UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
ENG. DE CONTROLE E AUTOMAÇÃO

IGOR FANTINEL DIESEL

ANÁLISE DO CONSUMO ENERGÉTICO DE UM SISTEMA DE TRANSMISSÃO DE DADOS POR REDES MÓVEIS

Porto Alegre
2019

IGOR FANTINEL DIESEL

ANÁLISE DO CONSUMO ENERGÉTICO DE UM SISTEMA DE TRANSMISSÃO DE DADOS POR REDES MÓVEIS

Trabalho de Conclusão de Curso (TCC-CCA) apresentado à COMGRAD-CCA da Universidade Federal do Rio Grande do Sul como parte dos requisitos para a obtenção do título de *Bacharel em Eng. de Controle e Automação* .

ORIENTADOR: Prof. Dr. Marcelo Götz

Porto Alegre
2019

IGOR FANTINEL DIESEL

ANÁLISE DO CONSUMO ENERGÉTICO DE UM SISTEMA DE TRANSMISSÃO DE DADOS POR REDES MÓVEIS

Este Trabalho de Conclusão de Curso foi julgado adequado para a obtenção dos créditos da Disciplina de TCC do curso *Eng. de Controle e Automação* e aprovado em sua forma final pelo Orientador e pela Banca Examinadora.

Orientador: _____
Prof. Dr. Marcelo Götz, UFRGS
Doutor pela Universität Paderborn - Paderborn, Alemanha

Banca Examinadora:

Prof. Dr. Ivan Müller, UFRGS
Doutor pela Universidade Federal do Rio Grande do Sul – Porto Alegre, Brasil

Prof. Dr. Marcelo Götz, UFRGS
Doutor pela Universität Paderborn - Paderborn, Alemanha

Prof. Dr. Valner Brusamarello, UFRGS
Doutor pela Universidade Federal de Santa Catarina – Florianópolis, Brasil

_____
Prof. Dr. Marcelo Götz
Coordenador de curso
Eng. de Controle e Automação

Porto Alegre, dezembro de 2019.

AGRADECIMENTOS

A minha família, em especial a meus pais, que sempre me apoiaram desde as primeiras decisões em me dedicar aos estudos, até nos momentos de ausência causados pelos mesmos, acompanhando minha trajetória.

Aos amigos e colegas do curso pelos anos de parceria. Em especial a Alex Treviso, que provavelmente revisou este trabalho mais vezes do que eu. Também a Davi Bobsin, pelos auxílios com python e todos os trabalhos em conjunto na faculdade. Sou grato por dividir todas estas etapas com vocês.

Aos professores Mário Sobczyk, Fabiano Wildner e Pedro Bolognese pela flexibilidade mostrada perante a oportunidades que surgiram durante este ano.

Ao meu orientador, Marcelo Götz, por ter aceitado este trabalho mesmo com uma parte tendo sido realizada a distância

Às amizades de longa data, que mesmo que a vida leve a uma diminuição do contato, tenho certeza que sempre estarão lá para apoiar.

Também às novas amizades formadas durante a faculdade, fortes e marcantes, por proporcionarem, sem dúvidas, a base necessária para vencer este ano cheio de diferentes objetivos, etapa marcante em minha vida.

"O maior inimigo do que eu mais quero, é o que eu quero agora."
Autor Desconhecido

RESUMO

Este trabalho trata da análise do consumo energético de um sistema de transmissão de dados voltado a aplicação em monitoramento remoto, onde as mensagens são enviadas pela rede GPRS. Para atingir este objetivo, desenvolveu-se um protótipo com a capacidade de enviar um dado e identificar o inicio e fim das etapas relacionadas à conexão com a rede móvel (GSM), rede de dados (GPRS) e servidor MQTT, para o qual os dados são enviados. Esta identificação ocorre por meio de um *firmware* implementado em um microcontrolador, que comanda o funcionamento de um módulo modelo SIM800L, do fabricante *SimCom*, dedicado a conexão e envio de dados pela rede GPRS. A aquisição dos dados de consumo do módulo em questão é dada pela adição de um resistor *shunt* em série com a alimentação, onde a medida da queda de tensão sobre este resistor, de valor constante, permite conhecer a corrente consumida pelo módulo. A medida desta tensão é realizada por um circuito integrado INA219, de forma intermitente, salvando os dados em registradores, cujo acesso pode ser realizado por comunicação I2C, que por sua vez é tratada por um segundo microcontrolador. Este microcontrolador adicional ainda assegura uma taxa de amostragem de 10kHz (amostras a cada 100μ segundos), enviando os dados adquiridos a um computador, onde posteriormente os dados são tratados, sendo possível isolar as etapas previstas de conexão. A partir dos dados de consumo adquiridos, um modelo matemático é proposto, a fim de estimar o consumo energético do módulo com diferentes taxas de envio de mensagem, estimando, também, a duração de baterias como alimentação do módulo.

Palavras-chave: Consumo Energético, Monitoramento Remoto, Redes Móveis, Internet das Coisas, GPRS.

ABSTRACT

This study focuses in the analysis of the energy consumption of a data transmission system aimed at remote monitoring applications, where messages are sent by the GPRS (2G) network. To achieve this objective, a prototype was developed with the ability to send data and identify the beginning and end of the steps related to connection to the mobile network (GSM), data network (GPRS) and MQTT server, to which the data is sent. This identification occurs through a *firmware* implemented in a microcontroller, which controls the operation of a module model SIM800L, from the manufacturer *SimCom*, dedicated to the connection and sending of data by the GPRS network. The acquisition of the consumption data of the module in question is given by the addition of a *shunt* resistor in series with the power supply, where the measurement of the voltage drop on this resistor, of constant value, allows the estimation of the current consumed by the module. The measurement of this voltage is carried out by an INA219 integrated circuit, that saves the data in registers, which can be accessed by I2C communication, handled by a second microcontroller. This additional microcontroller also ensures a sampling rate of 10kHz, sending the acquired data to a computer, where the data is later processed, with the possibility to isolate the foreseen connection steps. From the acquired consumption data, a mathematical model is proposed in order to estimate the energy consumption of the module with different message sending rates, also estimating the battery life as power supply of the module.

Keywords: Energy Consumption, Remote Monitoring, Mobile Networks, Internet of Things, GPRS.

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS

CCA	Curso de Eng. em Controle e Automação
IoT	*Internet of Things*
GSM	*Global System for Mobile Communications*
GPRS	*General Packet Radio Service*
TCP	*Transmission Control Protocol*
MQTT	*Message Queuing Telemetry Transport*
IDE	*Integrated Development Environment*
I/O	*Input/Output*
I2C	*Inter-Integrated Circuit*
RFID	*Radio-Frequency Identification*
UMTS	*Universal Mobile Telecommunication System*
HSPA+	*High-Speed Downlink Packet Access*
LTE	*Long Term Evolution*
SMS	*Short Message Service*
M2M	*Machine to Machine*
HTTP	*Hypertext Transfer Protocol*
FTP	*File Transfer Protocol*
SFTP	*SSH File Transfer Protocol*
SSH	*Secure Shell*
QoS	*Quality of Service*
TTL	*Transistor-transistor logic*
ADC	*Analogic-Digital converter*
LSB	*Least significant bit*
I2C	*Inter-Integrated Circuit*
SDA	*Serial Data*
SCL	*Serial Clock*

RAM	*Random Access Memory*
CPU	*Central Processing Unit*
USB	*Universal Serial Bus*
FIFO	*First in First out*

LISTA DE SÍMBOLOS

σ	Desvio padrão
ε	Erro amostral
z	Escore z
n	Número de amostras
μ	Micro

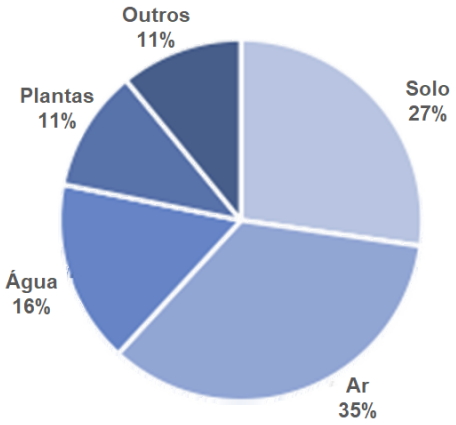
1 INTRODUÇÃO

Com a disseminação do IoT *(Internet of Things)* e o surgimento progressivo de novas tecnologias, o monitoramento remoto vem se mostrando como um instrumento eficiente para aquisição e monitoramento de dados não críticos e com baixas taxas de amostragem, garantindo uma continuidade na aferição dos mesmos, sem a necessidade de intervenção humana.

Segundo Talavera (2017), 62% dos artigos em inglês, relacionados a IoT e publicados na forma de documentos acadêmicos entre os anos de 2006 e 2016, exploram a sua utilização em monitoramento remoto na área agroindustrial. Entre outras aplicações, 25% dos documentos citam a sua utilização em controle, 6% em predição e 7% em logística. Considerando a área de monitoramento remoto, as aplicações mais comumente encontradas dizem respeito ao monitoramento do ar, do solo, de água, de animais e plantas. Dentre os objetivos das aplicações, encontram-se a determinação e monitoramento de parâmetros relacionados ao microclima, como temperatura e umidade.

Dentro do domínio do monitoramento remoto, a Figura 1 mostra a distribuição de artigos entre as aplicações citadas anteriormente.

Figura 1: Aplicações de IoT na área de monitoramento remoto.



Fonte: (TALAVERA et al., 2017)

Uma característica comum entre as aplicações citadas é a baixa taxa de amostragem dos dados. Como, geralmente, grandezas relacionadas ao microclima (temperatura e umidade) possuem uma alta inércia, taxas de amostragem na ordem de minutos ou até horas são utilizados. Entre o envio de um dado e outro, os dispositivos tendem a entrar no modo de economia de energia, estendendo o tempo de duração das baterias, que são necessárias já que em áreas remotas não há acesso a rede elétrica.

Segundo Borgia (2014), a eficiência energética é uma das principais características que deve ser levada em conta nos sistemas de IoT. Com uma grande possibilidade de otimização, é um ramo onde há um grande esforço de pesquisa. Esta importância é clara levando-se em consideração as aplicações de monitoramento citadas anteriormente, onde os dispositivos utilizados em áreas remotas são alimentados por baterias.

É comum encontrar na literatura propostas de melhoria da eficiência energética de redes de aquisição de dados baseadas em IoT, como a apresentada em (WANG; WANG et al., 2016). A proposta é baseada em dispositivos alimentados por bateria e propõe redes inteligentes, onde os intervalos e rotinas de entrada e saída do modo economia de energia foram otimizados para reduzir o consumo individual de cada dispositivo. Porém, não apresenta uma análise da ordem de grandeza do consumo de cada objeto, o que é proposto no presente trabalho.

Mesmo com diversas propostas de redes de comunicação sem fio criadas para atender as demandas trazidas pelo crescimento da IoT, as redes móveis de celulares ainda são comumente utilizadas, por disponibilizarem coberturas já consolidadas em diversas áreas e pelo baixo custo de módulos dedicados.

A motivação do desenvolvimento deste trabalho busca prover dados técnicos acompanhando o crescimento das tecnologias citadas, ao analisar o consumo energético de uma aplicação de monitoramento de dados, com o envio das mensagens realizado pela rede GPRS, possibilitando a estimação do tempo de duração de baterias como fonte de alimentação do sistema.

1.1 Objetivos

Para caracterizar o consumo energético de um ciclo completo de envio de dados, propõe-se o desenvolvimento de um sistema que permita adquirir dados relacionados a cada etapa de conexão com a rede, produzindo amostras suficientes para a apresentação de um modelo matemático. Entre as etapas, busca-se caracterizar a conexão com a rede GSM, conexão com a rede GPRS, abertura de um *socket* TCP, conexão com um servidor MQTT e envio dos dados. Além disto, pretende-se realizar a estimação do tempo de duração de baterias como alimentação do sistema, a partir da aplicação do modelo do consumo a diferentes taxas de envio de dados.

2 REVISÃO BIBLIOGRÁFICA

Nesta seção, serão apresentados os principais conceitos relacionados ao desenvolvimento deste trabalho.

2.1 Plataforma Arduino

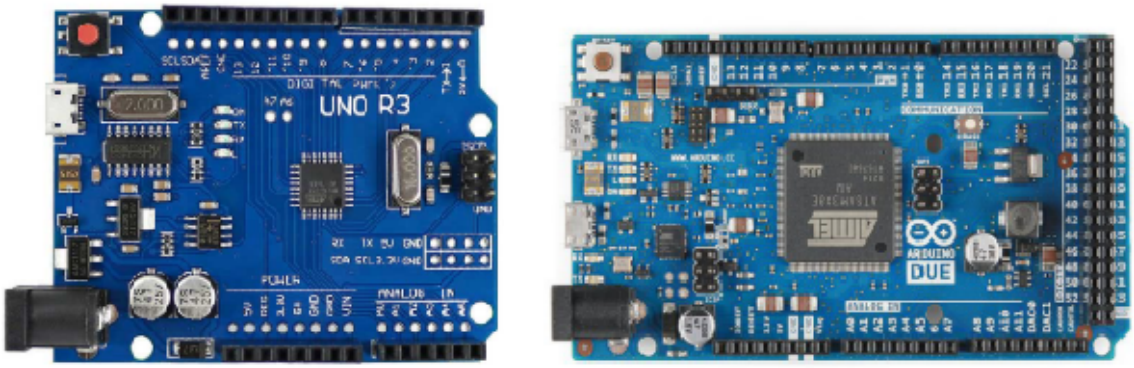
Arduino é, acima de tudo, uma plataforma *open-source* para criação de projetos eletrô - nicos. Isso significa que todos os arquivos de projeto da placa de circuito impresso e os diagramas esquemáticos, necessários para sua produção, são disponibilizados abertamente. O Arduino, muitas vezes considerado um microcontrolador, é, na verdade, uma placa de desenvolvimento completa baseada em um microcontrolador, contendo também todos os componentes necessários para o fornecimento de energia e para comunicação com o computador. Além da placa de circuito física programável, o Arduino consiste também em um software, ou IDE (*Integrated Development Environment*), executado em um computador com o objetivo de criar ou carregar o código desenvolvido para a placa física (MCROBERTS, 2018).

Segundo Monk (2017), o sucesso do Arduino pode ser atribuído, entre outras razões, ao fato de que o mesmo não se limita a placas com microcontrolador. Há, ainda, um grande número de componentes acessórios (chamados *shields*) compatíveis com o padrão Arduino. Essas placas são encaixadas diretamente nos pinos padrões do Arduino e, usualmente, contam com exemplos de códigos criados na IDE padrão. É importante também ressaltar que há diversos modelos de placas Arduino, como os modelos Uno, Due, Diecimila, Mega, Nano, citados por Monk (2017), que contam com diferentes modelos de microcontroladores mas sempre respeitam o padrão Arduino, sendo compatíveis com a IDE.

Dois modelos, citados por Monk (2017), são utilizados neste trabalho, sendo eles o Arduino UNO(R) e o Arduino DUE(R). O modelo UNO(R) apresenta em sua construção o microcontrolador ATMega328P, que tem como características técnicas: tensão de operação de 5V, 14 pinos digitais de I/O, 6 entradas analógicas, 32Kb de memoria *flash*, 2Kb de memoria RAM e um clock de 16MHz.

Já o Arduino DUE é baseado no microcontrolador SAM3X8E com um CPU ARM Cortex-M3. Este componente possui especificações mais robustas, como 52 pinos de I/O, 12 entradas analógicas, 512Kb de memoria *flash*, 96Kb de memória RAM e um *clock* de 84MHz. Além disso, conta com uma porta USB nativa, que atinge velocidades de transmissão de dados maiores que a presente no UNO e, por ter uma velocidade de *clock* maior, suporta uma velocidade de até 4MHz em sua interface I2C. Essa características de maiores velocidades nas interfaces de comunicação fazem do DUE uma melhor opção para aquisição de dados. A Figura 2 apresenta o Arduino UNO e o Arduino DUE.

Figura 2: Arduino UNO e Arduino DUE.



Fonte: O autor.

2.2 *IoT - Internet of Things*

O termo IoT *(Internet of Things)* é utilizado para representar uma infraestrutura global, dinâmica e com capacidades de se auto-configurar, baseada em protocolos padrões onde dispositivos virtuais e físicos (chamados *things*) têm identidades individuais, utilizando interfaces inteligentes e completamente integrados a rede global (VERMESAN, 2009).

Do inicio da IoT até os dias de hoje, pode-se considerar que este conceito foi marcado por três gerações. A primeira de dispositivos identificados com *tags* RFID (ATZORI; IERA; MORABITO, 2010), em que o objetivo era a identificação única de objetos e sua conexão. Com o crescimento do conceito de IoT, chega-se a segunda geração, marcada pela conexão de objetos pelo protocolo *Internet Protocol* (IP), que permitiu a a conexão de mais dispositivos de forma nativa (MAINETTI; PATRONO; VILEI, 2011). Atualmente, a terceira geração está marcada pela utilização de computação em nuvem e popularização do conceito, sendo centralizada em pessoas, conteúdo e serviços.

Para implementar estes conceitos, é necessária a criação de um padrão de comunicações para transmissão de informações entre dispositivos, sistemas e pessoas sem que haja ambiguidade nas etapas. Apesar das diferentes propostas encontradas na literatura, muitos dos conceitos são semelhantes e complementares, sendo o trabalho de (TRAPPEY et al., 2017) uma revisão dos padrões relacionados, onde o autor conclui que os elementos IoT podem ser divididos em quatro camadas: percepção, transmissão, computação e aplicação.

A camada de percepção tem por objetivo reconhecer e perceber informações geralmente composta por sensores e dispositivos como RFID. A camada de transmissão é responsável por enviar informações coletadas através da Internet. A camada de computação recebe, armazena e disponibiliza os dados para as aplicações clientes. A camada de aplicação é responsável por compartilhar as informações entre usuários e manter as informações seguras. Aprofundando-se na camada de transmissão, diversas tecnologias podem ser utilizadas para este fim, sendo algumas apresentadas na Tabela 1, que apresenta as tecnologias e suas vantagens.

Tabela 1: Características dos principais meios de transmissão de dados em IoT.

Tecnologia	Norma	Meio de transmissão	Bandas de frequência	Velocidade de transmissão	Distância máxima	Limitações
Ethernet	IEE 802.3 u/z	Cabo de cobre trançado, fibra otica	-	10 Mbps, até 100 Gbps	100m, até 50-70km	Conexão física entre os dispositivos
WiFi	I33 802.11 a/b/g/n	Sem fio	2,5 GHz, 5GHz	1-600 Mbps	até 100m	Interferência entre comunicações WiFi
WiMAX	I333 802.16 a/d/e/m	Sem fio	2-66 GHz	até 70 Mbps	Até 50km	Sensível a condições meteorológicas, alto custo
xDLS	ADSL, ADSL2+	Cabo de cobre trançado, cabo coaxial	até 2.2 MHz	12-55 Mbps	até 1.3 km	Comunicação não simétrica
Celular	GSM, GPRS, UMTS, HSPA+, LTE	Sem fio	900-1800 MHz 2100-1900 MHz 800-2600 MHz	9.6 kbps, 56-114 kbps, 56 Mbps (d) / 22 Mbps (u), 300 Mbps (d) / 75 Mbps (u)	10m até 100km	Spectro limitado
Satellite	BSM, DVB-S	Sem fio	4-8 GHz (C), 10-18 GHz (Ku), 18-31 GHz (Ka)	16 kbps até 155Mbps	GEO: 35 km MEO: 500m-15 km LEO: 200m-3km	Atraso de propagação de 280ms, alto custo inicial

Fonte: (BORGIA, 2014)

Segundo (BORGIA, 2014) entre as tecnologias com cabo a referência é a *Ethernet* (IEEE 802.3), que suporta transmissões entre 10Mbps até 100Gbps utilizando pares de fio trançado, cabo coaxial ou fibra ótica. Sua principal vantagem é ser mais confiável e robusta já que é menos suscetível a erros e interferência. No entanto, a necessidade de uma conexão física gera um custo alto de conexão e alto trabalho no caso de mudanças na rede. Por isso, uma forma comum de acessar a rede é por meio de tecnologias *Wireless*. Entre as diferentes tecnologias existentes, pode-se citar a família WiFi (IEEE 802.11a/b/g/n) e diversos protocolos criados especificamente para aplicações IoT, como ZigBee, LoRa(R) e SigFox.

Porém, as redes móveis de celular (GSM, GPRS, UMTS, HSPA+ e LTE) ainda apresentam um papel central na forma como os dados são transmitidos. Isso ocorre por oferecerem a possibilidade de conexão em ambientes espaçosos e remotos, permitindo não só a troca de dados de voz como também serviços de alto valor agregado (como SMS) (BORGIA, 2014).

2.3 *Machine-to-machine (M2M)*

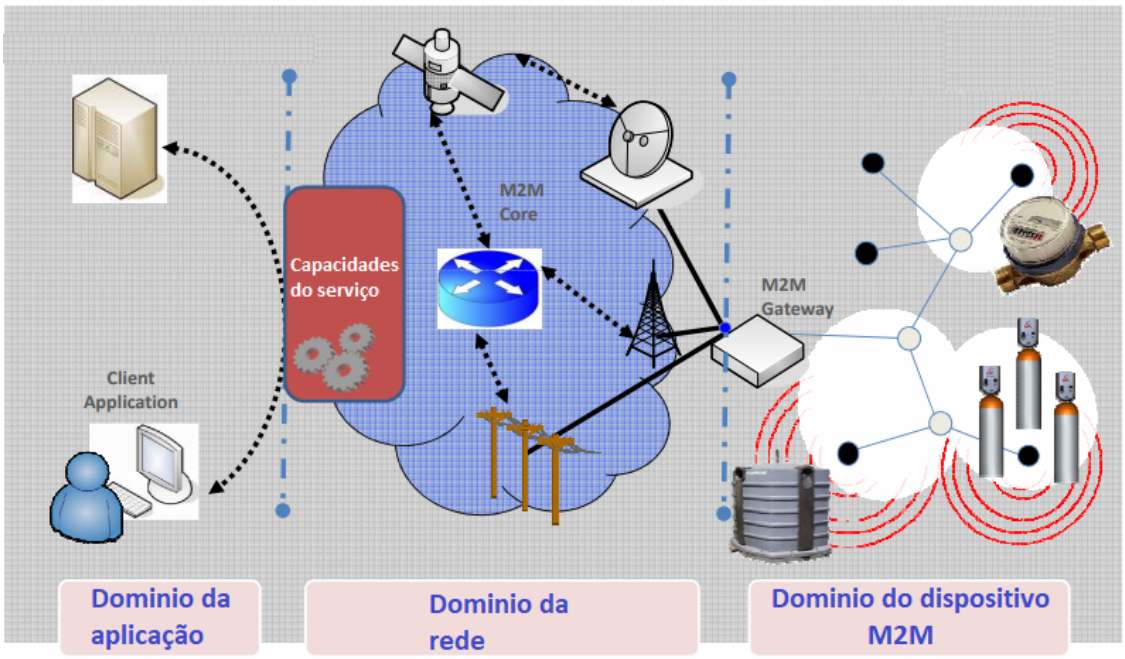
A popularização do IoT permite uma maior interação entre pessoas, de pessoas e dispositivos e entre dispositivos. Este último tipo de comunicação é chamado de M2M (*Machine-to-Machine*) (HOLLER et al., 2014). Segundo Holler (2014), M2M diz respeito a aplicações que contam unicamente com a comunicação entre dispositivos, via conexão cabeada ou *Wireless*. Esta comunicação ocorre sem a intervenção humana, automatizando processos de transferência de dados.

Segundo BORSWARTHICK et al. (2012), M2M possui um escopo flexível, onde seu papel é fornecer as condições para que um dispositivo troque informações bidirecionalmente com uma aplicação através de uma rede de comunicação, permitindo que o dispositivo, ou a aplicação, tomem decisões agindo de acordo com esta troca de dados.

Os dispositivos considerados M2M devem ter a capacidade de coletar dados em determinados ambientes, transmitindo os mesmos a uma aplicação por meio de uma conexão. Em muitos casos, os dispositivos coletores (geralmente sensores) não tem capacidade nativa de enviar os dados diretamente para a aplicação, sendo necessária a utilização de outro dispositivo agindo como intermediário na comunicação, repassando as informações (BOSWARTHICK; ELLOUMI; HERSENT, 2012).

Em 2009, o ETSI (*European Telecommunications Standard Institute*) formou um comitê técnico com o objetivo de criar um padrão para uma arquitetura de comunicação M2M, não totalmente desenvolvido pelo comitê mas também por outras organizações (BOSWARTHICK; ELLOUMI; HERSENT, 2012). Na especificação técnica que trata esse assunto, a ETSI (2013) apresenta uma arquiteturas de alto nível para a comunicação M2M. Em sua forma completa ela é dividida em dois domínios: Dispositivos & *Gateway* e Rede. Apresenta também uma arquitetura mais simplificada, que envolve os elementos mais importantes e define um novo domínio, chamado domínio de Aplicação. A versão simplificada é apresentada na Figura 3.

Figura 3: Arquitetura M2M Simplificada



Fonte: (ETSI, 2013)

Entre os elementos necessários para a arquitetura citada, a ETSI (2013) define os seguintes:

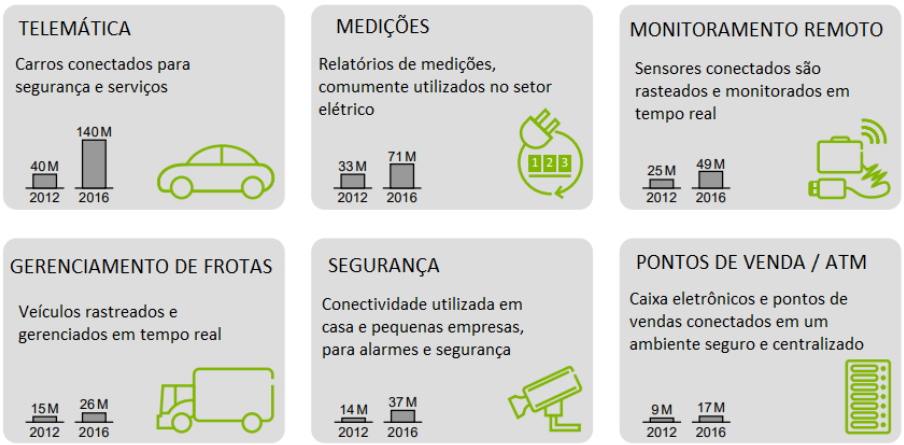
- **Dispositivo M2M**: Dispositivo capaz de responder a demandas de dados ou capaz de transmitir dados de forma autônoma;
- **Rede de área M2M** (Domínio do Dispositivo): Fornece a conectividade entre os dispositivos M2M e os Gateways;
- **Gateway M2M**: Usa as capacidades M2M para garantir o trabalho entre os dispositivos M2M e a interconexão com a rede de comunicação;
- **Rede de comunicação** (Domínio de rede): Comunicação entre o Gateway e a Aplicação;

- **Aplicação M2M**: Contém a camada onde os dados passam por varias aplicações e são processados para usos específicos.

Ainda, segundo ETSI, não há a necessidade de criação de novas tecnologias para a implementação da arquitetura básica de M2M, pois todos os domínios podem se basear em tecnologias já existentes e, em muitos casos, já difundidas pelo paradigma da IoT. Os protocolos Zigbee e Wifi como soluções para as redes de área, no domínio do dispositivo, são exemplos disso. Ainda, a utilização de redes móveis e comunicações por satélite no domínio de rede.

Considerando o ramo de M2M, a Figura 4 apresenta suas principais aplicações registradas entre os anos de 2012 e 2016. Em primeiro lugar, podemos notar a aplicação em telemática em veículos, como nos sistemas de auto-navegação, diagnóstico, sistemas anti-furto e, até, cobrança automática de pedágios.

Figura 4: Aplicações de M2M entre 2012 e 2016



Fonte: (HOLLER et al., 2014)

Com relação a aplicação em monitoramento remoto, Holler (2014) cita aplicações no setor de energia, bem como em setores de água e gás. Nestes casos, a eficiência energética dos dispositivos é de grande importância, já que as medições tendem a ocorrer em áreas remotas onde não há acesso a rede elétrica e a alimentação dos dispositivos ocorre por baterias. Esta aplicação é o objeto de estudo deste trabalho, onde deseja-se estimar o consumo de um dispositivo em aplicações similares as citadas.

Para encontrar uma ordem de grandeza do consumo, podemos citar o trabalho de (WANG; MANNER, 2010), onde uma comparação entre o consumo energético entre o envio de dados pelas redes 2G, 3G e WiFi é realizado. Nesse estudo, o consumo foi medido pela alimentação de um celular HTC ONE com variação no tamanho dos pacotes de dados enviados. Os pacotes variaram entre 40kbytes e 1000kbytes e a conclusão foi de que o protocolo 2G é o que mais consome com o aumento do tamanho dos pacotes de dados enviados, com consumo na ordem de 1W. Contudo, esses pacotes não condizem com as aplicações de monitoramento remoto, que tem por característica envio de pacotes menores (apenas alguns bytes) e com taxa de amostragem bastante baixa, possibilitando ao sistema entrar em modo de economia de energia entre os envios.

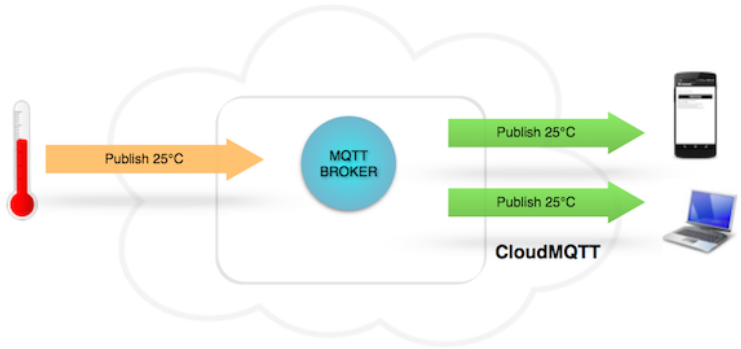
2.4 Servidor MQTT

Em redes de computadores, existem diversos protocolos de comunicação que gerenciam a transferência de dados e conexão entre dois ou mais computadores, como HTTP, FTP e SFTP. Quando tratamos de dispositivos, também existe a necessidade de um protocolo que gerencie a troca de informações, isto é, a troca de mensagens e dados entre os dispositivos que compõem a rede, de forma eficiente e que considere as características e limitações impostas pela aplicação.

Seguindo o paradigma de IoT com 4 camadas, os protocolos de comunicação fazem parte da camada de Transmissão (TRAPPEY et al., 2017). Um desses protocolos é o MQTT *(Messaging Queue Telemetry Transport)*, que foi criado no final dos anos 90 por Andy Stanford-Clark (IBM) e Arlen Nipper (Eurotech). Este protocolo adota a arquitetura de *publish/subscribe* voltado para dispositivos com restrições como redes inseguras, baixa largura de banda e alta latência (EUROTECH, 1999). A atual versão da norma que define o protocolo MQTT apresenta diversas características obrigatórias, como o uso de TCP/IP para conectividade.

Neste protocolo, os sensores e dispositivos de aquisição de dados são chamados clientes e se conectam a um servidor, chamado *broker*. O processo de envio de um dado para este servidor é chamado de *publish* e os dados devem ser endereçados a uma chave, chamada de tópico. Após o envio dos dados, outros clientes recebem a informação que é repassada pelo *broker*, o processo de recebimento de um dado publicado é chamado de *subscribe* (EUROTECH, 1999). A Figura 5 exibe um exemplo desse comportamento, com um cliente publicando um dado a um tópico e dois outros clientes inscritos no mesmo.

Figura 5: Exemplo de aplicação MQTT com clientes e *subscribers*



Fonte: (CLOUDMQTT, 2019)

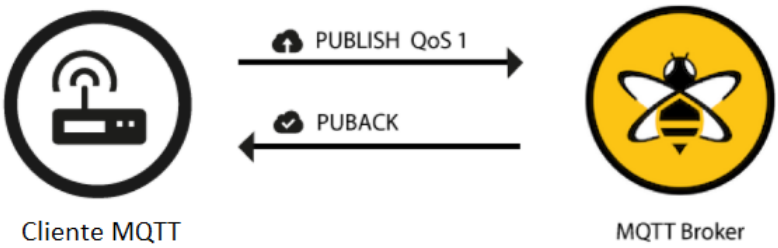
Uma das vantagens do protocolo MQTT é a facilidade de encontrar servidores mantidos na nuvem por empresas que oferecem serviços de hospedagem e, muitas vezes, disponibilizam planos básicos gratuitos. Um exemplo destes servidores é o da empresa *cloudMQTT* (CLOUDMQTT, 2019).

Outra característica desejável do protocolo MQTT é o chamado QoS *(Quality of Service)*, que deve ser especificado na abertura da conexão com o servidor. O nível de QoS é um acordo entre o servidor e o dispositivo que envia uma mensagem, garantindo o recebimento da mesma (EUROTECH, 1999). Basicamente, dependendo do nível de QoS especificado, o servidor responde ao dispositivo com um pacote de *acknowledgment*, indicando se a mensagem foi enviada com sucesso, característica desejável na aplicação estudada já que permite reconhecer as etapas de envio de um dado para o servidor MQTT. Os níveis de QoS suportados pelo protocolo MQTT são os seguintes:

- QoS 0: No máximo uma vez
- QoS 1: No mínimo uma vez
- QoS 2: Exatamente uma vez

Após o envio de uma mensagem, o servidor responde com um pacote chamado de PUBACK, que pode ser reconhecido pelo cliente de forma a garantir que a mensagem foi enviada com sucesso. Este pacote é utilizado neste trabalho para marcar o fim da etapa de envio de dados para o servidor MQTT. A Figura 6 ilustra um envio com nivel de QoS igual a 1.

Figura 6: Exemplo de *publish* de uma mensagem com QoS = 1



Fonte: (HIVEMQ, 2019)

2.5 Evolução das Redes Móveis

Após a invenção do telefone por Alexander Graham Bell em 1876, a troca de sinais era realizada por um cabeamento elétrico que interligava dois usuários nas extremidades opostas. E para cada par de usuários que desejasse trocar informações, era necessário cabeamento direto entre os dois aparelhos. Logo esse sistema se tornou caótico e surgiu então a primeira estação de comutação (TANENBAUM, 1997).

Assim, pela demanda surgiu a necessidade de criação de uma rede para atender a mobilidade do usuário, por meio de redes sem fio. As comunicações baseadas em redes sem fio são baseadas no princípio de que o movimento dos elétrons cria ondas eletromagnéticas que podem se propagar através do espaço livre. A quantidade de oscilações por segundo dessa onda é chamada de frequência e medida em Hz (TANENBAUM, 1997).

Nos anos 50, a empresa Bell Labs propôs o conceito de um sistema celular, permitindo a reutilização do espectro de frequências com aplicação em um sistema de comunicações móveis. No entanto, a necessidade de ligar os usuários do sistema fixo, com usuários do sistema móvel por meio de diferentes transceptores (estações rádio base), trazia a necessidade de uma grande capacidade de computação. Assim, o sistema não era viável com a tecnologia disponível na época. A sua execução acabou sendo viabilizada pela utilização de microprocessadores nos terminais, e em outubro de 1983, o primeiro sistema de celular foi posto em operação em Chicago, pela AT&T (BERNAL, 2002).

Segundo Gruber (2007), um sistema celular básico pode ser composto de:

- **Central de Comutação e Controle (CCC):** responsável pela validação dos assinantes, processamento de chamadas, interface com a rede de telefonia fixa e interface com outras CCCs, sendo elas da mesma operadora ou não. Faz parte da CCC também a BSC (Base Station Controller), responsável pelo gerenciamento das ERBs.

- **Estação Rádio Base (ERB):** é encarregada da comunicação com as estações móveis de determinada área. É a ERB que comunica-se com o assinante através da interface definida (CDMA, TDMA, GSM, etc.) e com a CCC.
- **Rede de telefonia pública comutada (RPTC):** insere o suporte a telefones fixos e interligam as CCCs de diferentes prestadores de serviços.
- **Estação Móvel (EM):** é o dispositivo portátil desenvolvido para comunicar-se com as ERBs em qualquer dos canais alocados.

De forma geral, podemos dizer que os telefones móveis tiveram três gerações distintas, de diferentes tecnologias: voz analógica, voz digital, voz digital e dados. Estas gerações são abordadas nas subseções posteriores.

2.5.1 Primeira Geração

Comumente conhecida como 1G, a primeira geração de sistemas celulares trouxe a tecnologia sem fio aos usuários. Os primeiros dispositivos celulares faziam uso do sistema analógico de telefonia móvel - AMPS, com múltiplo acesso por divisão de frequência, mas logo a popularização de sistemas celulares levou a necessidade de aumento da capacidade, motivando o desenvolvimento de sistemas de segunda geração (GRUBER, 2007).

2.5.2 Segunda Geração

A segunda geração da tecnologia móvel é marcada pelo início da utilização de tecnologias digitais na troca de informações. O mesmo canal de frequência agora é partilhado por diferentes utilizadores, podendo eles serem TDMA, CDMA e GSM (WALDMAN; YACOUB, 2000).

- **TDMA**: o acesso de terminais a uma mesma ERB é gerido alocando-se uma frequência para cada terminal, ou compartilhando da mesma faixa de frequência mas transmitindo em tempos diferentes (WALDMAN; YACOUB, 2000).
- **CDMA**: o acesso de canais que compartilham uma mesma banda de frequências é realizando pela geração de códigos diferentes pelos vários terminais (GARG; SMOLIK; WILKES, 1997).
- **GSM**: a estação móvel é carregada com um cartão inteligente, chamado cartão SIM, que pelo contrato de um serviço junto a uma operadora, passa a ter acesso a rede. Cada SIM possui um número único de 15 dígitos identificando o a estação móvel. Já o terminal é caracterizado por um número também com 15 dígitos, atribuído pelo fabricante, denominado IMEI (GRUBER, 2007).

A estrutura dos canais físicos do GSM possibilitou a introdução de serviços como SMS, Fax e transporte de dados com taxas de 2,4 a 9,6 Kbps. O crescimento das aplicações de dados como acesso à Internet, levou a necessidade de desenvolver soluções que permitissem o transporte de dados a taxas maiores (GRUBER, 2007). Surge assim, então, a segunda e meia geração.

2.5.3 Segunda e meia geração

Também conhecida como 2,5G, a segunda e meia geração implementa serviços de dados por pacotes, sem que haja a necessidade de uma conexão permanente. Com taxas

de até 144 kbps, é um passo intermediário na evolução para a 3G. Em um sistema GSM, são chamadas de tecnologias 2,5G os sistemas GPRS e EDGE (GRUBER, 2007).

A rede GPRS fornece a estação móvel conexão à Internet sem a necessidade de estabelecer uma chamada telefônica. Com taxas que, teoricamente, podem chegar a 171,2 kbps, é até três vezes mais rápida que as alcançadas previamente nas telecomunicações fixas. Segundo Gruber (2007), pode-de definir os seguintes passos para a conexão de um terminal a rede GPRS:

- Ao ser energizado, o terminal GPRS é reconhecido pela rede como ocorre com um terminal GSM para voz;
- Uma conexão GPRS é estabelecida ao adquirir-se um endereço de IP. Este endereço é normalmente dinâmico e é fornecido pela operadora móvel;
- O terminal fica então pronto para enviar e receber pacotes, onde pode assumir etados de *idle* (ocioso), *ready* (pronto) onde pode enviar e receber pacotes instantaneamente ou *stand-by*.

Já o sistema EDGE está relacionado ao aumento da capacidade de transmissão da interface, adicionando novas características na rede GSM mas mantendo compatibilidade com os telefones celulares GSM/GPRS e com os equipamentos da rede (GRUBER, 2007).

2.5.4 Terceira Geração

Conhecida como 3G, esta geração define serviços móveis que proporcionam melhorias na qualidade de voz, internet de alta velocidade e serviços multimídia. Um novo padrão de interface entre a estação móvel e a ERB, com canais de radio frequência de 5 MHz, foi necessário para seu desenvolvimento. Os principais sistemas são o WCDMA e o CDMA 1xEVDO, que são evoluções dos sistemas GSM e CDMA (GRUBER, 2007).

2.5.5 Quarta Geração

Com a crescente demanda por serviços de internet, onde serviços que a poucos anos eram inacessíveis hoje são considerados essenciais, a necessidade por uma maior largura de banda cresce a cada dia. Dentre os sistemas que utilizam a tecnologia 4G, pode-se citar WiMAX e o LTE. Ambos atendem os requisitos da telefonia da 4ª geração, propostos pelo ITU, que são alta qualidade dos serviços móveis e taxas de dados acima de 100 Mbps, atendendo cada vez mais as demandas trazidas pelos usuários (FERRAZ; GARCIA; NUNES, 2012).

2.6 Comandos Hayes/Comandos AT

O conjunto de comandos Hayes, hoje mais popularizado como Comandos AT (Atenção), é uma linguagem específica originalmente desenvolvida por Dennis Hayes para a empresa Hayes Smartmodem em 1981. O conjunto de comandos consiste de uma série de textos curtos que podem ser combinados para formar comandos e executam operações específicas da telecomunicação, como discagem, atendimento de ligação ou alguma troca de parâmetros de conexão. A grande maioria dos modens de internet discada (*dial-up*) utilizam o conjunto de comandos AT com diferentes variações. Hoje em dia, com a necessidade de controlar funcionalidades adicionais da telecomunicação, uma grande quantidade de novos padrões foram criados por diferentes fabricantes, mas todos continuam a

compartilhar a estrutura e sintaxe básicas dos comandos (ZHICONG; DELIN; SHUNXIANG, 2008).

É comum encontrar no mercado módulos de telecomunicação compatíveis com a plataforma Arduino que suportam esse conjunto de comandos, permitindo, além de realizar chamadas ou envio de mensagens SMS, a conexão com a internet por redes móveis. Um exemplo desses módulos é o SIM800L, da empresa *SIMCOM*, que é abordado com mais detalhes nas subseções posteriores. Uma lista dos comandos AT, suportados pelo SIM800L e mais comumente utilizados nas aplicações propostas nesse trabalho é apresentada na Tabela 2

Tabela 2: Lista de comandos AT básicos para a aplicação estudada.

Comando	**Resposta**	**Descrição**
AT+CREG	+CREG, <n>	*Status* de conexão com a rede GSM
AT+CGREG	+CGREG, <n>	*Status* de conexão com a rede GPRS
AT+CIPMODE	+CIPMODE=n	Seleciona modo da aplicação TCP (normal ou transparente)
AT+CIPMUX	+CIPMUX=n	Habilita modo de conexão de IP único
AT+CIPSTART	OK, CONNECT STATS	Inicio da conexão TCP com o *host*
AT+CIPSTATUS	CONNECTED, IPINITIAL	*Status* da conexão TCP
AT+CFUN	AT+CFUN=<n>	Habilita ou desabilita o modo avião (sem conexão)

Fonte: (COMPANY OF SIM TECH., 2016)

3 MATERIAIS E MÉTODOS

Com o objetivo de adquirir dados suficiente para possibilitar uma análise e caracterização do consumo energético da aplicação proposta, foi necessário projetar um circuito capaz de enviar dados de metrologia enquanto adquire dados do consumo energético. Esta seção discorre sobre as principais decisões relacionadas ao protótipo bem como apresenta suas etapas de criação.

3.1 Escolha da Rede de Envio de Dados

O passo inicial para andamento do projeto é a escolha da rede a ser analisada. Mesmo com o crescimento da disponibilidade de diferentes arquiteturas para o envio dos dados de aplicações de monitoramento, as redes móveis ainda são as mais utilizadas. A preferência pelas redes móveis pode ser explicada pelo seu baixo custo inicial e a sua área de cobertura que, principalmente no Brasil, ainda é maior que a das redes emergentes, como citado na Seção 2.2. Para o desenvolvimento deste projeto foi necessária a escolha de qual rede móvel seria caracterizada, entre as opções apresentadas na Seção 2.5.

Entre as opções disponíveis, 2G (GPRS), 3G e 4G, há uma grande diferença entre a taxa máxima de envio de dados, ou seja, a velocidade de conexão. Esse ganho de velocidade implica em um crescimento no consumo energético médio e na complexidade do protocolo, o que acaba encarecendo os módulos dedicados. A Tabela 3 apresenta uma comparação entre módulos do fabricante *SIMCOM*, com aplicações similares mas suporte a diferentes tecnologias.

Para a aplicação proposta neste trabalho, monitoramento em áreas remotas, a velocidade de conexão não é um parâmetro impactante, já que o tamanho dos dados adquiridos e enviados para a rede é pequeno. Ainda, os produtos disponíveis no mercado com o mesma aplicação tendem a utilizar a rede 2G. Por fim, fica clara a preferência pela utilização da rede 2G quando o custo é levado em conta, já que um módulo que apresenta as mesmas funcionalidades mas que suporte a rede 4G, chega a custar 8 vezes mais que o módulo 2G. Por estes motivos, a rede 2G (GPRS) foi escolhida para ser o objeto de análise deste trabalho.

Tabela 3: Comparação entre módulos com suporte a 2G, 3G e 4G

	Modelo	**Suporte**	**Velocidade (*uplink*)[bits/s]**	**Preço**
	800L	2G	170k	R$40,00
	5320E	3G	2M	R$240,00
	7600CE	4G	10M	R$345,00

Fonte: O autor.

3.2 Planta

Com a definição da tecnologia a ser analisada, inicia-se o processo de projeto e desenvolvimento do protótipo do sistema, apresentado nas subseções posteriores. Primeiramente, o *hardware* é abordado, onde são apresentados todos os módulos utilizados e suas características. Após, são apresentados os *scripts* ou *firmwares* ligados a aplicação.

3.2.1 Arquitetura - Hardware

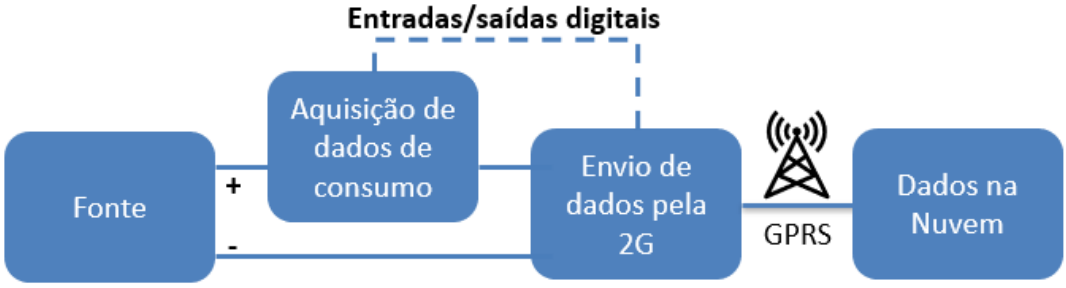
A complexidade desta aplicação dá-se pela necessidade de realizar dois processos diferentes e que, mesmo podendo ser considerados independentes, precisam ser interligados para o funcionamento global. Isto é, o envio de uma mensagem pode ocorrer normalmente mesmo que a medição de consumo não esteja ocorrendo. Também, a medição do consumo do módulo pode ocorrer a qualquer momento, estando ele enviando ou não mensagens para a rede. O leitor pode acreditar que, pelos motivos citados, as duas tarefas são completamente independentes, o que não é verdadeiro, já que o objetivo do trabalho requer identificar as parcelas relacionadas e cada etapa do processo de envio de uma mensagem, requisito que obriga as duas interfaces, a responsável pelo envio das mensagens e a responsável por adquirir os dados de consumo, a se comunicarem. Esta comunicação entre os processos é realizada por entradas digitais dos microcontroladores utilizados, atuando como *flags* que indicam o momento exato de realizar cada ação.

A Figura 7 apresenta um diagrama de blocos simplificado da aplicação, indicando onde ocorre a medição de consumo e os blocos principais necessários para o cumprimento dos objetivos propostos. Note a posição estratégica do bloco de aquisição de dados de consumo, logo após a fonte de alimentação, excluindo a necessidade de levar em conta o consumo da fonte nas estimativas. As seções subsequentes apresentam em detalhes cada um destes blocos e como ocorre seu funcionamento.

3.2.1.1 Envio de mensagens pela rede GPRS

Este pode ser considerado o principal bloco presente na aplicação desenvolvida, sendo responsável pelo envio dos mensagens pela rede GPRS, fonte principal das análises a serem realizadas no presente trabalho. O bloco conta com dois módulos independentes:

Figura 7: Diagrama de blocos da arquitetura básica da planta



Fonte: O autor.

o primeiro dedicado a conexão com a rede GPRS, que contém todo o *stack* do protocolo. O segundo, um microcontrolador responsável pelo controle do anterior, indicando o momento de iniciar a conexão e enviar uma mensagem.

Já introduzido brevemente na Seção 3.1, o módulo escolhido para lidar com a rede 2G é o módulo *SIM800L* da empresa *SIMCOM*, exibido na Figura 8. O módulo, possui a capacidade de lidar, internamente, com a rede GPRS, realizando a conexão com a torre e lidando com o envio e recebimento de mensagens quando necessário, suportando nativamente o protocolo TCP/IP para conexão com a Internet. Ainda, algumas de suas especificações técnicas importantes, segundo o *datasheet* do componente, são as seguintes:

- Tensão de alimentação: 3,4V até 4,4V
- Picos de consumo: Até 2A
- Consumo médio em conexão: 300mA

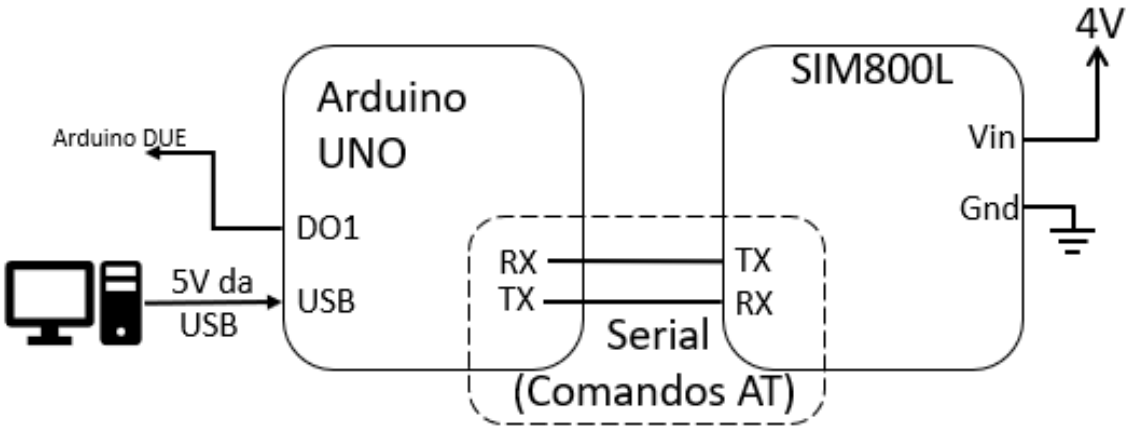
Figura 8: Módulo 2G Escolhido: SIM800L

Fonte: (FILIPEFLOP, 2019)

O microcontrolador que comanda o bloco é o ATMega328P, presente na plataforma de desenvolvimento Arduino e apresentado na Seção 2.1. Este módulo não trata diretamente com a rede GPRS, apenas realiza a comunicação com o módulo *SIM800L*, indicando o momento de iniciar uma conexão ou passando a mensagem que deve ser enviada pela rede.

A comunicação entre os dois módulos ocorre por uma interface serial TTL, suportada nativamente pelos dois módulos. A comunicação respeita a linguagem de comandos AT, apresentada na Seção 2.6. A Figura 9 apresenta um diagrama da conexão entre os dois módulos. Note que a alimentação de cada um é independente, sendo o Arduino alimentado por sua conexão USB e o SIM800L pelo bloco da fonte.

Figura 9: Diagrama da conexão entre SIM800L e o Arduino.



Fonte: O Autor.

Existe ainda uma conexão entre o pino digital DO1 do Arduino que não é ligada diretamente ao módulo SIM800L. Essa conexão é direcionada ao bloco de aquisição de dados de consumo e indica ao mesmo os momentos em que a aquisição de dados deve ser iniciada e finalizada, como no inicio e fim da conexão com a rede GSM, GPRS e MQTT ou envio de dados.

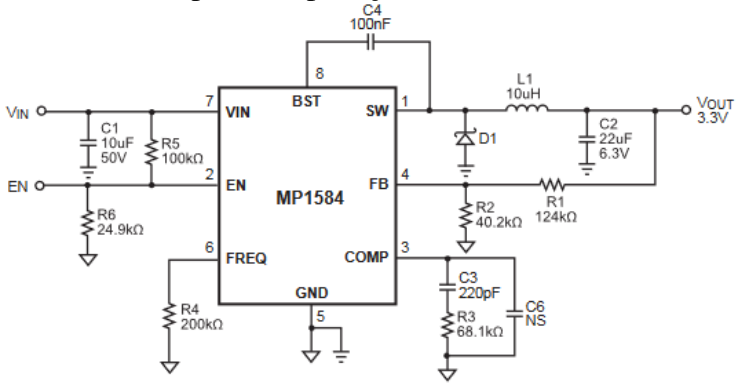
Uma máquina de estados para envio dos comandos AT para o módulo SIM800L, respeitando tempos pré-determinados, foi implementada na plataforma Arduino e é apresentada em detalhes em subseções posteriores.

3.2.1.2 Alimentação

A alimentação do módulo *SIM800L* deve ser mantida entre 3,4V e 4,4V. Segundo o fabricante, a alimentação do módulo pode ser realizada por baterias de *lithium* que costumam fornecer uma tensão de nominal de 3,6V, dentro da faixa do componente, porém não é um valor comum para fontes padrões disponíveis no mercado. Além disso, o fabricante informa que o módulo pode consumir até 2A em momentos de pico, o que deve ser levado em conta no dimensionamento do sistema de alimentação. Para atingir estes requisitos de projeto, optou-se por utilizar uma fonte com uma tensão maior que a necessária, mas que suporte picos de corrente de 2A ou mais, já que existem varias técnicas para rebaixar um valor de tensão, como conversores *buck*. Para esta aplicação, optou-se pela utilização de um conversor DC-DC rebaixador (*buck*), por apresentarem boa eficiência e precisão. A fonte escolhida, modelo R&T YX668 da marca Run&Teng, possui saída de 24V/3A. Para rebaixar a tensão de 24V para 4V, foi escolhido o conversor DC-DC MP1584, que pode rebaixar tensões de até 30V para os 4V necessários na aplicação. Ainda, o conversor suporta uma corrente nominal de 2A, chegando até 3A de pico, o que é suficiente para a aplicação necessária.

A Figura 10 apresenta um circuito conformador sugerido pelo fabricante para atender uma saída de 3,3V. Um módulo já contendo este circuito foi encontrado e adquirido para atender a aplicação.

Figura 10: Circuito típico de aplicação do conversor DC-DC MP1584.



Fonte: Fornecido pelo fabricante.

O modulo adquirido é parametrizado de fábrica para uma tensão de saída de 5V, acima da alimentação aceitável pelo módulo SIM800L. Para atingir o nível de 4,0V necessário, um resistor teve de ser dimensionado e alterado. Segundo o *datasheet* do componente, a tensão de saída do circuito é dada pelo divisor resistivo entre a tensão de saída e o pino FB do MP1584. A relação entre a tensão de saída e a tensão de *feedback* é apresentada em (1).

$$V_{out} = V_{FB}\frac{R_1 + R_2}{R_2}\ [V] \tag{1}$$

O *datasheet* do componente sugere a utilização de um resistor R_2 de $40,2k\Omega$ e diz que a tensão de *feedback* é de 0,8V. Porém, o módulo adquirido apresenta um resistor R_2 de $8,2k\Omega$. Substituindo os valores em (1) e isolando a variável de interesse (R_1), obtém-se a equação apresentada em (2).

$$R_1 = 10,25.(V_{out} - 0,8)\ [k\Omega] \tag{2}$$

Sabendo que a alimentação de entrada é de 24V e deseja-se uma saída de 4V, obtém-se o valor de $32,8k\Omega$ para a resistência a ser inserida no circuito. O valor comercial mais próximo encontrado para o resistor foi de $33k\Omega$, resultando em uma tensão de saída de $4,02V$.

3.2.1.3 Aquisição de dados de consumo

Após a definição do bloco responsável pela conexão e envio de mensagens pela rede, outro conjunto necessário para aplicação é o de aquisição dos dados relacionados ao consumo do módulo SIM800L.

O objetivo principal deste bloco é identificar as parcelas de consumo relacionadas as diferentes etapas do envio de pacotes pela rede 2G, como conexão com a rede GSM, conexão com a rede GPRS, conexão com o servidor MQTT e envio de mensagens. Para isto, é necessário uma taxa de amostragem suficiente para que seja possível reconstruir os formatos de onda associados a conexão, possibilitando então, identificar os momentos onde ocorre cada etapa. Para adquirir estes dados, a maneira mais comum encontrada na literatura é a medição do consumo de corrente do módulo, visto que sua tensão de alimentação é fixa. Assim, pela multiplicação entre a corrente consumida e a tensão é possível estimar o consumo de potencia do sistema. Portanto, além da alta taxa de amostragem, uma resolução na ordem dos miliAmpères é desejada.

Como os microcontroladores, geralmente, não suportam nativamente a leitura direta de sinais de corrente, é necessário converter um sinal de corrente para um sinal de tensão. Duas técnicas de medição de corrente foram investigadas antes da escolha final, utilizando sensor de efeito *hall* e utilizando resistor *shunt* em série com a alimentação.

Sensores de efeito *hall* são transdutores que respondem à presença de campos magnéticos com uma variação em sua tensão de saída. Diversos sensores de efeito *hall* podem ser encontrados no mercado, mas a grande maioria apresenta limites de operação na ordem de Ampères, acima da ordem de grandeza proposta para a presente aplicação. Um dos sensor com a menor excursão encontrada no mercado foi o sensor de efeito *hall* ACS714, que responde linearmente a correntes de -5A até 5A. Ainda, sua tensão de saída é centrada em 2,5V e varia 185mV/A. Levando em consideração a resolução de 10 bits do conversor ADC da plataforma Arduino que resulta em um LSB de 4,88mV pode-se perceber que a utilização deste sensor em conjunto com este microcontrolador resultaria em um LSB de 26,38mA, como apresentado em (3). Para os requisitos propostos, esta resolução não foi considerada suficiente.

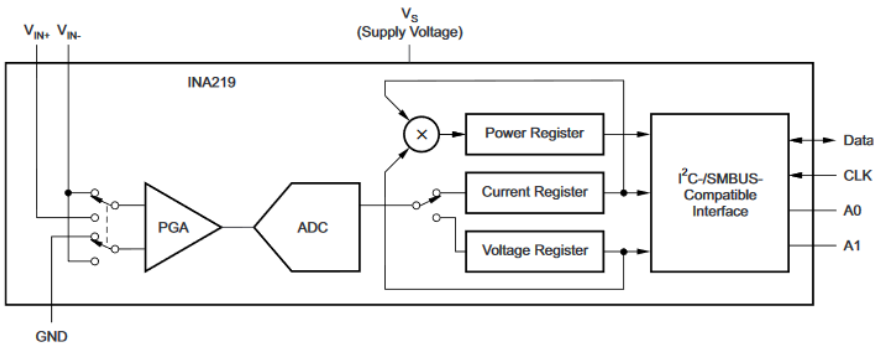
$$LSB_{ampères} = \frac{185mV}{4,88mV}.1A = 0,02638A \tag{3}$$

A segunda solução proposta é a utilização de um resistor *shunt* em série com a alimentação do módulo. O principio da utilização de um resistor é simples: a corrente que passa pelo resistor gera uma diferença de potencial sobre o mesmo que pode ser medida pela entrada analógica do microcontrolador. Como o valor do resistor é conhecido, é possível saber qual a corrente aplicada. Para isso, seria necessário a utilização de um resistor de baixa resistência, bem como algum circuito conformador de sinal. Porém, optou-se pela utilização de um módulo compatível com Arduino que já contém os componentes necessários (resistor *shunt* de $0,1\Omega$) e é baseado no circuito integrado INA219, do fabricante *Texas Instruments*. Entre as características técnicas do componente pode-se destacar as seguintes:

- Tensão de alimentação: 3V à 5,5V
- Faixa de tensão no barramento de teste: 0V à 26V
- Corrente máxima de operação: $\pm$ 3,2A
- Resolução: 0,8mA (12 bits no conversor ADC interno)
- Interface I2C

Além da medição de corrente, o módulo registra a tensão antes e após o resistor, bem como a potência consumida, calculada a partir da multiplicação entre a tensão e a corrente. O componente trabalha com registradores internos, que podem ser lidos e modificados por uma interface I2C, nativamente suportada pela plataforma Arduino. Note que a resolução de 0,8mA é maior do que a atingida pelo sensor *hall*, analisado anteriormente, e está dentro dos requisitos esperados. O fabricante ainda diz que a escrita no registrador de corrente, interesse desta aplicação, ocorre a cada 84μs, o que permite uma taxa de amostragem que corresponde aos requisitos. A Figura 11 exibe um esquemático simplificado do funcionamento do módulo. Já a Tabela 4 exibe a lista de registradores disponibilizados para leitura e escrita. Pelos motivos citados, este módulo foi o escolhido para a aplicação.

Figura 11: Esquemático simplificado INA219.



Fonte: (INSTRUMENTS, 2015)

Tabela 4: Registradores disponibilizados pelo INA219.

Endereço	**Nome**	**Descrição**	**Tipo**
00	Configuração	Configurações gerais	Leitura/Escrita
01	Tensão *Shunt*	Tensão após o resistor	Somente Leitura
02	Tensão da linha	Tensão antes do resistor	Somente Leitura
03	Potência	Potência medida	Somente Leitura
04	Corrente	Corrente no resistor	Somente Leitura
05	Calibração	Calibrações gerais	Leitura/Escrita

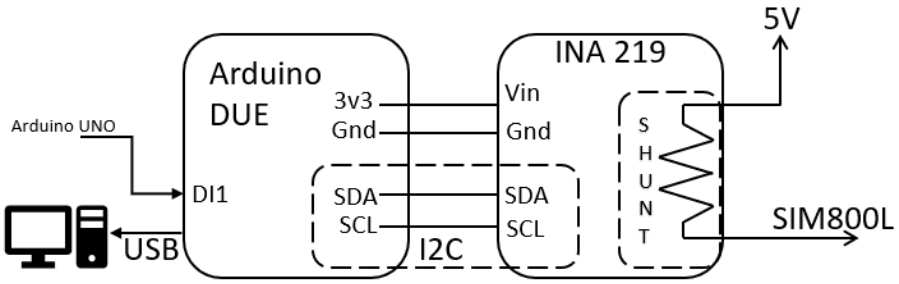
Fonte: (INSTRUMENTS, 2015)

Por configuração, é possível ativar um modo de operação onde o INA219 realiza a aferição dos dados de forma intermitente, gravando os dados obtidos em seus registradores e garantindo uma nova medida a cada 84μs. Porém, mesmo com esta interface completa e independente no que diz respeito a adquirir os dados, ainda é necessária a utilização de um microcontrolador para ler os registradores e processar os dados adquiridos, enviando-os para algum computador onde os mesmos podem ser analisados. Para esta tarefa, o microcontrolador escolhido foi um SAM3X8E, presente na placa de desenvolvimento Arduino DUE. Este microcontrolador contém um CORE ARM Cortex M3 que, entre outras características apresentadas na Seção 2.1, possui um USB nativo que pode trabalhar com altas taxas de envio de dados, comportamento importante para o presente trabalho.

A comunicação entre o DUE e o INA219 é realizada pelo protocolo I2C, que é um protocolo multimestre desenvolvido para conectar periféricos de baixa velocidade a outros de maior velocidade. Sua vantagem é de ser nativamente suportado pelas duas plataformas e ser de fácil implementação física, que exige apenas um par de fios, onde são conectados os barramentos chamados SDA (*Serial Data*) e SCL (*Serial Clock*). Já o envio dos dados para o computador é realizado pela USB nativa do Arduino DUE. O fato de ser uma USB nativa garante que os dados serão enviados na maior taxa possível e que nenhum será perdido durante o envio ao computador, já que o microcontrolador implementa uma interface de fila FIFO (*First Input First Output* internamente, enviando os dados assim que o barramento USB estiver livre. A Figura 12 apresenta um diagrama da ligação entre os componentes. Note ainda que o Arduino DUE é responsável pela alimentação do INA, conectando o V_{in} do INA a uma saída de 3v3 presente no Arduino. Por fim,

é importante destacar que uma das entradas digitais do microcontrolador é conectada a uma saída digital do Arduino UNO, utilizado no bloco de envio apresentado na Seção 3.2.1.1. Este pino serve para que o bloco de envio notifique ao bloco de aquisição de dados qual o momento certo para inicio e fim da aquisição, garantindo que os dados não serão adquiridos enquanto os módulos estiverem em estado de *stand-by*.

Figura 12: Esquemático de ligação entre o Arduino DUE e o INA219.

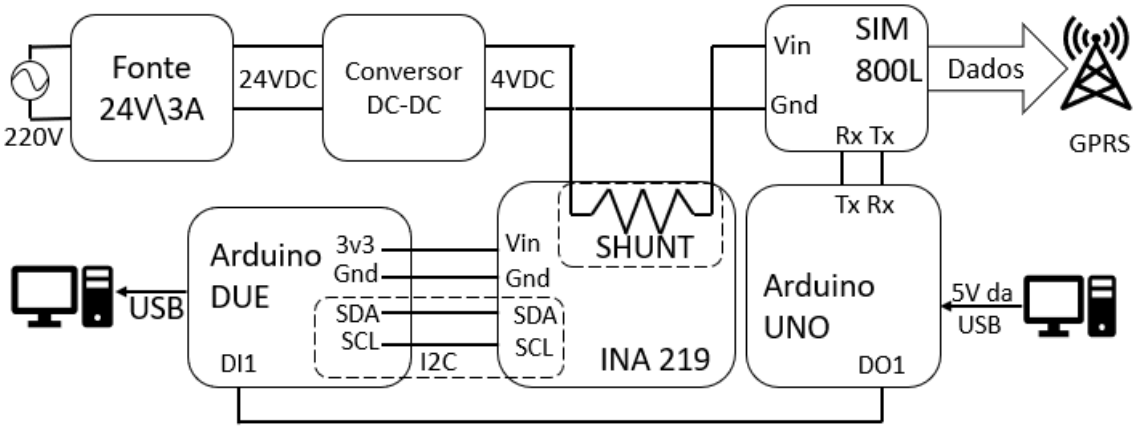


Fonte: O Autor.

3.2.1.4 Arquitetura final completa

O circuito final implementado consiste na ligação de todos os sub-blocos apresentados nas subseções anteriores. A montagem ocorreu em uma placa padrão, que é uma placa de fenolite contendo as ilhas de cobre mas não as trilhas. Dessa forma, para criar as trilhas, foram utilizados *jumpers* ou estanho. O esquemático completo do circuito implementado é apresentado na Figura 13. Foi necessária a adição de outros componentes, omitidos no esquemático final, para garantir a compatibilidade entre os blocos, como por exemplo, um divisor resistivo adicionado entre a saída digital do Arduino Uno e a entrada digital do Arduino Due. Este divisor foi necessário pois o Uno trabalha com tensões de 5V e o Due com 3V3, não sendo possível a ligação direta entre os dois componentes.

Figura 13: Esquemático final da planta



Fonte: O autor.

Apesar da ligação física do *hardware* estar completa, a sintonia entre os processos distintos realizados pelo circuito não está garantida. Por exemplo, apenas a conexão física não é suficiente para indicar ao Arduino Due o momento certo de iniciar a aquisição dos dados. A robustez do sistema só é garantida com a criação e execução dos programas a serem gravados em cada um dos microcontroladores. Estes programas são apresentados nas subseções posteriores.

Uma imagem da montagem final do *hardware* é apresentada no Apêndice B.

3.2.2 Software

Após a apresentação da montagem física do protótipo, ou seja, os módulos utilizados e a forma como eles são fisicamente arranjados e conectados, esta seção aborda os *scripts*, ou *firmwares*, responsáveis por comandar, controlar e principalmente, garantir a sintonia entre todos os módulos independentes.

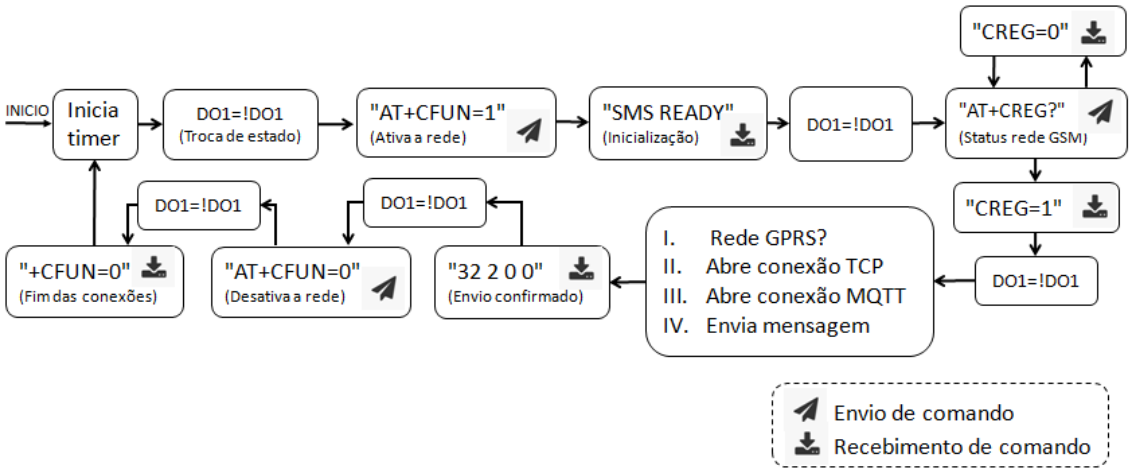
3.2.2.1 Envio de Dados pela Rede 2G (GPRS)

A principal fonte de análise deste trabalho é o módulo SIM800L, que tem a capacidade de enviar dados pela rede 2G. Porém, este componente é um módulo passivo, respondendo a comandos enviados por algum componente com inteligência. Como apresentado nas seções anteriores, o módulo escolhido para executar esta tarefa de comando é o Arduino UNO e o *firmware* gravado no microcontrolador pode ser considerado a inteligência necessária para realizar as ações previstas nos requisitos do projeto. O objetivo desta etapa é garantir a identificação dos momentos onde ocorre o fim da conexão com a rede GSM, com a rede GPRS, a abertura do *socket* TCP, conexão com o servidor MQTT e envio do dado.

A comunicação entre os dois módulos ocorre por uma interface serial, ou seja, os comandos AT, apresentados na Seção 2.6, são enviados como conjuntos de frases (*strings*) que são nativamente suportadas pelo SIM800L. Existem bibliotecas disponíveis para a plataforma Arduino que já implementam os comandos necessários, mas para garantir que a informação sobre o momento exato de conexão e envio de dados seja registrada, preferiu-se enviar os comandos AT diretamente. Para conectar ao servidor MQTT, uma conexão TCP é estabelecida, onde posteriormente os dados são enviados a nuvem.

A Figura 14 apresenta um diagrama simplificado da rotina implementada no Arduino UNO.

Figura 14: Diagrama simplificado do algoritmo implementado no Arduino UNO.



Fonte: O autor.

Note que existem estados em que a saída digital DO1 troca de nível. Esta troca é o que indica ao bloco de aquisição de dados os momentos certos para iniciar uma aquisição. Ainda, sempre que uma etapa da conexão é finalizada, a saída digital troca de nível novamente. O comportamento do algoritmo pode ser explicado da seguinte maneira: o pro-

grama inicia com um temporizador, que pode ser configurado pelo usuário. Ao fim deste tempo, a porta digital troca de nível e uma mensagem com o comando "AT+CFUN=1" é enviada pela porta serial, fazendo com que o módulo saia do estado em espera e inicie a conexão com a rede. O módulo passa por uma etapa de inicialização do *hardware* e retorna uma mensagem com os caracteres "SMS READY" ao fim da etapa. Na detecção desta mensagem, novamente a saída digital troca de nível e uma rotina de verificação da conexão com a rede GSM é iniciada.

Para saber se a conexão GSM foi estabelecida, é necessário enviar ao módulo o comando "AT+CREG?", até que uma resposta positiva seja recebida, levando a outra mudança de nível da saída digital. Uma segunda checagem de conexão é iniciada, desta vez pelo comando "AT+CGREG" que indica o sucesso da conexão com a rede GPRS levando a quarta troca de estado da saída digital. Após isso, é iniciada uma conexão TCP, com a utilização do comando "AT+CIPSTART". Aguarda-se uma nova mensagem de confirmação que resulta em outra troca de estado da saída digital. A próxima etapa do processo é a conexão com o servidor MQTT seguida do envio de uma mensagem ao servidor, ambas realizadas por meio de um pacote TCP e cuja resposta positiva resulta em novas trocas de estado da saída digital.

Com a resposta positiva de envio da mensagem verificada, o contador de mensagens é incrementado e o comando "AT+CFUN=0" enviado ao módulo, indicando que o mesmo pode retornar ao modo de espera e ocorre uma nova troca de nível da entrada digital, marcando o fim de um ciclo de envio de dado. Neste ponto, o *script* retorna ao estado inicial. É importante destacar que a mensagem enviada é um contador, formado por 2 bytes, incrementado a cada nova iteração do algoritmo.

Note que neste processo foram realizadas um total de 8 trocas de estado da saída digital, que marcam 7 etapas diferentes do ciclo completo. As etapas e os comandos recebidos que marcam seu fim são as seguintes:

- Inicialização do *hardware* do módulo: "SMS Ready"
- Conexão com a rede GSM: "+CREG=1"
- Conexão com a rede GPRS: "+CGREG=1"
- Conexão TCP: "CONNECTED"
- Conexão ao servidor MQTT: "32 2 0 0"
- Envio de uma mensagem ao servidor: "64 2 0 0"
- Fechamento das conexões: "+CFUN=0"

Foram também implementadas rotinas de verificação do tempo em cada uma das etapas do processo, a fim de evitar a entrada em *loops* infinitos que poderiam comprometer a aquisição de dados de forma recorrente. Sempre que um *time out* é percebido, a saída digital troca de estado o número de vezes necessário para marcar o fim da conexão, indicando ao bloco seguinte o fim da mesma. No caso do envio da mensagem para o servidor, a marcação de erro é realizada após 10 tentativas sem sucesso de envio de uma mensagem, diferentemente das demais etapas marcadas por limites de tempo.

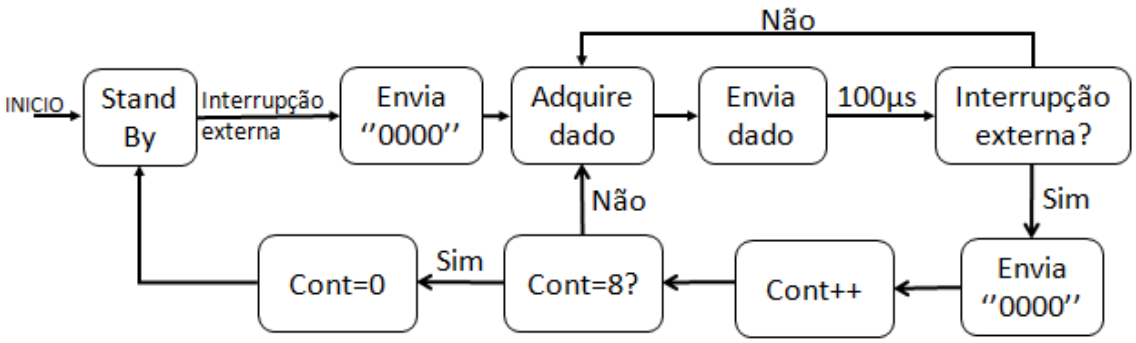
Este *script* marca o envio de mensagens pela rede GPRS ao mesmo tempo que indica ao bloco de aquisição de dados de consumo os momentos importantes que fazem parte do lote final de dados. A maneira como ocorre a operação de aquisição de informações de consumo é apresentada nas subseções posteriores.

3.2.3 Aquisição e Envio de Dados de Consumo para o Computador

Como citado nas subseções anteriores, o Arduino DUE, juntamente com o INA 219, é responsável por adquirir os dados de consumo e armazená-los, sendo o INA um componente passivo que responde aos comandos do Arduino. Para atender as especificações definidas, de adquirir um dado a cada 100μs, escolhida por ser uma amostragem suportada pelo INA, durante um período de conexão e envio de dados pela rede 2G, um *firmware* foi implementado e gravado no microcontrolador da plataforma. Um diagrama simplificado, que apresenta as etapas de um ciclo completo, é apresentada na Figura 15.

Um comportamento importante a se destacar desta etapa do processo é que, mesmo com a maior memória presente no microcontrolador do Arduino DUE, ele não é capaz de armazenar todos os dados adquiridos. Por isto, a solução encontrada foi enviar o dado adquirido para um computador, pela interface USB nativa da plataforma. Com isto, as etapas marcadas como envio de dado na máquina de estados, se referem a enviar o dado adquirido pela porta USB. Note também que, apesar de o envio de dados pela USB não garantir uma frequência fixa, ou seja, o envio de dados pela USB pode demorar tempos variados dependendo da utilização do barramento USB do computador, isto não interfere na aplicação, já que a taxa de amostragem do consumo é garantida por uma interrupção no microcontrolador que lê um dado do INA219 e é independente do envio de dados pela USB. Outra característica importante e desejável da USB nativa é a implementação de uma fila FIFO, que garante o envio de todos os dados mesmo que sem a garantia de uma frequência fixa.

Figura 15: Diagrama simplificado do algoritmo implementado no Arduino DUE



Fonte: O autor.

Dentro do *firmware*, uma interrupção externa foi criada para a entrada digital DI1, sendo chamada sempre que a mesma troca de estado.

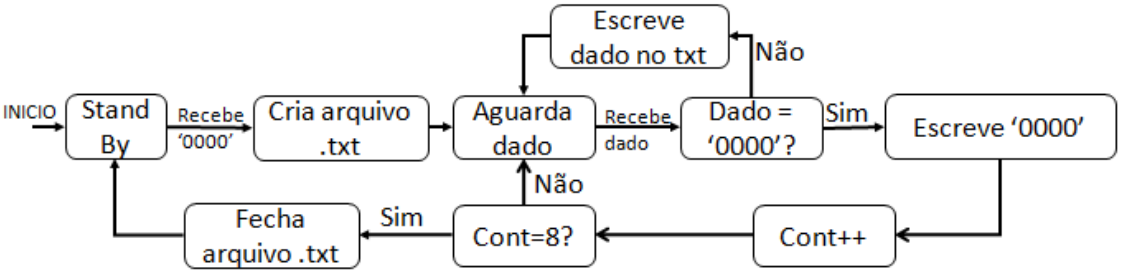
O comportamento do algoritmo pode ser explicado da seguinte maneira: o microcontrolador aguarda a troca de estado de sua porta digital (DI1, na Figura 15) para iniciar a obtenção dos dados. Esta porta, como apresentada no esquemático completo, está ligada a uma saída digital do Arduino UNO e indica ao bloco o momento de iniciar a aferição de dados, que é o momento onde se inicia a conexão do modulo SIM800L. Quando ocorre a troca de estado, um dado igual a "0000" é enviado pela porta USB e indica ao computador que o envio contínuo dos dados está iniciando. Entra-se em um laço de envio contínuo dos dados, sendo um dado adquirido e enviado a cada 100μs. Sempre que a interrupção externa é ativada, ou seja, sempre que há uma troca do estado da entrada digital, um dado formado por "0000" é enviado para a porta USB. Estas marcações realizadas por dados iguais a "0000" são posteriormente utilizadas para reconhecimento das etapas de conexão.

O *firmware* para de adquirir dados assim que a oitava troca de estado da porta é detectada, marcando o fim do ciclo completo.

Ainda, o algoritmo implementado garante que dados com zeros sejam enviados apenas nestes momentos, detectando possíveis erros ao adquirir dados iguais a zero e os modificando para “0001”.

Porém, o envio de dados pela USB não garante intrinsecamente que os dados serão armazenados no computador, nem que uma estrutura seja criado para salvar os mesmos. Dessa forma, um pequeno *script* na linguagem *Python* foi criado com o objetivo de receber os dados da USB e armazená-los em diferentes arquivos respeitando as regras impostas pela aplicação. Um diagrama simplificado, que explica este *script*, é apresentada na Figura 16.

Figura 16: Diagrama simplificado do algoritmo implementado em Python.



Fonte: O autor.

O comportamento do *script* é o seguinte: O programa aguarda o recebimento de algum dado que seja igual a “0000”, que é enviado pelo Arduino DUE. Assim que detecta o recebimento do mesmo, um arquivo de texto é criado em que o nome é um número incremental. Após a criação do arquivo, o programa entra em um laço que escreve todos os dados que chegam pela USB no arquivo de texto, até que ele perceba que um destes dados é, novamente, formado apenas por zeros. Estes dados compostos por zeros são marcações das etapas de conexão e a detecção de cada um deles implica no incremento de uma variável. A chegada desta variável a 8, ou seja, após a detecção de 8 dados compostos por zero, o arquivo de texto é fechado e salvo, levando o algoritmo ao seu estado inicial e marcando o fim do ciclo completo de envio de mensagens.

O resultado da combinação destes dois algoritmos é uma lista de arquivos de texto, que contém os dados de consumo do módulo SIM800L adquiridos a cada 100μs. Ainda, é possível detectar em cada um dos arquivos as etapas correspondentes a conexão com a rede e envio de dado para a nuvem, já que as transições são marcadas por um valor igual a “0000”, que não ocorre naturalmente nas medições.

Todos os *scripts* desenvolvidos e apresentados, estão disponíveis para verificação um um repositório online, mencionado do Apêndice A.

4 RESULTADOS

Nesta seção são abordadas a identificação das etapas relacionadas a conexão com rede analisada, bem como envio do dado pelo protocolo MQTT. Resultados associados ao consumo e envio de dados são apresentados e utilizados para a proposição de um modelo matemático para estimação do consumo energético do sistema.

4.1 Amostras adquiridas

Com o objetivo de caracterizar as etapas de envio pela rede GPRS, decidiu-se por enviar um dado para o servidor MQTT a cada 10 minutos, taxa de envio que se próxima das aplicações em áreas remotas. Inicialmente, foi escolhida a aferição de 200 amostras de ciclo completo, o que, pela taxa de envio escolhida, representa 34 horas de atividade do sistema, englobando um dia completo de atividade. Porém, foi detectada uma taxa de perda dos pacotes enviados, resultando em amostras incompletas (onde não foram englobadas todas as etapas de conexão), já que o *firmware* desenvolvido implementa uma rotina de *time out*. Esta taxa de perdas é abordada com mais detalhes nas seções posteriores.

Portanto, desta maneira foram adquiridas 144 amostras de ciclo completo, o que, a uma taxa de envio de 10 minutos, resulta em 24 horas de atividade do sistema. Esta amostras são utilizadas para a estimação do modelo de consumo, apresentado nas seções posteriores.

Por tratar-se de uma análise por amostragem, onde a população é infinita, existe um erro amostral associado a média calculada, indicando com um certo grau de confiança, que novas amostras devem se manter dentro da margem de erro. Esta margem de erro é dada pela Equação 4.

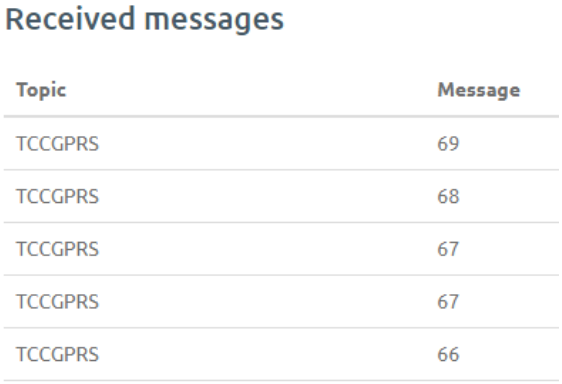
$$\varepsilon = z * \frac{\sigma}{\sqrt{n}} \tag{4}$$

Onde z é o escore-z (ou escore padrão), utilizado com o valor de 1,96, que corresponde a um grau de confiança de 95%. Já σ é o desvio padrão das amostras, calculado para cada etapa de conexão. E, por fim, n é o número de amostras obtidos, que neste caso é igual a 144. Esta equação é utilizada posteriormente para o calculo do erro amostral das etapas de envio de um dado.

A Figura 17 mostra a interface do servidor MQTT, onde dados foram enviados associados ao tópico "TCC2G". A mensagem enviada é um contador, que é incrementado a cada novo envio. Note que, pela utilização do nível 1 de QoS, como apresentado na Seção 2.4, as mensagens podem ser duplicadas, já que o *firmware* implementado verifica o recebimento de um dado, reenviando caso não receba a confirmação do sistema, mas

não garante que este envio é realizado apenas uma vez. Mesmo assim, esta necessidade de reenvio de dados acaba sendo modelada no consumo energético total do sistema.

Figura 17: Mensagens recebidas no servidor MQTT



Fonte: O autor.

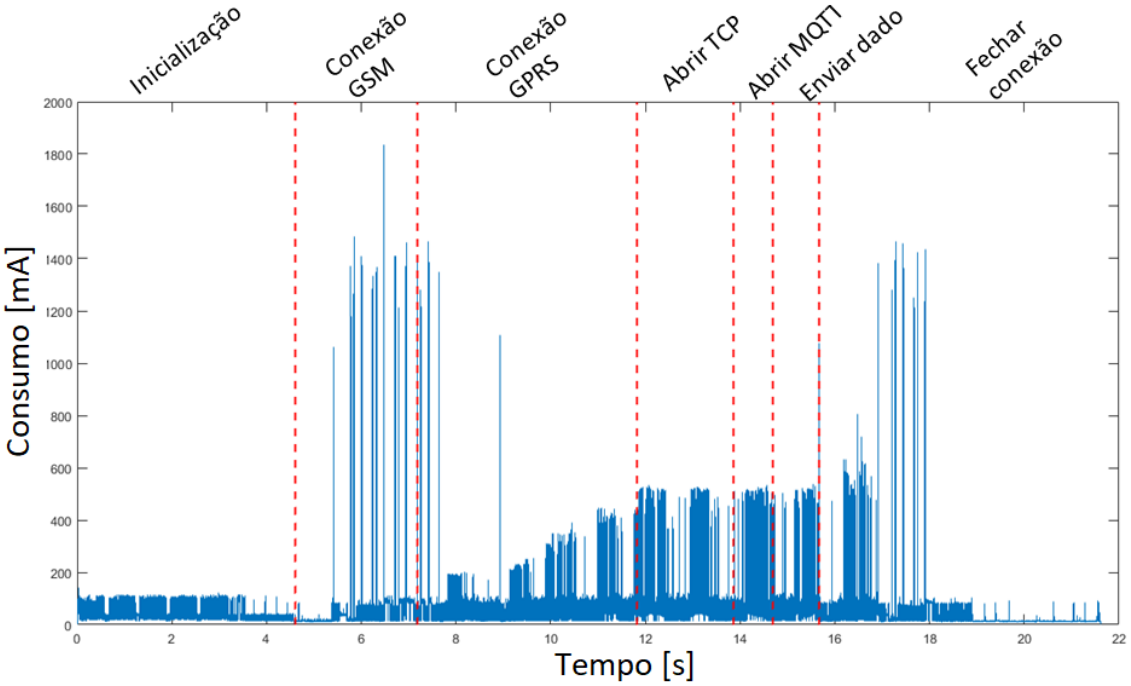
4.2 Análise do ciclo completo

A partir das amostras adquiridas pelo hardware desenvolvido, foi possível identificar e caracterizar as 7 etapas previstas do ciclo de envio de um dado pela rede GPRS com o protocolo MQTT. Os dados adquiridos, para cada ciclo, foram importados no software *Matlab*, onde realizou-se o processamento e verificação dos mesmos. Entre as operações realizadas no *Matlab*, pode-se destacar a identificação do início e fim de cada etapa, marcadas pela presença de um dado de valor zero, como mencionado nas subseções anteriores.

Para cada uma das etapas, e também para o ciclo completo, a integral do sinal foi processada de forma a adquirir o valor do consumo energético do sistema (em mA.s). A integral foi calculada numericamente pelo método da integração trapezoidal, disponível no *Matlab* pela função `trapz(X,Y)`. A unidade de $mA.s$ (mili-Ampères vezes segundos) foi escolhida para representar o consumo do sistema pois é comumente utilizada na literatura para dimensionar a carga de baterias ou pilhas, facilitando a posterior estimativa da duração das mesmas. Porém, dados como o consumo em mW.s também estão disponíveis já que são fornecidos nativamente pelo INA219, presente no bloco de aquisição de dados. Ainda, o tempo de duração de cada etapa, em segundos, foi calculado levando-se em consideração o número de amostras em cada etapa e a taxa de amostragem, de 100μ segundos.

A Figura 18 exibe o resultado de um ciclo completo de envio de dados, onde o eixo das abscissas é mensurado em segundos e o eixo das ordenadas em mA. É possível notar picos de consumo, próximos a 2000 mA, durante a conexão GSM, conexão GPRS e fechamento da conexão, bem como uma duração do ciclo completo de cerca de 22 segundos.

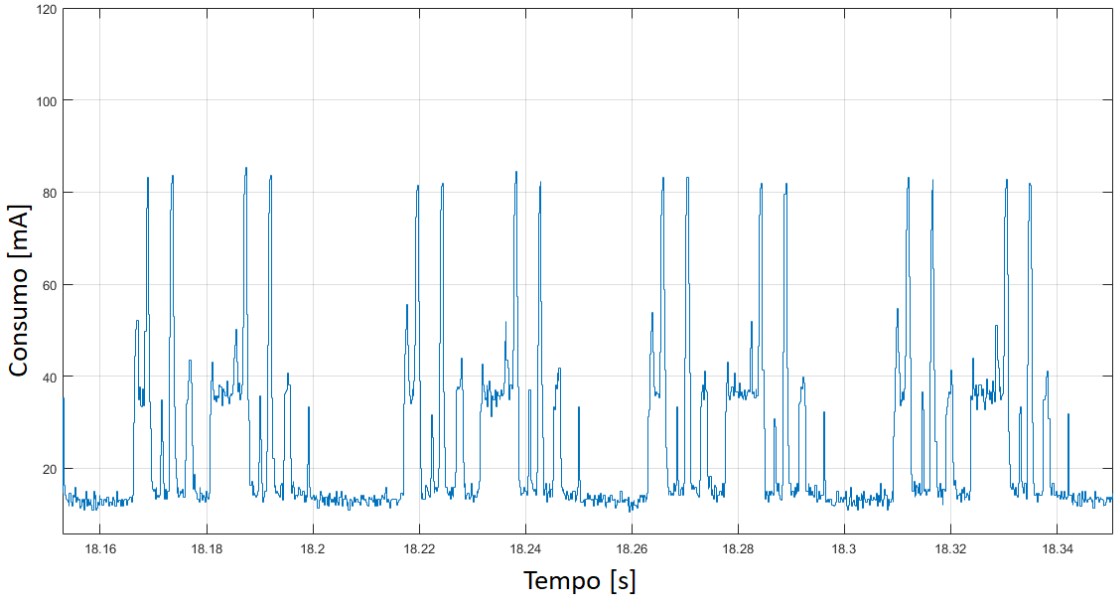
Figura 18: Exemplo de um ciclo completo



Fonte: O autor.

Note que a Figura 18 tem uma escala de tempo na ordem de segundos, onde não é possível perceber os formatos de onda associados a cada uma das etapas. Para ilustrar a taxa de amostragem utilizada e as informações adquiridas com a mesma, a Figura 19 exibe um uma parcela da etapa de fechamento da conexão, onde percebe-se mais detalhes nas amostras adquiridas, sendo possível perceber uma repetição dos dados.

Figura 19: Zoom no fechamento da conexão



Fonte: O autor.

Como especificado na metodologia, foram adquiridas 144 amostras de ciclo completo, executadas a cada 10 minutos, correspondendo a 24 horas do sistema em execução. O

processamento de todas estas amostras resultou nos valores apresentados nas tabelas 5 e 6.

A Tabela 5 exibe as médias encontradas para os tempos de cada uma das etapas de envio do dado. Apresenta também o desvio padrão, seguido do valor máximo e mínimo encontrados e, por fim, a margem de erro amostral para uma confiabilidade de 95%, calculada pela equação apresentada na Seção 4.1, para os 144 ciclos completos adquiridos. É possível notar que a maior variabilidade de tempo ocorre na etapa de conexão com a rede GPRS, onde um desvio padrão de 2,1818 segundos foi calculado e um o tempo de conexão máximo de 19,42 segundos foi observado. Já a etapa com menor variabilidade ocorre na inicialização do módulo, onde um desvio padrão de apenas 0,0096 segundos é apresentado. Esta baixa variabilidade na etapa de inicialização pode ser explicada pois a mesma depende apenas do *hardware* do módulo SIM800L, diferentemente da etapa de conexão GPRS, onde o módulo se comunica com a torre para realizar a conexão, estando sujeito a maiores interferências. Ainda, as etapas de conexão TCP e envio de dados também apresentam alta variabilidade. Com relação ao envio de dados, isto pode ser explicado pela necessidade de reenviar um dado, caso a primeira tentativa tenha falhado. O sistema foi projetado para tentar reenviar o dado até 10 vezes.

Tabela 5: Tempos para cada etapa do ciclo (em segundos)

Etapa	**Média**	**Desvio**	**Máximo**	**Mínimo**	**Margem de Erro**
Inicialização	4,5816	0,0096	4,6072	4,5548	95% $\pm$ 0,0019
Conexão GSM	3,1564	0,8574	5,1306	1,5649	95% $\pm$ 0,1681
Conexão GPRS	4,3834	2,1818	19,4201	3,6277	95% $\pm$ 0,4276
Conexão TCP	2,8604	1,0639	9,0070	1,6948	95% $\pm$ 0,2085
Conexão MQTT	1,1242	0,2925	1,8675	0,7689	95% $\pm$ 0,0573
Enviar dado	1,4757	0,9625	8,5022	0,7986	95% $\pm$ 0,1887
Fechar conexão	5,9866	0,8224	13,6454	5,6106	95% $\pm$ 0,1612
Ciclo completo	23,511	2,7227	38,1710	20,4220	95% $\pm$ 0,5336

Fonte: O Autor.

Já a Tabela 6 apresenta os dados do consumo energético de cada uma das etapas. A média é calculada a partir da integral numérica das etapas, realizada no *Matlab*. Também é exibida uma coluna com a média na unidade de mA.h. Assim como observado nos tempos, a maior variabilidade ocorre na etapa de conexão GPRS e a menor na inicialização do módulo. Foi possível observar também que o maior pico de conexão encontrado, entre todas as amostras, foi de 1971 mA, o que condiz com as especificações do fabricante, que alerta os projetistas sobre picos de consumo de até 2 A em momentos de conexão com a rede. Por fim, a margem de erro amostral é apresentada para uma confiabilidade de 95%.

Tabela 6: Consumo para cada etapa do ciclo

Etapa	Média mA.s	Desvio mA.s	Média mA.h	Pico máximo mA	Margem de Erro mA.s
Inicialização	248,91	1,29	0,06914	1117,6	95% ± 0,25
Conexão GSM	106,81	17,17	0,02967	1954,4	95% ± 3,36
Conexão GPRS	259,06	68,89	0,07196	1738,3	95% ± 13,50
Conexão TCP	205,89	58,33	0,05719	1442,6	95% ± 11,43
Conexão MQTT	90,12	18,93	0,02503	1304,7	95% ± 3,71
Enviar dado	104,82	56,37	0,02912	1369,3	95% ± 11,05
Fechar conexão	190,38	36,36	0,05289	1971,0	95% ± 7,13
Ciclo completo	1260,17	137,42	0,35005	1971,0	95% ± 26,93

Fonte: O Autor.

4.3 Modelo do ciclo de envio

Com o objetivo de estimar a duração de baterias para diferentes taxas de envios de dados, propõe-se um modelo matemático para o consumo energético do módulo estudado, baseado nos dados adquiridos e apresentados nas subseções anteriores. Para isto, a equação que modela o consumo do módulo, em mA, levando em consideração diferentes taxas de envio de dados por hora, é apresentado na Equação 5.

$$Consumo\ [mA] = \frac{1260,174\ mA.s}{3600\ s} * DadosPorHora \tag{5}$$

Onde 1270,17 mA.s é o consumo total do ciclo de envio, dado pela soma do consumo de todas as etapas, como apresentado na Tabela 6. Já 3600 é o número de segundos em uma hora, resultando em uma média do consumo durante uma hora. "DadosPorHora" é a variável que define a taxa de envio de dados, para a qual deseja-se estimar o consumo energético. Por exemplo, para enviar um dado a cada 10 minutos, temos que o valor de "DadosPorHora" é igual a 6. O resultado desta equação é o consumo constante, em mA, ao longo de 1 hora.

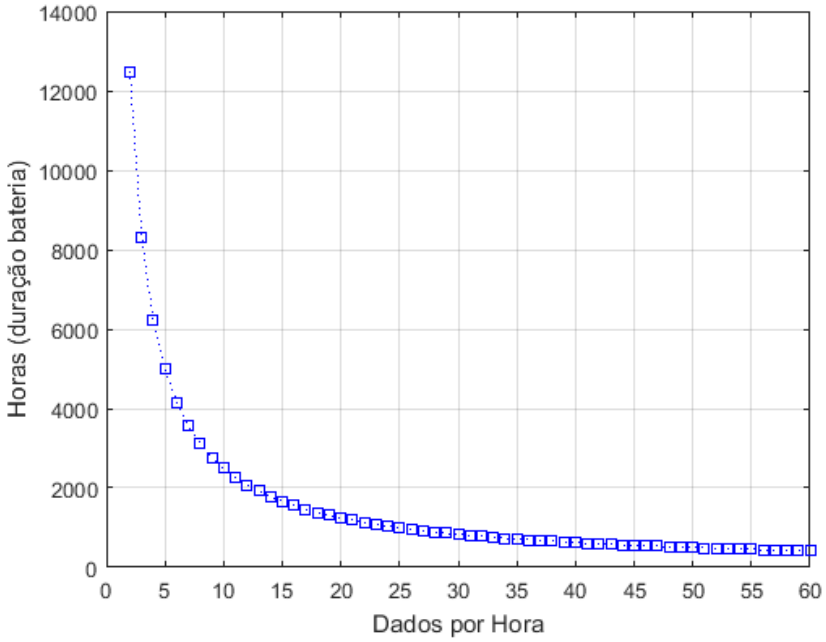
Esta equação pode ser utilizado em conjunto com modelos de duração de baterias, que geralmente são não lineares e levam em consideração variáveis como temperatura e SOH (*State of Health*) de baterias, para estimar sua curva de duração com relação ao consumo. Porém, apenas para fins de ilustração de utilização do modelo, o mesmo foi aplicado a um modelo linear de duração de baterias, que considera que toda sua capacidade (em mAh) é consumida de forma linear, não considerando o comportamento dinâmico das mesmas, o que é uma aproximação com grande erro com relação a duração real. Leva-se em consideração o consumo calculado pela equação 5 sendo o divisor da capacidade da bateria, comercialmente definida em mA.h. Chega-se, então, a Equação 6, definindo de maneira aproximada a duração em horas da bateria.

$$Duração\ Bateria\ [horas] = \frac{Capacidade\ [mAh]}{Consumo\ [mA]} \tag{6}$$

A fim de aplicar esta aproximação, considera-se a utilização de uma bateria de *Lithium* de 4,2V e capacidade de 8800 mA.h. Baterias com estas configurações são facilmente

encontradas no mercado e possuem a tensão necessária para alimentação do módulo SIM800L (4,2V). Aplicando-se as equações 5 e 6 para a bateria selecionada, temos como resultado o gráfico apresentado na Figura 20, que exibe a variação da duração da bateria (em horas) com relação a alteração da taxa de envio de dados.

Figura 20: Duração da bateria ao aumentar a taxa de envio



Fonte: O autor.

Alguns valores considerados importantes de duração de bateria, são exibidos na Tabela 7. O envio de 1 dado a cada 10 minutos é uma taxa comum nas aplicações citadas nas seções anteriores. Já o envio de 156 dados por hora, corresponde a não ter intervalo entre o envio dos dados, ou seja, é o pior caso de duração da bateria para a aplicação proposta.

Tabela 7: Duração da bateria para diferentes taxas

1 dado a cada	**Dados por hora**	**Duração (horas)**	**Duração (dias)**
60 min	1	20111,5	837
10 min	6	3351,8	139
5 min	12	1675,3	70
1 min	60	335,19	14
0,4 min	156	129,3	5,37

Fonte: O Autor.

Apesar do modelo de duração das baterias com relação ao consumo utilizado ser um modelo simplificado, que não considera o comportamento dinâmico das mesmas estando longe de representar o comportamento real em utilização, pode-se retirar informações que indicam a redução da duração da bateria para diferentes taxas de envios de dados.

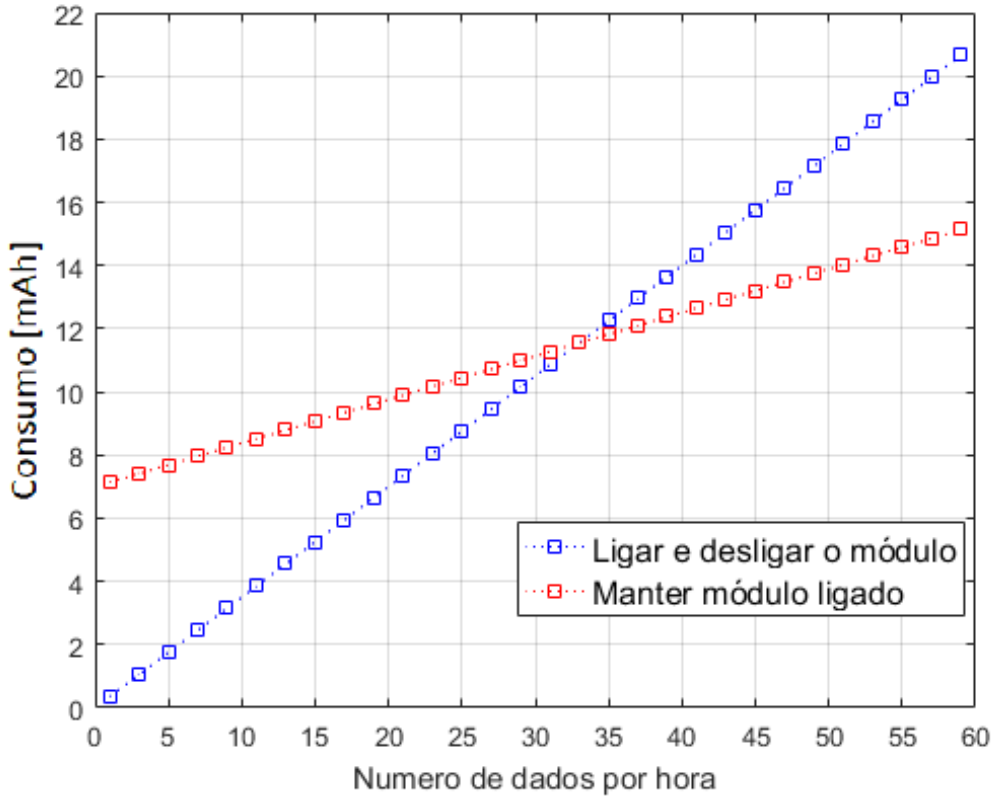
4.3.0.1 *Modelo mantendo o módulo ligado*

A aplicação analisada neste trabalho considera o desligamento do módulo SIM800L entre o envio dos dados, ou seja, o consumo zero quando o ciclo de envio não esta ocorrendo. Uma segunda opção para atender a esta aplicação, seria manter o módulo SIM800L ligado enquanto o envio de dados não está ocorrendo. Isto implica em um consumo constante do módulo, mas faz com que as etapas de inicialização, conexão GSM e conexão GPRS não sejam necessárias. O fabricante do módulo indica que o consumo, no modo *idle*, fica abaixo de 7 mA. Dessa forma, pode-se propor um segundo modelo, onde não ocorre o desligamento do módulo e, consequentemente, não há necessidade das etapas de conexão com a rede GSM e GPRS em cada envio. Chega-se então ao modelo representado pela Equação 7.

$$Consumo\ [mA] = \frac{614,78\ mA.s}{3600\ s} + (\frac{496,031\ mA.s}{3600\ s} * DadosPorHora) + 7mA \quad (7)$$

Onde o valor de 614,78 indica as etapas de inicialização do módulo, conexão com a rede GSM e conexão com a rede GPRS, etapas que ocorrem apenas uma vez na aplicação, quando o módulo é ligado. O valor de 496,031 mA.s corresponde a soma das etapas de conexão TCP, conexão MQTT, envio de dado e metade do fechamento da conexão, representando o fim da conexão TCP e MQTT mas sem desconectar da rede GSM e GPRS. Por fim, o valor constante de 7mA representa o consumo ininterrupto do módulo no modo *idle*. A Figura 21 exibe uma comparação entre os dois modelos apresentados, considerando o desligamento do módulo entre conexões e mantendo o módulo ligado.

Figura 21: Comparação entre modelos



Fonte: O autor.

É possível perceber um ponto de intersecção entre as duas retas, onde o consumo relacionado a ligar e desligar o módulo é maior do que se o módulo fosse mantido desligado. O ponto onde esta intersecção ocorre é no envio de 33 mensagens por hora (aproximadamente uma mensagem de 2 bytes a cada 2 minutos). Desta forma tem-se que é mais eficiente manter o módulo ligado entre envios, caso a taxa de envio seja maior do que 1 dado a cada 2 minutos.

4.4 Análise da perda de dados

O *firmware* responsável pelo envio dos dados para o servidor implementa rotinas de verificação de duração de cada etapa, evitando possíveis *loops* infinito, causados por qualquer erro de conexão. Dessa forma, existem amostras onde o envio do dado para o servidor não é completo.

Na aplicação estudada, dos 200 dados enviados, 41 foram perdidos, resultando em uma taxa de perda de 20,45%. Pela forma como o *firmware* foi implementado, é possível verificar em qual etapa do processo o erro ocorreu, já que, quando ocorre um *time out*, uma sequência de zeros é enviada para o computador forçando o fechamento do arquivo de texto. A Tabela 8 mostra informações sobre os dados perdidos. Note que a identificação de um erro na conexão TCP é caracterizada por uma duração maior do que 35 segundos. Já um erro na etapa de envio de dado é detectada após 10 tentativas de envio sem o recebimento de um *acknowledge* do servidor.

Tabela 8: Classificação da perda dos dados

Etapa	**Dados perdidos**	**Porcentagem**	***Time out***
Abrir conexão TCP	8	19,5%	35 segundos
Enviar dado	33	80,48%	10 tentativas

Fonte: O Autor.

Apesar de não fornecer informações adicionais sobre o motivo pelo qual o erro ocorreu, a identificação das etapas com maior propensão ao erro pode auxiliar o projeto de *firmwares* mais robustos, indicando quais etapas devem ser estudadas de maneira mais profunda.

5 CONCLUSÕES

Considerando o objetivo deste trabalho, de identificar e caracterizar o consumo energético de um ciclo de envio de dados pela rede GPRS, pode-se considerar que, através do protótipo desenvolvido, foi possível obter amostras suficientes para criação de um modelo matemático do consumo das diferentes etapas de conexão. O baixo desvio padrão encontrado para etapa de inicialização indica que a instrumentação desenvolvida tem boa repetibilidade, corroborando com a hipótese de que as variações nas demais etapas são causadas pela conexão com a rede e servidor. Com o modelo obtido é possível estimar o consumo energético para uma dada aplicação de maneira mais adequada, diminuindo a subjetividade neste aspecto do projeto.

O modelo matemático apresentado pode ser utilizado em conjunto com modelos mais complexos, que representam de maneira mais fiel o comportamento dinâmico de baterias, o que pode fornecer uma grandeza a ser tomada como base no projeto de produtos que façam uso desta tecnologia.

A utilização em conjunto dos dados obtidos de forma empírica e os dados de consumo fornecidos pelo fabricante, com relação ao modo *idle* do dispositivo, permitiu a modelagem de um segundo modo de funcionamento para o sistema, onde considerou-se o dispositivo sempre ligado. Com a comparação entre estes dois modos de funcionamento, é possível afirmar que existe um ponto onde o gasto energético em manter o módulo ligado é menor do que o necessário para religar o módulo e realizar a conexão com a rede. Estima-se que este ponto seja em uma taxa de envio de dados a cada 1,8 minutos, mas faz-se necessária uma maior investigação no consumo real do modo *idle*.

Já a identificação das etapas de abrir conexão TCP e envio de um dado como maiores pontos de falha de conexão, apesar de não apresentar os motivos, indica onde deve-se dedicar esforços no caso de projeto de *firmwares* com maior robustez, auxiliando e guiando projetistas.

Como sugestão para trabalhos futuros, o desenvolvimento de um *firmware* mais robusto, de forma a identificar o motivo da alta taxa de perda de dados pode enriquecer o modelo. Ainda, o aprofundamento do funcionamento do protocolo MQTT, bem como a comparação do consumo energético do mesmo para com outras opções de protocolos disponíveis, pode permitir a criação de diferentes modelos e enriquecer as comparações entre os modos de operação na aplicação estudada.

REFERÊNCIAS

ATZORI, L.; IERA, A.; MORABITO, G. The Internet of Things: A survey. *Computer Networks*, v. 54, n. 15, p. 2787–2805, 2010. ISSN 1389-1286.

BERNAL, P. S. M. *Comunicações móveis: tecnologias e aplicações*. [S.l.]: Erica, 2002.

BORGIA, E. The Internet of Things vision: Key features, applications and open issues. *Computer Communications*, v. 54, out. 2014.

BOSWARTHICK, D.; ELLOUMI, O.; HERSENT, O. *M2M Communications: A Systems Approach*. [S.l.]: Wiley, 2012. (EBL-Schweitzer). ISBN 9781119994756.

CLOUDMQTT. Hosted message broker for the Internet of Things. In: Disponível em: `https://www.cloudmqtt.com/docs/index.html`. Acesso em: 22 set. 2019.

COMPANY OF SIM TECH., S. A. SIM800 Series$_A TCommandManual_V$1.10. In: Disponível em: `https://www.cloudmqtt.com/docs/index.html`. Acesso em: 22 set. 2019.

ETSI, E. T. S. I. TS 102690 version 2.1.1: Especificação técnica. In: Disponível em: `http://www.etsi.org/deliver/etsi_ts/102600_102699/102690/02.01.01_60/ts_102690v020101p.pdf`. Acesso em: 19 out. 2019.

EUROTECH, I. B. M. C. MQTT specification version 3.1. out. 2010. In: PROCEEDINGS of the 24th WORKSHOP ON REAL-TIME PROGRAMMING. Schloss Dagstuhl, Germany: [s.n.], 1999. p. 9–14. Disponível em: <`https://public.dhe.ibm.com/software/dw/webservices/ws-mqtt/mqtt-v3r1.html`. Acesso em: 22 set. 2019.

FERRAZ, L. D. C.; GARCIA, R.; NUNES, D. Estudo comparative entre sistemas propostos para 4G: LTE e WiMAX móvel. In: CONGRESSO de Iniciação Cientifica do Inatel–INICITEL, Santa Rita do Sapucaı. [S.l.: s.n.], 2012.

FILIPEFLOP. Modulo SIM800L. In: Disponível em: `https://www.filipeflop.com/produto/modulo-gsm-gps-bluetooth-sim808/`. Acesso em: 28 nov. 2019.

GARG, V. K.; SMOLIK, K.; WILKES, J. E. *Applications of CDMA in wireless/personal communications*. [S.l.]: Prentice-Hall, Inc., 1997.

GRUBER, V. Sistema de monitoramento remoto baseado em rede de celular GSM/GPRS para gerenciamento de desgaste de pastilha de freio e vibração da torre em aerogeradores, 2007.

HIVEMQ. Hosted MQTT broker. In: Disponível em: `https://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-quality-of-service-levels/`. Acesso em: 28 nov. 2019.

HOLLER, J. et al. *From Machine-to-machine to the Internet of Things: Introduction to a New Age of Intelligence.* [S.l.]: Academic Press, 2014.

INSTRUMENTS, T. INA219Zero-Drift, BidirectionalCurrent/PowerMonitorWith I2C Interface. In: Disponível em: `https://www.ti.com/lit/ds/symlink/ina219.pdf`. Acesso em: 06 out. 2019.

KHAN, M. I.; GANSTERER, W. N.; HARING, G. Static vs. mobile sink: The influence of basic parameters on energy efficiency in wireless sensor networks. *Computer Communications*, v. 36, p. 965–978, 2013.

MAINETTI, L.; PATRONO, L.; VILEI, A. Evolution of wireless sensor networks towards the Internet of Things: A survey, p. 1–6, set. 2011.

MCROBERTS, M. *Arduino Básico*. [S.l.]: Novatec Editora, 2018. ISBN 9788575227114.

MONK, S. *Programação com Arduino: Começando com Sketches*. [S.l.]: Bookman Editora, 2017. ISBN 9788582604472.

TALAVERA, J. M. et al. Review of IoT applications in agro-industrial and environmental fields. *Computers and Electronics in Agriculture*, Elsevier, v. 142, p. 283–297, 2017.

TANENBAUM, A. S. *Redes de Computadores: Redes de Computadores*. [S.l.]: Editora Campus, 1997.

TRAPPEY, A. J. et al. A review of essential standards and patent landscapes for the Internet of Things: A key enabler for Industry 4.0. *Advanced Engineering Informatics*, v. 33, p. 208–229, 2017. ISSN 1474-0346.

VERMESAN, D. O. Internet of Things: Strategic Reserach Roadmap. *CERP IoT – Internet of things european research cluster*, v. 54, out. 2009.

WALDMAN, H.; YACOUB, M. D. *Telecomunicações-princıpios e tendências*. [S.l.]: Editora Érica, 2000.

WANG, K.; WANG, Y. et al. Green Industrial Internet of Things Architecture: An Energy-Efficient Perspective. *IEEE Communications Magazine*, v. 54, n. 12, p. 48–54, dez. 2016.

WANG, L.; MANNER, J. Energy Consumption Analysis of WLAN, 2G and 3G interfaces. *2010 IEEE/ACM Intel Conference on Green Computing and Communications: Cyber, Physical and Social Computing*, p. 300–307, 2010.

ZHICONG, Q.; DELIN, L.; SHUNXIANG, W. Analysis and Design of A Mobile Forensic Software System Based on AT Commands. *2008 IEEE International Symposium on Knowledge Acquisition and Modeling Workshop*, p. 597–600, 2008.

APÊNDICE A SCRIPTS DESENVOLVIDOS

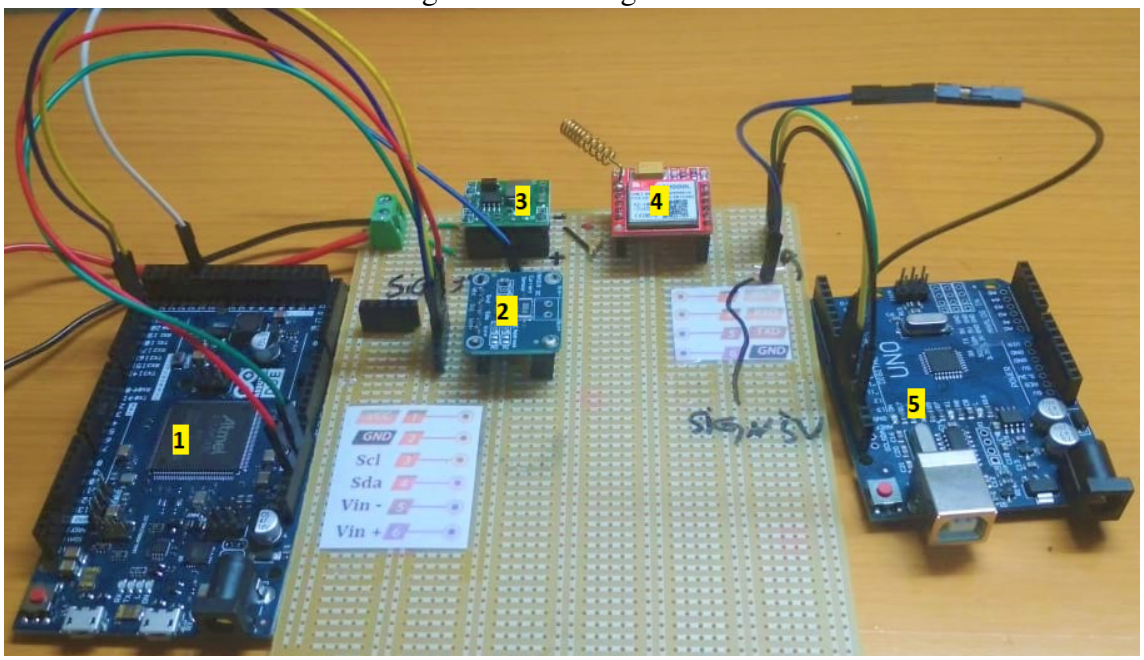
Para uma maior compreensão dos códigos desenvolvidos, sem que seja necessária a sua completa adição ao presente documento, os mesmos foram disponibilizados na plataforma de compartilhamento de códigos GitHub, acessível pelo seguinte endereço:

https://github.com/igordiesel/TCC2G

APÊNDICE B IMAGENS

A montagem final do protótipo é apresentado na Figura 22.

Figura 22: Montagem final.



Fonte: O Autor.

Os itens referenciados na Figura 22 são os seguintes:

1. Arduino DUE
2. INA219
3. Conversos DC-DC
4. SIM800L
5. Arduino UNO