Integrante: João Vitor - 15/00133229

# Integração dos sistemas eletrônicos

Conforme escolhido, será implementado uma rede *mesh, sendo* suportado até 254 estações, em que todos os estações irão poder comunicar-se entre si. Contudo, a central será a *master* da rede, conhecida como gateway com o ID 0.

Na Figura H está representado o esquemático de componentes a raspberry pi 3, na esquerda, será uma estação responsável por tratar e condicionar dados dos sensores, a msp430, embaixo, outra estação responsável pelos atuadores e senesores analogicos, raspberry pi 3 b+, na direta. é a central do sistema eletrônico e será a Gateway, responsável por:

* A rede *mesh* permite que dispositivos sejam endereçados pelo gateway (central), de modo que não precisa se preocupar com o endereçamento de cada um das estações ou a topologia da rede. Quando um nó está fora do alcance direto, ele faz uma rota através dos outros nós para então entregar os dados.
* As estações podem tanto transmitir quanto receber dados e para isso o rádio permite a comunicação através de dois canais, tal como qualquer rede bidirecional.
* Quando um nó se perde, ele deve se reconectar automaticamente à rede (se possível).

Utilizando a biblioteca do módulo NRF24L01, será feita todas as configurações apresentadas nas referências e atraves do projeto disponivel no repositório <https://github.com/nRF24/RF24> será feita a configuração da raspberry pi 3 no modo de estação, usando a biblioteca <https://github.com/spirilis/Enrf24> será instalado no IDLE Energia para configurar a comunicação para a msp430 e por fim usando o repositório <https://github.com/nRF24/RF24Network> será implemetando o gateway na central raspberry 3 b+.

Com essas configurações feitas espera-se que as estações consiga comunicar-se entre si configurando os IDs das estações de forma dinâmica e com a central com banda de 2.400-2.500MHz são reservadas para aplicações industriais, científicas e médicas (ISM) e possui taxa de transmissão de dados de 1M bit/s.

Referência([Anatel atribuição de faixas de frequência no Brasil](https://www.anatel.gov.br/Portal/verificaDocumentos/documento.asp?numeroPublicacao=314474&pub=original&filtro=1&documentoPath=314474.pdf))

**Hardware para comunicação entre estações e atuadores com a central**

A definição de hardware foi tomada baseando-se nos principais requisitos de comunicação do projeto: uma comunicação que oferece robustez e confiabilidade de entrega e coleta de pacotes de dados, com possibilidade de ser modelada e inserida a uma rede mesh ou estrela, baixo custo e consumo energético, alcance de até 100m e que estivesse dentro de uma faixa de frequência de operação permitida pela legislação brasileira.

O módulo nRF24L01, fabricado pela Nordic Semiconductor, cumpre os requisitos desejados e é facilmente encontrado para pronta entrega a um preço acessível. Ele opera na faixa de 2.4GHz, banda de 2.400-2.500MHz são reservadas para aplicações industriais, científicas e médicas (ISM) , além disso, possui um alcance de até 1Km em ambiente aberto.

Referência([Anatel atribuição de faixas de frequência no Brasil](https://www.anatel.gov.br/Portal/verificaDocumentos/documento.asp?numeroPublicacao=314474&pub=original&filtro=1&documentoPath=314474.pdf))

Adequado para projetos de baixíssimo consumo de energia, por possuir modos de economia de energia eficientes, o módulo suporta transmissões de 250Kbps, 1M bps e 2M bps.

As características principais do nRF24L01 incluem:

* 125 canais a 1 Mbps ou 63 canais a 2 Mbps
* Modulação GFSK
* De 900nA (modo power down) a 13.5mA(modo Rx a 2M bps)
* Comunicação SPI
* Half Duplex
* Até 6 conexões simultâneas sem interferência
* Preço 24,99 por unidade. Total 74,97 em 3 unidades.

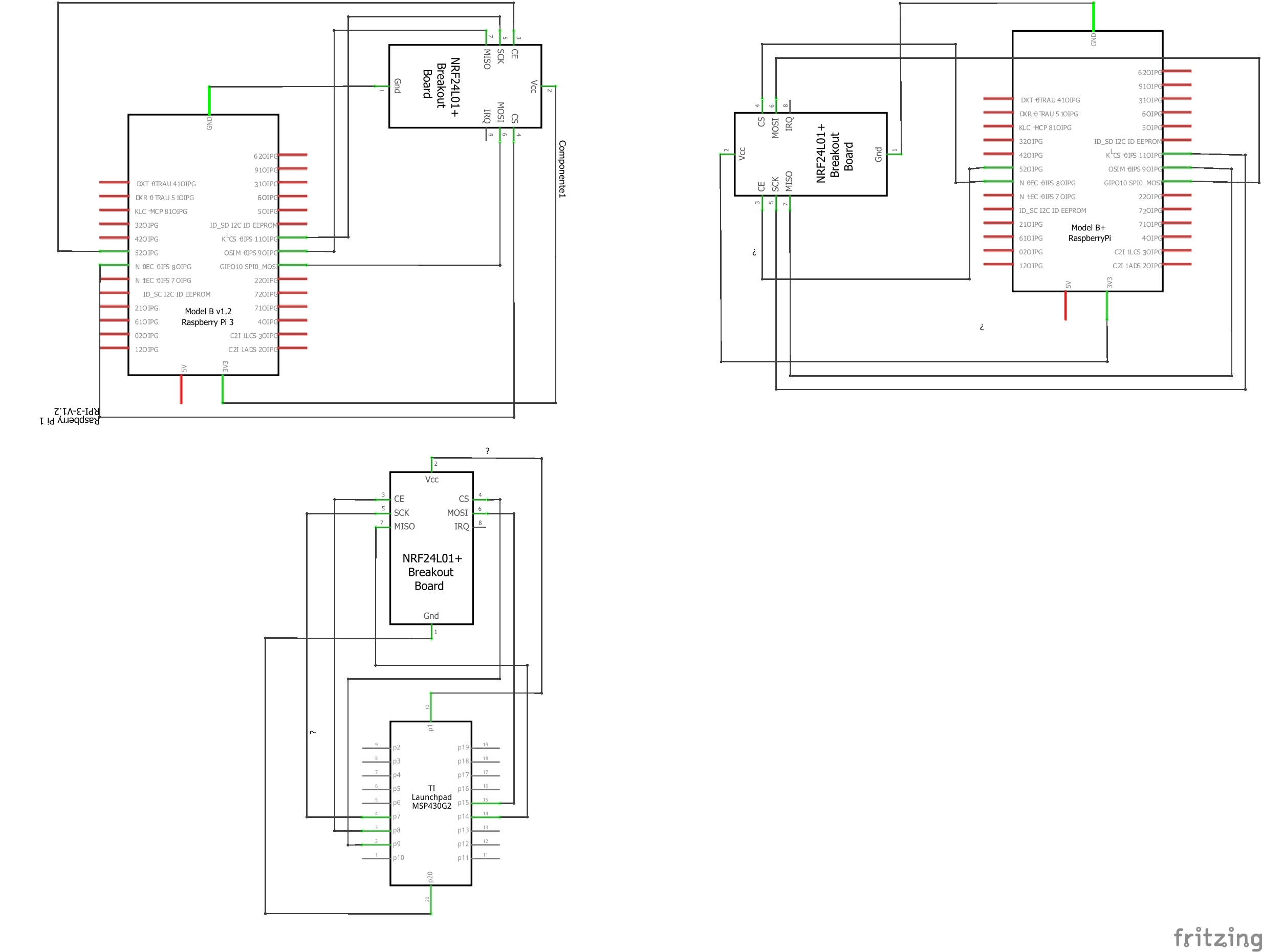


Figura H - Esquemático de pinagem para os sistemas de comunicação interna eletrônica.

Referência ([nRF24L01 Single Chip 2.4GHz Transceiver Product Specification](https://datasheet.octopart.com/NRF24L01-Nordic-Semiconductor-datasheet-10541936.pdf))

Referência:(<https://infocenter.nordicsemi.com/index.jsp?topic=%2Fcom.nordic.infocenter.sdk5.v12.0.0%2Fesb_users_guide.html>)

refs: <https://www.dobitaobyte.com.br/como-configurar-nrf24l01-com-raspberry/>

refs: <https://github.com/nRF24/RF24>

refs: <https://www.dobitaobyte.com.br/comunicacao-do-raspberry-com-arduino-utilizando-nrf24l01/>

refs: [https://www.dobitaobyte.com.br/rede-mesh-com-ardui no-e-raspberry-usando-nrf24l01/](https://www.dobitaobyte.com.br/rede-mesh-com-arduino-e-raspberry-usando-nrf24l01/)

refs: <https://github.com/nRF24/RF24Network>

refs: <https://www.dobitaobyte.com.br/como-configurar-nrf24l01-com-raspberry/>

refs: <https://unboxnbeyond.wordpress.com/2013/05/22/connecting-msp430-and-stellaris-launchpad-through-nrf24l01/>

refs: <https://github.com/spirilis/Enrf24>

refs:https://forum.43oh.com/topic/3237-energia-library-nordic-nrf24l01-library/

# Integração Eletrônica e Software

Para fazer a comunicação dos dados produzidos e tratados nos sensores foi escolhido o protocolo MQTT - Message Queuing Telemetry Transport. Como este protocolo fornece baixa latência, baixa complexidade, *playload* leve e é facilmente integrável nos hardwares escolhidos para transmissão de dados pelo time de eletrônica.

O protocolo é amplamente utilizado em projetos de IoT, o que fornece uma bibliografia rica para base de pesquisa e implementação. Com isso, será criado uma arquitetura descentralizada de *publishers* e *subscribers*, de modo que, a central atuará como um coordenador dos dados que saem dos sistemas de eletrônicos, bem como os dados que entrada para atuar no sistema com isso espera=se que a camada de comunicação seja robusta para a implementação do *blockchain* nas camadas acimas.*.*

## Arquitetura proposta usando MQTT

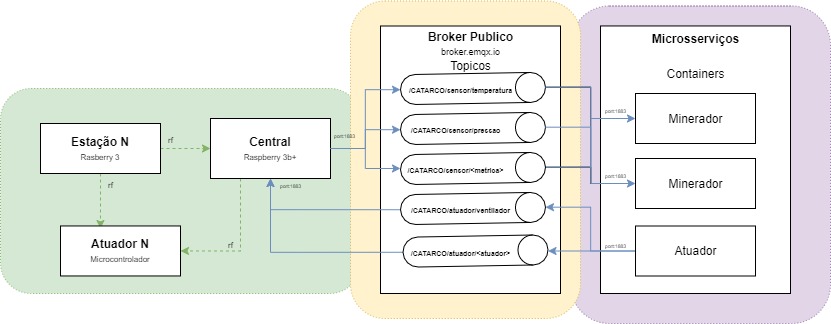
**

Figura X - Diagrama de integração do projeto. O bloco em verde são os recursos utilizados e implementados pela equipe de eletrônica: pode-se observar uma comunicação entre os recursos via radiofrequência utilizando o protocolo *Enhanced ShockBurst Protocol*. No bloco amarelo está o sistema de integração, um middleware utilizando MQTT na arquitetura de *publishers* e *subscribers.* Por fim, no bloco roxo os microsserviços conteinerizados consumindo por serviços minerados que fazem o processo de *blockchain* e os atuadores que enviam comandos ativos para o sistema eletrônico,

*Autoria própria.*

Como mostrado na Figura X, será implementado a arquitetura *publishers* e *subscribers* utilizando o protocolo MQTT. Essa arquitetura funciona de forma descentralizada, independente e escalável. A seguir serão apresentados os principais componentes da arquitetura proposta.

### Central

A central responsável por enviar os dados recebidos das estações. Para o projeto proposto será utilizado uma *Raspberry 3B+* pois possui métricas de conexão e processamento mais condizentes com a natureza da tarefa.

A central é conectada na internet o que facilita a implementação de um cliente de *publish* usando a biblioteca [paho-mqtt](https://pypi.org/project/paho-mqtt/) onde serão implementados *scripts* de escrita e leitura de tópicos do Broker utilizando TCP usando a porta 1833 padrão do MQTT.

Como o sistema atua com filas unidirecionais com garantia de pelo menos uma entrega, a central irá funcionar tanto em modo de *Publisher* publicando os dados dos sensores quanto como *Subscriber* consumindo dados da interface de software para fazer atuações pontuais.

Central atuando como *Publisher*

Os *scripts* irão enviar os dados coletados das estações e atuadores para os tópicos previamente criados no *Broker*, na Figura X é possível visualizar exemplos de tópicos no *Broker* para temperatura e pressão. Será criado tópicos para cada métrica analisada e as mensagens de escrita serão enviados em formato de *json* como visto na Figura Y. Exemplos de payload enviado para o tópico: {“temperatura”: 100 , “estacao\_id”: 1, “sensor\_id”: 1050} e {“pressao”: 100 , “estacao\_id”: 1, “sensor\_id”: 12}.

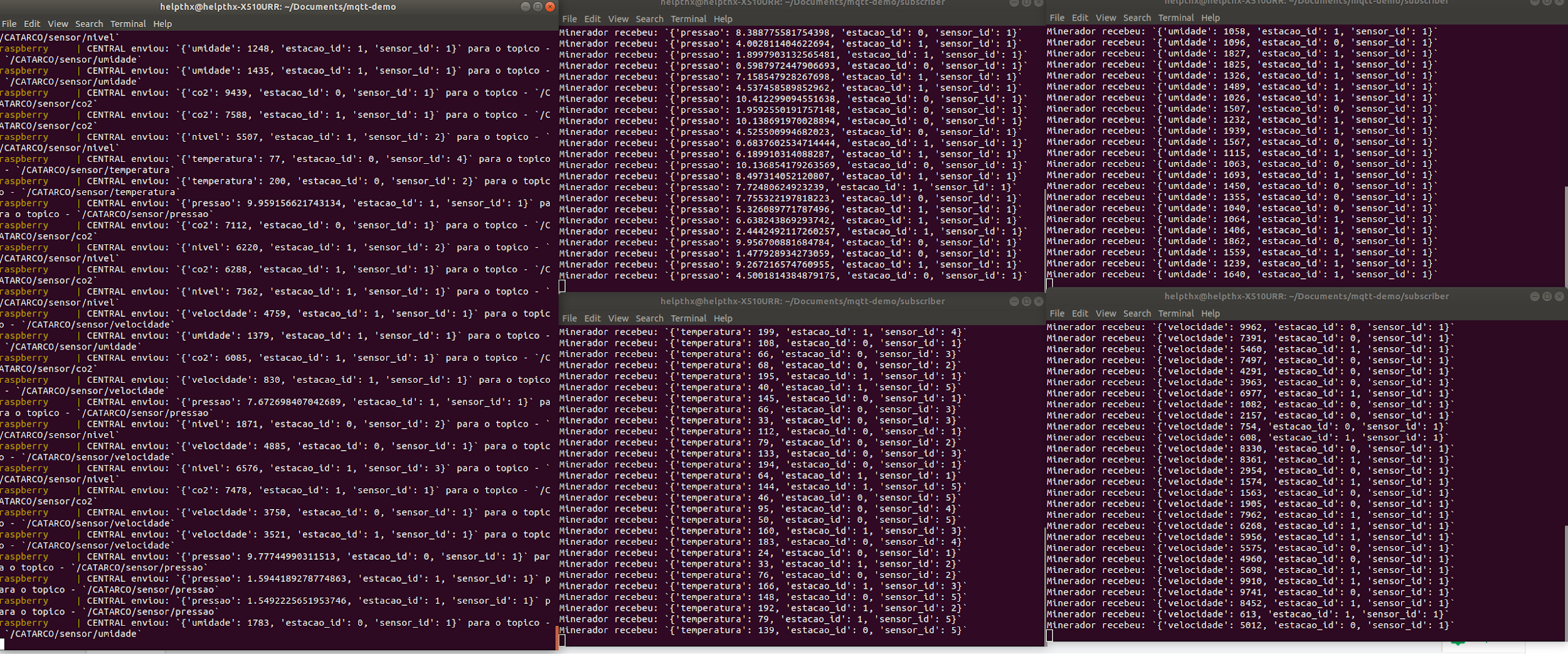


Figura Y - Simulação exemplo para a escrita e consumo do tópicos de temperatura, pressão, umidade e velocidade. No terminal da esquerda é mostrado a distribuição de dados pela central, nos terminais pequenos o consumo pelos moradores.

Como são tópicos diferentes o atributo da métrica muda de *payload* para *payload* dependendo da métrica sensorial analisada. Para não existir problemas de unidades e escalas, a central irá converter todas as métricas recebidas para unidades padrão SI nas melhores escolas oferecidas pelos sensores.

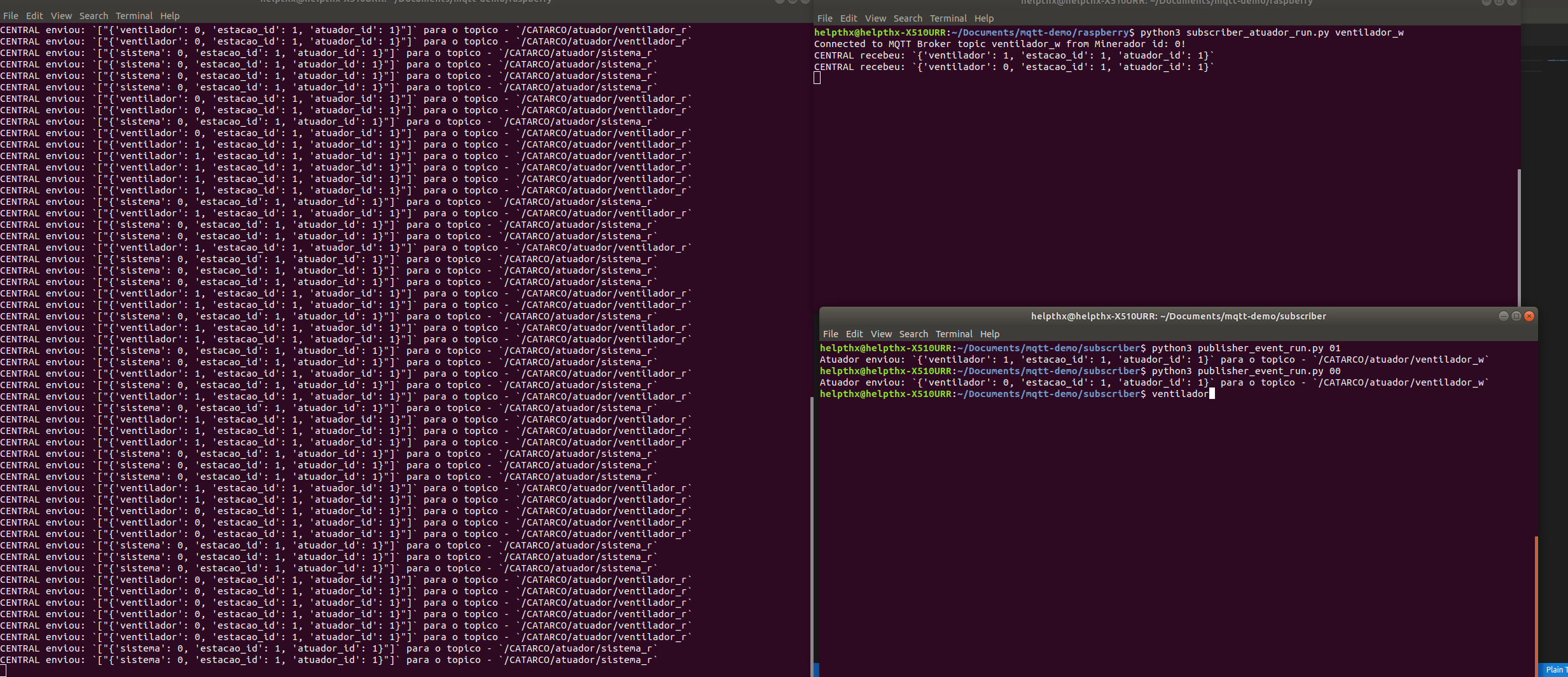
Central atuando como *Subscriber*

Figura P - Simulação do fluxo de dados para o usuário desligar alguns dos atuadores. No terminal da esquerda a central está publicando os status dos atuadores, no terminal superior direito a central está recebendo o comando da interface de software e por fim o terminal inferior direito está publicando a mudança no status do ventilador.

Os *scripts* irão implementar a lógica de Subscriber, que irão ler mensagens dos tópicos de atuadores e transmitir dados dos atuadores. Na Figura X é possível visualizar exemplos dos tópicos de atuação do sistema, onde recebem comandos vindos da interface de software e repassam para a central. Nesse sentido, a Central funciona como um *Subscriber* pois consome comandos dos serviços de atuação da interface de software. Exemplo de mensagem enviada pelo atuador da interface de software { “ventilador”: 0, estacao\_id: 1, “ventilador\_id”: 1}, indica que para o ventilador específico da estação específica deve ser desligado como exemplificado na simulação da Figura P.

Broker

O Broker funciona como central de escrita e consulta, desacoplado da infraestrutura de eletrônica e software, administrado por uma provedora de [brokers públicos](https://www.emqx.io/downloads#edge) que fornece criação de brokers com tópicos limitados por contratação*.* Para criação do protótipo a versão gratuita fornece o número suficiente de tópicos necessários. Contudo, para um ambiente produtivo será necessário a contratação de *tier* mais robusto com garantias melhores.

Para maior segurança e confiabilidade da transmissão de dados o Broker utiliza TLS encriptando os dados na transmissão e utiliza TCP para garantir a ordem e entrega dos dados nos tópicos. Para autenticação dos recursos da comunicação o Broker necessita de usuário e senha para autenticar a conexão dos *scripts* implementados. Com isso espera-se que todas as mensagens que saírem da central sejam analisadas pelos *subscribers.*

Os tópicos funcionam como filas *FIFO* unidirecionais, é garantido a ordem de entrega. Serão implementados tópicos para cada métrica de sensores e para atuadores. Como para os sensores o requisito é apenas mandar os dados para a interface de minedores de software, um tópico para cada métrica será o suficiente para *streamar* os dados. Contudo, para atuadores serão dois tópicos, um de leitura do estado do atuador e outro para a atuar de fato no atuador fazendo um trabalho *event-driven*. A Figura K exemplifica esses dois tipos de modos de trabalho nos tópicos.

*Microservices*

Essa camada da arquitetura corresponde às unidades computacionais que irão ser utilizadas pela equipe de software para a implementação da rede de *blockchain*, interface com o usuário, apresentação das informações do sistema e controle de algumas funcionalidades do protótipo.

Com isso esses microservices, atuaram, também, em dois modos: *publisher* e *subscriber.* Toda comunicação será feita via internet usando TCP na porta 1833.

Mineradores atuando como *Subscriber*

Esses serviços são responsáveis por toda a validação dos dados gerados nos sensores usando o *blockchain*. Para isso necessitam analisar os dados disponibilizados nos tópicos dos Brokers e fazer todo o processo de geração e validação de blocos.

Nesses serviços serão implementados clientes *subscribers* usando *scripts* em *python* usando a biblioteca [paho-mqtt](https://pypi.org/project/paho-mqtt/). Onde cada script irá ser responsável por conectar com tópicos específicos. A partir deste ponto, todos os dados produzidos pelos sensores podem ser consumidos pelos serviços criados pela equipe de software. A Figura K mostra o modo de *streaming* dos dados gerados nos sensores e na Figura X é possível verificar o fluxo de dados até entrar nos containers mineradores.

Atuadores atuando como *Subscriber e Publisher*

Esses *microservices* tem como finalidade transmitir comandos das *interfaces de software* para o protótipo, como exemplo: o usuário deseja desligar os ventiladores de uma estação específica pela interface.Essa comunicação será feita através de tópicos de escrita do atuador.

Devido a natureza unidimensional da fila, cada tipo de atuador do sistema terá um tópico de leitura para reportar o status e um tópico de escrita para mudança de status. A Figura K mostra essa natureza event-driven, onde o tópico de leitura estará sempre reportando o status do atuador, enquanto que o tópico de escrita estará aguardando mensagens de entrada pelos *microservices*. Toda essa comunicação é totalmente assíncrona o que facilita a escalabilidade e a complexidade de implementação.

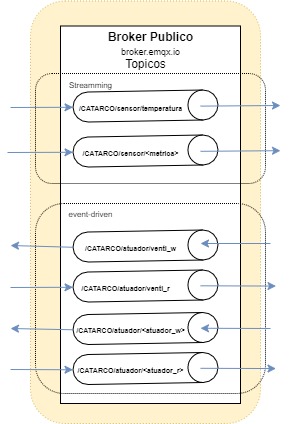


Figura K - Figura dos modos dos tópicos, modo streaming para publicação dos dados da central para a interface de minadores.