Houve algumas mudanças de Arquitetura entre as Sprints 1 e Sprint 2 que serão contempladas neste documento.

**Arquitetura**

A arquitetura de ambiente de desenvolvimento foi feita para a prototipação e experimentação prática das soluções de IoT que foram pensadas para o projeto. Neste cenário a estrutura necessária é equivalente a um projeto de automação residencial simples.

O microcontrolador escolhido foi o ESP-WROOM-32 da Espressif. Ele possui um bom número de pinos, baixo custo, recursos robustos de hardware em comparação a microcontroladores concorrentes e baixo consumo de energia com Wi-fi e Bluetooth integrados, um dos fatores chave para a viabilização do projeto. Além disso, pode ser programado pela IDE Arduino, usufruindo da colaboração da grande comunidade Arduino através de suas bibliotecas e da popularidade da linguagem C++, com vários projetos disponíveis como modelo.

Conectado ao microcontrolador há um circuito com anel de NeoPixel RGB 24 em circuito. O controle do NeoPixel é realizado pelo microcontrolador e para que haja comunicação deste com a internet, escolhemos o protocolo MQTT. O baixo consumo energético, menor consumo de dados, facilidade de utilização e escalabilidade fazem desse protocolo um dos mais utilizados para IoT e uma excelente opção para nosso projeto.

O MQTT é um protocolo basicamente composto por Clientes ligados a um broker, uma espécie de servidor que administra as mensagens de outros Clientes através de grupos de inscrição. Os Clientes que se inscrevem para receber mensagens desses grupos (Subscribers) recebem as mensagens dos Clientes que estão publicando (Publishers) nesses grupos. Um Cliente pode fazer as duas coisas: ser Publisher e Subscriber, do mesmo grupo ou de grupos diferentes, bastando ter o grupo inscrito como referência para enviar e/ou receber mensagens. Para implantar esse protocolo na placa microcontroladora, utilizamos a biblioteca PubSubClient para conectá-la a um broker.

O broker escolhido foi o HIVEMQ, um broker público disponível gratuitamente. Para fazer um protótipo da solução de iluminação ambiente proposta no projeto, o HIVEMQ foi o suficiente para testar a aplicabilidade da arquitetura. Para acessar o broker em seu endereço de servidor, é necessário conectar a placa microcontroladora à internet, o que foi feito com a biblioteca Wifi.h.

Uma vez configurada a placa, optamos por trabalhar com o Node-RED para tratar dados recebidos em determinados formatos e convertê-los em formatos legíveis por todas as partes que compõem nosso sistema (APIs, servidores, placas microcontroladoras, etc.). O Node-RED está hospedado em um servidor Bluemix IBM Apps do Cloud Foundry. Configuramos um flow com mqtt in e mqtt out para o HIVEMQ com as devidas configurações de inscrição de grupos para receber e transmitir os dados entre as partes. Também associados ao Node-RED os nós de Spotify e do Telegram. Do Spotify receberemos informações em JSON que serão associadas ao frontend para controle da aplicação pelo usuário do nosso app, no que diz respeito à ambientação musical. Do Telegram, receberemos informações em JSON que permitirão o controle da ambientação da iluminação por mensagens enviadas pelo usuário para o bot do aplicativo cadastrado para tal finalidade. Também incluímos um nó Watson Speech-to-Text para que as mensagens enviadas por áudio também pudessem ser interpretadas e convertidas em texto, seguindo então, o mesmo fluxo do Telegram para mensagens escritas. A placa microcontroladora envia como resposta o formato JSON que pode ser interpretado pelo Telegram, enviando uma mensagem de confirmação ou de erro ao usuário.

Para comunicação com o backend Java, uma classe foi criada com o projeto Eclipse Paho para gerar o cliente MQTT nessa linguagem. Para o frontend em React-Native, faremos a comunicação via JSON com o NODE-RED para controlar os dispositivos IoT remotamente.

É importante destacar a arquitetura aqui demonstrada não é considerada por este grupo uma solução para ambiente de produção dada a insegurança de um ambiente de broker público como o HIVEMQ e a potencial necessidade de escalabilidade, como a utilização de um servidor Bluemix lite da IBM que ficam limitadas nesse contexto. A arquitetura aqui demonstrada está, portanto, ilustrando nosso ambiente de desenvolvimento. O ambiente de produção está sendo devidamente arquitetado na medida em que estamos aprendendo novos conhecimentos nas matérias de DevOps Tools and Cloud Computing e Compliance and Quality Assurance, levando em consideração infraestrutura, estimativa de orçamento, carga de dados, plano de negócio, boas práticas etc.