FACULDADE SENAI DE TECNOLOGIA MECATRÔNICA

EXTRAÇÃO E PERSISTÊNCIA DE DADOS DE EQUIPAMENTOS UTILIZANDO MICROCONTROLADORES

EQUIPMENT DATA EXTRACTION AND PERSITENCE USING MICROCONTROLLERS

Wilson de Oliveira Andrade[1]
Paulo Sebastião Ladivez[2]
José Roberto dos Santos [3]
Thiago Tadeu Amici [4]

RESUMO

Este trabalho visa desenvolver uma plataforma de baixo custo, com o objetivo de centralizar as informações que são extraídas de equipamentos dos setores operacionais fabris e persisti-las, em tempo real, em um banco de dados. Gerando, assim, um histórico de eventos destes equipamentos para uma posterior consulta. Fazendo com que os responsáveis pela produção e tomadores de decisão possam ter uma visão mais precisa de todos seus equipamentos, assim, integrando *hardware*, *software* e Internet das Coisas, IoT. O foco será voltado, especificamente, para pequenas e médias empresas, mas poderá ser adaptado conforme necessário, pois será utilizado, em sua maior parte, *hardware* e *software* no modelo de código aberto.

ABSTRACT

This work aims to develop a low-cost platform, with the objective of centralizing the information that is extracted from equipment in the manufacturing operational sectors and persisting it, in real time, in a database. Thus, generating a history of events of this equipment for later consultation. So, those responsible for production and decision makers to have a more accurate view of all their equipment, integrating hardware, software and Internet of Things IoT. The focus will be specifically targeted at small and medium businesses, but it can be adapted as needed as it will be mostly used hardware and software in the open source model.

[1]Tecnólogo em Análise e Desenvolvimento de Sistemas. E-mail: andrade.wilson01@gmail.com
[2]Professor da Faculdade SENAI de Tecnologia Mecatrônica. E-mail: paulo.ladivez@sp.senai.br
[3] Especialista em Segurança da Informação da Faculdade SENAI de Tecnologia Mecatrônica.
E-mail: joseroberto @sp.senai.br
[4] Mestre em Automação e Controle de Processos da Faculdade SENAI de Tecnologia Mecatrônica.
E-mail: thiago.amici@sp.senai.br

1 INTRODUÇÃO

Com a necessidade sempre crescente de se produzir de modo mais eficaz e eficiente possível, a indústria tem a necessidade de se aperfeiçoar constantemente e muitas vezes se expandir para outras áreas não industriais. Conforme Schwab (2016) as tecnologias digitais, por exemplo, tem se tornado uma constante não só no mundo industrial, mas em diversas áreas como a da saúde, tais tecnologia digitais facilitam a extração, obtenção e tratamento de dados de forma sistemática.

De acordo com Hernandes (2011), sistemas de coleta automática e em tempo real de dados no chão de fábrica, refletem em um aumento da produtividade em até 10%, fato que pode tornar a empresa mais competitiva no cenário mundial.

As empresas utilizam frequentemente, equipamentos defasados no quesito informação pois, muitas vezes a troca deles não é viável financeiramente. Equipamentos modernos já são projetados para fornecer uma gama de informações de produção, temperatura, umidade entre outros, mas são muito caros em sua maioria. Assim, tomar a decisão de trocar todo parque de maquinário iria acarretar um alto custo final dos bens industrializados, fazendo com que a empresa não seja competitiva ou até mesmo impactando em sua permanência no mercado.

1.1 Indústria 4.0

A indústria 4.0, que pode ser considerada a quarta revolução industrial, representa um novo conceito dos sistemas produtivos empregando novas tecnologias para integrar máquina e humanos, compondo redes industriais frequentemente em localizações geográficas distintas. Para poder usar todo o potencial dessas novas tecnologias, é necessário um rompimento com os métodos tradicionais e, por consequência, novas técnicas de modelagem de sistemas devem ser consideradas. (KOLBERG e ZUHLKE, 2015).

1.2 Internet das coisas

Do inglês *Internet of Things* (IoT), a Internet das Coisas refere-se à integração de objetos físicos e virtuais em redes conectadas à Internet, permitindo que os objetos coletem, troquem e armazenem dados que serão processados e analisados, gerando informações e serviços em grande escala. (ALMEIDA, 2015). São muitas as possibilidades de objetos conectados: automóveis, *smartphones*, eletrodomésticos, artigos de vestuário, fechaduras, entre outros aparelhos.

O termo IoT foi mencionado pela primeira vez em 1999 por Kevin Ashton e era associada ao uso da tecnologia RFID (*Radio-Frequency IDentification*), porém, a definição se expandiu para outras aplicações, como na área da saúde, transporte, comunicação. (SUNDMAEKER et al., 2010).

1.3 Arduino

Arduino é uma plataforma de prototipagem eletrônica de código aberto, projetada com um microcontrolador com suporte de entrada e saída embutido. As placas Arduino são capazes de ler entradas, luz em um sensor, o aperto de um botão ou até mesmo uma mensagem do *Twitter* e transformar isto em uma saída, ativar um motor, acender um LED,

publicar algo *online*. Além disso, tem um baixo custo, algo em torno de US$50, pode ser programado em diversas plataformas, como *Windows*, *Macintosh OSX* e *Linux* e ainda tem milhares de bibliotecas disponíveis e de código aberto. (ARDUINO, 2019).

1.4 Modelo de código aberto

Generalizando, *software* de código aberto é um *software* que pode ser livremente acessado, usado, modificado e compartilhado por qualquer um. *Software* de código aberto é feito por muitas pessoas e distribuído sobre licenças de acordo com a definição de código aberto (OPENSOURCE, 2018).

Da mesma maneira, o *hardware* de código aberto refere-se às especificações de construção de um objeto físico que é licenciado de maneira que o referido objeto possa ser estudado, modificado, criado e distribuído por qualquer um (OPENSOURCE, 2019).

1.5 Objetivo

O intuito deste trabalho é o desenvolvimento de uma plataforma para extração e persistência de dados de produtividade em equipamentos industriais utilizando *hardware* de baixo custo a fim de disponibilizar essas informações em tempo real para *software* de controle de atividades.

2 DESENVOLVIMENTO

Esta parte do trabalho foi dividido em duas partes: *Hardware* e *Software*, respectivamente. Na parte de *hardware* será explicado qual microcontrolador foi utilizado e seus acessórios, já na parte de *software* é mostrado quais tecnologias foram utilizadas para o recebimento e persistência dos dados. Lembrando que será usado, quando possível, somente *Hardware* e *Software* no modelo *Open Source* e qualquer configuração e código fonte será disponibilizado integralmente e poderá ser editado, posteriormente, desde que mantenha o modelo.

2.1 *Hardware*

Neste trabalho foi utilizado um Arduino Uno mostrado na figura 1 com um *Shield Ethernet* modelo W5100 mostrado na figura 2 com o endereço IP 192.168.1.10, nessa mesma rede há um computador *Linux* que é o responsável por receber os dados de cada dispositivo.

Figura 1 - Arduino

Fonte: Arduino (2019)

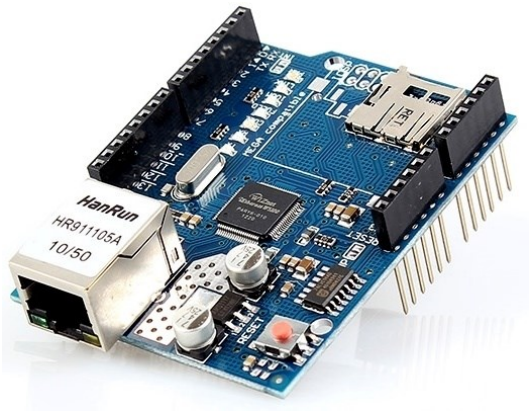
O código a seguir é um exemplo de como deve ficar o código que estará no Arduino, que será monitorado, lembrado que cada Arduino deve ter seu próprio endereço IP para que não haja conflitos na rede.

```
#include <SPI.h>
#include <Ethernet.h>

//Pino para o contador
const byte interruptPin = 2;

//Define o código do dispositivo
char deviceId[] = "S001";

//Contador de peças (volatile é para ser usado dentro da interrupção, se
não usar pode dar erro nas contagens)
volatile int contador = 0;

//MAC Address do shield
byte mac[] = {
  0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED
};
//Endereço IP do shield
IPAddress ip(192, 168, 1, 10);

//Define o EthernetServer e sua porta (80 é a porta padrão para http)
EthernetServer server(80);

void setup() {

  // Abre a porta serial
  Serial.begin(9600);
  while (!Serial) {
    ; // Aguarda a conexão da porta serial
  }
  Serial.println("Ethernet WebServer");

  //Inicializa a conexão ethernet com o ip e mac definidos na linha 15 e 18
  Ethernet.begin(mac, ip);

  // Verifica se o Ethernet Shield está conectado
  if (Ethernet.hardwareStatus() == EthernetNoHardware) {
    Serial.println("Ethernet shield não encontrado. Não é possível
continua");
    while (true) {
```

```
      delay(1); // Se o hardware nao existe não sai desse loop
    }
  }
  if (Ethernet.linkStatus() == LinkOFF) {
    Serial.println("O cabo Ethernet está desconectado");
  }

  // Inicializa o EthernetServer
  server.begin();

  //Mostra na Serial o ID do dispositivo e endereço IP
  Serial.print("O ID do Dispositivo é ");
  Serial.println(deviceId);

  Serial.print("O servidor está em ");
  Serial.println(Ethernet.localIP());

  //Define o comportamento do interruptPin
  pinMode(interruptPin, INPUT_PULLUP);
  //Quando o estado do interruptPin passar de HIGH para LOW, executar a
função incContador
  attachInterrupt(digitalPinToInterrupt(interruptPin), incContador,
FALLING);

}

void loop() {
  // Escuta por conexões de clientes
  EthernetClient client = server.available();
  if (client) {
    Serial.println("Novo cliente");
    // an http request ends with a blank line
    boolean currentLineIsBlank = true;
    while (client.connected()) {
      if (client.available()) {
        char c = client.read();
        Serial.write(c);
        // if you've gotten to the end of the line (received a newline
        // character) and the line is blank, the http request has ended,
        // so you can send a reply
        if (c == '\n' && currentLineIsBlank) {
          // envia um http response header com tipo json
          client.println("HTTP/1.1 200 OK");
          client.println("Content-Type: application/json;charset=utf-8");
          client.println("Connection: close");  // Fecha a conexão após
receber o response
          client.println();
          client.println("{");
          client.print("\"Equipamento\":\"");
          client.print(Ethernet.localIP());
          client.println("\",");
          client.print("\"Quantidade\":\"");
          client.print(contador);
          client.println("\",");
          client.println("\"Sensores\" : [");
          // Le os valores das 5 saidas analógicas
          for (int analogChannel = 0; analogChannel < 6; analogChannel++) {
            int sensorReading = analogRead(analogChannel);
            client.print("[\"analog");
            client.print(analogChannel);
            client.print("\",");
```

```
            client.print(sensorReading);
            if(analogChannel < 5)
              client.println("],");
            else
              client.println("]");
          }
          client.println("]");
          client.println("}");
          break;
        }
        if (c == '\n') {
          // you're starting a new line
          currentLineIsBlank = true;
        } else if (c != '\r') {
          // you've gotten a character on the current line
          currentLineIsBlank = false;
        }
      }
    }
    // espera para dar tempo de o browser receber os dados
    delay(1);
    // Encerra a conexão`
    client.stop();
    Serial.println("client disconnected");
  }
}

//Incrementa contador
void incContador() {
    contador++;
}
```

Figura 2 - Shield Ethernet



Fonte: Filipeflop (2019)

O código explica o que cada bloco faz, facilitando o entendimento, somente os trechos mais importantes serão elucidados. Foi conectado no pino 2 (Digital) um sensor indutivo, responsável pela contagem de peças do equipamento, toda vez que esse pino vai do valor *HIGH* para *LOW* uma interrupção é acionada executando a função incContador() que incrementa um à variável contador, no pino analógico 0 foi conectado um sensor de temperatura que monitora a temperatura de trabalho do equipamento, isso é útil para saber

se o equipamento está trabalhando numa temperatura de acordo com o estipulado pelo fabricante. No código é possível notar que o Arduino retorna as 6 portas analógicas, isso fica como exemplo de laço para varrer todas essas portas e retornar seus valores respectivos. Nas figuras 3 e 4 temos exemplos de dispositivos que podem ser acoplados nas portas analógicas e digitais do microcontrolador.

Figura 3 – Sensores indutivos

Fonte: Metaltex (2019)

Figura 4 – Sensor de Temperatura

Fonte: Melexis (2019)

2.2 *Software*

Para fazer a requisição e a persistência dos dados foi utilizado um computador com uma distribuição do *Linux* instalado como sistema operacional, a distribuição escolhida foi a Fedora que pode ser obtida em: https://getfedora.org/pt_BR/. A instalação do sistema é bem simples no próprio *site* do *download* tem uma boa explicação de como fazer, um ponto importante é configurar o endereço de rede para que o computador fique na mesma rede do Arduino, neste caso o endereço IP atribuído foi: 192.168.1.5.

Foi necessário instalar alguns pacotes no sistema operacional, o gerenciador de pacotes do Fedora é o DNF, para ter uma explicação do que este comando é capaz, abra o terminal e digite:

```
$ man dnf
```

O comando *man* faz referência a manual, ou seja, abre o manual do comando que deseja ser utilizado.

Para atualizar os pacotes instalados e ao mesmo tempo atualizar a lista de pacotes disponíveis para *download* é necessário executar o comando.

```
$ sudo dnf update
```

Os pacotes instalados para este projeto foram: Apache, PHP e MariaDB. A instalação desses pacotes são todas feitas através do comando dnf.

2.2.1 Instalação do Apache

O Apache é um *webserver* muito popular e usado com grande abrangência para sistemas *Linux*. O comando abaixo inicia a instalação do Apache Server.

```
$ sudo dnf install httpd
```

É necessário iniciar o serviço do apache e configurá-lo para iniciar junto ao sistema operacional com os seguintes comandos:

```
$ sudo systemctl enable httpd.service
$ sudo systemctl start httpd.service
```

2.2.2 Instalação do MariaDB

O MariaDB é um Sistema de Gerenciamento de Banco de Dados (SGBD), SGBDs são mecanismos de armazenamento e permitem a persistência e controle de dados. O SGBD usado neste artigo é do tipo relacional e tem esse nome, pois os dados das tabelas são relacionados por um campo em comum. Um SGBD relacional armazena dados em tabelas que são organizadas em colunas e linhas, cada linha contém os dados de uma única entidade e pode ser chamada de registro ou tupla, cada coluna, também chamada de atributo, possui um tipo de dado e por fim, uma chave, sendo o conjunto de um ou mais atributos que determinam a unicidade dos registros.

O MariaDB é uma alternativa do MySQL e é o banco de dados padrão do Fedora. Pode ser instalado diretamente do repositório padrão com o comando abaixo.

```
$ sudo dnf install mariadb-server
```

Da mesma maneira que o Apache, é necessário iniciar o serviço do MariaDB e fazê-lo iniciar junto ao sistema operacional.

```
$ sudo systemctl enable mariadb.service
$ sudo systemctl start mariadb.service
```

O MariaDB tem um passo extra para deixar o sistema mais seguro.

```
$ sudo mysql_secure_installation
```

O comando acima personaliza a instalação do MariaDB, uma dessas personalizações, por exemplo, é a mudança da senha de root que é vazia por padrão. Outras mudanças que também podem ser feitas com o comando são: remover acesso anônimo, desabilitar o acesso remoto do usuário root, remover os bancos de dados de teste e recarregar a tabela de permissões.

2.2.3 Instalação do PHP

PHP é uma das mais populares linguagens de programação. É usada mundialmente para desenvolvimento de *websites*, mas também pode ser muito utilizada na criação de *scripts* que é, também, o caso deste trabalho. O comando abaixo instala a última versão do PHP.

```
$ sudo dnf install php php-common
```

Alguns módulos extras podem ser instalados para facilitar algumas tarefas.

```
$ sudo dnf install php-mysqlnd php-xml php-json php-gd php-mbstring
```

No *script* “Server.php” que será mostrado em 2.2.6 é usado o módulo extra php-json para fazer desserialização de uma resposta com um arquivo json.

2.2.4 Regras de Firewall

Para permitir o acesso nos serviços HTTP e HTTPS é necessário fazer algumas modificações no *firewall* do sistema.

```
$ sudo firewall-cmd –permanent –add-service=http
$ sudo firewall-cmd –permanent –add-service=https
```

Após adicionar as regras acima é preciso recarregar o *firewall* para aplicar as modificações.

```
$ sudo systemctl reload firewalld
```

2.2.5 Instalação do PHPMyAdmin

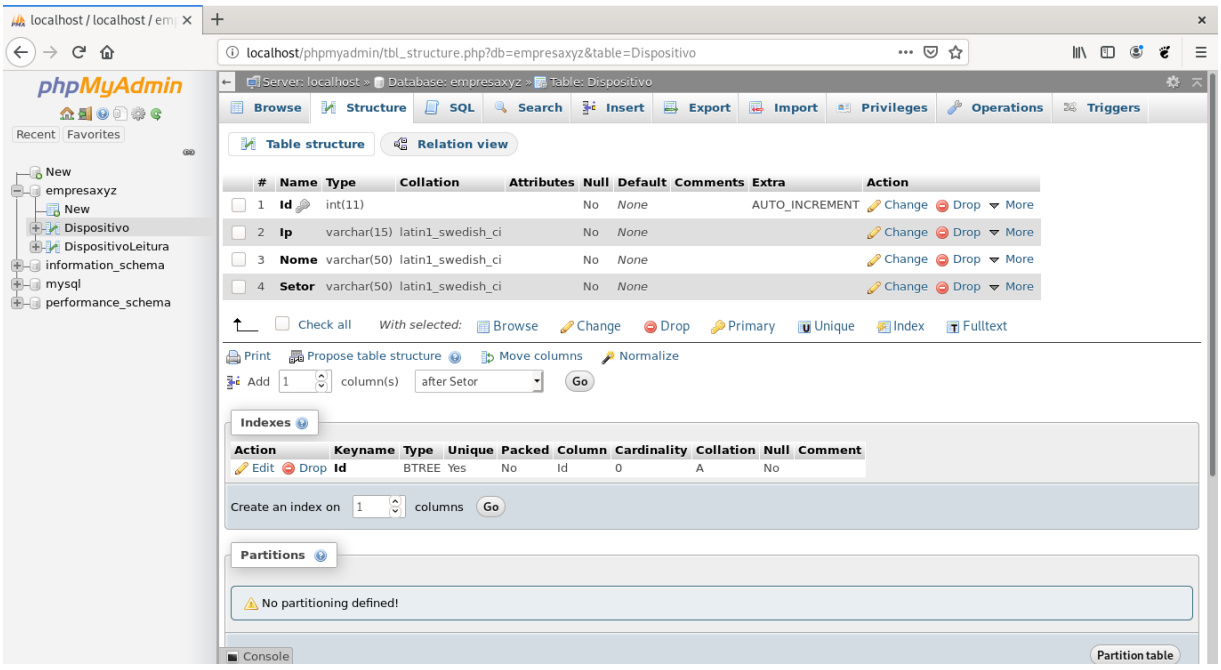
PHPMyAdmin é uma ferramenta de *software* escrita em PHP para gerenciar a instância do MariaDB/MySQL por uma interface WEB. Facilita bastante o trabalho para quem não tem facilidade com o terminal do *Linux* e comandos SQL. Para realizar a instalação do PHPMyAdmin é necessário executar o comando abaixo.

```
$ sudo dnf install phpMyAdmin
```

2.2.6 Configurações finais

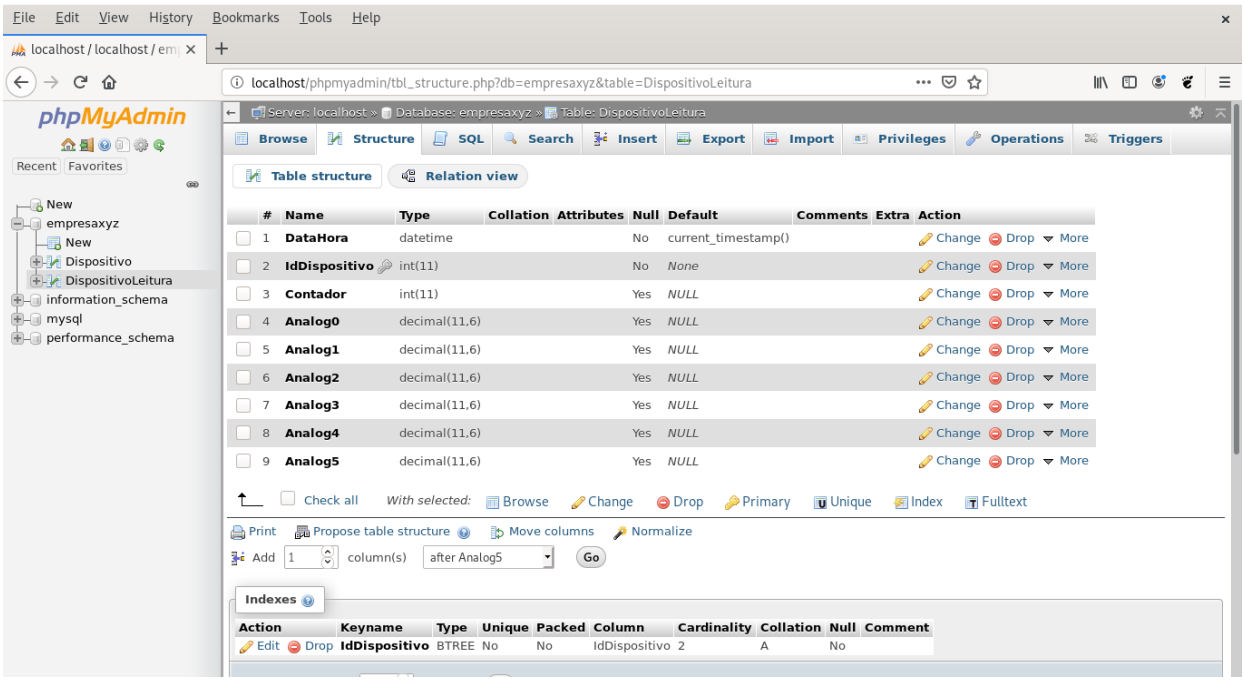
Foi criado um banco de dados com o nome “empresaxyz” com duas tabelas, Dispositivo e DispositivoLeitura, respectivamente. O conteúdo dessas duas tabelas é listado nas figuras 5 e 6.

Figura 5 - Tabela Dispositivo



Fonte: Elaborado pelo autor.

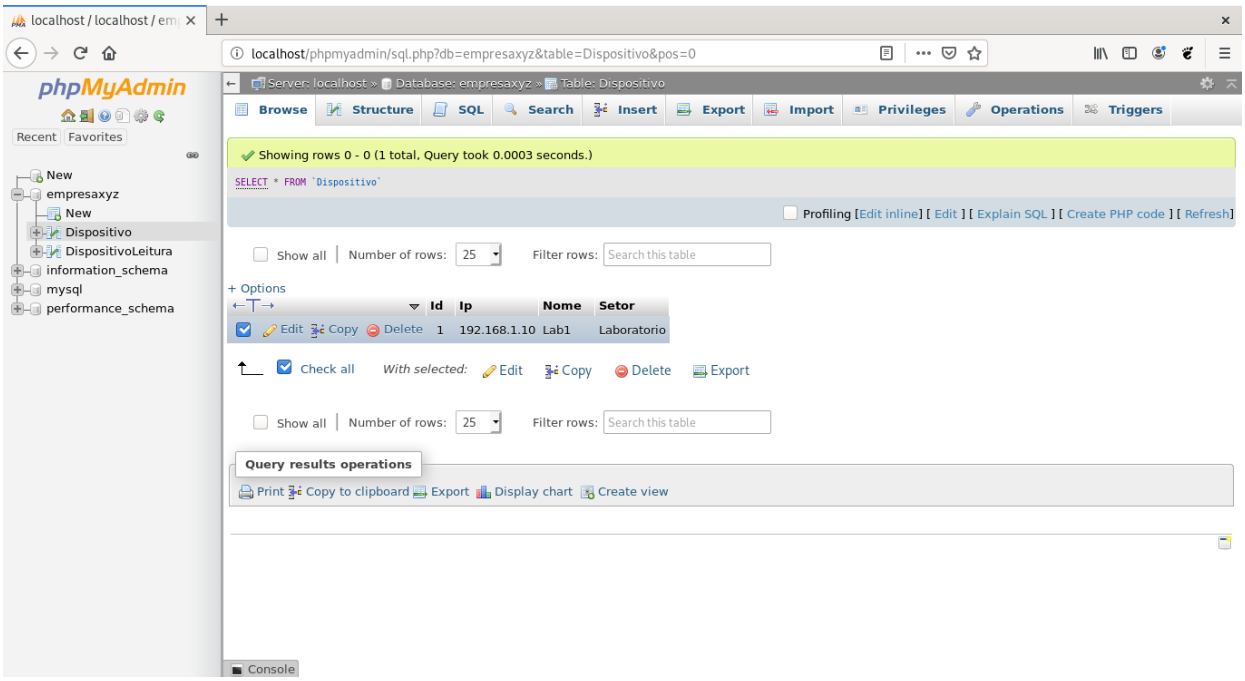
Figura 6 - Tabela DispositivoLeitura



Fonte: Elaborado pelo autor.

Para realizar as leituras é necessário ter um registro na tabela Dispositivo que remeta as configurações feitas no Arduino, o principal dado é o endereço IP. Esse registro é mostrado na figura 7.

Figura 7 - Registro na tabela Dispositivo



Fonte: Elaborado pelo autor.

Por fim, foi criado um *script* em PHP que é responsável pela leitura e persistência de dados. O nome do *script* é “Server.php” e ele está disponível integralmente no bloco abaixo.

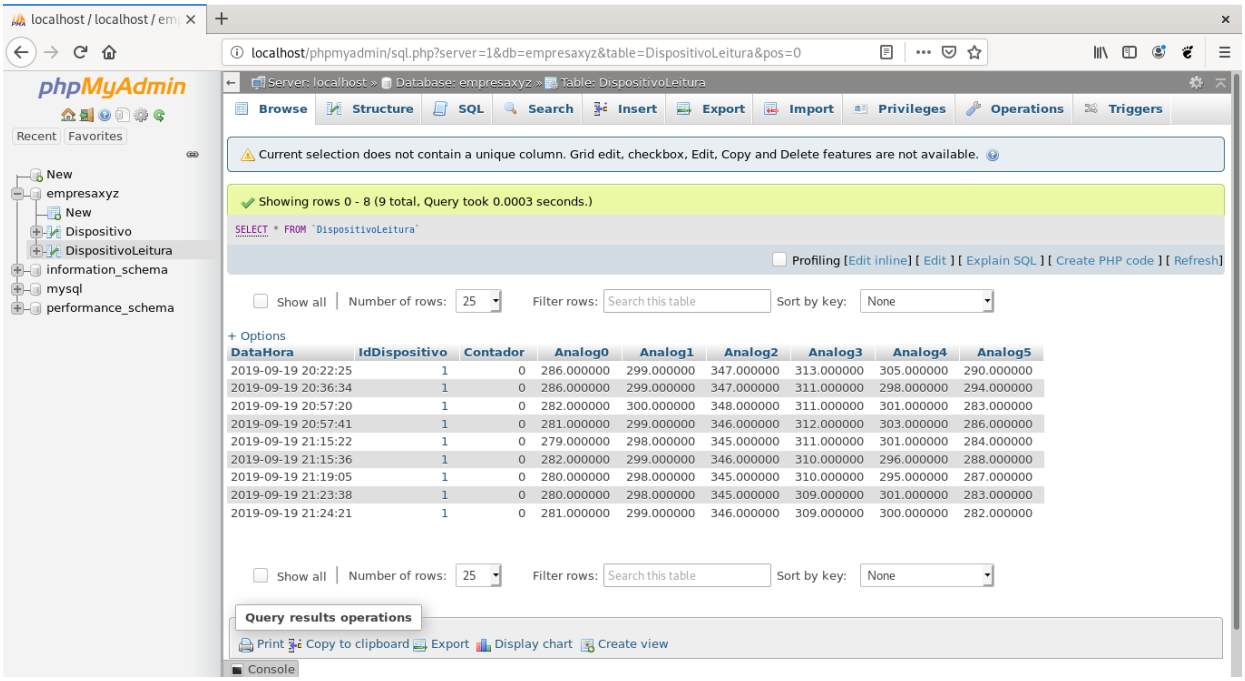
```
<?php

        $dispositivos = [];//new Dispositivo(1, "192.168.0.10",
"Dispositivo1", "1");
        $serverName = "localhost";
        $userName = "root";
        $password = "123456";
        $dbName = "empresaxyz";

        $conn = new mysqli($serverName, $userName, $password, $dbName);

        if($conn->connect_error) {
                die("Connection failed: " . $conn->connect_error);
        }

        $sql = "SELECT Id, Ip, Nome, Setor FROM Dispositivo";
        $result = $conn->query($sql);

        if($result->num_rows > 0) {
                while($row = $result->fetch_assoc()){
                        $dispositivos[] = new Dispositivo($row["Id"],
$row["Ip"], $row["Nome"], $row["Nome"]);
                }
        }
        else {
                echo "Nenhum dispositivo cadastrado", PHP_EOL;
        }
        $conn->close();

        foreach($dispositivos as $dispositivo){

                $url = "http://" . $dispositivo->getIp();
                $json = file_get_contents($url);

                if($json === false){
                        echo "Dispositivo não encontrado", PHP_EOL;
                }
                else{
                        $json_data = json_decode($json, true);
                        $leitura = new Leitura($dispositivo->getId(),
$json_data["Quantidade"],
$json_data["Sensores"][0][1],$json_data["Sensores"][1][1],$json_data["Senso
res"][2][1],$json_data["Sensores"][3][1],$json_data["Sensores"][4][1],$json
_data["Sensores"][5][1]);

                        $conn = new mysqli($serverName, $userName,
$password, $dbName);

                        if($conn->connect_error) {
                                die("Connection failed: " . $conn-
>connect_error);
                        }

                        $sqlInsert = "INSERT INTO DispositivoLeitura
(IdDispositivo, Contador, Analog0, Analog1, Analog2, Analog3, Analog4,
Analog5) Values (" . $leitura->getIdDispositivo() . "," . $leitura-
>getContador() . "," . $leitura->getAnalog0() . "," . $leitura-
```

```
>getAnalog1() . "," . $leitura->getAnalog2() . "," . $leitura->getAnalog3()
. "," . $leitura->getAnalog4() . "," . $leitura->getAnalog5() . ")";

                        if($conn->query($sqlInsert) === TRUE) {
                                echo "Leitura do dispositivo #" . $leitura-
>getIdDispositivo() . " no IP " . $dispositivo->getIp() . " adicionada com
sucesso", PHP_EOL;

                        }
                        else {
                                echo "Error: " . $sqlInsert, PHP_EOL,  $conn-
>error, PHP_EOL;
                        }
                        $conn->close();
                }
        }

        class Dispositivo {
                private $id;
                private $ip;
                private $nome;
                private $setor;

                function __construct($id, $ip, $nome, $setor) {
                        $this->id = $id;
                        $this->ip = $ip;
                        $this->nome = $nome;
                        $this->setor = $setor;
                }

                function getId() {
                        return $this->id;
                }

                function getIp() {
                        return $this->ip;
                }

                function getNome() {
                        return $this->idDispositivo;
                }

                function getSetor() {
                        return $this->idDispositivo;
                }
        }


        class Leitura {
                private $idDispositivo;
                private $contador;
                private $analog0;
                private $analog1;
                private $analog2;
                private $analog3;
                private $analog4;
                private $analog5;

                function __construct($idDispositivo, $contador, $analog0,
$analog1, $analog2, $analog3, $analog4, $analog5) {
                        $this->idDispositivo = $idDispositivo;
```

```
                        $this->contador = $contador;
                        $this->analog0 = $analog0;
                        $this->analog1 = $analog1;
                        $this->analog2 = $analog2;
                        $this->analog3 = $analog3;
                        $this->analog4 = $analog4;
                        $this->analog5 = $analog5;
                }

                function getIdDispositivo() {
                        return $this->idDispositivo;
                }

                function getContador() {
                        return $this->contador;
                }

                function getAnalog0() {
                        return $this->analog0;
                }

                function getAnalog1() {
                        return $this->analog1;
                }

                function getAnalog2() {
                        return $this->analog2;
                }

                function getAnalog3() {
                        return $this->analog3;
                }

                function getAnalog4() {
                        return $this->analog4;
                }

                function getAnalog5() {
                        return $this->analog5;
                }
        }
?>
```

Esse *script* basicamente faz a leitura de todos os registros na tabela Dispositivo e para cada registro dessa tabela faz as leituras dos dispositivos correspondentes pelo endereço IP, e para cada dispositivo faz a persistência das leituras na tabela DispositivoLeitura. Nesse trabalho está contemplado somente um dispositivo, mas, poderia ter facilmente um número maior. O resultado pode ser conferido na figura 8.

Figura 8 - Registros na tabela DispositivoLeitura



Fonte: Elaborado pelo autor.

Para fazer o *script* capturar dados continuamente, foi necessário incluir um cronjob, *job cronagram*, que funciona como um cronograma informando de quanto em quanto tempo a tarefa vai ser executada. Para criar o cronjob foi necessário executar os comandos abaixo.

```
$ sudo crontab -e
```

No editor de texto foi incluído as seguintes linhas:

```
* * * * * ( /usr/bin/php /CaminhoDoScript/Server.php)
* * * * * ( sleep 10 ; /usr/bin/php /CaminhoDoScript/Server.php)
* * * * * ( sleep 20 ; /usr/bin/php /CaminhoDoScript/Server.php)
* * * * * ( sleep 30 ; /usr/bin/php /CaminhoDoScript/Server.php)
* * * * * ( sleep 40 ; /usr/bin/php /CaminhoDoScript/Server.php)
* * * * * ( sleep 50 ; /usr/bin/php /CaminhoDoScript/Server.php)
```

Neste caso o *script* será executado a cada 10 segundos a partir do momento da criação do cronjob. Utilizando somente os agendamentos do cronjob é possível executar um comando no máximo a cada 60 segundos, por isto, foi feito o uso da função sleep que serve para aguardar uma certa quantidade de tempo para executar a tarefa. Este cronjob pode ser modificado para ser executado em qualquer momento e tempo desejado.

Com isso foi possível armazenar todas as informações do dispositivo com um intervalo de 10 segundos e gerar uma histórico que poderá ser útil em tomada de decisões futuras, planejamento e estratégias de produção.

3 CONCLUSÃO

Com a instalação deste dispositivo nos equipamentos dos setores operacionais, pode-se notar uma grande diferença na tomada de decisões do controle de produção, pois os responsáveis puderam ver com mais clareza quais equipamentos estavam ocupados e em quanto tempo ele estaria disponível para fabricar outro produto.

Também foi notável a diminuição de erros por interferência humana pois, todos os apontamentos de processos eram feitos de forma manual, com o novo dispositivo as informações de preparação de máquina, produção, parada para manutenção e conclusão de processos, puderam ser facilmente visualizados e armazenados para posterior consulta. Como trabalho futuro, pode-se fazer uma análise mais profunda destas informações para gerar estatísticas de produção e podendo até ajudar na análise de manutenções preditivas.

REFERÊNCIAS

ALMEIDA, Hyggo. Internet das Coisas: tudo conectado. **Computação Brasil**, v. 29, p. 7-8, 2015. Disponível em: https://www.sbc.org.br/images/flippingbook/computacaobrasil/computa_29_pdf/comp_brasil_2015_4.pdf. Acesso em: 05 jun. 2019.

ARDUINO. **What is arduino.** Disponível em: https://www.arduino.cc/en/Guide/Introduction. Acesso em: 05 jun. 2019.

FILIPEFLOP. **Ethernet Shield W5100 para Arduino**. Disponível em: https://www.filipeflop.com/produto/ethernet-shield-w5100-para-arduino. Acesso em: 05 jun. 2019.

GETFEDORA. **Fedora Linux**. Disponível em: https://getfedora.org/pt_BR/. Acesso em: 05 jun. 2019.

HERNANDES, Danniel de Souza. **Análise do impacto produtivo da implantação de um sistema de coleta de dados em tempo real integrado com ERP**. 75f. Dissertação - (Mestrado em Engenharia de Produção) - Universidade Nove de Julho, São Paulo, 2011. Disponível em: https://bibliotecatede.uninove.br/bitstream/tede/174/1/B_Daniel%20de%20Souza%20Hernandes.pdf. Acesso em: 05 jun. 2019.

KOLBERG, Dennis; ZÜHLKE, Detlef. Lean automation enabled by industry 4.0 technologies. **IFAC-PapersOnLine**, v. 48, n. 3, p. 1870-1875, 2015. Disponível em: https://www.sciencedirect.com/science/article/pii/S2405896315005984?via%3Dihub. Acesso em: 10 out. 2019.

MELEXIS. **Digital plug & play infrared thermometer in a TO-can**. Disponível em: https://www.melexis.com/en/product/MLX90614/Digital-Plug-Play-Infrared-Thermometer-TO-Can. Acesso em: Acesso em: 05 jun. 2019

METALTEX. **Sensores indutivos**. Disponível em: https://www.metaltex.com.br/produtos/automacao/sensores-indutivos/i-sensor-indutivo-cilindrico. Acesso em: 05/06/2019.

OPENSOURCE. **Basics of open source**. 2018. Disponível em: https://opensource.org/faq#osd. Acesso em: Acesso em: 05 jun. 2019

OPENSOURCE. **What is open hardware**. 2019. Disponível em: https://opensource.com/resources/what-open-hardware. Acesso em: 05 jun. 2019

SCHWAB, Klaus. **The fourth industrial revolution**. Geneva: World Economic Forum, 2016. 172 p. Disponível em: https://luminariaz.files.wordpress.com/2017/11/the-fourth-industrial-revolution-2016-21.pdf. Acesso em: 12 nov. 2019.

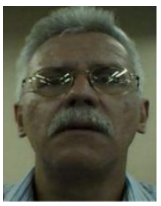
SUNDMAEKER, Harald et al. Vision and challenges for realising the Internet of Things. **Cluster of European Research Projects on the Internet of Things, European Commision**, v. 3, n. 3, p. 12, 2010. Disponível em: http://www.internet-of-things-research.eu/pdf/IoT_Clusterbook_March_2010.pdf. Acesso em: 12 nov. 2019.

SOBRE OS AUTORES

Wilson de Oliveira Andrade

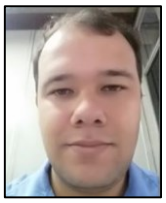
Possui graduação em Análise e Desenvolvimento de Sistemas pela Faculdade FATEC Zona Sul (2015), cursando atualmente a Pós-Graduação em Automação Industrial pela Faculdade SENAI de Tecnologia Mecatrônica (2019). Tem ampla experiência em redes de computadores e desenvolvimento de sistemas. É desenvolvedor na empresa Intemobile do Brasil.

Paulo Sebastião Ladivez



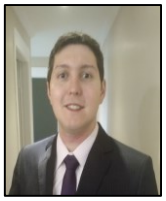
Possui graduação em Engenharia Elétrica pela Universidade Mogi das Cruzes (1984), com especialização em Tecnologias e Sistemas de Informação pela Universidade Federal do ABC (2013). Atualmente é professor da Faculdade SENAI de Tecnologia Mecatrônica, lecionando as disciplinas Projetos, Microcontroladores, Linguagem de Programação no Curso de Tecnologia em Mecatrônica Industrial e na Pós-Graduação em Automação Industrial. Tem experiência na área de Engenharia Eletrônica, com ênfase em Automação Industrial e Mecatrônica, atuando principalmente nos seguintes temas: Mecatrônica, Manufatura Digital, Redes Industriais, Automação Industrial, Microcontroladores e Controle.

JOSÉ ROBERTO DOS SANTOS



Atualmente ministra aulas na pós-graduação de Indústria 4.0 e na graduação em Tecnologia em Mecatrônica na Faculdade SENAI de Tecnologia Mecatrônica, que fica no SENAI Armando de Arruda Pereira. Assessora também o Instituto SENAI de Tecnologia Metalmecânica em projetos industriais com foco na Indústria 4.0. Durante 9 anos ministrou aulas pelo SENAI-SP, nos cursos de técnico em eletroeletrônica, cursos de aprendizagem industrial eletricista de manutenção e mecânico de usinagem, além de Formação Inicial e Continuada (FIC) com cursos voltados a área de redes de computadores e programação, possui treinamento de Linux, cisco e Microsoft. Possui Pós-graduação na área de segurança da informação pela Uninove (2016), graduação em tecnologia da informação e bacharel em sistema da informação (2009), além de superior em Automação industrial. Tem experiencia na área de Segurança da informação, administração de ambientes de redes Windows e Linux, automação indústria.

THIAGO TADEU AMICI



Atualmente ministra aulas na pós-graduação de Indústria 4.0 e na graduação em Tecnologia em Mecatrônica na Faculdade SENAI de Tecnologia Mecatrônica, que fica no SENAI Armando de Arruda Pereira. Assessora também o Instituto SENAI de Tecnologia Metalmecânica em projetos industriais com foco na Indústria 4.0. Durante 7 anos ministrou aulas pelo SENAI-SP, nos cursos de técnicos de Mecatrônica, Automação Industrial, Eletrônica e Eletroeletrônica, além de Formação Inicial e Continuada (FIC) com cursos voltados ao CLP da Siemens. Possui mestrado em Automação e Controle e Processos pelo Instituto Federal de Ciências e Tecnologia de SP (IFSP - 2018), graduação em Engenharia Elétrica pela Faculdade de Engenharia São Paulo (2012), graduação em Tecnologia em Automação Industrial pelo IFSP (2009) e ensino profissionalizante em Eletrônica pela Instituição Liceu de Artes e Ofícios de São Paulo (2002). Tem experiência na área de Engenharia Elétrica, Automação Industrial, Mecatrônica, Robótica e Indústria 4.0. Experiência internacional na aprovação de linha de produção (Cavemil) em Milão na Itália e sua instalação no Brasil. Participou do desenvolvimento do projeto, programação, montagem e apresentação da Linha de Manufatura Avançada Industrial 4.0 realizada em parceria entre o SENAI-SP e a ABIMAQ, que foi exposta na FEIMEC 2018 e da linha de Confecção 4.0, em parceria entre o SENAI-SP e a ABIT.