UNIVERSIDADE FEDERAL DE SANTA CATARINA

LABORATÓRIO DE EFICIÊNCIA ENERGÉTICA EM EDIFICAÇÕES

FELIPE AUGUSTO SANTOS BERTELLA

OCTOPUS, CENTRAL MODULE

Florianópolis,
2023

INTRODUÇÃO

Occupant-Centric Tool to Observe Practices of User and their Satisfaction - Octopus é um projeto desenvolvido pelo Pós-Doutorando e Pesquisador Mateus Bavaresco, que tem como objetivo desenvolver um produto responsável por monitorar, coletar e armazenar dados do ambiente interno, a fim de estudar o comportamento de usuários deste ambiente e poder agir para aumentar a eficiência da energia consumida.

Este relatório está voltado para o desenvolvimento de um protótipo de um módulo central responsável pelas funções de sensoriamento de variáveis gerais do ambiente, recebimento de dados dos sensores periféricos e armazenamento destes valores. Para realizar esta função, foi utilizado como componente principal um computador Raspberry Pi 3B+.

MONTAGEM

Para o desenvolvimento do módulo central, foram utilizados, além da Raspberry, sensores de variáveis gerais do ambiente, uma protoboard e cabos. Os seguintes sensores foram utilizados:

Grove - Dust Sensor - Sensor de Poeira e Fumaça: Sensor responsável pelo monitoramento da quantidade de poeira e fumaça no ambiente. Sua unidade é pcs/0.01cf (partículas por 0,01 pés cúbicos). Para a sua instalação conectar o pino 1 no GND, o pino 3 no 5V DC e pino 4 no GPIO 4. Demanda somente de uma porta digital para funcionar, pois trabalha num sistema de contagem de tempo LPO (Low pulse occupancy).

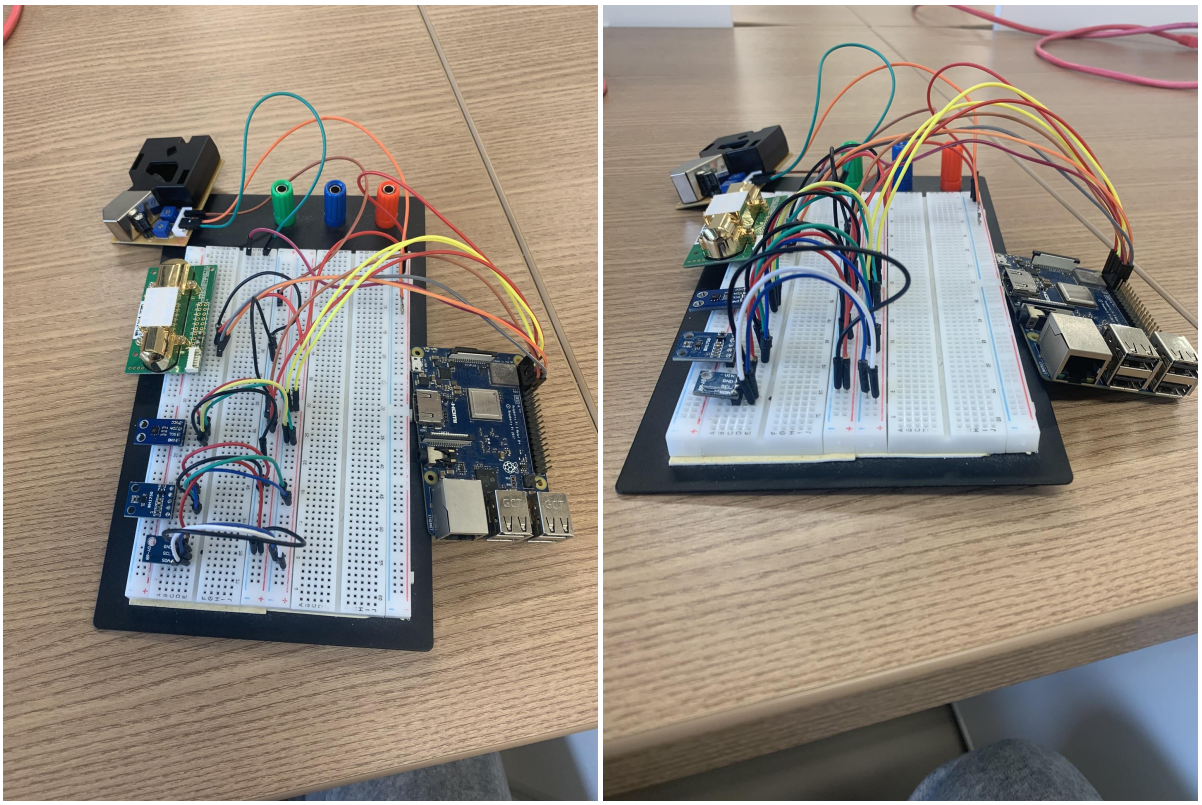
Bosch BMP180 - Sensor de Pressão e Temperatura: Sensor responsável pelo monitoramento da temperatura e da pressão atmosférica do ambiente. Funciona com uma interface i2c, portanto para a sua montagem é necessário conectar o pino VCC na saída 3,3V do módulo central, seu pino 2 no GND, pino SCL no GPIO 5 e pino SDA no GPIO 3. Utiliza uma biblioteca própria da Adafruit para seu funcionamento.

ROHM BH1750 - Sensor de quantidade de luz: Sensor responsável por medir a quantidade de luminosidade no ambiente. Sua unidade é Lux. Funciona com uma interface i2c, portanto para a sua montagem é necessário conectar o pino VCC na

saída 3,3V do módulo central, seu pino 2 no GND, pino SCL no GPIO 5 e pino SDA no GPIO 3. Utiliza também uma biblioteca da Adafruit para seu funcionamento.

Winsen MH-Z14A - Sensor de CO2: Sensor responsável por medir a quantidade de gás dióxido de carbono no ambiente. Sua unidade é ppm (partículas por milhão). Funciona através de comunicação UART. Para isso, conectar o pino 16 no GND, o pino 17 na saída 5V do módulo central, o pino 18 (RX do sensor) no GPIO 8 (TX do módulo) e o pino 19 (TX do sensor) no GPIO 10 (RX do módulo). Para calibrar o sensor, conectar seu pino 8 ao GND por 7 segundos.

Texas Instruments HDC1080 - Sensor de Temperatura e Umidade: Sensor responsável por medir a umidade relativa do ar do ambiente. Funciona com uma interface i2c, portanto para a sua montagem é necessário conectar o pino VCC na saída 3,3V do módulo central, seu pino 2 no GND, pino SCL no GPIO 5 e pino SDA no GPIO 3.



Imagens da prototipagem do Módulo Central e seus sensores

FUNCIONAMENTO

Todos os códigos estão disponíveis em: https://github.com/felipebertella/Octopus

O código feito para funcionar como módulo central realiza apenas três funções no momento: receber dados dos sensores, interpretá-los e armazená-los em uma database SQLite3. Para realizar esta função, ele está dividido em algumas partes essenciais.

1) Código dos sensores:

Para cada sensor, existe um código responsável pelo seu funcionamento. Os códigos estão localizados dentro da pasta CentralModule. O código principal que realiza as funções gerais está nomeado como main.py, nele está contido o loop principal e duas funções. Dentro deste loop, a função responsável por fazer a leitura dos sensores é chamada 10 vezes com intervalos de 30 segundos.

Ao iniciar o código, são importadas as duas classes “Monitoring” e “LocalDatabase”, localizadas, respectivamente, nos códigos “sensors.py” e “Database.py”. Após isso, são iniciadas as instâncias

Para realizar a leitura dos sensores, a função “readSensors()” é chamada. Dentro dela, existe outra função que está referenciada no código responsável pela leitura de todos os sensores, o código “sensors.py”. Dentro dele, existe uma classe chamada “Monitoring()”, que contém as funções que vão realizar a leitura dos sensores e armazenar os dados lidos em uma lista. Essa lista é formatada da seguinte maneira:

values = [LUX, TEMPERATURE, HUMIDITY, SMOKE AND DUST, CO2]

Após as 10 leituras e armazenamentos na lista, a função Average é chamada recebendo como parâmetro a lista com todos os valores adicionados. Nessa função é implementada uma manipulação de dados para calcular a média dos valores de cada sensor neste período de 5 minutos.

Depois de ter a média calculada e armazenada em uma nova lista, a função “database.insertData(averageData)” é chamada. Esta função está referenciada a uma classe localizada no código “database.py”, que insere o dado médio das últimas 10 leituras na database .sql.

2) Armazenamento:

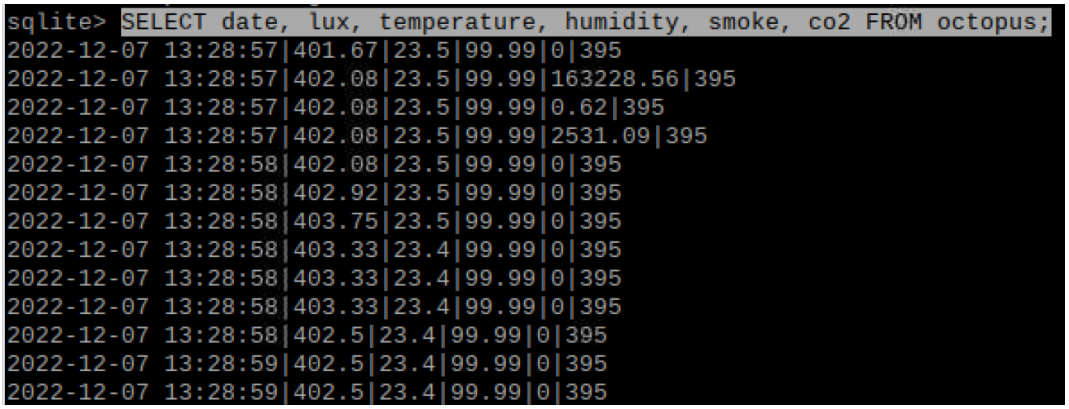
Como já mencionado, o armazenamento de dados é realizado em uma database do tipo SQLite 3. Para iniciar uma database, crie um arquivo .db na pasta onde os códigos estão localizados. Abra o arquivo Database.py, e altere o caminho referenciado na linha 13 do código, indicando o nome do arquivo .db. Após isso, inicialize o código main.py, que será responsável por criar uma tabela dentro deste arquivo.

Para acessar os dados armazenados na database, é necessário abrir a janela de comando e acessar o diretório onde estão localizados os códigos. Para fazer isso, coloque a seguinte sequência de códigos na janela:

```
cd Octopus
cd Rasp-Module
cd Config
cd CentralModule
sqlite3 Database.db
```

Ao colocar estes comandos, o arquivo .bd será aberto dentro da janela de comando e suas informações estarão disponíveis para coleta e amostragem. Para analisar os dados presentes na tabela criada, inserir o seguinte comando, mudando o nome da tabela caso tenha criado com um valor diferente:

```
SELECT date, lux, temperature, humidity, smoke, co2 FROM octopus;
```



Para exportar os arquivos para um arquivo .csv, crie um arquivo .csv na pasta onde a database está localizada e realize os seguintes passos dentro da instância do sqlite3 na janela de comando:

SELECT date, lux, temperature, humidity, smoke, co2 FROM octopus;

.tables

.header on

.mode csv

.output data.csv (insira o nome do arquivo criado)

3)Inicialização:

Após realizar a montagem do módulo de acordo com o determinado, é necessário inicializar um processo na janela de comando da Raspberry. Ao abri-la, digite (sem as aspas): “sudo pigpiod”. Esse comando inicializa as funções de input e output da Raspberry, possibilitando a aquisição de dados.

Após isso, digite na janela de comando: “i2cdetect -y 1”, ele tem a função de informar ao usuário quais id’s de comunicação I2C estão comunicando com a Raspberry. É um comando muito útil para verificação de falhas. No caso do módulo central, é necessário ter 3 Id’s aparecendo na matriz que será formada.

Por fim, abra o arquivo main.py contido dentro das pastas do projeto e inicialize-o. Caso a montagem esteja correta, os valores já começarão a ser coletados.

CONCLUSÃO

O projeto tem como objetivo monitorar o comportamento dos usuários de um ambiente e armazenar dados a respeito dele, estabelecendo um padrão para, em um futuro, poder controlar as variáveis para mantê-las otimizadas e eficientes para cada usuário. Em função disso, o desenvolvimento de um módulo central que coleta e armazena essas informações se fez necessário. Através do uso de sistemas computacionais e sensores elétricos, foi possível garantir precisão dos dados gerados além do processamento eficiente deles. Portanto, o desenvolvimento deste projeto é vital para o objetivo, visto que garante para o sistema as funções

necessárias para iniciar um projeto que visa tornar o uso da energia elétrica em ambientes mais eficiente.