Universidade Federal do Rio Grande do Norte
Instituto Metrópole Digital

SmartMetropolis – Plataformas e Aplicações para Cidades Inteligentes

WP3 – Sensoriamento

Entregável 02

Natal-RN, Brasil
Julho de 2016

Equipe Técnica

Docentes

Prof. Dr. Ivanovitch Medeiros Dantas da Silva (Coordenador) – Instituto Metropóle Digital
Prof. Dr. Allan de Medeiros Martins - Departamento de Engenharia Elétrica
Prof. MSc Antonio Wallace Antunes Soares - Instituto Metrópole Digital
Prof. MSc Eduardo Nogueira Cunha - Instituto Metrópole Digital
Prof. Dr. Gustavo Girão Barreto da Silva - Instituto Metrópole Digital
Prof. Dr. Julio Cesar Paulino de Melo - Instituto Metrópole Digital
Prof. MSc Wellington Silva de Souza - Intituto Metrópole Digital

Discentes

Ciro Martins Pinto
Danilo Mikael Costa Barros
Leonardo Augusto de Aquino Marques
Juliette de Paula Felipe de Oliveira
Sillas Samyr Silveira de Moura

Conteúdo

1 Introdução

1.1 Objetivos e Histórico das Ações

O grupo de trabalho WP-03, nomeado de Sensoriamento, surgiu com a necessidade de desenvolver soluções de hardware e software embarcados para o Projeto Smart Metropolis. De acordo com o planejamento do projeto, como descrito na Figura 1, a meta no primeiro ano de execução do WP-03 está vinculado ao desenvolvimento de uma solução para o monitoramento de água e energia.

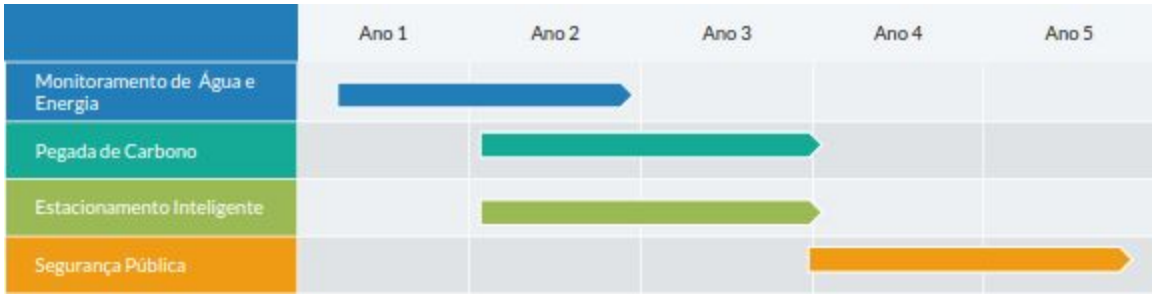


Figura 1. Cronograma das ações a serem desenvolvidas.

Os três primeiros meses de execução dessa solução foram vinculados à descrição geral da arquitetura de desenvolvimento, contendo propostas de entidades de hardware e software. Os resultados dessa etapa foram descritos no **Relatório 01**.

O desenvolvimento da arquitetura descrita no Relatório 01 foi executado nos meses 4, 5 e 6 de execução do projeto, os resultados seguem descritos nesse documento, nomeado de **Relatório 02.** Especificamente, foram desenvolvidas as seguintes ações:

- Protótipo inicial para o monitoramento de energia;
- Software de comissionamento e coleta de dados (descrito em detalhes no relatório referente ao WP-2 Aplicação);
- Projeto de placa de circuito impresso para integração GRSP com o módulo central;
- Integração dos módulos;
- Estudo sobre tecnologias para sensoriamento remoto;
- Levantamento de aspectos referentes à soluções encontradas no mercado (levantado na última reunião de coordendores e acompanhamento do projeto).

2 Soluções comerciais para o monitoramento de energia

2.1 Medidor de energia convencional

Também conhecido por contador Ferraris ou medidor de indução, o medidor de energia convencional foi um dos primeiros a serem utilizados. Seu princípio de funcionamento se baseia na teoria de Galileo Ferraris, onde é possível por associação de duas bobinas, uma para tensão e outra para corrente, saber a energia consumida. Porém, a utilização deste tipo de medidor de energia tem algumas desvantagens, como é o caso da baixa precisão e a necessidade de um usúario para efetuar a leitura manual da energia consumida. A Figura 2 ilustra um típico medidor de energia convencional.

Figura 2. Medidor de energia convencional.

2.2 Medidor de Energia Eletrônico

O medidor de energia eletrônico é a solução mais utilizada atualmente pelas concessionárias de energia elétrica. Apresenta inúmeras vantagens, como por exemplo, maior precisão nas medições, possibilidade de integração com outros sistemas. Apesar de possibilitar a conexão com sistemas externos, suas soluções são tipicamente "caixa preta", apenas os fabricantes possuem conhecimento sobre o seu funcionamento. A Figura 3 descreve um medidor de energia eletrônico convencional.

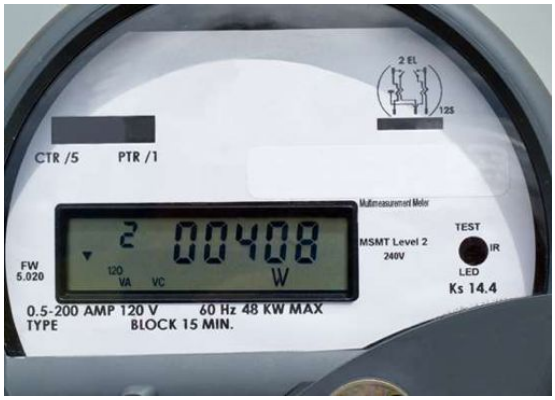


Figura 3. Medidor de energia eletrônico convencional.

2.3 Medidor Inteligente ou Smart Meter

Apesar dos medidores inteligentes ou Smart Meters já estarem disponíveis no mercado global, eles estão em constantes mudanças. Existem inúmeras vantagens dos medidores inteligentes, podem registrar em tempo real, ou próximo do real, o uso da eletricidade e a possibilidade de sua geração, oferecem a possibilidade de ler as informações no local ou remotamente, conectar com dispositivos pré-definidos, ler outros consumos como água e gás e até limitar o consumo pelo medidor inteligente, sendo possível ate cortar o fornecimento de energia. A Figura 4 descreve um típico *Smart Meter* residencial.

Figura 4. *Smart Meter* residencial da empresa GE.

2.4 Circuitos integrados para a medição de energia

Apesar de existirem vários medidores de energia comerciais, a inflexibilidade dessas soluções em integrar sistemas externos impedem que aplicações customizadas para os usúarios finais sejam desenvolvidas.

Um solução para o problema é a construção de um medidor próprio a partir de sensores de prateleira ("off-the-shelf"). Os circuitos integrados ADE7753 e ADE7758 são um exemplo dessa solução (Figura 5). Suas leituras e análises de dados apresentam um nível de acurárcia similar ao medidor de energia eletrônico convencional. Adicionalmente, esses integrados possuem um baixo custo (unidade inferior a R$ 4,00) e permitem a comunicação serial com outros dispositivos eletrônicos. Como desvantagem, apresentam um erro de leitura quando a corrente e tensão não são perfeitamente senoidais. Todavia, esse problema pode ser resolvido através do uso de filtros, justificando assim a sua adoção.

Figura 5. Circuito integrado ADE7753.

3 Descrição do protótipo desenvolvido

Uma visão geral da arquitetura do protótipo desenvolvido para o monitoramento de energia é descrito na Figura 6. A arquitetura é formada por cinco módulos:

- Módulo central (nomeado de placa integradora), representado pela Raspberry Pi;
- Módulo de sensoriamento (shields);
- Módulo de comunicação com a nuvem (wifi ou gprs);
- Ferramenta de comissionamento (aplicativo android);
- Coletor universal (descrito no relatório referente ao WP-2 Aplicação).

A seguir, detalhes técnicos dos referidos módulo serão descritos assim como uma discussão inicial sobre o protótipo desenvolvido.

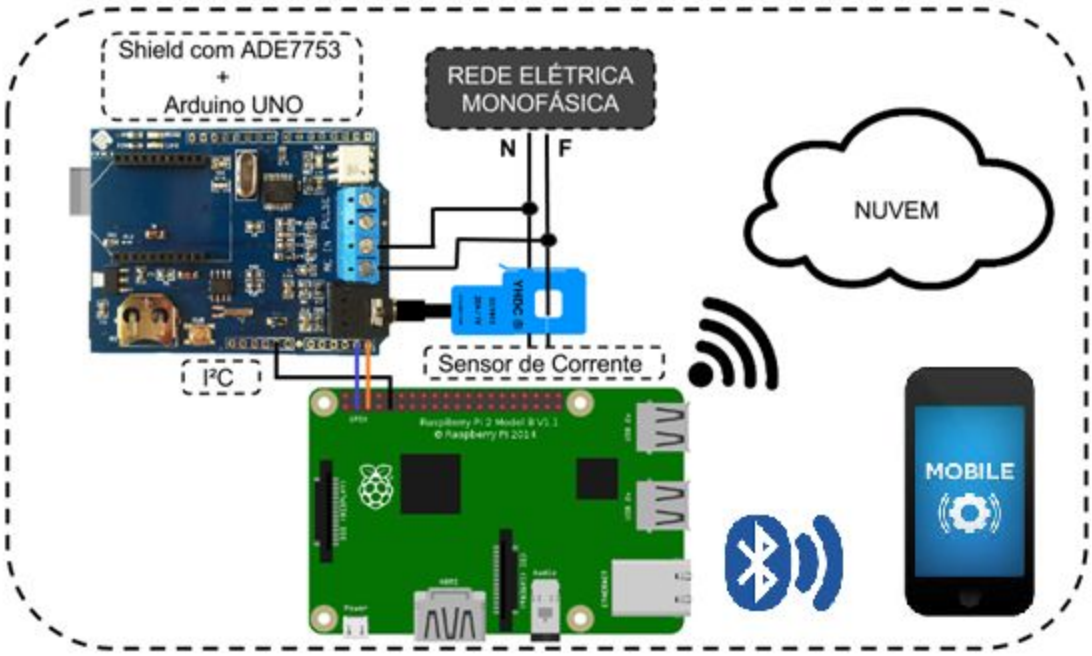


Figura 6. Arquitetura do protótipo para monitoramento de energia.

3.1 Discussão inicial

Durante a fase de planejamento (três primeiros meses de execução do projeto) foi definido a utilização da plataforma Raspberry Pi em detrimento a microcontroladores na composição do **módulo central**. Dessa forma, a integração entre o módulo de comunicação e as demais partes do projeto, que seria realizada em nível de hardware, agora será realizada em nível de software. Essa decisão simplifica o projeto e facilita a integração.

O **módulo de sensoriamento** está sendo desenvolvido para análise de redes monofásicas e baseado no chip ADE7753. Todavia, o produto final será baseado no ADE7758, o qual é mais indicado para rede trifásica. O uso do ADE7753 foi devido o projeto não possuir o ADE7758. A compra deverá ser realizado nas próximas fases de execução. Testes iniciais do módulo de sensoriamento estão sendo realizados a partir do Energy Shield juntamente com a plataforma Arduino. Nesses testes, são enviados para o módulo central as seguintes variáveis: tensão de pico e eficaz, corrente de pico e eficaz, potência aparente, ativa e reativa. A comunicação entre o módulo de sensoriamento e o módulo central é realizado através do protocolo de comunicação I²C.

O **módulo de comunicação** apresenta uma importância fundamental ao projeto, pois é o elemento responsável por comunicação os dados gerados pelo módulo de sensoriamento ao software de monitoramento. De acordo com o planjetamento do projeto, qualquer protocolo de

comunicação pode ser utilizado, todavia, a comunicação wifi e gprs foram adotadas devido apresentarem um desempenho satistatório e amplamente discutidos na literatura e com abertura de adoção na indústria. Ambos os protocolos foram testados nessa etapa do projeto, entretanto os resultados aqui presentes foram descritos utilizando o protocolo wifi devido a placa de circuito impresso do gprs ainda está em desenvolvimento.

O **módulo de comissonamento** é a ferramenta necessária para configurar todos os módulos descritos anteriormente. Uma aplicação android foi desenvolvida para realização de tal tarefa.

Após todos os módulos estarem desenvolvidos, uma integração com o **coletor universal** precisa ser realizada. Esse módulo é responsável também pela integração com o software de monitoramento. Detalhes são descritos no WP-2 Aplicação.

3.2 Módulo de sensoriamento

3.2.1 Comunicação com o módulo central

3.2.1.1 Protocolo I²C

Após um estudo acerca dos protocolos de comunicação entre circuitos integrados mais utilizados, foi decidido a adoção do protocolo I²C (*Inter-Integrated Circuit*). O protocolo foi desenvolvido pela Phillips. É amplamente usado em sistemas embarcados por facilitar a integração de um sistema de controle (mestre) com até 127 sensores e conversores (escravos) em um mesmo barramento. Outro ponto forte do I²C é a utilização de apenas 2 fios para realizar a transmissão de informações: *Serial Data* (SDA), linha bidirecional usada para transmissão de dados, e *Serial Clock* (SCL), linha usada pelo mestre para fornecer a velocidade de transmissão (frequência). Em outras palavras, apenas duas portas dos microcontroladores são ocupadas para comunicação. A Figura 7 descreve uma topologia típica do protocolo de comunicação I²C .

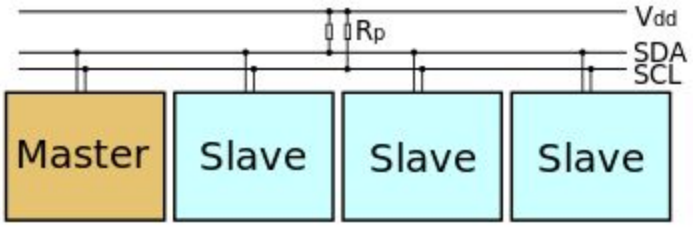


Figura 7. Barramento I²C com um mestre e três escravos.

Adicionalmente, o barramento I²C suporta o modo com múltiplos mestres. Nesse cenário mais de um dispositivo de controle (mestre) pode ser conectado ao barramento e ter acesso aos dados de todos os dispositivos escravos.

Os dispositivos conectados ao barramento I²C possuem um endereço fixo. O formato básico para endereçamento I²C é constituído por 1 byte, sendo 7 bits referentes ao endereço do

dispositivo escravo que o mestre deseja acessar, seguido por 1 bit indicador de leitura/escrita (1 para leitura e 0 para escrita). O mestre se comunica com os dispositivos escravos conforme descrição da Figura 8.

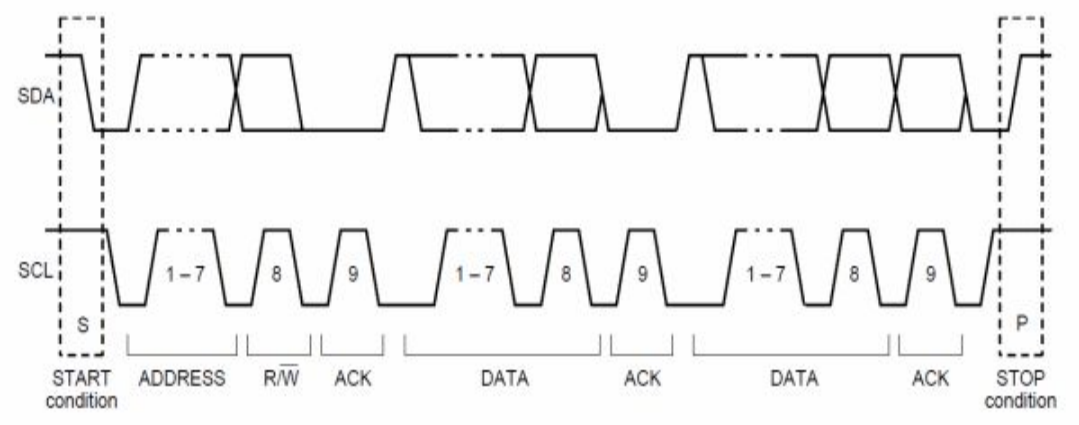


Figura 8. Formato da transmissão I²C.

A transmissão é inicializada pelo mestre através de um start bit. Em seguida, é enviado um byte contendo o endereço do escravo ligado ao barramento I²C (7 bits) seguido do bit indicativo de leitura e escrita (R/W). Todos os escravos conectados ao barramento recebem o start bit, os 7 bits de endereçamento e o bit de leitura/escrita, mas apenas o escravo que possui o endereço enviado pelo mestre responde enviando um bit de reconhecimento (ACK) na linha SDA informando para o mestre que está pronto para receber dados. Depois do ACK, o mestre envia os dados para o escravo (escrita) ou recebe dados do escravo (leitura). A confirmação do recebimento de cada byte é fornecida pelo bit ACK enviado pelo escravo. Se ainda houver dados para enviar/receber, um novo *start bit* é enviado após o ACK e a comunicação continua. Para encerrar a comunicação, o mestre envia um *stop bit* depois do ACK. A comunicação também é encerrada quando o escravo deixa de enviar o ACK para o mestre.

3.2.2 Programação do protocolo I²C no Raspberry Pi

O módulo central é constituído por um Raspberry Pi 2 Modelo B. Esse hardware controla toda a comunicação e, por isso, foi configurado como dispositivo mestre do sistema. O Raspberry Pi terá o controle de todos os escravos conectados ao barramento I²C . O programa mestre foi escrito em Java e, para isso, foi preciso instalar algumas bibliotecas e realizar algumas configurações extras. Primeiramente foi necessário habilitar o I²C para utilização do barramento no Raspberry Pi. Após o I²C ser habilitado, foi preciso instalar a biblioteca WiringPi para dar acesso ao GPIO. A instalação da biblioteca Pi4J foi também necessária para flexibilizar o desenvolvimento de Java na Raspberry Pi.

3.2.2.1 API Java de comunicação I²C mestre

A biblioteca Pi4J é fruto de um projeto que tem por objetivo tornar mais amigável implementações usando Java no Raspberry, ela usa internamente o WiringPi para o acesso de

baixo nível às entradas e saídas da GPIO e sua API possui um package para comunicação usando o protocolo I²C.

A API Java escrita para a comunicação com o módulo central (dispositivo mestre) utiliza as interfaces I2CBus e I2CDevice da Pi4J para criar o barramento de comunicação, criar os dispositivos conectados a esse barramento e ter acesso a métodos de leitura e escrita no barramento. Foram escritos métodos get para leitura dos dados dos sensores. Para essa versão teremos os métodos responsáveis por pegar os valores dos dados fornecidos pelo sensor de energia:

- getVrms(): retorna o valor da tensão RMS lida do sensor de energia conectado ao barramento I²C.
- getIrms(): retorna o valor da corrente RMS lida do sensor de energia conectado ao barramento I²C.
- getVpeak(): retorna o valor da tensão de pico lida do sensor de energia conectado ao barramento I²C.
- getIpeak(): retorna o valor da corrente de pico lida do sensor de energia conectado ao barramento I²C.
- getFrequency(): retorna o valor da frequência lida do sensor de energia conectado ao barramento I²C.
- etActiveEnergy(): retorna o valor da energia ativa lida do sensor de energia conectado ao barramento I²C.
- getApparentEnergy(): retorna o valor da energia aparente lida do sensor de energia conectado ao barramento I²C.
- getReactiveEnergy(): retorna o valor da energia reativa lida do sensor de energia conectado ao barramento I²C.

Cada método utiliza um byte específico para se comunicar com o Arduino (escravo). Por exemplo, quando o Arduino recebe o byte 00110001, retorna o valor Vrms; quando o Arduino recebe o byte 00110010, retorna o valor Irms; para cada dado do sensor existe um byte de requisição específico. Os métodos get enviam o byte para o dispositivo usando o método writeData() e o dispositivo escravo, depois que recebe o byte de requisição, fica aguardando a requisição de leitura (readData()) para enviar para o mestre o dado solicitado.

Além dos métodos get, há um método que escaneia o barramento em busca de dispositivos conectados (i2cBusScanner()) e pode ser usado para saber se os dispositivos estão conectados e prontos para comunicação.

3.2.3 Programação do protocolo I²C no Arduino

O *Energy Shield* usa a plataforma Arduino UNO para aquisição dos dados. Para a comunicação I²C com o Raspberry foi utilizada a biblioteca *Wire* do Arduino no modo escravo. O Arduino possui um endereço fixo (0x04) e esse endereço é entendido pelo Raspberry Pi como endereço do sensor de energia. Foram escritos dois métodos: um para receber dados e outro para enviar. O método receber dados guarda o byte que indica qual dado o mestre está solicitando e o método enviar dados envia dados requisitados pelo mestre. O mestre que decide quando a

comunicação vai iniciar e terminar e o Arduino fica apenas aguardando os comandos no barramento.

3.3 Módulo de comunicação

3.3.1 Tecnologias para sensoriamento remoto

Atualmente, temos disponíveis no mercado várias tecnologias que são, ou podem ser aplicadas a área de sensoriamento remoto. Nesta área, a característica mais importante é que, o consumo de energia seja o menor possível. Sendo assim, o uso de sistemas WiFi (baseados no padrão IEEE 802.11) e 3G/GPRS ficam restritos por consumirem mais do que outros concorrentes como o Bluetooth Smart (ou Bluetooth Low Energy - BLE), dispositivos que utilizam o padrão WirelessHART ou mesmo o ZigBee. Entretanto, os dois primeiros têm vantagens sobre os últimos por apresentarem um alcance de comunicação superior, especialmente o 3G/GPRS, e uma maior taxa de transferência de dados além de não dependerem de nos vizinhos para operarem, podendo ser instalado um único dispositivo em uma área isolada dos demais ou servirem como centralizadores dos dados, reunindo informações de uma rede de sensores ao redor e enviando todos os dados para uma central (Figura 9).

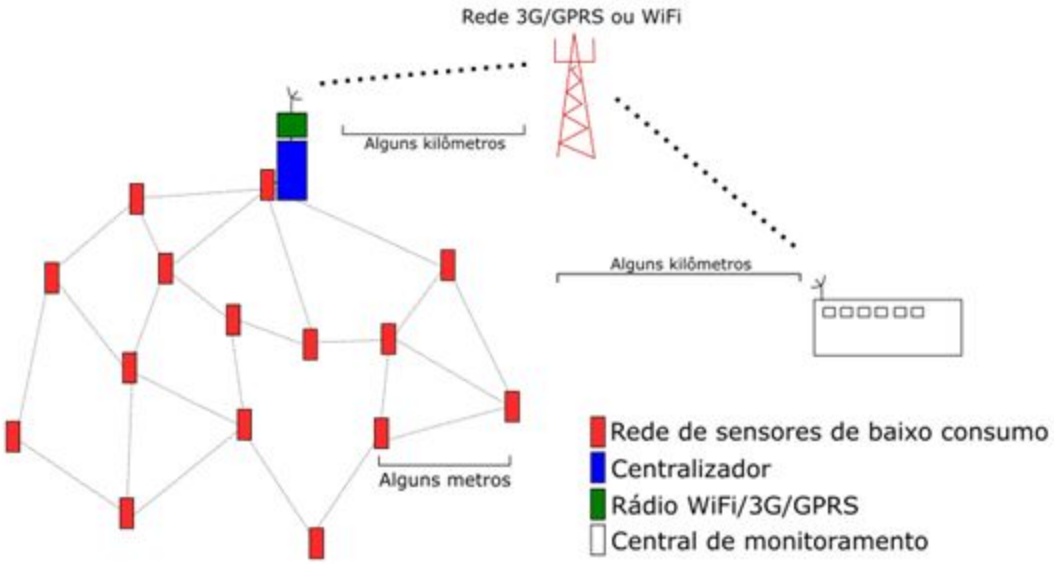


Figura 9: Exemplo de uma rede de sensoriamento sem fio com um centralizador de dados.

A seguir, será apresentada uma lista de tecnologias de comunicação que está em uso ou apenas em fase experimental, mas apresentam grande potencial para o monitormento remoto.

- WiFi padrão IEEE 802.11 – É o padrão mais comum, normalmente utilizado para acesso a internet em dispositivos móveis e automação residencial. Já vem embutido em algumas plataformas de desenvolvimento como o Raspberry Pi 3, Intel Edison,

Arduino Yun, dispositivos da linha ESP8266, ou Shields compatíveis com o Arduino. O alcance do sinal depende do aparelho e o conjunto de antenas utilizados. Para dispositivos móveis, normalmente se consegue distâncias que variam dos 70 a 200 metros em campo aberto, porém, se utilizadas antenas direcionais e bem posicionadas, como no caso de access points para uso em áreas rurais, pode-se obter alcances superiores a 50 Km.

- WiMAX (Worldwide Interoperability for Microwave Access, padrão IEEE 802.16) - É uma tecnologia de redes sem fio criada em 2002 por um consórcio de empresas para oferecer uma opção para as redes metropolitanas. A tecnologia opera em frequências na faixa que vai dos 2 aos 11GHz (para utilizações que não necessitam está na linha de visada) e 10 a 66Ghz (que necessitam está na linha de visada) e deve possuir um alcance teórico de até 50Km, sendo assim uma tecnologia de cobre a chamada "última milha" (Ligação com o usuário final). A AES Eletropaulo está utilizando em seu projeto piloto de Smart Grid, que segundo a empresa, é um dos maiores do país. A conexão será utilizada para, entre outras coisas, comandar os relés e comunicar a necessidade de troca do equipamento em caso de falha.
- RF Mesh - Mais especificamente "Narrowband Radio Frequency", É uma tecnologia para a implementação de redes sem fio e baixo consumo de bateria, onde cada nó da rede tem um alcance de até 50Km, teoricamente. Também está sendo utilizada pela AES Eletropaulo em seu projeto piloto de Smart Grid, no caso, a tecnologia está sendo utilizada para a interligação dos medidores inteligentes nas residências.
- WiFi, padrão IEEE 802.15.4 – É a base para ISA100.11, WirelessHART, MiWi, ZigBee e para o protocolo Thread. Esse padrão é utilizado em diferentes equipamentos e para diferentes propósitos que vão desde a automação e monitoramento na indústria, no caso do WirelessHart ou ISA100.11 até a interligação de aparelhos eletrodomésticos e aparelhos voltados para o comércio, no caso da Thread ou ZigBee.
- Bluetooth Classic – É utilizado na interligação de aparelhos como smartphones e computadores e notebooks, além de ser utilizado em uma grande quantidade de dispositivos que vão desde autofalantes sem fio até mouses e teclados sem fio. Pode ser encontrado na forma de pequenos adaptadores que podem ser conectados a portas USB ou em módulos de desenvolvimento como o módulo HC-05 ou HC-06. A plataforma de desenvolvimento Edison da Intel, também traz consigo um módulo bluetooth.
- Bluetooth Low Energy – É um novo padrão Bluetooth para dispositivos que não necessitam de altas taxas de transferências de dados. Apesar de operar de forma semelhante ao Bluetooth Clássico, as duas versões não são compatíveis entre si. Por isso a necessidade de que alguns novos modelos de Smartphones possuírem ambos. O BLE, como é chamado, foi criado para aplicações em sensoriamento remoto, ou dispositivos com baixo consumo de energia. Também pode ser encontrado em algumas plataformas de desenvolvimento como o nRF52 da Nordic Semiconductor, o BlueFruit da Adafruit entre outras.
- Outros dispositivos que operam na frequência de rádio – Existem muitos outros dispositivos que não utilizam um padrão específico de operação mas são utilizados em diversos locais para envio de comandos simples, alguns exemplos são o módulo nRF24L01 da Nordic Semiconductor que possui a capacidade de formar pequenas

redes ponto a ponto, assim como o módulo APC220. Ambos utilizam uma modulação GFSK para os dados.

- IrDA - Infravermelho de curto alcance – Esta tecnologia utiliza a luz infravermelha para o transporte da informação, por este motivo, é necessário que os aparelhos estejam em sua linha de visada. Este método é pouco utilizado hoje em dia, mas já foi incorporado por diversos dispositivos desde computadores pessoais, impressoras até telefones celulares para troca de informações como mídias e dados diversos. Atualmente a tecnologia Bluetooth substituiu o IrDA e este quase está em desuso. O mais próximo que pode ser encontrado atualmente é a utilização em controles remotos de eletrodomésticos.
- LASER – A comunicação utilizando lazer infravermelho ainda está em desenvolvimento, e assim como a IrDA, a comunicação por LAZER necessita que os aparelhos estejam na linha de visada, e apesar de o seu alcance está limitado a alguns quilômetros e sofrer interferências de eventos atmosféricos como neblina, poeira e chuvas. Sem estes inconvenientes, em 2001, a Agência Espacial Europeia (ESA) realizou um experimento em que dois satélites mantiveram comunicação por LAZER, estando orbitando a Terra a 832Km e 31.000Km, ou seja, uma distância superior a 30.000Km. O Lazer também é utilizado para conexão do chamado "último quilômetro" (Ligação com o usuário final que a operadora não cobre com a sua rede), fornecendo internet a uma taxa de transferência que varia entre 10Mbit/s a 1,25 Gbit/s ou até mais, dependendo da tecnologia do diodo lazer utilizado.

3.3.2 Alimentação do módulo GPRS

A alimentação dos módulos GPRS SIM800L, utilizados no projeto, é projetada para operar com uma voltagem entre 3,7V e 4,2V. Dessa forma, foi necessário projetar métodos para alimentação dos módulos, já que as tensões disponíveis na Raspberry Pi são de 5V ou de 3,3V.

Os testes iniciais com o módulo GPRS foram feitos utilizando a plataforma Arduino Uno e a alimentação realizada utilizando um diodo de código 1N4007 entre a saída de 5V do Arduino e o módulo GPRS conforme descrito na Figura 10.

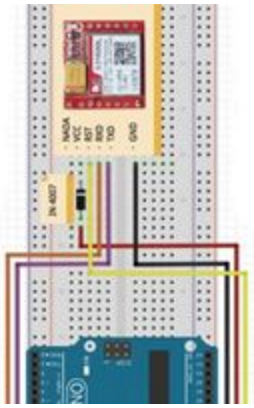


Figura 10. Alimentação através de um diodo.

Sabe-se que um diodo de silício apresenta uma queda de tensão de aproximadamente 0,7V entre seus terminais quando diretamente polarizado, dessa forma, o módulo recebia uma tensão de aproximadamente 4,3V (5V – 0,7V), e adicionando mais um diodo do mesmo tipo em série, a tensão irá cair para 3,6V, aproximadamente. Em ambos os casos, a tensão obtida está ligeiramente acima ou abaixo da suportada e como o módulo é bastante sensível quanto a tensão de operação, quando este era alimentado com 4,3V, a mensagem "over-voltage" era exibida no monitor serial, e em seguida, o módulo era desligado automaticamente como medida de proteção. Por outro lado, quando alimentado com 3,6V, o módulo apresentou instabilidade no seu funcionamento, vindo a reiniciar sempre que uma corrente maior era exigida (o módulo consome uma corrente de pico de 2A).

Para contornar o problema e se iniciar os estudos e testes, a alimentação do módulo foi feita, temporariamente, utilizando uma pilha (uma única célula) de íons de lítio de 1200mAh. Posteriormente, a pilha foi substituída pelo circuito integrado LM31, o qual representa um regulador de voltagem ajustável que suporta o pico de corrente do módulo, e se mostrou a solução ideal para o caso. A Figura 11 descreve a solução adotada.

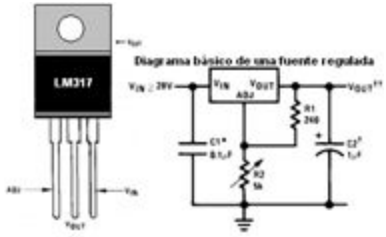


Figura 11. Regulador de tensão LM317.

3.3.3 Comunicação com o módulo central

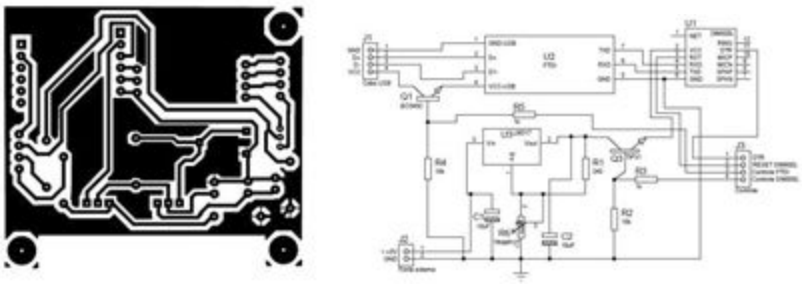
Após os testes iniciais para familiarização com o GPRS e o conjunto de comandos AT, utilizado por ele, foi iniciada a integração com o módulo central (Raspberry Pi).

Haja vista que a comunicação do GPRS é realizada por meio de comunicação serial (RS-232) e o Raspberry Pi já utiliza sua única interface serial para a comunicação com o console do sistema, foi necessário a utilização de um módulo adaptar USP para Serial. A solução encontrada foi buscar um produto de prateleira ("off-the-shelf"), como por exemplo, o PL2303 (Figura 12).

Figura 12. Módulo conversor USB/Serial PL2303.

A implementação de uma porta serial definida por software também foi estudada, porém, constatou-se que não seria possível, já que o sistema operacional do Raspberry Pi não é um Sistema de Tempo Real (RTOS – Real-Time Operating System). Para facilitar o manuseio e a

integração física, foi desenhada uma placa de circuito para acomodar todos os componentes (Figura 13). Reforçamos que essa placa ainda não foi prototipada, isso será realizado nos próximos entregáveis.



(a) (b)

Figura 13. Placa de circuito impresso (a) e seu respectivo esquemático eletrônico.

A operação da placa contendo o módulo GPRS é feita utilizando dois softwares que operam em conjunto: connect.py que utiliza os comandos AT descritos na documentação oficial do módulo GPRS e é responsável pela comunicação serial, e GPRS.java que recebe e trata os dados para posterior envio para a nuvem.

3.3.3.1 Comandos AT

Os comandos AT utilizados para configuração e operação do módulo GPRS Sim800L, são enumerados em documento disponível no site do fabricante. Os comandos necessários para uso no projeto podem ser: configuração inicial do módulo GPRS, modo de comunicação (voz, SMS, dados), operação do rádio FM, bluetooth e Geolocalização.

A Tabela 1 ilustra a sequência de comandos AT necessários para o “setup” da comunicação GPRS (palavras iniciadas por um “ponto” são as respostas dos comandos passados):

Comando	Descrição
AT+SAPBR=3,1,"Contype","GPRS"	Define os parâmetros (código 3) para conexões baseadas em IP. Os parâmetros "Contype" e "GPRS" indicam que vamos utilizar a conexão GPRS. Resposta esperada: OK
AT+SAPBR=3,1,"APN","www"	O mesmo que o anterior, o parâmetro '3' indica que estamos apenas setando as informações necessárias. Porém, os parâmetros "APN" e "www" indicam que estamos informando a APN da operadora. Resposta esperada: OK
AT+SAPBR=1,1	O código '1', desta vez, indica que estamos abrindo a conexão. Resposta esperada: OK
AT+SAPBR=2,1	O código 2 indica que estamos solicitando as informações da conexão estabelecida, o comando retorna o IP recebido. Resposta esperada: +SAPBR: 1,1,"100.68.176.136" OK

Tabela 1. Sequência de Comandos AT para o setup do módulo GPRS.

A Tabela 2, descreve a sequência de comandos relativos à configuração e envio dos dados para o endereço especificado. Segue a ordem correta:

Comando	Descrição
AT+HTTPINIT	Inicializa o serviço HTTP Resposta esperada: OK
AT+HTTPPARA="CID",1	Estabelece os parâmetros HTTP, a sintaxe é: <parâmetro>,<valor>. Aqui setamos um identificador do perfil da conexão. Resposta esperada: OK
AT+HTTPPARA="URL","http://url_de_destino/"	Informamos a URL de destino. Resposta esperada: OK
AT+HTTPPARA="CONTENT","application/x-www-form-urlencoded"	Informamos o tipo de encriptação que vamos utilizar. No caso, vamos utilizar o método POST, para enviar dados em forma de texto. Resposta esperada: OK
AT+HTTPDATA=size,X	Informamos o tamanho dos dados que vamos enviar (size) e o tempo (X) em milisegundos que o módulo GPRS irá esperar até que receba os dados Resposta esperada: DOWNLOAD A partir de agora, o módulo irá esperar pelos dados. Resposta esperada após passar os dados: OK
AT+HTTPACTION=1	Envia os dados via método POST, configurado acima. Resposta esperada se não houverem erros: OK, +HTTPACTION: 1,302,0 Resposta esperada caso ocorra algum erro no envio dos dados: +HTTPACTION: 1,603,0 OBS: No caso, o código 603 nos informa que houve um erro de DNS, para outros códigos, ver o manual.
AT+HTTPTERM	Fecha a conexão HTTP. Resposta esperada: OK

Tabela 2. Sequência de comandos relativos à configuração e envio dos dados para o endereço especificado.

Adicionalmente, o módulo suporta comandos utilizados para outras operações, como: desligar o módulo GPRS, acionar o modo de economia de energia etc. Na Tabela 3 estão listados alguns comandos com tais características.

Comando	Descrição
AT+CSCLK=2	Ativa o modo de economia de energia automático. Se depois de 5 segundos, não houve nenhuma operação no módulo GPRS, o modo de economia de energia passa a operar. O módulo acorda após receber algum sinal de interrupção vindo da porta serial ou se recebeu alguma mensagem de texto "SMS". Resposta esperada: OK
AT+CPOWD=1	Desliga de forma segura o módulo GPRS. Resposta esperada: NORMAL POWER DOWN
AT+IPR=X	Fixa o baund-rate da comunicação serial no valor X passado.
AT+COPS=?	Retorna uma lista de todas as operadoras disponíveis.
AT+COPS?	Retorna a operadora registrada pelo cartão SIM inserido.

Tabela 3. Comandos de propósito gerais suportados pelo módulo de GPRS.

3.3.3.2 Software gerenciador da porta serial - connect.py

A primeira versão do módulo de comunicação que controlará a placa GPRS e se comunicará com as demais partes do projeto, foi escrito em linguagem Python em conjunto com Java. Foi projetado uma arquitetura modular, separando as funcionalidades da comunicação serial, gestão dos dados e comunicação com os demais módulos do projeto. Essa abordagem facilita a manutenção e posteriores alterações do código.

O controle da porta serial foi feito em Python utilizando a biblioteca PySerial. Esta parte é responsável pela detecção automática do dispositivo conectado a uma das portas USB do Raspberry Pi. Uma vez detectado o dispositivo, é estabelecida uma conexão com o módulo GPRS. Em seguida, o módulo é inicializado com os parâmetros apropriados para a conexão e mantém a espera dos dados, desliga-o ou o mantendo em estado de baixo consumo de energia, dependendo da taxa de atualização dos dados definidas inicialmente nas configurações. Outras funções são: desligar o módulo GPRS e/ou o conversor USB como medida de economia de energia, preparar os dados para enviar para a nuvem, além de checar se toda a comunicação foi bem-sucedida.

O software é executado através da linha de comando. E até o momento, aceita seis tipos de parâmetros: "config", "starthttp", "post", "endhttp", "powerdown" e "testSerial". Também trabalha com o arquivo auxiliar "data", onde busca os dados e informações necessárias. Na próxima versão, haverá mais um arquivo auxiliar denominado "config", onde estará todos os

dados de configuração como APN, configurações da porta serial, URL de destino, além de outras informações importantes.

Descrição do arquivo auxiliar: neste arquivo estão todos os dados que serão enviados e também a url de destino. Está organizado da seguinte forma:

- **url** http://endereço_de_destino
- **data**
- Dados que serão enviados.

Descrição dos parâmetros:

- **config**: Checa se o módulo GPRS está respondendo, Inicializa o modo de comunicação GPRS, atribui um APN e uma URL de destino para os dados (recuperado do arquivo "data") e recebe um endereço IP que é armazenado no arquivo de mesmo nome.
- **starthttp**: Inicializa a comunicação http. Os dados são encriptados na forma "application/x-www-form-urlencoded" e podem ser enviados tanto utilizando os métodos GET ou POST.
- **post**: Este comando envia os dados armazenados em "data" utilizando o método POST para o endereço especificado no mesmo arquivo.
- **endhttp**: Se não há mais dados a enviar, encerra-se a conexão http utilizando este parâmetro.
- **powerdown**: Desliga o módulo GPRS para economizar energia. Após este comando, está previsto na próxima versão, o controle de um pino de saída digital do Raspberry Pi para o controle de ligar e desligar a alimentação do módulo através de um transistor de potência Tip31 ou outro semelhante que suporte a corrente do módulo.
- Outros parâmetros que serão implementados na próxima versão: Controle da interrupção do módulo GPRS através do pino denominado DTR e o controle da alimentação do módulo conversor USB/Serial através de um transistor do tipo BC548 ou semelhante (Transistor NPN de uso geral).

Todos os parâmetros retornam "OK" ou "ERRO + mensagem de erro", caso algo tenha saído errado.

3.3.3.3 Software gerenciador dos dados - GPRS.java

Este software é o responsável por toda a gerência da comunicação, trata-se de uma classe Java que recebe e organiza todos os dados necessários para a correta operação do software connect.py, que por sua vez, controla a comunicação serial com o módulo GPRS.

Métodos públicos da classe GPRS.java:

- **GPRS(String url, String apn)**: É o construtor. Recebe como parâmetros a url de destino e o APN da operadora do cartão Sim.
- **String scan()**: Aguarda 10 segundos pela conexão do módulo conversor USB/Serial e retorna o nome da porta utilizada (Na próxima versão, a conexão será automática).
- **boolean startGPRS()**: Prepara o módulo para a conexão GPRS (Deve ser executado primeiro, sempre ao ligar, depois de um RESET ou sleep), este método utiliza o

parâmetro “config” de connect.py e retorna um boolean que indica se a conexão foi feita corretamente ou se foi encontrado algum problema.

- **boolean loadData(String data)**: Prepara os dados para enviar para o módulo GPRS (Deve ser utilizado sempre antes de enviar os dados para a nuvem). Este método modifica o arquivo “data”.
- **boolean postOne()**: Este método inicializa o modo HTTP, envia os dados para a nuvem e termina a conexão. É ideal para o envio de dados ocasionalmente (vários segundos de intervalo), sendo que é obrigatório o reset do GPRS ou deixa-lo em modo sleep entre cada envio.
- **void sleepGPRS(boolean state)**: (Ainda não implementada, será utilizada na próxima atualização) Este método é o responsável por entrar e sair do modo de economia de energia do módulo GPRS.
- **void resetGPRS()**:(Ainda não implementada, será utilizada na próxima atualização) Realiza um reset no módulo GPRS, apaga toda a configuração e será necessário realizar uma nova inicialização do módulo, este método pode ser utilizado em casos de travamentos ou se observar algum funcionamento incomum durante a operação.
- **boolean initHTTP()**: Inicializa a conexão HTTP. Este método utiliza o parâmetro “starthttp” de connect.py.
- **boolean post()**: Após a inicialização da conexão http com initHTTP(), este método envia os dados previamente carregados por 'loadData()' para a nuvem.
- **boolean closeHTTP()**: Finaliza a conexão http, se não há mais dados a serem enviados e se deseja entrar em modo de economia de energia, deve-se utilizar este método antes.

Todos os métodos que retornam um valor booleano “boolean” retornam “true” se o comando foi efetuado com sucesso ou “False” se algo deu errado.

3.4 Ferramenta de comissionamento

O sistema de comissionamento para medição de energia é um aplicativo desenvolvido para a plataforma Android. Ele tem como funcionalidades comissionar e gerenciar um hardware capaz de monitorar a medição de energia. A aplicação se comunica com o hardware utilizando o protocolo de comunicação 802.15.1 (Bluetooth). Também é capaz de enviar comandos de configuração não só para efeitos de hardware, mas também, configurar toda a comunicação com hardware com nuvem onde serão armazenadas as informações. O aplicativo basicamente envia e recebe dados, fazendo com que o hardware e a medição de energia em si estejam em constante gerenciamento. Uma vez comissionado o hardware, a aplicação é capaz de coletar as informações salvas assim como editá-las caso necessário. O uso desse sistema é apenas feito em loco, já que sua principal função é comissionar o hardware para o funcionamento a distância.

3.4.1 Metodologia de desenvolvimento

Para o desenvolvimento do projeto foi-se utilizado um computador com capacidade para o desenvolvimento de aplicativos Android, assim como toda a plataforma necessitada. Primeiro foi-se desenvolvido um protótipo inicial que se assemelhava a um terminal de envio de comandos, esta parte serviu como base de teste da comunicação bluetooth com o hardware. Uma

vez que a comunicação estava funcionando da maneira correta, foi dado inicio ao desenvolvimento de novas funcionalidades, como o comissionamento da nuvem, que foi capaz de configurar o armazenamento de dados do hardware na nuvem para futuros diagnosticos. Por fim, as funcionalidades de configuração do hardware foram implementadas, como a escolha do dado, que tipo de informação coletar, escolher qual sensor utilizar, cadastrar sensores e dentre outras mais. Com a aplicação finalizada, seguiu-se para a etapa de testes, onde foi possível testar a aplicação na prática e em ambiente real. A etapa de testes foi bastante útil para encontrar problemas e consertá-los.

3.4.2 Funcionalidades suportadas

Abaixo seguem imagens da aplicação e suas etapas de usabilidade e configuração.

- Cadastro de Nuvem: Essa é a primeira etapa a ser feita e é a primeira tela que deve aparecer ao se iniciar a aplicação. Nela será configurada a comunicação com o servidor(Núvem) para que os dados coletados sejam enviados e armazenados para uma futura análise das informações.
- Menu de Opções: Após a configuração da Núvem, um menu com algumas opções estará disponível para que seja feito o comissionamento tanto da coleta das informações como também dos sensores e fontes de coleta de dados. A primeira opção fonte Coleta serve para atualizar as informações, ou seja, para que as configurações sejam rebuscadas e atualizadas. A segunda opção Cadastrar Fonte serve para que se possa cadastrar uma fonte de coleta onde serão acoplados sensores que irão prover dados para Núvem. A terceira opção Núvem serve para que se possa editar as informações salvas do comissionamento da núvem.
- Cadastrar Fonte: Aqui acontece quando se escolhe essa opção do menu de opções. Uma janela abrirá pedindo algumas informações da fonte de coleta para que ela possa ser cadastrada. O primeiro campo refere-se ao nome da fonte de coleta. O segundo campo é o identificador, ou seja, um valor que irá sempre representar aquela fonte em específico. No último campo, que é de seleção, o usuário irá escolher que tipo de fonte de coleta ele quer, seja Energia, água ou alguma outra opção que esteja disponível.
- Menu de Fontes: Uma vez adicionada uma fonte, o usuário poderá contar com um menu contendo a lista de todas as fontes já cadastradas podendo selecionar para ver os sensores já cadastrados na fonte ou adicionar outros sensores, que chamamos de TAGS.
- Cadastro de TAGS: Quando se adiciona uma fonte de coleta, é preciso habilitar os sensores que estão conectados a essa fonte, no caso, as TAGS. Para isso, basta segurar o botão item referente a fonte de coleta na lista apresentada e irá aparecer uma janela onde se poderá entrar com os dados da TAG (sensor). O primeiro campo é o nome da TAG. O segundo campo é uma seleção do tipo do sensor, no caso, ele pode ser NUMERIC (numérico) ou TEXT (texto).

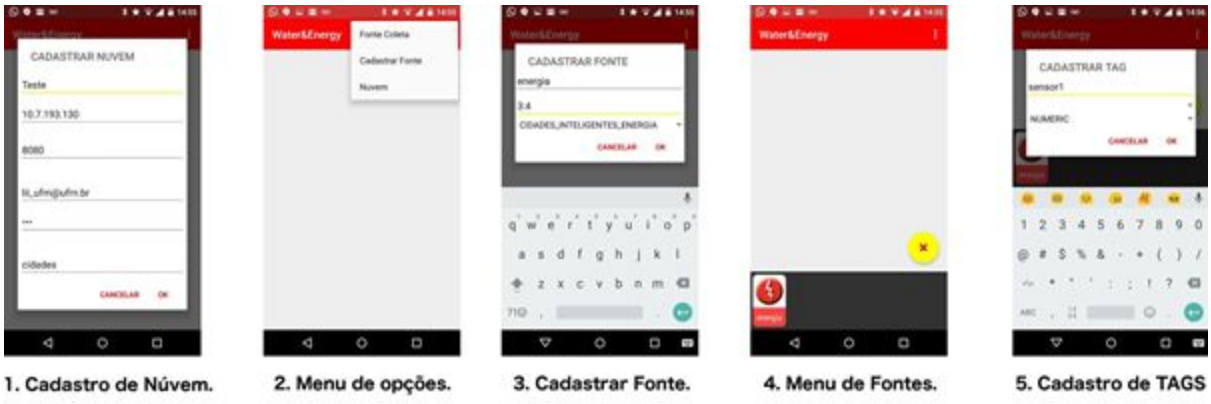


Figura 14. Telas de configuração da aplicação de comissionamento.

3.5 Módulo central e coletor universal

Como mencionado anteriormente, o módulo central é formado para uma Raspberry Pi. O software de integração do módulo central com o coletor universal para extração dos dados está descrito no Relatório 02 do WP-2 Aplicação.

4 Integração dos módulos

Para validar todos os conceitos e abordagens propostas foi desenvolvido um protótipo para realizar a medição de energia. Como teste, foi utilizado um carregador de celular e monitorado algumas variáveis referentes ao seu consumo de energia. A Figura 15 mostra o protótipo do hardware de medição de energia.

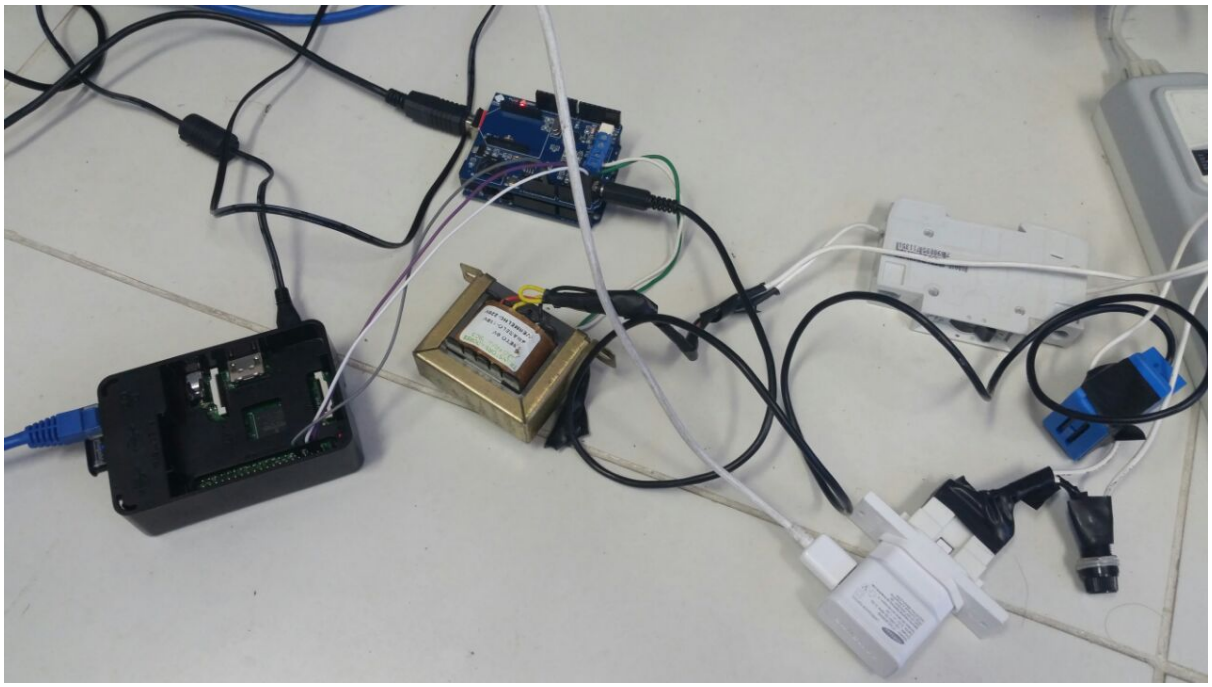


Figura 15. Protótipo do Hardware de medição de energia.

O primeiro protótipo faz o monitoramento da tensão e corrente RMS, tensão e corrente de pico, frequência e potência ativa, reativa e aparente de qualquer dispositivo elétrico conectado à ele. Como descrito anteriormente, todos os dados de energia são adquiridos no sensor de energia (Arduino UNO + *Energy Shield* + sensor de corrente) e passados para o módulo central (Raspberry Pi 2 Modelo B) via protocolo de comunicação I²C.

Para validar a aquisição dos valores monitorados, uma comunicação serial foi conectada ao Arduino e verificada através do *Serial Monitor* conforme descrito na Figura 16.

Adicionalmente, o link https://goo.gl/BEFU5I contém um vídeo mostrando a configuração do hardware de medição de energia através do aplicativo Android, no qual são configuradas as informações da nuvem, da fonte de dados e das tags monitoradas.

Após a configuração do hardware e do software de coleta, os dados passam a ser enviados para a nuvem, podendo ser visualizados através do BR-PlantViewer e BR-PlantHistorian, como pode ser visualizado no vídeo https://goo.gl/RbPbM1 (detalhes são descritos no Relatório 02 do WP-2 Aplicação). Percebam que apesar da Raspberry está conectada em uma interface ethernet (cabo azul), sua comunicação com a aplicação de monitoramento foi realizada através do protocolo wifi (dongle conectado na USB). O módulo GPRS apenas será testado no próximo entregável pois a placa expansora precisa ser ainda prototipada.

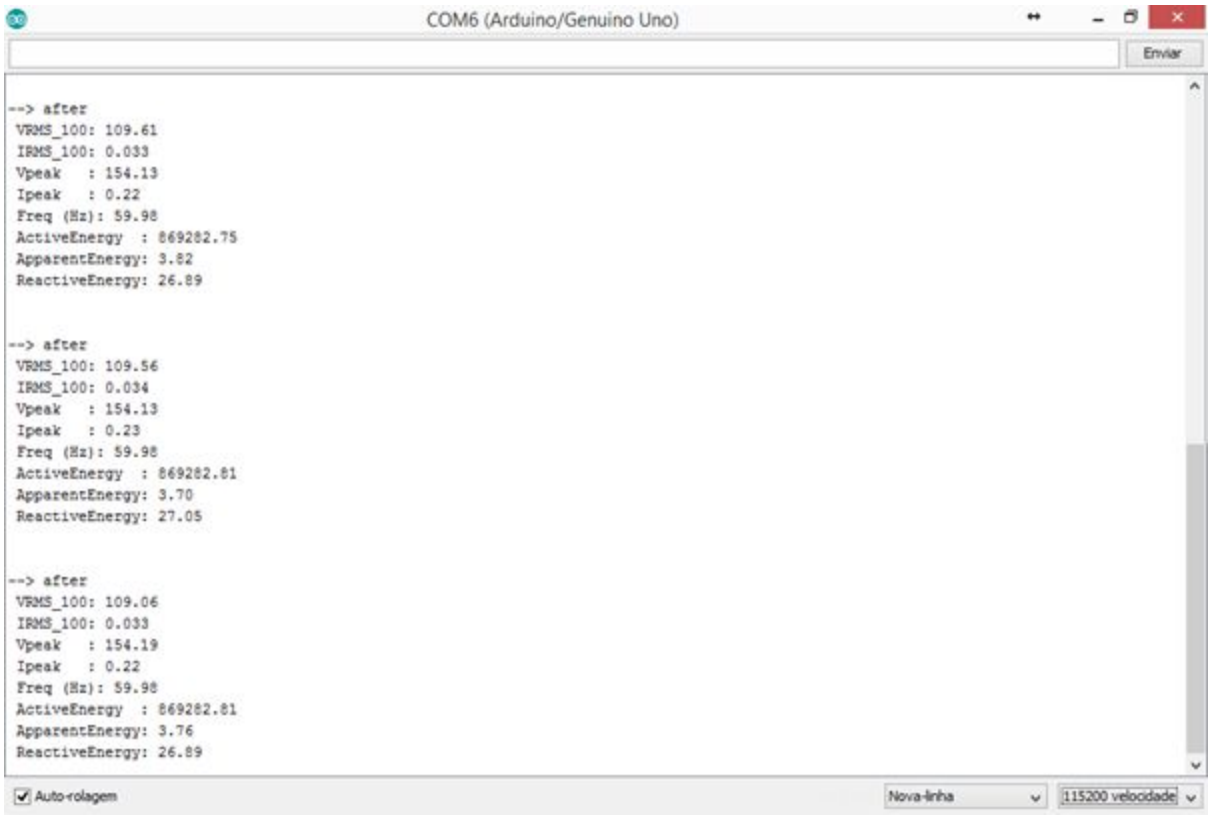


Figura 16. Aquisição de dados do sensor de energia, mostrado via Serial monitor do Arduino.

5 Próximos trabalhos

Após a integração de todas as partes, hardware de medição de energia, driver de aquisição de dados, software de coleta de dados, aplicativo Android e sistema na nuvem, os seguintes passos são objetivos do próximo entregável:

- Finalização do desenvolvimento do módulo de comunicação GPRS;
- Testes e melhorias no coletor universal à fim de torná-lo mais robusto;
- Melhorar a usabilidade do aplicativo Android de configuração do hardware.
- Miniaturizar o hardware de medição de energia.
- Realizar cálculos no BR-Plant pra mostrar para os gestores o consumo de energia consolidado.
- Entrevistas com a superintendência de infraestrutura da UFRN afim de identificar possíveis índices de qualidade para o problema de monitoramento de água e energia.

- Desenvolvimento do hardware de medição de água.

Referências

1 – ATWireless, acessado pela última vez em 28/07/2016. Disponível online em: http://www.atwireless.com.br/wireless

2 – IEEE, WIMAX, acessado pela última vez em 28/07/2016. Disponível online em: https://standards.ieee.org/about/get/802/802.16.html

3 – Revista Exame, Projeto de smart grid da AES Eletropaulo usará WiMAX, acessado pela última vez em 28/07/2016. Disponível online em: http://exame.abril.com.br/tecnologia/noticias/projeto-de-smart-grid-da-aes-eletropaulo-usara-wimax

4 - Lassen, T., Long-range RF communication: Why narrowband is the de facto standard, Texas Instruments White Paper, acessado pela última vez em 28/07/2016. Disponível online em: http://www.ti.com/lit/wp/swry006/swry006.pdf?DCMP=longrange&HQS=ep-wcs-lprf-longrange-contrib-whip-narrowband-wwe

5 - Simens, Wireless Mesh Solutions, Simens White Paper, acessado pela última vez em 28/07/2016. Disponível online em: http://w3.siemens.com/smartgrid/global/en/products-systems-solutions/smart-communication/distribution-communication-solutions/pages/wireless-mesh-solutions.aspx.

6 – NXP Co., Thread Networking Protocol, acessado pela última vez em 28/07/2016. Disponível online em: http://www.nxp.com/pages/thread-networking-protocol:THREAD-NETWORKING-PROTOCOL

7 – Bluetooth CO., Bluethoot Smart, acessado pela última vez em 28/07/2016. Disponível online em:https://developer.bluetooth.org/TechnologyOverview/Pages/BLE.aspx

8 – Adafruit, Bluetooth LE, acessado pela última vez em 28/07/2016. Disponível online em:https://www.adafruit.com/product/1697

9 – RFID CO., Modulação GFSK, acessado pela última vez em 28/07/2016. Disponível online em: https://sites.google.com/site/nearcommunications/modulacao-gfsk

10 – ESA Communication Department, Data transmission between European satellites using Laser Lights, acessado pela última vez em 28/07/2016. Disponível online em:http://www.esa.int/Our_Activities/Telecommunications_Integrated_Applications/A_world_first_Data_transmission_between_European_satellites_using_laser_light

11 - Acampora A., Bloom H., Krishnamurthy S., UniNet: A Hybrid Approach for Universal Broadband Access Using Small Radio Cells Interconnected by Free-Space Optical Links, IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, VOL. 16, NO. 6, AUGUST 1998.

12 – Laser Instruments, ATMOSPHERIC LASER COMMUNICATION, acessado pela última vez em 28/07/2016. Disponível online em:http://laseritc.com/?id=124

13 - Raspberry Pi 0, acessado pela última vez em 28/07/2016. Disponível online em: https://www.adafruit.com/

14 - SimCom, SIM800L, acessado pela última vez em 28/07/2016; disponível online em http://simcomm2m.com/En/.
15 – Arduino Co., Arduino Uno, acessado pela última vez em 28/07/2016; disponível online em: https://www.arduino.cc/
16 – ST Electronics, LM317 Datasheet, acessado pela última vez em 28/07/2016; disponível online em: http://www.st.com/
17 – ITU, Command line general format, ITU-T/Telecommunication Standardization Bureau 1999.
18 - SimCom, SIM800 Series_AT Command Manual_V1.09, acessado pela última vez em 28/07/2016; disponível online em: http://simcom.ee/documents/SIM800x/SIM800%20Series_GSM%20Location_Application%20Note_V1.01.pdf
19 – PySerial, Python, acessado pela última vez em 28/07/2016; disponível online em: https://pythonhosted.org/pyserial/