PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
ESCOLA POLITÉCNICA
ENGENHARIA DE COMPUTAÇÃO

GLEISER ALVAREZ ARROJO

DESENVOLVENDO APLICAÇÕES IOT NO HTNB32L

Porto Alegre
2023

GLEISER ALVAREZ ARROJO

DESENVOLVENDO APLICAÇÕES IOT NO HTNB32L

Trabalho de conclusão de curso de graduação apresentado na Escola Politécnica da Pontifícia Universidade Católica do Rio Grande do Sul, como requisito parcial para obtenção do grau de Engenheiro de Computação.

Orientador: Julio Cesar Marques de Lima

Porto Alegre

2023

GLEISER ALVAREZ ARROJO

DESENVOLVENDO APLICAÇÕES IOT NO HTNB32L

Trabalho de conclusão de curso de graduação apresentado na Escola politécnica da Pontifícia Universidade Católica do Rio Grande do Sul, como requisito parcial para obtenção do grau de Engenheiro de Computação.

Aprovada em _____ de _____ de _____.

BANCA EXAMINADORA:

Nome do Professor

_____

Nome do Professor

_____

Nome do Professor

_____

AGRADECIMENTOS

Gostaria de expressar minha profunda gratidão a Deus, fonte de todo amor e bondade, por guiar meus passos e abençoar minha jornada. Sou imensamente grato à minha família, cujo amor incondicional e apoio constante são verdadeiros pilares em minha vida. Agradeço também à HT Micron Semicondutores por fornecer o SiP (System-in-Package) que foi fundamental para o meu projeto, demonstrando sua confiança em meu trabalho e impulsionando meu crescimento profissional. Além disso, gostaria de expressar minha sincera gratidão ao professor Julio, cujo conhecimento, orientação e dedicação inestimáveis foram essenciais para o meu aprendizado e desenvolvimento acadêmico. Sou verdadeiramente abençoado por ter tido o privilégio de contar com essas influências significativas em minha jornada.

RESUMO

A Internet das Coisas (IoT) tem revolucionado a forma como interagimos com o mundo ao nosso redor, permitindo a conexão e comunicação entre dispositivos inteligentes. Neste trabalho, foi desenvolvida uma aplicação de Internet das Coisas para ser embarcada em veículos utilizando o SiP HTNB32L da HT Micron Semicondutores e a rede NB-IoT. A aplicação consiste em um detector de colisões frontais, que ao detectar uma colisão, envia uma mensagem para os serviços de emergência.

Palavras-chave: Internet das Coisas. NB-IoT. Carros Conectados.

ABSTRACT

The Internet of Things (IoT) has revolutionized the way we interact with the world around us, allowing connection and communication between smart devices. In this work, an Internet of Things application was developed to be embedded in vehicles using the SiP HTNB32L from HT Micron Semiconductors and the NB-IoT network. The application consists of a frontal collision detector, which, when detecting a collision, sends a message to the emergency services.

Key words: Internet of Things. NB-IoT. Connected cars.

LISTA DE ILUSTRAÇÕES

Nenhuma entrada de sumário foi encontrada.

LISTA DE SIGLAS

IoT - *Internet of Things*

UE - Equipamento de Usuário

UEs - Equipamentos de Usuário

NB - IoT - *Narrow Band - Internet of Things*

iMCP - *integrated Multi Chip Package*

LTE - *Long Term Evolution*

GSM - *Global System for Mobile Communications*

RF - Radiofrequência

LSE - *Low-Speed External*

HSE - *High-Speed External*

PN - *Part Number*

SDK - *Software Development Kit*

HAL - *Hardware Abstraction Layer*

API - *Application Programming Interface*

SUMÁRIO

1 INTRODUÇÃO

Internet of Things (IoT), do inglês, Internet das Coisas é um tema recorrente quando o debate é sobre revolução tecnológica. No mundo todo, empresas investem recursos em pesquisa e desenvolvimento de soluções eficientes e acessíveis para atender as demandas deste mercado.

Usando a Internet das Coisas, é possível conectar diversos tipos de sensores e atuadores para obter dados e atuar no meio para aumentar a eficiência das atividades nos diversos setores onde pode-se utilizar estes dispositivos.

Para conectar estes dispositivos é preciso utilizar tecnologias *wireless* de longo alcance, especialmente tecnologias que foram desenvolvidas para utilizar pouca energia dos dispositivos: as *LPWAN (Low-Power Wide Area Networks)*. As *LPWAN* mais utilizadas são Sigfox, LoRaWAN e NB-IoT. Redes desenvolvidas para aplicações que priorizam o baixo consumo de energia e longo alcance.

A HT Micron Semicondutores, empresa brasileira especializada em encapsulamento de memória e SiP (*System-in-Package)* para soluções de Internet das Coisas, já possui em seu catálogo de produtos para IoT: um SiP que utiliza a rede Sigfox; um SiP que utiliza a rede LoRaWAN; e está desenvolvendo um SiP que navega na rede NB-IoT.

Este trabalho tem por objetivo explorar a tecnologia NB-IoT e desenvolver uma aplicação de Internet das Coisas utilizando o SiP HTNB32L, fornecido pela HT Micron Semicondutores. A aplicação consiste em um detector de colisões de veículos que, ao detectar uma colisão frontal, envia uma mensagem para os serviços de emergência contendo a localização do veículo.

Utilizando tecnologias de ponta, como o SiP HTNB32L e a rede NB-IoT, e aplicando os conhecimentos obtidos durante a graduação em engenharia de computação, espera-se obter um sistema funcional que poderá servir de inspiração para ideias similares ou até mesmo, um produto para a indústria de Internet das Coisas.

2 DESCRIÇÃO DAS TECNOLOGIAS

Este trabalho explorou uma série de tecnologias utilizadas para o desenvolvimento de aplicações IoT: a rede NB - IoT e o iMCP HTNB32L, projetado pela HT Micron, o acelerômetro MPU-6050, o GNSS u-blox MAX-8Q e o protocolo de aplicação MQTT. Nesta seção, serão descritas estas tecnologias.

2.1 NB - IOT

NB - IoT - *Narrow Band Internet of Things* - é uma *wireless LPWAN (Low Power Wide Area Network)* destinada a aplicações de Internet das Coisas. É uma tecnologia rede de celular derivada da LTE, proposta pela 3GPP, a partir da release 13 com foco em aplicações de baixo consumo de energia, baixa taxa de transferência de dados, baixo custo e ampla cobertura.

2.1.1 *Narrow Band*

A rede NB-IoT utiliza uma única portadora com largura de banda de 180 kHz, tanto no *uplink* como no *downlink*. Isto é equivalente a um Bloco de Recursos Físicos (*PRB*) em LTE. Por ser de natureza de banda estreita, a rede requer pequenos pedaços de espectro da banda LTE ou GSM, por exemplo.

Como na rede LTE, a NB - IoT utiliza Multiplexação por Divisão de Frequência Ortogonal (*OFDM*) no *downlink,* com espaçamento de subportadora de 15 kHz e modulação QPSK. No *uplink,* a modulação pode ser BPSK ou QPSK e multiplexação por SC-FDMA, além de poder utilizar única subportadora com espaçamento de 3,75 kHz, ou 15 kHz, ou múltiplas subportadoras com espaçamento de 15 kHz.

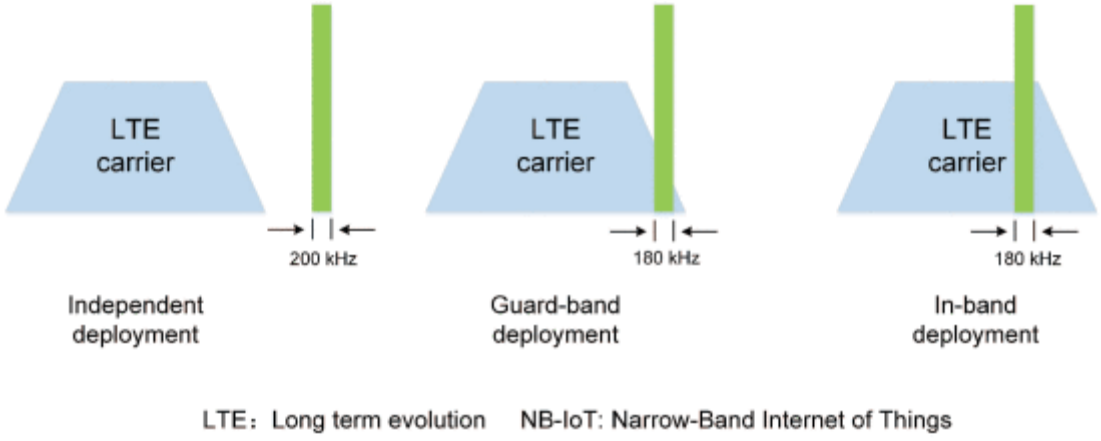
Tanto no *downlink* quanto no *uplink,* a taxa de transmissão de dados é entre 160 e 250 kbits / s.

2.1.2 Modos de Implantação

Sendo uma rede derivada da LTE, a NB - IoT pode ser implantada em três modos de operação, os quais são ilustrados na Figura 2.1:

- *Stand-alone*: utiliza uma banda de frequência independente que não se sobrepõe às portadoras LTE.
- *In-band*: utiliza a banda LTE existente para implantar.
- *Guard-band*: utiliza a borda da banda LTE existente para implantar.

Figura 2.1 - Modos de implantação NB-IoT



Fonte: Min Chen (2017)

2.1.3 *eDRX e PSM*

Com a finalidade de possibilitar o baixo consumo de bateria nos UEs, (Equipamentos de Usuário, em português) a rede NB - IoT implementa tecnologias que possibilitam reduzir o consumo de energia total dos dispositivos.

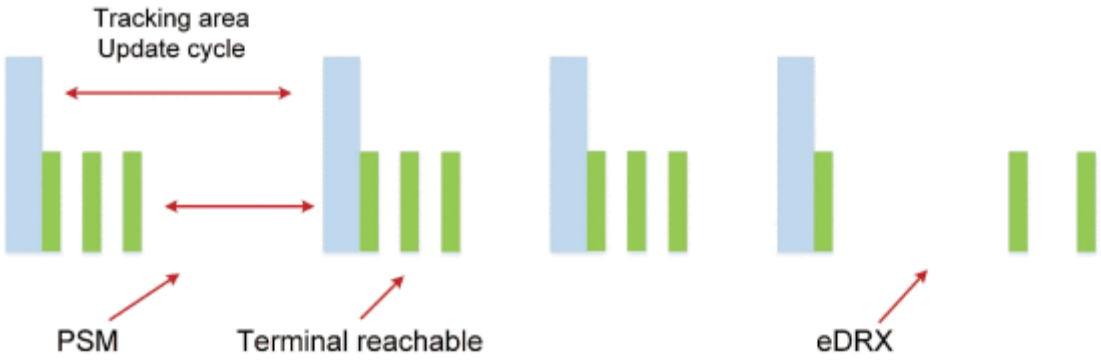
No modo *expanded Discontinuous Reception (eDRX)*, o UE mantém o estado ocioso, não podendo solicitar recursos para a rede, mas rastreia o canal de *downlink* para manter a sincronização com a rede e verificar se há dados de *downlink* pendentes.

Enquanto está no modo *eDRX*, o dispositivo alterna entre períodos de escuta ativa e períodos de hibernação. Desse modo, somente alguns *frames* serão recebidos a cada período de escuta, podendo perder alguns *frames* nos períodos de hibernação.

Em paralelo ao *eDRX,* pode-se utilizar o *Power Saving Mode (PSM)* para obter um tempo de espera mais longo entre mensagens. O dispositivo permanece conectado à rede, porém não pode ser alcançado, pois está hibernando. Por conta disso, mantém o rádio desligado e não é capaz de receber os *frames*.

Nesse modo a rede mantém as notificações destinadas ao UE até que o tempo de hibernação termine e o dispositivo volte a se comunicar com a rede. A Figura 2.2 ilustra como funcionam essas tecnologias.

Figura 2.2 - Ilustração do *eDRX* e *PSM*



Fonte: Min Chen (2017)

Utilizando essas tecnologias de economia de energia, é possível manter a hibernação profunda pelo maior tempo possível, alcançando, assim, a economia de energia no dispositivo.

2.1.4 Arquitetura da rede NB-IoT

Sendo uma rede com foco em Internet das Coisas, a rede NB-IoT segue um modelo de redes IoT amplamente utilizado. A arquitetura da rede é dividida em cinco partes, também ilustradas na Figura 2.3:

- Terminal NB-IoT: também chamados de Equipamentos de Usuário. São dispositivos com sensores para diversas aplicações com um SIM card para acessar a rede.

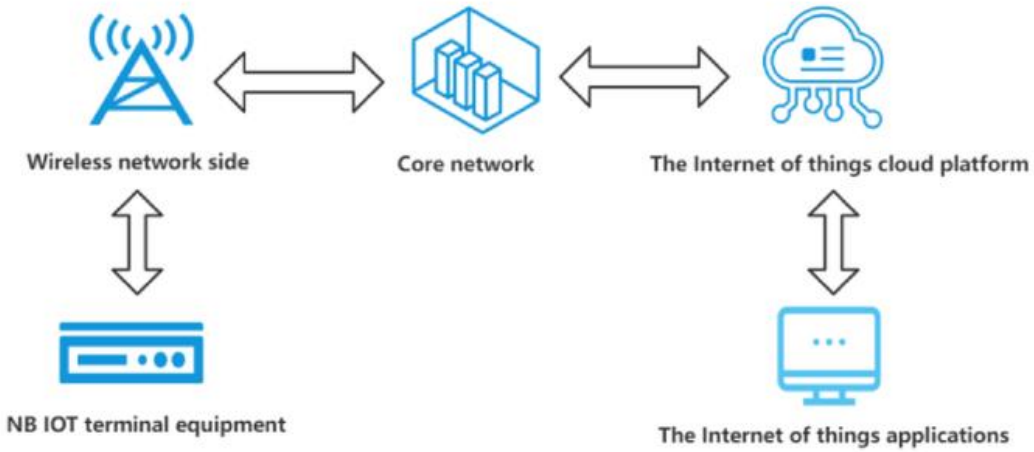
- *NB-IoT Base Station*: são as estações de interface para Terminais acessarem a Internet, disponibilizadas pelas operadoras de telecomunicações.

- *Rede Principal*: rede responsável por transmitir os dados dos Terminais para a plataforma de Nuvem IoT.

- Plataforma de Nuvem IoT: servidor onde acontece o processamento dos dados recebidos dos Terminais. Realizam o controle remoto dos equipamentos com envio e recebimento de dados.

- Aplicação IoT: realiza a implementação final do sistema, podendo ser um servidor ou software que analisa e trata os dados de acordo com os serviços IoT.

Figura 2.3 - Arquitetura da rede NB-IoT

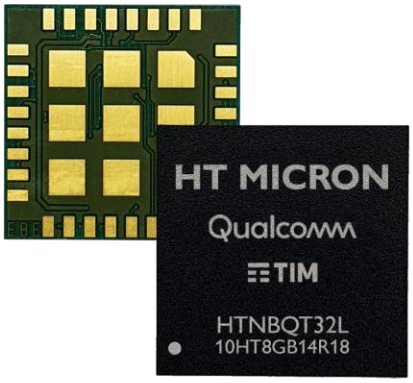


Fonte: Huale Zhu (2022)

2.2 IMCP HTNB32L

O iMCP HTNB32L é um *System-in-Package* produzido pela HT Micron Semicondutores para aplicações de Internet das Coisas, ilustrado na Figura 2.4. Trata-se de um circuito multicomponentes, pois contém um microcontrolador integrado a um modem LTE, uma chave RF, cristais LSE e HSE e, dependendo do Part Number, um e-SIM.

Figura 2.4 - iMCP HTNB32L



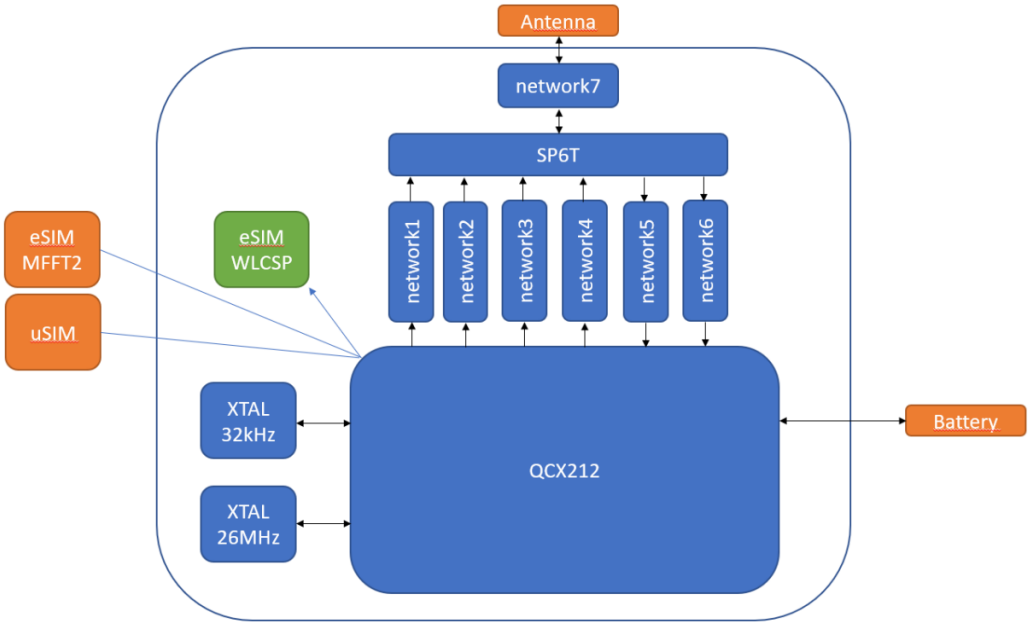
Fonte: GitHub da HT Micron (2023)

O SiP HTNB32L foi projetado para ser altamente compacto, com características de baixo consumo de energia que suporta a tecnologia de conectividade NB-IoT.

Para possibilitar estas características, o SiP foi desenvolvido com componentes que possuem um baixo consumo de energia. Os componentes são: um *transceiver* Qualcomm QCX212, que integra um microcontrolador ARM Cortex M3 e um Modem LTE, para comunicar com a rede NB-IoT. Estes componentes estão representados no diagrama de blocos da Figura 2.4. Além disso, o SiP integra um cristal LSE de 32768 Hz para referência de *Real Time Clock* e um cristal HSE de 26 MHz, responsável pelo *clock* de referência do sistema.

O HTNB32L possui dois PNs: um PN contendo um e-SIM encapsulado no sistema; e o outro PN, que não possui um e-SIM encapsulado. Neste último, a compra e integração do SIM Card, ou e-SIM é de responsabilidade do usuário.

Figura 2.5 - Diagrama de blocos do iMCP HTNB32L



Fonte: Product Spec Sheet - HT Micron Semicondutores (2023)

2.1.1 Especificações Técnicas

As especificações técnicas a seguir foram coletadas do documento de especificações técnicas, disponibilizado pela HT Micron:

- CPU: ARM Cortex M3 @ 204MHz/102MHz/26MHz
- 4 MB NOR flash
- 272Kb SRAM: 256Kb + 16Kb instruction cache
- Digital Interfaces
    - 2x I2C,
    - 3x UART,
    - 2x SPI,
    - 2-Channel 12-bit ADC,
    - 10x GPIOs:
        - 6x PWM,
        - 4x Timer
        - 1x WAKEUP
        - 1x AON (keeping output during deep sleep)

Neste projeto, foram utilizadas: uma interface I2C para fazer a escrita e leitura de dados no acelerômetro MPU-6050; uma interface UART para fazer a escrita e leitura de dados no GNSS u-blox MAX-8Q; e uma interface UART para imprimir informações de *debug* da aplicação em um terminal serial. Esta última interface sendo totalmente opcional, isto é, a aplicação funciona sem ela.

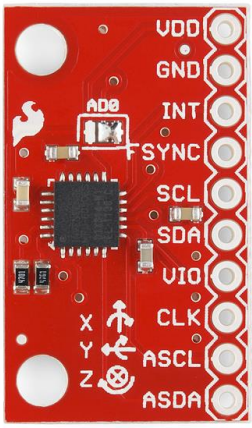
2.3 MPU-6050

O MPU-6050 é um circuito integrado que combina um acelerômetro de três eixos e um giroscópio de três eixos em um único encapsulamento. Esse dispositivo foi escolhido por estar disponível no Laboratório de Ensino de Eletrônica da PUCRS.

O acelerômetro mede a aceleração linear do dispositivo em três eixos (x, y e z), enquanto o giroscópio mede a velocidade angular em torno desses mesmos eixos. Essas medidas são utilizadas para determinar a orientação do dispositivo no espaço tridimensional.

O MPU-6050 se comunica com o microcontrolador ou processador do sistema por meio de uma interface I2C (Inter-Integrated Circuit), na qual é possível configurar o dispositivo escrevendo nos registradores, ou lendo os dados necessários.

Figura 2.6 - MPU-6050 Breakout



Fonte: Site da Sparkfun (2023)

2.4 U-BLOX MAX-8Q

O u-blox MAX-8Q é um módulo GNSS compacto projetado para fornecer posicionamento preciso e confiável em tempo real para aplicações, como navegação automotiva, rastreamento de veículos, dispositivos portáteis de navegação e geolocalização em geral.

O MAX-8Q é baseado na tecnologia GNSS (Global Navigation Satellite System), que usa sinais de satélite para determinar a localização, velocidade e hora do dia com alta precisão. Ele suporta os sistemas de satélite GPS (Estados Unidos), GLONASS (Rússia), Galileo (União Europeia) e BeiDou (China), o que aumenta a disponibilidade e confiabilidade do sinal em qualquer lugar do mundo.

O módulo é compacto, com dimensões de 16 x 16 x 6,5 mm, e é alimentado com uma tensão de 3,3V. Ele possui uma interface serial UART para comunicação com o microcontrolador do sistema, bem como saídas de dados em NMEA (National Marine Electronics Association) e UBX (u-blox proprietary) para uma ampla gama de aplicações.

O u-blox MAX-8Q, ilustrado na Figura 2.7, é conhecido por sua alta sensibilidade e baixo consumo de energia, o que o torna adequado para aplicações de bateria e dispositivos portáteis. Ele também inclui recursos adicionais, como um receptor SBAS (Satellite-Based Augmentation System), para correção de erros de posicionamento, e suporte a funções de rastreamento de movimento para aplicações em movimento.

Figura 2.7 - u-blox MAX-8Q

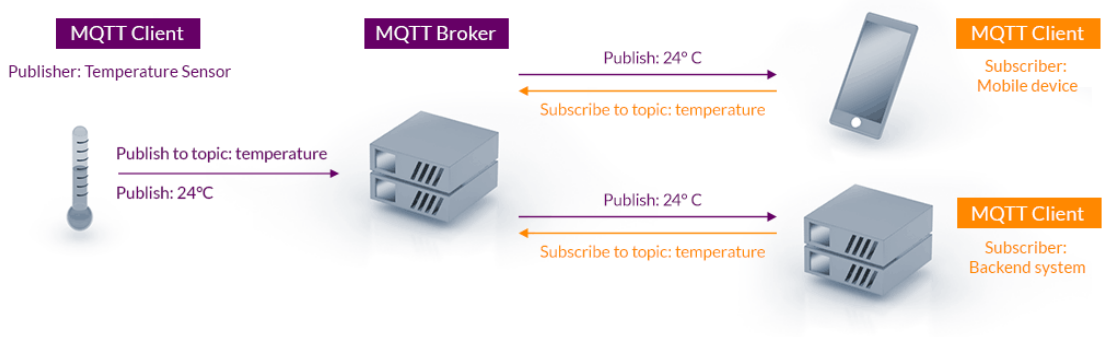
Fonte: Site da u-blox (2023)

2.5 MQTT

MQTT (*Message Queuing Telemetry Transport*) é um protocolo de mensagens leve e eficiente projetado para a comunicação entre dispositivos em redes com recursos limitados. Foi desenvolvido para atender às necessidades de comunicação confiável e de baixo consumo de energia em ambientes de Internet das Coisas.

O MQTT é baseado em um modelo de publicação/assinatura (*publish/subscribe*), ilustrado na Figura 2.8, em que os dispositivos são divididos em dois papéis principais: os publicadores (*publishers*) e os assinantes (*subscribers*). Os publicadores enviam mensagens para tópicos específicos, enquanto os assinantes se inscrevem em tópicos de interesse para receber essas mensagens.

Figura 2.8 - Arquitetura MQTT Publicação/Assinatura



Fonte: Site mqtt.org (2023)

O protocolo MQTT utiliza a arquitetura TCP/IP, criando conexões TCP entre os clientes e o Broker para enviar mensagens e assinar os tópicos. Desse modo, é possível garantir a entrega das mensagens publicadas e a assinatura dos clientes no tópico desejado. Também é possível utilizar criptografia SSL/TLS e mecanismos de autenticação de clientes no broker. Garantindo a segurança dos dados trafegados nesse protocolo.

3 APLICAÇÕES DE IOT

A Internet das Coisas é uma tecnologia em constante evolução, tornando os dispositivos e as aplicações cada vez mais inteligentes, eficientes e autônomos. Essa tecnologia é utilizada em diversas áreas, como na indústria, na agricultura, em cidades inteligentes, na área da saúde e em automóveis. Neste capítulo, serão abordados alguns exemplos de aplicações de Internet das Coisas, bem como a aplicação que foi desenvolvida para o HTNB32L e o ambiente de desenvolvimento.

3.1 EXEMPLOS DE APLICAÇÕES

No setor industrial, é válido citar aplicações da Internet das Coisas, como sensores de temperatura, umidade, vibração em uma fábrica, os quais permitem que os gerentes monitorem o ambiente e tomem ações em tempo real. Também é usada no controle de estoque e mercadorias, com etiquetas eletrônicas, que informam em quais locais aquele lote passou e onde foi armazenado.

Na agricultura, é possível monitorar o solo, o clima e as plantas, ajudando a otimizar a irrigação, fertilização e colheita. Isso pode melhorar a produtividade, reduzir o desperdício e ajudar os agricultores a tomarem decisões informadas sobre o cultivo.

Nas cidades inteligentes, pode-se utilizar a tecnologia para monitorar o tráfego, gerenciar iluminação pública, monitorar a qualidade do ar e da água e melhorar a eficiência do transporte público. Isso tem impactos positivos de forma efetiva, como, por exemplo, na redução do congestionamento, no aprimoramento da segurança pública e no monitoramento de problemas ambientais.

Na área da saúde, algumas aplicações incluem dispositivos vestíveis para monitorar sinais vitais, atividade física, dieta e sono. Esses dispositivos permitem que os médicos monitorem pacientes remotamente, detectem doenças precocemente e personalizem o tratamento.

Nos automóveis, a tecnologia é usada na conexão dos veículos para melhorar a segurança e a experiência do motorista. Os sensores monitoram o desempenho do veículo, a manutenção e os padrões de direção, para detectar e prevenir

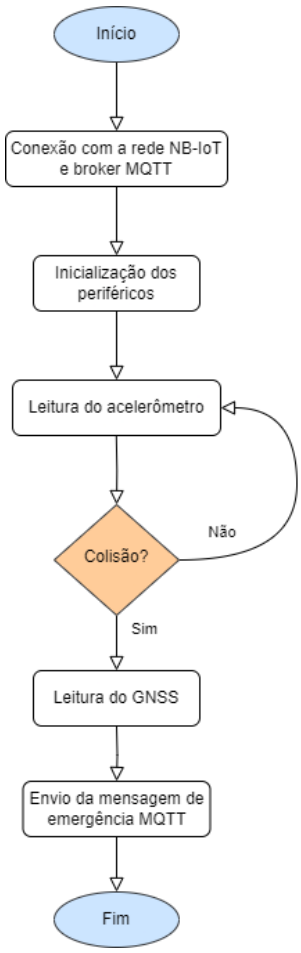
acidentes, podendo-se conectar com as cidades inteligentes e fornecer um serviço de resgate automatizado, sem precisar da ação do motorista.

3.2 APLICAÇÃO DESENVOLVIDA

Para este trabalho, foi desenvolvida uma aplicação de Internet das Coisas para ser embarcada em automóveis. A aplicação é um sistema de detecção de acidentes automatizado, que tem por objetivo detectar uma colisão e acionar os serviços de emergência disponíveis, enviando os dados de localização do automóvel.

A aplicação é composta por uma máquina de estados finitos, ilustrada na Figura 3.1. O início da aplicação se dá quando o HTNB32L é conectado na fonte de alimentação e o botão de reset é pressionado. Na sequência, a aplicação segue a máquina de estados que será descrita a seguir.

Figura 3.1 - Máquina de Estados da Aplicação



Fonte: Própria (2023)

3.2.1 Conexão com a rede NB-IoT e Broker MQTT

Antes de carregar a aplicação no HTNB32L, é necessário realizar a conexão do SiP com a rede NB-IoT, para poder fazer a comunicação do dispositivo com a internet. Essa conexão, ou registro, é feita por envio de comandos ao dispositivo a partir do *firmware* de comandos AT, disponibilizado pela HT Micron.

Para isto, foi necessário gravar o *firmware* de comandos AT no SiP. Logo após, foi enviada uma série de comandos através de um terminal serial para poder registrar o dispositivo na rede NB-IoT.

A sequência de comandos necessárias para fazer o registro é a seguinte:

1. AT$QCBAND=0,3,28\r\n; seleciona as bandas a serem utilizadas;
2. AT+CPSMS=0\r\n; desativa o PSM;
3. AT+CGDCONT=0,"IPV4V6","nbiot.gsim"\r\n; seleciona a rede;
4. AT+CEREG=3\r\n; inicia a busca manual pela rede;
5. AT+COPS=1,2,"72404"\r\n; seleciona a operadora;

A confirmação do registro na rede se dá pelo retorno da mensagem “+CEREG: 1”, indicando que o dispositivo foi registrado na rede com sucesso. Com o registro na rede, é possível utilizar a pilha TCP/IP para fazer a comunicação do dispositivo com a internet.

Os parâmetros enviados a partir dos comandos AT ficam salvos na memória não volátil do HTNB32L. Dessa maneira, é possível desligar o dispositivo, ou fazer a gravação de outra aplicação sem perder os dados de registro na rede NB-IoT. Assim que uma nova aplicação for gravada no SiP, o *firmware* é responsável por registrar o dispositivo na rede, sem a necessidade de intervenção do usuário.

Com o dispositivo registrado na rede NB-IoT, a partir dos comandos AT, foi feita a gravação do *firmware,* contendo a aplicação desenvolvida durante este trabalho. Após a gravação do novo *firmware*, o sistema resgatou os dados de registro na rede armazenados na memória e fez novamente o registro.

A partir do registro, foi possível conectar em um *broker* MQTT para enviar as mensagens de possíveis acidentes com o veículo. Neste trabalho, um *broker* MQTT representa um canal de comunicação com um serviço de emergência ou uma seguradora responsável pelo atendimento de ocorrências.

O *broker* MQTT utilizado foi o HiveMQ, um servidor MQTT gratuito amplamente utilizado para aplicações de carros conectados, como a desenvolvida

neste trabalho, disponível no endereço broker.hivemq.com através da porta TCP 1883.

A partir da conexão com o *broker* MQTT, é possível publicar mensagens em tópicos e assinar estes tópicos com diversos dispositivos, como outros dispositivos de IoT, aparelhos celulares e, até mesmo, outros servidores de aplicações.

Após a conclusão do registro na rede NB-IoT e da conexão do HTNB32L em um *broker* MQTT, o próximo estado da máquina de estados entra em ação.

3.2.2 Inicialização dos periféricos

Após a inicialização do sistema e registro na rede NB-IoT, o próximo estado da máquina de estados da aplicação é a inicialização dos periféricos: o acelerômetro MPU-6050 e o GNSS u-blox MAX-8Q.

Para fazer a interface com o MPU-6050, foi criado um driver para abstrair a escrita nos registradores do acelerômetro. Este driver contém todas as funções utilizadas para fazer a escrita e a leitura nos registradores do MPU-6050. Mais detalhes sobre o driver estão na seção 3.3.

A partir desse driver, foi possível configurar o acelerômetro da seguinte forma: seleção de *clock* a partir do oscilador interno; configuração do giroscópio em *stand by*, para o sistema gastar menos energia; seleção da escala de sensibilidade do acelerômetro (±2g, ±4g, ±8g ou ±16g), conforme os testes na seção 3.2.n; habilitação da escrita do acelerômetro na FIFO (fila First In First Out) interna do dispositivo; configuração da taxa de amostragem do acelerômetro na FIFO em 100 Hz.

A configuração padrão do u-blox MAX-8Q é utilizar o GPS para a localização e envio de seis mensagens do padrão NMEA (são elas: RMC, VTG, GGA, GSA, GSV, GLL), contendo a localização através da interface serial UART com frequência de 1 Hz. A mudança na configuração foi filtrar as mensagens para receber somente a mensagem GLL com a mesma frequência de 1 Hz.

Baseado nessas configurações, foi possível inicializar e realizar a integração dos periféricos necessários para a aplicação. Após a inicialização dos periféricos, é possível ler os dados do acelerômetro para identificar uma possível colisão.

3.2.3 Leitura dos dados do acelerômetro

A leitura dos dados do acelerômetro é feita periodicamente com base na leitura do registrador FIFO_R_W. Este registrador contém o primeiro dado de 8 bits escrito na FIFO do dispositivo.

De acordo com a inicialização do MPU-6050 descrita na seção 3.2.2, a FIFO do dispositivo armazena os dados de aceleração coletados dos eixos x, y e z. Cada eixo produz uma variação de aceleração que é armazenada como um dado de 16 bits. Portando, para fazer uma leitura de cada eixo do acelerômetro, foi lido duas vezes o valor do registrador FIFO_R_W.

O dispositivo permite a leitura em modo rajada, podendo acumular até 1024 bytes na FIFO interna antes de acorrer *overflow*, onde os dados mais antigos seriam substituídos por novos dados. Para evitar o *overflow*, a leitura da FIFO foi realizada a cada 500ms, lendo em rajada em torno de 300 bytes, ou 150 acelerações de eixos x, y e z. Dessa forma, garantiu-se que não ocorresse *overflow* na FIFO e nenhum dado fosse sobrescrito.

Antes de realizar uma nova leitura, os dados coletados são analisados no estado de Detecção de colisão. Caso não seja detectada uma colisão, a máquina de estados retorna para o estado atual e realiza novamente todo o processo de leitura dos dados.

3.2.4 Detecção de colisão

Após a leitura dos dados do acelerômetro e o armazenamento destes dados no *buffer* do dispositivo, tem-se a viabilidade de analisá-los e identificar variações que correspondam a padrões identificados em colisões.

Segundo Punetha, Kumar, Mehta (2012), acidentes entre 5g e 20g são considerados acidentes leves, ao passo que acidentes entre 30g e 40g são considerados acidentes médios, e acima de 50g são considerados acidentes graves.

Porém, devido a sensibilidade máxima do acelerômetro ser de ±16g, a maior gravidade de acidente que pode ser detectada é um acidente leve. Deste modo, a aplicação foi desenvolvida e testada para que sejam identificados acidentes, mesmo que somente os considerados leves.

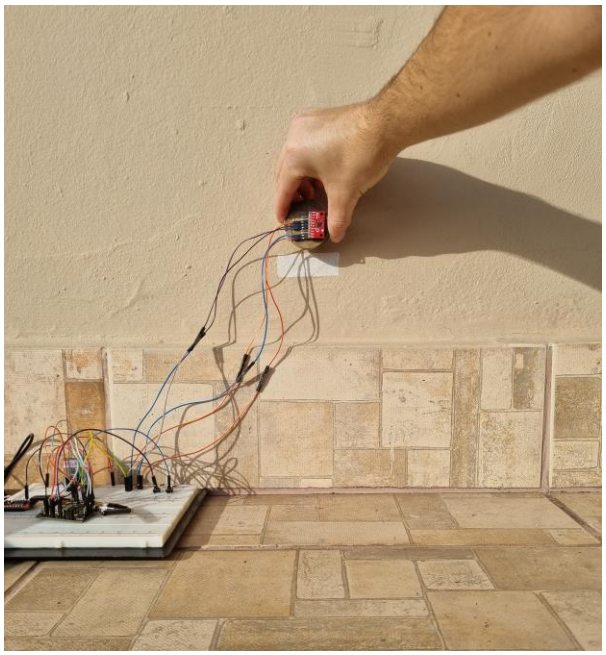
Os testes para simular uma colisão foram realizados com o protótipo descrito na seção 3.3.1. O acelerômetro MPU-6050 foi fixado em uma massa de prova, ilustrado na Figura 3.2, simulando a fixação em um automóvel.

Figura 3.2 - MPU-6050 fixado na massa de prova

Fonte: Própria (2023)

A colisão foi simulada soltando o acelerômetro fixado na massa de prova de uma altura de 20cm de uma bancada rígida. O acelerômetro foi posicionado inicialmente com o eixo x apontando para baixo, ilustrado na Figura 3.3, correspondendo a um veículo andando em linha reta.

Figura 3.3 - Posicionamento do acelerômetro nos testes



Fonte: Própria (2023)

A colisão do acelerômetro fixado na massa de prova com a superfície da bancada simula uma colisão frontal de um automóvel.

Para visualizar a variação da aceleração quando o acelerômetro atinge a superfície da bancada, foi utilizado o software Arduino IDE. Este software contém uma ferramenta gráfica que plota os dados recebidos pela interface serial em gráfico, facilitando a visualização dos dados. Mais detalhes da conexão serial estão na seção 3.3.1.

Foram realizados cinco testes para cada sensibilidade configurável no MPU-6050: ±2g, ±4g, ±8g e ±16g. No Apêndice A estão as imagens dos resultados deste teste de colisão. Em todas as configurações de sensibilidade do MPU-6050, as colisões atingiram o limite de registro de aceleração do dispositivo.

Portanto, a aplicação final foi desenvolvida configurando o acelerômetro com a máxima escala de aceleração: ±16g e definindo o limiar de colisão como 16g (ou 156 m/s). Desta maneira, quando o eixo x do acelerômetro atingiu o limite da escala: +16g, foi considerado como uma colisão frontal.

Utilizando este dispositivo com estas configurações, a aplicação é capaz de identificar acidentes considerados, no mínimo, acidentes leves. Ao identificar a colisão, a máquina de estados avança para o estado de Leitura do GNSS. Caso não seja detectada, retorna ao estado de Leitura dos dados do Acelerômetro.

3.2.5 Leitura do GNSS

O GNSS envia a mensagem NMEA GLL que contém a localização com a frequência de 1 Hz. Neste estado, o HTNB32L lê esta mensagem através da interface UART e armazena internamente em um buffer.

Na sequência, extrai a latitude e longitude contidas na mensagem e coloca na mensagem a ser enviada para os serviços de emergência. Feito isto, a máquina de estados passa para o Envio da mensagem MQTT.

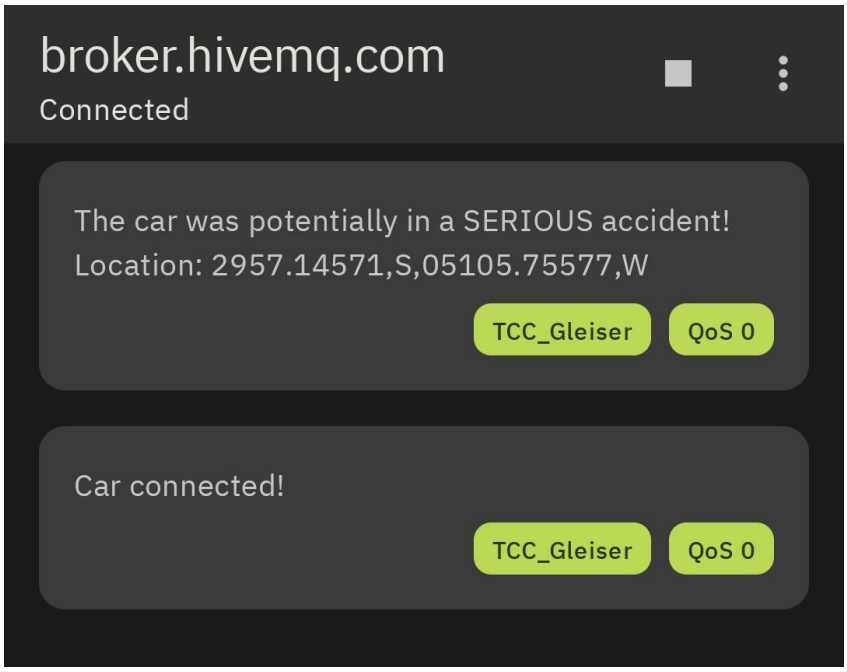
3.2.6 Envio da mensagem MQTT

No contexto deste trabalho, o *broker* MQTT simula uma central de atendimento ao usuário de uma seguradora ou concessionária responsável pela manutenção de uma via. De tal maneira, é possível publicar mensagens em um tópico MQTT, considerando que os serviços da seguradora ou concessionária façam parte de uma cidade inteligente, onde diversas aplicações de Internet das Coisas se comunicam.

A mensagem foi enviada para o tópico *"TCC_Gleiser"*, um tópico criado para desenvolver este trabalho. O conteúdo da mensagem enviada pela aplicação era: "*The car was potentially in a SERIOUS accident! Location: 2957.1457,S,05105.75577,W*".

Para simular a recepção em um serviço de uma seguradora ou concessionária, um *smartphone* foi conectado no mesmo *broker* MQTT e assinou o tópico *"TCC_Gleiser"*. Assim, foi possível receber a mensagem enviada pelo protótipo, ilustrado na Figura 3.4.

Figura 3.4 - *Smartphone* recebendo a mensagem MQTT



Fonte: Própria (2023)

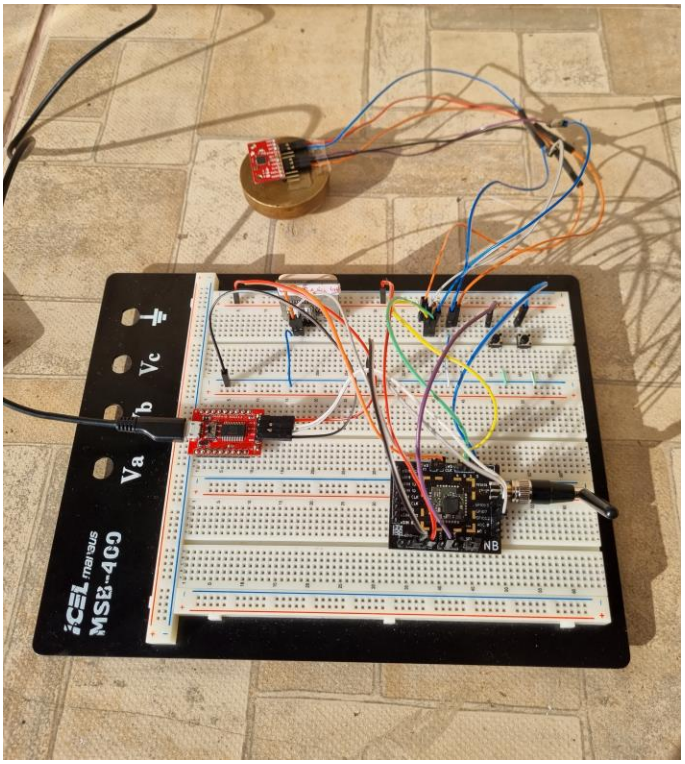
3.3 PROTÓTIPO E AMBIENTE DE DESENVOLVIMENTO DE SOFTWARE

3.3.1 Protótipo

Para desenvolver, testar e validar a aplicação apresentada neste trabalho, foi montado um protótipo com todos os dispositivos mencionados na Seção 2: um SiP HTNB32L, um acelerômetro MPU-6050 e um módulo GNSS u-blox MAX-8Q. Além destes dispositivos, foi utilizado um conversor USB/SERIAL FT232R, que possibilitou a alimentação do sistema, a gravação de *firmware* e o debug da aplicação juntamente de um terminal serial. O protótipo está ilustrado na Figura 3.5.

Foram utilizados uma *protoboard* e fios de para alimentar o circuito e fazer a conexão dos periféricos com o SiP HTNB32L e a conexão do HTNB32L com o conversor USB/SERIAL FT232R, que, por sua vez, foi conectado ao computador com um cabo USB. Além disso, foram conectados dois botões: um botão de *reset* para reiniciar o HTNB32L e um botão que ativa o *bootloader* do SiP, permitindo a gravação dos *firmwares*.

Figura 3.5 - Protótipo



Fonte: Própria (2023)

O conversor USB/SERIAL FT232R foi conectado na interface serial UART1, responsável pela gravação de *firmware* e debug da aplicação. O módulo GNSS u-blox MAX-8Q foi conectado na interface serial UART0, permitindo a comunicação deste periférico. O acelerômetro MPU-6050 foi conectado na interface I2C1, o que viabilizou a configuração e leitura de dados. O botão de *reset* foi conectado no pino NRST e o botão de *bootloader* foi conectado no pino GPIO1.

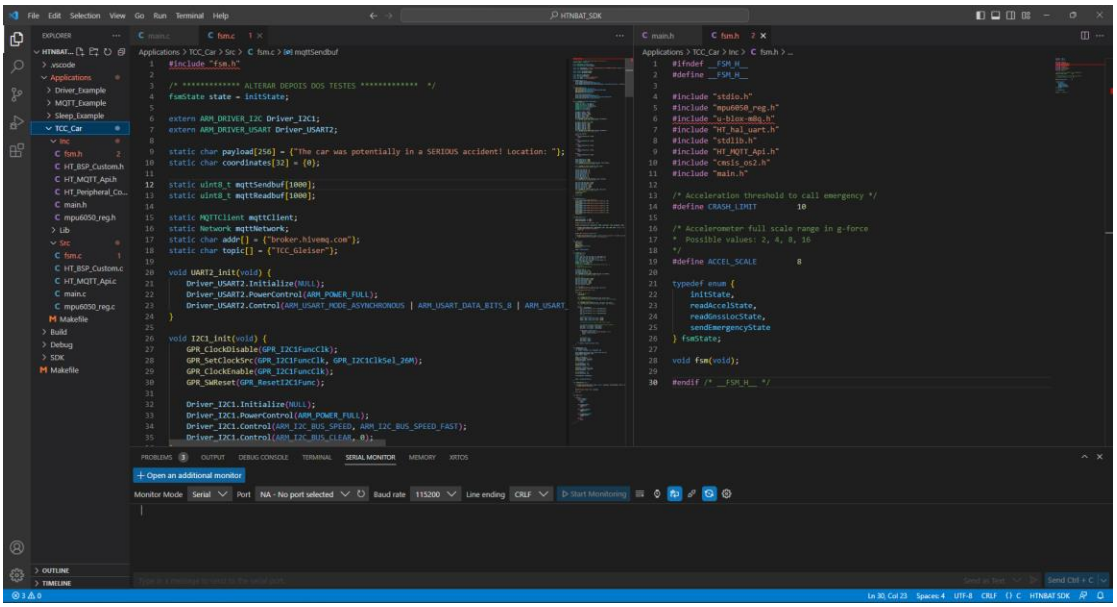
3.3.2 Ambiente de Desenvolvimento de Software

A HT Micron Semicondutores disponibiliza o Ambiente de Desenvolvimento de Software, ilustrado na Figura 3.6, para o desenvolvimento das aplicações através do GitHub oficial da empresa. Esse ambiente possui alguns exemplos de aplicações, que podem ser utilizados como ponto de partida na implementação.

O diretório “SDK” contém todas as funções disponíveis para o funcionamento do sistema, como: sistema de tempo real, HAL, APIs de periféricos e APIs do sistema em geral.

No diretório “Debug” há alguns *scripts* que automatizam o processo de compilação, gravação e *debug* do dispositivo.

Figura 3.6 - Ambiente de Desenvolvimento de Software



Fonte: Própria (2023)

A aplicação foi desenvolvida no diretório *Applications/TCC_Car* do Ambiente. Dentro desse diretório foram criados três subdiretórios:

- Inc: contendo os cabeçalhos dos códigos fonte.
- Src: contendo os códigos fonte da aplicação;
- Lib: contendo a biblioteca estática do microcontrolador;

No diretório Inc, há diversos arquivos de cabeçalho de código fonte, alguns deles disponibilizados pela HT Micron. Entre eles, o *HT_Peripheral_Config.h*, que define os pinos das interfaces de comunicação UART e I2C, utilizadas neste trabalho, além de outras interfaces: SPI e PWM. O arquivo *HT_BSP_Custom.h* contém o protótipo da função de customização de inicialização do HTNB32L. O arquivo *HT_MQTT_Api.h* contém a API com as funções que programam o cliente MQTT.

Os outros cabeçalhos presentes no diretório Inc fazem parte da aplicação desenvolvida para este trabalho. No arquivo *main.h*, há a inclusão de cabeçalhos necessários para a inicialização do sistema, além de algumas macros para controle do registro na rede NB-IoT e tamanho da pilha de memória que a aplicação irá utilizar. O arquivo *fsm.h* contém algumas macros de controle, um tipo *enum* definido *fsmState* que implementa os estados da Máquina de Estados da aplicação e o protótipo da função que define a Máquina de Estados. No arquivo *mpu6050_reg.h* foram definidos os registradores do acelerômetro, utilizados pela aplicação, e os protótipos das funções que fazem a leitura e escrita no dispositivo.

O diretório Lib contém a biblioteca estática disponibilizada pela HT Micron, na qual há o *kernel* do sistema de tempo real, a implementação de periféricos e outras diversas funções responsáveis pelo funcionamento do microcontrolador e *transceiver* QCX212.

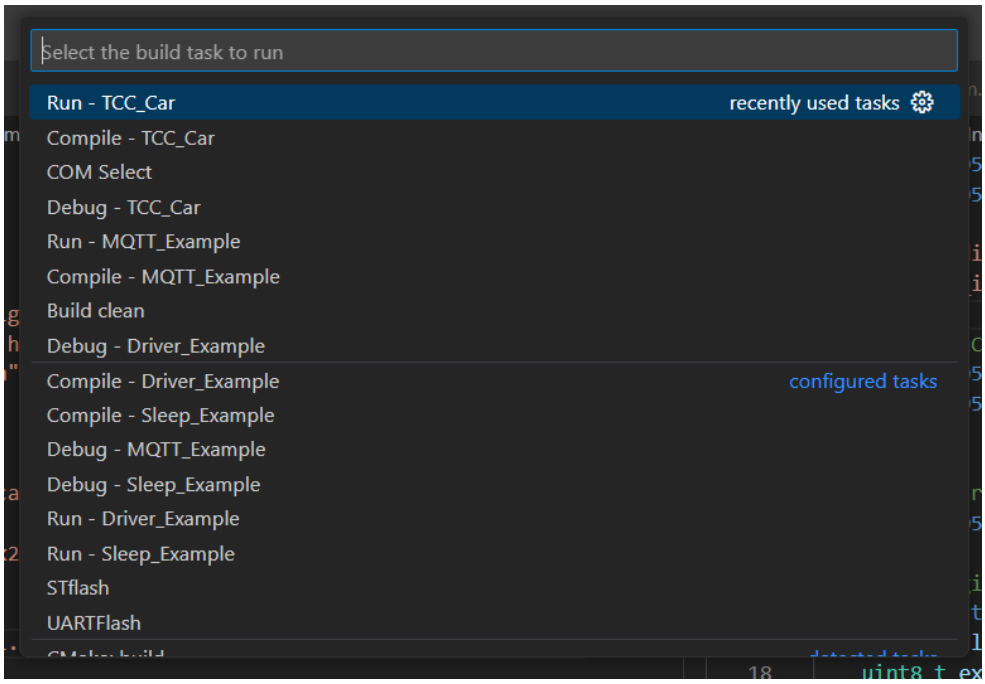
No diretório Src estão os arquivos fontes utilizados pela aplicação. Os arquivos *HT_BSP_Custom.c* e *HT_MQTT_Api.c* implementam as funções mencionadas nos cabeçalhos destes arquivos. Os arquivos *main.c*, *fsm.c* e *mpu6050_reg.c* foram desenvolvidos para este trabalho. O *main.c* contém as funções de inicialização do sistema de tempo real e as funções que controlam o registro na rede NB-IoT. Neste arquivo, acontece a chamada da função *fsm()*, implementada no arquivo *fsm.c*. Esta é a função principal que controla todos os estados da aplicação desenvolvida neste trabalho. O arquivo *mpu6050_reg.c* contém a implementação de todas as funções de escrita e leitura nos registradores do acelerômetro MPU-6050.

3.3.3 Compilação e Gravação da aplicação

Com a aplicação concluída, foi possível compilar os arquivos do diretório *TCC_Car*, convertendo-os em um arquivo binário, o *firmware*, e, na sequência, gravar no HTNB32L do protótipo.

As etapas de compilação e gravação do *firmware* são tarefas automatizadas implementadas pela HT Micron. Estas tarefas são *tasks* do Visual Studio Code, que podem ser chamadas a partir do atalho de teclado *CTRL+SHIFT+B*. Ao executar este atalho, abre-se uma janela com todas as tarefas disponíveis, ilustrado na Figura 3.7.

Figura 3.7 - Tarefas no Visual Studio Code



Fonte: Própria (2023)

A tarefa Run - TCC_Car automatiza o processo de compilação e gravação do *firmware* no SiP. Ao selecioná-la, o compilador cruzado entra em ação e gera o arquivo binário. Na sequência, o software de gravação da HT Micron grava o binário no HTNB32L.

Para ser efetuada a gravação do *firmware* no SiP HTNB32L, este deve ser colocado em modo de *bootloader*. Para isso, pressiona-se o botão que ativa este modo no SiP, na sequência, pressiona-se e solta-se o botão de *reset* e solta-se o botão de *bootloader*. Após este processo, o HTNB32L está em modo de *bootloader*, permitindo a gravação de um novo *firmware*.

4 CONCLUSÃO

A Internet das Coisas (IoT) tem revolucionado a forma como interagimos com o mundo ao nosso redor, permitindo a conexão e comunicação entre dispositivos inteligentes. Neste trabalho, foram exploradas uma série de tecnologias utilizadas no desenvolvimento de aplicações IoT, com foco na rede NB-IoT, o iMCP HTNB32L da HT Micron, o acelerômetro MPU-6050, o GNSS u-blox MAX-8Q e o protocolo MQTT.

Com estas tecnologias, foi possível construir um protótipo e uma aplicação que pode ser embarcada em veículos. Esta aplicação detecta quando o veículo sofre uma colisão frontal e envia uma mensagem para os serviços de emergências disponíveis.

Portanto, pode-se concluir que os objetivos propostos para este trabalho foram alcançados com sucesso, fornecendo uma base sólida para a implementação de soluções eficientes e inovadoras na área de Internet das Coisas.

Para buscar identificar uma colisão, um ensaio de colisão simples foi proposto, com a utilização do acelerômetro MPU-6050. O acelerômetro foi fixado a uma massa metálica e repetidos ensaios, soltando o conjunto massa-acelerômetro de uma determinada altura (20 cm) foram realizados. Ao atingir a bancada de teste, o resultado esperado, de uma variação brusca na aceleração, foi identificado. Utilizou-se diferentes faixas de sensibilidade do sensor, e embora o sensor MPU-6050 não seja o ideal para esta tarefa, foi possível observar uma repetibilidade nos dados coletados e a identificação do impacto.

Como a colisão de automóveis pode gerar sinais com mais de 50Gs, o sensor utilizado e disponível para o trabalho permite identificar apenas colisões leves. De qualquer forma, se houvessem disponíveis outros acelerômetros o processo de coleta de dados e de identificação de colisões seria o mesmo, e, portanto, não considerasse que esta limitação invalide os resultados obtidos, que foram satisfatórios e permitindo identificar o instante em que ocorre uma colisão.

Os dados detalhados do ensaio podem ser vistos no Apêndice A deste trabalho. As curvas coletadas foram obtidas com o auxílio do Arduino IDE, que disponibiliza um plotter digital, ferramenta que visualmente facilita a identificação das colisões.

REFERÊNCIAS

CHEN, Min; MIAO, Yiming; HAO, Yixue; et al. **Narrow Band Internet of Things**. IEEE Access, v. 5, n. 0, p. 20557–20577, 2017. Disponível em: https://ieeexplore.ieee.org/abstract/document/8038776. Acesso em: 7 jun. 2023.

MARTINEZ, Borja; ADELANTADO, Ferran; BARTOLI, Andrea; et al. **Exploring the Performance Boundaries of NB-IoT.** IEEE Internet of Things Journal, v. 6, n. 3, p. 5702–5712, 2019. Disponível em: https://ieeexplore.ieee.org/abstract/document/8666729. Acesso em: 7 jun. 2023.

BEYENE, Yihenew Dagne; JANTTI, Riku; RUTTIK, Kalle; et al. **On the Performance of Narrow-Band Internet of Things (NB-IoT)**. IEEE Xplore. Disponível em: https://ieeexplore.ieee.org/abstract/document/7925809. Acesso em: 7 jun. 2023.

ZHU, Huale; YAN, Wenliang; CHE, Gengjian; LIN, Huipin; HONG, Ming; GAO, Mingyu; et al. **Design and implementation of remote meter reading system based on cloud platform and NB IOT**. Pequim, [2022]. Disponível em: https://www.spiedigitallibrary.org/conference-proceedings-of-spie/12506/1250663/Design-and-implementation-of-remote-meter-reading-system-based-on/10.1117/12.2661989.full?SSO=1. Acesso em: 08 jun. 2023.

HT MICRON SEMICONDUTORES. **iMCP HTNB32L-XXX - HT Micron's NB-IoT SiP**. São Leopoldo, [2023]. Site. Disponível em: https://github.com/htmicron/htnb32l-xxx. Acesso em: 07 jun. 2023

SPARKFUN ELECTRONICS. **SparkFun Triple Axis Accelerometer and Gyro Breakout - MPU-6050**. Niwot, [2023]. Site. Disponível em: https://www.sparkfun.com/products/11028. Acesso em: 07 jun. 2023.

U-BLOX. **MAX-8 series**. Thalwil, [2023]. Site. Disponível em: https://www.u-blox.com/en/product/max-8-series. Acesso em: 08 jun. 2023.

MQTT. **MQTT: The Standard for IoT Messaging**. Site. Disponível em: https://mqtt.org/. Acesso em: 08 jun. 2023

HIVEMQ. **HiveMQ MQTT Broker**: Enterprise ready MQTT broker to move IoT data. Landshut, [2023]. Site. Disponível em: https://www.hivemq.com/hivemq/mqtt-broker/. Acesso em: 08 jun. 2023.

ARDUINO. **Arduino IDE 2.1.0**. Site. Disponível em: https://www.arduino.cc/en/software. Acesso em: 08 jun. 2023.

PUNETHA, Deepak; KUMAR, Deepak; MEHTA, Vartika. **Design and Realization of the Accelerometer based Transportation System**. Índia. 2012.

APÊNDICE A

Testes de colisão realizados com o acelerômetro MPU-6050 afixado em uma massa com objetivo de identificar padrões de colisão. Os testes foram realizados soltando o acelerômetro de uma altura de 20cm de uma bancada. Ao atingir a bancada, o resultado esperado é uma variação brusca na aceleração.

O teste foi dividido em 4 etapas. Em cada etapa o acelerômetro foi configurado com uma sensibilidade diferente. Na etapa 1 foi configurado para detectar variações de ±2g. Na etapa 2, configurado para detectar variações de ±4g. Na etapa 3, de ±8g. E na etapa 4, de ±16g.

Em cada etapa, o teste foi repetido cinco vezes, contendo amostras suficientes para identificar os padrões de colisão. Na sequência, serão apresentados os resultados obtidos a partir da captura dos dados com auxílio do software Arduino IDE, especialmente com a ferramenta Serial Plotter.

Os gráficos disponibilizados na sequência possuem a escala do eixo das ordenadas em m/s (metros por segundo) e no eixo das abcissas, as amostras coletadas do acelerômetro.

A linha azul representa os dados coletados do eixo x do acelerômetro, a linha amarela representa os dados coletados do eixo y do acelerômetro e a linha verde, os dados coletados do eixo z do acelerômetro.

ETAPA 1


COM5
value 1
value 2
value 3
Interpolate
RUN
25
20
15
10
5
0
-5
-10
-15
-20
-25
18265
18277
18289
18301
18314
Type Message
SEND
New Line
115200 baud


COM5
value 1
value 2
value 3
Interpolate
RUN
25
20
15
10
5
0
-5
-10
-15
-20
-25
19293
19305
19317
19329
19342
Type Message
SEND
New Line
115200 baud


COM5
value 1
value 2
value 3
Interpolate
RUN
25
20
15
10
5
0
-5
-10
-15
-20
-25
20612
20624
20636
20648
20661
Type Message
SEND
New Line
115200 baud


COM5
value 1
value 2
value 3
Interpolate
RUN
25
20
15
10
5
0
-5
-10
-15
-20
-25
21867
21879
21891
21903
Economia de bateria
Economia de bateria ativada
Considere a possibilidade de conectar o
Type Message
SEND
New Line

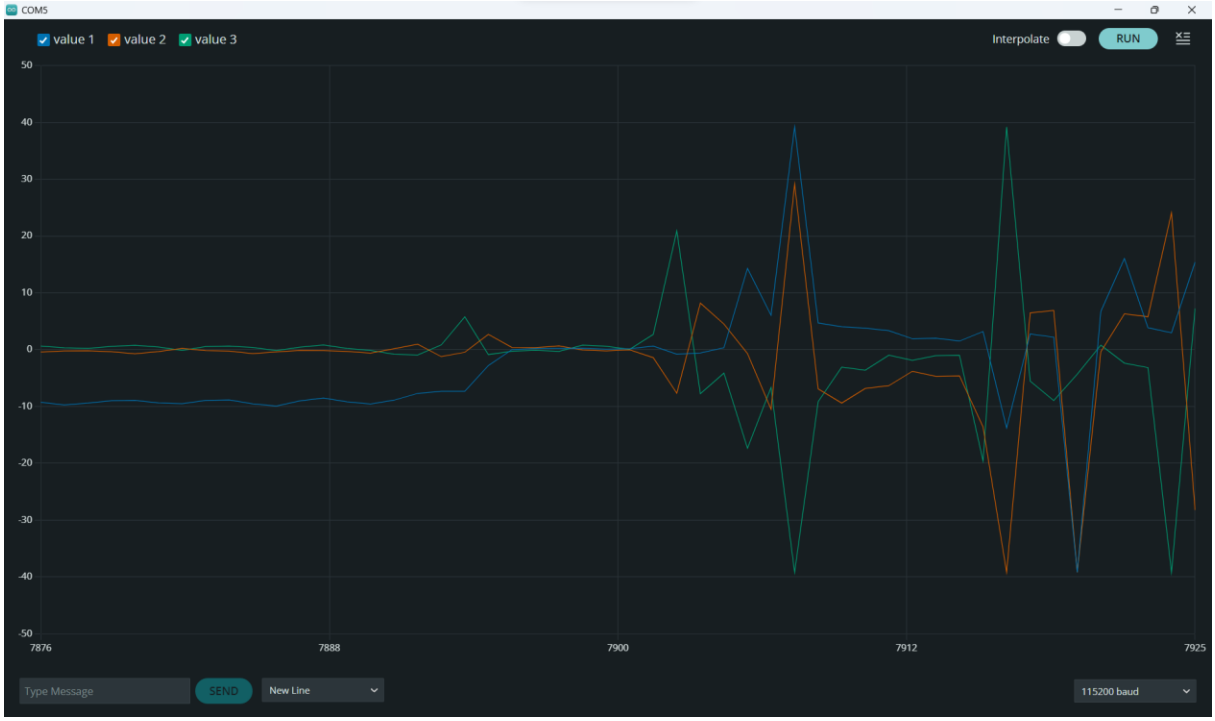

COM5
value 1
value 2
value 3
Interpolate
RUN
25
20
15
10
5
0
-5
-10
-15
-20
-25
22.691
value 1: -3,565
22664
22676
22688
22700
22713
Type Message
SEND
New Line
115200 baud

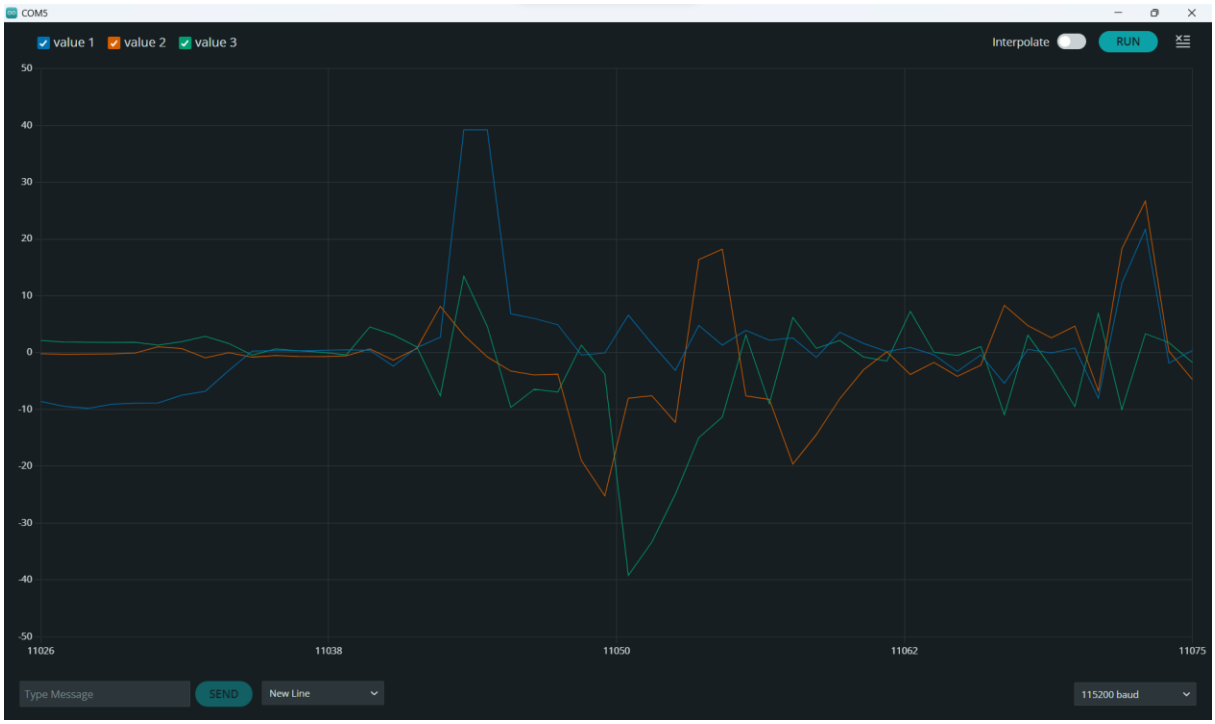
ETAPA 2


COM5
value 1
value 2
value 3
Interpolate
RUN
50
40
30
20
10
0
-10
-20
-30
-40
1848
1860
1872
1884
1897
Type Message
SEND
New Line
115200 baud


COM5
value 1
value 2
value 3
Interpolate
RUN
50
40
30
20
10
0
-10
-20
-30
-40
-50
5018
5030
5042
5054
5067
Type Message
SEND
New Line
115200 baud


COM5
value 1
value 2
value 3
Interpolate
RUN
50
40
30
20
10
0
-10
-20
-30
-40
-50
7048
7060
7072
7084
7097
Type Message
SEND
New Line
115200 baud


COM5
value 1
value 2
value 3
Interpolate
RUN
50
40
30
20
10
0
-10
-20
-30
-40
-50
7876
7888
7900
7912
7925
Type Message
SEND
New Line
115200 baud


COM5
value 1
value 2
value 3
Interpolate
RUN
50
40
30
20
10
0
-10
-20
-30
-40
-50
11026
11038
11050
11062
11075
Type Message
SEND
New Line
115200 baud

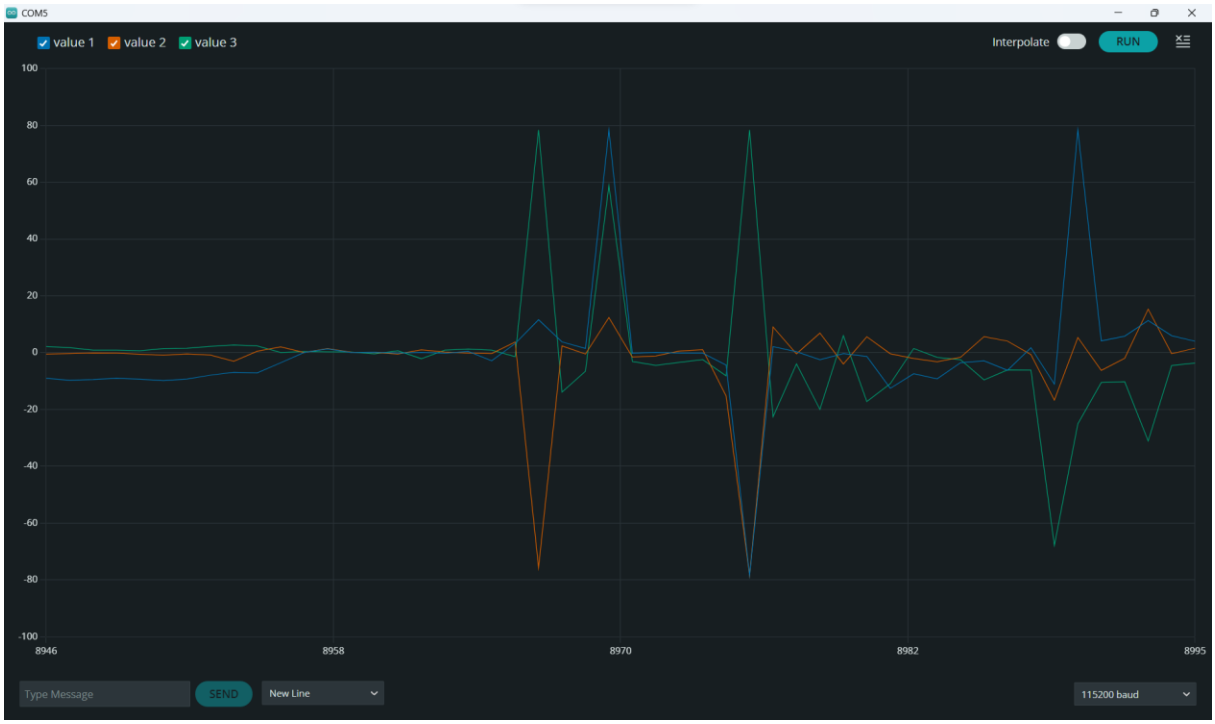
ETAPA 3


COM5
value 1
value 2
value 3
Interpolate
RUN
100
80
60
40
20
0
-20
-40
-60
-80
-100
2542
2554
2566
2578
2591
Type Message
SEND
New Line
115200 baud

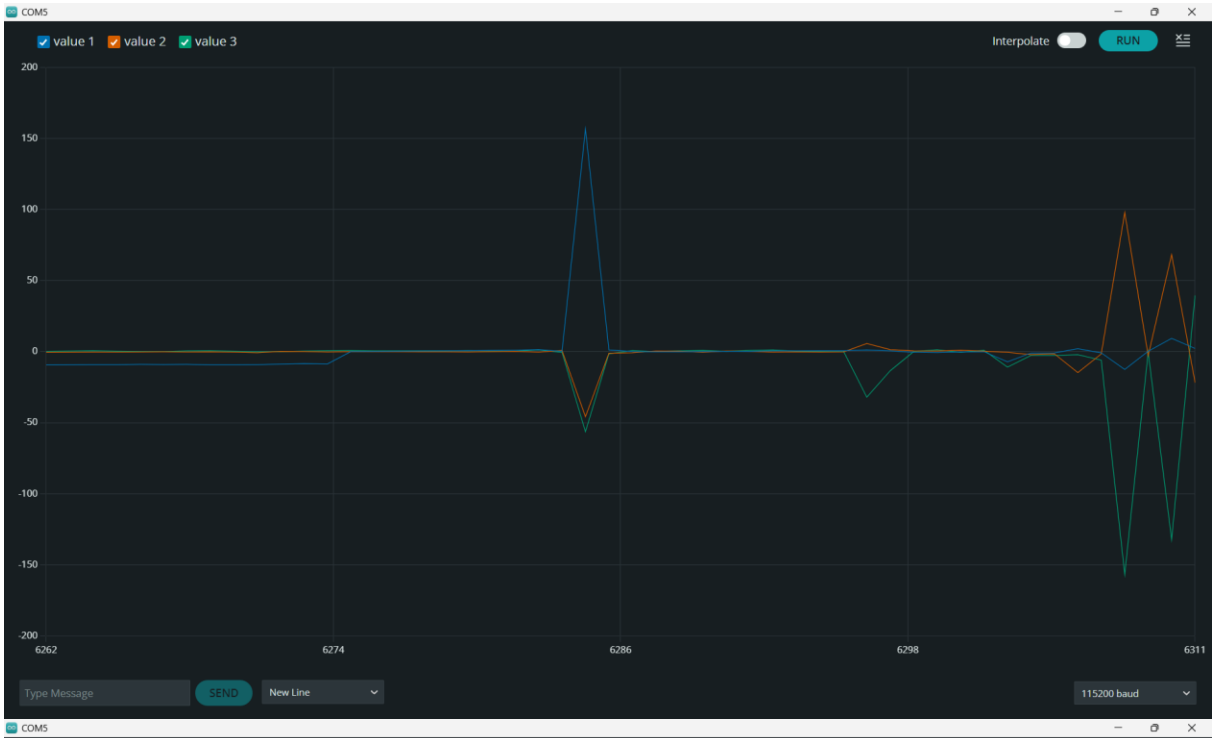

COM5
value 1
value 2
value 3
Interpolate
RUN
100
80
60
40
20
0
-20
-40
4790
4802
4814
4826
4839
Type Message
SEND
New Line
115200 baud


COM5
value 1
value 2
value 3
Interpolate
RUN
100
80
60
40
20
0
-20
-40
-60
-80
-100
7349
7361
7373
7385
7398
Type Message
SEND
New Line
115200 baud


COM5
value 1
value 2
value 3
Interpolate
RUN
100
80
60
40
20
0
-20
-40
-60
7853
7865
7877
7889
7902
Type Message
SEND
New Line
115200 baud


COM5
value 1
value 2
value 3
Interpolate
RUN
100
80
60
40
20
0
-20
-40
-60
-80
-100
8946
8958
8970
8982
8995
Type Message
SEND
New Line
115200 baud

ETAPA 4


COM5
value 1
value 2
value 3
Interpolate
RUN
200
150
100
50
0
-50
-100
-150
-200
6262
6274
6286
6298
6311
Type Message
SEND
New Line
115200 baud


COM5
value 1
value 2
value 3
Interpolate
RUN
100
50
0
-50
-100
-150
-200
11517
11529
11541
11553
11566
Type Message
SEND
New Line
115200 baud


COM5
value 1
value 2
value 3
Interpolate
RUN
50
0
-50
-100
-150
-200
13441
13453
13465
13477
13490
Type Message
SEND
New Line
115200 baud


COM5
value 1
value 2
value 3
Interpolate
RUN
100
50
0
-50
-100
-150
14154
14166
14178
14190
14203
Type Message
SEND
New Line
115200 baud