usando o método certo na quantidade/dose

adequada (*five 'R - right'concept* ou conceito dos cinco 'certos') (Zimmerman, 2008). A estrutura da AP está baseada na coleta e transferência de dados precisos para impulsionar APIs em processamentos centrais de tomada de decisão, muitas vezes em servidores ou plataformas de serviços de empresas. Esses geram informações que guiam ferramentas de AP. O Sistema de Agricultura de Precisão *(*PFS) englobou trabalhos de incremento técnico digital em agricultura, a partir dos anos 1990. A PFS já tinha sido uma quebra de paradigma tecnológico no setor agrícola, trazendo em um só tempo todas os nós tecnológicos acima comentados, antes da Agricultura 4.0 e, recentemente, a 5.0 (Tran; Nguyen, 2006, Bakthiari; Hematian, 2013).

O ESP8266 é uma família de microcontroladores (*Cadence Tensilica* LX106) de baixíssimo custo, fabricado na China, que possui comunicação por *Wireless Fidelity* (*wi-fi*) residente e baseado no chipe homônimo. O dispositivo está disponível no mercado brasileiro desde 2014, com o modelo ESP-01. Depois disso vários outros modelos foram lançados e disponibilizados, sendo os mais conhecidos pela comunidade *maker* além do próprio ESP-01, o ESP-07, o ESP-12e e o ESP-32 (este de 32bits, com *bluetooth*). A partir do ESP-07, o dispositivo pode vir com uma placa do tipo NodeMCU, que é uma plataforma de IoT, de baixo custo e *open-source* totalmente compatível com a plataforma Arduíno que, por sua vez, é livre para prototipagem eletrônica programável com uma linguagem homônima baseada em C/C++. Inicialmente, o NodeMCU incluía o ESP8266 no módulo ESP-12, mas recentemente foi disponibilizado o NodeMCU para ESP-32. Antes do ESP-12e os microcontroladores se conectavam por TCP/IP usando um conjunto de comandos AT (Hayes), mas, por uma reprogramação simples de *firmware* é possível trabalhar com a família ESP8266 em um ambiente Arduíno em melhor nível. Também existem os próprios kits da fabricante para desenvolvimento de software (SDKs) disponíveis *on-line*.

No entorno do ESP8266, existe uma vasta gama de dispositivos e sensores para os mais diversos fins, uma vez que com o NodeMCU, os ESPs se tornam compatíveis com virtualmente todo o *hardware* associado à plataforma Arduino. Assim, é de se esperar que seja bastante abundante na rede mundial de computadores, acessível por sites de busca, encontrar disponível gratuitamente um sem número de projetos e aplicações para a agricultura nesses dispositivos apresentados acima. Uma das aplicações aclamadas para agricultura com a plataforma apresentada reside na estação

de sensoriamento de temperatura e umidade. Em Fitopatologia, essas estações seriam muito úteis para monitoramento microclimáticos em áreas de produção vegetal. Dados microclimáticos são determinantes para o controle de pragas e doenças, havendo uma vasta literatura clássica referente aos sistemas de previsão e alertas de doença impulsionados por dados climáticos de temperatura, percentual de umidade relativa do ar (% UR), molhamento foliar etc (Campbell; Madden, 1990; Cavalcanti, 2021).

No contexto acima, o objetivo do presente trabalho foi o de apresentar proposição para desenvovimento de uma rede do tipo *peer-to-peer* (P2P) com dispositivos ESP8266 (ESP-1 e ESP-12e), baseada na recepção de um sinal simples de *wi-fi* em roteador central (provedor) por repetição desse sinal para captação de dados de sensores também construídos com ESP8266. Esses sensores devem estar distribuídos em área produtiva (vinhedo) próxima (até 200 m) do roteamento do sinal, transmitindo os dados para um servidor de vinhedo que, por sua vez, transmite o sinal para um receptor que se conecta ao roteador/modem do provedor do serviço.

A intenção do trabalho é, também, o de divulgar e incentivar que pessoas com interesse em agricultura possam arriscar construir seus próprios dispositivos, cujos insumos estão fartamente disponíveis no varejo a um custo bastante acessível a todos, com as partes de programação do controlador imersas em um ambiente *open source* e *maker* (como por exemplo, o *Project Hub* da plataforma Arduino) e de conectividade total.

Material e Métodos

O projeto proposto entrega uma sugestão de desenho de sistema de captação de dados microclimáticos no vinhedo por microestações (MSs) compostas cada qual de um ESP-01 (ou ESP-12 e ligado a um DHT11 ou DHT22, no caso de maior poder de investimento) conectado a um módulo ESP-01S com um sensor DHT11 capaz de registrar temperatura (°C) e o percentual de umidade relativa do ar (% UR). O sistema sugere o uso de ESP-12e (NodeMCU) como: servidor de recepção de dados das MSs e servidor de entrega de dados (repetidor) cujo sinal *wi-fi* pode ser detectado por um cliente (outro repetidor ou um computador com *wi-fi*). A proposição pode ser capaz de usar um “sinal de internet” (*wi-fi*) bastante popular em domicílios da zona rural que possuam campos de produção tão próximos quanto possível da

instalação provedora do serviço, para que se minimize o uso de repetidores na área de produção e se reduzam quedas no sistema.

Microestação para aquisição de dados de temperatura e percentual de umidade relativa do ar

A microestação (MS) é a composição de dois dispositivos: o ESP-01 conectado ao seu módulo DHT-11 (ESP-01S). Essa composição de hardware tem um marcante baixo custo e número grande de fornecedores de importação. Inicialmente, o ESP-01 deve ser preparado a receber programação por um ambiente Arduino de compilação de instruções passadas abaixo (Figura 1), e não por comandos AT que vêm como *default* no dispositivo. Para isso, é necessário baixar um gravador de *firmware* (*flasher*) e proceder a substituição da configuração *default*. O *flasher* recomendado foi o 'ESP8266flasher.exe' que funcionou perfeitamente com a rom 'nodemcu_20150213.bin'. Todos esses módulos podem ser encontrados gratuitamente na internet por requisições GitHub ou em sites de busca. De posse desse software, procede-se à gravação do *firmware* propriamente dita. Para isso, é necessário ligar o ESP-01 *default* em um sistema de gravação FTDI para TTL e RS232 *Serial* que pode ser obtida por um adaptador USB de baixo custo. O adaptador deve vir com um circuito integrado (CI) UART-USB CH340, sendo reconhecido no computador como uma porta serial (COM). Recomenda-se usar o Sistema Operacional (SO) *Microsoft Windows* (acima da versão 7) devido ao reconhecimento automatizado de portas seriais. É possível usar um SO *Linux*, no entanto, a configuração do CH340 e das portas COM são mais trabalhosas nesse SO. Há, entretanto muita documentação disponível na internet para orientar sobre esses procedimentos de transferência Serial no Linux. No caso do presente trabalho, testes feitos em Ubuntu 18.04 LTS (Debian).

Após a gravação do *firmware* de trabalho NodeMCU na ESP-01, é necessário procedimento de configuração da IDE Arduino para a transferência do código por uma porta COM. Recomenda-se que a detecção da porta COM possa ser feita pelo cliente de 'gerenciador de dispositivos' do próprio *Windows*. Deve-se, então, proceder ao carregamento no Ambiente de Desenvolvimento Integrado (IDE) Arduino do controlador de

placas ESP8266 Boards (3.0.2) – Generic ESP8266 Module (ou NodeMCU 0.9, para ESP-12) que está na biblioteca *online* da própria IDE.

A indicação de fontes de energia para os dispositivos ESP-01 propostos abaixo, vai depender da escolha de quem irá executar o projeto. O protótipo feito na Embrapa Uva e Vinho funcionou de modo estável com uma bateria do tipo célula de Li-ion 18650, 6800mah 3.7v, recarregável.

```
// Microestacao MS (ESP-01 ou ESP-12e);
// Modificado de exemplo da biblioteca ESP8266Wifi.h;
// Modificado do projeto datalogger DHT11 - Fabio R Cavalcanti;

// 1) Instanciando bibliotecas;
#include <EEPROM.h>
#include <ESP8266WiFi.h>
#include "Statistic.h"
#include "DHTesp.h"

// 2.1) Define os pinos;
#define       LedInterno      1        // ESP-01;

// 3) Variaveis globais;
const String  clienteID = "MS01";      // Microestacao 01;
float ua, ta;

// 4) Variaveis de autenticacao - WIFI e WIFI Module Mode e IP fixo para o
cliente1;
char ssid[] = "Servervin";
char pass[] = "fitopatologia";
// Definicoes para funcao e porta para WIFI;
int Porta  = 9001;
IPAddress Servervin(192,168,4,1);
WiFiClient Cliente;

// 5) Instanciando demais objetos e variaveis;
DHTesp dhtA;

Statistic ColetaT;

// ############## SETUP #############;
void setup()
  {
    Serial.begin(115200);
    pinMode(LedInterno, OUTPUT);
    digitalWrite(LedInterno, !LOW);
    // Iniciando conexao;
    Serial.print("Conectando a um servidor...");
    if(WiFi.status() == WL_CONNECTED)
    {
      WiFi.disconnect();
      WiFi.mode(WIFI_OFF);
      delay(50);
    }
    WiFi.mode(WIFI_STA);
    WiFi.begin(ssid, pass);
    Serial.println("... Conectando ao servidor <" + WiFi.SSID() + ">");
    // Conecta..;
    Conecta();
    digitalWrite(LedInterno, !HIGH);
    Serial.println("< Dispositivo Cliente [MICROESTACAO] conectado! >");

    // Dados de conexao;
    Serial.println("Conectado a : " + String(WiFi.SSID()));
    Serial.println("Sinal : " + String(WiFi.RSSI()) + " dBm");
    Serial.print("Endereco do Servidor : "); Serial.println(Servervin);
    Serial.print("Porta numero : "); Serial.println(Porta);
    Serial.print("MAC : "); Serial.println(String(WiFi.macAddress()));
    Serial.print("IP      do      dispositivo      Cliente      :      ");
Serial.println(WiFi.localIP());

    // 2.2) Define os pinos(DHT11);
    //dhtA.setup(04, DHTesp::DHT11);
    dhtA.setup(05, DHTesp::DHT11);
    //dht8.setup(05, DHTesp::DHT11);

    // Conectando o presente dispositivo como um Cliente;
    RequisitaServidor();
  }

// ############## LOOP ##############;
void loop()
  {
    DepuraDHT();
    Serial.println("<" + clienteID + "> " + "Temp:" + ta + "/ %UR:" + ua +
".");
    Cliente.println("<" + clienteID + "> " + "Temp:" + ta + "/ %UR:" + ua +
".");
    Cliente.flush();
    Serial.println("Dados acima enviados para o servidor.");

    delay(1000);

  }

// ############## Subrotinas #############;
void Conecta()  {
    while(WiFi.status() != WL_CONNECTED)
    {
      for(int f=0; f < 10; f++)
      {
        digitalWrite(LedInterno, !HIGH);
        delay(300);
        digitalWrite(LedInterno, !LOW);
        delay(300);
        Serial.printf("Erro tipo: %d\n", WiFi.status());
      }
      Serial.println("");
    }
    return;
  }

void RequisitaServidor()  {
    // Certifica que esta desconectado;
    Cliente.stop();

    // Se conectado, envia mensagem;
    if(Cliente.connect(Servervin, Porta))
    {
      Serial.println    ("<" + clienteID + "- CONECTADO!>");
      Cliente.println ("<" + clienteID + "- CONECTADO!>");
    }
    return;
  }

void Blink() {
  for (int f = 0; f < 4; f++) {
    digitalWrite(LedInterno, !digitalRead(LedInterno));
    delay(50);
  }
  return;
}

void DepuraDHT() {
  while (ColetaT.count() < 4 || ColetaT.pop_stdev() > 1.00 || isnan(ta) ==
true )  {
  ta = dhtA.getTemperature();
  ua = dhtA.getHumidity();
  ColetaT.add(ta);
  delay(dhtA.getMinimumSamplingPeriod());
  Serial.println(ta);
  Blink();
  }
  ta = ColetaT.average();
  Serial.println("-----"); Serial.println(ta);
  ColetaT.clear();
  return;
}
```

Figura 1. Microestação. *Script* em Ambiente de Desenvolvimento Integrado (IDE) Arduino contendo um método para um ESP-01 obter dados de temperatura e umidade relativa do ar do DHT-11, filtrar esses dados e servir como cliente de transmissão para o servidor da área produtiva (servevin). O código pode ser transferido para o dispositivo após configuração do *firmware* e subsequente configuração da IDE Arduino.

Servidor da área produtiva (servervin)

Sugere-se que este dispositivo seja montado sobre um ESP-12e. Este dispositivo (servervin) vai ter a função de receber os dados de todas as MSs ativas para o monitoramento, distribuídas na área produtiva. O código de transferência para que o ESP-12e atue como um servervin está apresentado na Figura 2.

```
// Servervin ESP-12e. ESP-01 possui limitacoes;
// Modificado de exemplo da biblioteca ESP8266WiFi.h;
// Modificado para projeto datalogger DHT11 - Fabio R Cavalcanti;
// ** Ordem de boot: 1) Servervin (AQUI); 2) MSs; 3) repetidor.

// 1) Instanciando bibliotecas;
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>

// 2) Define os pinos;
#define     LedInterno      2     // ESP-12e;
#define     MAXMS           6

// 3) Autenticacao wifi e ponto de acesso (servidor para clientes esp);
char ssid[] = "Servervin";
char pass[] = "fitopatologia";

// 4) Variaveis globais;
String Frase, Mensagem = "";
char buff[30];
char buffvar[] = "0.00";
String buffvarS = "";
String buffMS[MAXMS];
float ms;
boolean bandeira[] = {LOW, LOW, LOW, LOW, LOW, LOW, LOW};
int novocliente;
int cont[] = {0, 0, 0, 0};

// 5) Caracteristicas do Servidor;
ESP8266WebServer server(80);

// 6) Caracteristicas do ponto de acesso (Servidor para clientes esp);
WiFiServer  PontoAcesso(9001);
WiFiClient  MSs[MAXMS];

// ############## SETUP ##############;
void setup()  {
    Serial.begin(115200);
    // Setting The Mode Of Pins
    pinMode(LedInterno, OUTPUT);

    // Configura o Ponto de Acesso;
    ConfiguraWifi();

    server.begin();

  }

// ############## LOOP ##############;
void loop()  {
    while (bandeira[1] != HIGH || bandeira[2] != HIGH || bandeira[3] != HIGH)
    {
      ChecaClientesDisponiveis();
      TrataMensagens();
    }
    // Prepara Mensagem (Frase) para transfencia;
    if (Frase = "") {
      for (int f = 0; f < (novocliente + 1); f++)
      {
        bandeira[f] = LOW;
        Frase = Frase + buffMS[f];
      }
    }

    // Configurar rota para manipular a raiz "/";
    server.on("/", HTTP_GET, transmiteTCP);

    server.handleClient();

    Serial.println(Frase);

    WiFiClient client = PontoAcesso.available();

  }

// ############## Subrotinas ##############;
void ConfiguraWifi()  {
    // Interrompe qualquer wifi previo;
    WiFi.disconnect();
    // Configurando o modo wifi;
    WiFi.mode(WIFI_AP_STA);
    Serial.println("Modo wifi: Ponto de Acesso.");
    // Iniciando o Ponto de Acesso/Servidor;
    WiFi.softAP(ssid, pass);
    Serial.println("WIFI < " + String(ssid) + " > ... Iniciado.");
    delay(500);
// Capturando e imprimindo configuracoes do Servidor/Ponto de Acesso;
    IPAddress IP = WiFi.softAPIP();
    // IPAddress IP(192, 168, 1, 1);
    Serial.print("IP do Ponto de Acesso : "); Serial.println(IP);
    Serial.print("MAC      do      Ponto      de      Acesso      :      ");
Serial.println(String(WiFi.softAPmacAddress()));
    // Iniciando Ponto de Acesso/Servidor;
    PontoAcesso.begin();
    //PontoAcesso.setNoDelay(true);
    Serial.println("Ponto de Acesso iniciado.");
    return;
  }

void ChecaClientesDisponiveis()  {
    if (PontoAcesso.hasClient())  {
      // Read LED0 Switch To Low If High.
      if(digitalRead(LedInterno) == HIGH) digitalWrite(LedInterno, LOW);

      for(uint8_t i = 0; i < MAXMS; i++)    {
        //Encontra clientes ligados no wifi mas descontectados;
        if (!MSs[i] || !MSs[i].connected())    {
          // Confere se ha previamente clientes online para interromper;
          if(MSs[i])  {  MSs[i].stop(); }
          // Verifica se os clientes estao conectados ao servidor;
          if(MSs[i] = PontoAcesso.available())
          {
            novocliente = (i+1);
            Serial.println("Novo Cliente: " + String(novocliente));
          }
          // Continua procurando clientes;
          continue;
        }
      }
      //Sem pontos (clientes) livres/desconectados para rejeitar;
      WiFiClient MSs = PontoAcesso.available();
      MSs.stop();
    }
    else
    {
      // Blink enquanto nao houver clientes disponiveis ligados no wifi;
      digitalWrite(LedInterno, HIGH);
      delay(250);
      digitalWrite(LedInterno, LOW);
      delay(250);
    }
  }

void TrataMensagens()  {
    //baixa/recebe os dados enviados por clientes;
    for(uint8_t i = 0; i < MAXMS; i++)
    {
      if (MSs[i] && MSs[i].connected() && MSs[i].available())
      {
          while(MSs[i].available())
          {
            Mensagem = MSs[i].readStringUntil('\r');
            MSs[i].flush();
            Mensagem.toCharArray(buff, 30);
            buffvarS = Mensagem.substring(4,5);
            buffvarS.toCharArray(buffvar, 4);
            ms = atof(buffvar);
            EspecificaDadoCliente();
          }
        }
      }
    }

void  EspecificaDadoCliente()  {
      for (int f = 1; f < (novocliente + 1); f++)
      {
        if (ms == f) {
              cont[f] = cont[f] + 1;
              buffMS[f] = buff;
              Serial.println(buffMS[f] + " Cont:" + cont[f]);
              if (cont[f] == 2) { bandeira[f] = HIGH; cont[f] = 0; }
              }
        }
      return;
    }

//###### Transmite dado: Ponto de acesso -> Repetidor ######;
void transmiteTCP() {
  String pararepetidor = Frase;
  server.send(200, "text/plain", pararepetidor);
}
```

Figura 2. Servidor da área produtiva (servervin). *Script* em IDE Arduino contendo um método para um ESP-12e funcionar como um servidor coletor de dados de temperatura e umidade das MSs, tendo as mesmas como clientes. O código pode ser transferido para o dispositivo após configuração do *firmware* e subsequente configuração da IDE Arduino.

Servidor de entrega de dados (repetidor)

Da mesma forma que os outros nós do sistema acima descritos, o 2.3. pode ser montado em um ESP-12e, e vai ter a função de receber os dados do servidor (ponto de acesso) da área produtiva (servervin), ser capaz de ser detectado pelo roteador/modem da provedora de serviços e, por fim, transmitir (como cliente) para uma instância http:// os dados captados no vinhedo e transferidos para o servervin pelas MSs. O código de transferência para que o ESP-12e atue como esse repetidor está apresentado na Figura 3.

```
// Servervin ESP-12e. ESP-01 possui limitacoes;
// Modificado de exemplo da biblioteca ESP8266WiFi.h;
// Modificado para projeto datalogger DHT11 - Fabio R Cavalcanti;
// ** Ordem de boot: 1) Servervin (AQUI); 2) MSs; 3) repetidor.

// 1) Instanciando bibliotecas;
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
#include <ESP8266HTTPClient.h>

//  2)  Autenticacao  wifi,  ponto  de  acesso  (servidor  para  chamadas
browser/Postman);
const char* ssid = "Servervin";
const char* password = "fitopatologia";
const char* serverIP = "http://192.168.4.1";
//
const char* ssid_repetidor = "RepetidorESP";
const char* password_repetidor = "SenhaRepetidor";

String Frase_repetidor = "Aguardando atualização...";

// 3) Caracteristicas servidor e ponto de acesso;
ESP8266WebServer server(80);
WiFiClient client;

// ############## SETUP ##############;
void setup() {
  Serial.begin(115200);

  // Conectar-se à rede Wi-Fi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Conectando ao WiFi...");
  }
  Serial.println("Conectado ao WiFi");

  // Configurar rota para a página HTML
  server.on("/", HTTP_GET, [](){
    String pagina = "<html><body>";
    pagina += "<h1>TXUIU - ESP8266 WebServer</h1>";
    pagina += "<p>" + Frase_repetidor + "</p>";
    // Aqui se pode desenvolver uma pagina simples para apresentacao;
    pagina += "</body></html>";
    server.send(200, "text/html", pagina);
  });

  // Inicializar o ponto de acesso (AP) do repetidor
  WiFi.softAP(ssid_repetidor, password_repetidor);

  server.begin();
}

// ############## LOOP ##############;
void loop() {
  // Coloque qualquer código adicional aqui, se necessário

  solicita();
  Serial.println(Frase_repetidor);
  delay(3000);

}

// ############## Subrotina ##############;
void solicita() {
// Fazer uma solicitação HTTP GET para o servidor
  HTTPClient http;
  http.begin(client, serverIP);
  int httpCode = http.GET();

  if (httpCode == HTTP_CODE_OK) {
    String resposta = http.getString();
    Serial.println("Resposta do servidor:");
    Serial.println(resposta);
    Frase_repetidor = resposta;
  } else {
    Serial.println("Falha na solicitação HTTP para o servidor");
    Serial.println("Falha: " + String(httpCode));
  }
  http.end();
 }
```

Figura 3. Repetidor para o provedor. *Script* em IDE Arduino contendo um método para um ESP-12e funcionar como um servidor que transfere os dados coletados pelo 'servervin' para um cliente (outro ponto de acesso ou notebook capaz de acessar http://) cuja conexão está disponível pelo sinal de *wi-fi* do ponto de acesso, detectável por esse ESP-12e.

Resultados e Discussão

Na Agricultura do final dos anos 90, um grande salto tecnológico foi dado com a incorporação nos deslocamentos em campo e processos de informações baseados em sistemas de informação geográfica e georreferenciamento de

GPS e sensores mais simples e baratos. Um novo termo foi cunhado a partir disso, a Agricultura de Precisão (AP). Atualmente, uma AP está fundamentada em características bem específicas: a) escala da área produtiva; b) escala de tempo; c) variabilidade espacial e temporal; d) natureza da informação (responsividade, tempo-real, *offline* etc); e) custo de entrega (custo/benefício) e f) conectividade (Ahmad; Nabi, 2021).

Recentes avanços em conectividade rural permitiram uma crescente busca por aplicações de 'internet das coisas' (IoT). Com a IoT, há a possibilidade de transferência de dados em tempo-real para bancos de dados centrais e estes, por sua vez, impulsionarem interfaces de aplicação computacional (APIs) associadas ao manejo agrícola em plataformas de microsserviços (Thakur et al., 2019), tais plataformas sendo meramente repositórios de ferramentas digitais capazes de entregar informações para racionalizar a atividade agrícola. Ao tempo que a IoT participa da AP, é considerada parte integrante da atual abordagem em AP, conhecida como agricultura inteligente ou *Smart Farming* (SF) (Wolfert et al., 2017). A IoT na agricultura é definida por certos atributos: a) interconectividade; b) compatibilidade com interfaces (heterogeneidade); c) mudanças dinâmicas; d) adoção maciça de dispositivos/sensores; e) grandes *datasets*; f) pré-processamento dos dados sensoriados; g) reduzida dependência da intervenção humana; h) alta eficiência energética etc (Ahmad; Nabi, 2021).

No presente trabalho, foi desenvolvida uma proposta para desenvolvimento de um sistema de rede P2P (*peer-to-peer*) com dispositivos ESP8266 para aproveitamento de sinal *wi-fi* disponível por provedor de serviços em residência familiar próxima a área de vinhedos passível de ser atacada por doenças. A intenção do presente estudo é de apresentar uma ferramenta moldada à realidade típica de produção de uvas na Serra Gaúcha, baseada na agricultura familiar e vinhedos circunstantes à residência da família de viticultores (Lazzarotto et al., 2022). O monitoramento do microclima na área produtiva é fundamental para impulsionar sistemas de previsão e alerta de doença (Cavalcanti, 2021). O sistema foi desenvolvido em uma plataforma NodeMCU também compatível com Arduino (linguagem baseada em C/C++) para programar os ESPs-01 e 12e. Um protótipo P2P bem preliminar demonstrou estabilidade durante testes de 24 horas, com as três estações ESP-01S distantes 9,5 m umas das outras e todas 9,5 m do servervin,

responsável por receber os dados das estações, depurar e enviar para o 'servidor de entrega de dados' (repetidor) (Figura 4). Nesse caso, há o potencial de aumentar distâncias simplificando a rotina de repetição escrita para este trabalho através de outros pontos de acesso na rede P2P montada, tanto das MSs para o servervin quanto do servervin em direção ao repetidor para outros pontos de acesso ou servidor web.

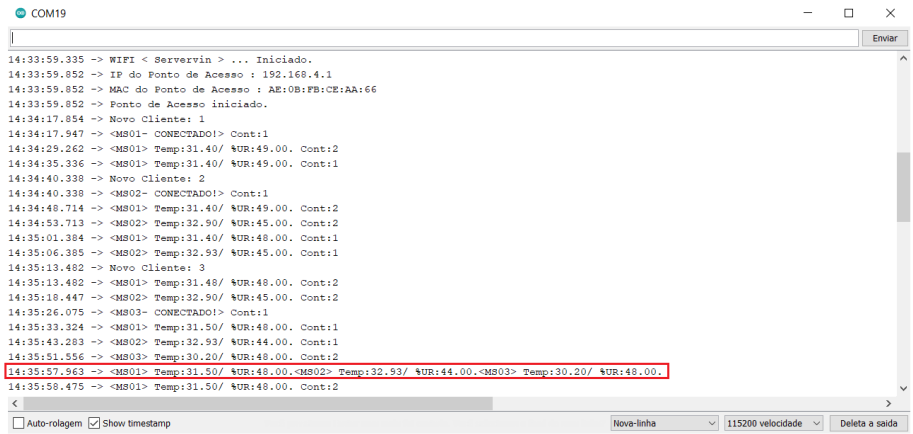


Figura 4. Recorte do monitor da IDE Arduino com a montagem do ponto de acesso do servidor da área produtiva (servervin) e conexão com três clientes (MSs). O servervin recebe três leituras (já tratadas/depuradas) de cada MS para construir uma *string* a ser enviada para o repetidor (box vermelho).

Nessa proposta, os dados são tratados na própria MS cujo código impede o envio para o servervin de valores aberrantes ou não numéricos (NaN) (Figura 1). Por isso, há um contador no servervin, na função *EspecificaDadoCliente*(), para oferecer uma 2ª triagem dos dados nesta instância (Figura 2). De posse da 2ª contagem de dados enviados pelas MSs, o servervin vai construir uma *string* (Frase) com os dados das três MSs (caso do protótipo testado) e enviar para o 'servidor de entrega de dados' (repetidor).

Plataformas como Arduino, Raspberry Pi, TelosB, SunSPOT, iSense, IRIS etc são adotadas pela SF associadas a sensores de marcas aclamadas como Qualcomm, Intel, Meter, Sensirion etc e sensores genéricos, mais baratos e cada vez mais confiáveis. Provavelmente a mais popular, a Arduino é

uma plataforma italiana de uso livre que está popularizando rapidamente as iniciativas domésticas de pessoas comuns querendo desenvolver dispositivos eletrônicos simples e úteis de modo muito facilitado (*Creative Commons Attribution-Sharealike* 3.0 *License*). Usuários em todo o mundo se organizam em comunidades baseadas na filosofia "Faça você mesmo" (do inglês, "*Do It Yourself* – DIY"), também chamadas de *maker*, com apoio da própria Arduino (*Project Hub*, https://projecthub.arduino.cc/). Com vasta partilha de informações e *templates* pela internet, essas comunidades conseguem abordar temas que vão desde o sensoriamento do ambiente, até criação de controle automatizado, controle de máquinas (relés) e até robótica (Kutter et al., 2011; Aqeel-Ur-Rehman et al., 2014; Mesas-Carrascosa et al., 2015; Thakur et al., 2019).

Como acima comentado, nos testes feitos com o protótipo alimentado por baterias de lítio (Li-ion 18650, 6800mah 3.7v) o monitoramento durou apenas 24 horas até o início das falhas dos sensores indicada por geração de valores NAN que são depurados e interrompidos logo nas MSs, na função *DepuraDHT*(). Nesses casos, a MS não chega a se desconectar, mas represa dados e não envia para o servervin. Durante o tempo médio dos ensaios, entretanto, o sistema P2P funcionou corretamente, mantendo o fluxo de dados seguindo o fluxo MSs, servervin e repetidor até a entrega das informações para, no caso do presente desenvolvimento de protótipo, um servidor http:// (repetidor) (Figura 5). De fato, a autonomia energética mostrada nos testes foi bastante aquém do que se esperaria de um produto acabado. No entanto, o presente trabalho apresenta uma proposição inicial para desenvolvimentos, a partir do qual, inúmeras melhorias podem (e devem) ser efetivadas até um protótipo funcional.

Em outras palavras, a intenção desse trabalho é :

- Propor um modelo de disposição e programação para uso de dispositivos de IoT (ESP8266) para levar conectividade à área de produção vegetal e fazer uso de sensoriamento *online* de baixo custo a ser explorado e desenvolvido pela comunidade.
- Ofertar um modelo a ser aprimorado em questões sobre emissão/recepção de sinal (antena), inserção de funções aos códigos propostos – como função *sleep*, função de administração de aquisição de energia por painel solar etc.

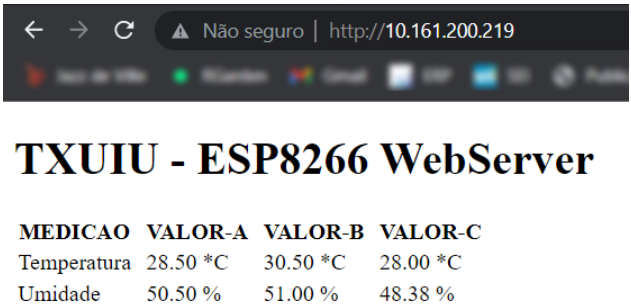


Figura 5. Recorte do monitor de um *browser* acessando um servidor de *backend* (http://) onde um documento de HTML (*frontend*) é executado para entregar os dados de monitoramento em uma chamada de terminal de acesso ao repetidor. Esses dados também poderiam facilmente adaptados a ser entregues por intermédio de *brokers* de MQTT, ou algum intermediário de dados de IoT em servidores web alimentados por repetidor.

Em situação de baixa autonomia, o uso de painéis solares de baixo custo pode fornecer uma fonte alternativa e renovável de energia para uma estação de IoT baseada em ESP8266 com DHT11, ideal para projetos de longa duração. Principalmente, a eficiência energética pode ser maximizada ao incorporar uma função *sleep*() na programação do controlador, colocando-o em um estado de baixo consumo quando não está transmitindo dados. Se assumirmos que o painel solar é um modelo de 10 W e que temos em média cinco horas de sol pleno por dia, isso seria 50 Wh de energia adicionada todos os dias. A bateria Li-ion 18650 de 6800 mAh 3.7 V tem uma capacidade de energia de cerca de 25,16 Wh (6800 mAh x 3.7 V/1000 = 25,16 Wh). Portanto, o painel solar poderia, teoricamente, recarregar a bateria duas vezes ao dia. Com a função *sleep*() implementada, o consumo de energia da MS poderia ser significativamente reduzido. Supondo que a função *sleep*() permite que o dispositivo gaste apenas 10% da energia que consumiria normalmente, isso dobraria a duração da bateria, de 24 horas para 240 horas (ou dez dias). Portanto, teoricamente, com um painel solar de 10 W e a implementação da função *sleep*, o dispositivo poderia funcionar indefinidamente, desde que haja um mínimo de sol todos os dias. No entanto, esta é uma simplificação, pois na realidade haverá perdas de eficiência e variações nas condições solares.

Tentativas subsequentes integrando essas abordagens serão tentadas, na evolução para um protótipo funcional.

De qualquer forma, um dos atrativos para justificar esforços em prototipagem envolvendo os dispositivos de IoT sugeridos é seu baixo custo inicial. A Tabela 1 apresenta uma cotação genérica do material utilizado, em dólares. Por pouco mais de quinhentos reais (cotação de dezembro de 2023) seria possível um investimento para desenvolver um pequeno sistema de monitoramento de microclima em área produtiva próxima a um ponto de internet com roteamento *wi-fi*. É claro que, para manter e desenvolver um pequeno sistema P2P nos moldes do que está sendo proposto, teria de haver um investimento prévio em gravadores FTDI, fontes de 3,3V e 5,0V, protoboards, cabos, bornes, fios, ferramentas etc.

Tabela 1. Cotação de consumíveis básicos usado na proposição de rede P2P com ESP8266 com três estações.

Item	Número de itens	Unidade (US$)	Total (US$)
Módulo *wi-fi* ESP8266 ESP-01	3	3,80	11,04
Módulo *wi-fi* ESP8266 ESP-12e (Nodemcu)	2	6,70	13,44
Adaptador ESP8266 (ESP-01S) com sensor de temperatura e umidade - DHT11	3	3,38	10,14
Bateria 18650 Li-Ion recarregável 3.7 V 3800 mAh Button-top	5	3,98	19,94
Painel solar 10 W	3	15,67	47,03
Suporte para 1 Bateria 18650 Li-ion	5	1,70	8,50
Total			110,09(1)

(1) Cotação do dólar comercial médio no dia 6/12/2023 no mercado brasileiro. Total geral de R$ 541,64.

Nos testes, os conjuntos de MSs, foram acondicionados em *cases* rudimentares específicos. Podem ser usadas as sugestões de *scripts* escritos para OpenSCAD que é *open source* e possui uma linguagem para modelagem 3D de sólidos, renderização e capacidade de exportar em padrão de estereolitografia (arquivo .stl) (Figura 6). Esse padrão é aceito em quase todos os fatiadores de modelos 3D *open source*. Também, podem ser utilizados modelos 3D prontos e disponibilizados na base de dados

Thingiverse. No caso do presente protótipo um exemplo de modelo (https://www.thingiverse.com/thing:4355656) pode ser usado, alargado para encaixe da bateria e impresso com PLA ou outro filamento em impressora 3D.

Caixa

```
module caixa() {
    difference() {
        cube([90, 50, 25]);
        translate([2, 2, 0])
        cube([86, 46, 22]);

        // suportes internos
        suportes_internos();
    }
}

module suportes_internos() {

    translate([9, 2, 15])
    cube([20, 20, 8]);

    translate([61, 2, 15])
    cube([20, 20, 8]);
}

caixa();
suportes_internos();
```

Tampa

```
module tampa() {
    difference() {
        cube([94, 54, 7]);
        translate([2, 2, 0])
        cube([90, 50, 4]);


        // Add grid pattern
        for (x = [2 + 10: 10: 94 - 12]) {
            for (y = [2 + 10: 10: 54 - 12]) {
                translate([x, y, 0])
                cube([5, 5, 7]);
            }
        }
    }
}



tampa();
```

Figura 6. Segmento de *scripts* em código OpenSCAD contendo sugestões de modelos para montagem de caixa e tampa para acondicionamento da proposta de estação com ESP8266 (ESP-01S), a serem fatiados em impressão 3D.

O presente protótipo de monitoramento P2P proposto pode ser utilizado como fonte de dados para impulsionar um sistema de alerta de doença que trabalhe com IoT e monitoramento de microclima usando vários pontos de captação de dados. Uma aplicação natural seria justamente um sistema que usasse dados provenientes de diferentes posicionamentos geográficos dentro de uma área de produção vegetal como o Módulo de Alerta (georreferenciado) de doença por *Heat Map*, da Embrapa, o MAHM, tendo como estudo de caso o míldio da videira (Cavalcanti, 2021).

Naturalmente, os esforços foram pensados tendo-se em mente um sistema P2P como uma ferramenta de aquisição de dados para impulsionar um microsserviço que use o Embrapa/MAHM para gerar padrões de dados de favorabilidade e alerta de doença. No entanto, evidências recentes mostram que, para o MAHM, as MSs podem ser dispostas mais distantes umas das outras para produzir mapas temáticos contrastantes (Cavalcanti, 2021), coisa que o sistema P2P tentado neste trabalho não conseguiria suportar. De

todo o modo, microsserviços já podem requisitar a API que está encapsulada na plataforma AgroAPI da Embrapa[1] e um sistema de IoT como a rede P2P sugerida neste trabalho poderia compor uma ferramenta para monitoramento e alerta de míldio em pequena escala.

Por fim, dados do P2P poderiam ser transferidos para um servidor *broker* de MQTT, também conhecidos como *clouds*. Um *Message Queuing Telemetry Transport* (MQTT) é um protocolo de comunicação que tem como foco a (IoT). O MQTT funciona sobre um protocolo TCP/IP como um sistema máquina-a-máquina (M2M) também baseado em comunicação cliente/servidor. O MQTT se popularizou pela simplicidade, baixo consumo de dados e pela possibilidade comunicação bilateral. Alguns são gratuitos. Armazenamento de dados em servidores/administradores e outras plataformas de *cloud*, como, por exemplo, *Amazon* (AWS), Google, Hadoop, Oracle são fundamentais para que dados prospectados por IoT sejam aproveitados para um fim prático (Ahmad; Nabi, 2021).

Conclusões

- Neste trabalho, foi entregue uma sugestão para desenvolvimento de rede de sensoriamento utilizando estações montadas com ESP8266 (ESP 01 e 12e), DHT11 (ESP-01S) e pontos de acesso, visando ao monitoramento de microclima em vinhedos próximos a fontes de sinal wi-fi.
- Essa proposição perfaz uma base de trabalhos para desenvolvimento de hardware de baixo custo para monitoramento do microclima de vinhedos próximos, permitindo transferência de dados a aplicações digitais para tomada de decisão, como por exemplo, sistemas de alerta de doença.
- Mesmo com baixa autonomia energética do protótipo-teste, as estações conseguiram transferir dados com precisão, permitindo vislumbrar o enorme potencial da IoT de baixo custo para desenvolvimentos DIY no setor vitivinícola.

[1] APOLINÁRIO, D.; BARBOSA, L. F; CAVALCANTI, F. R. **Desenvolvimento de API RESTful para alerta de patógenos em Python com Django Framework**. A ser publicado como Série Comunicado Técnico, Embrapa Agricultura Digital, Campinas, SP.

Referências

AHMAD, L.; NABI, F. **Agriculture 5.0:** Artificial Intelligence, IoT and Machine Learning. CRC Press, 2021. 243p.

AQEEL-UR-REHMAN; ABBASI, A.Z.; ISLAM, N.; SHAIKH, Z.A. A review of wireless sensors and networks' applications in Agriculture. **Computer Standards & Interfaces**, v. 36, n. 2, p. 263-270, Feb. 2014. DOI 10.1016/j.csi.2011.03.004.

BAKTHIARI, A. A.; HEMATIAN, A. Precision farming technology, opportunities and difficulty. **International Journal of Science & Emerging Technologies with Latest Trends**, v. 5, n. 1, p. 1–14, 2013.

CAMPBELL, C. L.; MADDEN, L. V. Introduction to plant disease epidemiology. New York: John Wiley & Sons, 1990. 532 p. ISBN 9780471832362.

CAVALCANTI, F. R. **Algoritmo (MAHM) para alerta georreferenciado de doença em redes de sensoriamento IoT de microclima: calibração e teste de um método para míldio, em dois vinhedos**. Bento Gonçalves, RS: Embrapa Uva e Vinho, agosto 2021. 25 p. (Embrapa Uva e Vinho. Circular Técnica, 124). Disponível em: http://www.infoteca.cnptia.embrapa.br/infoteca/handle/doc/1134000. Acesso em: 21 jun. 2023.

CHAKRABORTY, S.; DAS, P.; PAL, S. IoT Foundations and Its Application. In: PATTNAIK, P. ; KUMAR, R.; PAL, S.; PANDA, S, (Eds). **IoT and Analytics for Agriculture.** Studies in Big Data, v. 3. Singapore: Springer, 2020. pp. 51–68. DOI https://doi.org/10.1007/978-981-13-9177-4_3.

FAO. Food and Agriculture Organization of the United Nations. International Forum on Innovation in Agri-Food Systems to Achieve the SDGs, 2020, Riyadh, Saudi Arabia. **Anais…** FAO Regional Office for Near East and North Africa, de 17 a 17 march 2020. Disponível em: http://www.fao.org/neareast/events/agri-food-systems-innovation/en/. Acesso em: 21 mar. 2023.

KUTTER, T.; TIEMANN, S.; SIEBERT, R.; FOUNTAS, S. The role of communication and co-operation in the adoption of Precision Farming. **Precision Agriculture**, v. 12, n. 1, p. 2-17, Feb. 2011. DOI 10.1007/s11119-009-9150-0.

LAZZAROTTO, J. J.; FIORAVANÇO, J. C.; TAFFAREL, J. C. **Parâmetros para investimentos na produção orgânica e industrialização de uvas americanas e híbridas com alto padrão tecnológico**. Bento Gonçalves, RS: Embrapa Uva e Vinho, set. 2022. (Embrapa Uva e Vinho. Documentos, 133). 40p. Disponível em: http://www.infoteca.cnptia.embrapa.br/infoteca/handle/doc/1146270. Acesso em: 21 jun. 2023.

MESAS-CARRASCOSA, F. J.; VERDÚ SANTANO, D.; MERONO, J. E.; SANCHEZ DE LA ORDEN, M.; GARCIA-FERRER, A. Open Source hardware to monitor environmental parameters in precision agriculture. **Biosystems Engineering**, v. 137, p. 73-83, Sept. 2015. DOI 10.1016/j.biosystemseng.2015.07.005.

THAKUR, D.; KUMAR, Y.; KUMAR, A.; SINGH, P. K. Applicability of wireless sensor networks in Precision Agriculture: A review. **Wireless Personal Communications, v. 107, n. 1, p. 471-512, July 2019.** DOI 10.1007/s11277-019-06285-2

TRAN, V.; NGUYEN, N. V. The concept and implementation of precision farming and rice integrated crop management systems for sustainable production in the twenty-first century. **International Rice Comission Newsletter**, v. 55, p. 91–102, 2006. Disponível em: https://www.fao.org/3/a0869t/a0869t00.htm. Acesso em: 21 jun. 2023.

WOLFERT, S.; GE, L.; VERDOUW, C.; BOGAARDT, M. J. Big data in Smart Farming – A review. **Agricultural Systems**, v. 153, p. 69–80, May 2017. DOI https://doi.org/10.1016/j.agsy.2017.01.023.

ZIMMERMAN, C. **The Five 'R's' of Precision"**. Nov. 2008. [Online]. Disponível: http://precision.agwired.com/2008/11/11/the-five-rs-of-precision/. Acesso: 21 June 2023.

Embrapa

Uva e Vinho

MINISTÉRIO DA AGRICULTURA E PECUÁRIA

GOVERNO FEDERAL
BRASIL
UNIÃO E RECONSTRUÇÃO

CGPE 18360