INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA - CÂMPUS FLORIANÓPOLIS
DEPARTAMENTO ACADÊMICO DE MECATRÔNICA
PROGRAMA DE PÓS-GRADUAÇÃO EM MECATRÔNICA

FÁBIO VICTOR SCHREIBER

SISTEMA DE MEDIÇÃO SEM FIO DE CONSUMO DE ÁGUA DO EMPREENDIMENTO SAPIENS PARQUE UTILIZANDO PROTOCOLO MQTT

FLORIANÓPOLIS, JUNHO DE 2018.

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA - CÂMPUS FLORIANÓPOLIS
DEPARTAMENTO ACADÊMICO DE MECATRÔNICA
PROGRAMA DE PÓS-GRADUAÇÃO EM MECATRÔNICA

FÁBIO VICTOR SCHREIBER

SISTEMA DE MEDIÇÃO SEM FIO DE CONSUMO DE ÁGUA DO EMPREENDIMENTO SAPIENS PARQUE UTILIZANDO PROTOCOLO MQTT

Dissertação submetida ao Programa de Pós-Graduação em Mecatrônica do Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina, para obtenção do Grau de Mestre em Mecatrônica.

Orientador: Prof. Dr. Roberto Alexandre Dias

FLORIANÓPOLIS, JUNHO DE 2018.

Schreiber, Fábio Victor

Sistema de medição sem fio de consumo de água do empreendimento Sapiens Parque utilizando protocolo MQTT / Fábio Victor Schreiber; orientação de Roberto Alexandre Dias. – Florianópolis, SC, 2018.

150 p.

Dissertação (Programa de pós-graduação em Mecatrônica) – Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina.

Inclui referências.

1. MQTT. 2. Water Metering System. 3. Protocolos de Comunicação. 4. IoT. I. Dias, Roberto Alexandre. II. Instituto Federal de Santa Catarina. Departamento Acadêmico de Metal Mecânica. III. Título.

SISTEMA DE MEDIÇÃO SEM FIO DE CONSUMO DE ÁGUA DO EMPREENDIMENTO SAPIENS PARQUE UTILIZANDO PROTOCOLO MQTT

FÁBIO VICTOR SCHREIBER

Este trabalho foi julgado adequado para obtenção do Título de Mestre em Mecatrônica e aprovado na sua forma final pela banca examinadora do Curso de Pós Graduação em Mecatrônica do Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina.

Florianópolis, 08 de junho de 2018.

Banca Examinadora:

Roberto Alexandre Dias, Dr.

Mário de Noronha Neto, Dr.

Maurício Edgar Stivanello, Dr.

Valdir Noll, Dr.

Dedico este trabalho aos meus pais que sempre fizeram tudo que podiam para que seus filhos chegassem aonde chegaram e dedicação especial a minha esposa, parceira e companheira sempre presente com seu sorriso transbordando amor e alegria pelo mundo.

AGRADECIMENTOS

Ao Prof. Dr. Roberto Alexandre Dias, primeiro pela confiança e segundo pela liberdade de trabalho e oportunidades de aprendizados.

Ao amigo Everson Osvanir da Silva, grande profissional que contribuiu muito para o desenvolvimento dos códigos utilizados nesta dissertação.

À amiga Victória Zanetti, bolsista do NERsD com grande futuro, que colaborou muito com o desenvolvimento dos códigos em micro Python para os microcontroladores ESP8266.

Ao amigo George Tavares pela ajuda com o desenvolvimento do aplicativo de visualização dos dados coletados na WEB.

À Fundação CERTI e ao Sapiens Parque S.A. por oferecer condições e a oportunidade para a realização da pesquisa.

Agradeço às pessoas da minha família que entenderam as minhas ausências e um agradecimento especial ao meu pai Arnoldo Schreiber e a minha mãe Elena Maria de Mattos que sempre me incentivaram nos estudos e fizeram de tudo para me dar condições de chegar até aonde cheguei.

E por fim o agradecimento mais que especial à minha esposa Dayana Schreiber, amiga e companheira, que soube mais que ninguém entender as necessárias ausências durante o desenvolvimento deste trabalho.

“Se você quer construir um navio, não chame as pessoas para juntar madeira ou atribua-lhes tarefas e trabalho, mas sim os ensine a desejar a infinita imensidão do oceano.”

Antoine de Saint-Exupéry

RESUMO

Esta pesquisa teve como objetivo desenvolver um sistema de comunicação sem fio para medição do consumo de água no empreendimento Sapiens Parque, empregando protocolo MQTT. Para tanto foram realizadas pesquisas acerca do tema que envolvia medição remota de serviços públicos tais como, água, energia e gás e sobre os conceitos e utilização de tecnologias da Internet das Coisas. Foi realizado ainda um estudo detalhado sobre o protocolo MQTT demonstrando o funcionamento do modelo *publisher/subscriber*, formas de publicações de mensagens e os níveis de qualidade de serviços (QoS), identificado recomendações importantes acerca da escolha e definição dos tópicos, da utilização de caracteres coringas e a forma de interação com os *brokers* MQTT. Além disso, no decorrer do trabalho apresenta-se o módulo de desenvolvimento CC1200DK da *Texas Instruments*, hardware compatível com as normas europeias que utiliza e opera com o protocolo WM-Bus. Mostra-se também que este módulo de desenvolvimento pode operar no território brasileiro, pois atende as normas regulamentadoras da ANATEL, agência que regula a utilização das faixas de rádio frequência. Por se tratar de medição de consumo de água, foi realizada uma revisão sobre os métodos e tecnologias existentes para medição de líquidos. Sobre comunicação, apresenta-se uma pequena descrição sobre o protocolo WM-Bus e uma breve revisão sobre a comunicação por rádio frequência, sua origem e regulamentação. E por fim a proposta de solução com a descrição sobre o funcionamento dos códigos, suas características, testes com o protótipo de medição desenvolvido e os resultados alcançados, os quais demonstram que a utilização de um sistema de comunicação sem fio empregando protocolo MQTT em ambientes semelhantes ao Sapiens Parque é viável tecnicamente dentro do contexto de redução do tempo de leitura, agilidade de acesso as informações e na flexibilidade para conexão de diversos tipos de equipamentos diferentes, visto que a tecnologia utilizada não é proprietária.

Palavras chave: MQTT. Sistema de Medição de Água. Protocolos de Comunicação. IoT.

ABSTRACT

This research aimed to develop a wireless communication system for measuring water consumption in the Sapiens ParK project, using MQTT protocol. For that, research was done on the subject who involves remote measurement and the concepts about Internet of Things. A detailed study on the MQTT protocol was carried out, demonstrating how to use the publisher / subscriber model, also about publications of messages, levels of quality of services, important recommendations on the choice and definition of topics, how to use of wildcard characters and the MQTT brokers. In addition, in the course of the research, the Texas Instruments CC1200DK development module is presented as a platform that is compatible with European standards which uses and operates with the WM-bus protocol. It is also shown that this development module can operate in the Brazilian territory, according to ANATEL, regulatory agency of radio frequency bands. As a measure of water consumption, a review was made of existing methods and technologies for the determination of volumes consumed. About communication, a brief description is made for the WM-Bus protocol and also a small revision on radio frequency communication. Finally, the proposed solution, with the description of the developed codes, their characteristics, tests and prototype of the measurement developed and the results achieved, which demonstrate that the use of a wireless communication system employing MQTT protocol in Sapiens Park similar environments is technically feasible within the context of reduced reading time and agility of access the information and also to the flexibility to connect different types of different equipment, since the technology used is not proprietary.

Keywords: MQTT. *Water Metering System*. Communication Protocols. IoT.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

ABNT- Associação Brasileira de Normas Técnicas

ADC – *Analog-to-Digital Convert*

AHB - *Advanced High-perfomance Bus*

ANA – Agência Nacional de Águas

ANATEL – Agência Nacional de Telecomunicações

ASCII – *American Standard Code for Information Interchange*

B2G *-Building to Grid*

bps – bits por Segundo

Capes – Coordenação de Aperfeiçoamento de Pessoal de Nível Superior

CASAN – Companhia Catarinense de Águas e Saneamento

CERTI – Centro de Referências em Tecnologias Inovadoras

CPL – *Common Public License*

CPU –*Central process Unit*

dB – Decibel

EDL – *Eclipse Distribution License*

EPL – *Eclipse Public License*

ERP – *Enterprise Resource Planning*

EN - *Europäische Norm*

ESMA – *European Smart Meter Alliance*

FAO - *Food and Agriculture Organization of the United Nations*

GPIO – *General Purpose Input/output*

HTTP – *Hypertext Transfer Protocol*

I2C – *Inter-Integrated Circuit*

IBGE – Instituto Brasileiro de Geografia e Estatística

Id – *Identity*

IEEE - *Institute of Electrical and Electronics Engineers*

IERC – *Internet of Things European Research Cluster*

IoT – *Internet of Things*

IP - *Internet Protocol*

IrDA – *Infrared Data Association*

ISM – *Industrial, Scientific and Medical*

Kbps – Quilo bits por segundo

KHz – Quilo Hertz

LCD – *Liquid Crystal Display*

LED – *Light Emitting Diode*

M2M – *Machine to Machine*

M-Bus – *Metering Bus*

MHz –Mega Hertz

MIT – *Massachusetts Institute of Technology*

MQTT – *Message Queue Telemetry Transport*

ms – Milissegundos

OASIS – *Advancing Open Standards for the Information Society*

OMS-*Group*– *Open Metering System Group*

PBA – Plano Básico Ambiental

PWM – *Pulse-Width Modulation*

QoS – *Quality of Service*

RAM – *Random Access Memory*

REFO – *Internal Trimmed Low-Frequency Reference Oscillator*

RF – Rádio frequência

RFID - *Radio Frequency Identification*

RISC – *Reduced Instruction Set Computer*

ROM – *Read-Only Memory*

RTC – *Real Time Clock*

Rx – *Receiver*

SCADA – *Supervisory Control And Data Acquisition*

SciELO – *Scientific Electronic Library Online*

SENAI – Serviço Nacional da Indústria

SoC – *System on Chip*

SPI – *Serial Peripheral Interface*

SRD – *Short Range Device*

TCP - *Transmission Control Protocol*

TERESA - Trusted Computing Engineering for Resource Constraint Embedded System Application

Tx – *Transmitter*

UFSC – Universidade Federal de Santa Catarina

UART – *Universal Asynchronous Receive Transmit*

URL – *Uniform Resource Locator*

USCI – *Universal Serial Communication Interface*

UTF-8 – *8 bit Unicode Transformation Format*

VHLL – *Very High Level Language*

VLO – *Very Low Oscillator*

WEP – *Wired Equivalent Privacy*

Wi-Fi – *Wireless Fidelity*

WM-Bus – *Wireless Metering Bus*

WPA – *Wi-Fi Protected Access*

WPA2 - *Wi-Fi Protected Access II*

XMPP – *Extensible Messaging and Present Protocol*

SUMÁRIO

1 INTRODUÇÃO

Nesta pesquisa se investigou a constituição de um sistema de comunicação para medição de consumo de água no empreendimento Sapiens Parque, utilizando os conceitos de IoT e *Cloud Services*.

Sendo um empreendimento de inovação localizado no norte da ilha de Florianópolis com previsão de instalação de mais de 250 unidades empresariais distribuídas por uma extensão territorial de quase 4,3 milhões de metros quadrados, a criação do Sapiens Parque, além das questões técnicas de engenharias, foi fundamentada em estudos sobre os impactos socioambientais da região. Estes estudos apontaram algumas ações importantes e uma delas é a restrição sobre o consumo de água potável.

De acordo com a Nota Técnica 02 (SAPIENS PARQUE, 2014, p.8), o consumo de água potável proveniente da concessionária local, não poderá ultrapassar 50% do total consumido e para suprir a demanda fontes alternativas deverão ser utilizadas. Dentro destas fontes estão previstas a captação de água de chuva e o reuso de águas residuais tratadas pelo próprio empreendimento na proporção de 14% e 36% respectivamente, podendo sofrer pequenas alterações.

Para comprovação dos valores consumidos e principalmente para cobrança sobre os serviços de fornecimento, medidores individualizados deverão ser instalados em cada unidade, ou seja, ao final da implantação existirão cerca de 750 medidores de consumo para os três tipos de água.

Como uma empresa privada, com participação acionária majoritária do governo do estado de Santa Catarina, o Sapiens Parque possui regras específicas para contratação de serviços e compra de materiais e, somando-se a isto o extenso prazo de implantação, a compatibilização de equipamentos de medição será um problema visto que não existirão garantias de continuidade de aquisição de tecnologias proprietárias.

Dessa forma a utilização de um sistema independente de tecnologia proprietária é essencial para garantir a comunicação entre os equipamentos de medição do consumo de águas do Sapiens Parque, pois Turcu, Turcu e Gaitan (2012)

em trabalho realizado sobre o uso da tecnologia RFID (acrônimo do inglês – *Radio Frequency Identification*[1]) para sistema de distribuição de água, mostram as dificuldades em gerir um sistema de medição em um grande espaço territorial com diversos tipos de equipamentos que não possuem interoperabilidade.

Frente às limitações descritas acima e a consideração como uma única unidade consumidora perante a concessionária local, a gestão interna da distribuição de água potável bem como a respectiva cobrança dos valores consumidos deverão ser realizadas pela equipe do Sapiens Parque e a medição remota será um fator importante para agilidade e confiança nas informações, visto que as estimativas para o final da implantação indicam um período de quatro a sete dias para a leitura de todos os medidores, se realizadas manualmente.

Assim o objetivo principal deste trabalho foi o desenvolvimento de sistema de comunicação sem fio para medição do consumo de água no empreendimento Sapiens Parque empregando protocolo MQTT como alternativa às soluções comerciais proprietárias que dificultarão a interoperabilidade de equipamentos instalados dentro do Sapiens Parque.

Para esclarecer a questão do objetivo principal, alguns objetivos específicos foram traçados os quais são:

a) Pesquisa bibliográfica sobre medição remota;
b) Estudo detalhado sobre o protocolo MQTT;
c) Revisão bibliográfica sobre medição de líquidos;
d) Definição e estudo sobre o hardware utilizado na pesquisa;
e) Revisão sobre comunicação por radio frequência e pesquisa sobre o protocolo WM-Bus.

A pesquisa bibliográfica sobre medição remota foi realizada em diversos bancos de dados de diversas universidades, congressos e de institutos de pesquisas.

No estudo detalhado sobre o protocolo MQTT, levantou-se as informações sobre sua origem, os conceitos e forma de utilização, o níveis de qualidade de serviços

[1] Livre tradução: Identificação por Radio Frequência

(QoS) e seus impactos durante a transmissão das informações. Foi realizada uma pesquisa minuciosa sobre o funcionamento do modelo *Publisher/Subscriber* suas etapas de comunicação, demonstrando o que acontece durante uma conexão e também um relato sobre a criação de tópicos e as formas de inscrições e utilização de caracteres especiais (coringas).

Por se tratar de medição de consumo de líquidos, uma revisão bibliográfica detalhada foi realizada sobre as tecnologias e métodos existentes para determinação de consumo de água e suas diversas aplicações.

A definição do *hardware* para a utilização na pesquisa partiu do princípio que este deveria ser capaz de trabalhar com o protocolo WM-Bus o que foi garantido pela utilização do módulo de desenvolvimento CC1200DK da *Texas Instruments* o qual opera dentro das especificações da Norma Européia EN-13757 que dispõe sobre os formatos do protocolo WM-Bus.

Com base neste hardware, foi realizado um estudo para entender os componentes disponíveis para pesquisa demonstrando partes da sua composição.

Uma revisão sobre o surgimento da comunicação por rádio frequência foi elaborada apresentando o início das comunicações e suas faixas de utilização bem como uma breve pesquisa sobre o protocolo WM-Bus.

E por fim a definição da proposta de solução, as características de funcionamento dos seus componentes englobando os *hardwares*, protocolos, códigos de programação, *broker* MQTT e interface na WEB, onde se conclui que uma proposta de utilização de sistema de comunicação sem fio independente das soluções comerciais é viável dentro de certos parâmetros e disponibilidade de alguns serviços.

2 REVISÃO BIBLIOGRÁFICA

Para compreensão e embasamento desta dissertação, inicialmente propõem-se apresentar o levantamento de pesquisas realizadas acerca do tema proposto bem como detalhar alguns conceitos utilizados neste trabalho. Com isto serão apresentados alguns trabalhos relacionados, a relação do conceito da Internet das Coisas e a proposta de solução e o que é e como funciona o protocolo MQTT.

2.1 Trabalhos relacionados

Com a finalidade de identificar pesquisas em desenvolvimento acerca do tema desta dissertação, foram realizadas consultas nas seguintes bases de dados: Banco de teses e dissertações da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (Capes); *Scientific Eletronic Library Online* (SciELO), *Institute of Electrical and Electronics Engineers* (IEEE), congressos sobre sistemas de medições remotas e inteligentes e também em repositórios de teses e dissertações de universidades brasileiras e internacionais.

Durante a pesquisa identificou-se que muito dos trabalhos realizados trazem junto os conceitos de *cloud service*, *smartgrid, smart metering* e *Internet of Things* como partes integrantes para solução de medição remota de consumo sobre serviços públicos tais como fornecimento de água, gás e energia elétrica.

Um sistema de monitoramento distribuído por meio da Internet das Coisas, com o intuito de reduzir o consumo inadequado e as perdas de água em um sistema de distribuição, é proposto por Turcu, Turcu e Gaitan (2012). Neste estudo os autores se baseiam na tecnologia RFID utilizando sensores junto aos pontos de consumo e mostram as dificuldades em gerir um sistema de medição em um grande espaço territorial com diversos tipos de equipamentos que não possuem interoperabilidade. A proposta de solução utilizou um sistema de controle de supervisão e aquisição de

dados SCADA[2] e outra aplicação comercial de sistema de gestão empresarial ERP[3] para emissão de faturas para cobrança dos serviços públicos.

O mesmo problema de extensão territorial é ponto de preocupação desta dissertação e de forma semelhante como solução será utilizado equipamentos com tecnologia de rádio frequência que se comunicará entre si e alguns destes serão utilizados como pontos de concentração de dados que se comunicarão com o servidor por meio de uma rede local sem fio pertencente ao Sapiens Parque, mas como nem todos os equipamentos conseguirão ser inseridos nesta rede por causa da área de abrangência, a utilização da comunicação via rádio frequência é importante para aumentar o alcance dentro do empreendimento.

A preocupação com o supervisório é outro item importante a ser considerado neste trabalho, pois ainda segundo Turcu, Turcu e Gaitan (2012) alguns aspectos impactam diretamente na operação e manutenção do supervisório tais como: arquitetura, ferramentas de programação e definições do grupo de desenvolvimento do sistema supervisório.

A proposta de solução desta dissertação prevê a utilização de um ambiente de supervisão e aquisição de dados em um ambiente *online*, onde inicialmente estarão disponíveis informações básicas de consumo por unidade e posteriormente os valores globais do empreendimento.

Squartini et al. (2013, p. 614) abordam o assunto sobre *smart metering* e cita que vários esforços foram realizados para o desenvolvimento de protocolos de comunicação adequados para o gerenciamento e transmissão de dados e o *Metering-Bus* (M-Bus) é um exemplo com ampla difusão na Europa.

Ainda neste estudo identifica-se que dentro das frequências propostas pelo *OMS-Group* (*Open Metering System Group)* para equipamentos de medição, a frequência de 868 MHz apresenta melhor rendimento e vida útil de bateria quando confrontada à frequência de 169 MHz, mesmo que àquela necessite de maior potência

[2] SCADA – Supervisory Control And Data Acquisition

[3] ERP – Enterprise Resource Planningou em tradução livre

para manter o sinal de transmissão e com isto a proposta de solução desta dissertação, que prevê a utilização de um kit de desenvolvimento que trabalha na faixa de 868 MHz, encontra respaldo quando somado aos resultados apresentados por Spisante et al (2014, p.16) onde a questão do tamanho da antena torna-se um problema na frequência de 169MHz.

O potencial das tecnologias *smart meter* e *cloud service* apresentado por Dukan e Kovari (2013) incentiva a utilização de tais técnicas e reforça a tendência do desenvolvimento de sistemas de monitoramento onde cada vez mais, diversos equipamentos não somente de medições, mas todo e qualquer dispositivo poderá estar conectado à rede mundial e ser acessado remotamente. A questão do uso dos serviços em nuvem, reforçado pelos autores, é apresentado como solução para gerenciamento dos serviços associados à tecnologia de informação onde os prestadores de serviços podem utilizar técnicas de escalonamento de infraestrutura e também estruturas heterogêneas conforme a necessidade da região, bem como implantar infraestruturas de forma gradual e em diferentes regiões. No caso do Sapiens Parque, apesar de já existir uma infraestrutura implantada, com rede local sem fio e servidor de dados disponível, para ser utilizada pelo sistema de medição a manutenção dos dados e adequações de equipamentos podem não ser os mais interessante para o sistema e a utilização de um *cloud service* pode ser considerado.

Um modelo para companhia de serviços públicos, tais como água e energia elétrica, utilizando uma rede de sensores sem fio é proposto por Aghaei (2011), onde o autor realizou estudos para que, além de efetuar remotamente as leituras, as empresas pudessem suspender o fornecimento de água e energia caso os consumidores não realizassem os pagamentos mensais.

A implementação e análise de um dispositivo eletrônico que pode ser usado como um concentrador em um sistema de medição de gás inteligente é apresentado por Mihajlović et al. (2016). Neste trabalho os autores indicam a melhoria na qualidade do monitoramento da rede de gás, pois os dispositivos *smart meters* empregados conseguem em tempo real, realizar medições e detectar problemas no sistema de distribuição. Os autores abordam ainda que os equipamentos que trabalham com o

padrão WM-Bus estão sendo utilizados como base para a nova infraestrutura avançada de medidores e que estes dispositivos, que trabalham nas frequências subgiga Hertz, continuam evoluindo para se adaptarem a um ambiente em constante mudança, aproveitando-se das melhorias tecnológicas impulsionada pela Internet das Coisas. A proposta dos autores prevê a unificação do sistema de medição inteligente onde os medidores de gás, água e energia poderão se comunicar e enviar os dados de medição em tempo real, mas encontra problemas tanto na questão de disponibilidade de equipamentos inteligentes nos três serviços como também na colaboração entre as companhias de serviços públicos. No tema desta dissertação será abordado somente o consumo de água do empreendimento, com a instalação de uma unidade remota que atuará como concentrador de dados alocado em ponto estratégico ao alcance da rede de dados sem fio para receber as leituras dos medidores e transferi-las ao servidor de dados. Os demais serviços públicos não serão objeto deste trabalho.

Lee, Eun e Oh (2008) propõem um medidor digital de água com um sensor magnético e transferência de dados por meio do protocolo *ZibBee*. Neste trabalho os autores relatam a dificuldade encontrada para garantir as fontes de energia para os medidores digitais, pois em muitos deles as baterias são seladas e impossíveis de serem substituídas.

A utilização do protocolo *ZigBee* apresenta dificuldades com alcance da transmissão o qual é reforçado pelo trabalho de Cao e Zhang (2009), que utilizando uma rede de sensores sem fio e protocolo *ZigBee* para transferência dos dados encontraram problemas de interferências para distâncias superiores a 100 metros.

Estudo sobre os protocolos de comunicação e tecnologias para medição remota de água em edifícios residenciais foi realizado por Casale et al. (2016). Neste estudo os autores assinalam a dificuldade de transmissão de dados frente às interferências causadas pela própria edificação (lajes de concreto, paredes de alvenarias etc) e também pela restrição da própria capacidade de transmissão dos dispositivos subgiga hertz.

Testes realizados (WALLACE, 2017) na cidade de Oslo, capital da Noruega, com kit de desenvolvimento semelhante[4] ao utilizado na proposta de solução mostram ótimos resultados na frequência de 868 MHz com alcance de até 1,3km de distância entre dois equipamentos em linha reta sem obstáculos[5].

Por fim, o trabalho desenvolvido por Zivić, Ur-Rehman e Ruland (2015), indica que a granularidade e pontualidade das informações que possui um sistema inteligente de medição, melhoram significativamente a gestão dos ativos pelas companhias de energia, pois os próprios medidores de energia são utilizados como fontes de informações a respeito de falhas e qualidade no fornecimento energia. Realizando uma análise um pouco mais precisa verifica-se que este exemplo pode ser facilmente replicado para as companhias de água e saneamento já que os medidores de água podem fornecer informações a respeito do volume de água consumido em vários pontos distribuídos e com pequenas adaptações fornecer também a informação sobre a pressão do sistema destes pontos. Ainda neste trabalho, os autores apresentam as várias iniciativas relacionadas a projetos de medição inteligente ou *Smart Metering Projects*, entre eles destaca-se:

a) B2G – *Building to Grid – Smart Grids Model Region Salzburg.* Projeto relacionado a otimização da operação de rede por meio da cooperação entre redes de edifícios inteligentes;
b) *OPENmeter* – Projeto financiado pela união européia que teve como objetivo principal especificar um conjunto abrangente de padrões abertos e públicos para infraestruturas de medição avançada (AMI);
c) *European Smart Meter Alliance* (*ESMA*) – projeto que definiu e propagou as melhores práticas em medição inteligente em todos os países membros da união européia, buscando a economia de energia;
d) *TERESA – Trusted Computing Engineering for Resource Constraint Embedded System Application* – Fornece diretrizes para desenvolvimento

[4] CC1120 Evaluation Module Kit 868-915 MHz – disponível em www.ti.com/tool/CC112EMK-868-915 - acesso em 12/11/2017

[5] Estudo disponível em www.ti.com/lit/an/swra479a/swra479a.pdf - acesso em 12/11/2017

de processos de engenharia da computação confiável no setor de sistemas embarcados, onde as principais áreas de aplicação investigadas no *TERESA* foram: automotivo, controle doméstico, controle industrial e metrologia (gateway sem sistemas de medição inteligente);

e) *Smart Metering Project* – Projeto desenvolvido em parceria entre Letônia e Lituânia e também com apoio do programa de desenvolvimento regional europeu, contemplou pesquisa para implementação, teste, melhorias e ajustes dos sistemas automatizados de leitura dos medidores.

2.2 Internet das coisas e a proposta de solução

Mark Weiser (1991, p.1, tradução nossa)[6] idealizador da computação ubíqua mencionou em publicação no *Scientifc American Journal* que: “Elementos especializados de hardware e software, conectados por fios, ondas de rádio e infravermelhos, serão tão ubíquos que ninguém notará sua presença”.

Nesta publicação Weiser previu várias situações, hoje cotidianas, que no início da década de noventa ainda não era possíveis, tais como agendas interligadas, acesso a informações instântanes de trânsito, localização de pessoas em ambientes diversos, “surfar” na rede em páginas pessoais entre outras tantas coisas conectadas por meio da internet.

O foco de Weiser era que “as coisas” estariam conectadas entre si e na internet de um jeito tão comum aos seres humanos que estes aprenderiam usá-las de forma cotidiana tal que, essas“coisas” nem seriam percebidas como elas realmente são.

Segundo Lacerda (2015, p.45) “o termo ‘*Internet of Things’* (IoT) foi cunhado em 1999 por Kevin Ashton, co-fundador do *Auto-ID Center* do *Massachusetts Institute of Technology* (MIT).”.

---

[6] ´Livre tradução: “*Specialized elements of hardware and software, connected by wires, radio waves and infrared, will be so ubiquitous that no one will notice their presence*”.

Outra definição é feita por Atzori, Iera e Morabito (2010, p.1, tradução nossa) que menciona que IoT é:

> "... a presença generalizada em torno de nós de uma variedade de coisas ou objetos - como etiquetas de identificação de radiofrequência (RFID), sensores, atuadores, telefones celulares, etc. - que, por meio de esquemas de endereçamento únicos, são capazes de interagir uns com os outros e cooperar com seus vizinhos para alcançar metas comuns."

Neelie Kroes, vice-presidente da Comissão Européia[7] e comissária da Agenda Digital para Europa[8] (IERC, 2012, p.7, tradução nossa) menciona que as aplicações da IoT oferecem benefícios significativos às pessoas de um modo geral pois ajuda desde economizar energia e aumentar o conforto mas também a independência das pessoas para terem uma vida mais saudável.

Vermesan et al. (2012,p.26, tradução nossa) define a Internet das Coisas como:

> Uma infraestrutura de rede global dinâmica com capacidade de autoconfiguração baseada em protocolos de comunicação padrão e interoperáveis onde as "coisas" físicas e virtuais têm identidades, atributos físicos e personalidades virtuais e usam interfaces inteligentes e são perfeitamente integradas na rede de informações."

Percebe-se que com o advento da IoT as mudanças no nosso mundo estão cada vez mais rápidas. Quantidades enormes de informações estão disponíveis por meio das "coisas" instaladas fisicamente em diversos lugares, conectadas entre si e por meio da internet. Considera-se nesta pesquisa que o próprio ser humano podem ser considerado como "coisa", pois existem hoje sistemas de monitoramento da saúde das pessoas que estão conectadas a outros equipamentos ou mesmo ainda quando informações são inseridas como comentário a respeito de um estabelecimento

[7] https://europa.eu/european-union/about-eu/institutions-bodies/european-commission_pt acesso em 04/01/2018

[8] http://www.eurocid.pt/pls/wsd/wsdwcot0.detalhe?p_cot_id=6280 acesso em 04/01/2018

comercial em uma rede social ou um site de avaliação. Essa informação pode ficar disponíveis para qualquer pessoa em qualquer parte do mundo.

A disponibilidade de informações de forma fácil, rápida e precisa concede aos seres humanos o poder de atuar de forma mais apropriada, eliminando erros de tomadas de decisões, melhorando a qualidade das ações e diminuindo custos desnecessários.

Lacerda (2015, p.17), cita que:

> O extraordinário potencial da IoT é o poder que confere aos objetos de uso cotidiano de capturar, processar, armazenar, transmitir e apresentar informações. Interligados em rede, os objetos são capazes de realizar ações de forma independente e gerar dados em quantidade e variedade exponenciais, como produto das interações. Nesse contexto, a informação passa a fazer parte do ambiente, e configuram-se novas formas de atuação das pessoas no mundo.

Neste contexto, entende-se que o trabalho desenvolvido nesta dissertação também pode ser considerado uma solução contendo elementos de IoT, onde sensores instalados junto a medidores de água estarão conectados entre si por meio de uma rede que operará na faixa de rádio frequência e alguns destes estarão conectados a uma rede WiFi para transmissão de dados a um servidor para disponibilizar as informações por meio de uma interface na Internet.

A forma como estas informações serão utilizadas dependerá de cada usuário e como exemplos podemos citar: para a equipe de gestão do empreendimento Sapiens Parque, serão informações importantes para cobrança de valores monetários das unidades consumidoras, mas se for para uma unidade consumidora os dados poderão ser importantes para gerenciamento da proporção do tipo de água consumida adequações de processos para redução do consumo etc.

2.3 Protocolo MQTT

O protocolo MQTT, do inglês *Message Queue Telemetry Transport*, é um protocolo de mensagens, extremamente simples e leve que foi projetado para uso com dispositivos que não necessitam de grande consumo de banda de transmissão. Esses dispositivos interagem como *Publisher/Subscriber*, ou seja, a troca de informação é feita por publicações endereçadas aos subscritos nos ambientes de publicação chamados de *brokers* (intermediários). Este último faz o endereçamento das publicações e disponibiliza as publicações aos subscritos que estão registrados em determinados tópicos.

Pela simplicidade de codificação e sem a necessidade de muitas informações além do próprio dado a ser transmitido (sem *overhead*) tornou-se Ideal para o cenário da IoT.

Com o MQTT, os dispositivos não precisam estar sincronizados constantemente ao servidor. Os sensores, ao enviarem suas leituras, permitem que a própria rede descubra qual o caminho e a melhor sincronização para entregar a informação (YUAN, 2017). Além disto, o MQTT possibilita a utilização de dispositivos de baixo poder de processamento e baixa capacidade de memória.

Desenvolvido no final da década de 90, por Andy Stanford-Clark da IBM e por Arlen Nipperda da Arcom (hoje Eurotech) com o propósito de vincular sensores de linhas de oleodutos a satélites, é um protocolo para comunicação que não exige sincronismo entre o emissor e receptor tanto no espaço como no tempo (YUAN, 2017), isso favorece a escalabilidade em ambientes de redes que não são confiáveis sob a ótica de continuidade de serviços.

O MQTT tornou-se padrão aberto OASIS[9] em 2014, com suporte a diversas linguagens de programação e implementações em software livre (YUAN, 2017) e diferentemente de outros protocolos, tais como o HTTP (*Hypertext Transfer Protocol*) e

[9] OASIS – *Advancing open standards for the information society* - https://www.oasis-open.org/org

o XMPP (*Extensible Messaging and Present Protocol*), o MQTT consegue ser bem mais leve e flexível por causa do modelo de publicação e assinatura.

Muito popular nos projetos IoT, *M2M*[10] e embarcados, o MQTT também está ganhando espaço nas aplicações WEB e em dispositivos móveis que exigem distribuição de mensagens eficientes (HILLAR, 2017).

2.3.1 Modelo Publisher/Subscriber

O protocolo MQTT é baseado no modelo de publicação e subscrição de mensagens e tem por definição dois personagens, o *broker* de mensagens e o cliente.

O *broker*, também chamado de servidor MQTT, é um intermediário que tem a função de transportar as informações dos tópicos funcionando como o elo entre os publicadores e subscritos. Responsável pela autenticação e autorização dos clientes MQTT, é ele que recebe e disponibiliza as mensagens nos tópicos.

Um cliente pode ser qualquer “coisa”, desde um simples sensor de coleta de informações, passando por grandes servidores/processadores de dados ou até mesmo usuários comuns que possam interagir com o *broker* por meio de dispositivos tais como celulares ou computadores. Pode ser ainda ao mesmo tempo um publicador de mensagens bem como um personagem subscrito para receber as mensagens nas quais possui interesse. Essa troca de mensagens de publicação/ subscrição se dá por meio dos tópicos.

A mensagem é constituída pelo tópico e pelo dado a ser transmitido, este último conhecido como *payload*.

Tanto os publicadores como os subscritos podem estar inscritos em mais de um tópico ao mesmo tempo, bem como ser um agente publicador em um tópico e estar subscrito em outro.

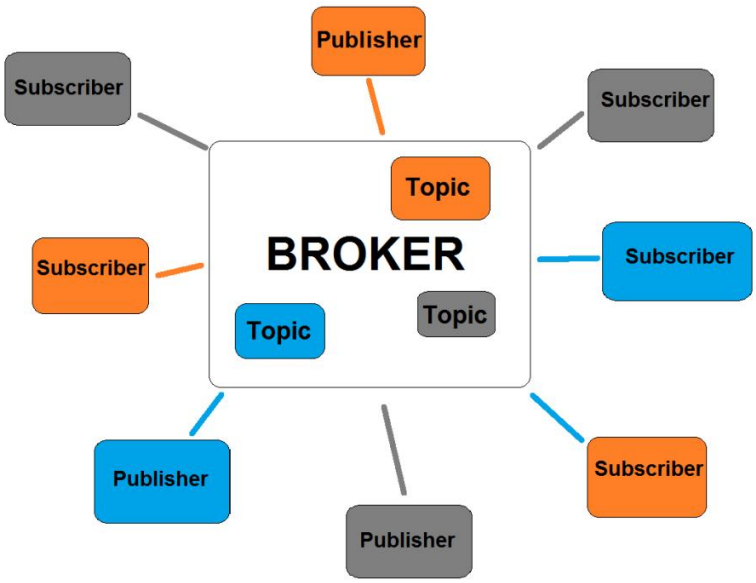
---

[10] *Machine-to-Machine*

Um tópico é um tema de interesse do cliente criado no *broker*. Semelhante ao conceito de URL (*Uniform Resource Locator*) é o local do *broker* onde serão inseridas as informações que estarão disponíveis para leitura pelos diversos clientes inscritos para recebê-las.

A figura 1 mostra um diagrama de comunicação entre publicadores e subscritos por meio de um *broker*.

Figura 1 - Modelo Publisher/ Subscriber



Fonte: Elaboração própria

A grande vantagem do MQTT é que, tanto os publicadores como os subscritos não precisam estar sincronizados para enviar e receber as mensagens, entretanto é possível enviar mensagens para o *broker* de forma síncrona e somente continuar a execução após a confirmação e recebimento.

Como o publicador e o subscrito estão desacoplados, não há como saber se existe algum inscrito no tópico que “ouvirá” as mensagens que serão enviadas e, dependendo dos requisitos da solução e níveis de qualidade dos serviços, às vezes é

necessário tornar os inscritos em publicadores para enviarem uma mensagem indicando que receberam e processaram as informações (HILLAR, 2017).

Para enviarem mensagens ou acessarem as informações dos tópicos cada cliente necessita realizar uma conexão com o servidor.

Existem diversos servidores MQTT disponíveis para várias plataformas, tais como LINUX, Windows e MacOS, com versões em código aberto, gratuitas ou ainda em versões pagas. A escolha do servidor MQTT deve levar em conta as necessidades específicas das aplicações que estão em desenvolvimento bem como os recursos financeiros disponíveis para o projeto.

Para realizar uma conexão, um cliente precisa enviar ao servidor um pacote de dados denominado CONNECT, o qual deverá possuir todas as informações necessárias para iniciar a conexão (HILLAR, 2017). O pacote de dados contido no CONNECT deve incluir os itens: *ClientId, CleanSession, UserName, Password, ProtocolLevel, KeepAlive* e *Will.*

O *ClientId* é a sequência de caracteres exclusivo que identifica cada cliente que se conecta ao servidor e é utilizado pelo servidor para identificar o estado que o cliente mantêm com o servidor em relação à sessão. Não são aceitos caracteres especiais tais como acentos, vírgulas, traços entre outros na formatação de um *ClientId*.

CleanSession é a *flag* que especifica o que acontece após um cliente se desconectar do servidor. Se for configurado como 1 (ou verdadeiro), as informações serão mantidas enquanto a sessão com o servidor permanecer ativa. Após ocorrer a desconexão todas as informações serão apagadas e o cliente precisará se inscrever novamente no tópico após reconectar. Se configurado como zero (ou falso) a sessão é dita como persistente, ou seja, quando o cliente se desconectar, todas as inscrições que existiam permanecerão ativas e todas as mensagens subsequentes serão armazenadas. A sessão persistente só funciona para os níveis de segurança QoS1 e QoS2 e o armazenamento durará até o momento da reconexão do cliente ou a capacidade de memória do servidor (HILLAR, 2017).

Password e UserName são respectivamente as senhas e os nome dos usuários a serem autenticados e autorizados a permanecerem em uma sessão com o

servidor. Para utilizar a autenticação por meio do nome de usuário e senha, as *flags* *UserName* e *Password* devem ser configuradas como 1.

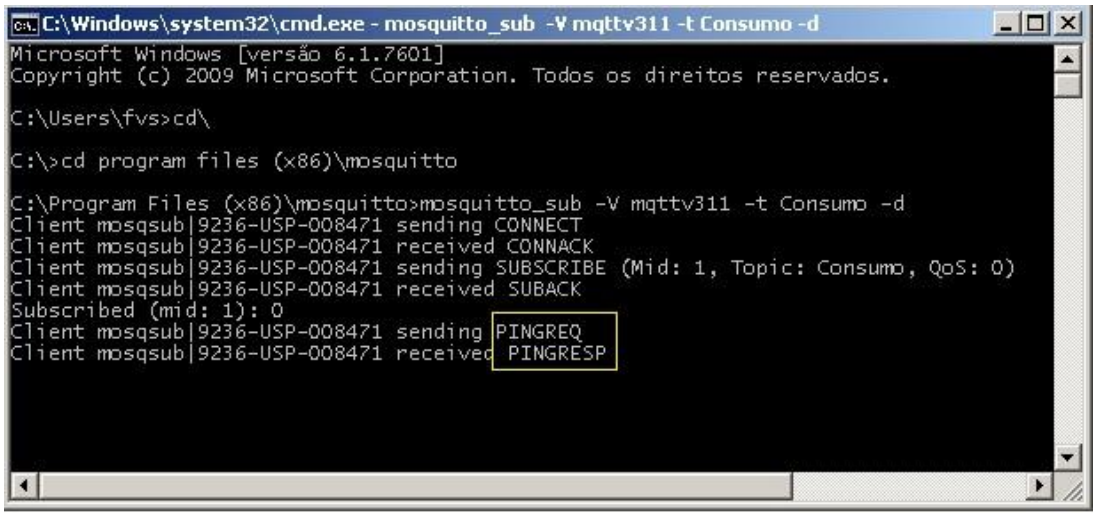
ProtocolLevel indica a versão do protocolo MQTT que o cliente solicita para ser utilizada pelo servidor.

KeepAlive é o tempo máximo, definido em segundos, que o cliente e servidor permanecem sem trocar mensagens, permitindo ao cliente perceber se o servidor não está mais disponível sem ter que aguardar o tempo limite do TCP/IP. Neste processo o cliente envia pelo menos uma mensagem dentro do tempo estipulado e na ausência de mensagens, é enviada uma mensagem “ping” muito pequena. Para habilitar esta função a *flag* dever ser configurada em 1.

Existem duas mensagens envolvidas no fluxo do *KeepAlive*, a primeira é o PINREQ que é enviado pelo cliente para indicar ao servidor que ele ainda está online, mesmo que ele não esteja enviando mensagens e a segunda é o PINRESP que é enviada pelo servidor após receber o PINREQ, informando ao cliente que ainda está disponível. Tanto o PINREQ como o PINRESP não possuem *payload.*

A figura 2 mostra a sequência e informações trocadas entre um cliente e servidor

Figura 2 - Mensagens PINREQ e PINRESP



Fonte: Elaboração própria

A *flagWill* define ao servidor que uma mensagem deve ser armazenada para envio aos demais clientes em caso de falha de conexão. Isto é útil para identificar possíveis clientes que são desconectados inesperadamente por problemas de rede ou outros problemas quaisquer.

Se o servidor não receber nenhuma mensagem ou um PINREQ dentro dos prazos estabelecidos ele fechará a conexão e se o cliente configurou a *flag* "*Will*" como 1, enviará a mensagem "*last will*" ou última vontade.

No caso de desconexão inesperada de um cliente o servidor deve publicar as mensagens de "*Will"* imediatamente e no caso de falha do servidor, este pode adiar a publicação até a reinicialização subsequente (OASIS, 2014).

Após receber o pacote CONNECT e autenticar e autorizar o usuário, o servidor envia um código de retorno CONNACK (*Connect Acknowledge*).

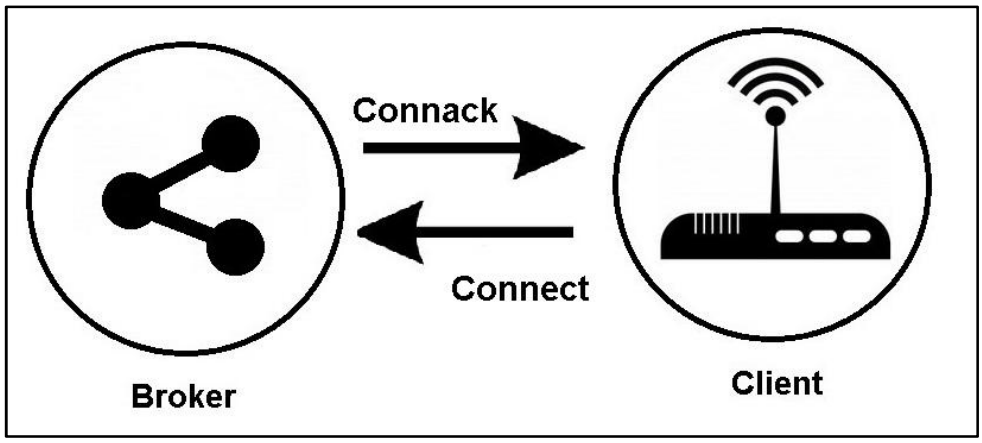
Se um pacote CONNECT adequado for enviado ao servidor e este por algum motivo não conseguir processá-lo, um código de retorno deve ser enviado ao cliente antes de fechar a sessão. O diagrama de conexão entre servidor e cliente pode ser visualizado na figura 3 e um resumo dos códigos de retorno da conexão pode ser visualizado na tabela 01.

Tabela 01 – Códigos de retorno de conexão

Valor	Descrição
0	Conexão aceita
1	Conexão recusada porque o servidor não suporta a versão do protocolo MQTT solicitada pelo cliente
2	A conexão foi recusada porque o *ClientId* (identificador de cliente) especificado foi rejeitado
3	A conexão foi recusada porque o serviço MQTT não está disponível
4	A conexão foi recusada porque o nome de usuário ou senha foi informado incorretamente
5	A conexão foi recusada porque a autorização falhou
6 a 255	Reservado para aplicações futuras

Fonte: Adaptado de OASIS *Standard* – MQTT *Version* 3.1.1

Figura 3 – Diagrama de conexão Servidor e cliente MQTT



Fonte: Adaptado de OASIS *Standard* – MQTT *Version* 3.1.1

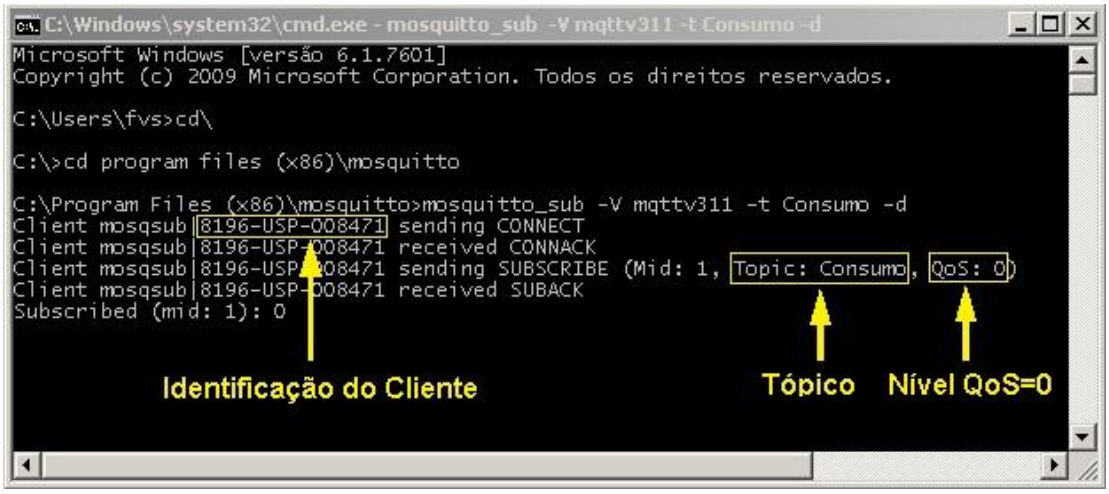
Após estabelecer uma comunicação, o cliente MQTT envia o pacote SUBSCRIBE para o servidor MQTT com um identificador de pacote no cabeçalho, os filtros de tópicos e o nível de qualidade de serviço. Se o nível de qualidade de serviço for omitido no pacote, o servidor assume o valor QoS0 como padrão.

O servidor, após receber um pacote SUBSCRIBE válido, responderá aos clientes inscritos de acordo com nível de qualidade desejado com o pacote SUBACK indicando a confirmação do assinante e confirmando o processamento do pacote SUBSCRIBE.

O pacote SUBACK incluirá o mesmo identificador de pacote recebido por meio do pacote SUBSCRIBE bem como um código para cada par de filtros de tópico e nível de qualidade de serviço QoS.

A figura 4 mostra a troca de informações, CONNECT, CONNACK, SUBSCRIBE e SUBACK entre cliente e servidor durante a conexão e inscrição do cliente “8196-USP-008471” no tópico “Consumo”.

Figura 4 – Mensagens entre cliente e servidor ao estabelecerem comunicação



Fonte: Elaboração própria

Se a subscrição foi bem sucedida o servidor começará a enviar todas as mensagens publicadas relacionadas aos tópicos solicitados.

2.3.2 Publicação de mensagens

Após estabelecer uma conexão com o servidor, um cliente MQTT está apto a publicar mensagens de acordo com os tópicos nos quais está inscrito.

Para realizar uma publicação o cliente precisa enviar um pacote de dados contendo um cabeçalho que inclua os seguintes campos e *flags*: *PackeId, Dup, QoS, Retain* e *TopicName* (HILLAR, 2017*).*

O *PackedId* é utilizado somente para os níveis Q0S1 e QoS2 e associa um valor numérico aos pacotes para identificá-los e possibilita dessa forma, distinguir as respostas a eles relacionados.

Assim como o *PacketId* o *Dup* também é utilizado para os níveis QoS1 e QoS2. O *Dup* quando configurado com 1, permite ao cliente ou servidor reenviar uma

mensagem previamente publicada quando os assinantes não receberam a primeira mensagem.

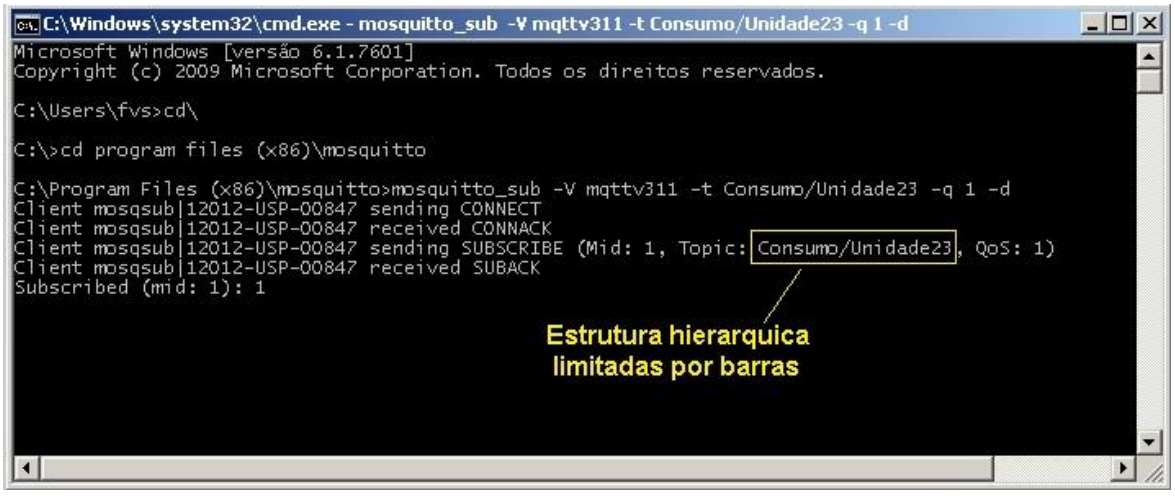
O QoS especifica o nível de qualidade de serviço e podem assumir os valores 0,1 e 2. Sendo o 0 o mais simples e o 2 o mais complexo.

Quando configurado em 1, a *flagRetain* indica ao servidor a forma de armazenamento das mensagens de acordo com o nível de qualidade de serviços especificados. Para o QoS0 não há armazenamento ou retenção de mensagens.

O *TopicName* é formado por uma *string* que identifica o tópico para qual a mensagem deve ser publicada. Os nomes dos tópicos possuem uma estrutura hierárquica que utilizam barras como limitadores (HILLAR, 2017).

A figura 5 mostra exemplo de estrutura hierárquica do tópico “Consumo/Unidade23”.

Figura 5 – Indicação da estrutura hierárquica limitadas por barras



Fonte: Elaboração própria

Após receber um pacote válido, o servidor identifica os subscritos correspondentes ao tópico publicado e encaminha as informações para esses clientes.

A comunicação entre o servidor e o cliente publicador é diferenciada de acordo com o nível de qualidade de serviços associado nas solicitações.

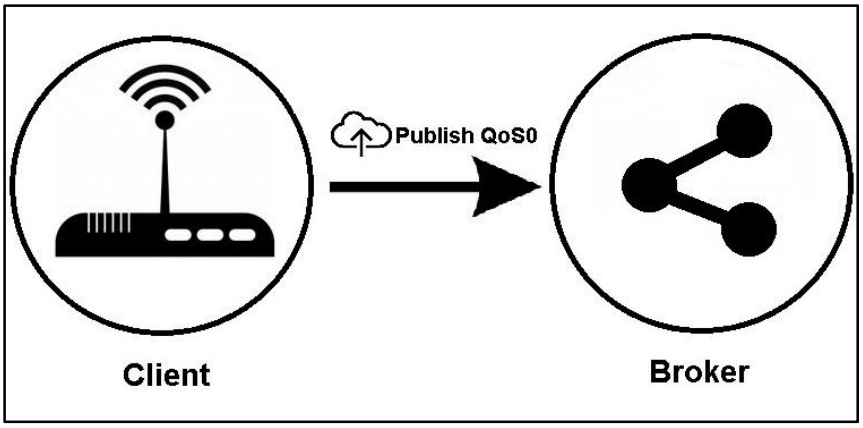
2.3.3 Nível de qualidade de serviço

O nível de qualidade de serviços (QoS – *Quality of Service*) é o acordo entre um remetente e um receptor sobre as garantias e sucesso de entrega da mensagem e, conforme o nível de QoS escolhido, existirá diferenças sobre a classificação do sucesso de uma mensagem entregue (HILLAR, 2017).

O MQTT possui três níveis de qualidade de serviço para garantia de entrega de uma mensagem e diz respeito apenas a entrega de uma mensagem de um único remetente para um único receptor, ou seja, um mesmo servidor pode atuar com vários subscritos com variados níveis de qualidade de serviço desde a recepção como a transmissão de mensagens.

O menor nível de qualidade é o QoS0. Neste nível as mensagens são entregues no máximo uma vez e ocorrem de acordo com os recursos da rede. Nas figuras 6 e 7 pode-se ver, respectivamente, um diagrama de conexão para publicação com nível QoS0 e diagrama de fluxo do protocolo QoS0.

Figura 6 – Diagrama de conexão para publicação com o nível QoS0



Fonte: Elaboração própria

Figura 7 – Diagrama de fluxo do protocolo QoS0

Sender Action	Control Packet	Receiver Action
PUBLISH QoS 0, DUP=0		
	---------->	
		Deliver Application Message to appropriate onward recipient(s)

Fonte: OASIS *Standard* – MQTT *Version* 3.1.1

Nenhuma confirmação de resposta é enviada pelo receptor e nenhuma repetição é realizada pelo remetente (OASIS, 2014). O remetente não armazena e nem agenda outro envio de mensagem e a vantagem está no menor *overhead* existente entre os três níveis de segurança.

Exemplos de troca de informações com um publicador nível Q0S0 e de troca de informações com um subscrito nível Q0S0 podem ser visualizados nas figuras 8 e 9 respectivamente.

Figura 8 – Comunicação com o publicador nível QoS0

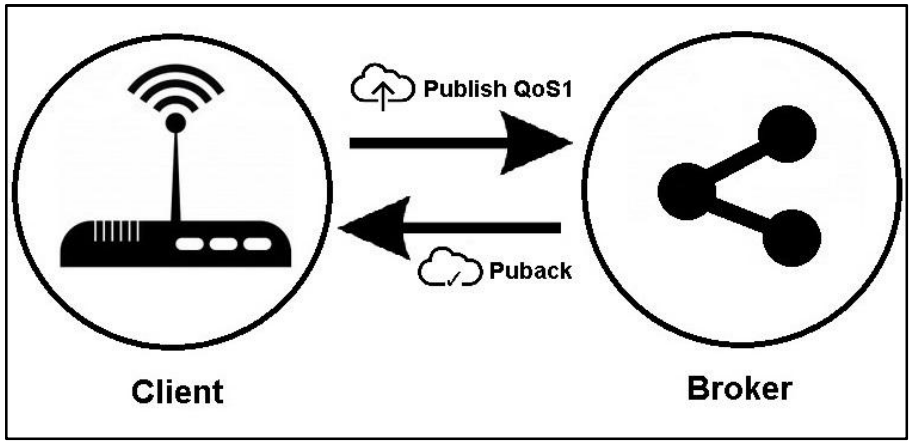
Fonte: Elaboração própria

Figura 9 – Comunicação com o subscrito nível QoS0

Fonte: Elaboração própria

No nível de qualidade QoS1, existe a garantia de entrega da mensagem pelo menos uma vez e a confirmação se dá por meio de um pacote de dados chamados PUBACK – *Publish Acknowlegement* (OASIS, 2014). As figuras 10 e 11 mostram respectivamente o diagrama de conexão para publicação com nível Q0S1 e o fluxo de protocolo QoS1.

Figura 10 – Diagrama de conexão para publicação com o nível QoS1



Fonte: Elaboração própria

Figura 11 – Diagrama de fluxo do protocolo QoS1

Sender Action	Control Packet	Receiver action
Store message		
Send PUBLISH QoS 1, DUP 0, <Packet Identifier>	---------->	
		Initiate onward delivery of the Application Message[1]
	<----------	Send PUBACK <Packet Identifier>
Discard message		

Fonte: OASIS *Standard* – MQTT *Version* 3.1.1

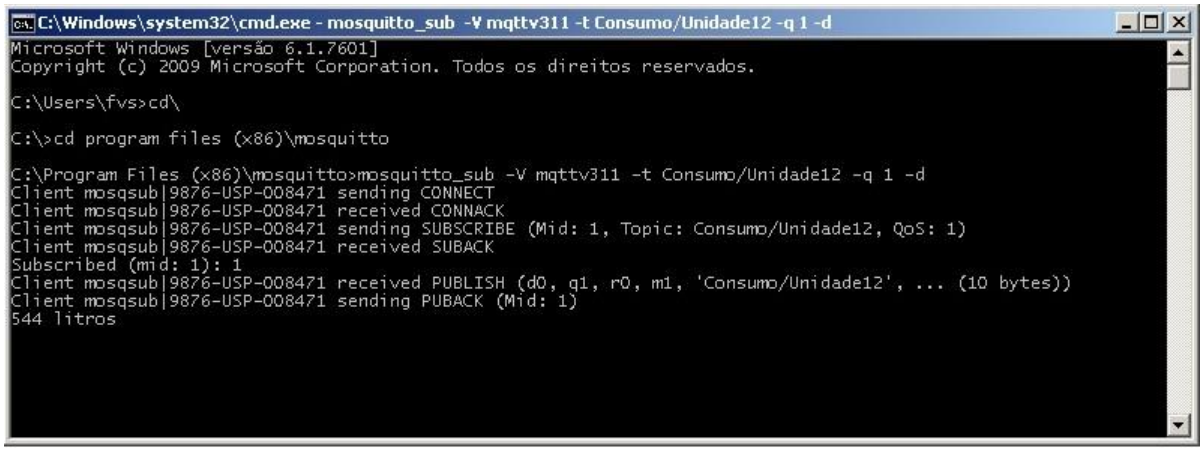
A grande desvantagem deste nível de qualidade de serviço é que ele pode gerar mensagens duplicadas caso o remetente não receber a confirmação de recebimento dentro do tempo estabelecido.

Exemplos de troca de informações com um publicador nível Q01 e de troca de informações com um subscrito nível Q0S1 podem ser visualizados nas figuras 12 e 13 respectivamente.

Figura 12 – Comunicação com o publicador nível QoS1

Fonte: Elaboração própria

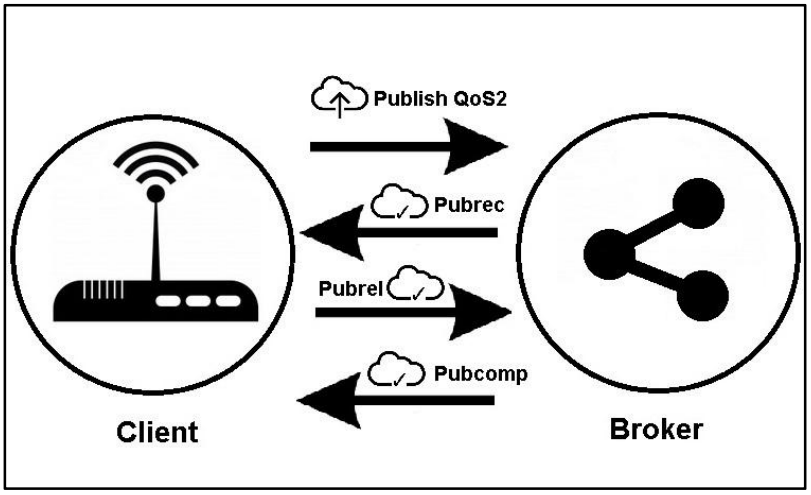
Figura 13 – Comunicação com o subscrito nível QoS1



Fonte: Elaboração própria

O nível de qualidade de serviço mais alto é o QoS2, mas em contrapartida é o mais lento porque exige da rede o maior consumo de banda (HILLAR, 2017). O processo de confirmação exige um tempo maior de resposta, pois a mensagem tem garantia de ser entregue apenas uma vez ao cliente (OASIS, 2014). Na figura 14 temos o diagrama de conexão para publicação com nível QoS2 e na figura 15 o diagrama de fluxo de protocolo nível QoS2.

Figura 14 – Diagrama de conexão para publicação com o nível QoS2



Fonte: Elaboração própria

Figura 15 – Diagrama de fluxo do protocolo QoS2

Sender Action	Control Packet	Receiver Action
Store message		
PUBLISH QoS 2, DUP 0 <Packet Identifier>		
	---------->	
		Method A, Store message Method B, Store <Packet identifier> then Iniciate onward delivery of Application Message
		PUBREC <Packet Identifier>
	<----------	
Discard message, Store PUBREC received <Packet Identifier>		
PUBREL <Packet Identifier>		
	---------->	
		Method A, iniciate onward delivery of the Application Message then discard message Method B, discard <Packet Identifier
		Send PUBCOMP <Packet Identifier>
	<----------	
Discard stored state		

Fonte: OASIS Standard – MQTT Version 3.1.1

Neste nível de qualidade existem dois métodos, denominados A e B, pelos quais o QoS2 pode ser utilizado pelo receptor e diferem-se entre si pelo momento da entrega da mensagem ao destino final.

No método A, a mensagem é armazenada após o recebimento pelo receptor que retorna a mensagem “PUBREC” para o emissor. Este último descarta o conteúdo da mensagem original e armazena o conteúdo do “PUBREC” recebido. Na sequência encaminha a mensagem “PUBREL” para o receptor que após recebimento inicia o processo de entrega da mensagem ao destino final e descarta a mensagem “PUBREL”. Por fim o receptor encaminha a mensagem “PUBCOMP” para emissor que descarta os dados armazenados.

A diferença do método B está no momento em que se inicia o processo de entrega ao destino final o qual começa quando a mensagem chega ao receptor. As confirmações de recebimento e o descarte das mensagens são semelhantes.

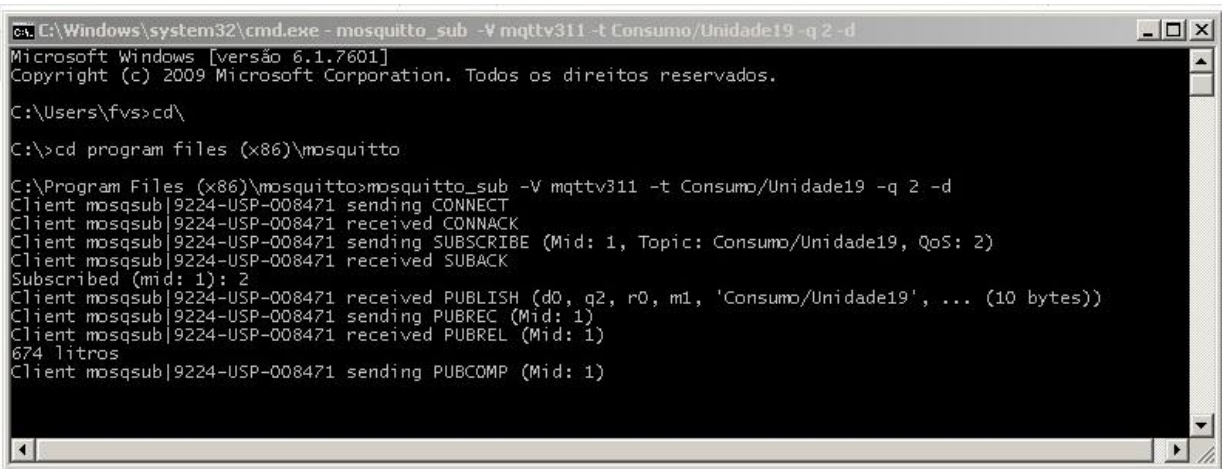
A escolha de qual nível de qualidade de serviço dependerá da aplicação e do projeto que está em desenvolvimento. Muitas vezes necessita-se que as mensagens sejam veiculadas com o menor consumo de banda em uma rede extremamente confiável e se algum dado for perdido não será crítico para o sistema. Neste caso o uso de nível QoS0 seria o mais adequado (HILLAR, 2017). Por outro lado, se a mensagem é extremamente importante em uma rede pouco confiável onde essas mensagens formam comandos para dispositivos que não podem ser executados de forma repetida a escolha pelo nível QoS2 deve ser considerada.

Nas figuras 16 e 17, podem-se ver respectivamente, exemplos de troca de informações com um publicador nível Q02 e de troca de informações com um subscrito nível Q0S2.

Figura 16 – Comunicação com o publicador nível QoS2

Fonte: Elaboração própria

Figura 17 – Comunicação com o subscrito nível QoS2



Fonte: Elaboração própria

Nos testes realizados durante o desenvolvimento da pesquisa utilizou-se apenas o nível de qualidade QoS0.

2.3.4 Escolha e definição de tópicos

Segundo Hillar (2017, p.42, tradução nossa), “a maneira mais fácil de entender nomes de tópicos no MQTT é pensar sobre eles como caminhos em um sistema de arquivos”[11].

Em uma situação onde existirão vários unidades consumidoras, com três tipos de água a serem medidas, a organização desses dados é extremamente importante no momento da publicação.

Um tópico é formado por uma sequência de caracteres (*string*) UTF-8, que é usada pelo *broker* para identificar as mensagens e direcioná-las para cada cliente

[11] Livre tradução de: “*The easiest way to understand topic names in MQTT is to think about them like paths in a file system*”.

subscrito. Pode-se utilizar qualquer caractere UTF-8 para criação de um tópico, mas para evitar problemas futuros recomenda-se evitar os sinais de mais (+) e o símbolo # (Sharp ou Hash)[12] pois são dois caracteres coringas no MQTT (HILLAR, 2107).

Os tópicos são *case-sensitive*, ou seja, um tópico Sapiens/Unidade01 é diferente do tópico Sapiens/unidade01.

A seguir seguem algumas recomendações para a criação de tópicos[13]:

a) O caractere $ (cifrão) não deve ser utilizado no início da *string* para criação de um tópico, pois muitos servidores MQTT reservam os tópicos que iniciam com este símbolo para publicação de dados estatísticos. Mesmo não existindo uma padronização oficial é prática comum utilizar $SYS/ para publicação dessas informações estatísticas. Alguns exemplos são indicados no sítio eletrônico www.mosquitto.org tais como:
   - $SYS/*broker/bytes/received* – indica o numero de bytes recebidos desde o início do uso do *broker*;
   - $SYS/*broker/clients/total* – Indica o numero total de clientes conectados e registrados no *broker*, sendo eles ativos ou inativos;
   - $SYS/*broker/clientes/maximum* – Indica o número máximo de clientes que ao mesmo tempo conectados ao servidor.

b) A utilização de uma barra no início do tópico também não é recomendada, pois introduzirá um nível de tópico com um caractere zero na frente da *string* que pode gerar confusão para os clientes e servidores;

c) Apesar de permitido a utilização de espaços na criação dos tópicos não é recomendada, pois pode criar problemas no momento de ler ou depurar um tópico;

d) O tamanho do tópico também será um problema se ele for muito extenso e por isto recomenda-se utilizar tópicos os mais curtos e sucintos possíveis;

---

[12] *Sharp* é utilizado no conceito de música e é equivalente ao sustenido em português. *Hash* vem do nome dado ao símbolo no inglês britânico. Fonte: http://www.teclasap.com.br/hashtag/

[13] As informações foram extraídas do sítio eletrônico: https://www.hivemq.com/blog/mqtt-essentials-part-5-mqtt-topics-best-practices. Acessado em 02/01/2018.

e) Utilizar caracteres ASCII[14] UTF-8 dificulta descobrir erros de digitação ou ainda encontrar problemas relacionados a *string* pois em muitos casos eles não são exibidos de forma correta;
f) Utilizar um identificador exclusivo no tópico pode ser muito útil para identificar quem enviou uma mensagem ou ainda executar uma autorização de forma que apenas clientes com o mesmo identificador no tópico possam publicar, por exemplo, um cliente com id1 pode publicar no status client1/status, mas não pode publicar no client2/status;
g) Não esquecer que podem ocorrer ampliações na rede de sensores e a estrutura de tópicos deve ser capaz de absorve esta ampliação sem a necessidade de mexer na estrutura principal;
h) Opte por tópicos específicos ao invés de tópicos genéricos. No caso da rede de medidores de água do Sapiens Parque pode-se utilizar Sapiens/Unidade01, Sapiens/Unidade02 etc, ao invés de Sapiens/Unidades.

2.3.5 Caracteres coringas (+) e (#)

Os caracteres “+” e “#” são utilizados pelos clientes para criação de filtros de tópicos para facilitar a inscrição junto aos servidores.

O símbolo “+” é usado para receber informações relacionadas a um item específico com origem em vários equipamentos. Ao invés de se inscrever nos seguintes tópicos Sapiens/Unit001/potable, Sapiens/Unit002/potable, Sapiens/Unit003/potable até o tópico Sapiens/Unit257/potable, podemos utilizar o “+” da seguinte forma Sapiens/+/potable. Se qualquer um dos medidores das 257 unidades do Sapiens Parque publicar alguma mensagem com o final “potable” os subscritos receberão a mensagem.

[14] Do inglês *American Standard Code for Information Interchange* – Código Padrão Americano para Intercâmbio de Informação – livre tradução.

Já o símbolo “#” é utilizado para filtros multi-nível, ou seja, para receber todas as mensagens referentes ao consumo de água potável, água de chuva e de reuso de uma das 257 unidades do Sapiens Parque o usuário poderá utilizar o símbolo “#”. Normalmente, a definição dos tópicos para receber cada uma destas mensagens seria Sapiens/Unit01/potable, Sapiens/Unit01/rain e Sapiens/Unit01/reused, mas ao utilizar o “#” a inscrição nos tópicos ficaria Sapiens/Unit01/#.

O uso dos coringas “+” e “#” deve ser feito com cuidado para evitar subscrições desnecessárias em uma grande quantidade de tópicos sem realmente conhecer a existência deles. Evitando a subscrição em tópicos que não são de interesse para o cliente evita o desperdício na utilização da banda de dados e o uso desnecessário do servidor MQTT (HILLAR, 2017).

2.4 Servidor MQTT Eclipse Mosquitto

Segundo o sítio eletrônico da Fundação Eclipse[15] o servidor MQTT Eclipse Mosquitto é um servidor de mensagens de código aberto licenciado (EPL/EDL)[16] pela Fundação Eclipse que implementa o protocolo MQTT.

O Mosquitto fornece uma implementação leve de servidor com protocolo MQTT que é compatível para diversas situações, que vão desde máquinas complexas até equipamentos embarcados de baixa potência e baixo poder de processamento. Além de aceitar conexões de aplicativos do cliente MQTT, o servidor Mosquitto possui um sistema de “*bridge”* que permite conectar-se com outros servidores MQTT em outras instâncias Mosquitto, possibilitando que as redes de servidores sejam construídas, permitindo o envio de mensagens de qualquer local na rede para outra qualquer, dependendo da configuração dessas pontes.

A licença EPL é uma licença de software de código aberto usada pela Fundação Eclipse que substitui a licença pública comum (CPL[17]) e permite ao usuário

[15] http://www.eclipse.org/org/ - acesso em 03/12/2017.

[16] *Eclipse Public License* e *Eclipse Distribution License* respectivamente

de programa licenciado usar, modificar, copiar e distribuir o trabalho e as versões modificadas, sendo que em alguns casos estes usuários devem liberar as suas próprias versões alteradas.

A EDL é a licença de distribuição da Fundação Eclipse, a qual autoriza os usuários redistribuírem e usarem nas formas originais ou binárias, com ou sem modificação, desde que atendidas algumas condições específicas[18].

Ainda de acordo com sítio eletrônico da Fundação Eclipse, o Mosquitto é um projeto de alto nível desta fundação que, como princípio não contém código próprio, mas hospeda uma grande variedade de projetos que cobrem uma diversidade de áreas temáticas. Estes projetos tendem a ter ciclos de vida limitados e que ao final do ciclo, alguns se transformam em artigos e outros são incorporados como tecnologia de base para outros projetos da Fundação Eclipse.

2.5 Python e o MQTT

O interpretador *Python* surgiu no início da década de noventa quando o programador holandês Guido van Rossum tentava resolver algumas deficiências de uma linguagem de programação chamada ABC[19] (PYTHON BRASIL, 2018).

De acordo com o sítio eletrônico da *Python* Brasil[20],

> *Python* é uma linguagem de altíssimo nível (*VHLL - Very High Level Language*), de sintaxe moderna, orientada a objetos, interpretada via *bytecode*, com tipagem forte (não há conversões automáticas) e dinâmica (não há declaração de variáveis e elas podem conter diferentes objetos), modular, multiplataforma, de fácil aprendizado e de implementação livre (PYTHON BRASIL, 2018).

---

[17] CPL – Common Public License - Licença pública comum – tradução nossa.

[18] Informações contidas no sítio eletrônico: https://www.eclipse.org/org/documents/edl-v10.php acesso em 02/01/2018

[19] https://homepages.cwi.nl/~steven/abc/ acesso em 03/01/2018

[20] http://python.org.br/ acesso em 26/12/2017

"O *Python* possui uma sintaxe clara e concisa, que favorece a legibilidade do código fonte, tornando a linguagem mais produtiva" (BORGES, 2010, P.13).

Além das diversas estruturas de alto nível como listas, dicionários, entre outras, possui também grande quantidade de módulos prontos para uso. O *Python* dispõe de recursos semelhantes a outras linguagens modernas de programação, suporta programação modular e funcional além da programação orientada a objetos.

Segundo Borges (2010, p.17), "o interpretador *Python* pode ser usado de forma interativa, na qual linhas de códigos são digitadas em um *prompt* (linha de comando) semelhante ao *Shell* do sistema operacional."

Com uma tipagem dinâmica, onde o tipo de variável é inserido pelo interpretador no tempo da execução, possui ainda diversas ferramentas para desenvolvimento, tais como IDEs, editores e *shells*.

O servidor Mosquitto possui uma versão *Python* o qual foi utilizada no desenvolvimento desta dissertação e para uso desta versão houve a necessidade de instalação por meio do pacote "paho" utilizando o comando "pip install paho-mqtt" no *prompt* de comando do Windows que pode ser visualizado na figura 18.

Figura 18 – Instalação do pacote paho-mqtt

Fonte: Elaboração própria

3 MEDIÇÃO DE CONSUMO DE ÁGUA

Nesta dissertação abordasse a necessidade de medição de consumo de água do empreendimento Sapiens e com isto percebe-se que é importante identificar como podem e quais são os métodos disponíveis para realizar essas medições.

A medição de líquidos só é possível de ser realizada de forma indireta e para isto os equipamentos possuem sistemas de conversões baseados, ou na medição da velocidade do fluído através de uma área de seção transversal conhecida ou na adição de unidades de volumes conhecidos em intervalos de tempos. Esses sistemas de conversões são divididos em dois elementos básicos: o primário e o secundário.

O elemento primário é a parte do equipamento que se encontra em contato com o fluido e o secundário é que faz a conversão das informações do elemento primário em leituras por meio de mostradores analógicos ou digitais.

Segundo Frangipani (2007, p.19), os medidores podem ser agrupados nas seguintes famílias:

a) Medidores deprimogênios;
b) Medidores eletrônicos;
c) Medidores velocimétrico;
d) Medidores volumétricos;
e) Medidores de canal aberto

3.1 Medidores deprimogênios

Os medidores deprimogênios trabalham sob o conceito da variação de pressão descrito no princípio de Bernoulli[21].

[21] Daniel Bernoulli importante matemático, físico e professor suíço que desenvolveu o princípio da hidrodinâmica. Fonte: http://www-history.mcs.st-andrews.ac.uk/Biographies/Bernoulli_Daniel.html acesso em 17/01/2018.

Os elementos primários para este tipo de medidores podem ser compostos por placas de orifício, bocais, tubos de Venturi ou tubos de Pitot, conforme podem ser visualizados na Figura 19. Estes dispositivos têm a função de provocar uma diferença de pressão nos dois lados do elemento durante a passagem do fluído que é proporcional a velocidade do fluído a ser medido. Essa variação de pressão é captada pelo elemento secundário que converte em sistemas de leituras possíveis de serem visualizados pelos usuários.

Figura 19 – Elementos primários para medidores deprimogênios

Fonte: Adaptado de www.flowmaster.com.br (2018)

3.2 Medidores eletrônicos

Os elementos primários dos medidores eletrônicos possuem dispositivos que transformam proporcionalmente os volumes medidos em impulsos elétricos e estes, por meio de circuitos adequados, são transformados em leituras numéricas que são disponibilizadas em mostradores para os usuários.

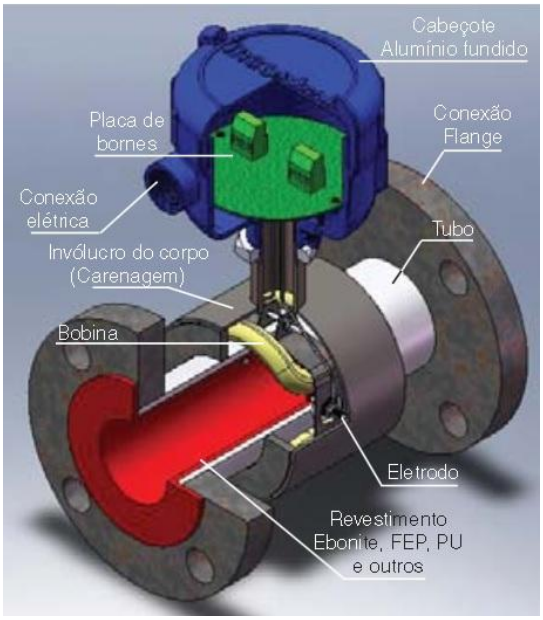
Por basear-se em sistemas eletrônicos, existem diferentes modelos de elementos primários com princípios de funcionamento variados dentro dos quais se destacam: Medidores magnéticos, medidores ultrassônicos e medidores tipo vórtice.

3.3 Medidores magnéticos

Os medidores magnéticos utilizam o princípio da lei de Faraday da indução magnética. Dois eletrodos inseridos no tubo estão em contato com o fluído a ser medido. A região dos eletrodos está sob a ação de um campo magnético, que polariza o líquido. Quanto maior a velocidade do líquido polarizado, maior a variação de campo e consequentemente maior será a tensão induzida que é captada pelos eletrodos. Um circuito eletrônico converte os valores de tensão em unidade de volume.

As partes componentes de um medidor magnético podem ser visualizadas na figura 20.

Figura 20 – Vista em corte de medidor magnético



Fonte: www.incontrol.ind.br (2018)

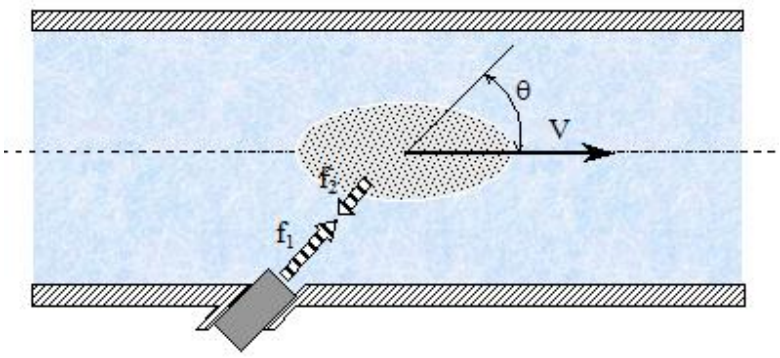
3.4 Medidores ultrassônicos

Os medidores ultrassônicos trabalham com dois princípios diferentes, o efeito Doppler[22] e o de medição de tempo de trânsito.

Os medidores de efeito Doppler baseiam-se na variação da frequência dos pulsos sonoros que ocorre durante a reflexão no fluído a ser medido.

No tubo por onde passa o fluído a ser medido é colocado um sensor que emitem sinais sonoros. O tempo de reflexão desses sinais é proporcional a velocidade do fluído que passa por dentro deste tubo. Desta forma é possível calcular o volume medido entre os pulsos. A figura 21 apresenta o modelo de funcionamento da medição pelo efeito Doppler.

Figura 21 – Princípio de medição de líquido por efeito Doppler



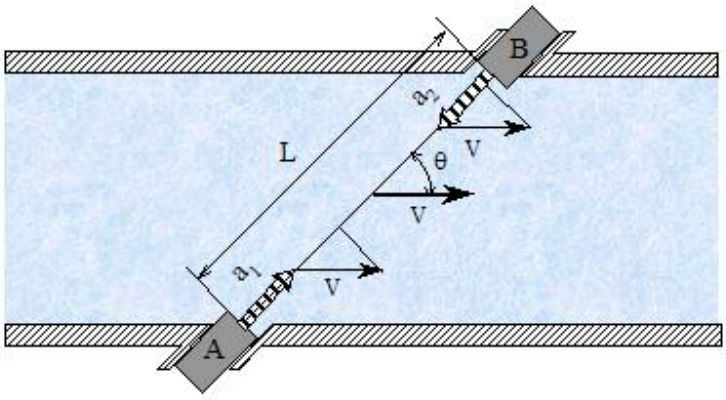
Fonte: www.consulteengenheiroeletronico.wordpress.com (2018)

[22] A aparente diferença entre a frequência em que as ondas de som ou de luz deixam uma fonte e aquela em que atingem um observador, causada pelo movimento relativo do observador e pela fonte de onda, foi descrita pelo físico austríaco Christian Andreas Doppler. Fonte:https://www.britannica.com acesso em 19/01/2018

Os medidores de tempo de trânsito trabalham com a diferença de tempo entre sinais sonoros nos dois percursos entre dois sensores montados de forma obliqua na tubulação.

Em um dos sentidos, o sinal sonoro será propagado de forma mais rápida, pois uma das componentes cartesianas do movimento está no mesmo sentido do movimento do fluído a ser medido. No sentido inverso, esta componente encontrará maior resistência na propagação. A diferença de tempos entre os dois sinais indica o volume que passou durante esta variação de tempo. A figura 22 mostra o modelo de funcionamento da medição por tempo de transição.

Figura 22 – Princípio de medição de líquido por medição de tempo de trânsito



Fonte: www.consulteengenheiroeletronico.wordpress.com (2018)

3.5 Medidores tipo vórtice

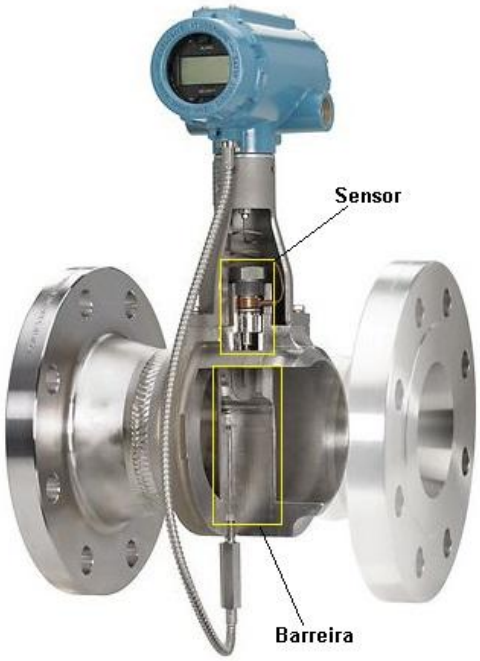
Estes medidores medem a velocidade do fluído com base na formação de vórtices quando este fluído passa por uma barreira instalada dentro da tubulação.

Um sensor instalado logo após a barreira identifica a frequência de formação dos vórtices gerando pulsos elétricos a cada mudança. Esses pulsos são transmitidos para um circuito eletrônico que converte os sinais de forma adequada para o usuário.

Quanto mais rápida a passagem do fluído mais rápido é a formação de vórtices em um mesmo intervalo de tempo.

A figura 23 mostra as partes internas de um medidor tipo vórtice.

Figura 23 – Medidor tipo vórtice



Fonte: Adaptado de www.emerson.com(2018)

3.6 Medidores velocimétricos

Segundo a Associação Brasileira de Normas Técnicas (NBR NM 212; 1999), medidor velocimétrico é: “Instrumento [...], que consiste de um elemento móvel acionado pela velocidade do fluxo da água, cujo movimento é transmitido por meios mecânicos ou outros, ao dispositivo indicador que totalizam o volume.”

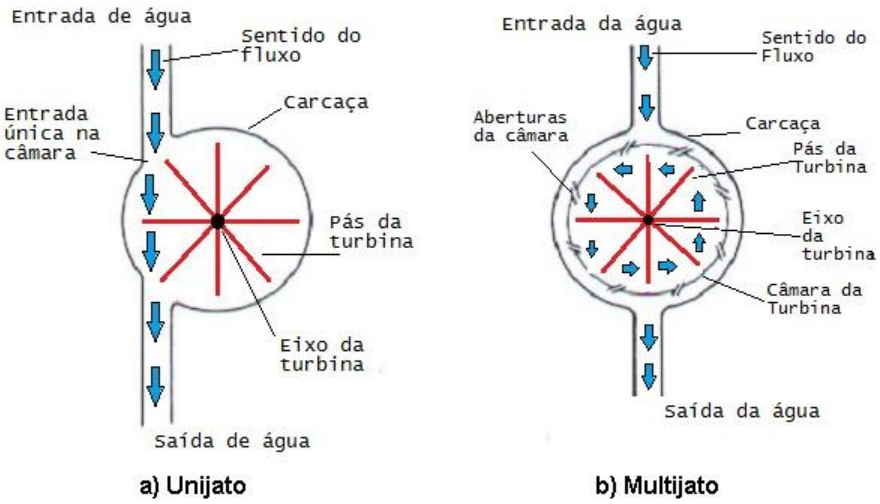
Geralmente os elementos primários dos medidores velocimétricos são compostos por pás ou turbinas montadas em posições diversas e o volume medido é proporcional a velocidade de movimento de rotação.

Frangipani (2007) cita alguns modelos de medidores velocimétricos, os quais são:

a) Medidores de jato único e múltiplos;
b) Medidores Woltman, verticais e horizontais;
c) Medidores de turbina ou hélice;
d) Medidores compostos;
e) Medidores proporcionais ou “shunt”.

Os medidores de jato único e medidores de jato múltiplo são equipamentos que possuem uma turbina ou uma hélice dentro de uma câmera onde o fluído passa tangenciando as pás desta turbina ou hélice. A diferença entre estes dois medidores está na quantidade de câmeras por onde o fluído entra para movimentar a hélice. Na figura 24 temos o diagrama de funcionamento dos medidores de jato único e jato múltiplo.

Figura 24 – Diagrama de funcionamento do medidor de jato único e multijato



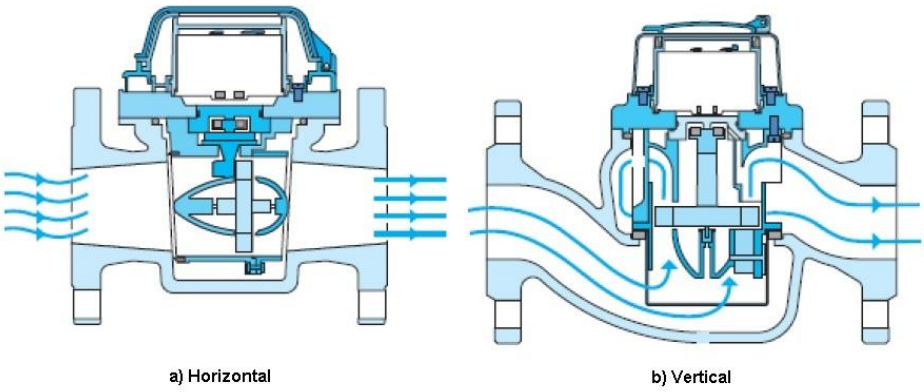
Fonte: http://saneamentocomercial.blogspot.com.br (2018)

Os medidores Woltman possuem este nome em homenagem ao engenheiro Reinhard Woltman[23], o qual inseriu uma turbina instalada dentro de um tubo fechado para medição de volumes de água.

Existem dois modelos, turbina horizontal (ou axial) e turbina vertical (perpendicular) que estão demonstrados na figura 25.

O conceito de medição está baseado na velocidade da turbina que possui uma proporcionalidade com o volume de água que passa pelo conduto fechado.

Figura 25 – Medidores tipo Woltmann

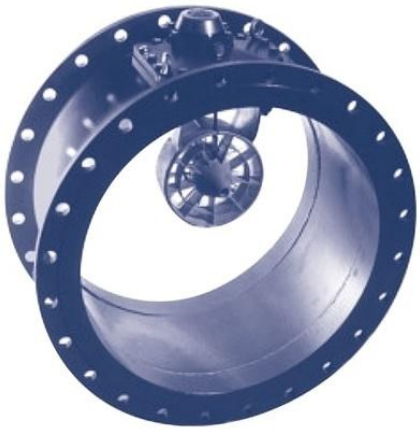


Fonte: Adaptado de Crainic (2012)

Os medidores de turbina ou hélice possuem semelhança construtiva com os de Woltman, mas com diferença na dimensão do elemento primário que não preenche toda a área útil do tubo onde é instalado. Um modelo deste medidor pode ser visualizado na figura 26.

[23]Engenheiro alemão, inventor do primeiro molinete hidráulico (1790) equipado com um tipo de turbina, a Woltman, inventada por ele, própria para medição de vazões de líquidos e gases em condutos. Fonte: http://www.dec.ufcg.edu.br/biografias/Reinhard.html acesso em 20/01/2018

Figura 26 – Medidor de turbina



Fonte: www.bermad.com.au(2018)

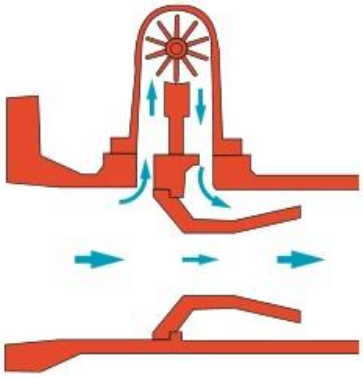
Os medidores compostos são fabricados com um medidor Woltman em paralelo com um medidor de pequena capacidade. Em grandes vazões a medição é realizada pelo medidor Woltman e em baixas vazões uma válvula fecha automaticamente desviando o fluído para o medidor de pequena capacidade. Um modelo de medidor composto pode ser visualizado na figura 27.

Figura 27 – Medidor composto

Fonte: www.bermad.com.au(2018)

Os medidores proporcionais fazem medições de volumes parciais do fluído por meio de um sistema paralelo. Na figura 28 vemos o diagrama de um medidor proporcional desviando parte do fluído para uma câmera para fazer a medição dos volumes, sendo este proporcional ao volume total que passa pela tubulação.

Figura 28 – Medidor proporcional – Diagrama de funcionamento



Fonte: www.itron.com(2018)

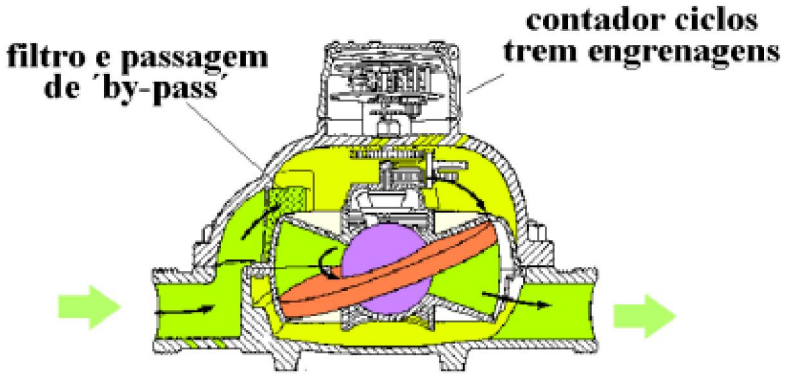
3.7 Medidores volumétricos

O princípio de funcionamento dos medidores volumétricos consiste em obter a quantidade de vezes que o fluído preenche uma câmera com volume conhecido e os modelos mais comuns são os de disco oscilante, pistão oscilante e de engrenagem.

Nos medidores de disco oscilante o elemento primário é composto por um disco com uma fenda radial, uma esfera no centro deste disco e um pino acoplado ao eixo da esfera. Quando o fluído penetra na câmera faz o disco girar e consequentemente o pino acoplado ao eixo da esfera realiza giros proporcionais ao volume do fluído.

A forma construtiva de um medidor de disco oscilante pode ser visualizado na figura 29.

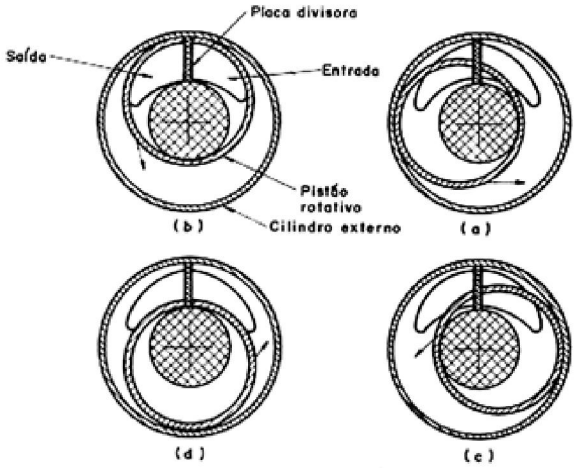
Figura 29 – Medidor de disco oscilante



Fonte: www.fem.unicamp.br (2018)

Os medidores de pistão oscilante possuem um pistão oco, com uma abertura longitudinal, que gira no eixo do cilindro de forma excêntrica. Em operação, a cada volta a abertura do pistão recebe um volume de fluído conhecido até ficar cheio e no movimento seguinte de rotação transfere o volume para outra área da câmera liberando espaço para o pistão continuar a coleta de fluído. A sequência de funcionamento de um medidor de pistão oscilante pode ser visualizada na figura 30.

Figura 30 – Medidor de pistão oscilante - fases



Fonte: www.consulteengenheiroeletronico.wordpress.com (2018)

Nos medidores de engrenagem, um conjunto de engrenagens está disposto tal que o fluxo do fluído promova a movimentação deste conjunto, sendo que o número de rotações é proporcional ao volume medido. Dois modelos de medidores de engrenagem podem ser visualizados na figura 31.

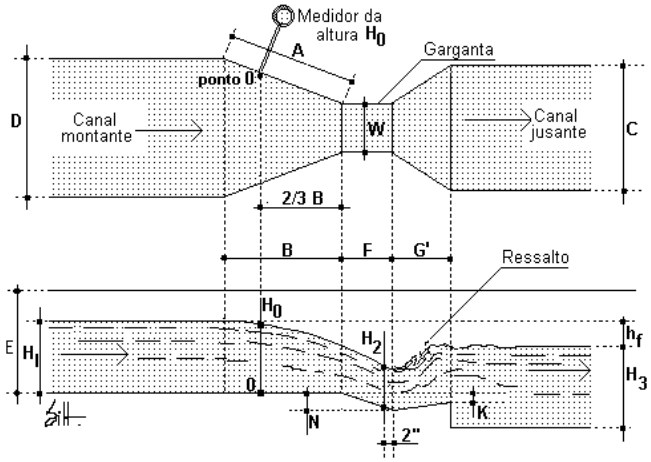
Figura 31 – Medidores de engrenagens

Fonte: www.digiflow.com.br(2018)

3.8 Medidores de canal aberto

Entre os modelos mais conhecidos dos dispositivos de medição em canal aberto é o medidor Parshall. Desenvolvido pelo engenheiro americano Ralph Leroy Parshall, é um método no qual se assemelha ao tubo de Venturi, mas trabalha com os valores de altura dos fluídos a montante e a jusante do canal. As figuras 32 e 33 mostram respectivamente o diagrama de funcionamento do modelo do medidor Parshall e uma calha Parshall em funcionamento.

Figura 32 – Esquema de uma calha Parshall convencional



Fonte: www.dec.ufcg.edu.br (2018)

Figura 33 – Calha Parshall

Fonte: www.saaeitauna.com.br(2018)

4 HARDWARE PARA DESENVOLVIMENTO DA PESQUISA

Neste capítulo apresentam-se o modelo de hardware utilizado na proposta de solução, quais os parâmetros utilizados para a escolha e algumas particularidades do equipamento.

Entende-se que a escolha de um equipamento para aplicar em medição remota sem fio não é uma tarefa simples quando não se tem uma lista de requisitos, pois existe uma grande diversidade de produtos disponíveis no mercado.

Em um contexto de indefinição de padrões a serem adotados para medição remota de consumo de água no sistema brasileiro e com base nas pesquisas bibliográficas realizadas e apresentadas no capítulo dois, adotou-se como referencia o padrão utilizado na Europa.

O padrão europeu utiliza as frequências subgiga hertz para comunicação entre equipamentos de medição em conjunto com protocolo WM-Bus.

Em pesquisa realizada acerca dos equipamentos de comunicação por radio frequência que possuem embarcado o protocolo WM-Bus chegou-se ao módulo de desenvolvimento *CC1200DK da Texas Instruments* que é uma plataforma desenvolvida para testes de desempenho de hardware e desenvolvimento de softwares para dispositivos subgiga hertz.

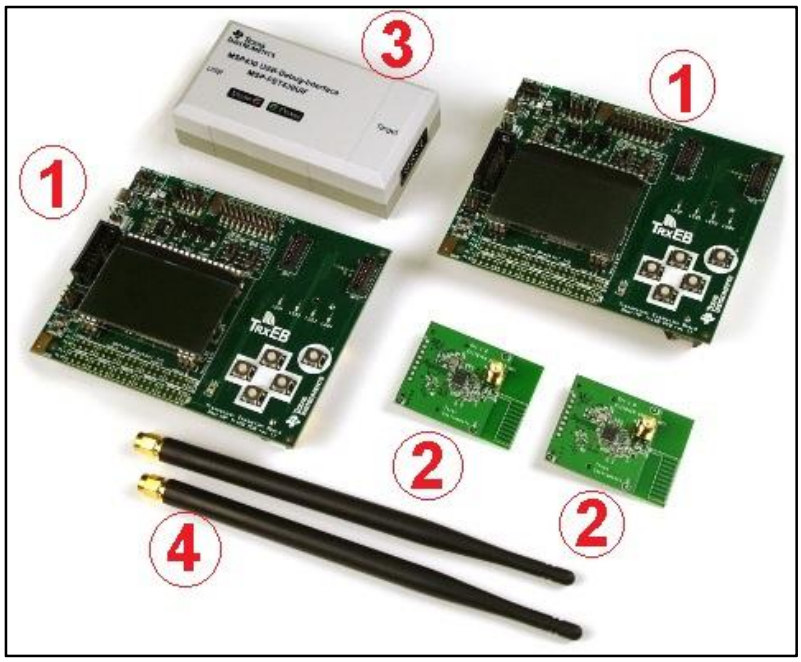
O módulo já possui um pré-programa de teste instalado, que com ele consegue-se realizar rápidas avaliações do dispositivo RF com diferentes configurações de pacote de dados, taxas de transferências e potências de transmissões.

No módulo estão inclusos:

a) Duas placas TrxEB – SmatRF *Trasnceiver Evaluation Board;*
b) Duas placas CC1200 *Evaluation Module Kit* 868-930Mhz;
c) Um depurador MSP-FET430UIF para MSP430;
d) Duas antenas para transmissão RF.

Os itens descritos acima podem ser visualizados na figura 34.

Figura 34 – Módulo de desenvolvimento CC1200DK - Componentes



Legenda:

1 - Placas TrxEB – SmatRF *Trasnceiver Evaluation Board;*

2- Placas CC1200 *Evaluation Module Kit* 868-930Mhz

3 - Depurador MSP-FET430UIF para MSP430

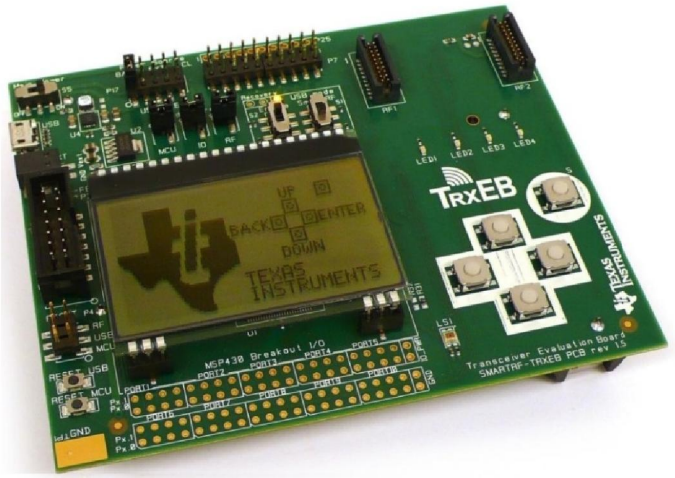
4 - Antenas para transmissão RF

Fonte: *SmartRF Transceiver Evaluation Board "TrxEB" user's guide*

4.1 Placa TrxEB – SmartRf Evaluation Board modelo MSP430F5438A

A placa TrxEB que pode ser vista na figura 35, atua com placa-mãe para diversos kits de desenvolvimento para circuitos integrados de baixa potência da *Texas Instruments* (TI, 2012). Nela podemos encontrar diversas interfaces de conexões que permitem a prototipagem rápida bem como testes de *hardwares* e *softwares*.

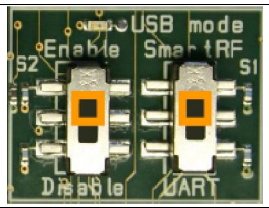
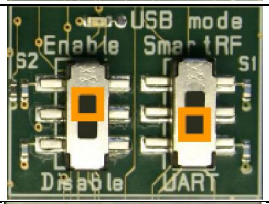
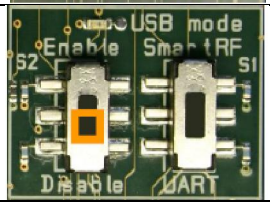
Figura 35 – SmartRF Transceiver Evaluation Board modelo MSP430F5438A



Fonte: *SmartRF Transceiver Evaluation Board "TrxEB" user's guide*

Possui três modos de operações que são selecionados por meio das chaves "S1" e "S2". As possíveis configurações dos modos de operações são mostradas na tabela 02.

Tabela 02– Configurações dos modos de operação do MSP430F5438A

Chaves	S1	S2	Modo de operação
	Enable	SmartRF	SmartRF
	Enable	UART	UART
	Disable	x	Disable

Fonte: *SmartRF Transceiver Evaluation Board "TrxEB" user's guide*

4.2 Microcontroladores da família MSP430

A família dos microcontroladores MSP430 da *Texas Instruments* é constituída por vários dispositivos com diferentes conjuntos de periféricos destinados a varias aplicações. Desenvolvido como dispositivo de ultrabaixa potência, é projetado para proporcionar uma vida útil prolongada da bateria quando utilizado em aplicações remotas.

A série MSP430F5438A[24] possui microcontroladores com três temporizadores de 16bits, um ADC[25] de alto desempenho de 12 bits, quatro interfaces de comunicação serial universal (USCIs), um multiplicador de hardware de acesso direto à memória, um módulo RTC[26] com recurso de alarme e até 87 pinos de entradas e saídas.

Aplicações típicas incluem a utilização em sensores analógicos e digitais, controles remotos de dispositivos e medidores portáteis.

As principais características desta família de microcontroladores são:

a) Ultrabaixo consumo de energia;
b) 256 KB de memória não volátil;
c) Arquitetura RISC de 16 bits, memória estendida e *clock* do sistema de até 25 MHz;
d) Sistema de gerenciamento de energia flexível, com supervisão e monitoramento de flutuações de tensão;
e) Sistema de *clock* unificado, com fonte de *clock* interno de baixa potência (VLO), fonte de referência interna de baixa frequência (REFO), cristal de 32MHz;
f) Temporizadores de 16 bits;
g) Até quatro interfaces de comunicação serial (UART, SPI síncrono, I2C);

---

[24] Disponível em http://www.ti.com/product/MSP430F5438A

[25] Analog-to-digital converter

[26] *Real-Time Clock* – Relógio em tempo real (livre tradução)

h) Conversor analógico digital de 12 bits;

i) Temporizador básico com recurso RTC.

O microcontrolador MSP430F5438A pode ser visualizado na figura 36.

Figura 36 – Microcontrolador MSP430F5438A

Fonte:http://www.ti.com/product/MSP430F5438A

4.3 Evaluation module kit CC1200

O módulo de avaliação CC1200 possui um transceptor de rádio CC1200 projetado para operar junto a um micro controlador e em ultrabaixa tensão em sistemas sem fio, destinado principalmente para atender as frequências ISM[27] e também dispositivos SRD[28] na faixa entre 164-190MHz, 410-475MHz e 820-950MHz.

O CC1200 pode automatizar várias tarefas relacionadas à RF comuns podendo assim aliviar muito o micro controlador.

A interface de conexão com a *Smart Transceiver Evaluation Board* é realizado por meio de dois conectores de 20 pinos já acoplados nas placas de circuito impresso.

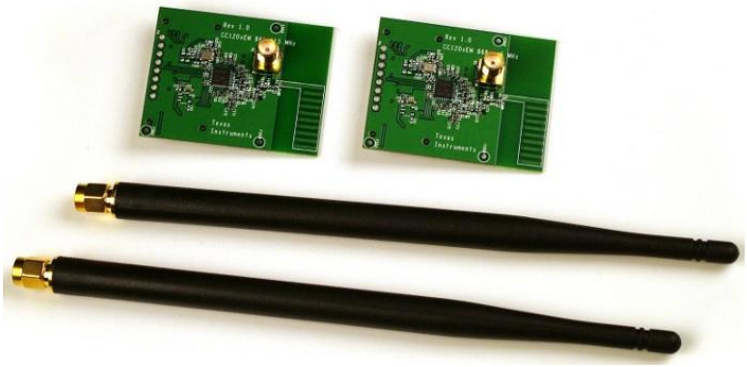
Alguns pinos de entrada e saída do CC1200 estão conectados às portas de interrupção do MSP430F5438A e também a porta de comunicação serial. Se não for

[27] ISM – Industrial, Scientific and Medical – A frequência de 868MHz é padrão ISM na Europa.
[28] SRD – Short Range Device

necessário a utilização de um periférico serial os pinos podem ser utilizados para outra aplicação de forma geral.

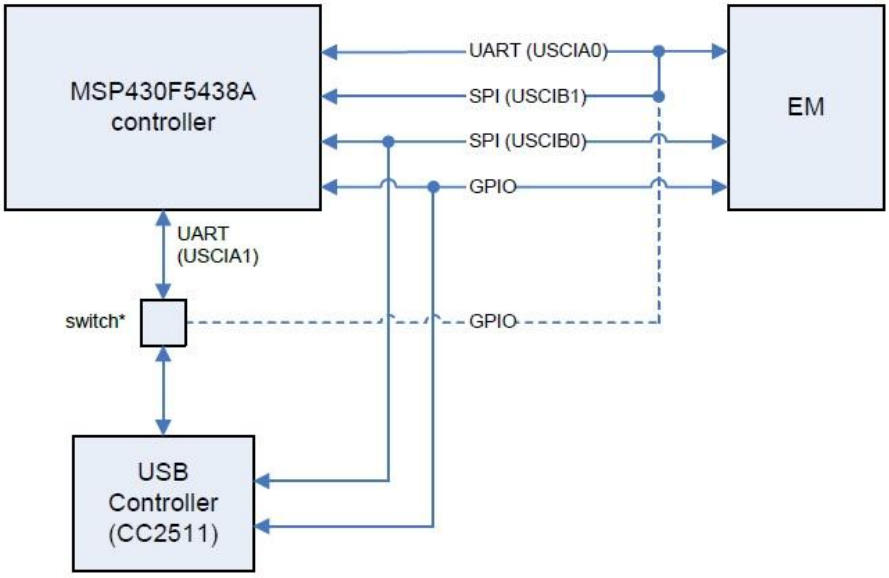
A figura 37 mostra dois conjuntos do transceptor de rádio CC1200 e a figura 38 o diagrama de conexão entre CC1200 e o microcontrolador MSP430F5438A.

Figura 37 – CC1200 *Evaluation Module Kit*



Fonte: *CC120x Evaluation Module Kit Quick Start Guide*

Figura 38 – Evaluation module Interface



Fonte: *SmartRF Transceiver Evaluation Board "TrxEB" user's guide*

5 COMUNICAÇÃO POR RÁDIO FREQUÊNCIA E WM-BUS

Historicamente, o início da era das comunicações sem fio pode ser atribuído a Guglielmo Marconi[29] quando em 1894 realizou a primeira transmissão de dados por meio da rádio frequência.

Em seu celeiro transformado em laboratório e sob influência das teorias de Maxwell, com base nas experiências de Hertz sobre o movimento das ondas eletromagnéticas e a utilização do tubo coesor desenvolvido por Onesti[30], Marconi transmitiu a letra “S” em código Morse através do ar em uma distância de poucas centenas de metros.

Desde então vários esforços foram realizados para aprimorar os sistemas de comunicação sem fio e hoje temos várias tecnologias em uso.

Quando se fala em comunicação não ficamos restritos a fala humana propriamente dita, mas sim qualquer interação entre um sinal de uma fonte emissora e uma resposta de um receptor. Dentro desta lógica podemos ter como exemplos: a abertura de um portão de garagem e a troca de canal de uma televisão por meio de um controle remoto, envio de leituras de sensores instalados em locais remotos de difíceis acessos a unidades de controle em grandes centros, conversa por meio de telefones celulares etc.

O espectro de radiação eletromagnética apresenta faixas de frequência com aplicações diversas e como exemplos podemos citar: o raio-X para diagnósticos médicos, raios ultravioletas utilizado em tratamento de esgoto sanitário, infravermelho na indústria de segurança patrimonial e no funcionamento de controles remotos, raios

---

[29] Guglielmo Marconi físico e inventor italiano que recebeu o prêmio Nobel de física em 1909 por desenvolver a telegrafia sem fio. Fonte: www.nobel-winners.com/Physics/guglielmo_marconi.html acesso em 21/01/2018

[30] Temistocle Calzecchi-Onesti foi um físico e inventor italiano que inventou o coesor, um tubo isolado contendo limalhas de ferro que conduziam eletricidade sob a ação de uma onda eletromagnética. Fonte: http://www.browsebiography.com/bio-temistocle_calzecchi_onesti.html acesso em 21/01/2018

gama como fonte primária de esterilização de equipamentos médicos e radiofrequência para comunicação e envio de dados.

Dentro da faixa de radiofrequência, de acordo com Parson (2000, p.4) apesar se propagarem em frequências de alguns poucos quilohertz, as frequências das ondas de rádio estão entre 30 kHz até 300 GHz.

Ainda segundo Parson (2000, p4) um tratado internacional divide o espectro de radiofrequência em bandas conforme tabela 03 a seguir:

Tabela 03– Designação debandas de frequência

Tipo de frequência	Frequências (range)
Extremamente baixa	<3KHz
Muito baixa	3 a 30 KHz
Baixa	30 a 300 KHz
Média	300 KHz a 3 MHz
Alta	3 a 30 MHz
Muito alta	30 a 300 MHz
Ultra-alta	300 MHz a 3 GHz
Super alta	3 a 30 GHz
Extra-alta	30 a 300 GHz

Fonte: Adaptado de Parson (2000)

No Brasil, a Agência Nacional de Telecomunicações – ANATEL é a responsável pela regulação do uso das faixas de radiofrequência e ela está regulamentada pela lei 9.472/1997, onde no artigo 158 especifica:

> "[...] a Agência manterá plano com a atribuição, distribuição e destinação de radiofrequências, e detalhamento necessário ao uso das radiofrequências associadas aos diversos serviços e atividades de telecomunicações, atendida duas necessidades específicas e as de suas expansões." (BRASIL, 1997)

Os equipamentos utilizados na proposta de solução desta dissertação operam na faixa de 868 a 930 MHz e de acordo com a resolução 680/2017 da ANATEL estes dispositivos se enquadram na classificação de radiação restrita.

O artigo 62 desta mesma resolução cita que: “[...] à prestação de serviços de interesse coletivo que utilizarem exclusivamente equipamentos de radiocomunicação de radiação restrita [...] são dispensados de licenciamento. (NR)” (ANATEL, 2017).

Os equipamentos de radiação restrita, na sua grande maioria podem ser utilizados sem a autorização da ANATEL para uso de radiofrequências, desde que não sejam usados como serviços de telecomunicação para terceiros e nem causem interferência em qualquer sistema operando em caráter primário ou secundário.

O Wireless Meter-Bus ou WM-Bus é o padrão adotado pela comunidade européia para leitura inteligente de medidores utilizados pelos operadores de serviços públicos, tais como água, eletricidade e gás. Utiliza as frequências de 169 MHz, 434 MHz e 868 MHz, que são as faixas livres de licença na Europa que consequentemente proporcionam menores custos para as operadoras de serviços públicos. Além disto, também possuem melhor propagação de ondas em locais de difícil acesso quando comparadas a frequência de 2,4GHz (MOHAN, 2015, p.14).

Dos documentos que estabelecem o uso da rádio frequências na Europa, o EN13757 é o que orienta o padrão WM-Bus. Publicado pelo *European Committee for Standardization*, o EN13757 é dividido em seis partes conforme a seguir[31];

a) EN13757-1 - Sistema de comunicação para medidores e leitura remota de medidores – Parte 1: Troca de dados;
b) EN13757-2 - Sistema de comunicação para medidores e leitura remota de medidores – Parte 2: Camada física e de conexão;
c) EN13757-3 - Sistema de comunicação para medidores e leitura remota de medidores – Parte 3: Camada de aplicação dedicada;
d) EN13757-4 - Sistema de comunicação para medidores e leitura remota de medidores – Parte 4: Leitura do medidor sem fio (Leitura do medidor de rádio para operação em bandas SRD);

[31] Livre tradução da lista de normas obtidas em https://standards.cen.eu/dyn/www/f?p=204:105:0. Acesso em 04/04/2018.

e) EN13757-5 - Sistema de comunicação para medidores – Parte 5: Retransmissão sem fio;
f) EN13757-6 - Sistema de comunicação para medidores – Parte 6: Barramento local.

Segundo o EN13757-4, que define os parâmetros das camadas físicas e de enlace para sistemas que utilizam rádio frequência para leitura de medidores remotos, com foco nas faixas de telemetria não licenciadas (SRD) de 868 MHZ a 870 MHz os modos de operação para dispositivos SRD são: S, T, C e R2.

O modo “S” ou estacionário é utilizado para comunicação uni ou bidirecional entre dispositivos estacionários ou móveis.

O modo “T” ou de transmissão frequente permite o medidor trabalhar com sistemas *walk-by* ou *drive-by*, pois envia constantemente as leituras em pequenos intervalos de tempos.

O modo “C” ou compacto é semelhante ao modo “T”, mas envia mais informações com a mesma energia.

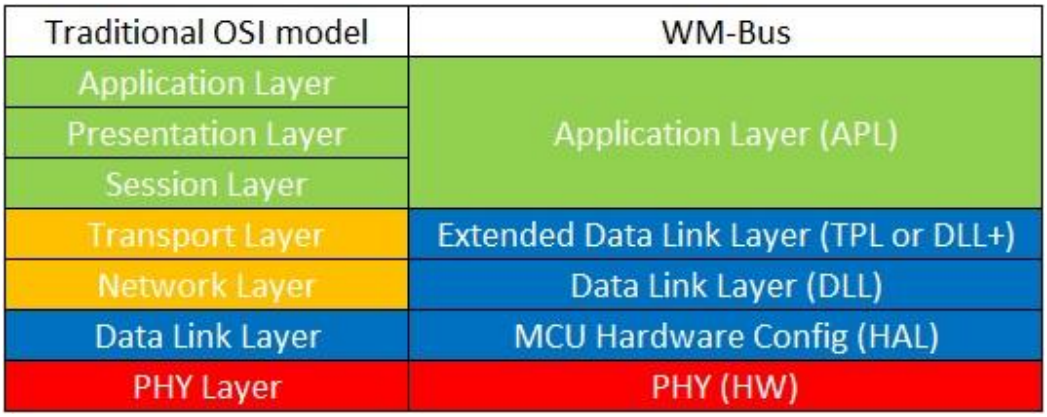
E os medidores que trabalham com sinais de *wakeup*, são dispositivos que operam no modo R2. Neste modo, os aparelhos entram em estado de recepção a espera do sinal de “acordar”. Após este sinal o medidor se prepara para a comunicação e recepção dos dados.

O conjunto CC1200DK, utilizado na proposta de solução, pode trabalhar nos modos de operação S, T e R2 definidos pela EN13757 e está em conformidade para integrar a solução de medição remota de consumo de água do Sapiens Parque segundo as definições da ANATEL.

O WM-Bus se popularizou na Europa devido a sua simplicidade, pois opera em formato de rede em estrela sem necessidade de IP e em frequências subgiga Hertz, oferecendo maior alcance e com mínimo tamanho de pilha de software e menor consumo de bateria (MOHAN, 2015, p.15).

Comparado ao modelo OSI, a arquitetura de pilhas do WM-Bus é um pouco diferente, pois possui menos requisitos nas suas camadas (MOHAN, 2015, p.16). A figura 39 mostra uma comparação entre a pilha WM-Bus e o modelo OSI.

Figura 39 – Comparação entre o modelo OSI e WM-Bus



Fonte: Adaptado de Mohan (2015)

Dependendo do modo e do tipo de dispositivo, a pilha pode ser implementada com menos de 32 KB de memória flash o que induz a utilização de um microcontrolador de baixo custo. A camada de aplicação é definida pelo próprio usuário

6 DEFINIÇÃO DO PROBLEMA

Neste capítulo apresenta-se o local de aplicação da pesquisa demonstrando quais as suas necessidades com relação à medição do consumo de água e as restrições impostas, tanto pela característica do empreendimento como pelos órgãos reguladores e ambientais.

6.1 Empreendimento Sapiens Parque

O Sapiens Parque é um parque de inovação tecnológica localizado no bairro da Cachoeira do Bom Jesus, junto à praia de Canasvieiras no norte da ilha de Santa Catarina. Distante a 20 quilômetros do centro de Florianópolis, é um ambiente projetado para atrair empreendimentos e promover o desenvolvimento sustentável, sócio econômico e ambiental da região. A distância entre o Sapiens Parque e o centro de Florianópolis bem como sua localização no norte da ilha podem ser visualizadas respectivamente nas figuras 40 e 41.

Figura 40 – Distância do Sapiens Parque ao centro de Florianópolis

Fonte: www.google.com/maps - Editado pelo autor

Figura 41 – Localização do Sapiens Parque

Fonte: Elaboração própria

Com extensão territorial maior que a região central de Florianópolis, a qual pode ser visualizada na figura 42, o Sapiens Parque tem previsão para receber mais de 400 empresas distribuídas em 1,3 milhões de metros quadrados em edificações a serem construídas e também a previsão de criação de mais de 30 mil postos de trabalhos diretos e entre 20 a 40 mil postos indiretos.

Figura 42 – Comparação área Sapiens e região central Florianópolis

Fonte: Elaboração própria

Por possuir características semelhantes à de um condomínio, o acesso aos serviços públicos de água, energia elétrica e comunicação seguirão regras específicas.

O fornecimento de energia elétrica e de comunicação deverá ser contratado diretamente com as empresas fornecedoras.

Para o serviço público de abastecimento de água, o empreendimento Sapiens Parque será responsável por receber e distribuir este serviço para todas as 257 unidades.

A rede de distribuição de água potável a ser instalada, que pode ser vista na figura 43, possuirá quase oito quilômetros de extensão e contará com três pontos de entrada para abastecimento e semelhante à rede de água potável, existirá outra rede em paralelo para fornecimento de água residual tratada com fornecimento direto da estação de tratamento de esgoto pertencente ao Sapiens Parque.

Figura 43 – Pontos de fornecimento de água - CASAN

Fonte: Elaboração própria

Existem regras específicas para o consumo de água e a Nota técnica 02 (SAPIENS PARQUE, 2014, p.8), inidica que o consumo de água potável fornecido pela

CASAN não poderá exceder o valor de 50% do consumo total das águas consumidas em todo o empreendimento e para complementar os outros 50%, a água de chuva e o reuso de águas residuais tratadas, deverão ser utilizadas pelas unidades consumidoras implantadas dentro do empreendimento. A proporção esperada é de 14% para água de chuva e 36% para águas residuais, podendo sofrer leve alterações.

Para cada ponto de consumo das unidades deverá ser instalado um medidor para garantir a cobrança dos valores de água potável e de reuso, bem como assegurar que os percentuais exigidos nas notas técnicas sejam atingidos. Ao final do período de implantação do Sapiens Parque espera-se existir mais 750 pontos de medição.

6.2 Medição do consumo de água

Para medir estes volumes, frente a grande extensão do empreendimento, é indispensável a utilização de um modelo diferente do método tradicional utilizado pelas companhias de água e esgoto, onde segundo Nayaka e Biradar (2015, p.13) ainda hoje, no mundo todo, a maioria das companhias de serviços públicos quer seja de água, eletricidade ou gás, continuam utilizando os serviços de leituras manuais, deslocando pessoas até o ponto de consumo para obter as informações.

O tempo estimado para efetuar a coleta de dados de forma manual em todas as unidades do Sapiens Parque será de aproximadamente 12 horas, ou seja, mais de um dia de trabalho de uma pessoa, se todos os medidores estiverem agrupados em locais de fácil acesso em cada unidade consumidora. Como a medição de água de chuva estará em local separado, muito provavelmente dentro da própria edificação, este tempo pode variar entre 32 a 53 horas, ou seja, para acessar a informação do consumo em todas as edificações pode-se levar de quatro a sete dias. Nas tabelas 04 e 05 podemos ver respectivamente o resumo dos tempos mínimos e máximos estimados para leitura dos medidores de água.

Tabela 04 – Tempo estimado mínimo de leitura manual

Tempo de leitura dos medidores - Potável e residual				
Qtd Unid	Medidor/ unidade	Total medidores	Tempo leitura (s)/relógio	Tempo leitura (h)
257	2	514	15	2,1

Tempo de leitura dos medidores - Chuva				
Qtd Unid	Medidor/ unidade	Total medidores	Tempo leitura (min)/relógio *	Tempo leitura (h)
257	1	257	5	21,4

* Considerado tempo mínimo de acesso ao relógio dentro da edificação pois a maioria dos relógios estarão instalados no subsolo

Tempo de deslocamento entre unidades			Tempo estimado de leitura	
Qtd Unid	Desl. Médio entre unidades (min)	Desl. total entre unidades (h)	Horas	Dias
257	2	8,6	32,1	4,0

Fonte: Elaboração própria

Tabela 05 – Tempo estimado máximo de leitura manual

Tempo de leitura dos medidores - Potável e residual				
Qtd Unid	Medidor/ unidade	Total medidores	Tempo leitura (s)/relógio	Tempo leitura (h)
257	2	514	15	2,1

Tempo de leitura dos medidores - Chuva				
Qtd Unid	Medidor/ unidade	Total medidores	Tempo leitura (min)/relógio *	Tempo leitura (h)
257	1	257	10	42,8

* Considerado tempo máximo de acesso ao relógio dentro da edificação pois a maioria dos relógios estarão instalados no subsolo

Tempo de deslocamento entre unidades			Tempo estimado de leitura	
Qtd Unid	Desl. Médio entre unidades (min)	Desl. total entre unidades (h)	Horas	Dias
257	2	8,6	53,5	6,7

Fonte: Elaboração própria

Outra preocupação além do faturamento é a necessidade de verificação contínua de possíveis vazamentos na rede de distribuição para evitar o desperdício de água e de custos que não poderão ser repassados para as unidades consumidoras.

Um sistema de medição que possa monitorar o volume de água fornecido pela CASAN e ao mesmo tempo os valores de consumo das unidades pode ajudar a avaliar perdas no sistema de distribuição.

Segundo estudo divulgado pelo Instituto Trata Brasil em 2013, o índice de perdas de água no sistema de distribuição no Brasil chegou a 37% e as perdas financeiras associadas chegaram a 39%. Isto representa mais de oito bilhões por ano a menos no faturamento das concessionárias[32]

Ainda neste estudo, entre as 100 maiores cidades do Brasil em numero de habitantes, Florianópolis está na vigésima sexta posição de menor índice de perdas de água na distribuição, mas ainda bem longe do ideal, pois as perdas no município chegaram a 33,72%.

O consumo médio estimado do Sapiens Parque é de 936 mil m³/ano no final da sua implantação. Se o índice de perdas de Florianópolis for referência para trabalhar como perda na distribuição de água dentro do Sapiens Parque, os valores somados anualmente chegarão a mais R$3.260.000,00 que não poderão ser repassados às unidades consumidoras.

Pensando ainda que se as perdas forem similares ao melhor índice deste estudo, o qual pertence à cidade de Limeira com 14,46%, os valores ainda serão altos e os gastos chegariam a R$1.400.000,00/ano.

Os valores acima foram obtidos por meio da tabela de preço praticados pela CASAN a partir de agosto de 2017[33] para unidades públicas.

Os recursos financeiros que poderão ser gastos para suprir possíveis perdas no sistema de distribuição mostram a importância do acesso às leituras de forma rápida

---

[32] Estudo completo disponível em: http://www.tratabrasil.org.br/datafiles/estudos/perdas-de-agua/Relatorio-Perdas-2013.pdf. Acessado em 03/11/2017

[33] www.casan..com.br – Valores referentes à tabela de 08/2017 – acesso em 11/11/2017

e precisa e com base nisto, um sistema de monitoramento deverá fornecer informações em tempo real para rápida intervenção da equipe de manutenção.

Outro fator importante a ser considerado sobre o consumo de água é o crescimento constante da demanda em contrapartida à dificuldade de ampliação da captação e tratamento.

Segundo dados da ANA - Agência Nacional de Águas[34],

> A capacidade total dos sistemas produtores instalados e em operação no País é de, aproximadamente, 587 m³/s, bastante próxima às demandas máximas atuais (em torno de 543 m³/s), demonstrando que grande parte das unidades está no limite de sua capacidade operacional (ANA, 2010, p.67).

Dados da FAO – *Food and Agriculture Organization of the United Nations*[35] mostram que no Brasil, entre 1992 e 2014, ocorreu uma redução de 25% na disponibilidade de recursos hídricos renováveis, totalizando uma queda de mais de 14 milhões de metros cúbicos de água por ano.

Ainda de acordo com dados da ANA, o município de Florianópolis apresenta risco de fornecimento necessitando de novo manancial para atender as demandas futuras. O consumo de água em Florianópolis passou de 21,34 milhões m³/ano em 2005 para 42,62 milhões m³/ano em 2015, crescimento de quase 100%, muito maior quando comparado com o da população que foi de quase 29% passando de 365 mil habitantes em 2005 para quase 470 mil habitantes em 2015. Esses dados podem ser visualizados na figura 44.

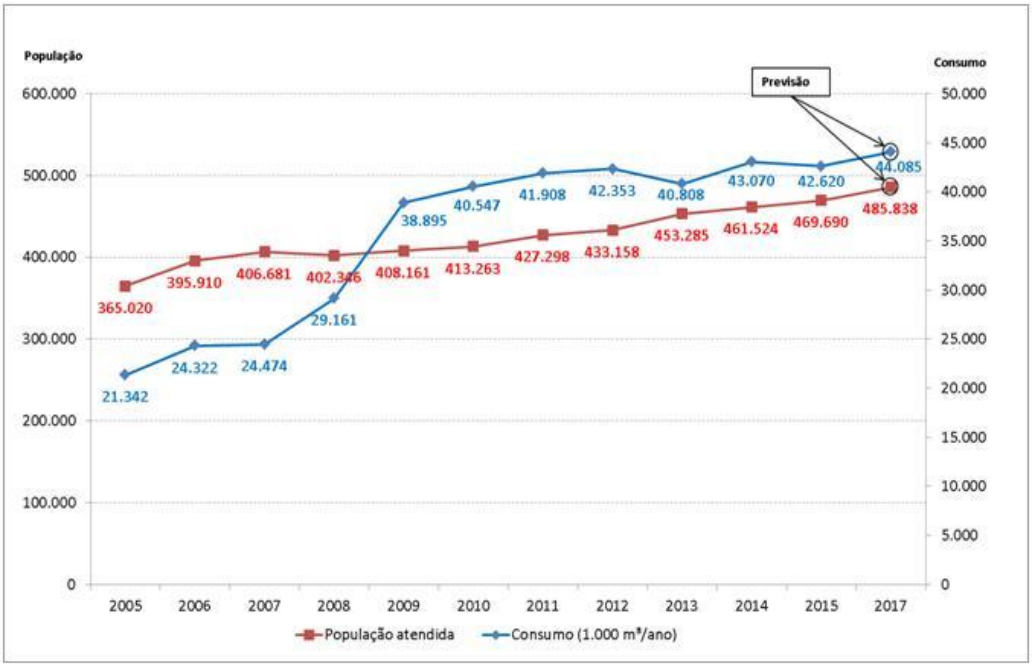
Estes valores tornam-se preocupantes se este aumento for semelhante aos demais estados brasileiros, pois a capacidade atual de produção de água no Brasil pode absorver apenas 7,5% de acréscimo no consumo.

---

[34] http://www3.ana.gov.br/portal/ANA - acesso em 11/11/2017

[35] *http://www.fao.org/nr/water/aquastat/data/query/index.html* - Acesso em 05/11/2017

Figura 44 – Gráfico crescimento população e consumo de água em Florianópolis



Fonte: Dados da ANA[36] e IBGE[37] - Elaboração própria

Ações de conscientização devem ser realizadas para diminuir o crescimento da demanda e disponibilizando informações de consumo de água em tempo real pode ser uma forma de promover o consumo consciente visto que Ahmad et al. (2016) e Lunzer (2007), afirmam a importância do *feedback* aos consumidores de energia, podendo ser proporcionado por meio de uma rede inteligente de monitoramento remoto, que em posse das informações sobre o consumo o próprio consumidor age de forma preventiva para economia de energia. Esta ação pode ser espelhada para o sistema de distribuição de água.

[36] Agência Nacional de Águas – disponível em <http:// app.cidades.gov.br/serieHistorica/#> acesso em 25/01/2018.

[37] Instituto Brasileiro de Geografia e Estatística – disponível em <https://cidades.ibge.gov.br/xtras/perfil.php?lang=&codmun=420540&search=santa-catarina|florianopolis>. Acesso em 21/01/2018

Tendo como referência as soluções empregadas nos sistemas de medição de energia elétrica, o uso de tecnologias *online* é visto como a alternativa para o problema apresentado, mas com ressalva à disponibilidade das fontes de energia, pois ao contrário do sistema elétrico os equipamentos de medição de água normalmente não possuem acesso à rede elétrica de distribuição.

Considerando ainda a evolução dos serviços de armazenamento de dados em nuvem junto com o avanço da tecnologia da Internet das Coisas, a utilização de equipamentos que consigam conversar e dispor as informações para acesso remoto de forma contínua e que possuam baixo consumo de energia, podem ser alternativas para a proposta de solução.

Levando em conta que no empreendimento existe hoje uma rede de acesso remoto sem fio e servidores de dados da própria administração central, a alocação de espaço para a criação de um serviço com disponibilidade de conexão à internet pode ser alternativa de armazenamento e gerenciamento dentro do próprio Sapiens Parque.

Tendo em vista que, o Sapiens Parque em sua constituição acionária, possui como sócio majoritário o Governo do Estado de Santa Catarina, espera-se com este estudo promover a discussão sobre medição remota para consumo de água e demonstrar que estes modelos podem ser desvinculados de soluções proprietárias dos fabricantes de relógios de medição de água, os quais geralmente não se comunicam entre si, o que para uma instituição pública, com regramentos de aquisições bem rigorosos que não garantem a continuidade de compra de modelos específicos, possuir um sistema flexível de comunicação independente dos modelos de medidores instalados facilitará muito a gestão dos ativos instalados em campo.

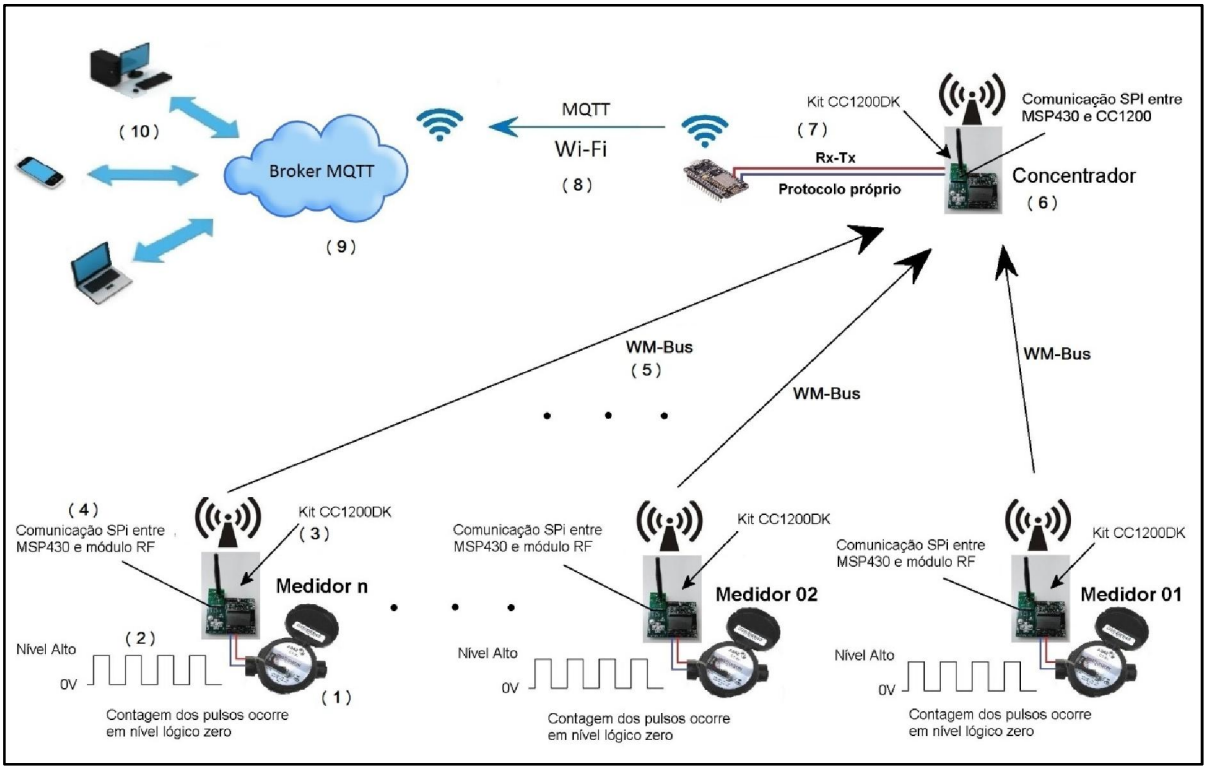
7 PROPOSTA DE SOLUÇÃO

7.1 Modelo geral da solução

A proposta de solução contemplada nesta dissertação prevê a utilização de um kit de desenvolvimento CC1200DK da *Texas Instruments* acoplado a um módulo WiFi ESP8266, que juntos formam o concentrador de dados para envio das informações ao servidor e também outro kit CC1200DK instalado junto a um medidor de água com saída pulsada para coleta e transmissão de dados dos volumes consumidos para o concentrador.

A modelagem do sistema pode ser visualizada na figura 45.

Figura 45 – Modelo de solução



Fonte: Elaboração própria

Onde:

1) Registrador mecânico velocimétrico unijato;
2) Sinais de pulsos ao nível baixo produzidos pela passagem do imã junto ao sensor conforme consumo de água;
3) Conjunto CC1200DK utilizado junto ao medidor de água para coletar informações de consumo;
4) Comunicação SPI entre placa TrxEB – *SmartRF Transciever Evaluation Board* e placa CC1200 *Evaluation Module Kit* 868-930Mhz;
5) Comunicação via radio frequência com protocolo WM-Bus;
6) Conjunto CC1200DK utilizado como concentrador de dados para as informações de consumo dos medidores;
7) Comunicação Rx-Tx por meio de protocolo próprio entre o conjunto CC120DK e a ESP8266;
8) Transmissão via Wi-Fi dos dados coletados por meio do MQTT;
9) Servidor MQTT na nuvem;
10) Acesso as informações por meio da internet.

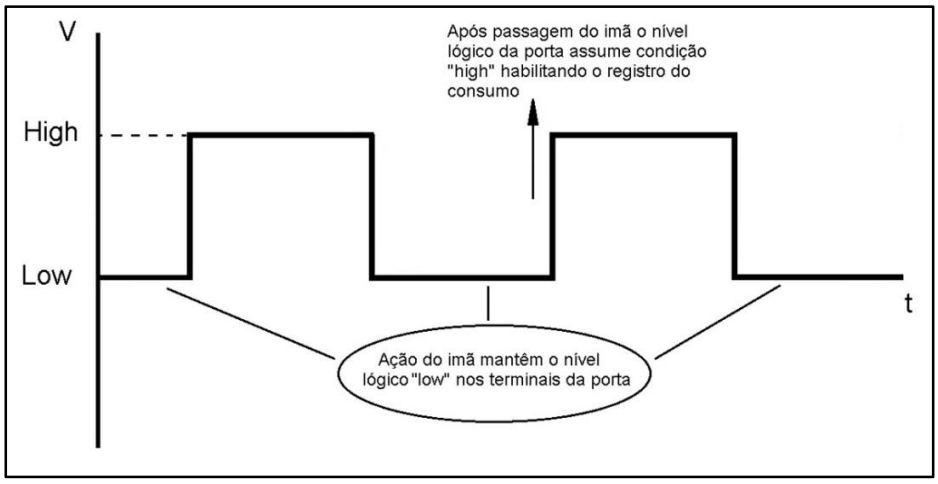
Para a pesquisa foi utilizado apenas um ponto de medição de água e um ponto como concentrador para validar a solução e não foi verificado o desempenho desta solução com vários equipamentos medidores e nem com mais de um concentrador. Também não foi tema da pesquisa a capacidade da rede formada entre os dois equipamentos e possíveis ampliações.

O processo de leitura começa com a medição por meio de um registrador mecânico velocimétrico de jato único que em um dos seus ponteiros possui um pequeno imã. A cada volta realizada por este ponteiro (que representa a medição do valor de um litro) ocorre uma interação com um sensor interruptor acoplado na parte externa do medidor. Esta interação fornece uma condição de curto circuito nos

terminais do cabo conector que está ligado a uma das portas GPIO[38] do módulo CC1200DK e também ao terminal terra.

A condição da porta de entrada e saída de dados do módulo CC1200DK foi programada com código de leitura para acumular o registro de cada litro consumido no momento em que ocorre a transição de subida, ou seja, após a liberação da condição de curto circuito nos terminais da porta de entrada, o registro efetivo do consumo de água ocorre quando a tensão nos terminais retorna ao nível lógico alto. Na figura 46 é apresentado o momento onde ocorre a efetivação do registro de consumo.

Figura 46 – Momento de registro de consumo



Fonte: Elaboração própria

O módulo CC1200DK possui em sua placa TrxEB, dez conjuntos com 8 portas para entrada e saída de dados. Algumas destas possuem aplicações compartilhadas, pois estão conectadas diretamente a outros blocos de funções, tais como, terminais de LEDs, terminais do LCD, conexões RX/TX entre outras.

Para monitorar os sinais de tensão provenientes do sensor interruptor instalado no medidor de consumo, foi utilizado a porta P2.5. A escolha desta porta

[38] GPIO – do Inglês General Purpose Input/Output – Portas Programáveis de Entrada e Saída de dados (livre tradução).

proporcionou um caminho alternativo para simulações de consumo, pois ela está associada ao botão "*down button*" e com ele é possível simular o sensor interruptor sem necessariamente existir um medidor de água acoplado à placa eletrônica. Este processo permanece acumulando os dados obtidos na memória do CC1200DK e, a cada 1 hora, este montante é transferido primeiro entre as placas do MSP430 e CC1200 via comunicação SPI[39] e desta última, os dados são enviados para o módulo concentrador via rádio frequência.

O módulo concentrador faz o processo inverso, ao receber os dados por meio da rádio frequência, transfere os dados para o microcontrolador MSP430 (via SPI). A comunicação serial RX/TX é utilizada em conjunto com um protocolo próprio, desenvolvido para a aplicação com o propósito de transferir as informações para o ESP8266.

No ESP8266, mediante protocolo próprio, os dados são enviados via WiFi para o *broker* MQTT e neste último as informações são tratadas adequadamente para a visualização de forma amigável para usuários.

7.2 Programação do código de leitura de dados

O desenvolvimento do código de leitura dos dados teve como base um dos códigos disponibilizados pela *Texas Instruments* em seu sitio eletrônico[40]. Várias modificações foram realizadas para que o código fosse adaptado ao uso para coleta de dados junto ao relógio de água.

Outra fonte utilizada foi o tutorial "*Beginning Microcontrollers with the MSP430"* de Gustavo Litovsky[41]. Neste tutorial o autor faz uma revisão geral sobre o MSP430 e suas funcionalidades com exemplos básicos de programação.

---

[39] *Serial Peripheral Interface* – Interface Periférica Serial

[40] http://www.ti.com/tool/CC1200DK - acesso em 15/06/2017

[41] http://www.glitovsky.com/Tutorialv0_4.pdf - acesso em 05/082017

A estrutura do código está baseada no tempo de monitoramento entre os pulsos provenientes do medidor de água e as interrupções provocadas pelo estouro dos tempos pré-programáveis.

Várias aplicações foram utilizadas no código entre elas podemos citar:

a) Interrupção de relógio de tempo real - RTC;
b) Função TX e RX;
c) createPacket;
d) updateLcd.

7.2.1 Interrupções de relógio de tempo real - RTC

As interrupções de relógio em tempo real criam condições para envio dos dados que foram coletados entre os tempos de leitura programados. No código desenvolvido a chamada é realizada por meio da função "*__interrupt void RTC_ISR(void)* ";

O módulo CC1200DK utiliza um microcontrolador modelo MSP430F5438A e este possui uma biblioteca RTC que encapsula rotinas comumente utilizadas para monitoramento do tempo (TI, 2016). Neste microcontrolador existem três módulos de operação do RTC, classificados em A, B e C.

No modo RTC-A podemos encontrar a função de relógio e calendário em tempo real e também a função contador para uso geral. O modo calendário disponibiliza as informações em tempo real de segundos, minutos, horas, dias da semana e do mês bem como a do ano. Possui ainda lógica de ajuste e autocorreção do tempo para anos bissextos e alarmes programáveis. No modo contador, o *clock* é obtido por sistemas internos (ACLK ou SMCLK)[42] que possuem versões pré-definidas de divisores. Quatros contadores individuais de 8 bits são conectados em cascatas para obter um contador de 32 bits. Isso proporciona intervalos de transbordo múltiplos de 8 bits.

[42] *Clock* auxiliar e Clock mestre do subsistema respectivamente

No modo RTC-B temos apenas suporte ao modo calendário sem o módulo de contador, mas com possibilidade de trabalhar no modo de baixa potência LPM3.5. Este modo de baixa potência desabilita a CPU e todas as funcionalidades digitais, incluindo a memória RAM quando não estão em uso proporcionando economia de energia.

Já no modo RTC-C, o módulo calendário possui alarmes programáveis e previsão para compensação do relógio de tempo real para variações da temperatura do cristal oscilador.

Nos dois modos RTC-B e RTC-C encontramos as mesmas informações do modo calendário do RTC-A bem como a lógica de ajuste para anos bissextos. Um resumo das funções de cada modo RTC pode ser visto na tabela 06.

Tabela 06 – Funcionalidades dos modos RTC

Feature	RTC_A	RTC_B LPM3.5, Calendar Mode Only	RTC_C Protection Plus Improved Calibration and Compensation
Calendar Mode	Yes	Yes	Yes
Counter Mode	Yes	No	Optional (device-dependent)(1)
Programmable Alarms	Yes	Yes	Yes
Password Protected Calendar Registers	No	No	Yes
Input Clocks	ALCK, SMCLK	32-kHz crystal oscillator	32-kHz crystal oscillator
LPM3.5 Support	No	Yes	Yes
Offset Calibration Register	Yes	Yes	Yes
Temperature Compensation Register	No	No	Yes
Frequency Adjustment Range	-2.035 ppm × 63 ≈ -128 ppm +4.069 ppm × 63 ≈ +256 ppm	-2.17 ppm × 59 ≈ -128 ppm +4.34 ppm × 59 ≈ +256 ppm	-240 ppm, +240 ppm(2)
Frequency Adjustment Steps	-2.035 ppm, +4.069 ppm	-2.17 ppm, +4.34 ppm	-1 ppm, +1 ppm
Temperature Compensation	With software, manipulating offset calibration value	With software, manipulating offset calibration value	With software using separate temperature compensation register
Calibration and Compensation Period	64 min	60 min	1 min
BCD to Binary Conversion	Integrated for Calendar Mode	Integrated for Calendar Mode plus separate conversion registers	Integrated for Calendar Mode plus separate conversion registers
Event/Tamper Detect With Time Stamp	No	No	Optional (device-dependent)(1)

(1) See the device-specific data sheet.
(2) Total adjustment range of offset calibration plus temperature compensation. See the RTC_C chapter for details.

Fonte: *User's Guide Texas Instruments,*2016

Todos os modos possuem funções de interrupção programáveis por meio de registros vetoriais de interrupção. As configurações dos módulos são realizadas por meio de tabelas de registro de funções que variam entre controle dos tempos em segundos, minutos, horas, dias da semana e do mês, mês, ano, contador e alarmes.

Para a programação do sistema de leitura de dados foi utilizado o modo calendário do RTC-A com os seguintes registros vetoriais de interrupção;

a) RTCTEVIE – habilita as interrupções de evento de tempo
b) RTCRDYIE – habilita as interrupções de leitura do relógio
c) RTCBCD – Seleciona modo de operação do relógio de tempo real (BCD ou Hexadecimal)
d) RTCHOLD – Habilita suporte do *clock* em tempo real
e) RTCMODE – Seleciona entre o modo contador ou calendário
f) RTCTEV – Habilita evento de tempo (horas ou minutos)

7.2.2 Função TX e RX

A função TX utilizada para a transmissão dos dados é executada pelas subfunções “radioTxISR”, “initTX” e runTX. A primeira opera com a interrupção e prepara o pacote de dados, a segunda configura o modo de transmissão TX para envio dos dados e a terceira executa a tarefa de envio.

Já a função RX trabalha com a recepção dos dados transmitidos e é executada por duas subfunções: a “radioRxISR” - que manipula os pacotes de dados após o recebimento e define as variáveis de acesso e limpa as “*flags*” de sinalização e, pela subfunção “initRX” - que coloca o rádio em modo de economia de energia (*Sniff mode*)[43].

Durante as operações de transmissão e recepção a função RX tem prioridade sobre a função TX.

[43] Sniff mode – Modo em que o dispositivo reduz a atividade para economia de energia.

7.2.3 Função createPacket

Esta função é chamada antes do envio dos dados. Ela constrói o pacote com os bytes correspondentes às leituras realizadas e insere os bytes de verificação cíclica de redundância – CRC[44] formando a seguinte sequência conforme a tabela abaixo.

Tabela 07 – Sequência de bytes no pacote de dados

Byte	Dado
1	Identificação do equipamento
2	Função à ser medida
3	Dado – dia
4	Dado – mês
5	Dado – ano (milhar e centena)
6	Dado – ano (dezena e unidade)
7	Dado – hora
8	Dado – minuto
9	Dado – segundo
10	Dado – volume em litros (milhar e centena)
11	Dado – volume em litros (dezena e unidade)
12	Nº sequencial da medição
13	CRC – 1º byte
14	CRC – 2º byte

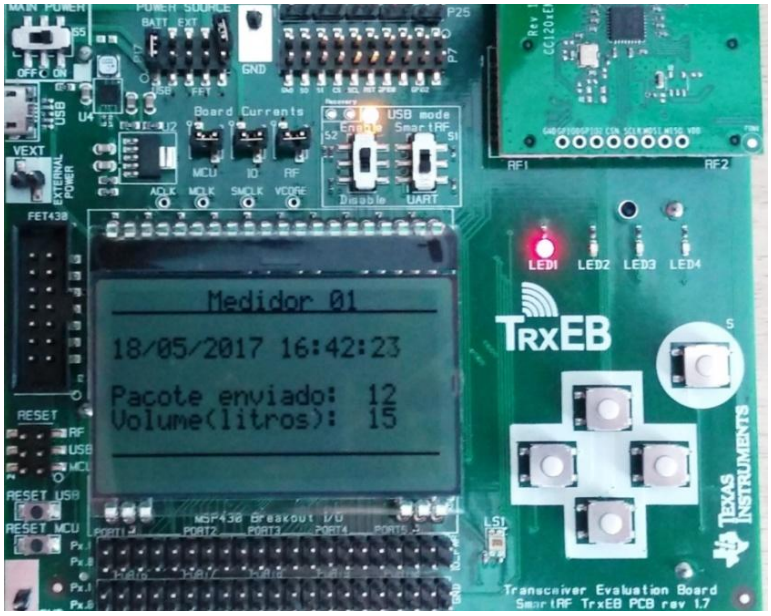
Fonte: Elaboração própria

7.2.4 Função updateLcd

O conjunto CC1200DK instalado junto ao medidor de água foi configurado para fornecer as informações de medição localmente por meio do display LCD de 128x64 pixels que se comunica com a placa TrxEB pela interface SPI o qual pode ser visualizado no figura 47. As informações disponíveis são: Identificação do medidor, data, hora, o numero sequencial do pacote de dados enviados e o volume em litros medido no período.

[44] CRC – do Inglês *Cycling Redundancy Check*

Figura 47 – Display LCD 128x64 pixels



Fonte: Elaboração própria

7.3 Programação do código para recebimento dos dados

Assim como o código desenvolvido para a leitura dos dados, a programação do dispositivo de recebimento teve também várias adaptações realizadas em um código disponibilizado pela *Texas Instruments* com base em informações importantes sobre programação obtidas dos manuais da Texas e de Litovky (2012).

O código está estruturado para habilitar o recebimento de dados quando ocorrer uma interrupção a qual informa que uma sequência de bytes precisa ser recebida pelo conjunto concentrador. Essa interrupção é monitorada pela função *packetSemaphoreRX.* Antes de habilitar o recebimento, a função limpa um buffer auxiliar, criado para recebimento dos bytes, verifica os bytes CRC, recebe o pacote via protocolo WM-Bus e já o encaminha para o MSP430 via protocolo SPI.

No MSP430, as informações são encaminhadas via protocolo próprio pelos canais RX/TX para o ESP8266.

Outras três funções foram utilizadas para suprir a demanda do recebimento além das descritas no código de envio de dados que são importantes para o funcionamento do processo, são elas:

a) initUARTConfigs
b) setSMCLK8MHz
c) calibrateRCOsc

7.3.1 Função initUARTConfigs

A função initUARTConfigs, configura o módulo de comunicação serial do MSP430 com os seguintes parâmetros:

a) Módulo SCI_A com modo UART
b) Portas P5.6 e P5.7 como TX e RX respectivamente
c) *Baud-rate* de 115200 com *clock* a 8MHz
d) 8N1 – 8 bits de dados, sem bit de paridade e 1 stop bit

No módulo USCI_A, além do suporte ao modo UART existe também suporte as seguintes configurações: Modo SPI, detecção automática de *baud-rate* e formação de pulsos para comunicação IrDA[45].

7.3.2 Função setSMCLK8MHz

Nesta função utiliza-se um oscilador interno de baixa frequência (32.768 KHz) para configurar o *clock* do conjunto em 8MHz.

A frequência do controlador digital é configurada inicialmente em 16MHz fazendo UCSCTL1= DCORSEL_5 e depois aplica-se o divisor 2 configurando os bits FLLD do UCSTL2 = FLLD_1.

---

[45] Infrared Data Association – www.irda.org – acesso em 11/11/2017

7.3.3 Função calibrateRCOsc

Quando se utiliza a função *Sniff*, onde o microcontrolador “adormece” enquanto não é exigida nenhuma tarefa, é necessário realizar ajustes periódicos da freqüência do oscilador, pois este pode sofrer alterações de acordo com a temperatura e a tensão de alimentação. Esta calibração periódica é realizada pela função calibrateRCOsc toda vez que a rotina initRX é executada.

7.3.4 Módulo WiFi ESP 8266

O ESP8266 é um dispositivo SoC (*System on Chip*) com módulo para rede WiFi 802.11 b/g/n/e/i embutido.

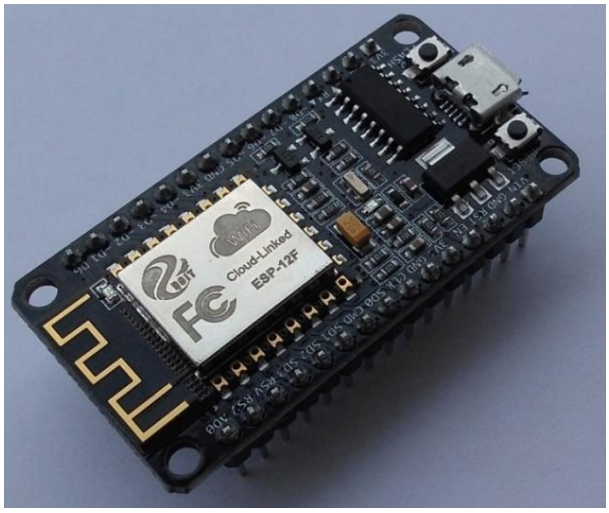
Com arquitetura RISC, no ESP8266 podemos encontrar conectores GPIO, barramentos de comunicação serial, entrada ADC, saída PWM, CPU, memória *flash*, sensor interno de temperatura, memória RAM para dados e instruções e memória ROM para *boot*.

Além da funcionalidade Wi-Fi, alguns modelos do ESP8266 possuem diversas outras características e um processador que permite integração com sensores externos e outros dispositivos por meio das GPIO’s.

Existem variados modelos disponíveis no mercado e para a solução proposta foi utilizado o modelo ESP-12f.

Com CPU de 32 bits e frequência de 80 MHz, suportam vários protocolos de segurança como WEP, WPA e WPA2 e por possuírem baixo consumo de energia em modo *sleep*, podem ser utilizados para projetos de IoT. A figura 48 mostra um ESP8266 modelo 12-f.

Figura 48 – Módulo WiFi ESP8266



Fonte: Elaboração própria

Pode ser utilizado como processador de aplicativos ou como adaptador Wi-Fi para qualquer projeto com micro controlador como conectividade simples por meio de interface UART ou pela interface AHB (*Advanced High-perfomance Bus*) da CPU.

7.3.5 Medidor de água com saída pulsada

O PBA do Sapiens Parque prevê a utilização de medidores ultrassônicos em todos os pontos de consumo de água, mas para validação da solução foi empregado um medidor de jato único velocimétrico da SAGA[46] modelo US-3.0 com saída pulsada e com capacidade nominal de 1,5 m³/h com emissão de pulsos a cada 1 litro. A figura 49 mostra o medidor utilizado para protótipo de solução.

[46] https://www.sagamedicao.com.br/ - acesso em 16/10/2017

Figura 49 – Medidor de jato único velocimétrico SAGA

Fonte: Elaboração própria

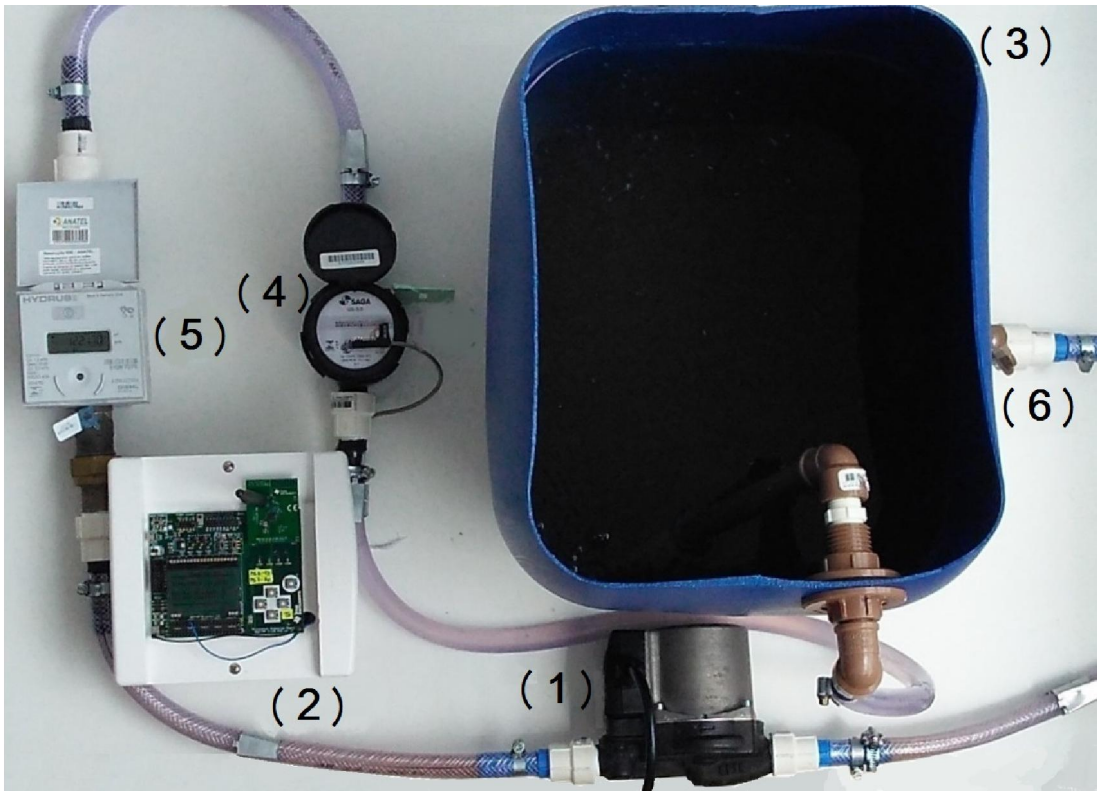
7.3.6 Unidade de medição portátil

Para testar e validar a proposta de solução foi construído uma estação portátil de medição contendo um reservatório de água, uma bomba para circulação de líquido, um medidor ultrassônico padrão utilizado pelo Sapiens Parque e o medidor velocimétrico da SAGA com saída pulsada ligados em série para a medição de consumo de água e também um conjunto CC1200 DK + placa ESP8266 para a coleta e envio das informações para broker.

O propósito do uso de um medidor ultrassônico padrão do Sapiens é de comparar o consumo registrado no medidor velocimétrico e com as leituras realizadas pelo sistema proposto.

A estação portátil pode ser visualizada na figura 50, a seguir.

Figura 50 – Estação portátil de medição



Fonte: Elaboração própria

Onde:

1) Motor bomba de circulação monofásica com capacidade de 0,8 m³/h;
2) Conjunto CC1200DK configurado para coleta de dados (medidor);
3) Reservatório de água com capacidade para 20 litros;
4) Medidor mecânico velocimétrico unijato;
5) Medidor eletrônico ultrassônico padrão Sapiens Parque;
6) Válvula para controle de vazão;

A bomba circulação da estação portátil proporciona ao circuito uma vazão média de 13l/min, mas com possibilidade de ser reduzida por meio da válvula de controle de vazão do reservatório de água.

O Eclipse[47] foi utilizado como *broker* MQTT e para visualização dos dados foram configuradas duas *dashboard*, sendo a primeira em um dispositivo móvel celular com o aplicativo IoT MQTT- *dashboard* e a segunda em um computador com o *Grafana dashboard*[48].

Inicialmente foram realizados envio de dados coletados em intervalos de 60 segundos durante o período de uma hora, variando a abertura da válvula, para verificar o funcionamento dos equipamentos, dos códigos e as soluções gráficas implementadas.

Após vários testes de comunicação passamos para a simulação de consumo diário visando a continuidade do serviço na nuvem e desempenho dos equipamentos.

Pelo fato dos requisitos de segurança predial dentro do Sapiens Parque ainda serem insuficientes para garantir a integridade da unidade de medição, o teste foi realizado na própria unidade remota com simulação de envio de dados a cada hora.

7.3.7 Dashboards

Para visualização e acompanhamento dos dados foram utilizadas duas plataformas disponíveis na WEB: A d*ashboard* do Grafana e Iot MQTT *dashboard*. A escolha destas plataformas teve como base a disponibilidade dos serviços por meio de assinaturas livres sem custo para o usuário e em especial pela disponibilidade de *widget* da própria plataforma que facilita a criação de ambientes amigáveis ao usuário.

O Grafana é um software utilizado para análises temporais, onde é possível reunir várias fontes de dados de diferentes locais quando se trabalha com códigos abertos. Utilizado em uma ampla variedade de casos que vai desde desenvolvimento de softwares empresariais (DevOps), Internet Industrial das Coisas (IIoT) e até na Tecnologia de publicidade (AdTech). Construído com tecnologias de código aberto,

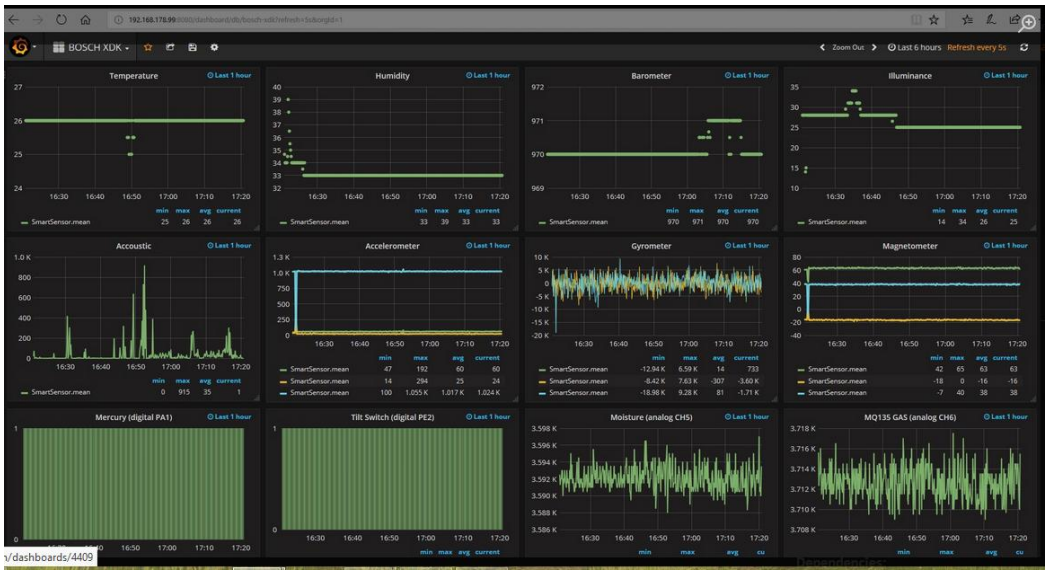
[47] Iot.eclipse.org – acesso em 02/11/2017

[48] https://grafana.com/dashboards - acesso e 22/01/2018

permite que o usuário utilize as próprias ferramentas de desenvolvimento evitando a dependência do SaaS[49].

A figura 51 apresenta uma tela de acompanhamento de vários sensores em forma gráfica de uma aplicação Bosch XDK node[50].

Figura 51 – Tela do Grafana dashboard



Fonte: Elaboração própria

O IoT MQTT dashboard é um aplicativo para *smartphone*, simples com várias funcionalidades, tais como:

a) Multi conexões;
b) Subscrição com função de notificação;
c) Publicação por meio de texto, botões, chaves etc.

Possui três telas de interface. A primeira mostra as conexões existentes onde é possível adicionar e configurar as conexões. A segunda e a terceira somente disponível após realizar uma conexão com um *broker* estão configuradas em formas de abas, realizam a função do subscrito e do publicador de um tópico.

---

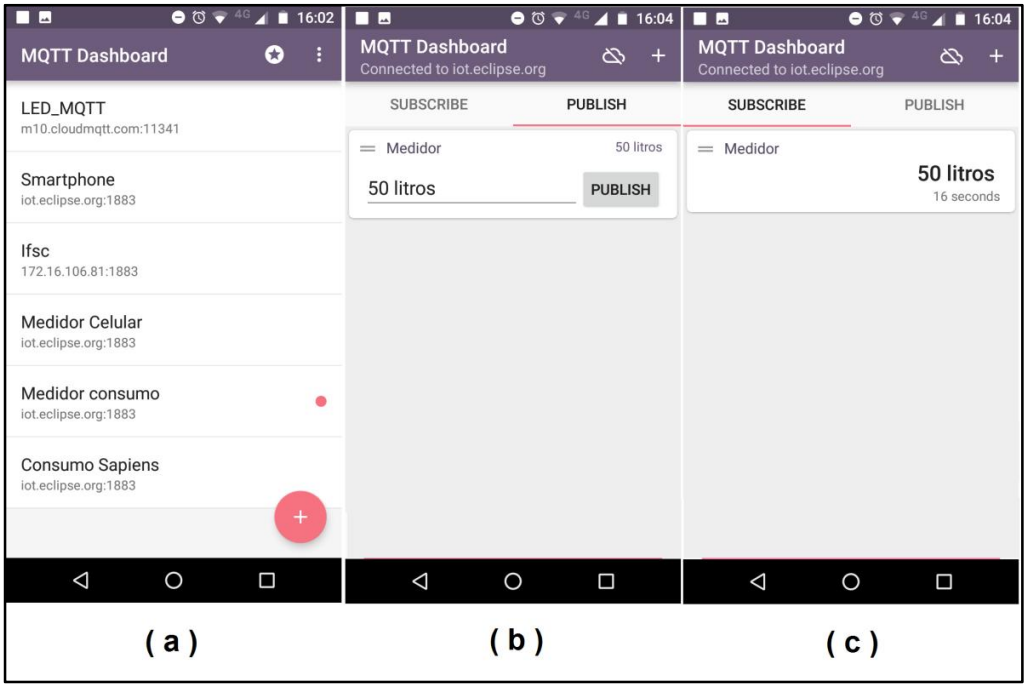
[49] SaaS – *Software as a Service* – Software como Serviço (tradução nossa).

[50] Disponível em https://grafana.com/dashboards/4409 - acesso em 12/04/2018.

A vantagem do IoT MQTT está em ser uma plataforma leve, com possibilidade de notificações e com alguns bons componentes de interface disponível.

Na figura 52 é possível visualizar as três telas do IoT MQTT onde: (a) tela inicial, (b) interface do publicador e (c) interface do subscrito.

Figura 52 – Telas do IoT MQTT dashboard



Fonte: Elaboração própria

8 RESULTADOS

As pesquisas e os encaminhamentos realizados para implementação da proposta de solução encontraram diversas dificuldades que precisaram ser contornadas durante o desenvolvimento desta dissertação.

A plataforma CC1200DK da *Texas Instruments* mostrou grande potencial para várias soluções que utilizam o protocolo WM-Bus ou protocolos próprios para transmissão de dados e comunicação serial, mas o desenvolvimento dos códigos necessitou grande tempo de estudo mesmo partindo de um modelo existente.

Pelos testes realizados para a transmissão de dados percebe-se que o sistema pode operar em distâncias razoáveis, garantindo uma grande área de cobertura, mas atenção especial deve ser dada as interferências ocasionadas pelas edificações. Mesmo prevendo a existência de uma rede de dados dentro do Sapiens Parque, equipamentos instalados de forma inadequada sem visão direcionada entre as antenas de rádio pode ser um problema para transmissão dos dados.

O registro do tempo e data das medições foi objeto de preocupação, mas a utilização de *dashboards* como interface na WEB integrada com banco de dados na nuvem mostrou que esses dados podem ser atribuídos pelos servidores e isto poderá reduzir o pacote de dados inicial de 14 bytes para sete bytes, deixando os dados de data e hora para serem inseridos pelas *dahsboards* e pelos *brokers*.

A utilização do ESP8266 apresentou problemas quando utilizado para envio dos dados recebidos no terminal RX, via comunicação UART/serial do conjunto CC1200DK. Em tempos aleatórios a comunicação com o *broker* era desfeita comprometendo o acompanhamento das medições periódicas. Vários testes foram realizados para identificar a possível causa, desde isolamento da saída serial, troca de placa ESP8266 por outra de mesmo modelo ou por modelos diferentes, alteração do *baudrate*, alteração de fontes de alimentação etc. A solução definitiva veio com a transformação dos dados enviados do padrão hexadecimal para string.

O primeiro *broker* utilizado nos testes foi o *cloud* MQTT[51], mas que se mostrou muito instável durante os testes com frequentes quedas de conexão. O motivo era o tempo sem comunicação com o *broker*. Para contornar este problema utilizou-se o *broker* do projeto Eclipse que permitia tempos maiores sem comunicação.

A *dashboard* do Grafana apresentou mínimas perdas que foram atribuídas aos problemas do próprio servidor, pois os dados também foram monitorados paralelamente por outra plataforma. A segunda plataforma, visualizada por meio de aplicativo instalado em *smartphone*, foi a IoT MQTT *dashboard.* Esta plataforma não apresentou perdas de dados.

Houve ainda a necessidade de realizar conversões de dados, pois o CC1200DK por facilidade na programação foi configurado para envio dos valores na base hexadecimal. Estes dados depois de enviados eram convertidos em string pela rotina do concentrador para envio ao *broker*. O tratamento final ficou por conta dos valores alocados no banco de dados após coletados no *broker*. Estes foram tratados e inseridos em forma gráfica para visualização na WEB.

8.1 Desempenho do MQTT

Em seu trabalho sobre desempenho do MQTT, GALLO (2018) utilizou duas implementações para identificar os tempos de transmissão de dados com o protocolo MQTT em rede Ethernet e rede Wi-Fi com mensagens de 32 bits.

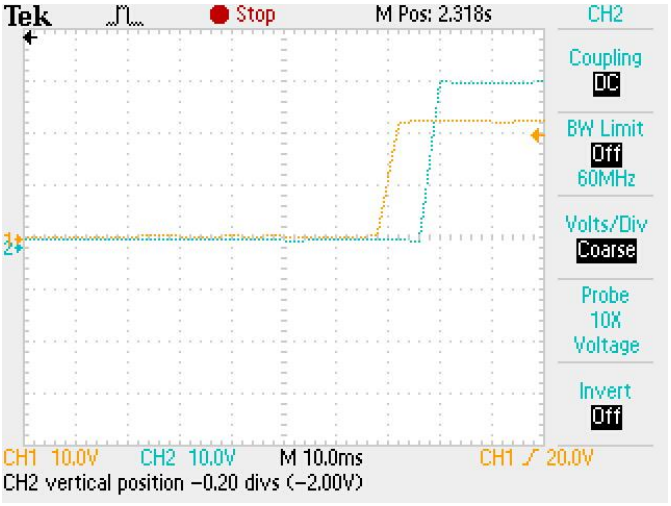
Na primeira implementação foram utilizadas duas placas *Beagle Bones Black*, rede *Ethernet* e o *broker* Mosquitto para simulação de clientes inscritos e subscritos no servidor.

O resultado, indicado na figura 53, foi obtido por meio da medição de tempo com osciloscópio o qual mostrou um atraso de 10 ms entre o envio e recebimento da mensagem.

---

[51] https://www.cloudmqtt.com – acesso em 23/11/2017.

Figura 53 – Diferença de tempo entre sinais na rede ETHERNET

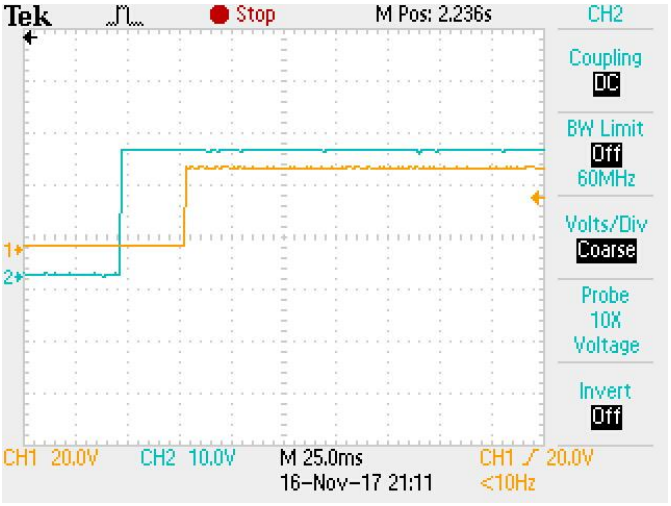


Fonte: GALLO (2018)

Na segunda implementação, substitui-se uma das placas *Beagle Bone Black* por uma placa ESP-8266-12F que possui módulo Wi-Fi para transmissão de dados e mantiveram-se as demais configurações.

Os resultados apresentaram um tempo maior de transmissão e recebimento que foram associados às limitações da rede Wi-Fi local utilizada no teste. O resultado pode ser visualizado na figura 54.

Figura 54 – Diferença de tempo entre sinais na rede WiFi



Fonte: GALLO (2018)

Em um trabalho mais aprofundado realizado por Lee et al. (2013), foram identificado os atrasos na transmissão de dados e perda de pacotes em cada um dos níveis de segurança QoS com equipamentos conectados a rede Wi-Fi e a rede com fio.

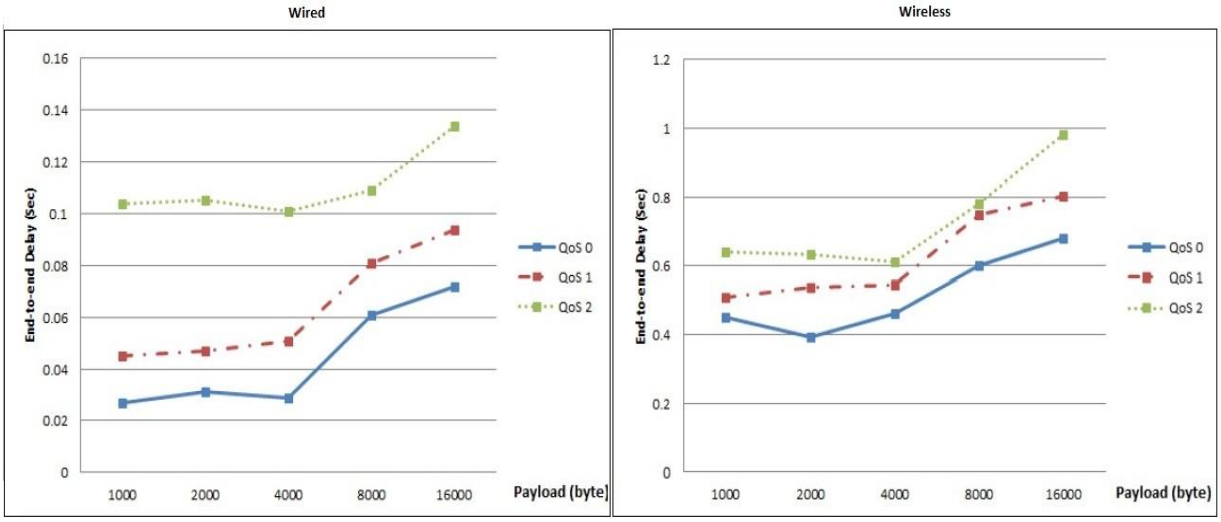
Para a realização dos testes foram utilizados *payloads* variados de 1.000 até 16.000 *bytes* e os resultados mostraram que os atrasos permanecem quase constantes com *payload* até 4.000 bytes com incremento do tempo a partir deste valor, tanto para a rede Wi-Fi com para rede com fio.

As perdas de pacotes associados também seguiram o mesmo perfil quase constante até 4.000 *bytes* de *payload* apresentando aumento de perdas acima disto.

A análise de correlação entre os tempos de envio de mensagens (*end-to-end*) e a perda de pacotes nos três níveis de segurança mostra uma relação positiva forte tanto para a rede *wireless* como *wired*.

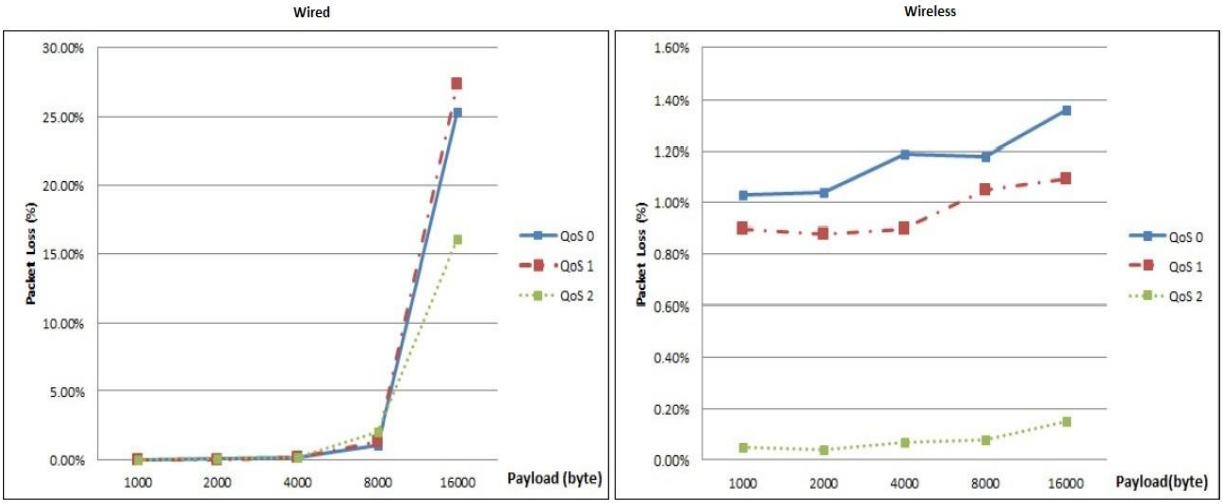
Os gráficos das medições e tabela de valores de correlação podem ser visualizados nas figuras 55 e 56.

Figura 55 – Gráficos de análise de tempo end-to-end rede wired e wireless



Fonte: Lee et al. (2013)

Figura 56 – Gráficos de perda de pacotes end-to-end rede wired e wireless



Fonte: Lee et al. (2013)

Tabela 08– Correlação entre perda de pacotes e tempo de transmissão

Nível QoS	*Wired*	*Wireless*
0	0,771574	0,878897
1	0,788814	0,884428
2	0,991416	0,904699

Fonte: Adaptado deLee et al. (2013)

Estes resultados indicam que na proposta de solução os atrasos e perdas serão muito baixos ao ponto de não interferirem no desempenho do sistema.

8.2 Teste de alcance de transmissão

O CC1200DK possui um pré-programa de teste para verificação do alcance de transmissão que já vem instalado com o equipamento[52].

Com este pré-programa é possível simular transmissões de 100, 1.000, 10.000 e 65.000 pacotes de dados de 3 até 60 Bytes, com taxa de transmissão de 1.200, 50.000 e 200.000 bps e trabalhar alterando as potências de saída (Tx) em 14dB, 10dB, 5dB, 0dB, -5dB, -10dB e -16dB.

Para a execução do teste, a opção disponível mais próxima foi de 100 pacotes dados de 15 Bytes, quantidade de dados imediatamente superior que será transmitida durante as medições dos consumos das unidades do Sapiens Parque.

Para a escolha do local levou-se em consideração a maior distância livre de obstáculos que pudesse existir dentro do terreno do empreendimento e também as edificações disponíveis e em construção bem como as unidades em estado de negociação. Esta opção proporcionou distância um pouco maior que 1.100 metros e será uma região que em curto prazo tempo possuirá canteiros de obras e posteriormente as unidades comerciais.

[52] Também disponível no sítio eletrônico: http://www.ti.com/tool/CC1200DK acesso em 20/08/2017

Para simular um medidor de água, um dos equipamentos foi colocado em altura similar próximo ao chão, enquanto que com o outro equipamento fez-se o deslocamento linear em linha reta mantendo-se a visão entre as antenas.

A distância e sentido da movimentação realizada durante o teste de alcance de sinal pode ser visualizado na figura 57.

Figura 57 – Distância percorrida para teste de alcance de transmissão

Fonte: www.google.com/maps - Editado pelo autor

Os testes realizados neste percurso abrangeram todas as configurações possíveis com envio de 100 pacotes de 15 bytes com registros até a primeira ocorrência de 100% perda dos dados. Os valores obtidos estão mostrados nas tabelas e gráficos a seguir:

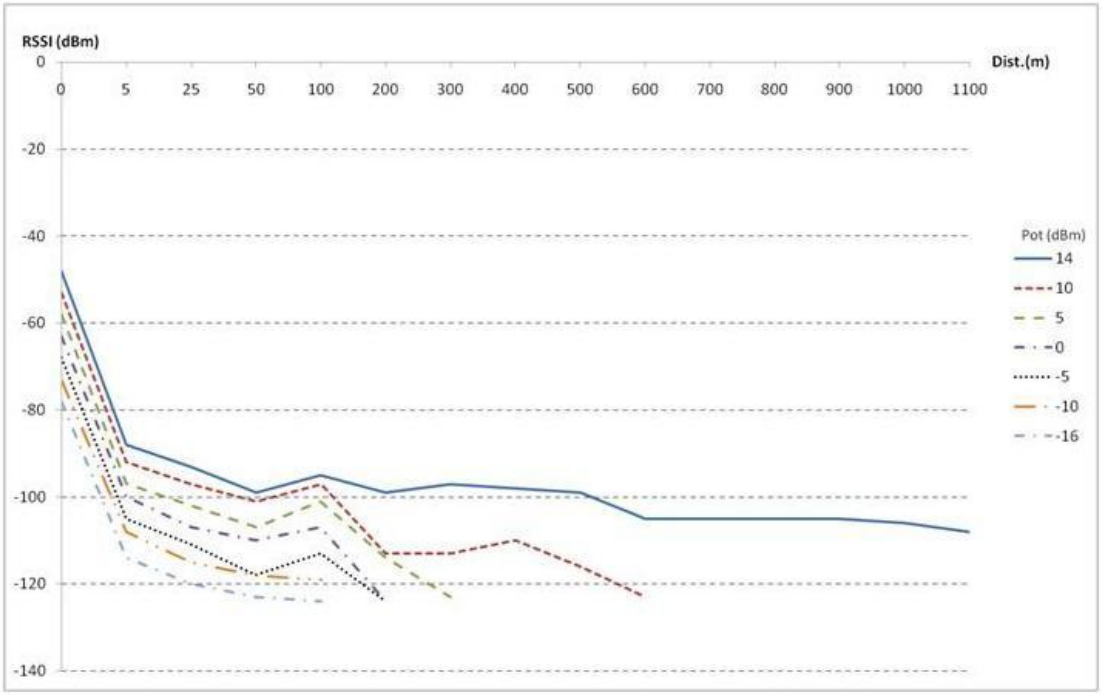
Tabela 09– Potência de sinal recebido com taxa de transmissão de 1,2 kbps

1,2 Kbits/s								
RSSI (dBm)		Potência (db)						
		14	10	5	0	-5	-10	-16
Distância (m)	0	-48	-53	-58	-63	-68	-73	-78
	5	-88	-92	-97	-100	-105	-108	-114
	25	-93	-97	-102	-107	-111	-115	-120
	50	-99	-101	-107	-110	-118	-118	-123
	100	-95	-97	-101	-107	-113	-119	-124
	200	-99	-113	-114	-124	-124	*	*
	300	-97	-113	-123	*	*	*	*
	400	-98	-110	*	*	*	*	*
	500	-99	-116	*	*	*	*	*
	600	-105	-123	*	*	*	*	*
	700	-105	*	*	*	*	*	*
	800	-105	*	*	*	*	*	*
	900	-105	*	*	*	*	*	*
	1000	-106	*	*	*	*	*	*
	1100	-108	*	*	*	*	*	*

* Não realizado pois na distância anterior a perda de pacotes já foi de 100%

Fonte: Elaboração própria

Figura 58 – Gráfico de potência de sinal recebido x distância - 1,2 kbps



Fonte: Elaboração própria

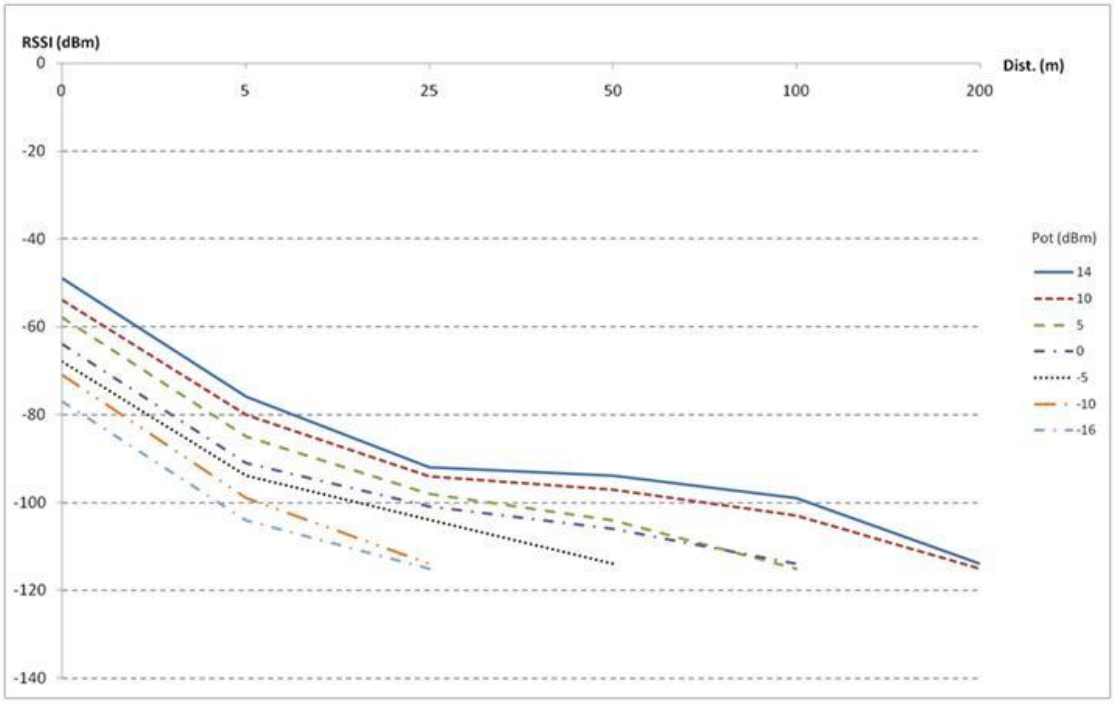
Tabela 10 – Potência de sinal recebido com taxa de transmissão de 50 kbps.

50 Kbits/s								
RSSI (dBm)		Potência (db)						
		14	10	5	0	-5	-10	-16
Distância (m)	0	-49	-54	-58	-64	-68	-71	-77
	5	-76	-80	-85	-91	-94	-99	-104
	25	-92	-94	-98	-101	-104	-114	-115
	50	-94	-97	-104	-106	-114	*	*
	100	-99	-103	-115	-114	*	*	*
	200	-114	-115	*	*	*	*	*
	300	*	*	*	*	*	*	*
	400	*	*	*	*	*	*	*
	500	*	*	*	*	*	*	*
	600	*	*	*	*	*	*	*
	700	*	*	*	*	*	*	*
	800	*	*	*	*	*	*	*
	900	*	*	*	*	*	*	*
	1000	*	*	*	*	*	*	*
	1100	*	*	*	*	*	*	*

* Não realizado pois na distância anterior a perda de pacotes já foi de 100%

Fonte: Elaboração própria

Figura 59 – Gráfico de potência de sinal recebido x distância - 50 kbps



Fonte: Elaboração própria

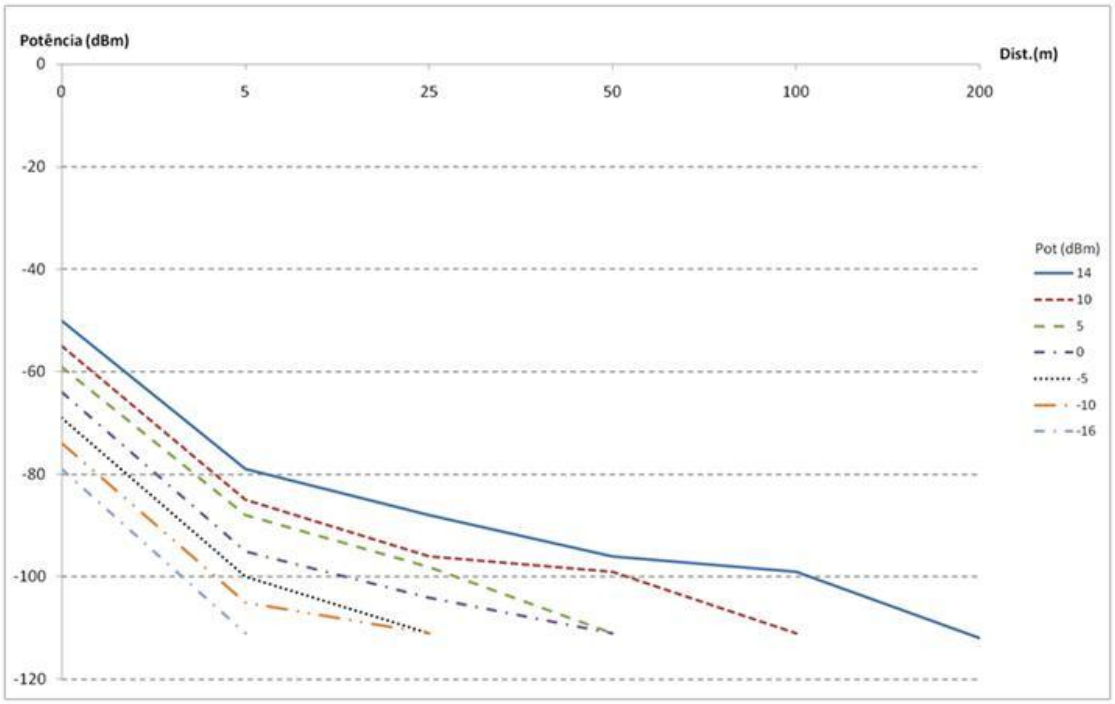
Tabela 11 - Potência de sinal recebido com taxa de transmissão de 200 kbps

200 Kbits/s								
RSSI (dBm)		Potência (db)						
		14	10	5	0	-5	-10	-16
Distância (m)	0	-50	-55	-59	-64	-69	-74	-79
	5	-79	-85	-88	-95	-100	-105	-111
	25	-88	-96	-98	-104	-111	-111	*
	50	-96	-99	-111	-111	*	*	*
	100	-99	-111	*	*	*	*	*
	200	-112	*	*	*	*	*	*
	300	*	*	*	*	*	*	*
	400	*	*	*	*	*	*	*
	500	*	*	*	*	*	*	*
	600	*	*	*	*	*	*	*
	700	*	*	*	*	*	*	*
	800	*	*	*	*	*	*	*
	900	*	*	*	*	*	*	*
	1000	*	*	*	*	*	*	*
	1100	*	*	*	*	*	*	*

* Não realizado pois na distância anterior a perda de pacotes já foi de 100%

Fonte: Elaboração própria

Figura 60 – Gráfico de potência de sinal recebido x distância - 200 kbps



Fonte: Elaboração própria

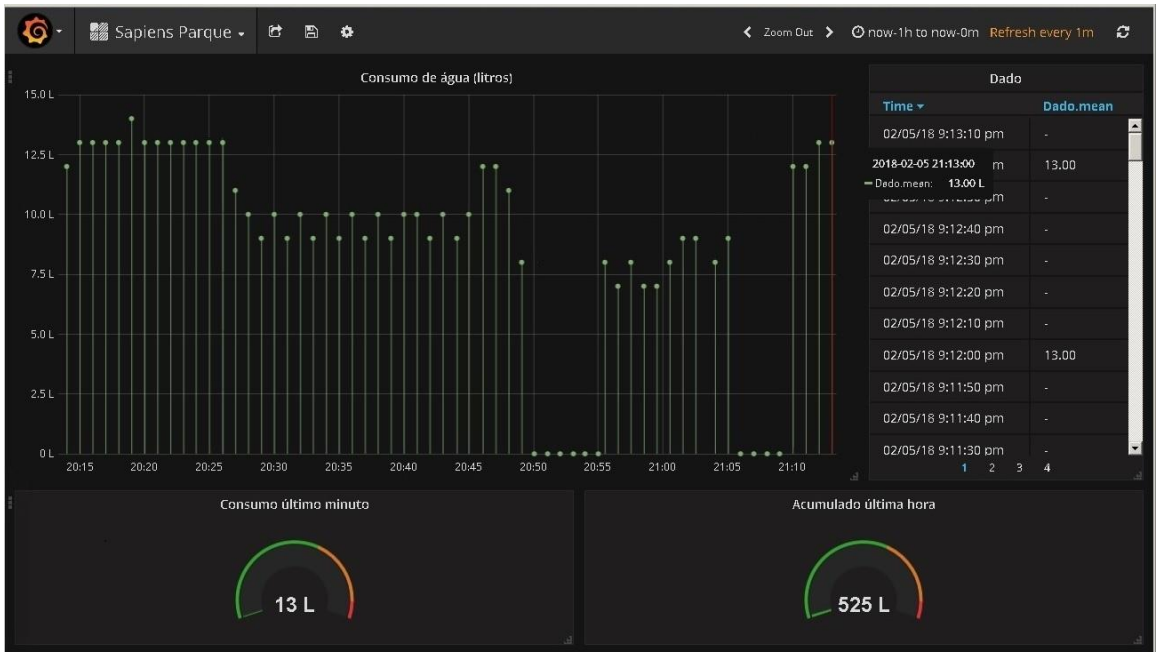
Percebe-se pela análise dos gráficos que para as taxas de transferências maiores, a distância alcançada pelo sinal diminui consideravelmente para todas as potências de saídas. Como os dados a serem transferidos serão compostos por um pacote de 14 bytes enviados em períodos regulares de 1 hora, a taxa de 1.200 bps será suficiente para atender a demanda do sistema.

Os valores obtidos em cada potência de saída para as três taxas de transmissões disponíveis pelo pré-programa PER TEST podem ser visualizadas separadamente no apêndice A.

8.3 Dashboards

No teste inicial para validação do sistema, foram coletados dados durante uma hora com intervalos de 60 segundos. Os dados em sua forma gráfica obtidos no aplicativo Grafana podem ser visualizados na figura 61.

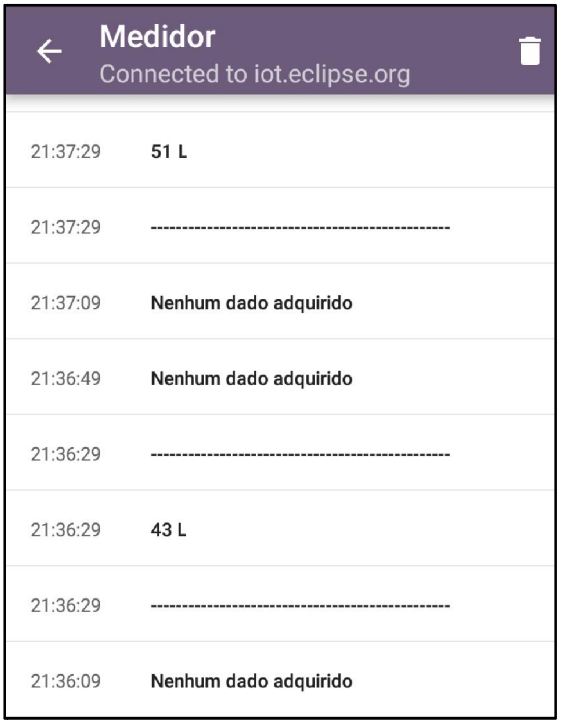
Figura 61 – Grafana Dashboard medição de consumo intervalo de 60s



Fonte: Elaboração própria

Na *dashboard* do IoT MQTT, instalada por meio de aplicativo para smartphones foi utilizada também para visualização do dados mas desta vez em forma de lista. A figura 62 mostra a tela de dados do IoT MQTT *dashboard*.

Figura 62 – Tela de visualização do IoT MQTT dashboard



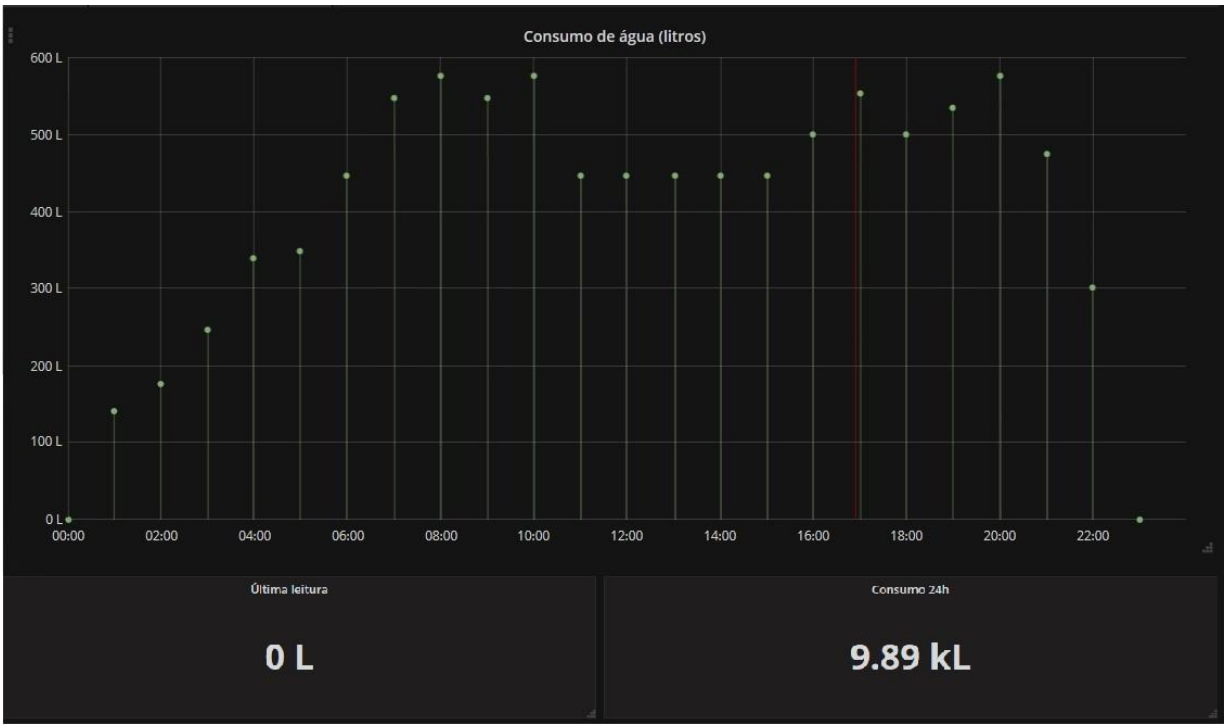
Fonte: Elaboração própria

Em paralelo as medições por meio da telemetria, leituras nos *display* dos dois medidores foram anotadas a cada minuto para comparação e estes valores podem ser visualizados no apêndice B.

No teste de consumo diário, o período de contabilização dos pulsos e envio de dados foi programado em intervalos de 60 minutos e durante um intervalo de 24 horas. Durante este período a válvula de controle de vazão foi alterada algumas vezes forçando o sistema trabalhar com valores diferenciados e no último intervalo a bomba de circulação foi desligada para identificar consumo nulo.

Os valores coletadas durante o intervalo de teste de 24 horas pode ser visualizado na figura 63.

Figura 6193 – Medição de consumo horário



Fonte: Elaboração própria

9 CONCLUSÕES

O objetivo geral deste trabalho foi o desenvolvimento de um sistema de comunicação sem fio para medição do consumo de água do empreendimento Sapiens Parque utilizando tecnologias não proprietárias para coleta e transmissão de dados para um servidor local ou remoto, pois sendo o Sapiens Parque uma empresa com regramentos específicos para aquisições de produtos e serviços, não possui garantias que conseguirá manter a continuidade da compra de qualquer modelo proprietário durante as suas fases de implantação.

No início desta dissertação, apresentou-se uma revisão bibliográfica abrangendo trabalhos relacionados com medição remota de serviços públicos de água, gás e energia elétrica bem como as tecnologias utilizadas. Foram abordados também os conceitos de *cloud service*, smart grid, *smart metering* e Internet das Coisas que são os conceitos básicos utilizados na proposta de solução.

Sendo o protocolo MQTT parte importante no processo de desenvolvimento da solução, foi realizada ainda na primeira parte da pesquisa uma descrição sobre este protocolo com os seus modelos e características de funcionamento.

Como se trata de medição de água, uma revisão sobre os métodos e tecnologias existentes para determinação de volumes consumidos foi apresentado no capítulo três onde se verificou grande quantidade de mecanismo e suas aplicações.

Uma apresentação detalhada do módulo de desenvolvimento CC1200DK da Texas *Instruments*, hardware utilizado no protótipo de solução, pode ser encontrada no capítulo quatro onde se apresenta as partes dos seus componentes.

No capítulo cinco temos uma pequena revisão sobre comunicação por rádio frequência e o enquadramento do módulo CC1200DK como equipamento de radiação restrita prevista na regulamentação da ANATEL e também a conformidade do seu uso segundo a norma Européia EN13757-4 que fala sobre os modos de operação do protocolo WM-Bus para dispositivos SRD. Ainda no capítulo cinco temos um breve relato sobre o protocolo WM-Bus, suas formas de operação e uma pequena comparação entre formatos de pilha do modelo OSI e WM-Bus.

A definição do problema é demonstrada no capítulo seis identificando as questões importantes para a pesquisa, fatores limitantes e pontos de atenção.

O modelo geral da solução é apresentado no capítulo sete, onde é feito a descrição dos principais pontos sobre a programação e configuração do CC1200DK e do módulo ESP8266.

No capítulo oito é feito o relato dos problemas encontrados durante a elaboração da pesquisa e suas soluções bem como apresentados todos os resultados obtidos nos testes realizados, tais como o desempenho do protocolo MQTT, alcance de sinal de rádio frequência do CC1200DK e publicação dos dados nas *dashboards*.

Com o desenvolvimento da dissertação foi possível identificar que a utilização de um sistema de comunicação sem fio utilizando protocolo MQTT com sistema próprio de medição por telemetria que utiliza a rádio frequência em ambientes semelhantes ao Sapiens Parque, é viável tecnicamente dentro do contexto de redução do tempo de leitura e agilidade de acesso as informações e também na flexibilidade para conexão de diversos tipos de equipamentos diferentes, visto que a tecnologia utilizada não é proprietária.

Atenção especial deve ser considerada na disponibilidade dos serviços dos *brokers*, visto que em várias situações a conexão apresentou dificuldades muito provavelmente por utilizar assinaturas livre de custos e sem garantias contratuais específicas.

Como trabalho futuro entende-se como importante estudar a vida útil das baterias, pois pelo tempo de testes e medições realizadas, a soma dos volumes medidos e simulados ultrapassou 60.000 pulsos o que equivale a quase sete anos de medição, considerando um pulso de dados a cada hora de consumo.

O módulo CC1200DK se mostrou versátil como protótipo de solução, mas deve-se aprofundar o estudo para validar a utilização como solução final, pois as dimensões das placas e antena são restrições importantes para a instalação junto aos medidores de água. Além disto, outros tipos de microcontroladores devem ser considerados, pois a escolha da placa de desenvolvimento CC1200DK foi baseada na

compatibilidade de operação com o protocolo WM-Bus, mas outras plataformas mais compactas e com custos mais acessíveis devem ser vistas como objeto de estudo.

Outro ponto a ser considerado é a ampliação dos serviços de medição de consumo com a instalação de sensores de pressão junto aos medidores como monitoramento da disponibilidade dos serviços de abastecimento, integrando-o as bases de dados da concessionária com o objetivo de mapear e melhorar o atendimento aos consumidores.

A integração do serviço de medição de energia elétrica com o de água potável também pode ser um objeto de estudo, pois isto ampliaria muito o alcance da rede e a granularidade dos pontos atendidos.

A interferência dos sinais de outras frequências pode ser outro objeto de estudo. Não foi considerado se o sistema interfere em algum serviço disponível no Sapiens Parque ou se algum serviço poderá interferir no sistema de medição.

E por último um estudo da integração do sistema de medição com um sistema de gerenciamento empresarial pode ser interessante para o empreendimento e seus usuários.

REFERÊNCIAS

AGHAEI, Babak. *Using Wireless Network in Water, Electricity and Gas Industry*.***3rd International Conference on Eletronics Computer Technology***, Kanyakumari, 8-10 abr. 2011. V2. p. 14-17.

AHMAD, Muhamad Wassem; et al. *Building energy metering and environmental monitoring – A state-of-the-art review and directions for future research.* ***Energy and Buildings***, 24 de março de 2016. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0378778816302158>. Acesso em: 01/04/2017.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR NM 212: Medidor velocimétrico de água potável fria até 15 m³/h.**Rio de Janeiro, p.19. 1999.

ASSSOCIAÇÃO PYTHON BRASIL. **Python Brasil**. <http://python.org.br/>. Acesso em 03/01/2018.

ATZORI, Luigi; IERA, Antonio; MORABITO, Giacomo. *The Internet of Things: A Survey.* ***Computer Networks****,*Atlanta, 28 outubro 2010. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1389128610001568>. Acesso em 05 março 2017.

BORGES, Luiz Eduardo. **Python para Desenvolvedores**. 2. Ed. Rio de Janeiro: Edição do Autor, 2010. 360p.

BRASIL. Agência Nacional de Águas. **ATLAS Brasil:** Abastecimento Urbano de Água Panorama Nacional Volume I. Brasília, 2010. 70p.

BRASIL. Agência Nacional de Telecomunicações. **Resolução nº 680,**de 27 de junho de 2017. Aprova o Regulamento sobre Equipamentos de Radiocomunicação de Radiação Restrita e altera o Regulamento dos Serviços de Telecomunicações, o Regulamento de Gestão da Qualidade do Serviço de Comunicação Multimídia, o Regulamento do Serviço de Comunicação Multimídia e o Regulamento do Serviço Limitado Privado.

BRASIL. **Lei nº 9.472**, de 16 de julho de 1997. Dispõe sobre a organização dos serviços de telecomunicações, a criação e funcionamento de um órgão regulador e outros aspectos institucionais, nos termos da Emenda Constitucional nº 8, de 1995. Brasília, 1997.

CAO,Liting; TIAN, Jingwen; LIU, Yanxia. *Remote Real Time Automatic Meter Reading System Based on Wireless Sensor Networks.* ***2008 3rd international Conference on Innovative Computing Information***, Dalian, 18-20 jun. 2008.p. 591.

CASALE, Antonio; LUANA Spadafina; PORCELLI, Alessandro. *A water Meter Reading Middleware for Smart Consumption Monitoring.* ***Workshop on Environmental, Energy and Structural Monitoring Systems (EESMS),*** Bari, 13-14 jun. 2016. p. 1-6.

CRAINIC, Monica Sabina. *A short history of residential water meter part I Mechanical Water Meters with moving parts*. **Installations for Building and Ambiental Comfort Conference XXI – edition,** Timisoara, 18-20 abr. 2012. p. 27-35.

DUKAN, Peter; KOVARI, Attila.*Cloud-based Smart Metering System.* ***14th IEEE International Symposium on Computational Intelligence and Informatics****,* Budapeste, 19-21 nov. 2013. p. 499-502.

EUROPEAN STANDARD.**EN13757-4**: *Communication systems for meters and remote Reading of meter – PART4: Wireless meter readout (Radio Meter Reading for operation in the 868-870 MHz SRD band).* Bruxelas, 41p.

FAO – *Food and Agriculture Organization of the United Nations.* ***AQUASTAT Database.*** Sistema global de informações da água da FAO. Disponível em <http://www.fao.org/nr/water/aquastat/data/query/index.html>. Acesso em: 05 de maio de 2017.

FRANGIPANI, Márcio. **Guias práticos:** técnicas de operação em sistemas de abastecimento de água. Brasília: Secretaria Nacional de Saneamento Ambiental, 2007. 5 v.

GALLO, Victória Zanetti Marçal. **Análise de desempenho: MQTT.** 15 jan. 2018. Disponível em: <http://victorianersd.blogspot.com.br/2018/01/analise-de-desempenho-mqtt.html>. Acesso em: 16/01/2018.

HILLAR, Gastón C. ***MQTT Essentials – A Lightweight IoT Protocol.***1. ed. Birmingham: Packt, 2017. 263p.

IBGE - Instituto Brasileiro de Geografia e Estatística. **O Brasil em Síntese**. Sistema agregador de informações do IBGE sobre os munícipios e estados do Brasil. Disponível em <https://cidades.ibge.gov.br/>. Acesso em: 21 de janeiro de 2018.

KROES, Neelie. *Ethical implications of tomorrow's digital society*.***IERC – Internet of Things European Research Cluster****,* Halifax, 2012. Disponível em <http://www.internet-of-things-research.eu/documents.htm>.Acesso em 03/01/2018.

LACERDA, Flávia. **Arquitetura da Informação Pervasiva:** projetos de ecossistemas de informação da Internet das Coisas. 2015. 226 f. Tese (Doutorado em Ciência da Informação) – Faculdade de Ciência da Informação, Universidade de Brasília. Brasília.

LEE, Shinho; et al. *Correlation Analysis of MQTT Loss and Delay According to QoS Level.* ***The International Conference on Information Networking 2013 (ICOIN)***.Bangkok, 28-30 jan. 2013. p. 714-717.

LEE, Young-Woo; EUN, Seongbae; OH, Seung-Hyueb.*WirelessDigital Water Meter with Low Power Consumption for Automatic Meter Reading.* ***International Conference on Convergence and Hybrid Information Technology - ICHIT 2008.***Daejeon, 28-30 ago. 2008. p. 639-645.

LITOVSKY, Gustavo. ***Tutorials.*** *Electrical Engineering, Embedded Systems and the world that uses them.*18 mai. 2012. Disponível em: <http://glitovsky.com/blog/?page_id=21>. Acesso em: 11 ago. 2017

LUNZER, Horst. *Intelligent Metering*. ***5th IEEE International Conference on Industrial Informatics.*** Viena, 23-27 junho 2007. p. 1215 – 1219.

MIHAJLOVIĆ, Zivorad; et al. *Home Energy Monitoring System Based on Open Source Software and Hardware.* ***17th International Conference on Computer Systems and Technology – ComsSystTech'16****,* Palermo, 23-24 abr. 2016. ACM ICPS. 1164. p. 145-150.

MOHAN, Vivek. *Understanding Wireless M-Bus for the European Smart Meter Market.* **EP&Dee – Electronics Products & Design – Eastern Europe.** nº13, 9, p 14-16, nov, 2015. Disponível em <https://issuu.com/esp2000/docs/epdee_november_0915_digital>. Acesso em: 04/04/2018.

NAYAKA, Raja Jitendra; BIRADAR, Rajashekhar C.*A Survey on wireless network applications in automated public utilities control and management.***Journal of Telecommunication and Information Technology,** Warsaw, 2015.n.3, p. 13-24.

PARSON, John David.***The Mobile Radio Propagation Channel***. 2. Ed. Chinchester: JohnWiley & Sons Ltd, 2000. 418p.

*OASIS - ADVANCING OPEN STANDARDS FOR THE INFORMATION SOCIETY.****MQTT Version 3.1.1***.29 outubro 2014. Disponível em: <http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html#_Toc398718043>. Acesso em: 30/12/2017.

SAPIENS PARQUE. **Nota Técnica nº 02**, de setembro de 2014. Avaliação dos Volumes de Reservação de Águas de Consumo Considerando Diversas Fontes. 2014.

SPINSANTE, Susanna; et al. *Wireless M-Bus Sensor Networks for Smart Water Grids: Analysis and Results.* ***International Journal of Distributed Networks****,* jan. 2104. p. 1-16.

SQUARTINI, Stefano; et al. *Wireless M-Bus Sensor Nodes in Smart Water Grids: the Energy Issue.* ***Fourth International Conference on Intelligent Control and Information Processing (ICICIP).***Beijing, 9-11 jun. 2013. p. 614-619

TI – *Texas Instruments User's Guide.* ***MSP430x5xx and MSP430x6xx Family.*** Texas. 2016. 1189p.

TURCU, Cristina; TURCU, Cornel; GAITAN, Vasile.*An Internet of Things Oriented Approach for Water Utility Monitoring and Control.****Advances in Computer Science***, Praga, 24-26 set. 2012. *Recent Advances in Computer Engineering Series*. 5. p. 175-180.

VERMESAN, Ovidiu. *Europe's IoT Strategic Research Agenda 2012*. ***IERC – Internet of Things European Research Cluster****,* Halifax, 2012. Disponível em <http://www.internet-of-things-research.eu/documents.htm>. Acesso em 03/01/2018.

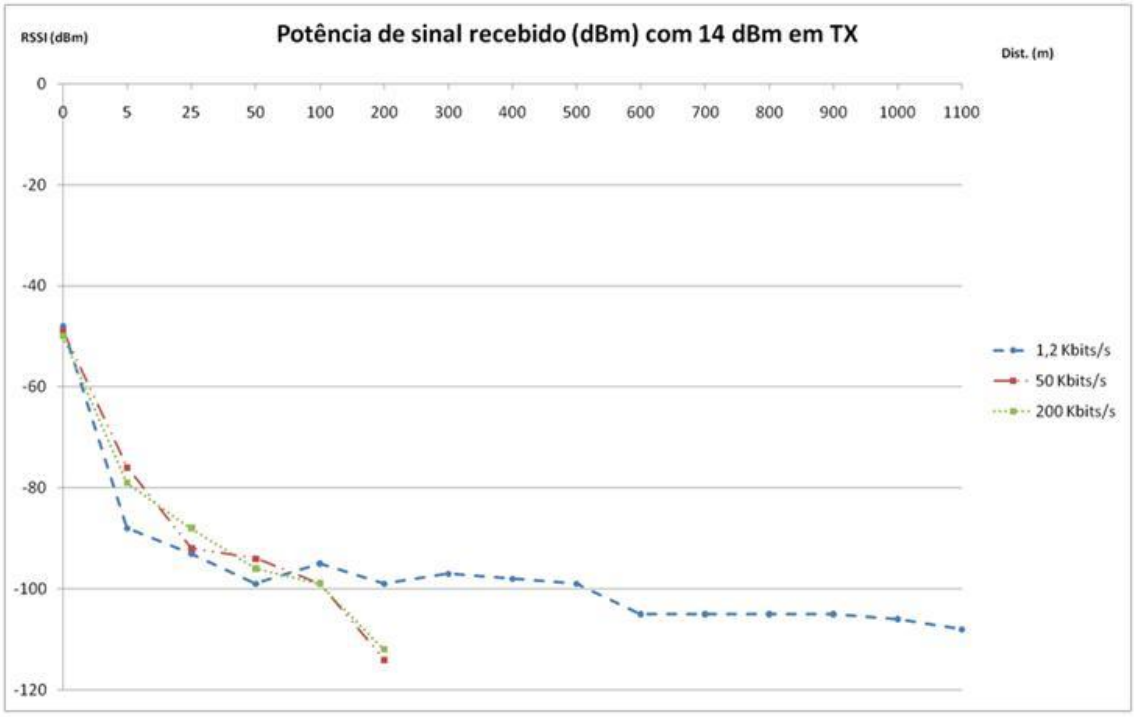
WALLACE, Richard. ***Achieving Optimum Radio Range.*** Dallas: Texas instruments, 2017. 23p.

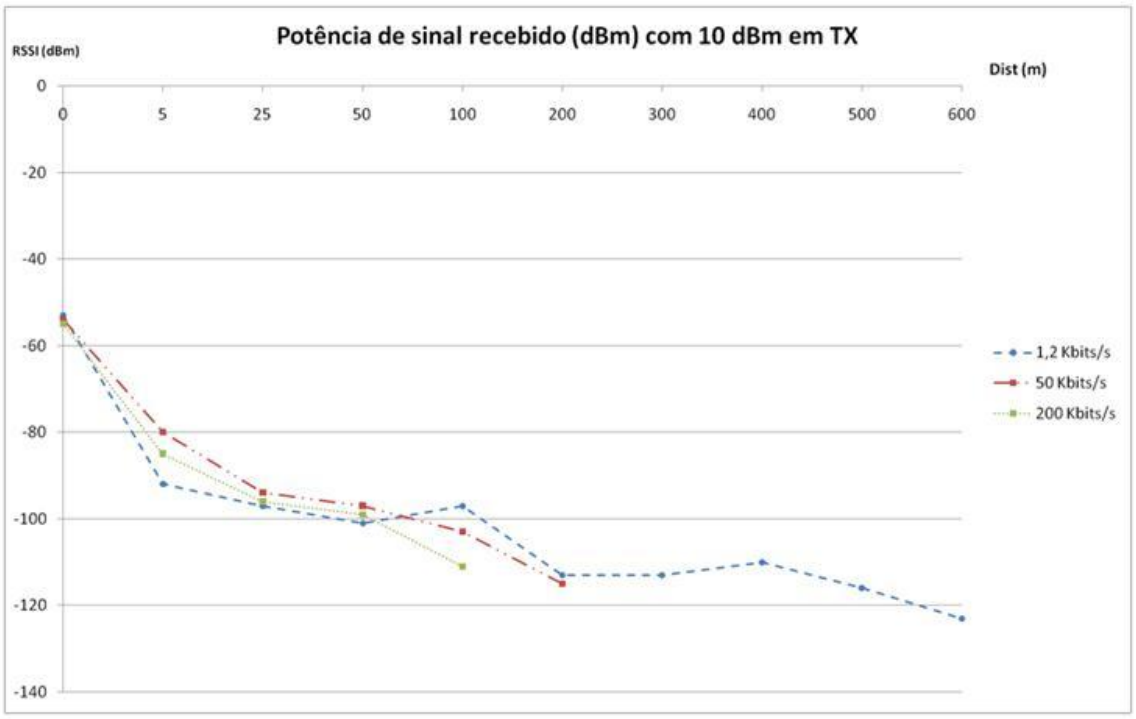
WEISER, Mark. *The Computer for the 21st Century.****Scientific American****,* Nova York, set. 1991. p. 94-104.

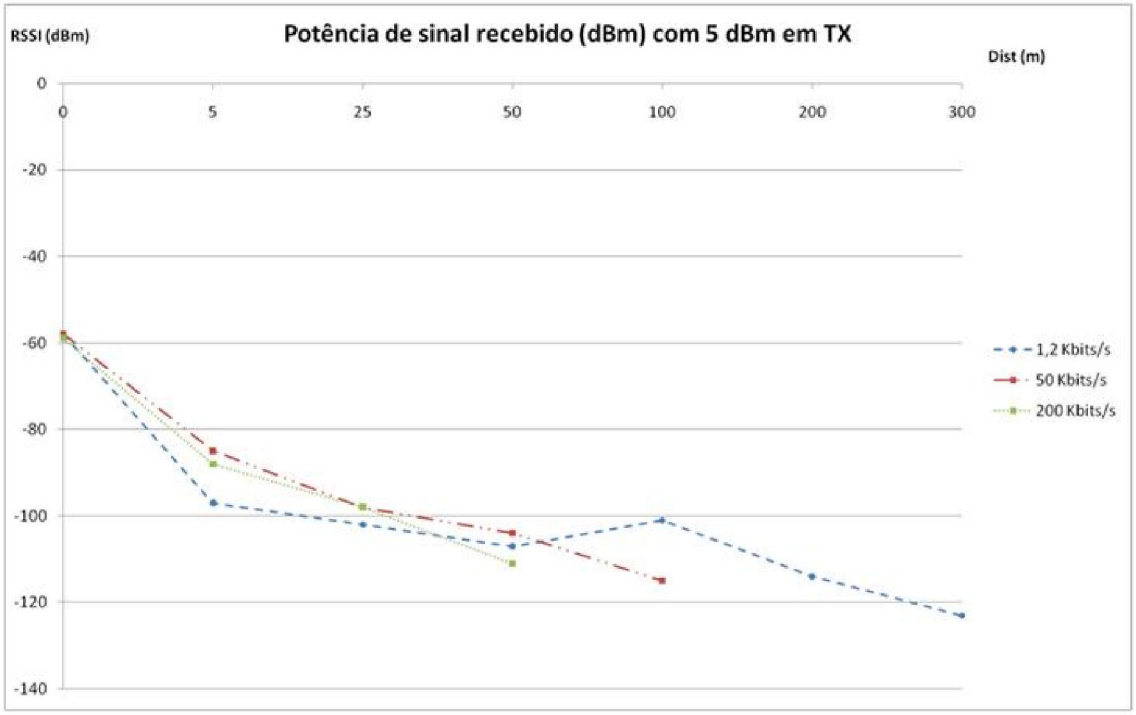
YUAN, Michael. ***Getting to know MQTT***.*Why MQTT is one of the best network protocols for the Internet of Things.* 12 maio 2017. Disponível em: <https://www.ibm.com/developerworks/library/iot-mqtt-why-good-for-iot/index.html>. Acesso em 19/11/2017.

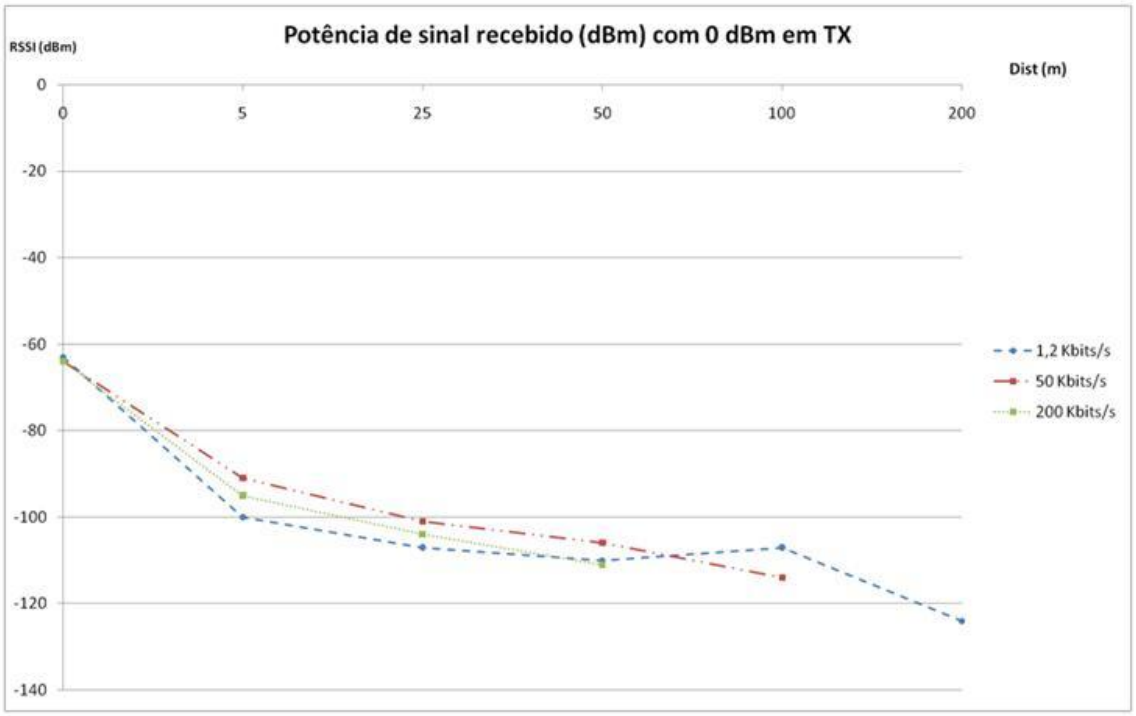
ŽIVIĆ, Nataša; UR-REHMAN,Obaid; RULAND, Christoph. *Evolution of Smart Metering Systems.****23rd Telecommunications Forum TELFOR 2015****,* Belgrado, 24-25 nov. 2015. p. 635-638.

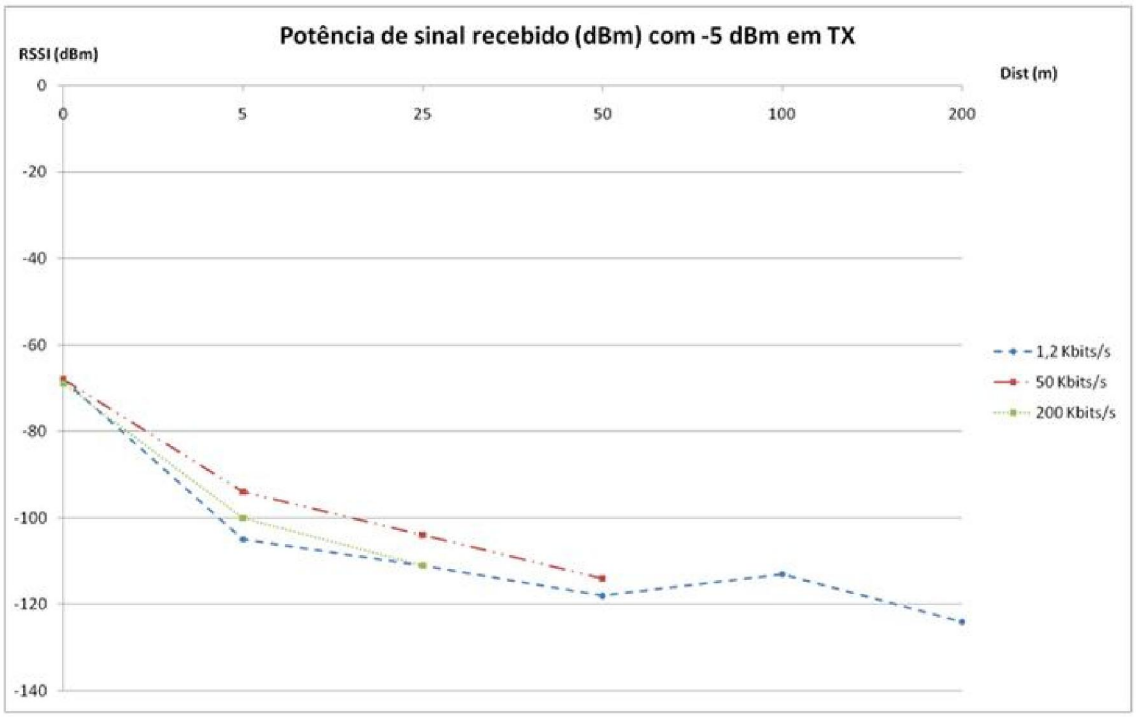
APÊNDICE A – GRÁFICO DOS SINAIS RECEBIDOS PARA TAXA DE TRANSFERÊNCIA E POTÊNCIAS DE SAÍDAS DISPONÍVEIS NO PER TEST

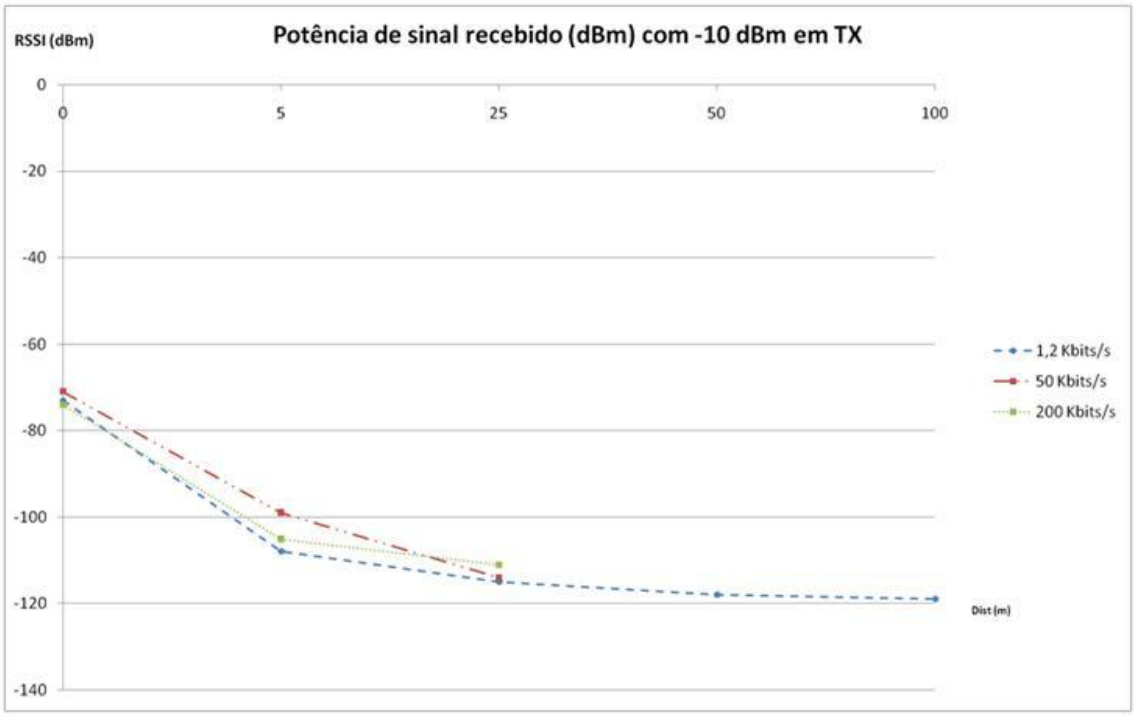


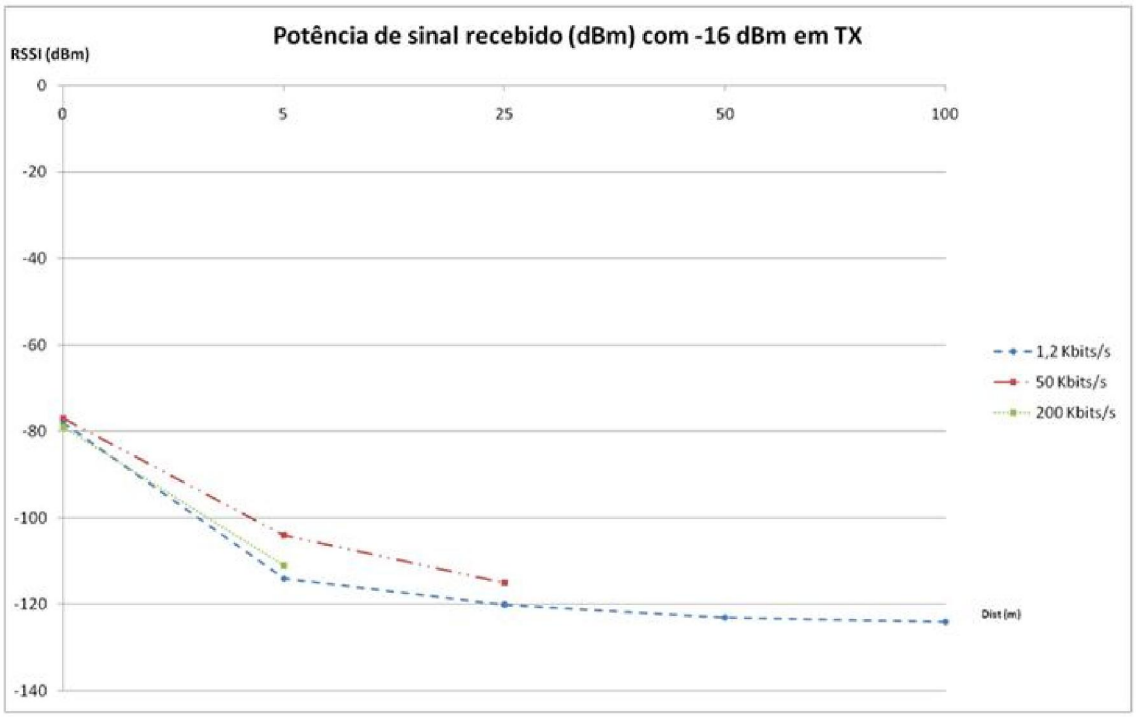



Potência de sinal recebido (dBm) com 5 dBm em TX
RSSI (dBm)
Dist (m)
0
-20
-40
-60
-80
-100
-120
-140
0
5
25
50
100
200
300
1,2 Kbits/s
50 Kbits/s
200 Kbits/s


Potência de sinal recebido (dBm) com 0 dBm em TX
RSSI (dBm)
Dist (m)
0
-20
-40
-60
-80
-100
-120
-140
0
5
25
50
100
200
1,2 Kbits/s
50 Kbits/s
200 Kbits/s


Potência de sinal recebido (dBm) com -5 dBm em TX
RSSI (dBm)
Dist (m)
0
-20
-40
-60
-80
-100
-120
-140
0
5
25
50
100
200
1,2 Kbits/s
50 Kbits/s
200 Kbits/s


Potência de sinal recebido (dBm) com -10 dBm em TX
RSSI (dBm)
Dist (m)
0
-20
-40
-60
-80
-100
-120
-140
0
5
25
50
100
1,2 Kbits/s
50 Kbits/s
200 Kbits/s


Potência de sinal recebido (dBm) com -16 dBm em TX
RSSI (dBm)
0
-20
-40
-60
-80
-100
-120
-140
0
5
25
50
100
Dist (m)
1,2 Kbits/s
50 Kbits/s
200 Kbits/s

APÊNDICE B – TABELA DO TESTE DE MEDIÇÃO – MEDIDOR ULTRASSONICO, MEDIDOR VELOCIMÉTRICO E DASHBOARD.

Leitura (minuto)	Ultrassonico		Velocimétrico		Dashboard (litros)
	Medição(m^3)	Consumo (m^3)	Medição (m^3)	Consumo (m^3)	
0	120,367		0,14		
1	120,379	0,012	0,152	0,012	12
2	120,392	0,013	0,165	0,013	13
3	120,405	0,013	0,178	0,013	13
4	120,418	0,013	0,191	0,013	13
5	120,431	0,013	0,204	0,013	13
6	120,445	0,014	0,218	0,014	14
7	120,458	0,013	0,231	0,013	13
8	120,47	0,012	0,244	0,013	13
9	120,483	0,013	0,257	0,013	13
10	120,496	0,013	0,27	0,013	13
11	120,511	0,015	0,283	0,013	13
12	120,523	0,012	0,296	0,013	13
13	120,536	0,013	0,309	0,013	13
14	120,547	0,011	0,32	0,011	11
15	120,557	0,01	0,33	0,01	10
16	120,566	0,009	0,339	0,009	9
17	120,576	0,01	0,349	0,01	10
18	120,585	0,009	0,358	0,009	9
19	120,594	0,009	0,368	0,01	10
20	120,603	0,009	0,377	0,009	9
21	120,613	0,01	0,387	0,01	10
22	120,622	0,009	0,396	0,009	9
23	120,633	0,011	0,406	0,01	10
24	120,642	0,009	0,415	0,009	9
25	120,652	0,01	0,425	0,01	10
26	120,661	0,009	0,435	0,01	9
27	120,672	0,011	0,445	0,01	10
28	120,681	0,009	0,455	0,01	10
29	120,69	0,009	0,464	0,009	9
30	120,7	0,01	0,474	0,01	10

Leitura	Ultrassonico		Velocimétrico		Dashboard (litros)
	Medição(m³)	Consumo (m³)	Medição (m³)	Consumo (m³)	
31	120,709	0,009	0,483	0,009	9
32	120,719	0,01	0,493	0,01	10
33	120,731	0,012	0,505	0,012	12
34	120,743	0,012	0,517	0,012	12
35	120,754	0,011	0,527	0,01	11
36	120,762	0,008	0,535	0,008	8
37	120,762	0	0,535	0	0
38	120,762	0	0,535	0	0
39	120,762	0	0,535	0	0
40	120,762	0	0,535	0	0
41	120,762	0	0,535	0	0
42	120,762	0	0,535	0	0
43	120,77	0,008	0,543	0,008	8
44	120,776	0,006	0,55	0,007	7
45	120,784	0,008	0,558	0,008	8
46	120,793	0,009	0,565	0,007	7
47	120,799	0,006	0,572	0,007	7
48	120,807	0,008	0,58	0,008	8
49	120,816	0,009	0,589	0,009	9
50	120,826	0,01	0,598	0,009	9
51	120,833	0,007	0,606	0,008	8
52	120,842	0,009	0,615	0,009	9
53	120,842	0	0,615	0	0
54	120,842	0	0,615	0	0
55	120,842	0	0,615	0	0
56	120,842	0	0,615	0	0
57	120,854	0,012	0,627	0,012	12
58	120,866	0,012	0,639	0,012	12
59	120,879	0,013	0,652	0,013	13
60	120,892	0,013	0,665	0,013	13
Total	0,525		0,525		525

ANEXO A – CÓDIGO EM “C” PARA COLETA DE DADOS - MEDIDOR DE ÁGUA.

```
//********************** Código adaptado de http://www.ti.com/lit/swra428 ************************
//
//************************       Adaptado por: Everson Osvanir da Silva         ************************
//
//*************************************************************************************************************
//! @file       cc1200_rx_sniff_mode_tx.c
//! @brief      This program demonstrates Rx Sniff Mode for the 38.4 kbps
//                setting available in SmartRf Studio.
//  The transmitter sends a packet every time a button is pushed and the receiver implements
//  RX Sniff Mode to reduce the current consumption DN511 (http://www.ti.com/lit/swra428)
// explains how the register settings are found.
//
//  Copyright (C) 2013 Texas Instruments Incorporated - http://www.ti.com/
//  Redistribution and use in source and binary forms, with or without modification, are permitted
//  provided that the following conditions are met:
//
//  Redistributions of source code must retain the above copyright notice, this list of
//  conditions and the following disclaimer.
//
//   Redistributions in binary form must reproduce the above copyright  notice, this list of conditionsand
//   the following disclaimer in the  documentation and/or other materials provided with the distribution.
//
//   Neither the name of Texas Instruments Incorporated nor the names of its contributors may be used
//   to endorse or promote products derived from this software without specific prior written permission.
//
// THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
//AND ANY EXPRESS OR IMPLIED WARRANTIES, //INCLUDING, BUT NOT LIMITED TO, THE
//IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
//ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
//LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
//DAMAGES (INCLUDING, BUT NOTLIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
//SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
//CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR
//TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF
//THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
//
//*****************************************************************************
//* INCLUDES

#include <string.h>
#include <stdio.h>
#include "msp430.h"
#include "lcd_dogm128_6.h"
#include "hal_spi_rf_trxeb.h"
#include "cc120x_spi.h"
#include "stdlib.h"
#include "bsp.h"
#include "io_pin_int.h"
#include "bsp_led.h"
#include "projeto_vazao_reg_config.h"
#include "RTC.h"
#include "Globals.h"
```

```
//* DEFINES

#define ISR_ACTION_REQUIRED     1
#define ISR_IDLE                0
#define PKTLEN                  20 // 1 < PKTLEN < 126
#define GPIO3                   0x04
#define GPIO2                   0x08
#define GPIO0                   0x80
#define VAZAO_PIN               0x20

//* LOCAL VARIABLES

static uint8  packetSemaphoreTX, packetSemaphoreRX;
static uint32 packetCounter = 0;

//* STATIC FUNCTIONS

static void initMCU(void);
static void registerConfig(void);
static void createPacket(uint8 txBuffer[], uint8 function, struct dataConsumo *data);
static void radioTxISR(void);
static void updateLcd(uint16 dado1, uint16 dado2);
static void initTX(void);
static void vazaoPinISR(void);
static void initRX(void);
static void runRX(void);
static void runTXbackRX(uint8 buffer[], uint16 sizeBuf);
static void radioRxISR(void);
static void calibrateRCOsc(void);

//* USER

#define set_bit(y,bit)  (y|=(1<<bit))   // set bit x from byte Y
#define clr_bit(y,bit)  (y&=~(1<<bit))  // clear bit x from byte Y
#define cpl_bit(y,bit)  (y^=(1<<bit))   // toggle bit x from byte Y
#define tst_bit(y,bit)  (y&(1<<bit))    // test bit
uint8 timeEventFlag = FALSE;
uint16 x=0, y=0, z=0;
uint8 txBuffer[PKTLEN + 1];
struct dataConsumo vazao;
uint16 contaPulso;
uint8 acumulado = 0;
uint16 recebido;

void init() {

   WDTCTL = WDTPW + WDTHOLD;            // Desliga Watchdog timer
P4DIR |= 0x03;                      // Define pinos 4.0 e 4.1 como saída (0000 0011)
RTC_A_CalendarMode();

   // Initialize MCU and peripherals
initMCU();

   // Write radio registers
registerConfig();
```

```
P2DIR &= ~VAZAO_PIN;                // Garante P2.5 como entrada
P2REN |= VAZAO_PIN;                 // Habilita pullup/pulldown do pino 2.5 (0010 0000)
   P2OUT |= VAZAO_PIN;                 // Garante input "high" nas Portas P2.1 às P2.5

// Connect ISR function to VAZAO_PIN
ioPinIntRegister(IO_PIN_PORT_2, VAZAO_PIN, &vazaoPinISR);

// Interrupt on falling edge
ioPinIntTypeSet(IO_PIN_PORT_2, VAZAO_PIN, IO_PIN_FALLING_EDGE);
   P2REN |= 0x20;                    // Habilita pullup/pulldown do pino 2.5 (0010 0000)

   // Clear ISR flag
ioPinIntClear(IO_PIN_PORT_2, VAZAO_PIN);

   // Enable interrupt
ioPinIntEnable(IO_PIN_PORT_2, VAZAO_PIN);
initTX();
initRX();
}

void main(void) {

contaPulso = 0;
struct dataConsumo vazao;
init();
updateLcd(contaPulso, recebido);
while(1) {

if(timeEventFlag)  {

//1 - Monta pacote: id func data hora consumo
        //2 - calcula CRC e adiciona no fim do pacote
        //3 - Envia pacote pelo Radio
        //4 - Aguarda confirmacao de recebimento do concentrador
acumulado++;
vazao.id = ID;
vazao.dia = RTCDAY;
vazao.mes = RTCMON;
vazao.ano = RTCYEAR;
vazao.hora = RTCHOUR;
vazao.minuto = RTCMIN;
vazao.segundo = RTCSEC;
vazao.pulsos = contaPulso;
vazao.acumulado = acumulado;

createPacket(txBuffer, F_VAZAO, &vazao);
x = sizeof(txBuffer);
runTXbackRX(txBuffer, x);
contaPulso = 0;
timeEventFlag = FALSE;
}
updateLcd(contaPulso, recebido);
}
}
```

```
/*****************************************************************************
*  @fn         __interrupt void RTC_ISR(void)
*  @brief      Interrupcao do RTC.
*/

#if defined(__TI_COMPILER_VERSION__) || defined(__IAR_SYSTEMS_ICC__)
#pragma vector=RTC_VECTOR
__interrupt void RTC_ISR(void)
#elif defined(__GNUC__)
void __attribute__ ((interrupt(RTC_VECTOR))) RTC_ISR (void)
#else
#error Compiler not supported!
#endif {

switch(__even_in_range(RTCIV,16)) {

case RTC_NONE:                      // No interrupts
break;
case RTC_RTCRDYIFG:                   // RTCRDYIFG
     __no_operation();                // every second
break;
case RTC_RTCTEVIFG:                   // RTCEVIFG
     P4OUT ^= 0x02;                   // Interrupção de acordo com o tempo
timeEventFlag = TRUE;
break;
case RTC_RTCAIFG:                    // RTCAIFG
break;
case RTC_RT0PSIFG:                    // RT0PSIFG
break;
case RTC_RT1PSIFG:                    // RT1PSIFG
break;
case 12: break;                     // Reserved
case 14: break;                     // Reserved
case 16: break;                     // Reserved
default: break;
}
}

/*****************************************************************************
*@fn         vazaoPinISR
*@brief      Função de interrupação do pino digital contador de vazao.
*/
static void vazaoPinISR(void) {

contaPulso++;
updateLcd(contaPulso, recebido);
   cpl_bit(P4OUT,0);                    // Alterna bit

   // Clear ISR flag
ioPinIntClear(IO_PIN_PORT_2, VAZAO_PIN);
}

/*****************************************************************************
*@fn         initTX
*@brief      Configura o modo TX para enviar dados.
*/
```

```
static void initTX(void) {

static uint8 marcState;

   // Connect ISR function to GPIO0
ioPinIntRegister(IO_PIN_PORT_1, GPIO0, &radioTxISR);

   // Interrupt on falling edge
ioPinIntTypeSet(IO_PIN_PORT_1, GPIO0, IO_PIN_FALLING_EDGE);

   // Clear ISR flag
ioPinIntClear(IO_PIN_PORT_1, GPIO0);

   // Enable interrupt
ioPinIntEnable(IO_PIN_PORT_1, GPIO0);

// Update LCD
   //updateLcd();

   // Calibrate radio
trxSpiCmdStrobe(CC120X_SCAL);

   // Wait for calibration to be done (radio back in IDLE state)
do {
cc120xSpiReadReg(CC120X_MARCSTATE, &marcState, 1);
   } while (marcState != 0x41);

}

/*******************************************************************************
*   @fn         runTX
*   @brief      Transmits a packet.
*/
static void runTXbackRX(uint8 buffer[], uint16 sizeBuf)  {

       // Write packet to TX FIFO
cc120xSpiWriteTxFifo(buffer, sizeBuf);

       // Strobe TX to send packet
trxSpiCmdStrobe(CC120X_STX);

       // Wait for packet to be sent
while(packetSemaphoreTX != ISR_ACTION_REQUIRED);

       // Clear semaphore flag
packetSemaphoreTX = ISR_IDLE;

       // Set radio in RX Sniff Mode
trxSpiCmdStrobe(CC120X_SWOR);
}
/*******************************************************************************
* @fn          radioTxISR
* @brief       ISR for packet handling in TX. Sets packet semaphore
*              and clears interrupt flag
*/
```

```
static void radioTxISR(void)  {

   // Set packet semaphore
packetSemaphoreTX = ISR_ACTION_REQUIRED;

// Clear ISR flag
ioPinIntClear(IO_PIN_PORT_1, GPIO0);
}

/*****************************************************************************
*   @fn         initRX
*   @brief      Puts radio in RX Sniff Mode and waits for packets.
*/
static void initRX(void) {

uint8 marcState;

   // Connect ISR function to GPIO2
ioPinIntRegister(IO_PIN_PORT_1, GPIO2, &radioRxISR);

// Interrupt on falling edge
ioPinIntTypeSet(IO_PIN_PORT_1, GPIO2, IO_PIN_RISING_EDGE);

   // Clear ISR flag
ioPinIntClear(IO_PIN_PORT_1, GPIO2);

   // Enable interrupt
ioPinIntEnable(IO_PIN_PORT_1, GPIO2);

   // Update LCD
updateLcd(contaPulso, recebido);

   // Calibrate radio
trxSpiCmdStrobe(CC120X_SCAL);

   // Wait for calibration to be done (radio back in IDLE state)
do
{
cc120xSpiReadReg(CC120X_MARCSTATE, &marcState, 1);
      } while (marcState != 0x41);

   // Calibrate the RCOSC
calibrateRCOsc();
}

/*****************************************************************************
*   @fn         runRX
*   @brief      Puts radio in RX Sniff Mode and waits for packets. A packet counter is incremented
*  for each packet received and the LCD is updated
*/
static void runRX(void) {

uint8 rxBuffer[128] = {0};
uint8 rxBytes;

// Infinite loop
```

```
      // Wait for packet to be received
while(packetSemaphoreRX != ISR_ACTION_REQUIRED);

      // Clear semaphore flag
packetSemaphoreRX = ISR_IDLE;

      // Read number of bytes in RX FIFO
cc120xSpiReadReg(CC120X_NUM_RXBYTES, &rxBytes, 1);

// Read all the bytes in the RX FIFO
cc120xSpiReadRxFifo(rxBuffer, rxBytes);

recebido = (rxBuffer[3]<<8) + rxBuffer[4];

// Update LCD
updateLcd(contaPulso, recebido);
}

/*****************************************************************************
*@fn        calibrateRcOsc
*   @brief      Calibrates the RC oscillator used for the eWOR timer. When this
*               function is called, WOR_CFG0.RC_PD must be 0
*/
static void calibrateRCOsc(void)  {

uint8 temp;

   // Read current register value
cc120xSpiReadReg(CC120X_WOR_CFG0, &temp,1);

   // Mask register bit fields and write new values
temp = (temp & 0xF9) | (0x02 << 1);

   // Write new register value
cc120xSpiWriteReg(CC120X_WOR_CFG0, &temp,1);

   // Strobe IDLE to calibrate the RCOSC
trxSpiCmdStrobe(CC120X_SIDLE);

   // Disable RC calibration
temp = (temp & 0xF9) | (0x00 << 1);
cc120xSpiWriteReg(CC120X_WOR_CFG0, &temp, 1);
}

/*****************************************************************************
*   @fn        radioRxISR
*   @brief     ISR for packet handling in RX. Sets packet semaphore and clears interrupt flag
*/
static void radioRxISR(void)  {

   // Set packet semaphore
packetSemaphoreRX = ISR_ACTION_REQUIRED;

runRX();
```

```
    // Clear ISR flag
ioPinIntClear(IO_PIN_PORT_1, GPIO2);
}

/******************************************************************************
*   @fn         initMCU
*   @brief      Initialize MCU and board peripherals
*/
static void initMCU(void)  {

    // Init clocks and I/O
bspInit(BSP_SYS_CLK_8MHZ);

// Init LEDs
bspLedInit();

    // Init buttons
    //bspKeyInit(BSP_KEY_MODE_POLL);

    // Initialize SPI interface to LCD (shared with SPI flash)
bspIoSpiInit(BSP_FLASH_LCD_SPI, BSP_FLASH_LCD_SPI_SPD);

    // Init LCD
lcdInit();

    // Instantiate transceiver RF SPI interface to SCLK ~ 4 MHz
    // Input parameter is clockDivider
    // SCLK frequency = SMCLK/clockDivider
trxRfSpiInterfaceInit(2);

    // Enable global interrupt
    _BIS_SR(GIE);
}

/******************************************************************************
*   @fn         registerConfig
*   @brief      Write register settings as given by SmartRF Studio found in
*               cc1200_rx_sniff_mode_reg_config.h
*/
static void registerConfig(void){

uint8 writeByte;

    // Reset radio
trxSpiCmdStrobe(CC120X_SRES);

    // Write registers to radio
for(uint16 i = 0;
        i < (sizeof(preferredSettings)/sizeof(registerSetting_t)); i++)
{
writeByte = preferredSettings[i].data;
cc120xSpiWriteReg(preferredSettings[i].addr, &writeByte, 1);
    }
}

/******************************************************************************
```

```
*   @fncreatePacket
*   @brief      This function is called before a packet is transmitted. It fills
*               the txBuffer with a packet consisting of a length byte, two
*               bytes packet counter and n random bytes
*
*               The packet format is as follows:
*               |-----------------------------------------------------------|
*               |           |           |           |       |       |
*               | pktLength | pktCount1 | pktCount0 | rndData |.......| rndData|
*               |           |           |           |       |       |
*               |-----------------------------------------------------------|
*                txBuffer[0] txBuffer[1] txBuffer[2]  ..... txBuffer[PKTLEN]
*   @param      Pointer to start of txBuffer
*/
static void createPacket(uint8 buffer[], uint8 function, struct dataConsumo *data) {

uint8 count=0;

switch (function)
   {
case F_VAZAO:  {

count++;
buffer[count++] = data->id;                 // Length byte
buffer[count++] = function;
buffer[count++] = data->dia;                // MSB of packetCounter
buffer[count++] = data->mes;                  // LSB of packetCounter
buffer[count++] = (uint8) (data->ano >> 8);
buffer[count++] = (uint8) data->ano;
buffer[count++] = data->hora;
buffer[count++] = data->minuto;
buffer[count++] = data->segundo;
buffer[count++] = (uint8) (data->pulsos >> 8);
buffer[count++] = (uint8) data->pulsos;
buffer[count++] = data->acumulado;

for(uint8 i = count; i < (PKTLEN + 1); i++) buffer[i] = 0;
crc( buffer, count, &buffer[count], &buffer[count++], 0);
buffer[0] = count;
break;
     }

case F_DATA_HORA: {
break;
     }

default:
break;
}
}

/*****************************************************************************
*   @fn         updateLcd
*   @brief      Updates LCD buffer and sends bufer to LCD module
*/
```

```
static void updateLcd(uint16 dado1, uint16 dado2)  {

unsigned char str[20];

   /* Para modo BCD */
sprintf((char *) str, "%d%d/%d%d/%d%d%d%d %d%d:%d%d:%d%d",  (RTCDAY & 0xF0) >> 4,
RTCDAY & 0x0F,
                                        (RTCMON & 0xF0) >> 4,
                                        RTCMON & 0x0F,
                                        (RTCYEAR & 0xF000) >> 12,
                                        (RTCYEAR & 0x0F00) >> 8,
                                        (RTCYEAR & 0x00F0) >> 4,
                                        RTCYEAR & 0x000F,
                                        (RTCHOUR & 0xF0) >> 4,
                                        RTCHOUR & 0x0F,
                                        (RTCMIN & 0xF0) >> 4,
RTCMIN & 0x0F,
                                        (RTCSEC & 0xF0) >> 4,
                                        RTCSEC & 0x0F);
    // Update LDC buffer and send to screen
lcdBufferClear(0);
lcdBufferPrintString(0, "Medidor 01", 38, eLcdPage0);
lcdBufferSetHLine(0, 0, LCD_COLS - 1, 7);
lcdBufferPrintString(0, (char *) str, 0, eLcdPage2);
lcdBufferPrintString(0, "Pacote enviado:", 0, eLcdPage4);
lcdBufferPrintInt(0, dado2,100, eLcdPage4);
lcdBufferPrintString(0, "Volume(litros):", 0, eLcdPage5);
lcdBufferPrintInt(0, dado1, 100, eLcdPage5);
lcdBufferSetHLine(0, 0, LCD_COLS - 1, 55);
lcdSendBuffer(0);
}
```

ANEXO B – CÓDIGO EM “C” PARA CONCENTRADOR DE DADOS.

```
//********************** Código adaptado de http://www.ti.com/lit/swra428 ************************
//
//************************       Adaptado por: Everson Osvanir da Silva          ************************
//
//*****************************************************************************************************************
//! @file       cc1200_rx_sniff_mode_rx.c
//  The transmitter sends a packet every time a button is pushed and the receiver implements
//  RX Sniff Mode to reduce the current consumption  DN511 (http://www.ti.com/lit/swra428)
// explains how the register  settings are found.
//
//  Copyright (C) 2013 Texas Instruments Incorporated - http://www.ti.com/
//  Redistribution and use in source and binary forms, with or without modification, are permitted
//  provided that the following conditions are met:
//
//  Redistributions of source code must retain the above copyright notice, this list of
//  conditions and the following disclaimer.
//
//   Redistributions in binary form must reproduce the above copyright  notice, this list of conditions and
//   the following disclaimer in the  documentation and/or other materials provided with the distribution.
//
//   Neither the name of Texas Instruments Incorporated nor the names of its contributors may be used
//   to endorse or promote products derived from this software without specific prior written permission.
//
// THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
//AND ANY EXPRESS OR IMPLIED WARRANTIES, //INCLUDING, BUT NOT LIMITED TO, THE
//IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
//ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
//LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
//DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
//SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
//CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR
//TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF
//THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
//
//*****************************************************************************

//* INCLUDES

#include <string.h>
#include <stdio.h>
#include "msp430.h"
#include "lcd_dogm128_6.h"
#include "hal_spi_rf_trxeb.h"
#include "cc120x_spi.h"
#include "stdlib.h"
#include "bsp.h"
//#include "bsp_key.h"
#include "io_pin_int.h"
#include "bsp_led.h"
#include "projeto_vazao_reg_config.h"
#include "uart.h"
#include "Globals.h"

//* DEFINES
```

```
#define ISR_ACTION_REQUIRED     1
#define ISR_IDLE                0
#define PKTLEN                  20 // 1 < PKTLEN < 126
#define GPIO3                   0x04
#define GPIO2                   0x08
#define GPIO0                   0x80

//* LOCAL VARIABLES

static uint8  packetSemaphoreTX, packetSemaphoreRX;
static uint32 packetCounter = 0;
uint16 dado = 0;

//* STATIC FUNCTIONS

static void initMCU(void);
static void registerConfig(void);
static void runTXbackRX(void);
static void radioRxISR(void);
static void updateLcd(uint16 dado1, uint8 buffer[]);
static void calibrateRCOsc(void);
static void initRX(void);
static void initTX(void);
static void runRX(void);
static void runTX(void);
static void radioTxISR(void);
static void createPacket(uint8 txBuffer[], uint16 dado);
void init(void);
void initUARTConfigs(void);
void setSMCLK8MHz(void);

uint8 rxBuf[PKTLEN + 1] = {0};
uint16 enviado = 0, recebido = 0;
unsigned char flagProcessaMsg = FALSE;

/***************** USADO PELA UART ********************************************/
void setSMCLK8MHz();

// UART Port Configuration parameters and registers
UARTConfig cnf;
USCIUARTRegs uartUsciRegs;
USARTUARTRegs uartUsartRegs;

// Buffers to be used by UART Driver
unsigned char uartTxBuf[200];
unsigned char uartRxBuf[200];


/*****************************************************************************
*   @fn         main
*   @brief      Runs the main routine
*/
void main(void) {
unsigned char byteCRC1, byteCRC2;
uint16 x;
```

```
uint8 bufferAux[PKTLEN + 1] = {0};

init();

uartSendDataInt(&cnf,(unsigned char *)"Hello\r\n", strlen("Hello\r\n"));
   // Enter runRX, never coming back
while(1)  {

     //runRX();
updateLcd(enviado, rxBuf);

if(packetSemaphoreRX == ISR_ACTION_REQUIRED)  {

runRX();

for(x = 0; x < PKTLEN + 1; x ++) bufferAux[x] = 0; // limpar buffer auxiliar
for(x = 0; x < rxBuf[0]-2; x++) bufferAux[x] = rxBuf[x];

//testa CRC
crc( bufferAux, bufferAux[0]-2, &byteCRC1, &byteCRC2, 0); //testa o CRC

          if((byteCRC1 != rxBuf[rxBuf[0]])||(byteCRC2 != rxBuf[rxBuf[0] + 1]))  {

// Set radio in RX Sniff Mode
trxSpiCmdStrobe(CC120X_SWOR);

                    }
                    else
                    {
                              flagProcessaMsg = TRUE;
                    }

for(x = 0; x < 10000; x++);
runTXbackRX();
updateLcd(enviado, rxBuf);

       // Clear semaphore flag
packetSemaphoreRX = ISR_IDLE;

}

if(flagProcessaMsg)
}
}

/*****************************************************************************
*@fn         init
*@brief      Inicializa a configuracoes.

*/
void init(void) {

   // Initialize MCU and peripherals
initMCU();

   // Write radio registers
```

```
registerConfig();

initTX();
initRX();

initUARTConfigs();
updateLcd(enviado, rxBuf);
}

/*****************************************************************************
*@fn         initUARTConfigs
*@brief      Inicializa a UART.
*/

void initUARTConfigs(void) {

 // Disable MSP430 Watchdog Timer
        WDTCTL = WDTPW | WDTHOLD;

        // Configure SMCLK to 8MHz
        setSMCLK8MHz();

        //*******UART Specific Configuration*******************
        initUartDriver();

        // Configure UART Module on USCIA1
        cnf.moduleName = USCI_A1;

        // Use UART Pins P5.7 and P5.6
        cnf.portNum = PORT_5;
        cnf.RxPinNum = PIN7;
        cnf.TxPinNum = PIN6;

        // 115200 Baud from 8MHz SMCLK
        cnf.clkRate = 8000000L;
        cnf.baudRate = 115200L;
        cnf.clkSrc = UART_CLK_SRC_SMCLK;

        // 8N1
        cnf.databits = 8;
        cnf.parity = UART_PARITY_NONE;
        cnf.stopbits = 1;

        int res = configUSCIUart(&cnf,&uartUsciRegs);
        if(res != UART_SUCCESS) {

                // Failed to initialize UART for some reason
                __no_operation();
        }

        // Configure the buffers that will be used by the UART Driver.
        // These buffers are exclusively for the UART driver's use and should not be touched
        // by the application itself. Note that they may affect performance if they're too small.

        setUartTxBuffer(&cnf, uartTxBuf, 200);
        setUartRxBuffer(&cnf, uartRxBuf, 200);
```

```
        enableUartRx(&cnf);
        __enable_interrupt(); // Enable Global Interrupts
}

/*****************************************************************************
 * \brief Initializes SMCLK to 8MHz
*/
void setSMCLK8MHz(void)
{
        // Set clock to 8MHz
        UCSCTL3 |= SELREF_2;          // Set DCO FLL reference = REFO
        UCSCTL4 |= SELA_2;            // Set ACLK = REFO
        __bis_SR_register(SCG0);      // Disable the FLL control loop
        UCSCTL0 = 0x0000;             // Set lowest possible DCOx, MODx
        UCSCTL1 = DCORSEL_5;          // Select DCO range 16MHz operation
        UCSCTL2 = FLLD_1 + 244;       // Set DCO Multiplier for 8MHz
                                      // (N + 1) * FLLRef = Fdco
                                      // (244 + 1) * 32768 = 8MHz
                                      // Set FLL Div = fDCOCLK/2
        __bic_SR_register(SCG0);      // Enable the FLL control loop

        // Worst-case settling time for the DCO when the DCO range bits have been changed is
        // n x 32 x 32 x f_MCLK / f_FLL_reference. See UCS chapter in 5xxUG for optimization.
        // 32 x 32 x 8 MHz / 32,768 Hz = 250000 = MCLK cycles for DCO to settle
        __delay_cycles(250000);

        // Loop until XT1,XT2& DCO fault flag is cleared
        do
        {
        // Clear XT2,XT1,DCO fault flags
        UCSCTL7 &= ~(XT2OFFG + XT1LFOFFG + XT1HFOFFG + DCOFFG);
        SFRIFG1 &= ~OFIFG;            // Clear fault flags
        }while (SFRIFG1&OFIFG);       // Test oscillator fault flag
}

/*****************************************************************************
*@fn        initTX
*@brief     Configura o modo TX para enviar dados.
*/
static void initTX(void)
{
static uint8 marcState;

   // Connect ISR function to GPIO0
ioPinIntRegister(IO_PIN_PORT_1, GPIO0, &radioTxISR);

// Interrupt on falling edge
ioPinIntTypeSet(IO_PIN_PORT_1, GPIO0, IO_PIN_FALLING_EDGE);

   // Clear ISR flag
ioPinIntClear(IO_PIN_PORT_1, GPIO0);

   // Enable interrupt
ioPinIntEnable(IO_PIN_PORT_1, GPIO0);

   // Update LCD
```

```
    //updateLcd();

    // Calibrate radio
trxSpiCmdStrobe(CC120X_SCAL);

    // Wait for calibration to be done (radio back in IDLE state)
do {
cc120xSpiReadReg(CC120X_MARCSTATE, &marcState, 1);
   } while (marcState != 0x41);
}

/*****************************************************************************
*   @fn         runTX
*   @brief      Transmits a packet.
*/

static void runTXbackRX(void) {

    // Initialize packet buffer of size PKTLEN + 1
uint8 txBuffer[PKTLEN + 1];

    // Infinite loop
    //while(TRUE) {

        // Create a random packet with PKTLEN + 2 byte packet counter + n x random bytes
enviado++;
createPacket(txBuffer, enviado);

        // Write packet to TX FIFO
cc120xSpiWriteTxFifo(txBuffer, sizeof(txBuffer));

        // Strobe TX to send packet
trxSpiCmdStrobe(CC120X_STX);

        // Wait for packet to be sent
while(packetSemaphoreTX != ISR_ACTION_REQUIRED);

        // Clear semaphore flag
packetSemaphoreTX = ISR_IDLE;

        // Update LCD
        //updateLcd();

        // Set radio in RX Sniff Mode
trxSpiCmdStrobe(CC120X_SWOR);
}

/*****************************************************************************
* @fn          radioTxISR
* @brief       ISR for packet handling in TX. Sets packet semaphore and clears interrupt flag
*/
static void radioTxISR(void) {

    // Set packet semaphore
packetSemaphoreTX = ISR_ACTION_REQUIRED;
```

```
   // Clear ISR flag
ioPinIntClear(IO_PIN_PORT_1, GPIO0);
}

/*****************************************************************************
*   @fn         createPacket
*   @brief      This function is called before a packet is transmitted. It fills the txBuffer with
*    a packet consisting of a length byte, two bytes packet counter and n random bytes
*
*               The packet format is as follows:
*               |-----------------------------------------------------------|
*               |          |          |          |        |      |        |
*               | pktLength | pktCount1 | pktCount0 | rndData |.......| rndData|
*               |          |          |          |        |      |        |
*               |-----------------------------------------------------------|
*                txBuffer[0] txBuffer[1] txBuffer[2]  ..... txBuffer[PKTLEN]
*
*   @param       Pointer to start of txBuffer
*   @return      none
*/
static void createPacket(uint8 txBuffer[], uint16 dado) {

txBuffer[0] = PKTLEN;                       // Length byte
txBuffer[1] = (uint8) (packetCounter >> 8);   // MSB of packetCounter
txBuffer[2] = (uint8) packetCounter;          // LSB of packetCounter
txBuffer[3] = (uint8) (dado >> 8);
txBuffer[4] = (uint8) dado;

for(uint8 i = 5; i < (PKTLEN + 1); i++) txBuffer[i] = 0;
}

/*****************************************************************************
*   @fn         initRX
*   @brief      Puts radio in RX Sniff Mode and waits for packets.
*/
static void initRX(void)
{
uint8 marcState;

   // Connect ISR function to GPIO2
ioPinIntRegister(IO_PIN_PORT_1, GPIO2, &radioRxISR);
// Interrupt on falling edge
ioPinIntTypeSet(IO_PIN_PORT_1, GPIO2, IO_PIN_RISING_EDGE);

   // Clear ISR flag
ioPinIntClear(IO_PIN_PORT_1, GPIO2);

   // Enable interrupt
ioPinIntEnable(IO_PIN_PORT_1, GPIO2);

   // Update LCD
updateLcd(enviado, rxBuf);

   // Calibrate radio
trxSpiCmdStrobe(CC120X_SCAL);
```

```
   // Wait for calibration to be done (radio back in IDLE state)
do {
cc120xSpiReadReg(CC120X_MARCSTATE, &marcState, 1);
   } while (marcState != 0x41);

   // Calibrate the RCOSC
calibrateRCOsc();

   // Set radio in RX Sniff Mode
trxSpiCmdStrobe(CC120X_SWOR);
}

/*****************************************************************************
*   @fn         runRX
*   @brief      Puts radio in RX Sniff Mode and waits for packets. A packet counter
*               is incremented for each packet received and the LCD is updated
*/
static void runRX(void) {

   //uint8 rxBuffer[PKTLEN + 1];
uint8 rxBytes;
   //unsigned char str[PKTLEN + 1];
uint8 i = 0;

for(i = 0; i < (PKTLEN + 1); i++) rxBuf[i] = 0;

       // Wait for packet to be received
while(packetSemaphoreRX != ISR_ACTION_REQUIRED);
       // Clear semaphore flag
       //packetSemaphoreRX = ISR_IDLE;

       // Read number of bytes in RX FIFO
cc120xSpiReadReg(CC120X_NUM_RXBYTES, &rxBytes, 1);

// Read all the bytes in the RX FIFO
cc120xSpiReadRxFifo(rxBuf, rxBytes);

       //uartSendDataInt(&cnf, rxBuf+1, rxBuf[0]);
uartSendDataInt(&cnf, rxBuf, rxBuf[0]+1);
}

/*****************************************************************************
*   @fn         calibrateRcOsc
*   @brief      Calibrates the RC oscillator used for the eWOR timer. When this
*               function is called, WOR_CFG0.RC_PD must be 0
*/
static void calibrateRCOsc(void) {

uint8 temp;

   // Read current register value
cc120xSpiReadReg(CC120X_WOR_CFG0, &temp,1);

   // Mask register bit fields and write new values
temp = (temp & 0xF9) | (0x02 << 1);
```

```
    // Write new register value
cc120xSpiWriteReg(CC120X_WOR_CFG0, &temp,1);

    // Strobe IDLE to calibrate the RCOSC
trxSpiCmdStrobe(CC120X_SIDLE);

    // Disable RC calibration
temp = (temp & 0xF9) | (0x00 << 1);
cc120xSpiWriteReg(CC120X_WOR_CFG0, &temp, 1);
}

/*******************************************************************************
*   @fn         radioRxISR
*   @brief      ISR for packet handling in RX. Sets packet semaphore and clears interrupt flag
*/
static void radioRxISR(void) {

    // Set packet semaphore
packetSemaphoreRX = ISR_ACTION_REQUIRED;

    // Clear ISR flag
ioPinIntClear(IO_PIN_PORT_1, GPIO2);
}
/*******************************************************************************
*   @fn         initMCU
*   @brief      Initialize MCU and board peripherals
*/
static void initMCU(void) {

    // Init clocks and I/O
bspInit(BSP_SYS_CLK_8MHZ);

    // Init LEDs
bspLedInit();

    // Init buttons
    //bspKeyInit(BSP_KEY_MODE_POLL);

    // Initialize SPI interface to LCD (shared with SPI flash)
bspIoSpiInit(BSP_FLASH_LCD_SPI, BSP_FLASH_LCD_SPI_SPD);

    // Init LCD
lcdInit();

    // Instantiate transceiver RF SPI interface to SCLK ~ 4 MHz
    // Input parameter is clockDivider
    // SCLK frequency = SMCLK/clockDivider
trxRfSpiInterfaceInit(2);

    // Enable global interrupt
    _BIS_SR(GIE);
}

/*******************************************************************************
*   @fn         registerConfig
*   @brief      Write register settings as given by SmartRF Studio found in
```

```
*                  cc1200_rx_sniff_mode_reg_config.h
*/
static void registerConfig(void) {

uint8 writeByte;

   // Reset radio
trxSpiCmdStrobe(CC120X_SRES);

   // Write registers to radio
for(uint16 i = 0;
       i < (sizeof(preferredSettings)/sizeof(registerSetting_t)); i++) {
writeByte = preferredSettings[i].data;
cc120xSpiWriteReg(preferredSettings[i].addr, &writeByte, 1);
   }
}

/*******************************************************************************
*   @fn         updateLcd
*   @brief      Updates LCD buffer and sends buffer to LCD module
*/
static void updateLcd(uint16 dado1, uint8 buffer[]) {

   // Update LDC buffer and send to screen
lcdBufferClear(0);
lcdBufferPrintString(0, "Concentrador2", 0, eLcdPage0);
lcdBufferSetHLine(0, 0, LCD_COLS - 1, 7);
lcdBufferPrintString(0, "Recebido:", 0, eLcdPage3);
lcdBufferPrintString(0, buffer, 80, eLcdPage3);
lcdBufferPrintString(0, "Enviado:", 0, eLcdPage4);
lcdBufferPrintInt(0, dado1, 80, eLcdPage4);
lcdBufferPrintString(0, "RX", 0, eLcdPage7);
lcdBufferSetHLine(0, 0, LCD_COLS - 1, 55);
lcdBufferInvertPage(0, 0, LCD_COLS, eLcdPage7);
lcdSendBuffer(0);
}
```

ANEXO C – CÓDIGO EM “C” RTC.

```
//**********************      Autor: Everson Osvanir da Silva          **********************

/*
 * RTC.c
 *
*  Created on: 21 de ago de 2017
 *      Author: eosri
 */
#include "RTC.h"

void RTC_A_CounterMode()
{
// Setup RTC Timer
RTCCTL01 = RTCTEVIE + RTCSSEL_2 + RTCTEV_0;   // Counter Mode, RTC1PS, 8-bit ovf
                            // overflow interrupt enable
RTCPS0CTL = RT0PSDIV_2;                                 // ACLK, /8, start timer
RTCPS1CTL = RT1SSEL_2 + RT1PSDIV_3;                 // out from RT0PS, /16, start timer
}

void RTC_A_CalendarMode()
{
// Configure RTC_A
 RTCCTL01 |= RTCTEVIE + RTCRDYIE + RTCBCD + RTCHOLD + RTCMODE + RTCTEV_1;
                            // RTC enable, RTC hold
                    // RTCBCD => disable BCD mode
                                        // enable RTC read ready interrupt
                                    // enable RTC time event interrupt
                                        // RTCTEV_0 => enable RTC time event minute changed
                                        // RTCTEV_1 => enable RTC time event hour changed

RTCYEAR = 0x2018;                       // Year
RTCMON = 0x02;                          // Month = 0x04 = April
RTCDAY = 0x11;                          // Day = 0x05 = 5th
RTCDOW = 0x07;                          // Day of week = 0x01 = Monday
RTCHOUR = 0x18;                         // Hour = 0x10
RTCMIN = 0x20;                          // Minute = 0x32
RTCSEC = 0x00;                          // Seconds = 0x45

RTCADOWDAY = 0x2;                       // RTC Day of week alarm = 0x2
RTCADAY = 0x20;                         // RTC Day Alarm = 0x20
RTCAHOUR = 0x10;                        // RTC Hour Alarm
RTCAMIN = 0x23;                         // RTC Minute Alarm

RTCCTL01&= ~(RTCHOLD);                  // Start RTC calendar mode
}
```