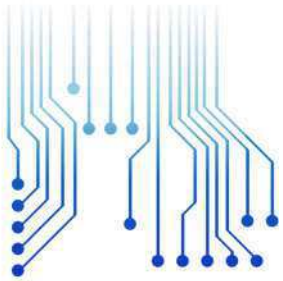
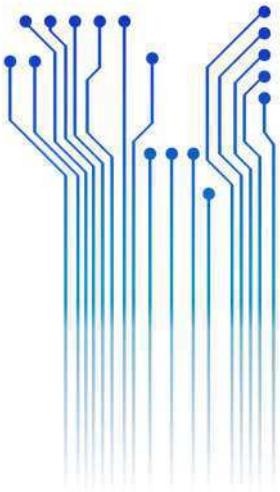


CURSO DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA

HUMBERTO DE BRITO RANGEL NETO

RELATÓRIO DE ESTÁGIO

DESENVOLVIMENTO DE APLICAÇÃO DE BAIXO CONSUMO ENERGÉTICO COM CONECTIVIDADE SEM FIOS



Campina Grande
2019

HUMBERTO DE BRITO RANGEL NETO

DESENVOLVIMENTO DE APLICAÇÃO DE BAIXO CONSUMO ENERGÉTICO COM CONECTIVIDADE SEM FIOS

Relatório de Estágio Integrado submetido à Unidade Acadêmica de Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Eletrônica de Sistemas Embarcados

Orientador:
Professor Gutemberg Gonçalves dos Santos Júnior, D. Sc.

Campina Grande
2019

HUMBERTO DE BRITO RANGEL NETO

DESENVOLVIMENTO DE APLICAÇÃO DE BAIXO CONSUMO ENERGÉTICO COM CONECTIVIDADE SEM FIOS

Relatório de Estágio Integrado submetido à Unidade Acadêmica de Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Eletrônica de Sistemas Embarcados

Aprovado em 17 de Abril de 2019.

Marcos Ricardo Alcântara Morais
Universidade Federal de Campina Grande
Avaliador

Gutemberg Gonçalves dos Santos Júnior, D. Sc.
Universidade Federal de Campina Grande
Orientador, UFCG

Dedico este trabalho primeiramente a Deus, que por Sua graça salva pecadores, e aos pais que Ele me deu, que desde muito cedo se esforçam em me pavimentar os caminhos da vida.

AGRADECIMENTOS

Agradeço sinceramente aos engenheiros da NXP que me acompanharam durante o período de estágio, em especial ao meu tutor Marco Merlin, pelo conhecimento e experiência compartilhada, disposição contínua em ajudar e confiança para dar e receber sugestões e feedback.

Agradeço aos professores e funcionários da Universidade Federal de Campina Grande (UFCG) que me preparam para essa experiência bem-sucedida, provendo as ferramentas que me foram úteis e também necessárias. Em especial, agradeço aos professores cuja empatia para com os alunos e zelo pelo conhecimento andam lado a lado: Marcos Morais e Gutemberg Júnior, pelas aulas e laboratórios de Arquiteturas Digitais, Circuitos Lógicos e orientação nos Projetos de Excelência em Microeletrônica, assim como pela orientação final deste relatório; Edmar Gurjão, desde as aulas de Álgebra Linear até o suporte no fim do curso; Damásio Júnior, desde o primeiro período. Agradeço sobretudo ao professor Carlos de Araújo, por desenvolver o projeto de intercâmbio na UFCG com contínua atenção e cordialidade memoráveis.

Agradeço a CAPES, pela provisão da bolsa de estudos do Programa BRAFITEC, e ao *Institut National des Sciences Appliquées* (INSA Toulouse) e a seus professores, pela recepção e portas abertas na procura de estágios.

"And men go abroad to admire the heights of mountains, the mighty waves of the sea, the broad tides of rivers, the compass of the ocean, and the circuits of the stars, yet pass over the mystery of themselves without a thought."
— St. Augustine of Hippo, Confessions.

RESUMO

Descreve-se neste relatório as atividades realizadas durante o período de estágio integrado no setor de Conectividade da NXP SEMICONDUCTORS FRANCE. O estágio teve como objetivo o desenvolvimento de uma aplicação de firmware em C++, a ser embarcada no microcontrolador NXP QN9080, de baixo consumo de energia. A autonomia de um botão sem fios, com conexão Bluetooth e alimentado por bateria de 220mAh, deveria ser no mínimo cinco anos, utilizando a placa NXP QN9080-DK e seu respectivo *Software Development Kit* (SDK) como base. Fazendo uso dos subsistemas embarcados na NXP QN9080-DK, do SDK, do padrão *Bluetooth Low Energy* (BLE), de medições do consumo de corrente elétrica em diferentes configurações, estimações de autonomia da bateria e em conformidade com os requisitos iniciais, o estagiário desenvolveu uma aplicação de autonomia estimada em cinco anos e dois meses e doze dias para 200 interações diárias com o usuário (consideravelmente acima das condições normais de uso). Além disso, o estagiário escreveu também, como parte das atividades do estágio, notas de aplicação (*Application Note)*, de título "Criando Aplicações *Bluetooth Low Energy* usando o QN9080" (*Creating Bluetooth® Low Energy Applications using QN9080*), com descrição do passo a passo das medidas a serem tomadas para obtenção resultados similares com outras aplicações embarcadas.

Palavras-chave: Sistemas embarcados, Baixo consumo, Bluetooth Low Energy, QN9080.

ABSTRACT

This report describes the activities that were performed during six months of internship at the Connectivity Division of NXP SEMICONDUCTORS FRANCE. The internship was aimed at developing a low-power C ++ firmware application for operating a Bluetooth-enabled, 220mAh battery-powered smart button with autonomy of five years, using the NXP QN9080-DK board and its respective Software Development Kit (SDK). By using the subsystems embedded on the NXP QN9080-DK, the SDK, the Bluetooth Low Energy (BLE) standard, measurements of electrical current consumption in different configurations, estimations of battery autonomy, and in accordance with the initial requirements, an application with estimated autonomy of five years, two months and twelve days was developed, considering 200 user interactions per day. In addition, an Application Note, entitled "Creating Bluetooth® Low Energy Applications using QN9080", was written as part of the internship activities, with step-by-step procedures to be taken by developers and NXP clients as a guide to reach similar results with their embedded applications.

Keywords: Embedded systems, Low power consumption, Bluetooth Low Energy, QN9080.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ADC	*Analog to Digital Converter*, Conversor Analógico Digital.
AHB	*Advanced High-performance Bus*, Barramento Avançado de Alta Performance.
AMBA	*Advanced Microcontroller Bus Architecture*, Arquitetura Avançada de Barramento de Microcontrolador.
CI	Circuito integrado.
DAC	*Digital to Analog Converter*, Conversor Digital Analógico.
DK	*Development Kit*, Kit de Desenvolvimento.
FPU	*Floating Point Unit*, Unidade de Ponto Flutuante.
FSP	*Fusion Signal Processor*, Processador de Sinal de Fusão.
GPIO	*General Purpose Input Output*, Portas Programáveis de Entrada e Saída.
HID	*Human Interface Device*, Dispositivo de Interface Humana.
I2C	*Inter-Integrated Circuit*, Protocolo Serial Para Dois Fios.
IoT	*Internet of Things*, Internet das Coisas.
MCU	*Microcontroller Unit,* Microcontrolador.
NFC	*Near Field Communication*, Protocolos de Comunicação de Campo Próximo.
PWM	*Pulse Width Modulation*, Modulação de Largura de Pulsa.
RAM	*Random Access Memory*, Memória de Acesso Aleatório.
R&D	*Research & Development*, Pesquisa & Desenvolvimento.
RF	Radiofrequência.
ROM	*Read-Only Memory*, Memória Somente de Leitura.
SDK	*Software Development Kit*, Kit de Desenvolvimento de Software.
S&A	*Systems & Applications*, Sistemas & Aplicações.
SPI	*Serial Peripheral Interface*, Interface Serial de Periféricos.
TI	*Texas Instruments*, empresa de semicondutores.
UART	*Universal Asynchronous Receiver Transmitter*, Transmissor Receptor Assíncrono Universal
USB	*Universal Serial Bus*, Barramento Serial Universal.
VANT	Veículo Aéreo Não Tripulado.
WLCSP	*Wafer Level Chip-Scale Package*.

SUMÁRIO

1 INTRODUÇÃO

Se houvesse uma enumeração das expressões mais comuns no contexto da engenharia eletrônica, a expressão *Lei de Moore* certamente seria listada entre as mais populares ou conhecidas. Sua origem está na segunda metade do século XX, no início da era dos circuitos integrados.

Em 1965, Gordon Moore, cofundador da Intel, publicou um artigo [1] na *Electronics Magazine* afirmando que a densidade de transistores em microprocessadores continuaria *a dobrar a cada ano* com mesmo custo e desempenho como vinha sendo observado desde 1959, corrigido em 1975 para *a cada dois anos*, o que passaria a ser chamado de *Lei de Moore*

Com base em dados os anos anteriores, ele percebeu que a quantidade de transistores por circuito integrado aumentava de forma consistente, a miniaturização da eletrônica estava em curso.

A previsão, porém, feita para os dez anos seguintes, tem perdurado até os dias atuais, quase 60 anos depois, e a miniaturização dos sistemas, simbolizada pela lei de Moore, continua impactando significantemente a economia e sociedades.

Algo que dificilmente apareceria na enumeração hipotética inicial seria uma expressão relacionada à *densidade energética das baterias*. Uma das razões para isso poderia ser o fato de que a densidade energética das baterias não cresceu nem mesmo próximo do ritmo de Moore, fato bastante dissonante da realidade da indústria de semicondutores.

Segundo Venkat [2] [3], ao passo que o número de sistemas eletrônicos integrados tem dobrado por área de circuito a cada dois anos, acompanhado por um aumento na necessidade de energia elétrica, a *densidade de energia de baterias* dobrou em um período de 150 anos, quando comparada com seus modelos do século XIX a partir da substituição de materiais, ficando muito atrás da *Lei de Moore* e, consequentemente, constituindo um gargalo energético.

Os impactos dessa limitação energética ficam mais claros se avaliados a qualidade dos serviços e o crescimento da indústria de sistemas eletrônicos. Nos setores de dispositivos móveis e médicos, monitoramento, automotivo, VANTs, aviões elétricos

etc., pode-se pontuar o aumento de custos de fabricação, de manutenção, de popularização e a diminuição da experiência do usuário em virtude da restrição energética.

Diante de tal situação, e possivelmente enquanto houver limitação energética, tem sido imprescindível para fabricantes e desenvolvedores de sistemas eletrônicos a busca e a adoção de medidas que aumentem a eficiência energética dos sistemas eletrônicos nos seus níveis de hardware, firmware e software, do projeto à aplicação.

1.1 OBJETIVOS DO ESTÁGIO

O estágio teve por objetivo central o desenvolvimento de uma aplicação "botão sem fios", dispositivo transmissor capaz de processar comandos físicos feitos pelo usuário e enviá-los para um receptor sem fios, centrada em consumo de potência e no perfil de corrente elétrica, que fosse usada em um contexto de automação residencial, como um interruptor sem fios, de autonomia mínima de cinco anos, assim como a escrita de notas de aplicação documentando tal desenvolvimento. O estudante deveria se ocupar em adaptar uma estrutura de software de um exemplo existente a um novo microcontrolador (MCU) NXP QN9080. Os seguintes pontos detalham as atividades principais que possibilitaram alcançar o objetivo:

- Compreender as capacidades do MCU NXP QN9080, do Kit de Desenvolvimento (NXP QN9080-DK) e do SDK;
- Compreender e definir as especificações para a solução de automação residencial requerida, botão sem fios conectado via BLE;
- Compreender as especificações da tecnologia *Bluetooth Low Energy*;
- Desenvolver um firmware em C++ a ser embarcado no chip QN9080, para solução de automação residencial com autonomia maior que cinco anos;
- Estimar autonomia, medir e validar consumo de energia da solução proposta
- Aprimorar o consumo energético da mesma solução;
- Documentar, em notas de aplicação, as decisões arquiteturais com impacto direto na eficiência energética.

O estágio profissional é uma disciplina obrigatória na estrutura curricular do curso de Engenharia Elétrica da UFCG, sendo um requisito à conclusão do curso. Isso posto,

este relatório tem como objetivo apresentar os conhecimentos adquiridos e relatar as principais atividades realizadas durante o estágio integrado realizado na empresa NXP SEMICONDUCTORS FRANCE, no período compreendido entre 03 de fevereiro de 2017 e 03 de agosto de 2018, totalizando 868 horas, sob a orientação técnica do engenheiro Marco Merlin.

1.2 NXP SEMICONDUCTORS

NXP Semiconductors é uma empresa holandesa líder mundial em soluções eletrônicas, *mixed signal*, de alta performance que, combinadas com um profundo foco em aplicações, tem por slogan a frase *conexões seguras e necessárias para um mundo mais inteligente*.

Operando em mais de 33 países e com mais de 60 anos de experiência, a *NXP Semiconductors* se mantém em primeiro lugar na indústria de identificação eletrônica (cartões bancários, *E-Government*, *Mobile NFC*, cartões de transporte público e de acesso), na indústria automotiva, na produção de transistores de potência para RF, entre outros setores.

Dentre os maiores clientes estão *Apple*, *Bosch*, *Continental*, *Ericsson*, *Gemalto*, *Giesecke and Devrient*, *Huawei*, *Hyundai*, *Kona*, *Nokia Networks*, *Panasonic*, *Samsung* e *ZTE*.

Fig. 1. Localização da sede onde o estágio foi desenvolvido.

Fonte: [11].

Fig. 2. Fachada do local de estágio.

Fonte: [11].

O estágio foi desenvolvido na sede *NXP Sophia-Antipolis*, uma das quatro sedes na França. Trata-se de uma sede de pesquisa e desenvolvimento (R&D) que agrupa várias

atividades de engenharia no domínio de alta tecnologia de semicondutores, como por exemplo:

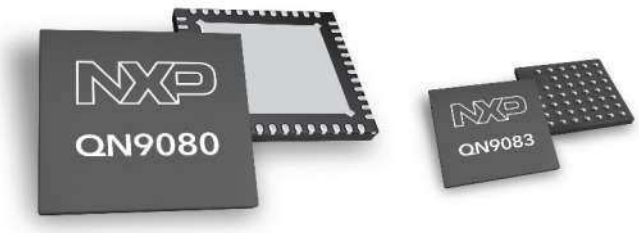
- Microcontroladores, com a concepção de processadores para aplicações embarcadas;
- Soluções de segurança da informação, com a concepção de circuitos integrados *NFC* ou *Secure Element*;
- Soluções de conectividade, com a concepção de circuitos integrados Bluetooth e WiFi;
- Concepção de circuitos integrados de processamento de áudio para o mercado de dispositivos móveis;
- Desenvolvimento de algoritmos de processamento embarcado de áudio;
- Desenvolvimento e manutenção de software embarcado para microcontroladores.

Em atividade desde 2006, com mais de uma centena de engenheiros compondo as equipes *Microcontrollers*, *Secure Mobile Terminal* e *Secure Monitoring & Control*. As equipes criam soluções completas e seguras nos domínios de internet das coisas, automotivo e do grande público.

2 EMBASAMENTO TEÓRICO

2.1 NXP QN9080 E O KIT DE DESENVOLVIMENTO (DK)

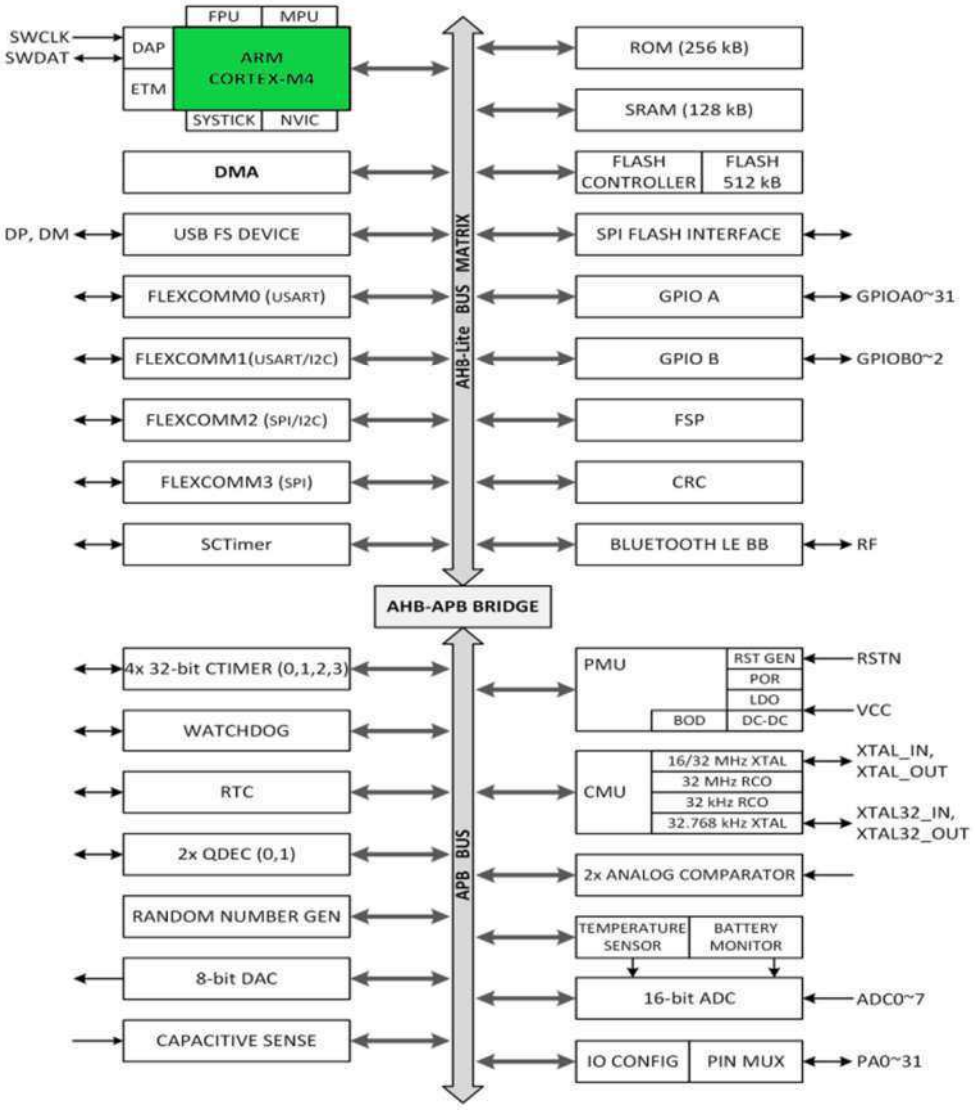
Fig 3. System-on-Chip NXP QN9080 nas diferentes versões.



Fonte: [7].

Com foco no mercado de vestimentas inteligentes, automóveis, eletroeletrônicos, monitores de saúde e de atividades, na automação residencial, inúmeros dispositivos de interface máquina-homem (HID), entre outros, a *NXP Semiconductors* desenvolveu o QN9080. Com dimensões de 3.28mm x 3.20mm na versão WLCSP, o QN9080 trata-se de um MCU de baixo consumo de energia, com processador Cortex-M4F de 32-bits da ARM, e vários periféricos como transceptor RF (Bluetooth Low Energy), memórias (ROM, Flash de 512KB, RAM de 128 KB e porta quad SPI), interfaces seriais (USART, I2C, SPI, USB 2.0), até 35 GPIO, 8 ADCs de 16-bit e DACs de 8-bits, timers, PWM, decodificadores de quadratura, unidades de gerenciamento de clock (osciladores internos e externos) e de potência, sensor de temperatura, entre outros. A comunicação entre todos os subsistemas é feita pelo padrão ARM *Advanced Microcontroller Bus Architecture* (AMBA) através de multi-camadas do *Advanced High-performance Bus* (AHB). O núcleo de processamento conta ainda com uma unidade adicional de processamento de sinal (FSP).

Fig. 4. Diagrama de blocos do SoC NXP QN9080.



Fonte: [6].

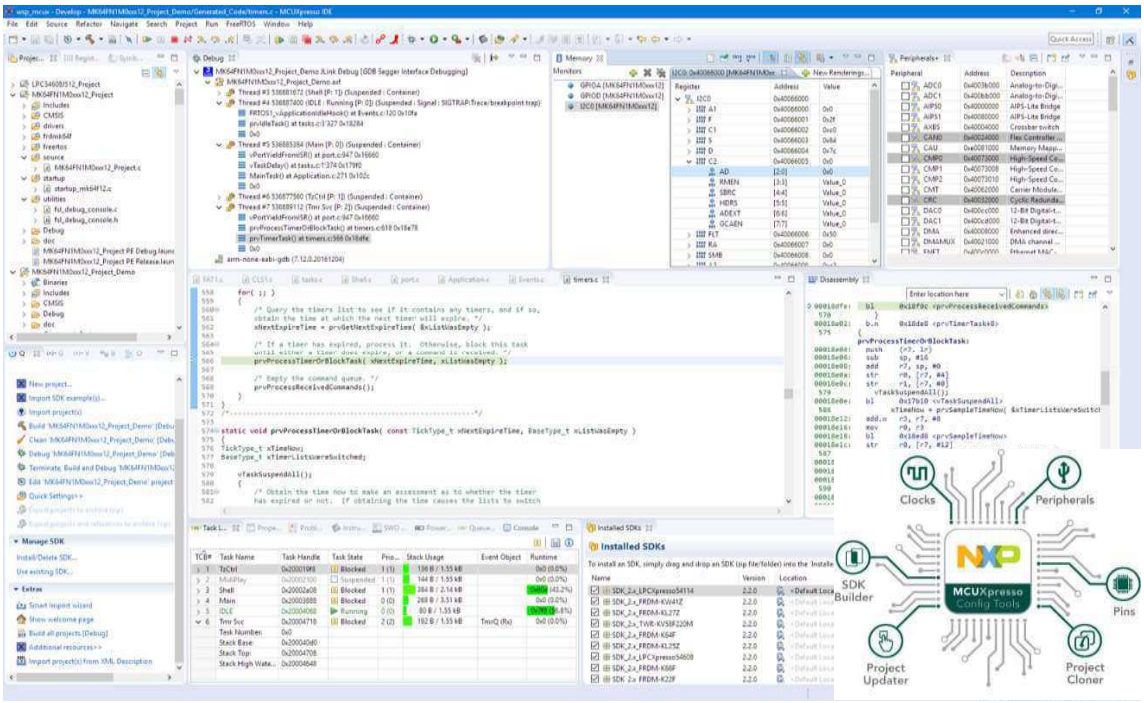
Atualmente os sistemas em chips eletrônicos são complexos, ocupando menos espaço, consequentemente a configuração desses passou também a ser mais extensa e exigir mais tempo dos desenvolvedores de produtos. Para tornar mais simples o uso de seus chips, grandes fabricantes de semicondutores se tornaram também fabricantes de software, fornecendo projetos genéricos e consultoria.

Dentre as soluções que são hoje oferecidas estão software para programação e depuração (*debugging)* (IDE), kits de desenvolvimento software (SDK) contendo projetos de uso com demonstrações de uso de periféricos, assim como kits de desenvolvimento dos microcontroladores (DK). Tais soluções foram úteis no

desenvolvimento das atividades do estágio, em que se utilizou o *MCUXpresso IDE*, o *QN9080-SDK* e o *QN9080-DK*.

O *MCUXpresso IDE* é um software baseado em Eclipse e GCC para editar, compilar e depurar (*debugging*) projetos em microcontroladores, permite também configurar de forma simples pinos, clocks e periféricos.

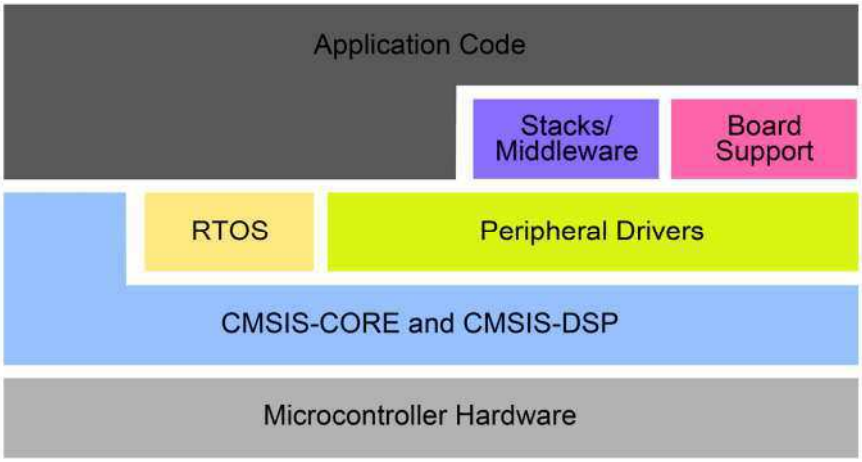
Fig. 5. Interface da IDE MCUXpresso.



Fonte: Autor.

O Kit de Desenvolvimento de Software (SDK) é um conjunto de software disponibilizado gratuitamente pela *NXP Semiconductors*, para simplificar e acelerar o desenvolvimento de aplicações com as variadas famílias de microcontroladores e processadores. No SDK é possível obter software de execução *BareMetal* e RTOS (*Real-Time Operating Systems*, do inglês, Sistema Operacional em Tempo Real), stacks e middleware integrados, software de referência, drivers e exemplos de uso de periféricos específicos etc.

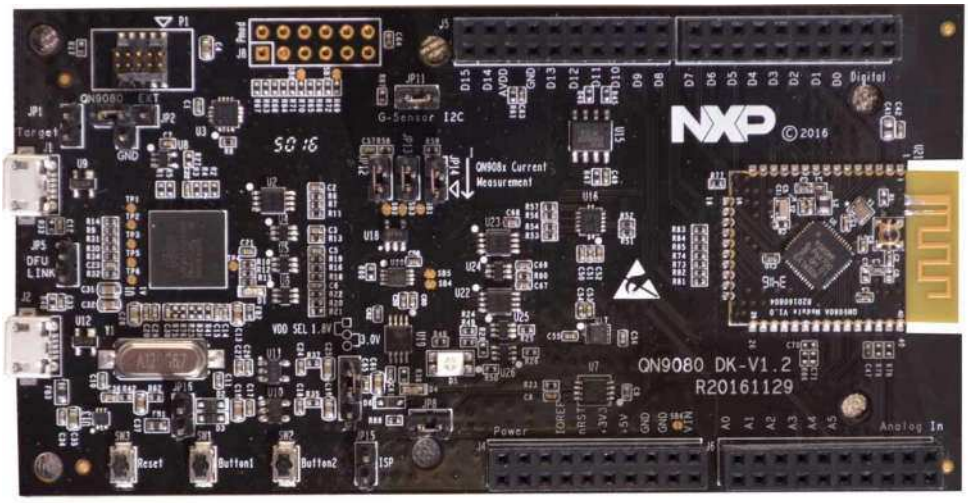
Fig. 6. Diagrama de blocos do SDK do MCUXpresso.



Fonte: [8].

O Kit de Desenvolvimento *QN9080-DK*, também usado no estágio, é uma placa concebida para a avaliação das funções e performance do CI QN9080, assim como o desenvolvimento de aplicações.

Fig. 7. Foto da placa de desenvolvimento NXP QN9080-DK vista de cima.



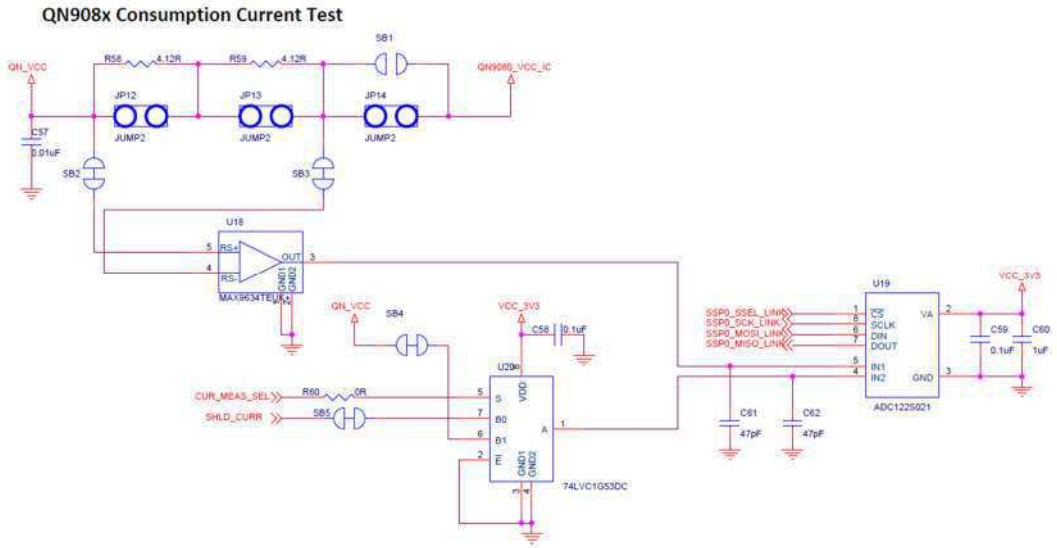
Fonte: [9].

No lado esquerdo da placa, uma das porta USB fornece a energia para todos os CIs e se conecta ao microntrolador LPC4322, que funcionará como a ponte entre o computador e as portas JTAG/SWD do QN9080 para programação e depuração. A outra conecta-se diretamente às portas JTAG/SWD do CI, caso o desenvolvedor deseje utilizar um depurador externo.

Outras características de valor no desenvolvimento do estágio são a interface Arduino, em que estão disponíveis os todos os GPIOs, botões e LEDs, e principalmente

o módulo de medição de consumo de corrente elétrica pelo CI QN9080, módulo este integrado à placa, mostrado abaixo.

Fig. 8. Recorte do diagrama do circuito integrado presente na NXP QN9080-DK.



Fonte:[9].

O módulo de medição de corrente basea-se na tensão entre os terminais de resistores de precisão, dois de 4.12 Ω, que podem ser colocados em série a partir da configuração de jumpers. O CI MAX9634T mede a corrente e amplifica com um ganho de 25, o ADC122S021 fará a conversão de analógico para digital, com precisão de 12-bits e frequência de amostragem de 50 a 200 ksps. O processamento final é feito pelo LPC4322 e enviado para o computador na interface do MCUXpresso IDE. Esse procedimento foi executado inúmeras vezes como verificação para modificações nos projetos e validação de resultados anteriores.

Outra maneira de medição da corrente consumida pelo QN9080 é a partir de um amperímetro de precisão, ou analisador de potência, conectado aos jumpers com os resistores em aberto, sendo em série com a alimentação. Esse método foi usado poucas vezes por conta da escassa disponibilidade do amperímetro, prioritariamente próximo ao fim do estágio para validação final dos valores obtidos.

2.2 BLUETOOTH LOW ENERGY

Tecnologias como Bluetooth Low Energy, Zigbee, ANT, são similares em atender às especificações mínimas na era dos acessórios conectados, comunicação de curta distância, baixa taxa de dados, longa autonomia de baterias (escala de meses e anos) e autonomia da aplicação. Elas, porém, diferem quanto ao nicho de aplicações. Cabe aos desenvolvedores, de acordo com as especificações e requisitos de seus projetos, decidir a arquitetura a ser usada com protocolos, parâmetros, modos de conexão etc. O Bluetooth Low Energy (BLE) permite a conectividade sem fio do MCU QN9080, sendo central no objetivo do estágio, como destacado em 2.1.

2.2.1 LINHA DO TEMPO DO BLUETOOTH E O BLUETOOTH LOW ENERGY

Muitas denominações surgem quando se trata de Bluetooth, e.g. Low Energy, Classic, Smart, Basic Rate (BR), Enhanced Data Rate (EDR), Smart Ready, Single-Mode, Dual-Mode. A maneira mais clara de compreender todas elas é por um resumo temporal de seus desenvolvimentos.

Em 1994, [3] Ericsson inventa um protocolo de comunicação sem fios para substituir a comunicação por cabos que utilizavam o padrão RS-232 para conectar periféricos.

Em 1998, o grupo Bluetooth Special Interest Group (SIG) é formado, inicialmente por cinco grandes fabricantes de eletrônicos, Ericsson, IBM, Intel, Nokia e Toshiba. No ano seguinte, o grupo lança a primeira versão de especificações para a tecnologia Bluetooth, já contando com mais de 4000 empresas membro.

Em 2000 são lançados no mercado o primeiro celular com conexão Bluetooth, assim como os primeiros headset, protótipo de mouse, placa para computadores etc.

Em 2001, a Nokia começa a pesquisar alternativas baseadas em Bluetooth para aplicações de baixo consumo, baixa complexidade e de baixo custo, chamada pelo nome de *Bluetooth Low End Extension*. A ideia da Nokia era de aproveitar o mercado de dispositivos móveis que já utilizavam Bluetooth e adicionar a extensão, ou uma configuração de baixo consumo, via atualização de software, o que acabou não se concretizando

Em 2004, o *SIG* lança a versão das especificações, Bluetooth 2.0 + EDR. Assim, dando origem ao *Enhanced Data Rate* (EDR), atualização que permitiu um aumento de

fluxo de dados de ~750Kbps para 2.1Mbps, mantendo a compatibilidade com a geração anterior, chamada de Basic Rate (BR). Nos anos seguintes, várias versões atualizadas do protocolo, (BR/EDR) foram publicadas pelo SIG, em termos latência, segurança etc.

Em 2006, a Nokia publica, com o nome de Wibree, a tecnologia que havia iniciado a desenvolver cinco anos antes. As negociações para adicioná-la ao padrão Bluetooth se iniciam e serão efetivadas em 2009.

Em 2010, a versão Bluetooth 4.0 é publicada com atualizações do Bluetooth BR/EDR e, finalmente, com inclusão do novo Bluetooth Low Energy, também chamado de Bluetooth Smart, com velocidade de transfêrencia de dados de aplicação de até 250Kbps. Apesar dos mesmos nomes, o desenvolvimento do Wibree, ou Bluetooth Low Energy, seguiu direções muito opostas ao Bluetooth Classic, que por fim os incompatibilizou a nível de rádio, como pode ser visto na tabela abaixo.

Visto que dispositivos BLE e Classic não são capazes de se comunicarem, alguns fabricantes implementam ambas especificações, sendo chamados de Dual-Mode, e.g. aparelhos celulares. Outros, que possuem apenas a implementação do BLE, são chamados de Single-Mode e se comunicarão apenas com outros dispositivos BLE.

Fig. 9. Gráfico representativo da interoperabilidade de diferentes versões da tecnologia Bluetooth.

Fonte: http://www.summitdata.com/blog/bluetooth-smart-bluetooth-smart-ready/

Tabela 1: Comparativo entre as versões Classic e Low Energy da tecnologia Bluetooth.

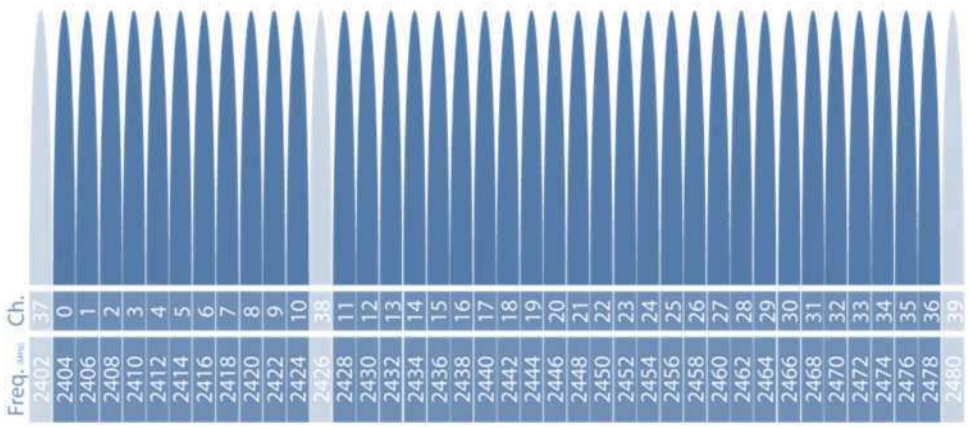
	Bluetooth Classic (BR/EDR)	**Bluetooth Low Energy (LE)**
Banda de frequência	2.4 GHz ISM Band (2.402 – 2.480 GHz)	2.4 GHz ISM Band (2.402 – 2.480 GHz)
Canais	79 canais separados por 1MHz	40 canais separados por 2 MHz (3 para advertising, 37 para dados)
Uso dos canais	Frequency Hopping Spread Spectrum	Frequency Hopping Spread Spectrum
Potência consumida	1 (como referência)	~0.01 a 0.5 a depender do caso de uso
Modulação	GFSK, π/4 DQPSK, 8 DPSK	GFSK
Fluxo de dados	EDR PHY (8DPSK): 3 Mb/s EDR PHY (π/4DQPSK): 2Mb/s BR PHY (GFSK): 1 Mb/s	LE 2M PHY: 2 Mb/s LE 1M PHY: 1 Mb/s LE CodedPHY (S=2): 500 Kb/s LE CodedPHY (S=8): 125 Kb/s
Topologias	Ponto-a-ponto	Ponto-a-ponto Broadcast Mesh

Fonte: [10].

2.2.2 CANAIS DO BLE: *ADVERTISING* E *DATA CHANNELS*

A tecnologia BLE permite a transmissão de dados de duas formas: através de uma conexão, ou mesmo sem conexões (similar à transmissão de sinais de televisão, chamado de *Broadcasting*).

Fig. 10. Representação dos canais e divisão destes entre *Advertising channels* (verde) e *Data channels* (vermelho) na tecnologia Bluetooth Low Energy.



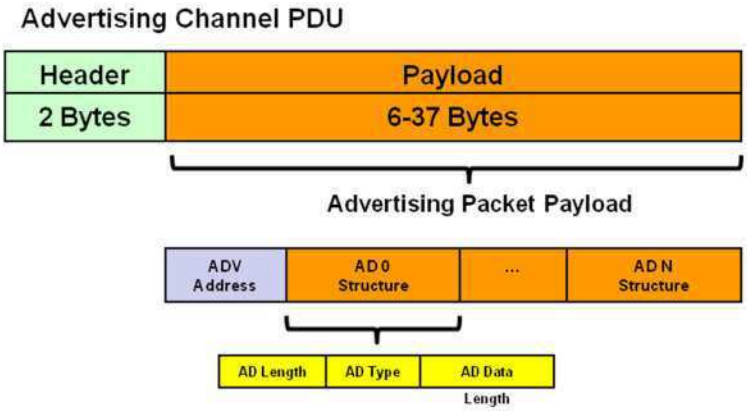
Fonte: [13]

Para estabelecer uma conexão, um dispositivo utilizará os canais de *advertising* para transmitir suas informações, como identificação, nome local, potência de transmissão, UUID dos serviços oferecidos etc. Outro dispositivo que esteja observando

estes canais receberá as informações e responderá com um pedido de conexão. Apenas após confirmada a conexão, os dois dispositivos passarão a usar os canais de dados para comunicação, de acordo com a sequência de canais estabelecida e conhecida apenas pelos dois, esquema adaptativo do *Frequency Hopping Spread Spectrum* (FHSS).

Para o caso de *broadcasting,* i.e., para enviar dados sem conexão, o primeiro dispositivo definirá o *header* do *Protocol Data Unit* (PDU) como não-conectável não direcionado, o que impedirá recebimento de pedidos de conexão. Esse tipo de comunicação é comum para dispositivos do tipo *Beacon*, usado em localizadores ou medidores de proximidade, que transmitem continuadamente informações sobe o próprio nó.

Fig. 11. Representação da divisão de bytes no *PDU* para o canal de *Advertising*.



Fonte: [11].

Fig. 12. Descrição dos tipos de *PDU* disponíveis para dispositivos BLE.

```
%  PDU Types
%  b3b2b1b0 Packet Name
%  0000 ADV_IND connectable undirected advertising event
%  0001 ADV_DIRECT_IND connectable directed advertising event
%  0010 ADV_NONCONN_IND non-connectable undirected advertising event
%  0011 SCAN_REQ scan request
%  0100 SCAN_RSP scan response
%  0101 CONNECT_REQ connection request
%  0110 ADV_SCAN_IND scannable undirected advertising event
%  0111-1111 Reserved
```

Fonte: [5].

O BLE é definido por meio de *Profiles* (Perfis). Os *Profiles* são um ou mais serviços que oferecem suas respectivas definições de funcionamento no *stack*. O *Generic Access Profile*, conforme o nome indica, é o responsável pelo modo de acesso ou *air*

interface, entre os dispositivos que se comunicarão. Ele estabelece dois modos, como já visto, e *roles* (ou papéis) referentes aos modos:

Tabela 2: Divisão dos papéis nos modos *Beacon* e *Non-Beacon*.

Modo	Beacon:	Non-Beacon:
***Roles* (Papéis)**	*Broadcaster*	*Peripheral*
	Observer	*Central*

O Generic Atribute Profile (GATT), explicado nas próximas páginas, e o GAP são dois perfis considerados mandatórios pela especificação SIG Bluetooth Low Energy. Eles compõem a base de funcionamento de qualquer dispositivo compatível com BLE, definindo *air interface*, dados e transações.

2.2.3 ARQUITETURA

A arquitetura Bluetooth é dividida em três camadas principais, *Application*, *Host* e *Controller*, e cada uma delas pode ser implementada em um CI diferente. No caso do QN9080, todas as camadas estão no mesmo CI:

Tabela 3: Descrição de cada uma das camadas do Bluetooth Low Energy.

Application	A mais alta camada de todo o sistema, responsável pela: Lógica, Interface com usuário. Botões, sensores etc Fluxo de dados do sistema Interface com o *stack* do protocolo Bluetooth.
Host	A camada mais alta do *stack* do protocolo Bluetooth que inclui: *Generic Access Profile* (GAP) *Atribute Protocol (ATT)* *Generic Atribute Profile (GATT)* *Logical Link Control and Adaptation Protocol (L2CAP)* *Security Manager (SM)* *Host Controller Interface (HCI)*
Controller	A camada mais baixa do *stack* do protocolo Bluetooth que inclui: Host Controller Interface (HCI) Link Layer (LL) Physical Layer (PHY)

Fonte: [4].

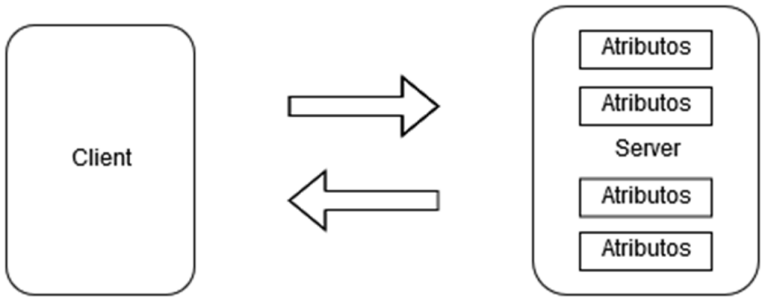
As especificações *Bluetooth* [5] também apresentam um protocolo específico para comunicação entre *Host* e *Controller* chamado *Host Controller Interface* (HCI). O HCI permite a interoperabilidade entre as camadas do *stack* Bluetooth produzidas por diferentes fabricantes.

Um dos pilares da tecnologia Bluetooth é a interoperabilidade de dispositivos de diferentes fabricantes. Por meio dos *Profiles*, o *SIG* estabelece serviços, regras e protocolos que devem ser seguidos pelos fabricantes, esses *Profiles* estão no *Host*. O *ATT*, *GATT* e *GAP* são elementos mandatórios para que um dispositivo seja considerado compatível com Bluetooth Low Energy, eles proverão a estrutura e definições base relativas aos dados que serão trocados e à configuração da conexão, respectivamente.

2.2.4 *ATRIBUTE PROTOCOL* (ATT) E *GENERIC ATRIBUTE PROTOCOL* (*GATT*)

O *Atribute Protocol,* ou ATT, é um protocolo que estabelece a organização de quaisquer informações (que serão armazenadas, acessadas, enviadas, recebidas etc.) como *atributos* no *stack* Bluetooth. De acordo com [4], atributos são a menor entidade de dados definida pelo ATT, logo, o *Client* não acessará ou modificará unidades de bits e bytes do *Server*, fará isso por meio dos atributos do Server.

Fig. 13. Diagrama ilustrativo da relação *Client-Server*.



Fonte: Autor.

O ATT é um protocolo baseado no modelo *Client* e *Server*, portanto assim também é a Tecnologia Bluetooth. De acordo com BLE, cada dispositivo será ou *Client*, ou *Server*, ou ainda os dois. O *Server* oferecerá suas funções e serviços e o *Client* iniciará o pedido para fazer uso de tais funções e serviços, chamados atributos. Na prática, um celular (*Client*) conectado a um sensor IoT (*Server*) fará o pedido para saber qual a temperatura (atributo do Server) no momento. O *Server* então enviará a resposta do pedido iniciado pelo *Client*.

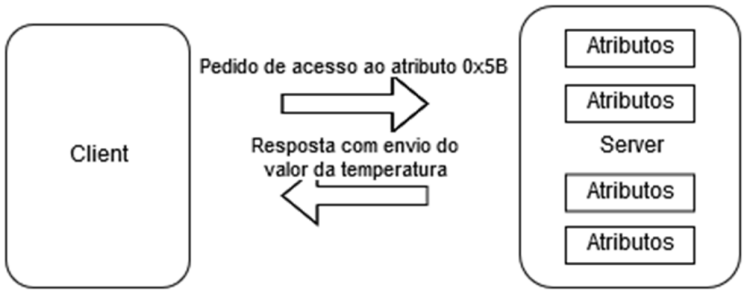
Cada *Server* terá dados organizados em atributos, e cada um de seus atributos conterá informações sobre si (um *atribute handle* de 16-bit, um identificador universalmente único (UUID), um conjunto de permissões de acesso daquele atributo) e os dados importantes ao usuário (valor do atributo, o dado em si).

Tabela 4: Descrição da estrutura de um atributo (*Attribute*).

Componentes de um atributo		Exemplo
Handle	Identificador único (a nível da memória do *Server*) de 16-bits. O acesso aos atributos do *Server* se torna mais eficiente com o uso desses pequenos identificadores, comparados com os UUID.	0x050B
UUID	Identificador Universalmente Único que descreverá o tipo do atributo, pode ter 16, 32 ou 128 bits, e assim o tipo de dado que aquele atributo contém.	0x1234 De acordo com as especificações SIG, refere-se a *Temperature*, em graus Celsius
Permissões	Metadados que especificam as operações que podem ser executadas e nível de segurança que determinado atributo possui.	Read only, no security
Valor	Os dados reais que são desejados pelo usuário, limitado a 512 bytes, normalmente definido pela especificação SIG.	34.53

A informação sobre os atributos oferecidos por um *Server* é requerida pelo *Client*, *Server Discovery*, no início de uma conexão e armazenada por ele, o que permite a este acessar cada um dos atributos pelo *handle* e não necessariamente pelo UUID, enviando menos informação a cada transmissão de pedido. É possível ao *Client*, caso o Server seja estático, armazenar definitivamente as informações referentes aos atributos do *Server*, para que em uma conexão futura não seja necessária a re-execução do *Server Discovery*, esse processo é chamado de *Attribute Caching*.

Fig. 14. Diagrama ilustrativo de uma interação entre *Client e Server*.



Fonte: Autor.

Podemos resumir as operações principais de troca de dados entre *Client* e *Server* em quatro, conforme a tabela abaixo. Vale lembrar que as operações sobre os atributos serão efetuadas somente se estiver de acordo com as permissões do atributo em questão.

Tabela 5: Principais funções desempenhadas por cada um dos papéis.

Client: acessa os atributos (dados) do Server	*Write*: *Client* envia o comando de escrita de atributo.
	Read: *Client* pede para ler um atributo, *Server* responde com o valor.
Server: expõe seus atributos ao Client	*Indication*: O *Server* possui dados e deseja atualizá-los ao *Client*, enviando uma indicação do atributo. O Client responderá com uma confirmação.
	Notification: O mesmo que *Indication*, sem a confirmação.

A hierarquia dos dados também é definida pelo *GATT*, conforme mostrado pelo diagrama abaixo. É importante destacar que um mesmo dispositivo pode ter vários *roles*, ou papéis, diferentes ao mesmo tempo. Isto porque diferentes camadas do *stack Bluetooth* estabelecerão seus diferentes *roles*.

Tabela 6: Resumo de papéis assumidos por um dispositivo BLE em cada camada.

Camada Host: GATT	*Server*: Dispõe de dados e serviços e os expõe ao *Client* *Client*: Acessa e utiliza os dados e serviços do *Server*
Camada Host: GAP	*Beacon*: *Broadcaster*: Transmite por *Advertisements* *Observer*: Recebe as informações pelos canais de *advertising* *Non-Beacon*: *Central*: Inicia a conexão *Peripheral*: Aceita a conexão
Camada Controller: Link Layer	*Master*: Entra em uma conexão pelo *Initiating State* *Slave*: Entra em uma conexão a partir do *Advertising State*

Fonte: [3].

2.3 REQUISITOS DA SOLUÇÃO DE AUTOMAÇÃO RESIDENCIAL

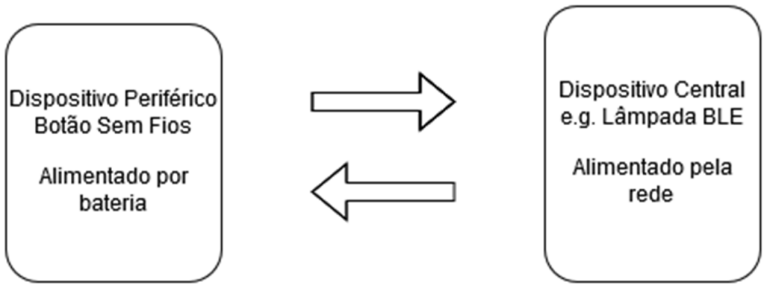
A proposta do estágio foi o desenvolvimento de uma solução eletrônica de automação residencial. O objetivo final era a validação das capacidades, em especial o baixo consumo de energia, do MCU QN9080.

A solução proposta trata-se de um botão conectado:

- Baseado no MCU QN9080
- Conectado via *Bluetooth Low Energy* com um dispositivo central
- Alimentado por uma bateria de 3V, 220mAh

- Autonomia de mais de cinco anos com uma bateria
- Interface com o usuário de quatro botões
- Latência deve ser menor que 100ms para boa experiência do usuário

Fig.15. Diagrama ilustrativo da arquitetura da aplicação, com seus atores e funções.



Fonte: Autor.

Apesar da simplicidade, essa mesma aplicação foi vendida pela NXP Semiconductors em 2016 a *British Gas*, que era o segundo maior consumidor de tecnologia *ZigBee* no Reino Unido, utilizando, porém, a tecnologia *ZigBee* e outro MCU visto na figura abaixo.

Fig. 16. Imagem ilustrativa da aplicação descrita acima.

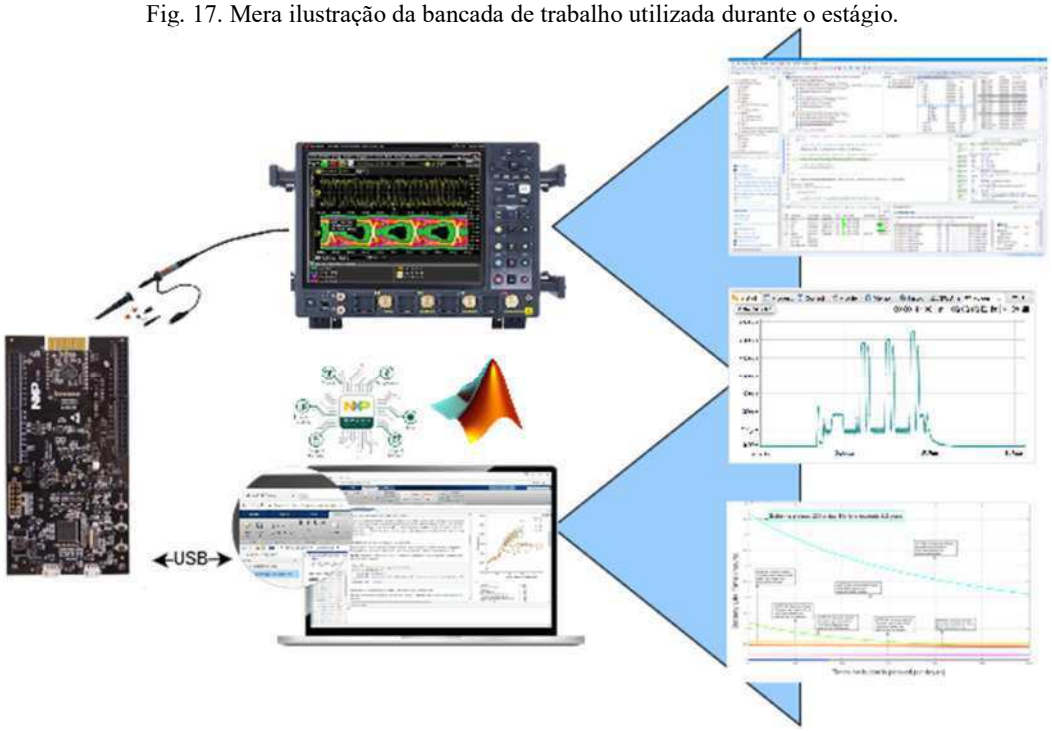
Fonte: Autor.

2.4 MEDIÇÃO E ESTIMAÇÃO DE AUTONOMIA

Um dos grandes desafios do estágio foi obter uma aplicação que tivesse autonomia estimada superior a cinco anos, com a carga disponibilizada por uma bateria de 220mAh. Para ir de encontro a esse desafio, foi necessário medir o consumo do SoC sob diversas configurações e estimar o tempo de autonomia. Como foi mostrado na seção 2.1, o Kit de Desenvolvimento QN9080-DK possui um circuito integrado para a medição da corrente elétrica que é consumida pelo MCU QN9080. Esse circuito foi essencial para o bom andamento do estágio, pois facilitou e simplificou a validação das medidas adotadas para a redução do consumo médio de energia, e consequentemente aumento da autonomia da

bateria da futura aplicação. Os valores em tempo real eram obtidos a partir da aba *Power Measurement Tools* da IDE *MCUXpresso*, e em seguida salvos no formato .CSV, para posterior manipulação no MATLAB, no qual se obtinha o valor médio de consumo de corrente elétrica e da autonomia da bateria ou autonomia da aplicação.

Fig. 17. Mera ilustração da bancada de trabalho utilizada durante o estágio.



Fonte: Autor.

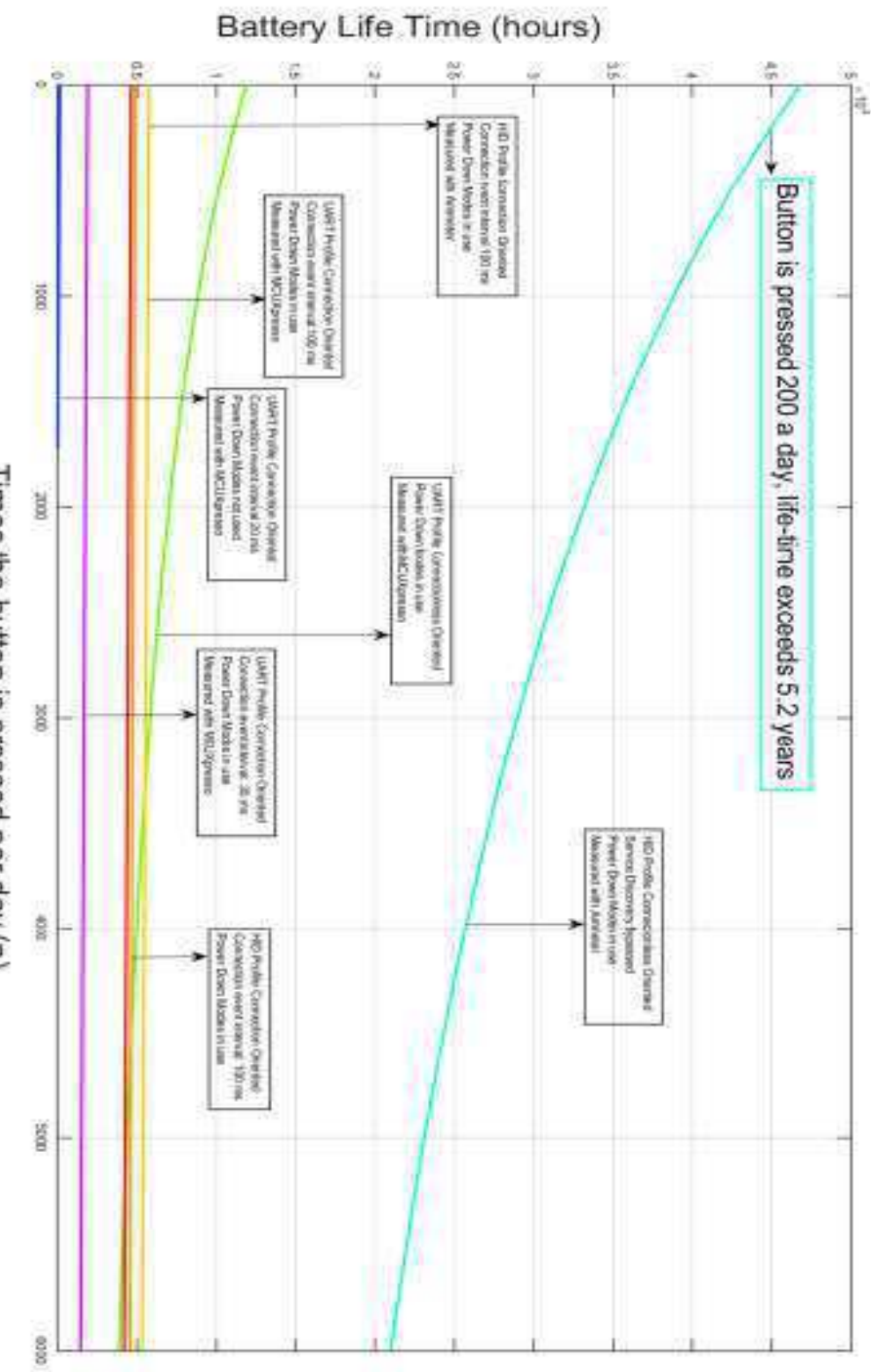
Dados os requisitos da aplicação botão conectado, (em 2.3 acima) foi considerado "o número de interações do usuário por dia" como a variável independente e "a autonomia da bateria" como a variável dependente.

Para obtenção do consumo médio de corrente para *n* interações do usuário por dia, foi considerado o valor médio de corrente consumida pelo QN9080 quando um dos botões da aplicação era apertado, assim também como o valor de corrente quando o MCU não era apertado, ou seja, quando estava em *stand-by*. Por fim, a partir do consumo médio de corrente para *n* interações, dividia-se a capacidade da bateria por esse valor, encontrando-se a estimativa de tempo de autonomia da bateria. As estimativas foram feitas com o número de interações diárias, *n*, de 0 a 6000 vezes.

Com o auxílio do MATLAB, foi possível plotar análises como a da figura abaixo, que relaciona as modificações feitas a nível de FW com a estimativa final da autonomia da bateria em função do número de interações do usuário. Apenas para validar o objetivo,

foi definido que o usuário utilizaria a aplicação, em média, duzentas vezes, 200, por dia, considerado como muito além do caso de uso.

Na imagem abaixo é possível ver a evolução da autonomia a partir de cada uma das modificações feitas que serão descritas na próxima seção. Partindo do projeto disponibilizado pelo *SDK*, que apontava para uma autonomia de pouco mais de três semanas à bateria, até o projeto final que permitiu uma autonomia de cinco anos, dois meses e doze dias para uma média de 200 interações diárias do usuário.



Fonte: Autor.

Fig. 18. Comparativo da autonomia estimada relacionado com diferentes confuguraçõe de funcionamento da aplicação.

3 ATIVIDADES DESENVOLVIDAS

3.1 DEFINIÇÕES DO PLANO DE DESENVOLVIMENTO

A fim de estabelecer um mapa para as atividades que seriam executadas no estágio, o estudante enumerou os seguintes pontos e apresentou em seguida ao tutor do estágio, assim como a chefes de equipes, para confirmação. As atividades previstas foram:

1) Analisar a documentação da aplicação "botão conectado" vendida à *British Gas*, desenvolvida para outro MCU e utilizando a tecnologia *ZigBee*, a fim de assegurar-se dos requisitos da solução do estágio. A análise foi baseada em documentos internos da empresa, requisitos e especificações, relatórios do desenvolvimento, relatórios de testes e validações.
2) Em seguida, definir e documentar o escopo, i.e., requisitos e especificações da aplicação "botão conectado" objetivado pelo estágio, com base no projeto da *British Gas*. Acelerar o desenvolvimento a partir do uso do *SDK*.
3) Familiarizar-se com o *testbench* descrito em 2.4. a partir de validação do consumo de energia documentado do *SDK*.
4) Implementar as medidas de economia de energia já conhecidas e utilizadas pela equipe de Sistemas e Aplicações (S&A), *Power Down Modes* para o MCU, validar e documentar os resultados quanto ao consumo de energia.
5) Pesquisar e implementar medidas de redução do consumo de energia em sistemas embarcados com BLE.
6) Escrever notas de aplicação para futura publicação.

Após reunião com tutor, chefe de equipe R&D RF e chefe de equipe S&A, que se deu na segunda semana, os pontos enumerados foram aprovados e o estagiário iniciou as atividades conforme previsto.

3.2 MEDIÇÕES E ESTIMAÇÕES INICIAIS

Durante as primeiras semanas, foi designado ao estagiário que se familiariza-se com a placa QN9080-DK, com o SDK e com os equipamentos de laboratório disponíveis (que eram o notebook com *MCUXpresso IDE* capaz de fazer a aquisição da corrente instantânea consumida pelo MCU e um osciloscópio).

A primeira atividade feita foi a caracterização do consumo do MCU em diferentes estados, e.g. *Scanning, Advertising, Stand-by, Low Power* etc, através do perfil *BLE Power Profiles.* A partir da modificação de parâmetros e rotinas do código e visualização do impacto nas formas de onda de corrente instantânea, o estagiário pôde compreender melhor as rotinas que eram efetuadas e a implementação da arquitetura BLE no firmware provido pela NXP. Foi possível validar alguns relatórios e *Application Notes*, encontrar valores errados e encontrar as justificativas para os erros.

Em seguida, o estagiário escolheu e apresentou ao tutor o perfil *UART_Wireless*, como perfil do SDK com o qual trabalharia no projeto, justificando a escolha pela versatilidade do perfil. Utilizando este perfil, algumas comparações e estimativas foram feitas: comparação de consumo entre dispositivos *Master* e *Slave*, na definição de parâmetros, desligamento de periféricos do MCU etc.

Poucas semanas depois o estagiário sugeriu a troca do perfil *UART_Wireless* pelo H.I.D., tanto pela especificidade compatível com a aplicação, como pela implementação deste perfil no SDK ser menor e mais eficiente.

Por fim, definido o perfil, foi dado continuidade às atividades previstas, descritas na seção anterior.

3.3 REDUÇÃO DO CONSUMO DE ENERGIA A PARTIR DO MCU

Dispositivos de interface humana (HID), *wearables*, assim como tantos outros, acessórios de aparelhos celulares, podem demandar alta performance de processamento e ao mesmo tempo eficiência no gerenciamento de potência. Ambos requisitos são atendidos pelo núcleo do MCU QN9080, o core ARM Cortex-M4.

O processador dispõe de uma Unidade de Gerenciamento de Potência (PMU) que integra modos *Sleep* e *Power-Down* controlados por *software*, gerenciamento de clocks e

retenção opcional de estado. De forma simples, os modos de baixo consumo, quando tem suas funções chamadas, desativam determinados blocos internos e periféricos do sistema, que consequentemente passará a consumir menos energia. Essa capacidade é perfeitamente compatível com aplicações cíclicas, ou seja, que não estão continuadamente ativas, mas passam grandes intervalos em *stand-by*.

Tabela 7: Descrição dos modos de operação do SoC.

Modo	
Ativo	Processador está em funcionamento. É possível otimizar o consumo configurando clocks e periféricos.
Sleep	O clock do CPU é parado e a execução de instruções é suspensa até a atividade de alguma fonte de *Wake-up*. O estado do processador, SRAM interna e registradores (incluindo dos periféricos) são mantidos.
Power-Down 0 e 1	0: O fornecimento de potência é encerrado para todo o sistema, com exceção da Unidade Gerenciamento de Potência (sempre ativa), 32k XTAL, Oscilador RC 32k, *Sleep Timer*, RTC. Todos os blocos analógicos são desligados. 1: mesmo que 0, porém a fonte do clock de 32k, sensor capacitivo, RTC, também são necessariamente desligados.

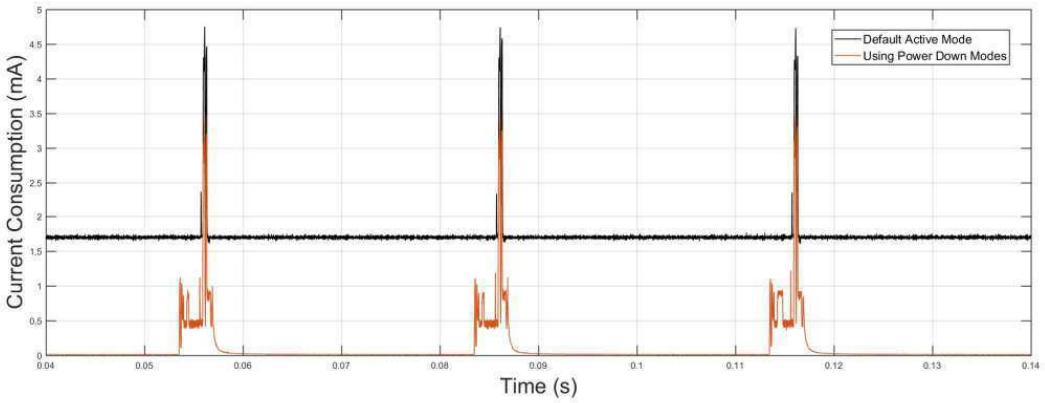
Fonte: [6].

A biblioteca que configura e fornece as funções para ativação dos modos de baixo consumo já havia sido desenvolvida pela equipe de S&A, e foi disponibilizada ao estagiário. Coube ao estagiário adicionar a biblioteca ao projeto, na IDE, e configurá-la para que os modos de baixo consumo fossem chamados corretamente, depurar os erros de compatibilidade que surgiram e garantir que os requisitos estivessem sendo atendidos. Logo após a implementação, o estagiário fez validações dos valores de consumo de corrente para certificar que a biblioteca estava funcionando corretamente e conforme os valores da tabela abaixo:

Tabela 8: Fontes de saída dos modos de baixo consumo.

Modo	Consumo de Corrente	Fontes de *Wake-Up*
Sleep		Configurável
Power Down 0	~2µA	Reset e Brownout Detector Comparador Analógico GPIO previamente selecionados Interrupção do *Sleep Timer* Interrupção do RTC Interrupção do sensor capacitivo
Power Down 1	<1µA	IO externos Comparador Analógico Outras fontes configuradas

Fig.19. Visualização comparativa do consumo instantâneo de corrente em dois modos de consumo.



Fonte: Autor.

A figura acima torna visível o impacto do uso de modos *Power Down* no consumo instantâneo de corrente. Entre os eventos de conexão, assim como entre eventos de *Advertising* o MCU permanecia ativado, porém sem desenvolver nenhuma atividade específica. Quando implementadas as funções de baixo consumo, o *stand-by* é substituído por *Power Down* e a corrente consumida cai drasticamente. Quando necessário realizar o *pooling*, para manter a conexão BLE ativa, o timer de baixo consumo do *stack* BLE interrompe o modo Power Down.

A aplicação botão sem fios possui limitações energéticas muito maiores do que o dispositivo ao qual se conectaria, e.g. lâmpada BLE, pois é alimentado por bateria [ref figura em 2.3]. Diante disso, o estagiário, com base na literatura [4], comparou o consumo de corrente dos *roles peripheral* e *central.* O estagiário apresentou seus resultados ao tutor e engenheiro da equipe de S&A e recebeu confirmação sobre sua decisão de que o botão sem fios teria o *role* de periférico na camada *Link Layer* do *stack* BLE. Isso significa que o dispositivo entraria em uma conexão a partir do estado *Advertising*, que consome muito menos energia do que o estado *Scanning*, executado pelo dispositivo central.

Tabela 9: Comparativo de valores de consumo e autonomia para diferentes configurações da aplicação.

	Modo Ativo	**Modo *Power Down* 0**
Corrente média consumida entre eventos de conexão	1.69mA	12µA
Corrente média consumida	1.73mA	114.6µA
Estimação autonomia de bateria *Coin Cell* 220mAh	127 horas ou 5 dias	1930 horas ou 80 dias

O estagiário identificou que os valores obtidos para o consumo de corrente do MCU no modo *Power Down 0* não estavam de acordo com o esperado, i.e., em torno de

2µA. Após pesquisa, foi identificada a causa no *Datasheet* do MCU: devido a tensões de offset do amplificador operacional, presente no circuito integrado responsável pela medição da corrente consumida mostrado em 2.1.2, as medições de corrente menores que 150µA não devem ser consideradas como precisas.

Diante disso, o estagiário entrou em contato e enviou o projeto em FW para membros da equipe de S&A, para que estes fizessem uso do equipamento próprio para essa situação, um amperímetro da *Keysight* com capacidade de medir corrente na escala de pico-ampères. Após isso os valores foram validados.

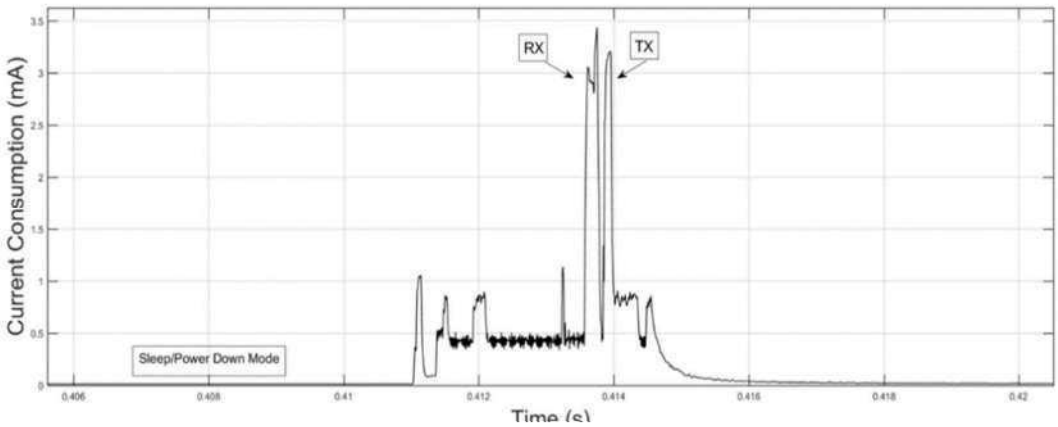
Por fim, uma descrição passo a passo de como implementar as bibliotecas, depurar *bugs* e reduzir o consumo quando o *stack* BLE não é utilizado foi documentada pelo estagiário e adicionada à *Application Note*.

3.4 REDUÇÃO DO CONSUMO DE ENERGIA A PARTIR DO BLE

3.4.1 CONFIGURANDO PARÂMETROS DO *STACK BLE*

Conforme a figura abaixo revela, o pico de consumo de corrente elétrica do MCU ocorre quando o rádio é ativado (para receber ou transmitir informações). Os parâmetros do *stack* BLE, que possuem impacto direto sobre o funcionamento do rádio, terão também impacto sobre o consumo de energia.

Fig. 20. Visualização do consumo instantâneo de corrente com picos de consumo no uso.



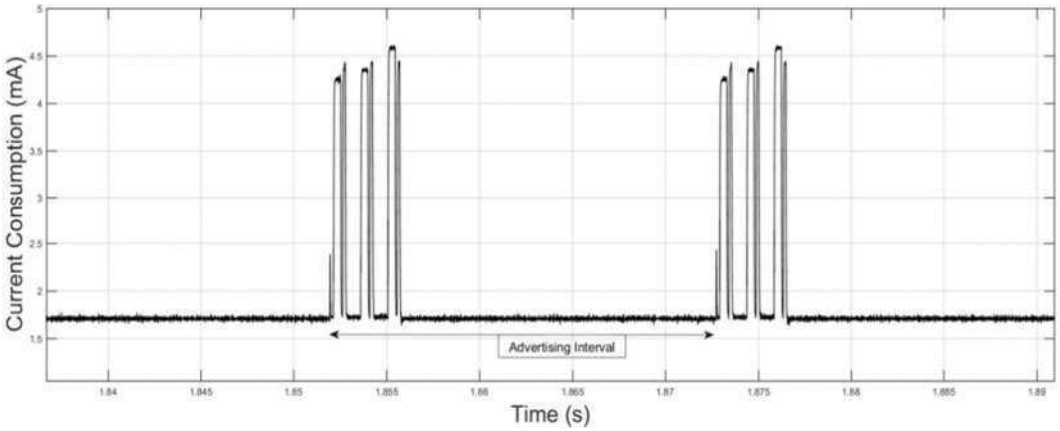
Fonte: Autor.

Durante o estágio, os parâmetros abaixo foram modificados para o aumento da autonomia da bateria, considerando também os outros requisitos do projeto, e.g. latência

do comando. Os resultados foram semanalmente apresentados ao tutor e engenheiros da equipe de S&A.

Advertising Interval: refere-se ao período entre as transmissões de *advertising*, por parte do dispositivo periférico (nesse caso o botão sem fios), de 20ms a 10.24s. Como explicado em 2.2.2., o dispositivo periférico enviará informações sobre si mesmo e aguardará resposta, o pedido de estabelecimento de conexão, do dispositivo central (*Scanner*). Por um lado, quanto mais longo for o período do *Advertising Interval*, menos energia será consumida pelo dispositivo. Por outro lado, quanto mais curto for o período, mais provável que o dispositivo central encontre o periférico e estabeleça rapidamente a conexão.

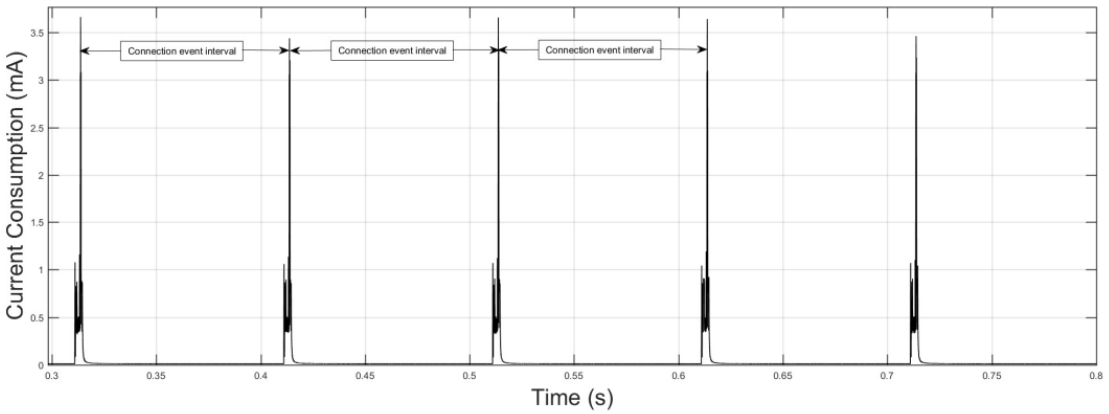
Fig. 21. Visualização consumo instantâneo de corrente com o dispositivo em *Advertising*.



Fonte: Autor.

Connection Interval: trata-se do período entre eventos de conexão, i.e., *connection events*., de 7ms a 4s. Os dois dispositivos, estando conectados, trocarão informações nos momentos determinados por esse período e, logo depois, desativarão os módulos de rádio, podendo mesmo entrar em modo de baixo consumo até o próximo evento. Mesmo que não possuam informações a serem compartilhadas, eles realização o evento, trocando pacotes vazios, a fim de manter a conexão ativa. Caso fosse escolhido um período longo, os dispositivos teriam um baixo consumo médio, porém a latência dos comandos aumentaria.

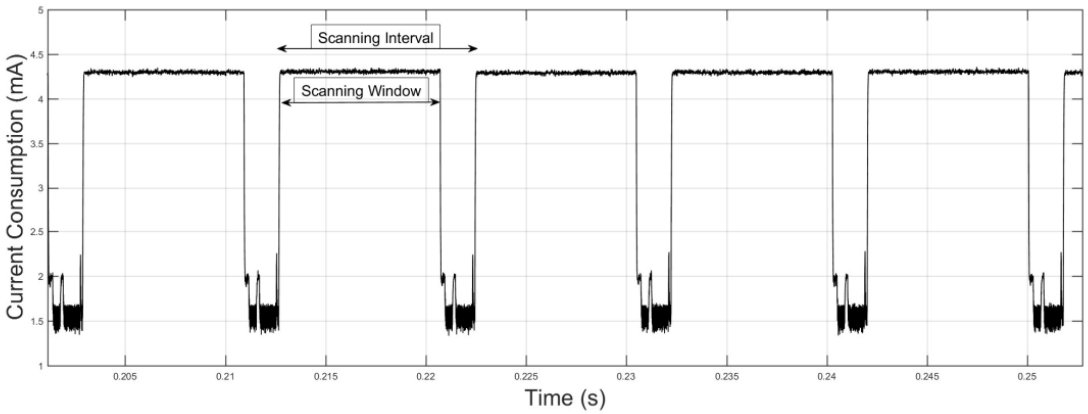
Fig. 22. Visualização do consumo instantâneo de corrente durante conexão.



Fonte: Autor.

Scanning Interval e *Scanning Window*: *Scanning interval* é paralelo ao *Advertising interval*, porém, no dispositivo central. Enquanto um dispositivo está transmitindo informações sobre seus serviços e aguardando possíveis pedidos de conexão, o dispositivo central ativará o seu módulo de rádio para receber os sinais presentes nos canais de *advertising*. Encontrando um dispositivo periférico disponível, ele responderá à mensagem transmitida e fará o pedido. *Scanning Interval* define o período entre eventos de *scanning*, enquanto que o *Scanning Window* define o tempo em que o rádio ficará ativo em um canal, aguardando a transmissão de um periférico, tempo este contido no *Scanning Interval*. A partir do gráfico abaixo, percebe-se o alto consumo de corrente para o dispositivo central nesse evento.

Fig. 23. Visualização do consumo instantâneo de corrente com dispositivo em *Scanning*.

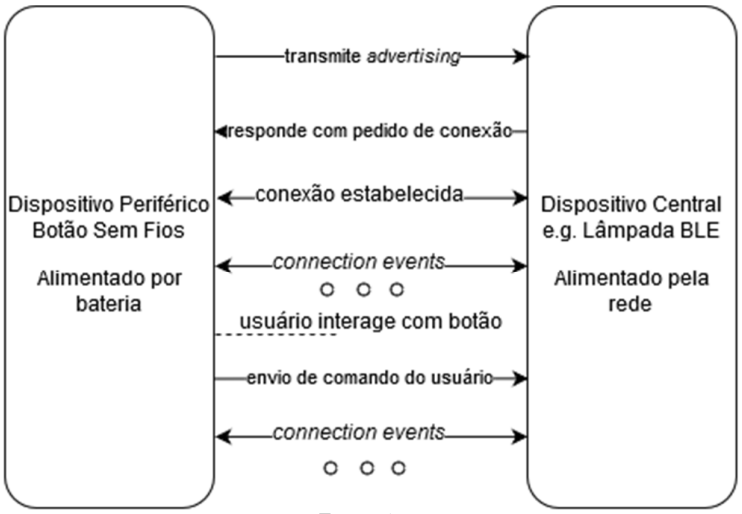


Fonte: Autor.

3.4.2 TORNANDO A APLICAÇÃO *CONNECTIONLESS ORIENTED*

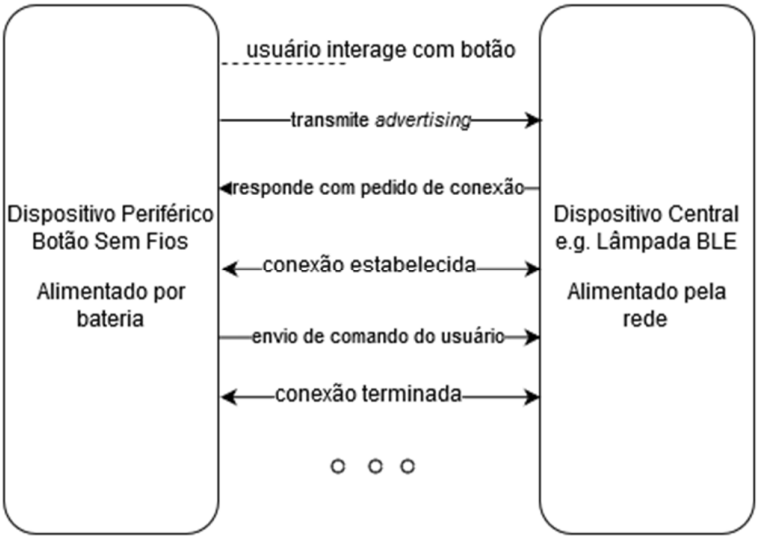
Depois de verificados os *tradeoffs* existentes quanto aos parâmetros BLE, latência e consumo de energia, o estagiário apresentou a ideia, vista na literatura, de manter os dispositivos conectados apenas enquanto houvesse dados a serem atualizados. Após o envio bem-sucedido de um comando, e.g. acendimento de uma lâmpada, a conexão seria finalizada, não sendo necessário os *poolings*, ou *connection events*. A ideia foi aceita para implementação e assim foi feito pelo estagiário, configurando as rotinas dos dispositivos conforme os diagramas abaixo.

Fig. 24. Sequência de eventos com a aplicação orientada à conexão.



Fonte: Autor.

Fig. 25. Sequência de eventos com a aplicação orientada à não-conexão.



Fonte: Autor.

Visto que a conexão seria feita e mantida apenas caso o usuário apertasse algum dos botões, o dispositivo central deveria estar sempre pronto para conectar-se e receber o comando. O dispositivo central, e.g. lâmpada BLE, estaria conectado à rede elétrica, logo o seu consumo de energia não é prioridade tanto quanto o caso do botão. O estagiário configurou o dispositivo central para que, após o fim de uma conexão, não entrasse em *stand-by*, nem em modo de baixo consumo, mas que voltasse ao estado *scanning*, aguardando para o momento em que o dispositivo periférico fosse ativado e, através do *advertising*, se colocasse disponível para conexão e envio de dados. A fim de economizar tempo no estabelecimento da conexão, e consequentemente energia dos dois dispositivos, a conexão foi definida como direcional: da parte do central, i.e., a busca era filtrada primeiramente pelo endereço, qualquer dispositivo encontrado com endereço diferente daquele do botão sem fios era desconsiderado; da parte do periférico, qualquer pedido de conexão era ignorado, se tivesse como origem o dispositivo central de endereço conhecido era diretamente aceito. O estagiário pode comprovar um ganho na autonomia da bateria, porém ainda não suficiente aos requisitos estabelecidos. Como pode ser visualizado na Fig. 28, a estimativa de autonomia para 200 interações diárias passou de aproximadamente oito meses, seguindo um decrescimento linear em função da quantidade de interações com o usuário, para dezesseis meses, ou um ano e quatro meses, seguindo um decrescimento exponencial em função da quantidade de interações.

3.4.3 USANDO *ATTRIBUTE CACHING*

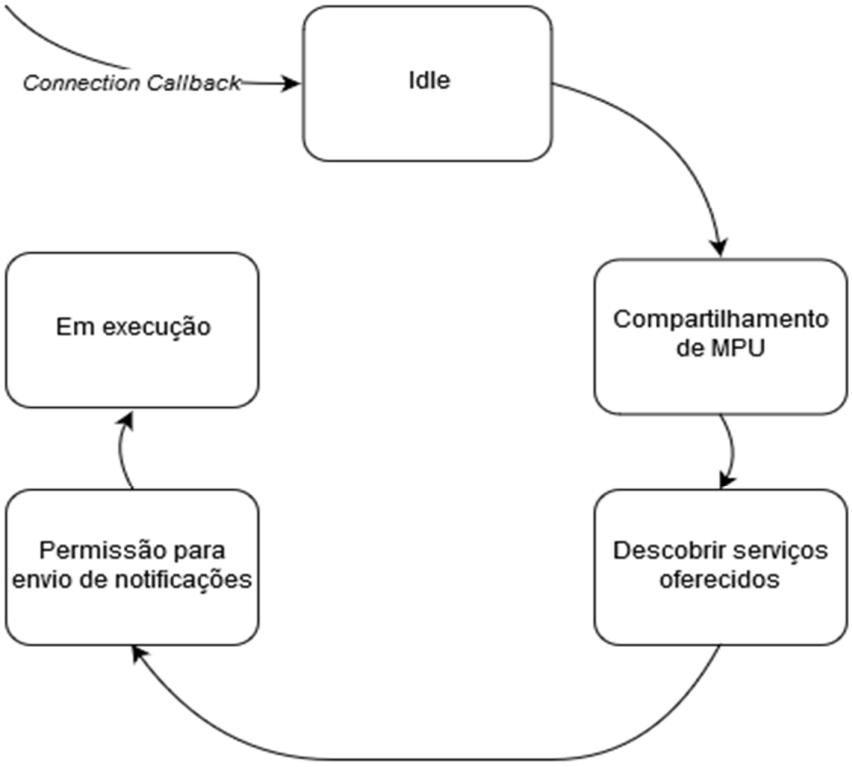
Duas situações foram verificadas pelo estagiário após 3.4.2.: apesar da diminuição no consumo de corrente com a implementação da orientação *connectionless*, a latência era consideravelmente superior a 100ms, em torno de 2s, e muitas transações eram feitas entre os dispositivos a cada envio de comando, apesar de os comandos necessitarem apenas de uma transação para serem enviados, tendo impacto significante no consumo energético.

Foi necessário que o estagiário estudasse então quais são as rotinas definidas pela especificação SIG BLE para o estabelecimento de uma conexão e identificasse como elas estavam implementadas no SDK da NXP para o MCU QN9080. Assim foi feito, e a causa da alta latência a cada envio de comando foi identificada pelo processo de *Service Discovery*. O *Service Discovery*, ou Descoberta de Serviços, é o procedimento de

identificação de todos os perfis, serviços, características, incluindo os *handles*, as permissões, os UUIDs etc., executado pelo *Client* quando se conecta ao *Server*.

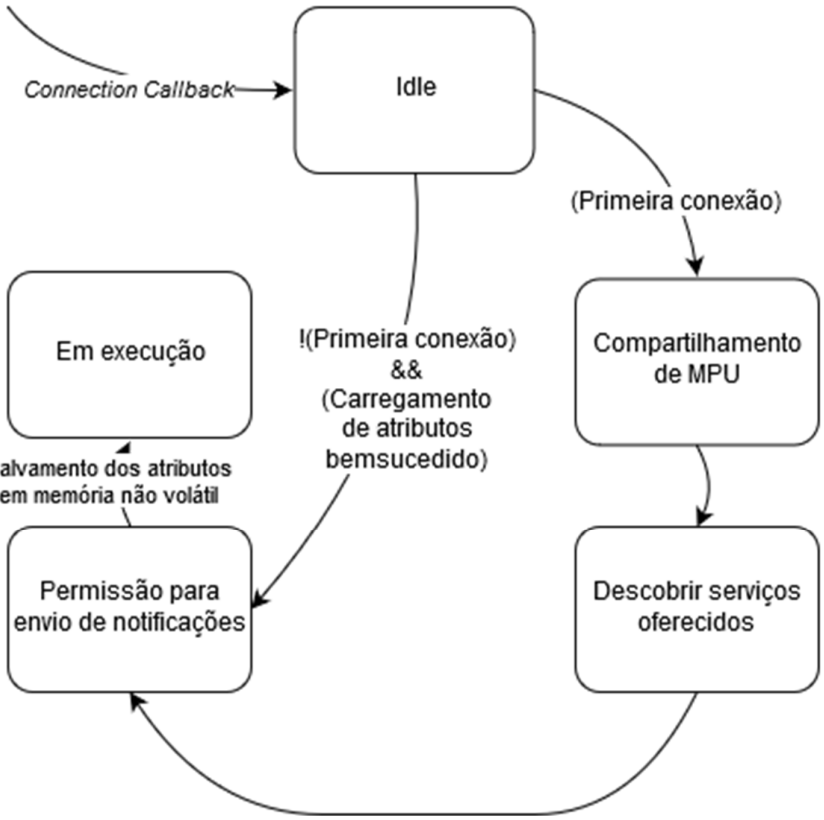
Após identificado e apresentado ao tutor, o estagiário sugeriu a modificação mostrada nas figuras abaixo, i.e., salvar, na primeira conexão após *reset*, os atributos do *Server* em memória não volátil do *Client* e ignorar o procedimento nas conexões seguintes. Foi necessário esclarecer que os serviços oferecidos pelo *Server*, não se alterariam durante o uso da aplicação, seriam inalterados pela aplicação.

Fig. 26. Diagrama de tarefas executadas pelo MCU sem modificações.



Fonte: Autor.

Fig. 27. Diagrama de tarefas executadas pelo MCU após modificações.

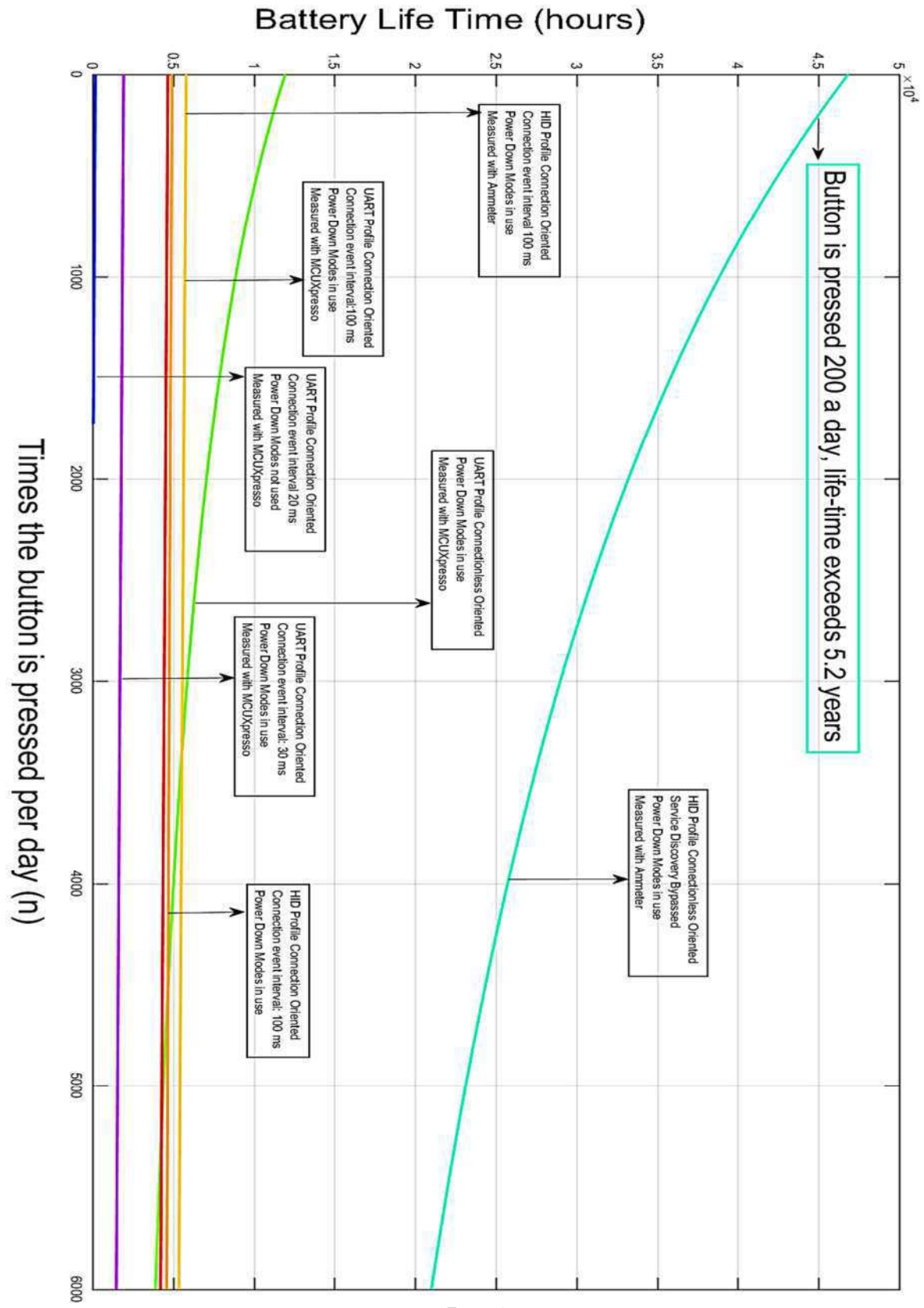


Fonte: Autor.

Essa última modificação possibilitou um aumento considerável na estimativa de autonomia da bateria. Foi necessário, ainda, fazer outras modificações relativas aos parâmetros BLE discutidos em 3.4.1. Tendo em vista a orientação *connectionless* e o requisito de baixa latência, foi definido o *Advertising Interval* como mínimo. Essa escolha foi feita a fim de que as chances de o botão sem fios ser encontrado rapidamente fossem altas, mesmo com maior consumo. Da mesma forma, o *Connection Interval* foi definido como mínimo, para que as transações necessárias ao envio do comando fossem feitas rapidamente e a conexão fosse rapidamente encerrada. O *Scanning Interval* e *Scanning Window* foram aumentados para os mesmos fins.

O gráfico mostrado na Introdução encontra-se a seguir, em maior tamanho. Cada uma das estimações apresenta uma descrição reduzida das configurações implementadas.

Fig. 28. Comparativo da autonomia para diferentes de configurações.

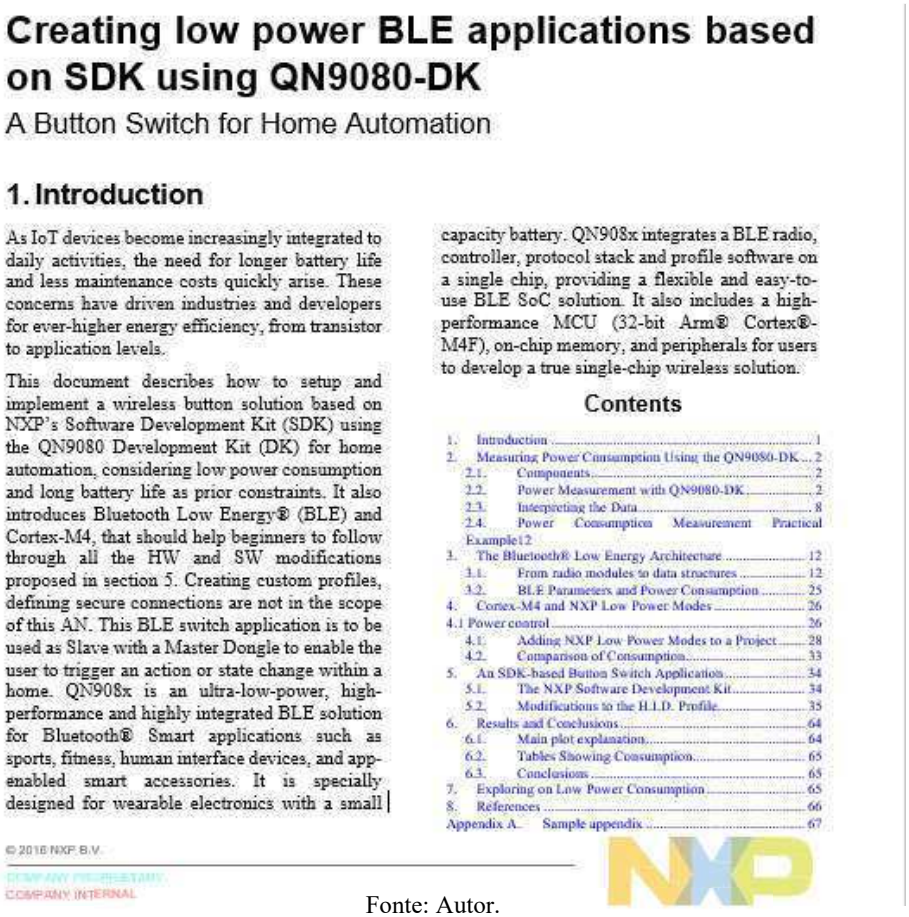


Fonte: Autor.

3.5 ESCRITA DAS NOTAS DE APLICAÇÃO

Por fim, foi iniciada a escrita das notas de aplicação, com uma descrição passo a passo de como criar, com base no SDK da NXP, uma aplicação de baixo consumo. Um ciclo contínuo de revisão e apresentação de resultados, dúvidas e pedidos de feedback ao tutor, engenheiros da equipe de SW BLE, S&A, foi mantido nos últimos meses. Depois de correções e adaptações do escopo, o conteúdo foi definido por: caracterização do consumo de potência a partir da corrente instantânea com o QN9080-DK, uma revisão simples na arquitetura BLE e nas capacidades do Cortex-M4, a implementação em si da aplicação com adaptações para ganho na autonomia da bateria e resultados e conclusões. Foram adicionadas ainda ao final algumas sugestões de medidas que poderiam trazer maior redução para os desenvolvedores e que não foram implementadas por não caber ao escopo ou ao tempo destinado ao estágio.

Fig. 29. Capa das notas de aplicação desenvolvida.



Creating low power BLE applications based on SDK using QN9080-DK

A Button Switch for Home Automation

1. Introduction

As IoT devices become increasingly integrated to daily activities, the need for longer battery life and less maintenance costs quickly arise. These concerns have driven industries and developers for ever-higher energy efficiency, from transistor to application levels.

This document describes how to setup and implement a wireless button solution based on NXP's Software Development Kit (SDK) using the QN9080 Development Kit (DK) for home automation, considering low power consumption and long battery life as prior constraints. It also introduces Bluetooth Low Energy® (BLE) and Cortex-M4, that should help beginners to follow through all the HW and SW modifications proposed in section 5. Creating custom profiles, defining secure connections are not in the scope of this AN. This BLE switch application is to be used as Slave with a Master Dongle to enable the user to trigger an action or state change within a home. QN908x is an ultra-low-power, high-performance and highly integrated BLE solution for Bluetooth® Smart applications such as sports, fitness, human interface devices, and app-enabled smart accessories. It is specially designed for wearable electronics with a small capacity battery. QN908x integrates a BLE radio, controller, protocol stack and profile software on a single chip, providing a flexible and easy-to-use BLE SoC solution. It also includes a high-performance MCU (32-bit Arm® Cortex®-M4F), on-chip memory, and peripherals for users to develop a true single-chip wireless solution.

Contents

Fonte: Autor.

4 CONCLUSÃO

Durante o estágio, aqui relatado, foi possível aplicar efetivamente conhecimentos e técnicas obtidas na Universidade Federal de Campina Grande, assim como no *Institut National des Sciences Appliquées*.

Habilidades transversais, como comunicação, expressão em diferentes línguas, cultivo de relacionamentos interpessoais com outros estagiários, engenheiros, supervisores, chefes e outros funcionários assim como a de ser ativo em colaborar para um ambiente agradável de trabalho foram exercitadas.

A situação exigiu que o estagiário se adaptasse e aprendesse novas técnicas e tecnologias, como aconteceu para o BLE, assim como usar e configurar quase que diariamente equipamentos e circuitos de medição, laboratório, osciloscópios, amperímetros. Isso proveu gradualmente confiança para ir além da busca em adicionar valor às soluções desenvolvidas. A experiência de dar início a um documento que será oferecido a clientes desenvolvedores para a melhoria de seus produtos foi também motivo de satisfação e senso de responsabilidade do estagiário.

Como citado na primeira versão das notas de aplicação, a autonomia da bateria pode ser significativamente aumentada em comparação com o projeto inicial do SDK, de duração de três semanas. É válido lembrar que os projetos do SDK, como o HID, têm por objetivo prover aos desenvolvedores uma base genérica para exploração das capacidades do MCU. A versão final do projeto HID adaptado, focado em baixo consumo, teve autonomia da bateria estimada em cinco anos, dois meses e duas semanas para duzentas interações diárias do usuário, de acordo com as especificações definidas inicialmente.

Como grande parte do trabalho foi finalizada em menos de cinco meses, as últimas semanas foram utilizadas também na busca de objetivos secundários: desenvolvimento do hardware de um circuito específico, placa de avaliação, com interface capacitiva de toque, NFC, alimentado por bateria de 220mAh.

5 REFERÊNCIAS

[1] *Introduction to Microcontroller*. 2004. G. Jack Lipovski.

[2] *Introduction to Microcontrollers*, 2007. Gunther Gridling, Bettina Weiss.

[3] *Introduction to Bluetooth Low Energy*. Acesso: https://www.nxp.com/video/lesson-3-introduction-to-bluetooth-low-energy:LESSON-3-KW41Z-WIRELESS-LAB

[4] *Getting Started with Bluetooth Low Energy*. 2014. Kevin Townsend, Carles Cufí, Akiba, and Robert Davidson.

[5] *Bluetooth Core Specifications*. Acesso: https://www.bluetooth.com/specifications/bluetooth-core-specification

[6] *QN908x Ultra low power Bluetooth 5 System-on-Chip Solution Product Data Sheet*.

[7] Página do MCU *QN908x Ultra-low-power Bluetooth Low Energy SoC Solution.* Acesso: https://www.nxp.com/products/wireless/bluetooth-low-energy/qn908x-ultra-low-power-bluetooth-low-energy-system-on-chip-solution:qn9080

[8] Página do *MCUXpresso Software Development Kit*. Acesso: http://www.nxp.com/support/developer-resources/software-development-tools/mcuxpresso-software-and-tools/mcuxpresso-software-development-kit-sdk:mcuxpresso-sdk

[9] QN908x *Ultra low power Bluetooth 5 System-on-Chip Development Kit User's Guide*.

[10] Acesso: www.bluetooth.org

[11] Acesso: http://microchipdeveloper.com/wireless:ble-link-layer-packet-types

[12] Acesso: https://www.nxp.com

[13] Acesso: https://www.rs-online.com/designspark/investigating-the-arcane-world-of-bluetooth-42-low-energy-ble