Projeto de geolocalização em ambientes internos

Daniel O. de Almeida[1]
Ian C. S. Nascimento[2]

[1]Instituição/Empresa Senai Cimatec, Email: daniel.oa95@gmail.com;
[2]Instituição/Empresa Senai Cimatec, Email: iannascimento@hotmail.com;

Resumo

Este trabalho apresenta um projeto envolvendo módulos de conectividade sem fio *Xbee S2C* configurados para monitorar objetos ou pessoas em ambientes internos a partir de uma rede de sensores.Este projeto fornecer alternativas tecnológicas para atividades de geolocalização realizadas em ambientes demarcados, registrando a posição, através da leitura da intensidade de sinal *RSSI* em uma rede de sensores, a fim de sustentar as melhores práticas produtivas.
Palavras-chave: Rede; Xbee ; Geolocalização.

Abstract

This work presents a project involving Xbee S2C wireless connectivity modules configured to monitor objects or people indoors from a network of sensors. This project provides technological alternatives for exploration activities in demarcated environments, recording the position, by reading of the RSSI signal strength in a sensor network, in order to support the best productive practices.

Keywords: Network; Xbee; Geolocation.

1. Introdução

O primeiro sistema de posicionamento espacial foi desenvolvido exclusivamente para fins militares em meados do século XX. O sistema norte americano de posicionamento global (GPS) abriu portas para o uso desta tecnologia com a operação de satélites geoestacionários em diversas aplicações a partir da década de 80. Ao longo dos anos, outros sistemas foram desenvolvidos e também estão em operação, como é o caso do GLONASS (sistema Russo) ou em desenvolvimento, como o sistema Europeu GALILEO e o sistema Chinês COMPASS.Aplicações diversas nos meios de transporte, como no deslocamento de automóveis, trens, aeronaves e navios, no planejamento de rotas, contribuindo com aumento de velocidade nos deslocamentos, otimização energética e aumento de segurança aos usuários, incluindo serviços de rastreabilidade; agricultura de precisão, na identificação precisa de objetos em campo, em georastreabilidade de produtos agrícolas, no cadastro rural, demarcação; em meio ambiente com monitoramento ambiental, levantamento e proteção de recursos naturais, coleta de pontos amostrais georreferenciados etc.(EMBRAPA TERRITORIAL,2018)

Em geral, o princípio de funcionamento do sistema de GPS está baseado em medir o tempo que cada sinal emitido pelo satélite demora em encontrar a antena do receptor. Os satélites encontram-se em órbita geoestacionária, em que acompanham a rotação da terra. Os receptores são dispositivos eletrônicos de uso geral que estão na superfície da terra. A distância entre o satélite e o receptor na Terra é calculada com base na velocidade de propagação do sinal no vácuo (300.000 km/s) e no tempo de propagação do sinal (que é medido por um relógio atômico). Para determinar a posição do receptor na Terra se faz necessário conhecer a distância de no mínimo três satélites distintos, que por meio da trilateração e de outras técnicas de refinamento matemático, permitem apontar uma posição no espaço. Com o sinal de 1 (um) satélite em órbita, é possível realizar a leitura de um ponto em uma superfície esférica. Tendo os sinais de 2 (dois) satélites em órbita trabalhando em sintonia, é possível realizar a leitura de um ponto em uma superfície plana (circunferência). Por fim, com os sinais de 3(três) ou mais satélites em órbita trabalhando em sintonia, é possível realizar a leitura de um ponto no espaço(interseção dos sinais). A figura 1(um) ilustra o sistema de trilateração via satélites do GPS.

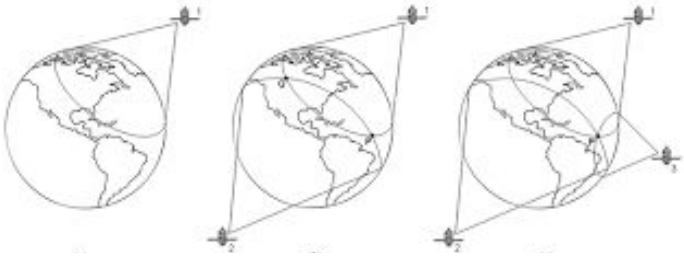

Figura 1: Sistema de trilateração via satélites [1]

Os sistemas de localização global como GPS e o GLONASS possuem alta eficiência em ambiente externo (*outdoor*). No entanto, devido a obstáculos naturais e fenômenos de interferências, esses sistemas quando aplicados em ambientes confinados (edificações, galpões, construções subterrâneas etc) não possuem a mesma performance. Por conta disto, foram desenvolvidos sistemas específicos para ambientes internos (*indoor*).Os sistemas de rastreamento em tempo real (RTLs) são utilizados para monitorar objetos e pessoas em um edifício ou uma área confinada, com destaque para o beacon.

O beacon é um hardware que utiliza a tecnologia BLE - *Bluetooth low energy* - com base em protocolos de baixo consumo energético, exemplo: wifi, *zigbee*, *bluetooth* .Estes dispositivos podem ser configurados e programados para receber sinais de potência, estabelecer uma rede de comunicação e transmitir *frames* de informações. Esta tecnologia é considerada promissora por atender diversas atividades de mercado, por ser uma alternativa para questões que envolvam ambientes confinados e custo acessível.

1.1 Objetivo do projeto

O projeto tem por objetivo ser capaz de monitorar o trajeto de pessoas ou objetos, a fim de determinar sua posição em um ambiente confinado utilizando o padrão de comunicação ZigBee entre módulos de comunicação Xbee S2C sem fio.O projeto almeja apresentar um sistema de monitoramento informando o histórico de posição do dispositivo ao longo do tempo.

O desenvolvimento deste protótipo permite somar esforços com as estratégias modernas de controle/produção fornecendo monitoramento em tempo real com baixo consumo de energia, segurança da informação e custo acessível.Setores industriais são sensíveis a demandas otimizadas de mercado, eficiência energética e custo operacional; exemplo disto é a área de misturação de uma fábrica pneus que é parte primária em uma linha de produção, normalmente automatizada.

Em um cenário industrial, especificamente na fábrica de pneus da Continental, localizada no município de Camaçari/BA, o processo produtivo exige a harmonia de setores complementares da fábrica.Um deste setores é a misturação, área que fornece a borracha processada em mantas para as máquinas de preparação do pneu, tais como: Extrusoras, Calandras e máquinas de talão.Neste setor, a dinâmica do processo é executado por empilhadeiras livres (transportadores) que têm por objetivo movimentar a borracha em forma de manta para as máquinas do processo produtivo (Máquinas finais).Se tratando de alta produtividade, o monitoramento dos transportadores pode auxiliar na performance do setor através do registro da sua

posição, monitorando a quantidade de trabalho diário que a empilhadeira consegue fazer buscando a matéria-prima nos elevadores e/ou na área de armazenamento, e disponibilizando na máquina.O cenário conta com diversas empilhadeiras operando ao mesmo tempo, inclusive em outros setores para além do cenário descrito nesta leitura.

Em linhas gerais, a monitoria deste setor destacado possibilita planejar novas estratégias de produção, promover revezamento de transportadores, gerenciar paradas de manutenção, otimizar a produtividade e gerenciar os recursos humanos empregados nesta tarefa.

2 . REFERENCIAL TEÓRICO

2.1 Observações

O sistema GPS fornece a localização de um objeto por meio de sinais eletromagnéticos. Evidentemente a operação dessa grandeza vetorial é classificada com respeito a sua natureza, direção e energia de propagação.Isso indica que primeiro: a propagação de um sinal eletromagnético sofre distorções de acordo com a interação com outros sinais eletromagnéticos e com as propriedades do meio(reflexão, refração, absorção, difração, interferência, espalhamento e polarização); segundo: a determinação de um ponto no espaço é resultado de análise matemática vetorial robusta, tornando esta aplicação complexa e de alto custo de implementação.

Os sistemas *indoor* são alternativas eficientes para fornecer a localização interna de um objeto também por meio de sinais eletromagnéticos. A relevância desta tecnologia está em utilizar um método diferente, que tem base na transmissão e recepção de dados buscando o sinal de potência dos dispositivos em rede. Esta trama formada permite minimizar distorções dos obstáculos presentes no ambiente, principalmente aqueles que possuem valência magnética e afetam negativamente a magnitude do sinal, tornando acessível e dentro da realidade financeira.

2.2 Protocolo Zigbee

A ZigBee Alliance é uma associação constituída por várias empresas de diferentes segmentos, entre elas a Motorola, Philips, Samsung, Siemens, Analog Devices, Texas Instruments. O objetivo desta aliança é desenvolver produtos e padrões *wireless* que sejam de baixo custo, baixo consumo, seguros e confiáveis.Partindo deste pressuposto desenvolveram o protocolo ZigBee junto do IEEE. (COUTO,2013)

O ZigBee é um *standard* que define um conjunto de protocolos de comunicação para dispositivos, com baixas taxas de transmissão de dados, sem fios. Os dispositivos ZigBee podem trabalhar em três bandas de frequência: 868 MHz , 915 MHz e 2,4 GHz. Na maioria das vezes é escolhida a frequência de 2,4 GHz, pois é a que permite uma taxa de transmissão mais elevada e também fornece um maior número de canais.O protocolo ZigBee é o futuro das redes de sensores sem fios, pois possui uma valência muito importante nos dias de hoje; baixo consumo energético. (COUTO,2013)

A figura 2(dois) ilustra alguns modelos de módulos zigbee xBee que são utilizados para fins de comunicação em rede.

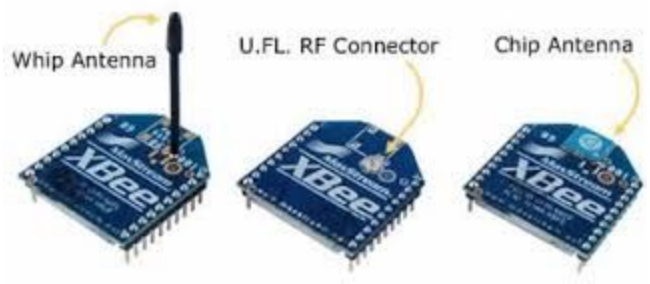

Figura 3: Principais modelos de módulos Xbee [2]

No modo transparente a interação entre XBee e Microcontrolador/Computador é feita de forma direta.Este modo de funcionamento é simples, uma vez que os dispositivos transmitem diretamente para a porta serial toda a informação recebida. Os dados transmitidos não sofrem alteração em relação à sua forma original porém, oferecem um baixo nível de robustez sendo um método ineficaz de se comunicar com computadores. Neste modo as configurações são realizadas através de comandos AT. Uma trama em modo API começa sempre pelo carácter 0x7E (*start delimiter*). Qualquer informação recebida antes desta referência de início é ignorada. Os dois bytes seguintes (*length*) permitem saber qual o tamanho dos dados enviados. Os dados estão contidos na API-*Specific Structure*. O último byte (Checksum) permite confirmar se os dados são corretamente recebidos pelo módulo de destino. O API-*Specific Structure*, campo de dados, é dividido em duas partes, cmdID e o *cmd Data.* O campo *cmdID* permite identificar o tipo de trama (*Modem Status, AT Command, ZigBee Transmit Request, ZigBee Transmit Status* ou *ZigBee Received Packet*). O campo *cmd*

Data contém toda a informação que se pretende enviar e a sua estrutura pode variar consoante o tipo de trama. (COUTO,2013)

A figura 3(três) ilustra a operação de um frame API do tipo *Transmit Request*, utilizado para enviar mensagens estruturadas.

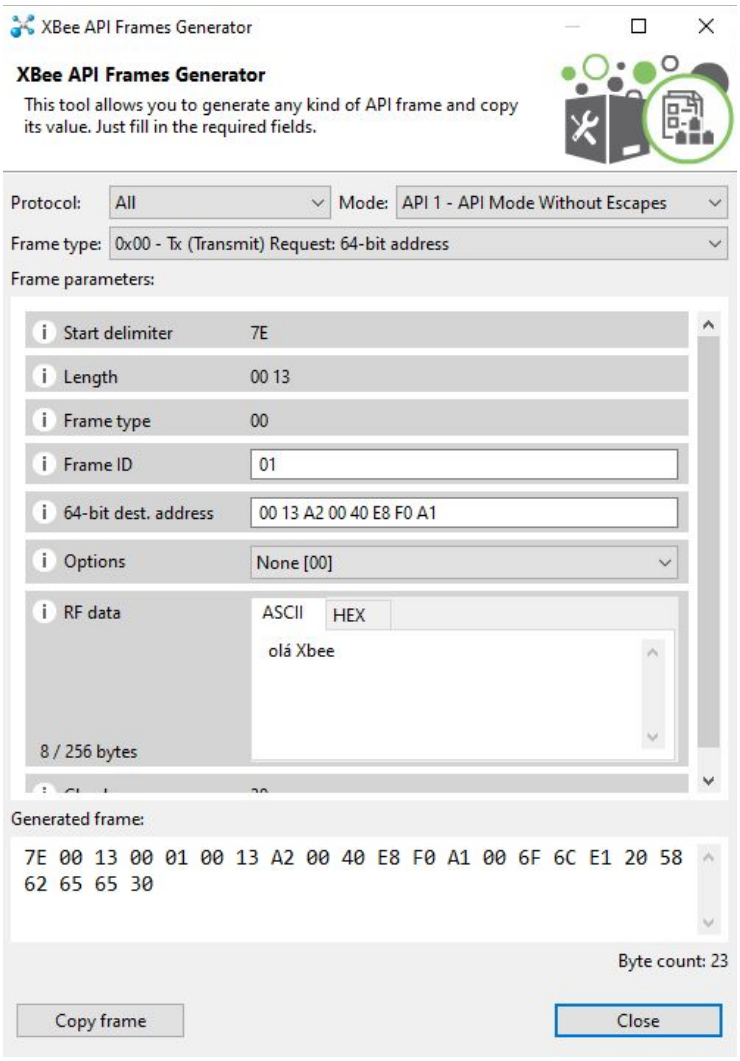

Figura 3: Gerador de frames XCTU

3. Metodologia

Neste trabalho foi elaborado uma solução com base em algoritmos, linguagem de programação python, módulos xBee S2C, sistema gerenciador de banco de dados *postgresql* e interface gráfica, a fim de validar as pretensões de aplicação da tecnologia em destaque e fornecer um protótipo. Para isto, foi utilizado o XCTU que é um aplicativo multiplataforma gratuito, fabricado pela digi internacional, que fornece aos desenvolvedores interação com os módulos Digi RF por meio de uma interface gráfica de fácil acesso. Os primeiros passos foram dados com a configuração da rede(*pan id*), direcionamento de mensagens(*broadcast* ou endereçado), função na rede (roteador ou coordenador), modo de operação (API- *application programming interface* ou AT- *transparent application)* e testes de alcance/qualidade do sinal. A figura 4(quatro) ilustra a narrativa de configuração nesta plataforma descrita.

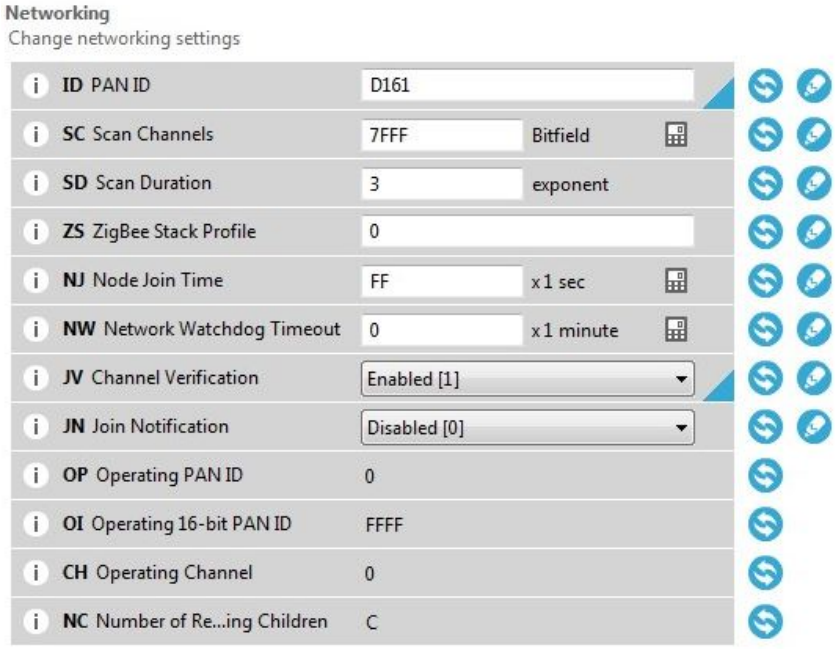

Figura 4: Configuração de xbee via xctu[3]

Para o devido funcionamento do circuito utilizando a plataforma STM Núcleo precisou-se modificar e adequar o código escrito preliminarmente na plataforma ARDUINO para a utilizada, tarefa executada com apoio do Orientador. A geolocalização acontece por meio de triangulação na qual dois dispositivos encontram-se posicionados nas extremidades do espaço em questão, numa mesma linha vertical ou horizontal, enquanto o terceiro dispositivo é acoplado ao objeto móvel que deseja-se monitorar. Através do comando AT AS(*Active Scan*) do protocolo Zigbee

é possível determinar a potência de sinal entre as antenas dos dispositivos ligados na trama, o valor de potência em RSSI (*Received Signal Strength Indicator*) é convertido aproximadamente em metros através da fórmula:

$$distância = 10^{\frac{RSSI_{1m} - RSSI_{Atual}}{10 \times cte_{meio}}}$$

Equação 1: Distância Vs. Potência de sinal

Sendo assim, obtêm-se dois valores de RSSI, do móvel para ambos os beacons fixos, o valor de distância entre os beacons fixos é conhecido. Com os três valores de distância forma-se um triângulo e pode-se finalmente encontrar o ângulo formado entre os beacons fixos e o dispositivo móvel através de uma manipulação na equação de Lei dos Cossenos:

$$ângulo = acos(\frac{distancia_{mb1}{}^2 - distancia_{mb2}{}^2 - distancia_{bb}{}^2}{-2 \times distancia_{bb} \times distancia_{mb2}})$$

Equação 2: Manipulação da lei dos cossenos.

A partir deste valor de ângulo e a distância do dispositivo móvel para uma das antenas fixas aproxima-se o modelo em questão para um triângulo retângulo para encontrar os valores de posição X e Y em relação a antena considerada na posição: (0,0), conforme o esquema a seguir:

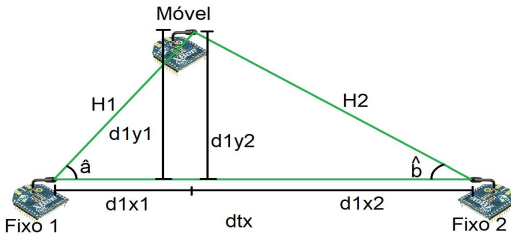

Figura 5: Esquema de funcionamento e cálculos da movimentação

$$d1y1 = \frac{H1}{sen(\hat{a})}$$

Equação 3: relação de triângulo retângulo para encontrar distancia Y

$$d1x1 = \frac{H1}{cos(\hat{a})}$$

Equação 4: relação de triângulo retângulo para encontrar distância X

Os cálculos são feitos num esquema de redundância para garantir que a leitura de uma antena para outra não esteja tão defasada ou problemática entre os pontos fixos e o móvel, ou seja, as mesmas equações são aplicadas para ambos os ângulos para verificar e garantir a igualdade entre as variáveis d1y1 e d1y2 e que a soma de d1x1 e d1x2 seja igual a distância entre os beacons fixos 1 e fixo 2. Os valores de posição X e Y são enviados através do protocolo (0x10) *Transmit Request* do módulo ZigBee conectado ao STM para o outro módulo posicionado no computador de supervisão. O fluxograma da figura 6(seis) ilustra a dinâmica do algoritmo obtenção de potência de sinal e cálculo de distância:

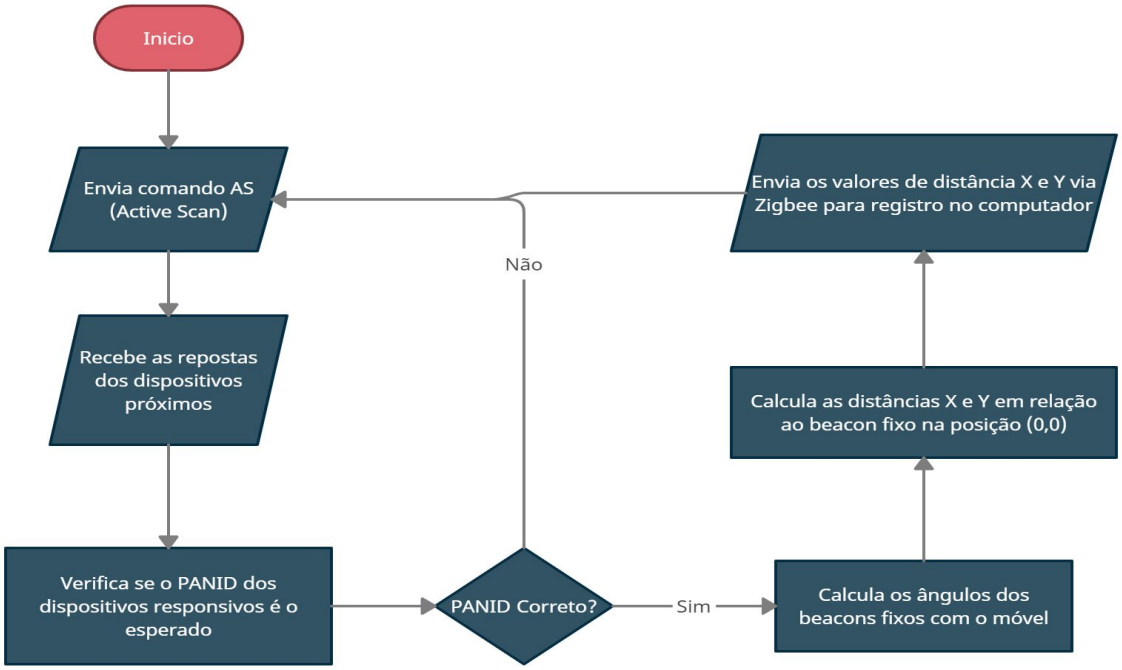

Figura 6: Fluxograma do código do STM

Em sequência foi utilizado o software *visual studio code* como plataforma para desenvolver o algoritmo responsável pela leitura da porta serial, o algoritmo da arquitetura do banco de dados e as configurações com o sistema gerenciador de banco de dados(Postgresql*),* tudo suportado em linguagem python(Apêndices A,B,C). Em resumo, o algoritmo de leitura da serial foi responsável por traduzir as informações API

suportadas pelo dispositivo coordenador da rede, manipulando o frame recebido e comunicando ao banco de dados o registro das informações de interesse (neste caso foi a posição espacial em coordenadas cartesianas ‘x ‘e ‘y’). Este algoritmo de leitura permite também o registro da identificação do transportador, período de trabalho e a data atual como forma de monitoria. O fluxograma da figura 7(sete) ilustra a dinâmica deste algoritmo de leitura da serial.

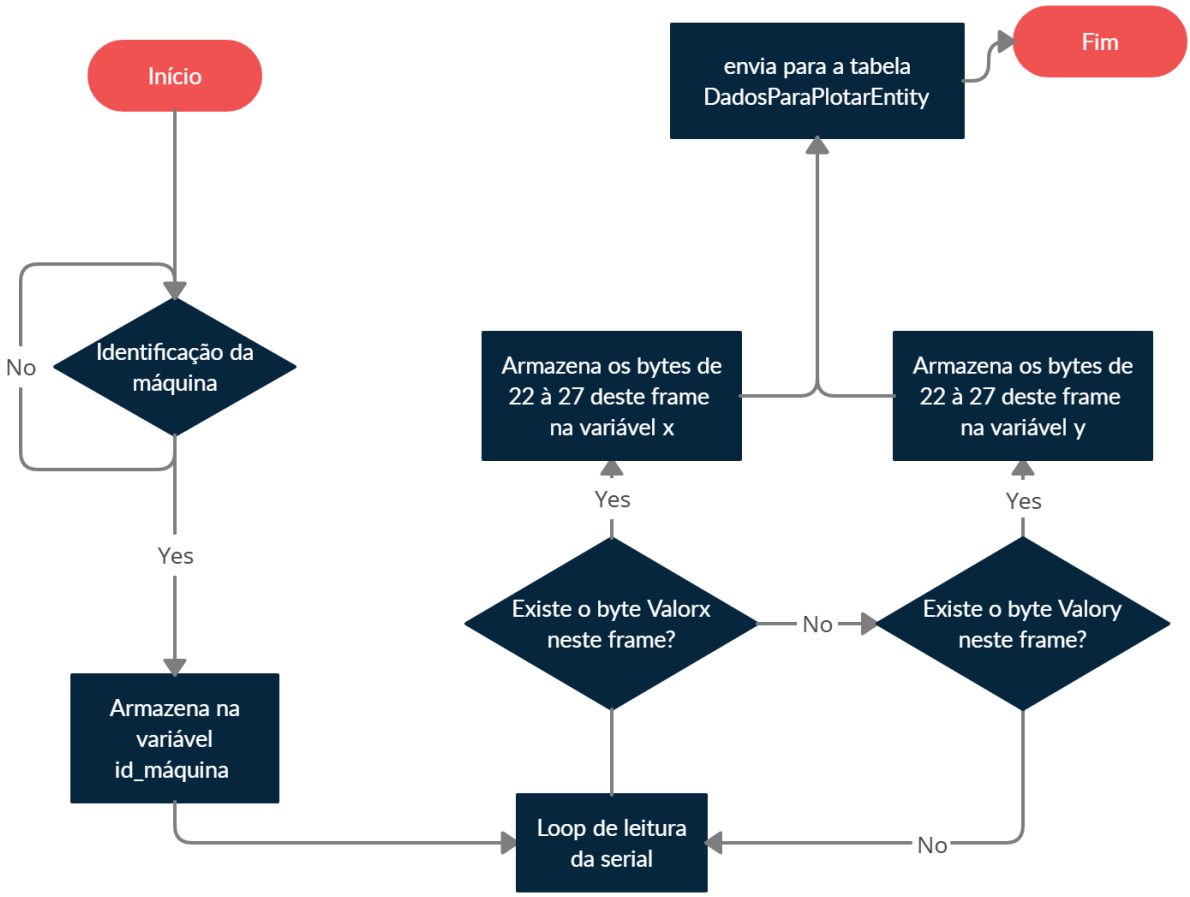

Figura 7: Fluxograma de leitura da Serial

4. Resultados e Discussões

O cenário de testes escolhido para desenvolver o protótipo foi uma residência localizada em Salvador/BA contendo: dois quartos, um banheiro, uma cozinha e uma sala.Os testes consistiram em localizar, por um tempo de 10 minutos, o dispositivo móvel em coordenadas demarcadas na residência que representassem respectivamente: banheiro, cozinha e quarto.A figura 8(oito) ilustra a planta da residência.

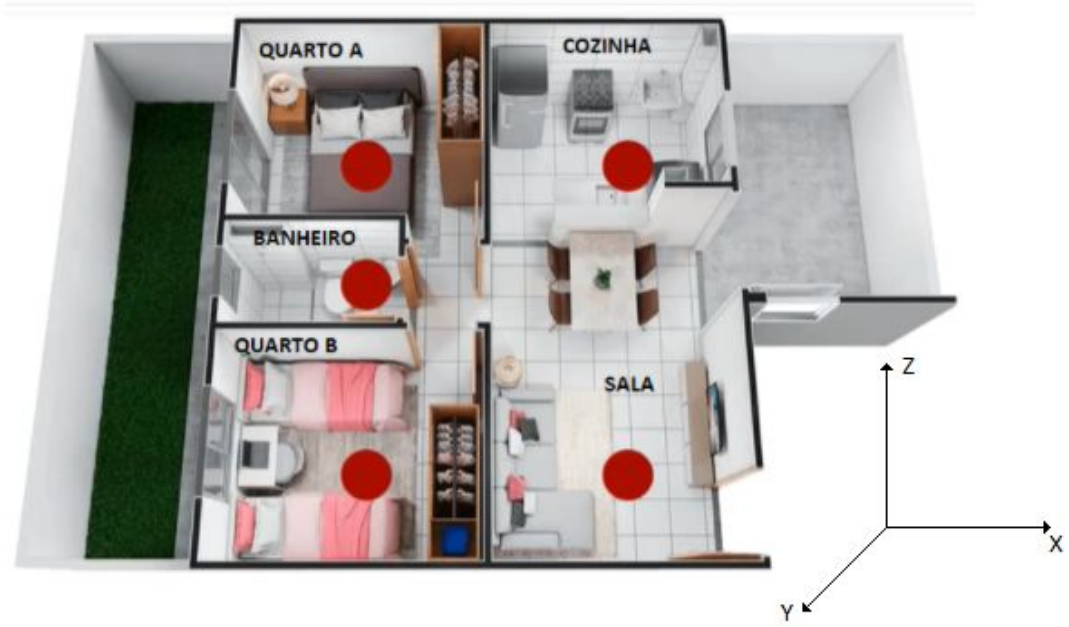

Figura 8: planta residencial

A localização dos pontos destacados foram mensurados através de trena a laser a fim de mapear as coordenadas reais que configuram as localizações do banheiro, cozinha e quarto “A”.O módulo para conexão com computador foi posicionado no quarto “A”.Os módulos fixos separados por uma distância de 1(um) metro foram posicionados também no quarto “A”.Os valores medidos são apresentados na tabela ..

PONTOS MEDIDOS	BANHEIRO	COZINHA	QUARTO A
X	560 mm	1840 mm	560 mm
Y	1250 mm	750 mm	750 mm

Tabela 1: coordenadas da residência

Utilizando a técnica de triangulação explicitada neste trabalho, foi possível calcular os valores estimados das coordenadas do dispositivo móvel, em relação ao dispositivo fixo 1.Os dados registrados são apresentados na tabela 2 através de filtragem média móvel realizado no algoritmo, e as figura 9(nove) e 10(dez) apresentam o comportamento das estimativas de posição.

PONTOS ESTIMADOS	BANHEIRO	COZINHA	QUARTO A
X	342(-218) mm	1530(-310) mm	345(-215) mm
Y	986(-264) mm	701(-49) mm	727(-23) mm

Tabela 2: coordenadas estimadas

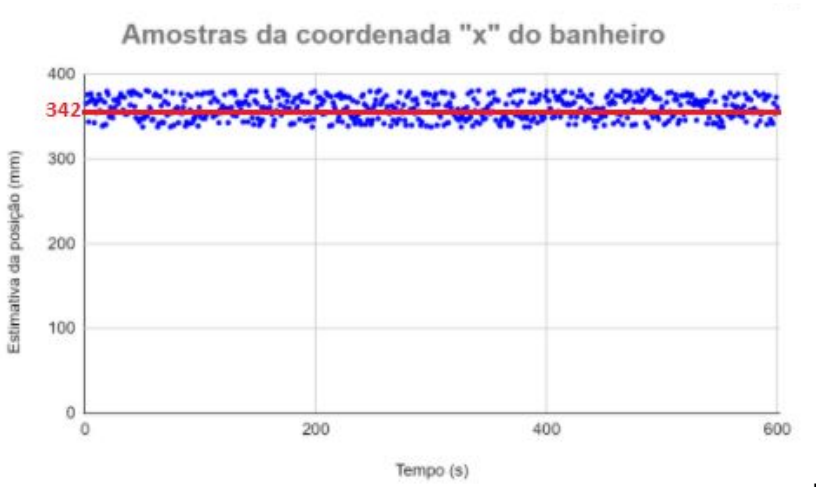

Figura 9: Amostras da coordenada x do banheiro

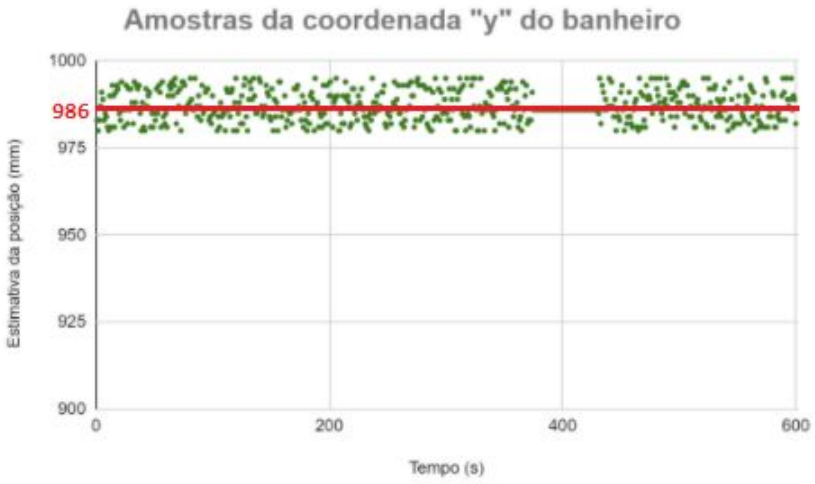

Figura 10: Amostras da coordenada y do banheiro

Os gráficos das amostras de coordenadas possuem característica semelhante com alternância de valores devido à instabilidade de comunicação e efeitos de distorção do sinal.Foi possível perceber que o sinal possui estabilidade parcial devido ao trajeto e a disposição dos obstáculos na residência, e em alguns momentos houve a estabilidade total de uma das coordenadas.

5. Considerações finais

O objetivo deste trabalho foi a implementação de um sistema de posicionamento em ambientes internos utilizando uma rede de sensores de baixo consumo, através do protocolo de comunicação ZigBee.A técnica de estimativa foi construída com base no conceito de triangulação de sinais de potência entre os dispositivos fixos e o dispositivo móvel.

O cenário de testes foi uma residência e o projeto contou com a configuração de dois dispositivos fixos xBee S2C PRO alimentados por bateria; um dispositivo móvel xBee S2C PRO acoplado ao microcontrolador STM32L476RGT6U e um dispositivo xBee S2C PRO acoplado ao módulo serial para conexão com desktop.Em paralelo foi construído um sistema de gerenciamento em banco de dados suportado pela plataforma *Postgresql* com intuito de registrar os dados de posição calculados pelo algoritmo embarcado no microcontrolador.A exposição das informações de posicionamento estimada foram apresentadas em supervisório *web server*.

Através das análises dos gráficos e os resultados das medidas estimadas, observou-se que quanto menor o número de dispositivos fixos no cenário, maior é o erro em comparação com a posição medida.Isso se deve ao fato de existir interferência de sinal à medida que o dispositivo móvel ultrapassa pelos obstáculos físicos.A principal dificuldade encontrada está em compreender o funcionamento da comunicação no protocolo *ZigBee* e a tradução do frame *transmit request*.Outros pontos como a configuração do banco de dados e a construção do sistema *web server* também determinam perícia sobre o assunto.

De modo geral o objetivo macro foi alcançado e o projeto apresentou estimativa de posição muito próxima da realidade medida, porém ainda há muitos fatores que podem ser investigados.O primeiro deles trata da expansão da trama com a inclusão de mais dispositivos fixos para minimizar o erro na leitura dos pontos; o segundo trata da implementação de outros métodos para determinar a posição, como técnicas de classificação estatísticas ou métodos de classificação baseado em inteligência artificial; por fim desenvolver um sistema supervisório que aponte a trajetória do móvel no ambiente ou desenvolver um sistema mobile que otimize as tarefas de produtividade em ambiente industrial/comercial tende a agregar valor ao projeto.

6. Referências

[1] DIAS, N. W. Processo de trilateração de satélites - Introdução ao GPS. UNITAU. Taubaté.

[2] COUTO, L. M. R. Sistema de localização indoor com base no protocolo ZIGBEE. ISEP - Instituto Superior de Engenharia do Porto. [S.l.], 2013.

[3] EMBRAPA TERRITORIAL. Satélites de Monitoramento. Campinas, 2018. Disponível em: < https://www.embrapa.br/satelites-de-monitoramento >. Acesso em: 17 dez. 2020

[4] SISTEMA DE POSICIONAMENTO INDOOR UTILIZANDO UMA REDE DE SENSORES SEM FIOS BASEADA NO PROTOCOLO DE COMUNICAÇÃO ZIGBEE. Orientador: Prof. Dr. Natanael Rodrigues Gomes. 2018. Trabalho de Conclusão de Curso (Graduação em Engenharia Elétrica) - Universidade Federal de Santa Maria (UFSM), Santa Maria, RS, 2018.

[5] BRÁS, Luís – Desenvolvimento de Sistema de Localização Indoor de Baixo Consumo. Dissertação de Mestrado em Engenharia Electrónica e Telecomunicações orientada pelo professor José Fonseca e apresentada na Universidade de Aveiro, em 2010.

[6] G. N. Sol Teixeira, L. M. Rodríguez Peralta, “Comparing ZigBee, Bluetooth, UWB,and Wi-Fi”, in Encyclopedia of Networked and Virtual Organizations, Idea Groups,Vol. I, pp. 288 - 296, ISBN: 978-1-59904-885-7, 2008.

[7] TADAKAMADLA, Shashank – Indoor Local Positioning System For ZigBee, Based On RSSI. Dissertação de Mestrado em Ciência da Engenharia Eléctrica orientada pelo Prof.Bengt Oelmann e apresentada na Mid Sweden University, em 2006.

[8] FERNANDES, Maria – Redes de Sensores para uso Humano. Dissertação de Mestrado em Engenharia Electrónica e Telecomunicações orientada pelo Dr.Nuno Borges de Carvalho e apresentada na Universidade de Aveiro, em 2010.

[9]Principais modelos de módulos xBee disponível em: <https://www.google.com/imgres?imgurl=http%3A%2F%2Fnewtechonline.com.br%2Fwp-content%2Fuploads%2F2016%2F02%2FDigi-XBee.jpg&imgrefurl=http%3A%2F%2Fnewtechonline.com.br%2Fproduto%2Fxbee-zigbee%2F&tbnid=z9giEB_3Z8OU_M&vet=12ahUKEwi8mLrEidrsAhUhA7kGHZb3AkUQMyguegUIARCVAg..i&docid=3SF9KGs7NM_-_M&w=3888&h=2592&q=zigbee%20xbee%20modelos&ved=2ahUKEwi8mLrEidrsAhUhA7kGHZb3AkUQMyguegUIARCVAg>

[10]Configuração do xbee disponível em: <https://www.digi.com/resources/documentation/digidocs/90001458-13/default.htm#reference/r_fw_settings.htm%3FTocPath%3DConfigure%2520your%2520modules%7CConfiguration%2520working%2520mode%7C_____3>

[11] HIGHTOWER, Jeffrey; BORRIELLO, Gaetano - Location Sensing Techniques.Computer Science and Engineering, University of Washington, Seattle, 2001.

[12] JUNIOR, Valmir - sistema de localização para ambientes fechados baseado na potência do sinal recebido em rede ZigBee, Instituto Federal do Espírito Santo;Vitória, 2011.

[13] PRIWGHARM, Ratana; SRIVILAS, Kiri; CHERNTANOMWONG, Panarat -Indoor Localization System using RSSI Measurement in Wireless Sensor Network Based on ZigBee Standard. Faculty of Engineering King Mongkut's Institute of Technology Ladkrabang, 2010.

[14] LIN, Tsung-Nan, LIN, Po-Chiang - Performance Comparison of Indoor Positioning Techniques based on Location Fingerprinting in Wireless Networks. Institute of Communication Engineering National Taiwan University, International Conference on Wireless Networks, Communications and Mobile Computing,2005.

[15] XBee & XBee-PRO OEM RF Module Antenna Considerations. Digi Application Note, 2012.

Apêndice A - Algoritmo leitor da porta serial

```
#!/usr/bin/python2.7
import serial
import time
import DbContext as db
from DadosParaPlotarEntity import DadosParaPlotar as Maquina
import datetime



db.DbContext.create_all_tables()
posicao_x = 0
posicao_y = 0
ser = serial.Serial('COM3')
x = 0
y = 0
VALUE_SERIAL = 0

id_maquina = input('Digite a identificacao da maquina: ')


while 1==1:
    VALUE_SERIAL = ser.readline(29)

    if b'valorx' in VALUE_SERIAL:
        x = float(VALUE_SERIAL[21:28].strip())
        print (" o valor de x é " + str(x))

    if b'valory' in VALUE_SERIAL :
        y = float(VALUE_SERIAL[21:28].strip())
        print (" o valor de y é " + str(y))

    maquina = Maquina(periodo = 8, dataEHora = datetime.datetime.now(), posicao_x = x, posicao_y = y,  Id_maquina = id_maquina,)
    db.DbContext.add(maquina)

    maquina = db.s.query(Maquina).filter(Maquina.Id_maquina == id_maquina).first()
    db.s.commit()


ser.close()
```

Apêndice B - Algoritmo configuração do banco de dados

```
from sqlalchemy import create_engine
from sqlalchemy.orm import sessionmaker
import Config
from DadosParaPlotarEntity import Base

engine = create_engine(Config.DATABASE_URI)
Session = sessionmaker(bind=engine)
s = Session()

class DbContext:

    @staticmethod
    def create_all_tables():
        Base.metadata.create_all(engine)

    @staticmethod
    def delete_all_tables():
        Base.metadata.drop_all(engine)

    @staticmethod
    def add(model):
        s.add(model)
        s.commit()
        print ('add the model')
        s.close()
        print ('close the session')
```

Apêndice C - Algoritmo da tabela

```
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy import Column, Integer, String, Text, DateTime, Float, Boolean, PickleType


Base = declarative_base()

class DadosParaPlotar(Base):

    __tablename__="DadosParaPlotar"
    id = Column(Integer, primary_key = True)
    periodo = Column(Integer, nullable = False)
    dataEHora = Column(DateTime, nullable = False)
    Id_maquina = Column(String, nullable = False)
    posicao_x = Column(String, nullable = False)
    posicao_y = Column(String, nullable = False)

    def __repr__(self):
        return "<DadosParaPlotar(periodo= '{}', dataEHora= '{}', posicao_x= '{}', posicao_y= '{}', Id_maquina= '{}')>"\
            .format(self.periodo, self.dataEHora, self.posicao_x ,self.posicao_y , self.Id_maquina)
```

Apêndice D - Algoritmo de envio, tratamento e cálculo de distância:

```
/* USER CODE BEGIN Header */
  * File Name          : freertos.c
/* USER CODE END Header */

/* Includes ------------------------------------------------------------------*/
#include "FreeRTOS.h"
#include "task.h"
#include "main.h"
#include "cmsis_os.h"
#include <math.h>

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
#include "usart.h"
/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */

/* USER CODE END PD */

/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */

/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/
/* USER CODE BEGIN Variables */
#define power_1m -32
#define prop_meio_x10 24.6

/* USER CODE END Variables */
osThreadId defaultTaskHandle;
uint32_t defaultTaskBuffer[ 128 ];
osStaticThreadDef_t defaultTaskControlBlock;
osThreadId ProcessaDadosHandle;
uint32_t ProcessaDadosBuffer[ 1024 ];
osStaticThreadDef_t ProcessaDadosControlBlock;
```

```
57 osThreadId ProcessaDadosHandle;
58 uint32_t ProcessaDadosBuffer[ 1024 ];
59 osStaticThreadDef_t ProcessaDadosControlBlock;
60 osMessageQId DataRxHandle;
61 uint8_t DataRxBuffer[ 16 * sizeof( Beacon_t ) ];
62 osStaticMessageQDef_t DataRxControlBlock;
63
64 /* Private function prototypes -----------------------------------------------*/
66
67 float_t rssi_to_meter(int8_t rssi);
68
69
70 /* USER CODE END FunctionPrototypes */
71
72 void StartDefaultTask(void const * argument);
73 void funcProcessaDados(void const * argument);
74
75 void MX_FREERTOS_Init(void); /* (MISRA C 2004 rule 8.1) */
76
77 /* GetIdleTaskMemory prototype (linked to static allocation support) */
78 void vApplicationGetIdleTaskMemory( StaticTask_t **ppxIdleTaskTCBBuffer, StackType_t **ppxIdleTaskStackBuffer, uint32_t *pulIdleTaskStackSize );
79
80 /* USER CODE BEGIN GET_IDLE_TASK_MEMORY */
81 static StaticTask_t xIdleTaskTCBBuffer;
82 static StackType_t xIdleStack[configMINIMAL_STACK_SIZE];
83
84 void vApplicationGetIdleTaskMemory( StaticTask_t **ppxIdleTaskTCBBuffer, StackType_t **ppxIdleTaskStackBuffer, uint32_t *pulIdleTaskStackSize )
85 {
86   *ppxIdleTaskTCBBuffer = &xIdleTaskTCBBuffer;
87   *ppxIdleTaskStackBuffer = &xIdleStack[0];
88   *pulIdleTaskStackSize = configMINIMAL_STACK_SIZE;
89   /* place for user code */
90 }
91 /* USER CODE END GET_IDLE_TASK_MEMORY */
92
93 /**
94   * @brief  FreeRTOS initialization
95   * @param  None
96   * @retval None
97   */
98 void MX_FREERTOS_Init(void) {
99   /* USER CODE BEGIN Init */
100
101   /* USER CODE END Init */
102
103   /* USER CODE BEGIN RTOS_MUTEX */
104   /* add mutexes, ... */
105   /* USER CODE END RTOS_MUTEX */
106
107   /* USER CODE BEGIN RTOS_SEMAPHORES */
108   /* add semaphores, ... */
109   /* USER CODE END RTOS_SEMAPHORES */
110
111   /* USER CODE BEGIN RTOS_TIMERS */
112   /* start timers, add new ones, ... */
113   /* USER CODE END RTOS_TIMERS */
114
115   /* Create the queue(s) */
116   /* definition and creation of DataRx */
117   osMessageQStaticDef(DataRx, 16, Beacon_t, DataRxBuffer, &DataRxControlBlock);
118   DataRxHandle = osMessageCreate(osMessageQ(DataRx), NULL);
119
120   /* USER CODE BEGIN RTOS_QUEUES */
121   /* add queues, ... */
122   /* USER CODE END RTOS_QUEUES */
123
124   /* Create the thread(s) */
125   /* definition and creation of defaultTask */
126   osThreadStaticDef(defaultTask, StartDefaultTask, osPriorityNormal, 0, 128, defaultTaskBuffer, &defaultTaskControlBlock);
127   defaultTaskHandle = osThreadCreate(osThread(defaultTask), NULL);
128
129   /* definition and creation of ProcessaDados */
130   osThreadStaticDef(ProcessaDados, funcProcessaDados, osPriorityNormal, 0, 1024, ProcessaDadosBuffer, &ProcessaDadosControlBlock);
131   ProcessaDadosHandle = osThreadCreate(osThread(ProcessaDados), NULL);
132
133   /* USER CODE BEGIN RTOS_THREADS */
134   /* add threads, ... */
135   /* USER CODE END RTOS_THREADS */
136
137 }
138
139 /* USER CODE BEGIN Header_StartDefaultTask */
140 /**
```

```
146 void StartDefaultTask(void const * argument)
147 {
148   /* USER CODE BEGIN StartDefaultTask */
149     uint8_t aTxHdlc[]= {0x7E,0x00,0x04,0x08,0x01,0x41,0x53,0x62};
150
151
152
153   /* Infinite loop */
154   for(;;)
155   {
156       if(UartReady == IDLE)
157       {
158           HAL_UART_Transmit_DMA(&huart1, aTxHdlc, sizeof(aTxHdlc));
159           UartReady = BUSY;
160       }
161       else
162       {
163           osDelay(100);
164       }
165
166       osDelay(5000);
167
168   }
169   /* USER CODE END StartDefaultTask */
170 }
171
172 /* USER CODE BEGIN Header_funcProcessaDados */
173 /**
174 * @brief Function implementing the ProcessaDados thread.
175 * @param argument: Not used
176 * @retval None
177 */
178 /* USER CODE END Header_funcProcessaDados */
179 void funcProcessaDados(void const * argument)
180 {
181   /* USER CODE BEGIN funcProcessaDados */
182     float_t dist_B1_B2_Fixa = 1.1;
183     float_t distance[64];
184     float_t ang_alfa;
185     float_t ang_beta;
186     float_t d1y1;
```

```
187     float_t d1y2;
188     uint32_t dist_y;
189     uint32_t dist_x;
190     float_t d1x1;
191     float_t d1x2;
192 
193     uint16_t panID[64];
194     //uint8_t aTxHdlc[127];
195     static Beacon_t Device;
196 
197     portBASE_TYPE xHigherPriorityTaskWoken = pdTRUE;
198 
199     uint8_t cnt = 0;
200     uint8_t indice = 0;
201   /* Infinite loop */
202   for(;;)
203   {
204       while(xQueueReceiveFromISR(DataRxHandle, &Device, &xHigherPriorityTaskWoken) == pdTRUE)
205       {
206           if((Device.panId[0]==0x8A&&Device.panId[1]==0x8E)||(Device.panId[0]==0xC9&&Device.panId[1]==0x6F))
207           {
208               cnt++;
209               distance[indice] = rssi_to_meter(Device.rssi);
210               panID[indice++] = ((Device.panId[0]<<8)|Device.panId[1]);
211           }
212 
213           if(indice>1 && indice%2==0)
214           {
215 
216               if((panID[indice-2]==0x8A8E && panID[indice-1]==0xC96F)||(panID[indice-2]==0xC96F && panID[indice-1]==0x8A8E))
217               {
218                   if(panID[indice-2]!=panID[indice-1])
219                   {
220                       ang_alfa = acos((pow(distance[indice-2],2)- pow(dist_B1_B2_Fixa,2) - pow(distance[indice-1],2))/(-2*dist_B1_B2_Fixa*distance[indice-1]));
221                       ang_beta = acos((pow(distance[indice-1],2)- pow(dist_B1_B2_Fixa,2) - pow(distance[indice-2],2))/(-2*dist_B1_B2_Fixa*distance[indice-2]));
222 
223                       if(ang_alfa>=ang_beta&&distance[indice-2]<=distance[indice-1])
224                       {
225                         d1y1 = 1000*(sin(ang_alfa)*distance[indice-2]);
226                         d1y2 = 1000*(sin(ang_beta)*distance[indice-1]);
227 
228                         d1x1 = 1000*(cos(ang_alfa)*distance[indice-2]);
```

```
229                     d1x2 = 1000*(cos(ang_beta)*distance[indice-1]);
230
231                     dist_x = ((d1x1+((1000*dist_B1_B2_Fixa)-d1x2))/2);
232
233                     dist_y = ((d1y1+d1y2)/2);
234                   }
235
236                   if(ang_alfa<=ang_beta&&distance[indice-2]<=distance[indice-1])
237                   {
238                     d1y1 = 1000*(sin(ang_beta)*distance[indice-2]);
239                     d1y2 = 1000*(sin(ang_alfa)*distance[indice-1]);
240
241                     d1x1 = 1000*(cos(ang_beta)*distance[indice-2]);
242                     d1x2 = 1000*(cos(ang_alfa)*distance[indice-1]);
243
244                     dist_x = ((d1x1+((1000*dist_B1_B2_Fixa)-d1x2))/2);
245
246                     dist_y = ((d1y1+d1y2)/2);
247                   }
248
249                   if(ang_alfa>=ang_beta&&distance[indice-2]>=distance[indice-1])
250                   {
251                     d1y1 = 1000*(sin(ang_alfa)*distance[indice-1]);
252                     d1y2 = 1000*(sin(ang_beta)*distance[indice-2]);
253
254                     d1x1 = 1000*(cos(ang_alfa)*distance[indice-1]);
255                     d1x2 = 1000*(cos(ang_beta)*distance[indice-2]);
256
257                     dist_x = ((d1x1+((1000*dist_B1_B2_Fixa)-d1x2))/2);
258
259                     dist_y = ((d1y1+d1y2)/2);
260                   }
261
262                   if(ang_alfa<=ang_beta&&distance[indice-2]>=distance[indice-1])
263                   {
264                     d1y1 = 1000*(sin(ang_beta)*distance[indice-1]);
265                     d1y2 = 1000*(sin(ang_alfa)*distance[indice-2]);
266
267                     d1x2 = 1000*(cos(ang_beta)*distance[indice-1]);
268                     d1x1 = 1000*(cos(ang_alfa)*distance[indice-2]);
269
270                     dist_x = ((d1x1+((1000*dist_B1_B2_Fixa)-d1x2))/2);
```

```
271
272                         dist_y = ((d1y1+d1y2)/2);
273                     }
274
275                     send_x_distance(dist_x);
276
277                     send_y_distance(dist_y);
278                 }
279             }
280         }
281         if(indice>63)
282         {
283             indice = 0;
284         }
285
286
287     }
288
289
290 //    if(UartReady == IDLE)
291 //        {
292 //            HAL_UART_Transmit_DMA(&huart1, aTxHdlc, tamanho de sua msg);
293 //            UartReady = BUSY;
294 //        }
295 //        else
296 //        {
297 //            osDelay(10);
298 //        }
299
300     osDelay(100);
301   }
302   /* USER CODE END funcProcessaDados */
303 }
304
305 /* Private application code ---------------------------------------------------*/
306 /* USER CODE BEGIN Application */
307 float_t rssi_to_meter(int8_t rssi){
308
309     float_t meters = 0;
310
311     meters = pow(10,((power_1m-rssi)/prop_meio_x10));
312     return meters;
```

```
313   }
314
315 void send_x_distance(uint32_t value_x){
316     uint8_t send_msg[]= {0x7E,0x00,0x1B,0x10,0x01,0x00,0x13,0xA2,0x00,0x40,0xB0,
317             0x98,0x96,0xFF,0xFE,0x00,0x00,0x76,0x61,0x6C,0x6F,0x72,0x78,0x20,0xBF,0xBF,0xBF,0xBF,0xBF,0xBF,0xE8};
318     uint8_t divider_x[6];
319     long pot_1;
320     long pot_2;
321     uint8_t count=0;
322     uint8_t cont = 0;
323     long sum = 0;
324
325     for(count=0;count<6;count++)
326     {
327         pot_1 = pow(10,6-count);
328         pot_2 = pow(10,5-count);
329
330         divider_x[count] = (value_x%(pot_1)/pot_2);
331
332         send_msg[24+count] = 0x30+divider_x[count];
333     }
334
335     for(cont=3;cont<30;cont++)
336         {
337             sum = sum+send_msg[cont];
338         }
339
340     send_msg[30] = (0xFF-(sum&0xFF));
341
342     if(UartReady == IDLE)
343           {
344               HAL_UART_Transmit_DMA(&huart1, send_msg, sizeof(send_msg));
345               UartReady = BUSY;
346               osDelay(500);
347           }
348           else
349           {
350               osDelay(100);
351           }
352
353
354   }
```

```
void send_y_distance(uint32_t value_y){
    uint8_t send_msg[]= {0x7E,0x00,0x1B,0x10,0x01,0x00,0x13,0xA2,0x00,0x40,0xB0,0x98,0x96,0xFF,0xFE,0x00,0x00,
            0x76,0x61,0x6C,0x6F,0x72,0x79,0x20,0xBF,0xBF,0xBF,0xBF,0xBF,0xBF,0xE7};
    uint8_t divider_y[6];
    uint8_t count=0;
    uint8_t cont = 0;
    long pot_1;
    long pot_2;

    long sum = 0;

    for(count=0;count<6;count++)
    {
        pot_1 = pow(10,6-count);
        pot_2 = pow(10,5-count);

        divider_y[count] = (value_y%(pot_1)/pot_2);

        send_msg[24+count] = 0x30+divider_y[count];
    }

    for(cont=3;cont<30;cont++)
    {
        sum = sum+send_msg[cont];
    }
    send_msg[30] = (0xFF-(sum&0xFF));

    if(UartReady == IDLE)
              {
                  HAL_UART_Transmit_DMA(&huart1, send_msg, sizeof(send_msg));
                  UartReady = BUSY;
                  osDelay(500);
              }
              else
              {
                  osDelay(100);
              }


  }
```