IMPLEMENTAÇÃO DO PROJETO 4 - MONITORAMENTO DE UM AMBIENTE - UFRJ

Introdução

A implementação do projeto de monitoramento de um ambiente controlado foi dividida em três partes: parte eletrônica (hardware), parte lógica (software) e uma parte mecânica (caracterização do ambiente - caixa)

Falaremos inicialmente da parte eletrônica, depois de software e depois da parte mecânica (montagem da caixa). Explicaremos alguns conceitos tecnológicos, de maneira formal, da implementação das três partes para melhorar o entedimento da implementação.

* Esses conceitos foram adquiridos com pouca leitura e se mostraram úteis (funcionais) na prática. Logo, algumas coisas podem não ser exatamente como descrevemos, mas foi dessa forma que entedemos e conseguimos implementar.

Parte eletrônica

Materiais utilizados:
(+) Dois sensores:
1. Sensor de temperatura e umidade (DHT11)
2. Sensor de temperatura, pressão atmosférica e altitude (nível do mar) - (BMP280).

(+) Placa microcontroladora: NodeMCU - ESP8266
(+) Nove fios dupoint
(+) Um carregador de celular (5v tensão para placa)

Implementação:

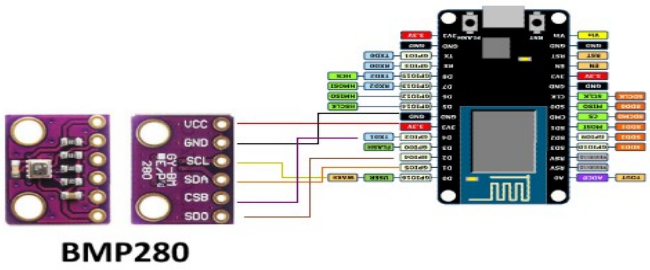
A implementação foi feita ligando os sensores à placa, usando os fios dupoint, e emitindo 5v de tensão para a mesma funcionar.

O sensor BMP280 possui um microcontrolador capaz de aceitar e receber dados, então a conexão dele com qualquer placa microcontroladora deve ser feita através de um protocolo. Usamos o protocolo SPI (Serial Perifericall Interface), apesar dele suportar também o protocolo I2C (Inter Integration Circuit).

O protocolo SPI trabalha com a determinação, mestre - escavo, dos dispositivos conectados, i.e, ele define quem é o mestre e quem são os escravos da conexão. Nesse caso, só temos um mestre (placa microcontroladora) e um escravo (BMP280 - que é o único sensor que possui um microcontrolador, o DHT11 não possui, ele usa meios mais “simples” de capturar os dados e enviar até a placa).

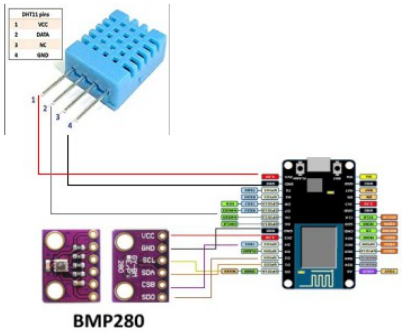
O sensor mestre (placa) e o escravo (sensor BMP) se comunicam usando quatro fios (linhas):
1. Uma linha CLK -> Linha de clock (pulso) da conexão, pois essa é conexão síncrona, isto é, envia ou recebe dados (realiza funções) segundo pulsos eletrônicos periódicos.
2. Uma linha MISO (Master Input Slave Output) -> Uma linha de saída do escravo e entrada na placa (usada no caso, para receber os dados capturados)
3. Uma linha MOSI (Master Output Slave Input) -> Uma linha de entrada no escravo e saída do mestre (usada para que o sensor receba comandos da placa, mas na configuração em software, não implementamos isso)
4. Uma linha CS (Define quem é o mestre e quem é o escravo) -> Se ela estiver setada em 1, ou seja, estiver ligada, o mestre é quem manda corrente para ela. Se ela estiver em 0, o mestre é quem manda dados pelas outras linhas de comunicação.

Essas quatro linhas são ligadas nos respectivos pinos do sensor (BMP) e pinos digitais da placa. Além disso, demos tensão para o sensor (3v3) e fechamos o circuito com atteramento (GND).



O outro sensor DHT11 foi mais simples de ligar, até porque ele não precisa de um protocolo de comunicação para enviar ou receber dados. Ele apenas envia dados sob forma de corrente, que são interpretadas segundo uma rotina (em software) pela placa.

Logo, a configuração eletrônica (completa) da placa ficou assim:




Após a ligação desses nove fios, passamos para implementação em software.

Parte de software: Consiste no código da placa + aplicação web (dashboard) como interface pro usuário

Usamos o ArduinoIDE para programar, compilar, fazer o upload do código para placa e acompanhar sua execução (do código) de forma serial.

Baixamos extensões pro ArduinoIDE, para fazê-lo reconhecer a placa e trabalhar com ela e usar algumas rotinas (do repositório do Arduino) pro microcontrolador (ESP8266) dela.

As rotinas que usamos:

*#include <ESP8266WiFi.h>* -> Habilitar o Wifi
*#include <PubSubClient.h>* -> Permitir a conexão, envio e recebimentos de dados segundo protocolo MQTT
*#include <DHT.h>* -> Usar o sensor DHT11
*#include <ArduinoJson.h>* -> Trabalhar com mensagens em json
*#include <Adafruit_Sensor.h>* -> Usar o sensor BMP
*#include <Adafruit_BMP280.h>* -> Usar o sensor BMP

O código da placa consiste em rotinas que utilizam de classes, funções, dessas bibliotecas incluidas, para fazer certas funções.

O código está todo comentado e pode ser encontrado no repositório de onde encontra-se esse projeto.

Em linhas gerais, temos:

f: habilita Wifi -> f: conecta MQTT -> f: sensorizaAmbiente -> f: envia dados MQTT

Além disso, temos uma função, que lida com eventos que chegam em MQTT. Eles chegam em formato json, são desserializados e setam uma variável no código, de forma que a aplicação web possa mudar dados na placa. Nesse caso, a aplicação web muda o intervalo de notificação dos dados monitorados.

g: eventos MQTT <- aguarda dados da aplicação web

A aplicação web foi feita no NodeRed, uma plataforma de programação em blocos da IBM, escrita em Javascript e que funciona em Node.js (Javascript Server-Side). Ela pode ser acessada no link que encontra-se no repositório.

E por fim, a montagem da caixa e seus respectivos testes.

Parte mecânica:

Fizemos dois furos na caixa, um para passar os fios, de forma a deixar os sensores dentro da caixa e outro para inserir um tubo, de forma a inserir “elementos” modificadores, como ar quente, gás nitrogênio etc.

Todos os testes se apresentaram bem sucedidos

O gás nitrogênio, por exemplo, inserido num laboratório mostrava uma mudança significativa dos dados coletados no tempo. Menor umidade, diminuição da temperatura etc.