UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

MARCELO GABRIEL DOS SANTOS QUEIROZ VITORINO

SISTEMA DE HARDWARE E SOFTWARE PARA MONITORAMENTO ESTRUTURAL DE EDIFICAÇÕES

CAMPINA GRANDE - PB

2021

MARCELO GABRIEL DOS SANTOS QUEIROZ VITORINO

SISTEMA DE HARDWARE E SOFTWARE PARA MONITORAMENTO ESTRUTURAL DE EDIFICAÇÕES

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientadora: Joseana Macêdo Fechine Régis de Araújo.

CAMPINA GRANDE - PB

2021

V845s Vitorino, Marcelo Gabriel dos Santos Queiroz.
Sistema de hardware e software para monitoramento estrutural de edificações. / Marcelo Gabriel dos Santos Queiroz Vitorino. – 2021.

10 f.

Orientadora: Profa. Dra. Joseana Macêdo Fechine Régis de Araújo.
Trabalho de Conclusão de Curso - Artigo (Curso de Bacharelado em Ciência da Computação) - Universidade Federal de Campina Grande; Centro de Engenharia Elétrica e Informática.

1. Monitoramento estrutural de edificações. 2. Structural Health Monitoring. 3. Acelerômetro. 4. Edificações. 5. Placa NodeMCU. 6. Dashboard de monitoramento. 7. Transmissão wireless de acelerações. 8. Internet das coisas. 9. Sensor MPU6050. 10. Microcontrolador. 11. Arduino. I. Araújo, Joseana Macêdo Fechine Régis de. II. Título.

CDU:004.8(045)

Elaboração da Ficha Catalográfica:

Johnny Rodrigues Barbosa
Bibliotecário-Documentalista
CRB-15/626

MARCELO GABRIEL DOS SANTOS QUEIROZ VITORINO

SISTEMA DE HARDWARE E SOFTWARE PARA MONITORAMENTO ESTRUTURAL DE EDIFICAÇÕES

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA:

Professora Dra. Joseana Macêdo Fechine Régis de Araújo.

Orientadora – UASC/CEEI/UFCG

Professor Dr. Kyller Costa Gorgônio.

Examinador – UASC/CEEI/UFCG

Professor Tiago Lima Massoni

Professor da Disciplina TCC – UASC/CEEI/UFCG

Trabalho aprovado em: 25 de Maio de 2021.

CAMPINA GRANDE - PB

RESUMO (ABSTRACT)

The structural monitoring (Structural Health Monitoring - SHM) is done with the objective of maintaining the safety reliability of the construction during its use, and to identify in advance possible problems that damage the structures. In general, a conventional dynamic SHM is composed of accelerometers, signal acquisition systems and visualization software. However, these solutions have a high cost, especially with regard to the hardware that makes up a SHM, which makes it difficult to popularize this type of monitoring. In this work, a hardware-software solution for structural monitoring was developed, whose acceleration data is collected from an accelerometer, processed on a NodeMCU card, transmitted wirelessly, to a server that displays the monitoring data on a Dashboard. The solution was evaluated based on the proper functioning of the acquisition, processing, transmission and exhibition modules. The system was introduced to Civil Engineering professors at the Federal University of Campina Grande, who reported the high potential of the solution for use in projects with real buildings.

Sistema de Hardware e Software para Monitoramento Estrutural de Edificações

Marcelo G. dos S. Q Vitorino
Graduando em Ciência da Computação Universidade Federal de Campina Grande
Campina Grande, Paraíba, Brasil
marcelo.vitorino@ccc.ufcg.edu.br

Joseana M. F. R. de Araújo
Unidade Acadêmica de Sistemas e Computação
Universidade Federal de Campina Grande
Campina Grande, Paraíba, Brasil
joseana@computacao.ufcg.edu.br

Marília M. C. de Araújo
Unidade Acadêmica de Engenharia Civil
Universidade Federal de Campina Grande
Campina Grande, Paraíba, Brasil
mariliamarcy@gmail.com

RESUMO

O monitoramento estrutural (*Structural Health Monitoring* - SHM) é feito com o objetivo de manter a confiabilidade de segurança da construção durante sua utilização, e identificar antecipadamente possíveis problemas que danifiquem as estruturas. De maneira geral, um SHM dinâmico convencional é composto por acelerômetros, sistemas de aquisição de sinais e softwares de visualização. Entretanto, essas soluções possuem um alto custo, sobretudo no que diz respeito ao hardware que compõe um SHM, o que dificulta a popularização desse tipo de monitoramento. Neste trabalho, foi desenvolvida uma solução hardware-software para monitoramento estrutural, cujos dados de aceleração são coletados a partir de um acelerômetro, processados em uma placa NodeMCU, transmitidos via *wireless*, para um servidor que exibe em um *Dashboard* os dados do monitoramento. A solução foi avaliada a partir do funcionamento adequado dos módulos de aquisição, processamento, transmissão e exibição. O sistema foi apresentado a professores de Engenharia Civil da Universidade Federal de Campina Grande, os quais relataram o alto potencial da solução para utilização em projetos com edificações reais.

Palavras-chave

Monitoramento Estrutural, SHM, Edificações, Acelerômetro, NodeMCU, Dashboard.

1. INTRODUÇÃO

O processo construtivo de uma edificação é constituído de diversas etapas, desde a análise do solo até os projetos hidrossanitários, elétricos, estruturais e arquitetônicos. No entanto, para a execução dos projetos de uma edificação, é necessário o conhecimento prévio da disposição dos elementos estruturais e um pré-dimensionamento, para posterior análise e dimensionamento das vigas, pilares, lajes e fundações. Cada projeto isoladamente deve estar em concordância com o projeto estrutural, afinal, esse será o responsável pelo suporte de todo o peso da edificação.

Após a conclusão da obra, devido à exposição natural e a sua utilização, torna-se necessário o monitoramento da saúde estrutural a fim de manter a confiabilidade de segurança da construção durante sua utilização, bem como evitar danos que evoluam para desabamentos, afundamentos de piso, trincas, dentre outros problemas.

De maneira geral, o monitoramento estrutural (SHM) pode ser feito a partir de diversas abordagens, desde software e hardware ou até mesmo pelo conjunto dos dois componentes, conforme afirma Araújo et al. (2017). No entanto, em se tratando de hardware (acelerômetros convencionais e sistemas de aquisição de sinais), esses podem ser equipamentos que necessitam de alto investimento, resultando em limitações para a implantação de um SHM.

Nesse cenário, um Arduino pode surgir como alternativa eficiente e de baixo custo para esse fim. Trata-se de um hardware de código aberto, criado em 2005, que podem ser conectados a sensores (de luz, temperatura, umidade, acelerações etc), realizar processamento, se conectar a outros dispositivos e exibir saídas a partir de diversos componentes (led, alto-falante, display e inúmeras outras possibilidades de interação com o ambiente) (SOUZA et al., 2017). Neste sentido, a comunidade científica vem adotando o Arduino, uma vez que, além de apresentar confiabilidade, estudos estimam que esses tipos de dispositivos podem representar reduções de 4 a 6 vezes nos custos em comparação com o uso de equipamentos convencionais (GALDINO e CURY, 2016).

Apesar da existência de estudos que mostram a utilização do Arduino para leitura de acelerações (ABDULLAHI et al., 2019); e de sistemas automatizados para aquisição de dados (PALMA E SILVA et al., 2019), é importante, para o monitoramento, a utilização de uma aplicação web, em conjunto com a coleta, processamento, transmissão e visualização das acelerações, que possa ser acessada a partir de qualquer localização.

Neste contexto, a partir de um projeto fruto de uma parceria entre UFCG, UFPB e UnB, que objetiva o desenvolvimento de um SHM por meio de vários Arduíno, este trabalho foi centrado no desenvolvimento do mecanismo de transmissão dos dados, através de um dispositivo *wireless* conectado ao Arduino, via placa NodeMCU. As acelerações, definidas em variáveis x, y e z, são armazenadas na nuvem e é disponibilizado ao usuário um *Dashboard* com gráficos que exibem os valores das acelerações ao longo de um período de tempo e que permitem o monitoramento da saúde estrutural da obra.

A solução busca impactar em companhias públicas ou privadas de manutenção e monitoramento de estruturas e prover uma solução de baixo custo que facilite o controle das edificações em tempo real.

2. OBJETIVOS

A seguir, a descrição dos objetivos geral e específicos do trabalho desenvolvido.

2.1 Objetivo Geral

Este trabalho teve como objetivo desenvolver a transmissão *wireless* de acelerações, a partir de uma placa NodeMCU, com o armazenamento na nuvem, para apresentação dos registros estruturais em um *Dashboard* de monitoramento.

2.2 Objetivos específicos

- Criação de um servidor web para armazenamento de registros das acelerações da estrutura x,y,z na nuvem;
- Estabelecimento da comunicação entre a placa Node MCU e o servidor web a partir de rede *wireless*;
- Criação de um *Dashboard* para apresentação das acelerações.

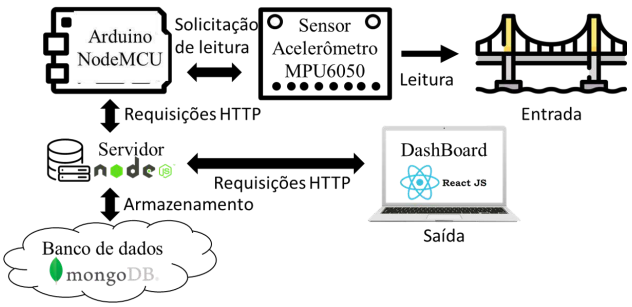
3. DESCRIÇÃO DA SOLUÇÃO

Nesta seção, serão descritos os componentes utilizados, suas respectivas funções e conexão desses para o desenvolvimento da solução.

3.1 Visão Geral

O sistema desenvolvido representa uma aplicação de internet das coisas (*Internet of things* - IoT), que consiste em uma rede de objetos incorporados a sensores, software e outras tecnologias com o objetivo de conectar e trocar dados com outros dispositivos e sistemas pela internet [8]. Na Figura 1, é apresentado o diagrama do sistema de monitoramento estrutural.

Figura 1: Visão geral do sistema de monitoramento estrutural.



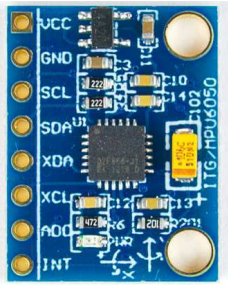
Fonte: autoria própria.

No sistema, um acelerômetro realiza a leitura dos dados de vibração, o Arduino se conecta a uma rede *wireless* e, transforma os dados para o formato JSON, realiza a conexão com um servidor web e realiza o envio dos dados. O servidor é responsável por se conectar com o banco de dados na nuvem e salvar os registros recebidos do Arduino. A interface gráfica desenvolvida, usada para representar o *dashboard* do sistema, exibe os dados registrados para o usuário final.

3.1.1 Sensor MPU6050

Para realizar as leituras de acelerações, foi utilizado o acelerômetro do tipo MPU6050 (Figura 2), que permite a utilização do protocolo I2C (mestre/escravo) e se conecta facilmente ao Arduino NodeMCU.

Figura 2: Acelerômetro e Giroscópio 3 Eixos Mpu6050.



Fonte: Extraído de [6].

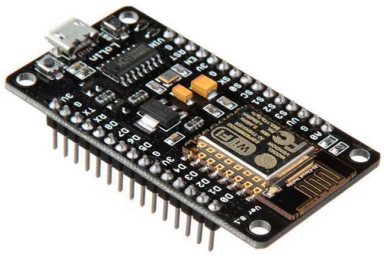
A seguir, as principais características do módulo.

- Chip: MPU-6050;
- Tensão de Operação: 3-5V;
- Conversor AD de 16 bits;
- Comunicação: Protocolo padrão I2C;
- Faixa do Giroscópio: ±250, 500, 1000, 2000°/s;
- Faixa do Acelerômetro: ±2, ±4, ±8, ±16g; e
- Dimensões: 2 x 1,6 x 0,1mm.

3.1.2 Microcontrolador

Para realizar a conexão com o acelerômetro, a partir da comunicação I2C, bem como utilizar a conexão com a internet e enviar os dados coletados em tempo real para o servidor web, foi utilizado o ESP32 WiFi / Bluetooth ESP-WROOM-32 na placa NodeMCU (Figura 3). A ESP32 permite criar variadas aplicações para projetos de IoT, acesso remoto, web servers, dataloggers, entre outros.

Figura 3: ESP32 WiFi / Bluetooth ESP-WROOM-32 na placa NodeMCU.



Fonte: Extraído de [7].

A seguir, as principais características do módulo.

- Processador: Xtensa 32-Bit LX6 Dual Core;
- Clock: 80 a 240 MHz (Ajustável);
- Memória ROM: 448 KB;

- Memória SRAM: 520 Kb;
- Memória Flash Externa: 4 Mb;
- Tensão de Alimentação: 2,7 a 3,6 VDC;
- Tensão de nível lógico: 3,3 VDC;
- Corrente de consumo: 80 mA (típica);
- Corrente de consumo: 500 mA (máxima);
- Interfaces: Cartão SD, UART(3 canais), SPI (3 canais), SDIO, I2C (2 canais), I2S (2 canais), IR, PWM LED (2 canais) e PWM motor (3 canais);
- Tipos GPIO: Digital IO (36), ADC 12-Bits (16 canais), DAC 8-Bits (2 canais), Sensor Capacitivo (10 canais); LNA pré-amplificador.
- WiFi 802.11 b/g/n: 2,4 a 2,5 GHz;
- Segurança WiFi: WPA/WPA2/ WPA2-Enterprise/ WPS;
- Criptografia WiFi: AES / RSA / ECC / SHA;
- Bluetooth 4,2 BR / EDR e BLE (Bluetooth Low Energy);
- RTC Integrado de 8 Kb (Slown / Fast);
- Sensor integrado: Temperatura e Hall;
- Temperatura de trabalho: -40° a +85° C;
- Compatível com a IDE do Arduino;
- Dimensões: 25,5 x 18,0 x 3,1 mm;
- Datasheet ESP-WROOM-32; e
- Datasheet do chip: ESP32-D0WDQ6.

3.2 Ambiente de Desenvolvimento

A plataforma utilizada para o envio dos dados do Arduino para o servidor web foi o ambiente integrado de desenvolvimento (IDE) Arduino 1.8.12 (64 bits), na versão de instalação para sistema operacional Linux. O código foi implementado utilizando linguagens de programação C e C++.

Para o desenvolvimento do servidor web e do *dashboard* de apresentação dos dados, foi utilizada a IDE VSCode, um editor de código-fonte desenvolvido pela Microsoft que permitiu o desenvolvimento da aplicação web para conexão com o banco de dados e visualização das informações coletadas no Arduino. A implementação foi feita utilizando a linguagem de programação JavaScript, a partir dos *frameworks* backend NodeJS, frontend ReactJS e o software de banco de dados MongoDB, que permitiu o armazenamento de dados na nuvem a partir da plataforma MongoDB Atlas.

4. RESULTADOS

Nesta seção, serão apresentados os resultados obtidos após a implementação dos módulos de software e hardware.

4.1 Processo de Desenvolvimento

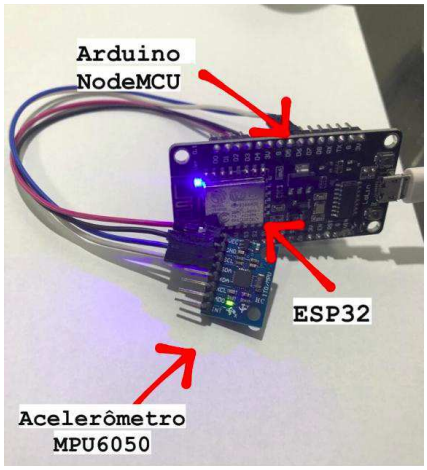
O sistema foi desenvolvido conforme etapas a seguir.

1) **Conexão Arduino NodeMCU e servidor NodeJS.** Nesta etapa, foi utilizada a biblioteca ArduinoJSON, responsável por armazenar os dados coletados no formato JSON. O servidor implementado usa esse formato para reconhecer os dados enviados via comunicação HTTP, utilizando o método POST.
2) **Conexão do servidor NodeJS e o banco de dados MongoDB.** Foi criado um servidor com a tecnologia NodeJS, que segue o estilo de código da API orientada a eventos para aplicativos de rede, promovido pelo NodeMCU. Os registros são armazenados na nuvem, com espaço de 512 MB de Disco, gratuito, oferecido pelo MongoDB Atlas.
3) **Conexão do *dashboard* ReactJS e do servidor NodeJS.** Utilizando também requisições HTTP, foi desenvolvido um *dashboard* escrito com o *framework* ReactJs contendo um gráfico que apresenta as acelerações armazenadas no servidor.

4.2 Conexão dos Componentes

A conexão entre o Arduino NodeMCU e o acelerômetro está apresentada na Figura 4. Os leds acesos significam que os dispositivos estão conectados com sucesso e recebendo uma corrente elétrica recomendada. O led na ESP32 pisca a cada leitura de aceleração e envia esse dado para o servidor web.

Figura 4: Conexão do Arduino com o acelerômetro.



Fonte: autoria própria.

4.3 Codificação da Solução

A seguir, a descrição da codificação, que contempla aquisição, processamento, armazenamento e exibição da informação.

- Subsistema de aquisição e processamento de dados. Consiste no desenvolvimento dos passos 1) e 2) descritos na Seção 4.1. O código apresentado na Figura 5 implementa a conexão da placa com o servidor web e o envio dos dados por meio de requisições HTTP. Nesse subsistema, as seguintes variáveis são armazenadas na nuvem: acelerações x,y,z, Localização e Temperatura.
- Subsistema de apresentação. Esse sistema foi implementado com um conjunto de classes estruturadas a partir do *framework* web React JavaScript. O código também realiza requisições HTTP para recuperar os dados armazenados na nuvem e apresentá-los na Dashboard (Figura 6).

Figura 5: Código da conexão do Arduino com o servidor web.

```
#include <ESP8266WiFi.h>
#include <Wire.h>
#include <ArduinoJson.h>

const int MPU_ADDR =      0x68;
const int WHO_AM_I =      0x75;
const int PWR_MGMT_1 =    0x6B;
const int GYRO_CONFIG =   0x1B;
const int ACCEL_CONFIG =  0x1C;
const int ACCEL_XOUT =    0x3B;

const int sda_pin = D5;
const int scl_pin = D6;

int16_t AcX, AcY, AcZ, Tmp, GyX, GyY, GyZ;

const char* SSID = "example";
const char* PASSWORD = "example";

const char* exampleHost = "example.com";

const char* IpApiHost = "ip-api.com";

WiFiClient client;

StaticJsonBuffer<300> jsonBuffer;
JsonObject& object = jsonBuffer.createObject();
JsonObject& data = object.createNestedObject("data");

JsonObject& accel = data.createNestedObject("accel");
JsonObject& temp = data.createNestedObject("temp");
JsonObject& gyro = data.createNestedObject("gyro");

JsonObject& accelX = accel.createNestedObject("accelX");
JsonObject& accelY = accel.createNestedObject("accelY");
JsonObject& accelZ = accel.createNestedObject("accelZ");

JsonObject& gyroX = gyro.createNestedObject("gyroX");
JsonObject& gyroY = gyro.createNestedObject("gyroY");
JsonObject& gyroZ = gyro.createNestedObject("gyroZ");

void iniciaMestreEscravo()
{
  Wire.begin(sda_pin, scl_pin);
}

void writeRegMPU(int reg, int val)
{
  Wire.beginTransmission(MPU_ADDR);
  Wire.write(reg);
  Wire.write(val);
  Wire.endTransmission(true);
}

uint8_t readRegMPU(uint8_t reg)
{
  uint8_t data;
  Wire.beginTransmission(MPU_ADDR);
  Wire.write(reg);
  Wire.endTransmission(false);
  Wire.requestFrom(MPU_ADDR, 1);
  data = Wire.read();
  return data;
}


void findMPU(int mpu_addr)
{
  Wire.beginTransmission(MPU_ADDR);
  int data = Wire.endTransmission(true);
}

void confereMPU(int mpu_addr)
{
  findMPU(MPU_ADDR);

  int data = readRegMPU(WHO_AM_I);

  if(data == 104)
  {
    data = readRegMPU(PWR_MGMT_1);
  }
}

void carregaAcelerometro()
{
  setSleepOff();
  setGyroScale();
  setAccelScale();
}

void setSleepOff()
{
  writeRegMPU(PWR_MGMT_1, 0);
}


void setGyroScale()
{
  writeRegMPU(GYRO_CONFIG, 0);
}

void setAccelScale()
{
  writeRegMPU(ACCEL_CONFIG, 0);
}

void leDadosAcelerometro()
{
  Wire.beginTransmission(MPU_ADDR);
  Wire.write(ACCEL_XOUT);
  Wire.endTransmission(false);
  Wire.requestFrom(MPU_ADDR, 14);

  AcX = Wire.read() << 8;
  AcX |= Wire.read();
  AcY = Wire.read() << 8;
  AcY |= Wire.read();
  AcZ = Wire.read() << 8;
  AcZ |= Wire.read();

  Tmp = Wire.read() << 8;
  Tmp |= Wire.read();

  GyX = Wire.read() << 8;
  GyX |= Wire.read();
  GyY = Wire.read() << 8;
  GyY |= Wire.read();
  GyZ = Wire.read() << 8;
  GyZ |= Wire.read();

  Serial.print("AcX = "); Serial.print(AcX);
  Serial.print(" | AcY = "); Serial.print(AcY);
  Serial.print(" | AcZ = "); Serial.print(AcZ);
  Serial.print(" | Tmp = "); Serial.print(Tmp/340.00+36.53)
  Serial.print(" | GyX = "); Serial.print(GyX);
  Serial.print(" | GyY = "); Serial.print(GyY);
  Serial.print(" | GyZ = "); Serial.println(GyZ);

  delay(50);
}

void reconnectaWiFi()
{
  if(WiFi.status() == WL_CONNECTED)
    return;

  WiFi.begin(SSID, PASSWORD);
}

void conectaWiFi()
{
  reconnectaWiFi();
}

void transformaEmJson()
{
  object["nodeID"] = "mcu1";

  accel["accelX"] = AcX;
  accel["accelY"] = AcY;
  accel["accelZ"] = AcZ;

  data["temp"] = Tmp/340.00+36.53;

  gyro["gyroX"] = GyX;
  gyro["gyroY"] = GyY;
  gyro["gyroZ"] = GyZ;
}

void fazPOST()
{
  if(client.connect(rpiHost, 80))
  {
    client.println("POST /addAcceleration HTTP/1.1");
    client.println("Host: example:80");
    client.println("Content-Type: application/json");
    client.print("Content-Length: ");
    client.println(object.measureLength());
    client.println();
    object.printTo(client);
  }
}

void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
  Serial.begin(115200);
  conectaWiFi();
  iniciaMestreEscravo();
  carregaAcelerometro();
  confereMPU(MPU_ADDR);
}

void loop() {
  leDadosAcelerometro();
  transformaEmJson();
  fazPOST();
}
```

Fonte: autoria própria

4.4 *Dashboard* de Monitoramento

Na Figura 6, é apresentada uma tela do *dashboard* com o gráfico de acelerações. O eixo x apresenta cada leitura realizada pelo Arduino e o eixo y apresenta o quantitativo das acelerações.

Figura 6: *Dashboard* de monitoramento.

Fonte: autoria própria.

5. CONSIDERAÇÕES FINAIS

Para o presente projeto, foram desenvolvidos os módulos para leitura das medidas de aceleração, leitura de geolocalização, envio de dados do Arduino para o servidor web, com armazenamento na nuvem e exibição, por meio de gráfico, das medidas de aceleração atualizadas. O envio dos dados do Arduino para o servidor se dá por meio de requisições HTTP, no formato JSON, utilizando uma rede *wireless*.

Ao realizar a apresentação da solução para engenheiros civis da área de estruturas, o *feedback* foi positivo, sobretudo no que se refere à tecnologia de envio dos dados em tempo real para um servidor na nuvem e na apresentação dos dados em uma *dashboard*.

5.1 Limitações

O trabalho atendeu ao objetivo proposto e o funcionamento do sistema pôde ser utilizado por profissionais de engenharia civil. Entretanto, para testes em construções civis, de acordo com o tamanho da estrutura, o sistema deve ser acoplado com mais de uma placa de arduino para realização de leituras e fazer o envio dos dados de forma simultânea para o servidor web.

5.2 Trabalhos Futuros

As adequações sugeridas pelos usuários de teste (engenheiros) serão implementadas posteriormente. As sugestões foram apresentadas com intuito de adequar o sistema para uso em edificações reais, a saber: Conexão entre várias placas de arduino para coleta e envio de dados provenientes da leitura de diversos sensore simultaneamente; Utilização da função *scheduler* no Arduino, para colocá-lo em *stand by* a maior parte do tempo e ativá-lo apenas quando for realizada uma série de leituras periodicamente. Com este procedimento, pode-se otimizar o armazenamento; Adicionar o mapa com a localização do Arduino, e atualizar o gráfico com medidas de aceleração com filtros de datas e legendas.

6. AGRADECIMENTOS

Agradeço, em primeiro lugar a Deus, por ser minha fortaleza em todos os momentos, como também a meus pais Marcelo e Cristina, por sempre me incentivarem aos estudos e se esforçarem sempre pelo meu melhor. A minha irmã Marcelle, por participar e me incentivar no aprendizado da língua Inglesa, essencial na minha formação e a minha noiva, Sabrina, por desde a escola estar sempre presente, sendo meu porto seguro nos momentos mais difíceis, aconselhando, me incentivando e sendo uma verdadeira companheira de estudo, trabalho e vida. Agradeço também às professoras Joseana e Marília por todo o suporte necessário para o desenvolvimento deste trabalho.

7. REFERÊNCIAS

[1] ARAÚJO, C. D. (2017) Metodologia baseada em redes neurais artificiais para a detecção de danos estruturais. Em tese (Doutorado em Estruturas e Construção Civil)—Universidade de Brasília, Brasília.

[2] GALDINO, E. e CURY, A. (2017). Development of low-cost wireless accelerometer for structural dynamic monitoring. Revista Interdisciplinar De Pesquisa Em Engenharia, 2(25), 10-19.

[3] DE SOUZA, A. R. et al. (2011). A placa Arduino: uma opção de baixo custo para experiências de física assistidas pelo PC. Revista Brasileira de Ensino de Física, 33(1), 01-05.

[4] ABDULLAHI, S. I. et al. (2019). Accelerometer Based Structural Health Monitoring System on the Go: Developing Monitoring Systems with NI LabVIEW. International Journal of Online and Biomedical Engineering (iJOE), 15(07), 32-51.

[5] PALMA E SILVA, J. B. L. et al. (2019). Desenvolvimento de sistema de baixo custo para monitoramento de integridade estrutural Matéria - Rio de Janeiro, 24(4). Matéria (Rio J.) vol.24 no.4 Rio de Janeiro 2019 Epub Nov 25, 2019

[6] Acelerometro E Giroscopio 3 Eixos Mpu-6050 Mpu6050 <https://www.microwat.com.br/arduino-prototipagem/modulos-e-componentes/acelerometro-e-giroscopio-3-eixos-mpu-6050-mpu6050> Acessado em 15/05/2021

[7] Como programar o Node Mcu com arduino IDE <https://www.robocore.net/tutoriais/como-programar-nodemcu-arduino-ide> Acesssado em 15/05/2021

[8] O que é Internet of Things (IoT) <https://www.oracle.com/br/internet-of-things/what-is-iot/> Acesssado em 15/05/2021

Sobre os autores:

Marcelo Gabriel dos Santos Queiroz Vitorino. Graduando em Ciência da Computação, com 4 anos de experiência em Desenvolvimento de Software. Desenvolvedor Web Full Stack de Software na IT4Process gmbh há 1 ano e 10 meses. Possui formação acadêmica complementada com um intercâmbio acadêmico na Hochschule Koblenz em 2019.

Joseana Macêdo Fechine Régis de Araújo. Professora orientadora.

Marília Marcy Cabral de Araújo. Professora orientadora.