Desenvolvimento de dois protótipos de Projetos IoT

Vitor Maia[1]

[1]Universidade Federal do Rio de Janeiro (UFRJ)

Abstract. *The Internet of Things is present in several areas, such as homes, cities, industry and commerce. It permits to integrate and manipulate several devices through sensoring, actuators and communication. This work presents two IoT projects. The first describes the development of an integration between facial recognition and temperature and lighting management in a room. The second describes the development of a smart shelf, which identifies the objects positioned on it.*

Resumo. *A Internet das Coisas está presentes em diversas áreas, como casas, cidades, indústria e comércio. Com ela é possível integrar e manipular diversos dispositivos através de sensores, atuação e comunicação. Este trabalho apresenta dois projetos IoT. O primeiro descreve o desenvolvimento de uma integração entre reconhecimento facial e gerenciamento de temperatura e iluminação em uma sala. O segundo descreve o desenvolvimento de uma estante inteligente, que identifica os objetos posicionados sobre ela.*

1. Introdução

A Internet das Coisas (IoT) é um paradigma que permite compor sistemas de software com objetos de endereço único equipados com comportamentos de identificação, sensibilidade e atuação, além de possuirem processamento capaz de comunicação e cooperação para atingir os objetivos [Motta et al. 2019].

Projetos IoT são desenvolvidos para os mais diversos intuitos, como o monitoramento de plantações [Visconti et al. 2020], verificação do estado do concreto em prédios [Misra et al. 2020], detecção de alagamento em cidades [Hasbullah et al. 2020], monitoramento da temperatura e da respiração de pacientes [Shahanim et al. 2020], entre outros. Os projetos citados possuem em comum o uso de sensores, dos quais são extraídas informações que caracterizam a entidade estudada. A análise destes dados pode ser utilizada para deduzir informações, tomar decisões ou apenas disparar atividades.

Descrevemos a implementação de dois projetos IoT. O primeiro projeto monitora a presença de pessoas em uma sala e gerencia as preferências de temperatura e iluminação destas pessoas, transformando a sala em um **espaço inteligente**: área equipada com sensores que permite entender o que está acontecendo nela [Lee and Hashimoto 2002]. O segundo projeto apresenta um protótipo de estante inteligente que identifica quais objetos estão sobre ela. Apesar dos objetivos distintos, ambos os projetos são desenvolvidos de forma semelhante: através do uso de Arduinos conectados a módulos e sensores, comunicação por Bluetooth e tratamento das informações em um Raspberry Pi.

Este artigo apresenta mais três seções além da introdução. A seção 2 apresenta trabalhos relacionados. A seção 3 apresenta as tecnologias utilizadas, e descreve design e implementação de ambos os projetos, com seus respectivos trabalhos futuros. Por fim, a seção 4 apresenta a conclusão.

2. Trabalhos Relacionados

Para o desenvolvimento do projeto foram utilizadas as tecnologias descritas na subseção 3.1. Os estudos a seguir constroem diversas soluções IoT com os mesmos componentes e protocolos utilizados neste trabalho.

O desenvolvimento de uma estrutura de gerenciamento de fazendas é descrita por [Visconti et al. 2020]. A solução é utilizada para calibrar irrigações e fertirrigações através de ESP8266 (Wi-Fi) alimentado por energia solar coletada por um robô. A solução também utiliza módulos BLE HM-10 para gerenciar contêineres de produtos colhidos, além de uso de leitores e tags RFID para coletar e comunicar para smartphones informações sobre o status dos alimentos.

Em [Kinchin 2018] é apresentado um projeto que ensina como criar um termômetro através de leitura de um termistor com Arduino. Em [Kumar et al. 2017] é apresentado um projeto, também controlado por Arduino, que utiliza um fotoresistor cuja leitura de sinal ativa um buzzer.

Um sistema de aluguel e cadastro de livros em uma biblioteca é apresentado por [Hussein and Hassan 2017], desenvolvido com auxílio de um Arduino ligado a um módulo leitor RFID RC522, utilizado para ler tags que identificam livros. Esta estrutura é apoiada por um sistema de cadastro e por um banco de dados. Um leitor RC522 é utilizado similarmente por [Orji et al. 2018] para desenvolver uma maçaneta com controle de acesso. UID de tags são comparadas a valores registrados em um sistema de acesso.

3. Design do Projeto

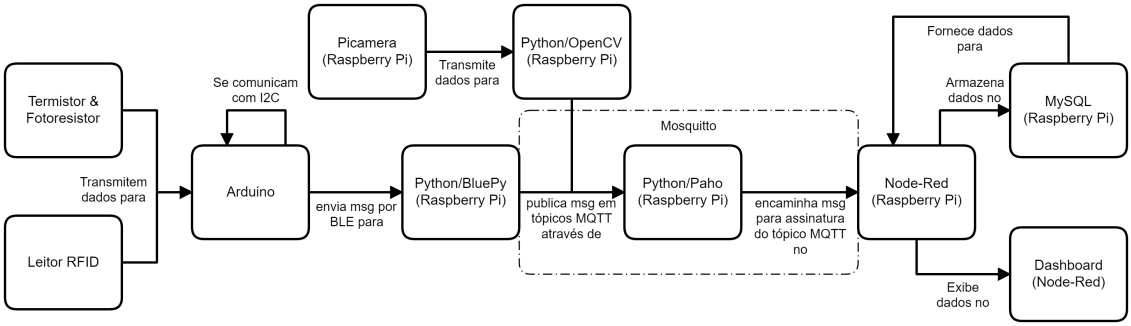
Esta seção apresenta as soluções desenvolvidas. A subseção 3.1 apresenta os componentes eletrônicos, protocolos e softwares utilizados. A subseção 3.2 descreve o projeto de preferências de iluminação e temperatura. A subseção 3.3 descreve o projeto de estante inteligente. O relacionamento entre todas as tecnologias apresentadas está representado de forma simplificada na figura 1.

3.1. Tecnologias Utilizadas

Para a construção das soluções, foram utilizados os seguintes componentes eletrônicos:

- **Arduino**: Através de um microcontrolador, permite manipulação de dados digitais (0 ou 1) e analógicos (0 até 1023). Foram utilizados para manipulação de dados coletados de sensores e de módulos.
- **Raspberry Pi**: Um computador com sistema Raspbian contendo Wi-Fi, Bluetooth, câmera, entradas USB e HDMI, além de pinos digitais que permitem uso semelhante ao Arduino. Foi utilizado para instalação de banco de dados, broker MQTT, reconhecimento facial e coleta de dados enviados por Bluetooth.
- **Resistor**: Utilizado para limitar a corrente elétrica em circuitos. No contexto deste trabalho os resistores comuns serão abstraídos das explicações.
- **Termistor e Fotoresistor**: Termistores são resistores capazes de variar a resistência de acordo com a mudança de temperatura do ambiente. Analogamente, fotoresistores variam a resistência de acordo com a intensidade de iluminação que recebem. Foram utilizados como sensores de temperatura e iluminação.
- **BLE HM-10**: Módulo que permite comunicação de dados através de Bluetooth Low Energy. Foi utilizado para comunicação entre Arduinos e Raspberry Pi.

Figura 1. Relacionamento entre as tecnologias, em ambos os projetos.



- **RFID RC522**: Módulo leitor de tags RFID. Permite ler ou escrever dados nas tags. Pode-se escrever até 96 bits de dados, porém apenas os bits de Serial Number foram utilizados neste trabalho.

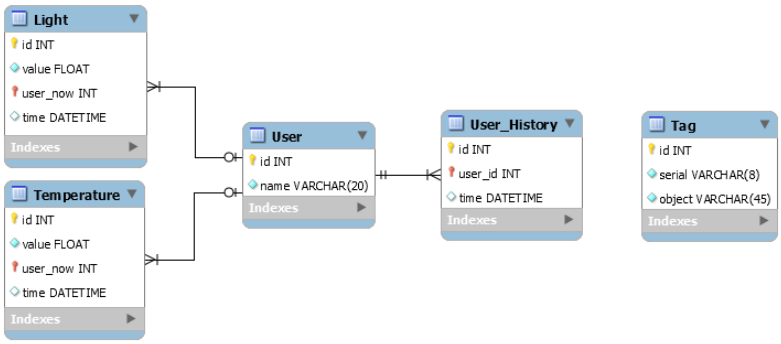
Foram utilizados os seguintes protocolos:

- **I2C**: Protocolo de camada física, serial, multi-mestre e multi-escravo, capaz de conectar microcontroladores por fios. Para a comunicação de dados ocorrer os clocks precisam estar em sincronia, por isso são conectadas as entradas SCL (clock) e SDA (dados). Foi utilizado para permitir comunicação entre Arduinos.
- **Bluetooth Low Energy**: Protocolo WPAN capaz de conectar objetos distantes em até 50 metros, em topologia Piconet, na qual um mestre pode se conectar a até sete escravos. Foi escolhido como protocolo de camada de enlace devido à má qualidade do Wi-Fi disponível na sala, e pela presença de BLE no Raspberry Pi. Módulos HM-10 são escravos do Raspberry Pi nas soluções apresentadas.
- **MQTT**: Protocolo de camada de aplicação para transporte de mensagens, do tipo publish/subscribe, no qual um dispositivo publica mensagens em um tópico de um broker MQTT, e outros dispositivos podem assinar este tópico para receber estas mensagens. O MQTT executa com TCP na camada de transporte, e o QoS em todos os casos foi 2, para garantir a entrega das mensagens apenas uma vez.

Foram utilizados os seguintes softwares:

- **MySQL**: Instalado no Raspberry Pi para armazenar dados históricos de sensores e para apoiar a identificação de objetos lidos por tags RFID. A figura 2 apresenta o modelo ER do banco de dados criado, que será explicado nas próximas seções.
- **Node-Red**: Ferramenta para desenvolvimento baseado em fluxos instalada no Raspberry Pi. Foi utilizado para assinar tópicos MQTT, executar queries no MySQL e exibir informações relevantes ao usuário em um dashboard.
- **Mosquitto**: Ferramenta usada como broker MQTT. Instalado no Raspberry Pi.
- **Thonny**: IDE Python pré-instalada no Raspbian. Programas em python tiveram como finalidade: (i) receber mensagens por Bluetooth e encaminhar com MQTT usando as bibliotecas BluePy e Paho, e (ii) desenvolver o reconhecimento facial através da biblioteca OpenCV.
- **Arduino IDE**: Ferramenta de desenvolvimento para Arduino, com C++. Foram utilizadas as bibliotecas Wire (I2C) e MFRC522 (RFID).

Figura 2. Modelo ER de ambos os projetos.



3.2. Projeto 1: Preferências de Temperatura e Iluminação

Este projeto tem como objetivo identificar usuários, gerenciar as preferências de temperatura e iluminação para estes usuários no ambiente, e ligar lâmpadas e ar condicionado automaticamente a partir de decisões sobre os dados coletados. Para isso, foram realizados dois desenvolvimentos complementares: (i) reconhecimento facial e (ii) coleta de dados de sensores para a identificação de preferências.

3.2.1. Reconhecimento Facial

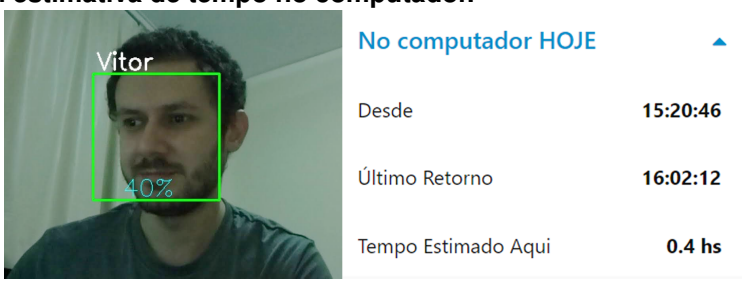
O Raspberry Pi possui módulo de câmera, chamada de PiCamera. A imagem capturada é de boa qualidade e pode ser utilizada em conjunto com ferramentas de machine learning para interpretação do que está sendo capturado. A biblioteca **OpenCV** possui implementação de algoritmos de visão computacional e foi utilizada para o reconhecimento facial. Foram realizadas três etapas: **Treinamento**, **Reconhecimento** e **Registro**.

Treinamento. Dois usuários utilizam o ambiente em momentos alternados. Para que o algoritmo saiba quais rostos reconhecer, um total de 50 fotos foram tiradas de cada usuário para treinar o reconhecimento. Foi utilizada a função *train()* do objeto *LBPHFaceRecognizer*, para aprender com as fotos. Através desta função é gerado um arquivo *.yml*, com conhecimento adquirido sobre os rostos. O código de treinamento é executado pontualmente, sempre que se deseja atualizar o conjunto de fotos.

Reconhecimento. Um segundo código python é executado continuamente para a identificação de rostos com a PiCamera. A captura é feita através da função *cv2.VideoCapture()*. Um objeto *LBPHFaceRecognizer* é novamente utilizado, desta vez sua função *read()* para ler o *.yml*, e o reconhecimento a cada quadro do video é feito com o algoritmo *CascadeClassifier*. O código identifica rostos, calcula a probabilidade de ser de um dos usuários conhecidos, e assim supõe quem está na imagem. Se o rosto encontrado for atribuído a algum usuário conhecido, seu nome é publicado no tópico MQTT *room/user*, a cada 30 segundos. A figura 3 mostra um exemplo de reconhecimento facial.

Registro. Há uma execução constante do Node-Red no Raspberry Pi, com um fluxo iniciado com uma assinatura ao tópico MQTT *room/user*. A presença do usuário é registrada frequentemente na tabela MySQL *User_History* (Figura 2). Assim é possível estimar o tempo que certo usuário ficou na frente do computador, o ponto da sala para onde a câmera aponta. Esta informação é exibida no dashboard do Node-Red (Figura 3).

Figura 3. (i) Exemplo de captura de reconhecimento facial com OpenCV. (ii) Dashboard com estimativa de tempo no computador.



3.2.2. Preferências de Temperatura e Iluminação

Dois Arduinos Uno **igualmente configurados** com termistor e fotoresistor foram posicionados em pontos distantes do ambiente. Cada Arduino também está ligado a um módulo BLE HM-10, para se comunicar com o Raspberry Pi, como apresentado na figura 4.

Quanto à **abstração dos dados**, no nível *Signal* os sensores transmitem voltagem convertida pelas entradas analógicas dos Arduinos em valores entre 0 e 1023. No nível *Pixel*, dados de termistores são convertidos em ºC e de fotoresistores são calibrados de acordo com o contexto de iluminação da sala, mapeado para valore entre 0% e 100%. No nível *Feature* os dados são interpretados como frio, quente, escuro ou claro a partir da comparação entre as leituras recentes e as preferências do usuário. No nível *Symbol*, a interpretação é utilizada para ligar lâmpadas e ar-condicionado em caso de necessidade.

Em ambos os Arduinos, leituras dos sensores são armazenadas a cada 50ms, 20 vezes, e então é calculada a **mediana**. A média não foi utilizada pois poderia não condizer com a realidade caso houvesse alteração brusca de valor, em especial no caso da coleta de iluminação. As medianas são enviadas por BLE e recebidas por um programa python no Raspberry Pi, que verifica eventuais quedas de pareamento, verifica a validade dos dados, e encaminha para o respectivo tópico MQTT. Há um total de quatro tópicos MQTT, um para cada medição (temperatura e iluminação) recebida de cada Arduino.

Um fluxo no Node-Red se inicia com assinatura aos tópicos. Para ambas as medições, nós Aggregator calculam a **média** de dados recebidos ao longo de 30s. Espera-se que a mediana calculada nos Arduinos apoiada pela média calculada no Aggregator sejam suficientes para retornar valores precisos. Os valores são registrados no MySQL **em nome do usuário presente naquele momento**, caso algum exista.

No BD (Figura 2), usuários são pré-registrados na tabela *User*, e os valores do ambiente são salvos a cada 30s nas tabelas *Light* e *Temperature*, identificando o usuário através da chave estrangeira *user_now*. Um dashboard foi criado no Node-Red para exibir: (i) os valores mais recentes, (ii) um gráfico com histórico das últimas duas horas, e (iii) estimativas de preferências do usuário atual a partir de interpretação dos dados (Figura 5).

As medições são sempre registradas para o usuário atualmente presente. Porém, por exemplo, uma temperatura de 30ºC pode ser registrada para um usuário que odeia calor, momentos antes de este usuário decidir ligar o ar condicionado. Este valor, apesar de registrado, não pode ser utilizado nos cálculos de preferência deste usuário. Para tal, stored procedures no MySQL filtram **outlier** antes da exibição de preferências.

Figura 4. Protótipo com Termistor, Fotoresistor e módulo Bluetooth.

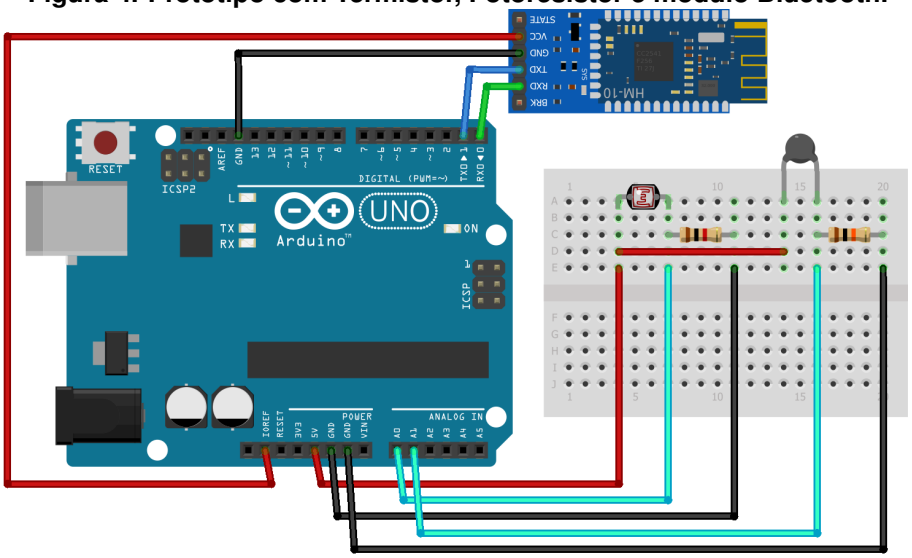
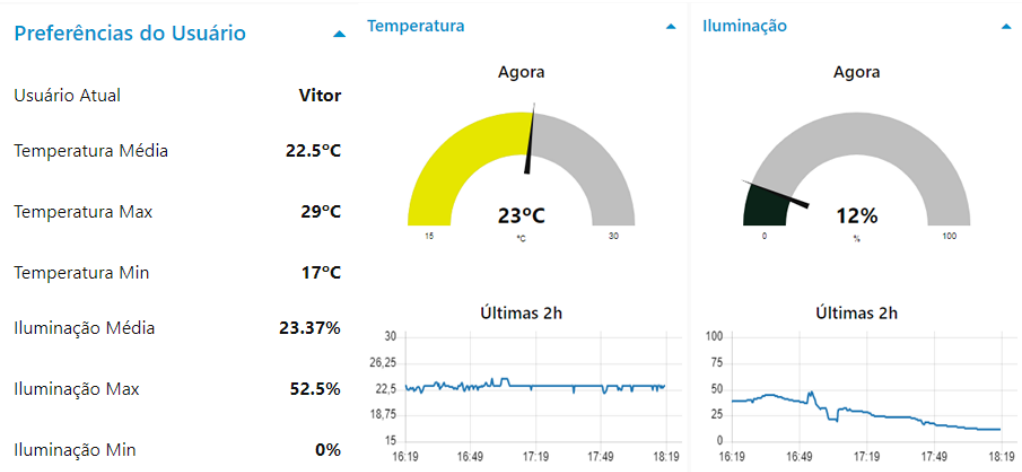


Figura 5. Dashboard com informações do ambiente e preferências.

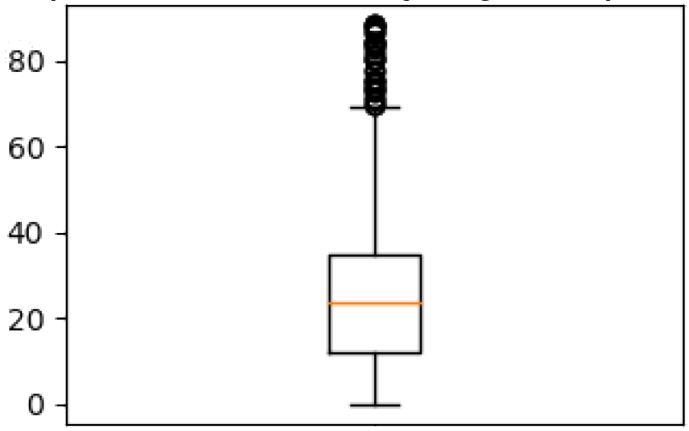


Duas stored procedures executam queries que buscam todos os dados registrados para um usuário. **Média** e **desvio padrão** são utilizados sobre este conjunto para identificar **outliers** e ignorá-los. Por fim, são retornados os valores mínimo, máximo e médio sem os **outliers** (figura 5). A figura 6 apresenta um boxplot gerado com todos os valores de iluminação para um certo usuário. Quanto maior a quantidade de dados coletada ao longo do tempo, melhores serão estas estimativas.

3.2.3. Trabalhos Futuros

Lâmpadas poderiam ser acesas automaticamente caso o valor atual de iluminação esteja abaixo do intervalo de preferência do usuário. Analogamente, ar condicionado poderia ser ligado caso a temperatura atual esteja acima do intervalo. A ativação destes dispositivos pode ser gerenciada com Arduinos, que utilizariam as informações dos valores de preferência, recebidos por BLE, para ativar tanto lâmpadas quanto ar condicionado liga-

Figura 6. Boxplot com valores de iluminação registrados para um usuário.



dos a módulos relé. Por inexperiência do autor com relés, esta etapa do projeto não foi desenvolvida, e será um trabalho futuro.

Este projeto foi desenvolvido no contexto de uma única sala, e o reconhecimento facial buscou usuários apenas em uma área limitada, com foco no uso de um computador. O projeto pode ser expandido para o contexto de uma casa, com tratamentos de dados por cômodos. Mais câmeras podem ser posicionadas de modo que toda a casa seja coberta. Este projeto ajudaria a manter o ambiente agradável para todos os moradores e poderia gerar economia de energia ao manter dispositivos ativados apenas quando necessário.

3.3. Projeto 2: Estante Inteligente

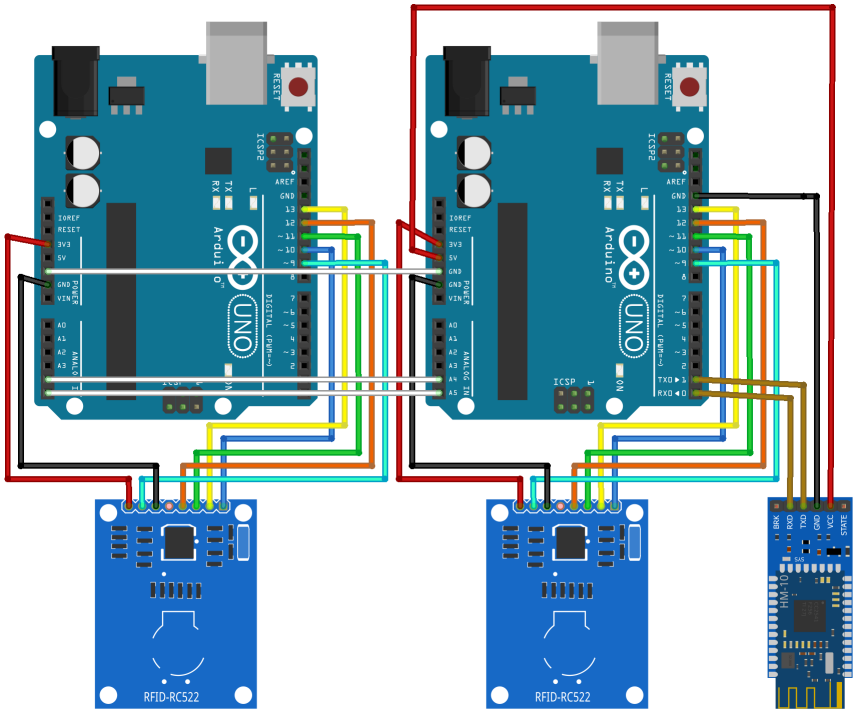
Foi desenvolvido o protótipo de uma estante capaz de identificar qual objeto está sobre ela. Em alguns pontos da estante são dispostos leitores RFID (módulos RC522), e os objetos a serem identificados devem possuir uma tag RFID acoplada à base. Quando objetos são posicionados nos locais específicos da estante onde há algum leitor RFID, a informação de que aquele objeto está lá é armazenada. Esta informação é comunicada por bluetooth e exibida em um dashboard.

3.3.1. Arquitetura

A figura 7 apresenta o protótipo. Arduinos comunicam-se entre si através dos fios brancos, com o protocolo I2C. Apenas um dos Arduinos possui ligação ao módulo BLE HM-10, e ambos possuem seu próprio leitor RFID RC522.

O protocolo I2C permite que diversos Arduinos sejam configurados como mestres ou escravos. O mestre especifica um escravo através da função *beginTransmission(id_slave)* e transmite bytes através da função *write(data)*. O escravo utiliza *onReceive(func_name)* para registrar uma função a ser chamada sempre que dados chegarem do mestre. O Arduino com módulo BLE, um escravo, envia **mensagens** por BLE logo que identifica um objeto, enquanto o mestre precisa repassar a mensagem para o escravo, para que este envie. A decisão de usar I2C em conjunto com apenas um módulo HM-10 é motivada pelo desejo de minimizar a quantidade de componentes eletrônicos da solução, tornando-a mais barata.

Figura 7. Protótipo da estante inteligente.



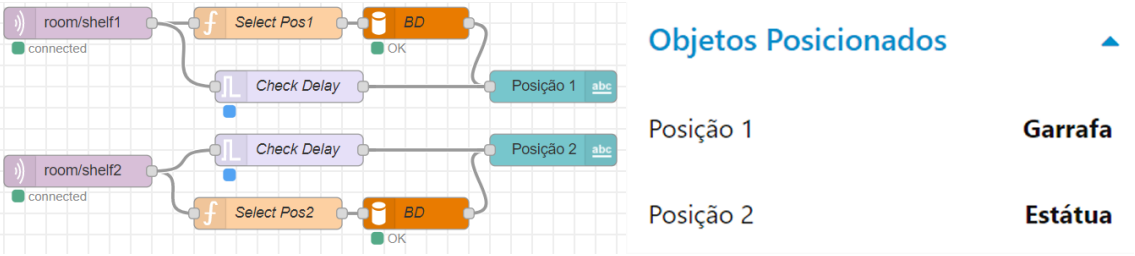
As mensagens são definidas no formato **Posição | Serial**, sendo **Posição** um número de 2 dígitos que indica onde está o leitor na estante, e **Serial** o número de série que identifica a tag RFID lida, sempre hexadecimal com 8 dígitos. Por exemplo, a mensagem **02 | E4589A2A** indica que o objeto cuja tag é identificada por E4589A2A está na posição 2 da estante. Ausência de objetos é comunicada através de **Posição | NONE**.

Três situações geram comunicação por BLE: (i) no momento exato em que um objeto é posicionado no leitor, (ii) a cada três segundos, para informar presença contínua, e (iii) apenas uma vez, após seis segundos de ausência, envia NONE. Caso o HM-10 perca o pareamento no momento do posicionamento de um objeto, a situação (ii) auxilia no envio, mesmo que com atraso. A mensagem de ausência de objetos é enviada apenas uma vez, e se perdida, é possível deduzir esta informação através do tempo sem mensagens.

Em uma primeira versão da implementação, o nome do objeto (ex: Cartão) era salvo na tag como Object Class, e este nome era enviado em conjunto com Posição e Serial. Porém, a comunicação I2C não se comportou consistentemente com mensagens maiores do que 15 dígitos, nem com caracteres especiais. Por isso a modelagem foi adaptada para identificar os nomes com auxílio do BD. A tabela *Tag* (figura 2) vincula um nome de objeto a um serial. Assim, através de um comando **INSERT** com **ON DUPLICATE KEY UPDATE** é possível atualizar apenas o nome do objeto, quando necessário.

Foi utilizada a biblioteca python BluePy para parear o Raspberry Pi com o módulo HM-10. Os Seriais das mensagens são encaminhados para os tópicos MQTT *room/shelf1* e *room/shelf2* dependendo da posição indicada na mensagem. Um fluxo no Node-Red os assina, e sempre que uma mensagem é recebida, verifica no BD qual o objeto correspondente. O nome do objeto é então exibido em um dashboard. A figura 8 apresenta o fluxo no Node-Red e o dashboard.

Figura 8. (i) Fluxo no Node-Red com lógica da estante. (ii) Dashboard com nome dos objetos atualmente em cada posição da estante.



3.3.2. Trabalhos Futuros

O protótipo foi feito com apenas dois leitores RFID para validação da ideia, mas é tecnicamente viável utilizar uma quantidade maior de Arduinos e leitores, mantendo apenas um Arduino com módulo HM-10 e os demais em comunicação através de I2C.

Uma quantidade maior de leitores RFID poderia ser posicionada matricialmente, e compor uma superfície inteligente. Por exemplo, 9 leitores poderiam ser posicionados 3x3. A posição de um objeto passaria a ter um significado, devidamente programado em fluxos no Node-Red, e a posição indicada na mensagem passaria a ser no formato (X,Y). Seria possível, por exemplo, desenvolver um jogo da velha.

Semelhantemente, no contexto de uma casa inteligente pode-se usar o posicionamento de objetos em uma estante como meio lúdico de enviar comandos ao ambiente. Por exemplo, certo objeto na estante simbolicamente poderia ligar a luz do quarto.

Diversas estantes poderiam ser utilizadas como prateleiras de um mercado. A identificação de presença/ausência de produtos facilitaria reposição de estoque, além de ensinar ao mercado sobre os hábitos de compra dos clientes. Um conjunto de dados históricos sobre momentos de retirada de produtos permitiria a execução de algoritmos de machine learning e data warehousing para otimização do posicionamento de produtos nas estantes. A solução descrita em [Mekruksavanich 2020] realiza controle semelhante através da leitura de objetos presentes no carrinho.

Como trabalho futuro, mais leitores RFID serão utilizados para verificar os limites da comunicação I2C neste cenário. Serão exploradas as configurações em fila e em matriz. Além disso, os Arduinos Uno serão substituídos por Arduinos Nano, por serem mais baratos e ocuparem menos espaço.

4. Conclusão

Este trabalho apresentou o desenvolvimento de dois projetos IoT que compartilham um mesmo Raspberry Pi, utilizado para comunicação através de Bluetooth Low Energy, gerenciamento através de Node-RED e armazenamento através de MySQL. O primeiro projeto descreve uma solução que identifica usuários em uma sala e deduz suas preferências de temperatura e iluminação através de dados de sensores. O segundo projeto apresenta um protótipo de estante inteligente, feita com leitores RFID conectados a Arduinos. O desenvolvimento destes projetos apontou pontos de melhoria e permitiu a identificação de diversas possibilidades de trabalhos futuros.

Referências

Hasbullah, A., Rahimi, A., Amrimunawar, A., Redzwan, F., Mahzan, N., Omar, S., and Zin, N. (2020). Flood and notification monitoring system using ultrasonic sensor integrated with iot and blynk applications: Designed for vehicle parking. *Journal of Physics: Conference Series*, 1529:022050.

Hussein, R. and Hassan, A. (2017). Simulation of radio frequency identification based library management system. *Journal of Engineering and Sustainable Development*, 21:161–170.

Kinchin, J. (2018). Using an arduino and cheap thermistor to make a simple temperature sensor. *Physics Education*, 53:063008.

Kumar, D., Singh, L., Virdi, G., Singh, G., and Channi, H. K. (2017). Designing and modeling of arduino based light sensor. *IJSART*.

Lee, J. H. and Hashimoto, H. (2002). Intelligent space - concept and contents. *Advanced Robotics*, 16(3):265–280.

Mekruksavanich, S. (2020). Supermarket shopping system using rfid as the iot application. *2020 Joint International Conference on Digital Arts, Media and Technology with ECTI Northern Section Conference on Electrical, Electronics, Computer and Telecommunications Engineering, ECTI DAMT and NCON 2020*, pages 83–86.

Misra, D., Das, G., and Das, D. (2020). An iot based building health monitoring system supported by cloud. *Journal of Reliable Intelligent Environments*, 6.

Motta, R., Silva, V., and Travassos, G. (2019). Towards a more in-depth understanding of the iot paradigm and its challenges. *Journal of Software Engineering Research and Development*, 7:3.

Orji, E., Cv, O., and Nduanya, U. (2018). Automatic access control system using arduino and rfid. *The Journal of Scientific and Engineering Research*, 05:333–340.

Shahanim, N., Amirnazarullah, M., Jafri, M., and Abdullah, S. (2020). Iot based patient monitoring system using sensors to detect, analyse and monitor two primary vital signs. *Journal of Physics: Conference Series*, 1535:012004.

Visconti, P., De fazio, R., Velázquez, R., Del Valle Soto, C., and Giannoccaro, N. I. (2020). Development of sensors-based agri-food traceability system remotely managed by a software platform for optimized farm management. *Sensors*, 20:1–43.