PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
Instituto de Ciências Exatas e Informática (ICEI)
Departamento de Ciência da Computação (DCC)
Curso de Engenharia de Computação

Apostila de Trabalhos Práticos da Disciplina Internet das Coisas

Curso de Engenharia de Computação

Elaborada pelo Professor: Júlio C. D. Conway
Revisão 1.0 de 01/2021
Revisão 2.0 de 02/2022
Sujeito a revisão

Especificações dos Trabalhos
3° Período

INTRODUÇÃO

Este documento define as regras e etapas dos trabalhos da disciplina IoT referente ao 3º período. Os trabalhos estão em ordem crescente de complexidade e correspondem às etapas de construção de um "esqueleto" do TI. Ao final dos trabalhos todas as etapas da arquitetura básica de uma aplicação de IoT estarão implementadas, e estão mostradas na a Figura 1. Bastará então aos grupos adequarem este *template* ao seu trabalho particular de TI. O trabalho 1 é a proposta de projeto.

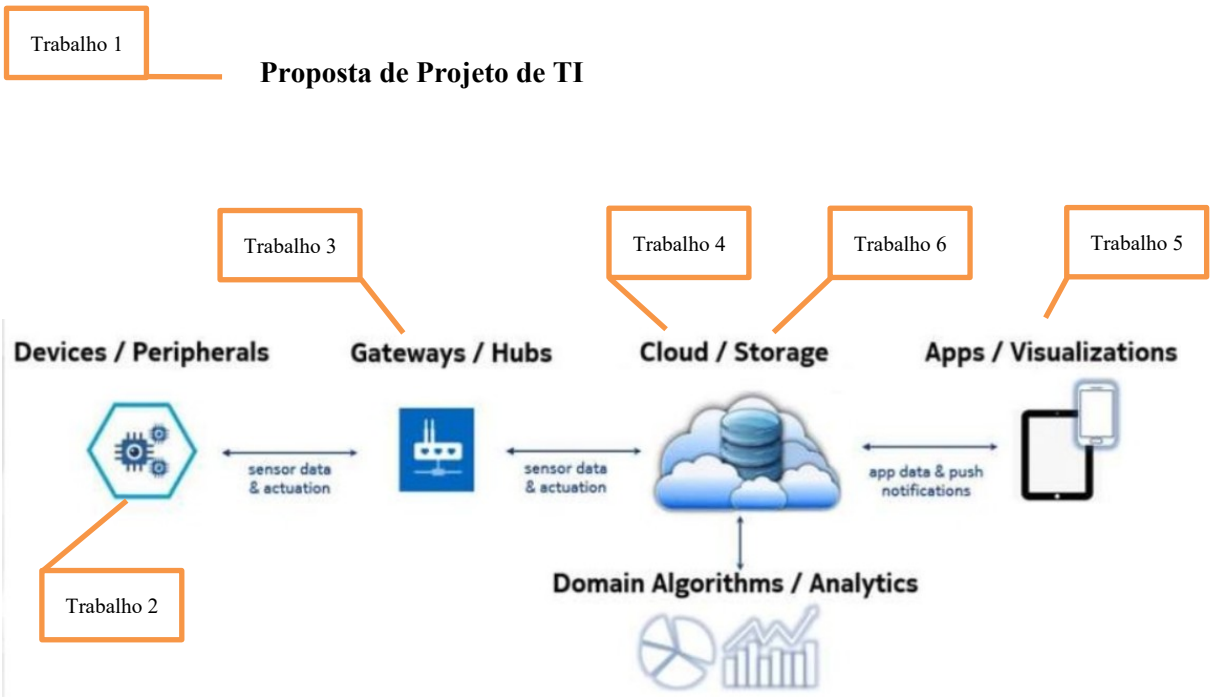

Figura 1 – Arquitetura Básica de IoT

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
Instituto de Ciências Exatas e Informática (ICEI)
Departamento de Ciência da Computação (DCC)
Curso de Engenharia de Computação

REGRAS PARA ENTREGA DE TRABALHOS

1. **Os relatórios poderão ser feitos em grupo de até quatro alunos.**

2. **Formas de Entrega**

   2.1. Vídeo

   Sempre que possível, e principalmente nas montagens práticas, pode-se gravar um vídeo. Entretanto, este vídeo deve explicar e mostrar claramente o que foi feito, e obrigatoriamente deve ou devem aparecer todos ou pelo menos um elemento do grupo no vídeo. Postar o vídeo ou o link.

   2.2. Relatório

   Todos os relatórios deverão ser postados no Canvas. Só será aceito arquivo em PDF, sendo que outras formas de entrega serão descartadas. Entregar um relatório por grupo se for o caso.

   Sugestão: vá montando (copiando) as listagens dos programas e saídas da tela, fotos de montagens no TinkerCad ou práticas em um arquivo WORD. Ao final salve como PDF e poste este arquivo.

   Obs: cada relatório descreverá a forma de entrega.

REGRA DE PONTUAÇÃO PARA TRABALHOS ENTREGUES FORA DO PRAZO:

- **1 dia de atraso: 90% do valor do trabalho**
- **2 dias de atraso: 80% do valor do trabalho**
- **3 dias de atraso: 70% do valor do trabalho**
- **4 dias de atraso: 60% do valor do trabalho**
- **5 dias de atraso: 50% do valor do trabalho**
- **6 dias de atraso: 40% do valor do trabalho**
- **7 dias de atraso: 30% do valor do trabalho**
- **Após uma semana de atraso, o trabalho não será aceito em hipótese alguma.**

Caso não consiga entregar o trabalho completo, entregue do jeito que está, no Canvas. Resumindo:

O que deve ser postado no Canvas:

- **Um arquivo em PDF contendo:**
- **Listagem do(s) seu(s) programa(s);**
- **Figura(s) de todas as da(s) tela(s) de saída em modo texto (tela escura) com a saída do programa (resultados).**

TRABALHO 1: Proposta de projeto do TI
Valor: 5 pontos

Neste trabalho deverá ser proposto o tema do Trabalho de TI, que se constitui em uma aplicação de IoT de acordo com a Figura 1. Para tanto, o grupo deverá escolher um tema para o seu Trabalho Interdisciplinar.

- Os alunos deverão criar uma aplicação IoT para enviar dados de sensores *(*temperatura, umidade, pressão, vazão, gases, etc) para a nuvem, e também e receber comandos de atuação, através de um dispositivo (device) de aquisição de dados e atuação com capacidade de comunicação em rede. O dispositivo deverá utilizar uma plataforma de prototipação, como por exemplo as sugeridas abaixo:
  – Arduino
  – Raspbarry
  – ESP32
  – ST32, etc

- Os dados deverão ser visualizados em um aplicativo mobile, que deverá permitir o envio de comandos para o dispositivo.

- O trabalho deve tentar resolver algum problema ou demostrar alguma aplicabilidade no mundo real. Abaixo estão listadas algumas sugestões de temas, mas os grupos são encorajados a proporem seus próprios temas.

Sugestões de Projetos:

- **Controle de Posto de Gasolina**
  - Simulação de Bomba de Combustível (pulsos/litro)
  - Várias Bombas
  - Visualização/Atuação no Celular
    - Litros vendidos/bomba
    - Ligar/Desligar bomba

- **Controle de Fazenda**
  - Contagem de Gado
  - Produção de Leite (litros/m)
  - Controle de acesso: Porteira
  - Atuadores
    - Fechadura
  - Visualização/Atuação no Celular
    - Abrir/Fechar Fechadura
    - Número de cabeças

- **Controle de Indústria**
  - Monitoramento de Sensores:
    - Temperatura, Fumaça, Gases,etc
  - Atuadores
    - Sprinklers
  - Visualização/Atuação no Celular
    - Abrir/Fechar sprinkler
    - Visualizar estado dos Sensores

- **Horta Inteligente**
  - Monitoramento de Sensores:
    - Temperatura
    - Umidade
  - Atuadores
    - Sprinklers
  - Visualização/Atuação no Celular
    - Abrir/Fechar sprinkler
    - Visualizar estado dos Sensores
      - Temperatura e umidade

- **Alarme Residencial/Casa Inteligente**
  - Monitoramento de Sensores:
    - Portas
    - Janelas
    - Movimento
  - Atuadores
    - Sirene
  - Visualização/Atuação no Celular
    - Abrir/Fechar portas/janelas
    - Ligar/Desligar Sirene

A proposta deverá conter os seguintes itens. Além disso, esta proposta servirá como a parte inicial do relatório final do TI.

1. **TEMA**

2. **OBJETIVOS**

   2.1. JUSTIFICATIVA
   2.2. MOTIVAÇÃO

3. **METODOLOGIA**
   3.1. Apresentar de maneira detalhada como os objetivos propostos serão alcançados, ou seja, descrever as etapas e métodos que serão utilizados para a realização do TI.
   3.2. Cada grupo deverá eleger um líder do grupo, que será responsável pelo cumprimento do cronograma. Para tanto, deverá utilizar metodologias ágeis, tais como SCRUM.

4. **CRONOGRAMA**: confeccionar o cronograma de atividades que servirá de base para o controle das atividades de cada elemento do grupo.

5. **O QUE DEVE SER POSTADO NO CANVAS:**
   - **Um arquivo em PDF contendo a proposta com os itens acima.**

TRABALHO 2: Devices/Periferals
Valor: 10 pontos

Este trabalho consiste em um projeto no **TinkerCad**, com a seguinte especificação:

1. Hardware

1.1. O projeto do hardware simulado deverá usar o Arduino para controlar os sensores e atuadores descriminados abaixo. Os sensores e atuadores abaixo são sugestões, mas pode usar os sensores e atuadores do seu projeto, desde que sejam 3 sensores e 3 atuadores.

1.2. Sensores

1.2.1. Sensor de Temperatura
1.2.2. Sensor Ultrassônico
1.2.3. Sensor de luminosidade (LDR)

1.3. Atuadores

1.3.1. Ligar uma lâmpada através de um relé caso escureça (simular com LDR baixa luminosidade);
1.3.2. Acender um LED Vermelho caso a temperatura ultrapasse um valor (escolha o valor, simulando por exemplo 40º)
1.3.3. Piscar um LED verde cada vez mais rápido, quando a distância simulada no sensor ultrassônico diminuir, ficando permanentemente aceso para uma distância zero.

2. Software

Fazer o programa de controle de sensores e atuadores conforme a especificação acima. Simular no TinkerCad o funcionamento do circuito.

2.1. O que deve ser postado no Canvas:

- Postar o link do trabalho feito no TinkerCad no SGA, de modo que se possa simular novamente o circuito, para fins de correção;

- O grupo poderá optar em fazer o trabalho com componentes físicos, já com a plataforma que será usada no TI. Para correção, deverá ser feito um vídeo com no máximo 5 minutos mostrando o funcionamento real do circuito. Está opção é estimulada, pois serve como base para a parte de hardware do TI. Postar o vídeo ou link.

TRABALHO 3: Gateways/Hubs: Familiarização com o Protocolo MQTT Valor: 5 pontos

Neste trabalho o aluno deverá explorar os conhecimentos adquiridos na teoria sobre o protocolo MQTT. Para tanto deverão ser utilizadas as seguintes ferramentas:

1.1. **Broker MQTT** de teste disponibilizado pelo Eclipse Mosquitto: test.mosquitto.org

1.2. **Cliente MQTTBox**: Baixar e usar o programa gratuito MQTTBox:

https://www.microsoft.com/pt-br/p/mqttbox/9nblggh55jzg#activetab=pivot:overviewtab

para a criação de clientes MQTT. Este programa permite a conexão de diversos clientes MQTT com o Broker de forma simultânea, possibilita a publicação e inscrição em tópicos, além de disponibilizar outras configurações.

1.3. Configurar o Broker de teste Mosquitto (test.mosquitto.org) no MQTTBox.

O projeto consiste na simulação do monitoramento de sensores e ativação/desativação de atuadores (relés) de uma indústria hipotética, onde se deseja monitorar temperatura, gases tóxicos e fumaça. Além disso, mensagens de alerta podem ser enviados devido a detecção de que algum valor de sensor ultrapassou o seu threshold (limiar). Existem também atuadores (relés) que podem ser acionados no caso de temperatura alta, presença de gases tóxicos e fumaça. A indústria hipotética consiste em três áreas, onde cada área possui os seguintes elementos:

a. Área 1:
- i. Sensor de Temperatura
- ii. Sensor de Gases Tóxicos
- iii. Relé Atuador
- iv. Aviso de Alerta

b. **Área 2**:
- i. Sensor de Temperatura
- ii. Sensor de Gases Tóxicos
- iii. Relé Atuador
- iv. Aviso de Alerta

c. **Área 3**:
- i. Sensor de Fumaça
- ii. Relé Atuador
- iii. Aviso de Alerta

2. O que deve ser feito

2.1. Você deverá elaborar o nome de cada tópico. Um tópico é um nome que segue uma estrutura hierárquica, conforme visto em sala de aula. Dê nomes sugestivos para a funcionalidade do tópico;

2.2. Para cada área acima, utilizar o MQTTBox para criar os tópico para cada elemento da respectiva área;

2.3. Para cada tópico de sensor de cada área faça:
- 2.3.1. Publicar um valor para cada sensor (um número qualquer entre 0 e 100, por exemplo).
- 2.3.2. Assinar o respectivo tópico;
- 2.3.3. No MQTTBox, capturar os *prints* do Publisher e Subscriber correspondentes, mostrando o valor publicado e o valor recebido no assinante;

2.4. Para cada tópico de atuador de cada área faça:
- 2.4.1. Publicar no respectivo tópico: Liga Rele e Desliga Rele;
- 2.4.2. Assinar o respectivo tópico;
- 2.4.3. No MQTTBox, capturar os *prints* do Publisher e Subscriber correspondentes, mostrando o valor publicado e o valor recebido no assinante;

2.5. Para cada tópico de aviso de alerta de cada área faça:
- 2.5.1. Publicar um aviso de alerta: “Limite ultrapassado” ou Operação Normal”
- 2.5.2. Assinar o respectivo tópico;
- 2.5.3. Capturar os *prints* do Publisher e Subscriber correspondentes, mostrando o valor publicado e o valor recebido no assinante;

3. O que deve ser postado no Canvas:

3.1. Um relatório em **PDF** contendo:
- 3.1.1. Nome dos integrantes do grupo se for o caso;
- 3.1.2. Objetivo do Trabalho;
- 3.1.3. Ferramentas e recursos utilizada com breve descrição de cada ferramenta;
- 3.1.4. Um desenho mostrando a estrutura hierárquica dos tópicos;
- 3.1.5. Todos os prints do MQTTBox capturados no item 2;

TRABALHO 4: Cloud
Valor: 15 pontos

1. Neste trabalho o grupo poderá acessar a nuvem para transmitir os dados reais dos sensores utilizando o protocolo MQTT. Para tanto é disponibilizado o código abaixo, que é um template completo para acesso a nuvem e compatível com o Arduino e ESP32.

```
#include <arduino.h>
#include <WiFi.h>
#include <PubSubClient.h>

#define LED_BUILTIN 2
#define PIN_LED 2


/* Definicoes para o MQTT */
#define TOPICO_SUBSCRIBE_LED         "topico_liga_desliga_led"
#define TOPICO_PUBLISH_TEMPERATURA   "topico_sensor_temperatura"

#define ID_MQTT  "IoT_PUC_SG_mqtt"     //id mqtt (para identificação de sessão)

// casa Fatima
const char* SSID     = "NET_2G43C248"; // SSID / nome da rede WI-FI que deseja se conectar
const char* PASSWORD = "F843C248"; // Senha da rede WI-FI que deseja se conectar

// casa Conway
//const char* SSID     = "VIRTUA 202"; // SSID / nome da rede WI-FI que deseja se conectar
//const char* PASSWORD = "BrCon2657"; // Senha da rede WI-FI que deseja se conectar

const char* BROKER_MQTT = "test.mosquitto.org";
int BROKER_PORT = 1883; // Porta do Broker MQTT

//Variáveis e objetos globais
WiFiClient espClient; // Cria o objeto espClient
PubSubClient MQTT(espClient); // Instancia o Cliente MQTT passando o objeto espClient

long numAleatorio;


/* Prototypes */
void initWiFi(void);
void initMQTT(void);
void mqtt_callback(char* topic, byte* payload, unsigned int length);
void reconnectMQTT(void);
void reconnectWiFi(void);
void VerificaConexoesWiFIEMQTT(void);

/*
   Implementações
*/

/* Função: inicializa e conecta-se na rede WI-FI desejada
   Parâmetros: nenhum
   Retorno: nenhum
*/
void initWiFi(void)
{
  delay(10);
  Serial.println("------Conexao WI-FI------");
  Serial.print("Conectando-se na rede: ");
  Serial.println(SSID);
  Serial.println("Aguarde");

  reconnectWiFi();
}


/* Função: inicializa parâmetros de conexão MQTT(endereço do
```

```
           broker, porta e seta função de callback)
   Parâmetros: nenhum
   Retorno: nenhum
*/
void initMQTT(void)
{
  MQTT.setServer(BROKER_MQTT, BROKER_PORT);   //informa qual broker e porta deve ser conectado
   MQTT.setCallback(mqtt_callback);              //atribui função de callback (função chamada
quando qualquer informação de um dos tópicos subescritos chega)
}

/* Função: função de callback
           esta função é chamada toda vez que uma informação de
           um dos tópicos subescritos chega)
   Parâmetros: nenhum
   Retorno: nenhum
*/
void mqtt_callback(char* topic, byte* payload, unsigned int length)
{
  String msg;

  /* obtem a string do payload recebido */
  for (int i = 0; i < length; i++)
  {
    char c = (char)payload[i];
    msg += c;
  }

  Serial.print("Chegou a seguinte string via MQTT: ");
  Serial.println(msg);

  /* toma ação dependendo da string recebida */
  if (msg.equals("L"))
  {
    digitalWrite(PIN_LED, HIGH);
    Serial.println("LED aceso mediante comando MQTT");
  }

  if (msg.equals("D"))
  {
    digitalWrite(PIN_LED, LOW);
    Serial.println("LED apagado mediante comando MQTT");
  }
}


/* Função: reconecta-se ao broker MQTT (caso ainda não esteja conectado ou em caso de a
conexão cair)
           em caso de sucesso na conexão ou reconexão, o subscribe dos tópicos é refeito.
   Parâmetros: nenhum
   Retorno: nenhum
*/
void reconnectMQTT(void)
{
  while (!MQTT.connected())
  {
    Serial.print("* Tentando se conectar ao Broker MQTT: ");
    Serial.println(BROKER_MQTT);
    if (MQTT.connect(ID_MQTT))
    {
      Serial.println("Conectado com sucesso ao broker MQTT!");
      MQTT.subscribe(TOPICO_SUBSCRIBE_LED);
    }
    else
    {
      Serial.println("Falha ao reconectar no broker.");
      Serial.println("Havera nova tentatica de conexao em 2s");
      delay(2000);
    }
  }
}

/* Função: verifica o estado das conexões WiFI e ao broker MQTT.
           Em caso de desconexão (qualquer uma das duas), a conexão
           é refeita.
   Parâmetros: nenhum
```

```
   Retorno: nenhum
*/
void VerificaConexoesWiFIEMQTT(void)
{
  if (!MQTT.connected())
    reconnectMQTT(); //se não há conexão com o Broker, a conexão é refeita

  reconnectWiFi(); //se não há conexão com o WiFI, a conexão é refeita
}

/* Função: reconecta-se ao WiFi
   Parâmetros: nenhum
   Retorno: nenhum
*/
void reconnectWiFi(void)
{
  //se já está conectado a rede WI-FI, nada é feito.
  //Caso contrário, são efetuadas tentativas de conexão
  if (WiFi.status() == WL_CONNECTED)
    return;

  WiFi.begin(SSID, PASSWORD); // Conecta na rede WI-FI

  while (WiFi.status() != WL_CONNECTED)
  {
    delay(100);
    Serial.print(".");
  }

  Serial.println();
  Serial.print("Conectado com sucesso na rede ");
  Serial.print(SSID);
  Serial.println("\nIP obtido: ");
  Serial.println(WiFi.localIP());
}


void setup() {
  Serial.begin(9600); //Enviar e receber dados em 9600 baud
  delay(1000);
  Serial.println("Disciplina IoT: acesso a nuvem via ESP32");
  delay(1000);
  // programa LED interno como saida
  pinMode(PIN_LED, OUTPUT);
  digitalWrite(PIN_LED, LOW);    // apaga o LED

  // GERANDO TEMPERATURA COMO UM NÚMERO ALEATÓIO
  // inicializa o gerador de números aleatórios.
  // um pino analógico desconectado irá retornar um
  // valor aleatório de tensão em analogRead()
  randomSeed(analogRead(0));

  /* Inicializa a conexao wi-fi */
  initWiFi();

  /* Inicializa a conexao ao broker MQTT */
  initMQTT();
}

// the loop function runs over and over again forever
void loop() {

  // cria string para temperatura
  char temperatura_str[10] = {0};


/* garante funcionamento das conexões WiFi e ao broker MQTT */
  VerificaConexoesWiFIEMQTT();

  // gera um valor aleatório de temperatura entre 10 e 100
  numAleatorio = random(10, 101);


  // formata a temperatura aleatoria  como string
  sprintf(temperatura_str, "%dC", numAleatorio);

  /*  Publica a temperatura */
```

```
  MQTT.publish(TOPICO_PUBLISH_TEMPERATURA, temperatura_str);

  Serial.print("Gerando temperatura aleatoria: ");
  Serial.println(temperatura_str);

  /* keep-alive da comunicação com broker MQTT */
  MQTT.loop();

  /* Refaz o ciclo após 2 segundos */
  delay(2000);
}
```

2. O que deve ser postado no Canvas:

2.1. Um relatório em **PDF** contendo:

2.1.1. Nome dos integrantes do grupo se for o caso;
2.1.2. Objetivo do Trabalho;
2.1.3. Ferramentas e recursos utilizados com breve descrição de cada ferramenta;
2.1.4. Listagem do Programa
2.1.5. Um vídeo curto (máximo 5 minutos) mostrando o funcionamento completo do trabalho.

TRABALHO 5: Apps/Visualizations
Valor: 5 pontos

1. O trabalho deverá ter um Aplicativo Mobile para visualização dos dados de sensores e botões.

Existem alguns aplicativos nos quais somente se configura visualizações e comandos, não sendo necessário desenvolver código, como por exemplo o MQTT DASH:
https://www.embarcados.com.br/mqtt-dash/
https://www.filipeflop.com/blog/esp32-e-mqtt-dashboard-android/

Entretanto fica a cargo do grupo decidir se usa uma solução pronta ou desenvolve um aplicativo mobile, por exemplo, em conjunto com outra disciplina, no TI.

Para este trabalho, utilizar os sensores e atuadores do trabalho 2 desta especificação.

Deverá ser apresentado, em conjunto com os códigos, um vídeo de no máximo 3 minutos mostrando o funcionamento do aplicativo mobile, em conjunto com o hardware..

2. O que deve ser postado no Canvas:

2.1. Um relatório em **PDF** contendo:
- 2.1.1. Nome dos integrantes do grupo se for o caso;
- 2.1.2. Objetivo do Trabalho;
- 2.1.3. Ferramentas e recursos utilizados com breve descrição de cada ferramenta;
- 2.1.4. Um vídeo curto (máximo 5 minutos) mostrando o funcionamento completo do trabalho.

TRABALHO 6: Banco de Dados (Storage)
Valor: 15 pontos

Neste trabalho o aluno deverá explorar os conhecimentos adquiridos sobre Banco de Dados, tanto na Disciplina de BD quanto os conhecimentos passados em sala de aula. Este BD será local a uma máquina padrão (PC ou Notebook). Para tanto deverão ser utilizadas as seguintes ferramentas:

1.1. **Broker MQTT** de teste disponibilizado pelo Eclipse Mosquitto: test.mosquitto.org

1.2. Utilizar o Template que será fornecido para acessar o Broker e atualizar o BD.

1.3. Baixar e instalar o BD MySQL WorkBench.

1.4. Baixar versão do Python 3.7.x ou abaixo. Não instale a versão 3.8 (ainda está com problemas)

1.5. Instalar um ambiente de desenvolvimento para Python. Sugestão: PYCHARM

O projeto consiste na simulação do monitoramento do sensor de temperatura utilizado no trabalho 2. Através do template em Python a ser fornecido, os dados do tópico de temperatura do trabalho 2 deverão ser armazenados no BD local.

2. O que deve ser feito

2.1. Usar o mesmos tópicos do sensor de temperatura;

2.2. Assinar o tópico na aplicação a ser desenvolvida, baseada no template fornecido;

2.3. Guardar o dado no BD;

2.4. Visualizar os dados no MySQL Workbench

2.5. Após guardar os dados no BD, mostrar na tela (tela simples, modo texto) e ficar atualizando assim que os dados chegarem no (s) tópico (s).

3. O que deve ser postado no Canvas:

3.1. Um relatório em **PDF** contendo:

3.1.1. Nome dos integrantes do grupo se for o caso;
3.1.2. Objetivo do Trabalho;
3.1.3. Ferramentas e recursos utilizados com breve descrição de cada ferramenta;
3.1.4. Descrição do Hardware, mostrando todos os sensores que estão sendo monitorados;
3.1.5. Todos os prints capturados no item 2.4;
3.1.6. Um vídeo curto mostrando todos os acessos especificados, com pelo menos um elemento do grupo apresentado. Postar o link do vídeo.

Roteiro para o vídeo ou captura de telas:

1. Colocar o device (ESP32, Arduino, etc) no ar, interligando com o teste.mosquito.org.
2. Verificar se está tudo correto com o MQTTBOX.
3. Executar o template de acesso aos tópicos através do Python. O código está abaixo e também no Canvas, como mostra a Figura 2.
4. Usar o MQTTBOX para visualizar os dados dos tópicos assinados e também para executar os comandos "termina" e "delete", conforme está no template de código.

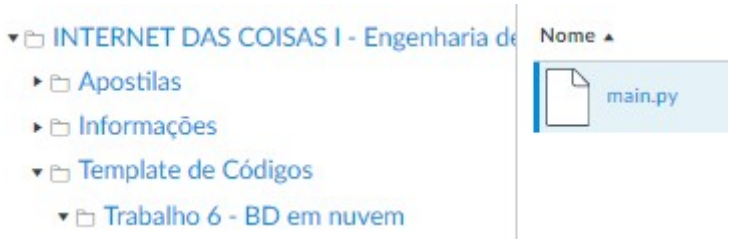

Figura 2

5. Exemplo de Aplicação de acesso a um BD local Python

```
import paho.mqtt.client as mqtt

# importe o conector do Python com o MySQL: instalar novamente neste env
import mysql.connector

# agora é necessário criar um objeto de conexão: encontra o MySQL e informa as credenciais
para se conectar ao banco
# instalar novamente o concetor: pip install mysql-connector-python
con                          =                          mysql.connector.connect(host='localhost',
database='IoT',user='root',password='BrCon2657')

# verifique se a conexão ao BD foi realizada com sucesso
if con.is_connected():
    db_info = con.get_server_info()
    print("Conentado com sucesso ao Servidor ", db_info)

    # a partir de agora pode-se executar comandos SQL: para tanto é necessário criar um objeto
tipo cursor
    # o cursor permite acesso aos elementos do BD
    cursor = con.cursor()

     # agora você pode executar o seu comando SQL. Por exemplo o comando "select database();"
mostra o BD atual selecionado
    cursor.execute("select database();")
    # crio uma variável qualquer para receber o retorno do comando de execução
    linha = cursor.fetchone()
    print("Conectado ao DB", linha)

    # createTable é usada no comando SQL para  criar a tabela dadosIoT, que só tem um registro
chamado "mensagem"
    createTable = """
                CREATE TABLE dadosIoT (id INT AUTO_INCREMENT,
                                       mensagem TEXT(512),
                                       PRIMARY KEY (id));
            """

    # este par try/except verifica se a tabeja  já está criada. Se a tabela não existe, cai no
try e é criada
    # se existe, cai no except e só mostra a mensagem que  a tabela existe
    try:
        cursor.execute(createTable)
    except:
        print("Tabela dadosIoT já existe.")
        pass

def print_hi(name):
    # mensagem inicial
    print(f'Hi, {name}')  # Press Ctrl+F8 to toggle the breakpoint.

# esta função é a função callback informando que o cliente se conectou ao servidor
def on_connect(client, userdata, flags, rc):
    print("Connectado com codigo "+str(rc))

    # assim que conecta, assina um tópico. Se a conexão for perdida, assim que
    # que reconectar, as assinaturas serão renovadas
    client.subscribe("topico_sensor_temperatura")


# esta função é a função callback que é chamada quando uma publicação é recebida do servidor
def on_message(client, userdata, msg):
    print("Mensagem recebida no tópico: " + msg.topic)
    print("Mensagem: "+ str(msg.payload.decode()) + "°")

    # ao receber um dado, insere como um registro da tabela dadosIoT
    cursor = con.cursor()
                    cursor.execute("INSERT      INTO      dadosIoT      (mensagem)      VALUES
('{}')".format(str(msg.payload.decode())))
    con.commit()
```

```
    cursor.execute("SELECT * FROM dadosIoT")
    myresult = cursor.fetchall()
    print(myresult)

    if str(msg.payload.decode().strip()) == "termina":
        print("Recebeu comando termina.")
        if con.is_connected():
            cursor.close()
            con.close()
            print("Fim da conexão com o Banco dadosIoT")

    if str(msg.payload.decode().strip())  == "delete":
        cursor.execute("TRUNCATE TABLE dadosIoT")


if __name__ == '__main__':
    print_hi('Olá Turma.')

    client = mqtt.Client()
    client.on_connect = on_connect
    client.on_message = on_message

    client.connect("test.mosquitto.org", 1883, 60)
    #client.connect("broker.hivemq.com", 1883, 60) # broker alternativo

    # a função abaixo manipula trafego de rede, trata callbacks e manipula reconexões.
    client.loop_forever()
```

Especificações do TI
Valor: 25 pontos

O trabalho de TI deverá ser uma aplicação de Internet das Coisas (IoT), baseado na Arquitetura padrão mostrada na Figura 1. A princípio o TI deverá ser feito em grupos de até 4 alunos (o mesmo grupo dos trabalhos), sendo que cada grupo deverá apresentar uma proposta de tema do TI. Todos os componentes desta arquitetura serão abordados tanto nas aulas teóricas quanto nos trabalhos práticos da disciplina, e todo o suporte básico de hardware e software será dado, cabendo aos grupos irem além, de modo que, baseado neste conhecimento básico, elaborarem o seu Trabalho Interdisciplinar de acordo com a sua proposta.

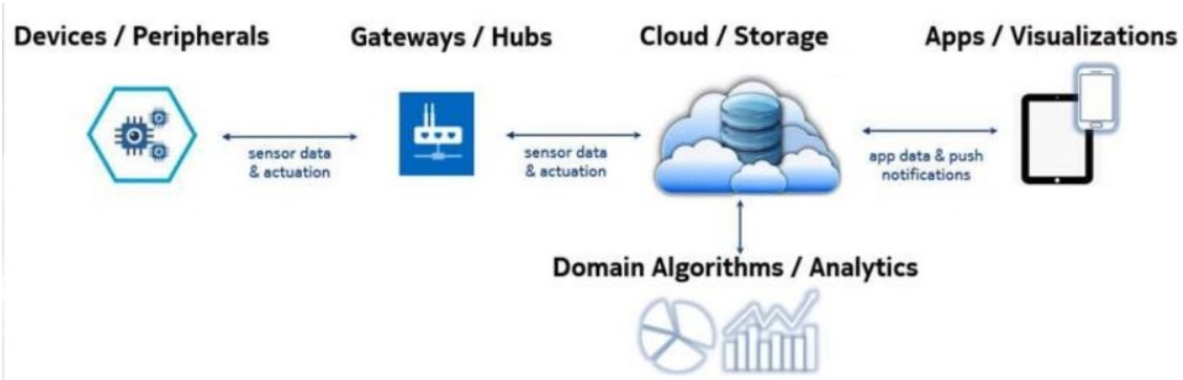

Figura 1 – Arquitetura Básico de uma Aplicação IoT

1. Requisitos mínimos:

1.1 – O trabalho deverá contemplar:

1.1.1 Um dispositivo de coleta de dados de sensores com capacidade de envio destes dados para a nuvem, utilizando plataformas de prototipação (reais ou simuladas):
1.1.1.1 – Arduino
1.1.1.2 – Raspbarry
1.1.1.3 – ESP32
1.1.1.4 – ST32, etc

1.1.2 – Um protocolo de Comunicação em Nuvem
1.1.2.1 – MQTT (recomendado)
1.1.2.2 – Sockets

1.1.3 Servidor em Nuvem (existem várias opções free)
1.1.3.1 – AWS
1.1.3.2 – Google Cloud
1.1.3.3 – Cloud MQTT
1.1.3.4 – Azure

1.1.4 O trabalho deverá ter um Aplicativo Mobile
Existem alguns aplicativos nos quais somente se configura visualizações e comandos, não sendo necessário desenvolver código, como por exemplo o MQTT DASH (https://www.embarcados.com.br/mqtt-dash/).
Entretanto fica a cargo do grupo decidir se usa uma solução pronta ou desenvolve um aplicativo mobile, por exemplo, em conjunto com outra disciplina, no TI.

1.2 Qualquer elemento da Plataforma de Prototipação/Simulador escolhido poderá ser utilizado:
Led, Led RGB, Sensor de Luz, Buzzer, Sensor de distância, sensor de Temperatura e umidade do ar, Sensor de unidade do solo, Servo motor ou outro elemento adicional que o grupo projetar;

1.3 Deverá ser apresentado, em conjunto com os códigos,

RELATÓRIO FINAL

O relatório final deverá ser postado no Canvas contendo:

1.4.1 – Introdução
1.4.2 Objetivos
1.4.2.1 Objetivo Geral
1.4.2.2 Objetivos Específicos
1.4.3 Motivação
1.4.4 Projeto de Hardware
1.4.4.1 Diagrama Elétrico
1.4.5 Projeto de Software
1.4.5.1 Fluxograma e descrição de todas as funcionalidades do programa;
1.4.5.2 Linguagem utilizada e justificativa da sua utilização;
1.4.6 Custo do Projeto
1.4.7 Conclusão
1.4.8 Anexos (que o grupo julgar necessário)

1.5 – LISTAGEM DOS PROGRAMAS:

A listagem de todos os programas deverá ser entregue.

1.6 – VÍDEO

Um vídeo de no máximo 20 minutos com todos os integrantes do grupo, onde cada um apresentará uma parte de vídeo.

2) O que será avaliado pelo professor:

- O cumprimento dos requisitos mínimos, cada item não cumprido reduz a nota;
- A entrega do Relatório;
- A entrega dos códigos Fonte;
- A originalidade e inovação do sistema proposto (só acender leds não é nem um nem outro!)
- Aplicação de Metodologia ágil: as equipes devem evidenciar como foi aplicada metodologia ágil/Scrum no desenvolvimento do projeto.

3) Formato da entrega:

O grupo deverá apresentar o trabalho distribuído conforme as pastas ilustradas na Figura 1 a seguir. Elas deverão ser compactadas e submetidas pelo SGA ou Canvas nas datas estipuladas no SGA. Não existe adiamento.

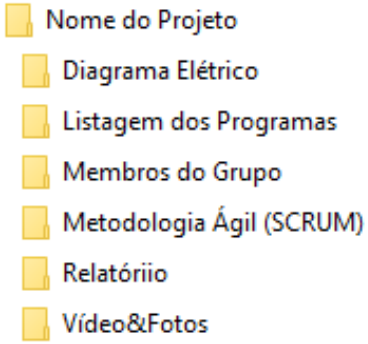

Figura 1 – Formatação das pastas no trabalho a ser submetido

4) Critério de avaliação

Distribuição dos pontos no trabalho	
Requisitos Mínimos/Relatório	6
Hardware	8
Software	8
Metodologia Ágil / Inovação proposta	3
Total	25