

**UNIVERSIDADE FEDERAL DE SANTA CATARINA  
DEPARTAMENTO DE ENGENHARIA ELÉTRICA E  
ELETRÔNICA**

Cassiano Candido da Silva

***INTERNET OF SHRIMP: USO DA TECNOLOGIA  
LORAWAN® PARA MONITORAMENTO DE  
CARCINICULTURAS MODERNAS.***

Florianópolis

2019



Cassiano Candido da Silva

***INTERNET OF SHRIMP: USO DA TECNOLOGIA  
LORAWAN® PARA MONITORAMENTO DE  
CARCINICULTURAS MODERNAS.***

Trabalho de Conclusão de Curso submetido ao Departamento de Engenharia Elétrica e Eletrônica da Universidade Federal de Santa Catarina para a obtenção do Grau de Bacharel em Engenharia Eletrônica.  
Orientador: Prof. Richard Demo Souza, Dr.

Florianópolis

2019

Ficha de identificação da obra elaborada pelo autor,  
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Silva, Cassiano Candido da  
INTERNET OF SHRIMP : USO DA TECNOLOGIA LORAWAN  
PARA MONITORAMENTO DE CARCINICULTURAS MODERNAS /  
Cassiano Candido da Silva ; orientador, Richard  
Demo Souza, 2019.  
63 p.

Trabalho de Conclusão de Curso (graduação) -  
Universidade Federal de Santa Catarina, Centro  
Tecnológico, Graduação em Engenharia Eletrônica,  
Florianópolis, 2019.

Inclui referências.

1. Engenharia Eletrônica. 2. IoT. 3.  
Carciniculturas. 4. LoRaWAN. 5. LPWAN. I. Demo  
Souza, Richard. II. Universidade Federal de Santa  
Catarina. Graduação em Engenharia Eletrônica. III.  
Título.

Cassiano Candido da Silva

**INTERNET OF SHRIMP: USO DA TECNOLOGIA  
LORAWAN® PARA MONITORAMENTO DE  
CARCINICULTURAS MODERNAS.**

Este Trabalho de Conclusão de Curso foi julgado aprovado para a obtenção do Título de “Bacharel em Engenharia Eletrônica”, e aprovado em sua forma final pelo Departamento de Engenharia Elétrica e Eletrônica da Universidade Federal de Santa Catarina.


Florianópolis, 11 de dezembro 2019.



---

Prof. Fernando Rangel de Sousa, Dr.  
Coordenador do Curso

Banca Examinadora:



---

Prof. Mário Noronha Neto, Dr.  
Instituto Federal de Santa Catarina



---

Prof. Richard Demo Souza, Dr.  
Orientador



---

Eng. Elço João dos Santos  
Khomp®



Este trabalho é dedicado a minha mãe  
Rosinete Matos Candido.





## **AGRADECIMENTOS**

Agradeço a minha mãe que bancou o custo das viagens que precisei fazer para completar este trabalho e meu orientador Richard Demo Souza por tornar possível o meu desejo de escrever minha monografia no assunto que pretendo seguir minha carreira profissional.



## RESUMO

O objetivo deste estudo foi desenvolver e testar a viabilidade do uso da tecnologia LPWAN no cenário da carcinicultura moderna para o monitoramento das variáveis aquáticas de dentro das estufas. Atualmente, o monitoramento depende de equipamentos manuais e exames laboratoriais, que podem levar horas ou até dias para darem o resultado. As redes de longa distância e de baixa potência LPWAN são fundamentais na proliferação da *Internet* das Coisas (do inglês *Internet of Things* - IoT) e permitem que dispositivos espacialmente dispersos se comuniquem por longas distâncias por meses usando esquemas de modulação dedicados. O conceito *Internet of Shrimp* (IoS) explora a harmonia do uso de sistemas modernos de criação de camarão e LPWAN baseado em *LoRa*® para fornecer acesso aos dados das condições da água dos tanques de criação de camarão. Isso foi alcançado através dos aparelhos da *Khomp*® desenvolvidos para a linha de IoT que usam a camada física *LoRa*® e o protocolo *LoRaWAN*®. Neste estudo, a viabilidade do conceito foi testada através da realização de um experimento em uma fazenda intensiva no sul da Bahia. A Qualidade de Serviço (QoS) e a RSSI foram medidas para avaliar o efeito do canal de comunicação. Juntamente com o uso da diversidade temporal, um QoS de praticamente 100% foi alcançado para todos os nós, afirmando a viabilidade da tecnologia LPWAN no monitoramento de carciniculturas modernas que usam sistemas intensivos.

**Palavras-chave:** LPWAN. *LoRa*®. IoT. Monitoramento. Carcinicultura.



## ABSTRACT

The aim of this study was to develop and test the feasibility of using LPWAN technology in the modern shrimp scenario to monitor the aquatic variables inside the greenhouses. Currently, monitoring depends on manual equipment and laboratory tests, which may take hours or even days to give the result. Low Power Wide Area Networks (LPWAN) are instrumental in the proliferation of the Internet of Things (IoT) and allow spatially dispersed devices to communicate over long distances for months using dedicated modulation schemes. The Internet of Shrimp (IoS) concept exploits the harmony of using modern shrimp farming systems and LoRa®-based LPWAN to provide access to water conditions data from shrimp farming ponds. This was achieved through Khomp® devices developed for the IoT line using the LoRa® physical layer and LoRaWAN® protocols. In this study, the viability of the concept was tested by conducting an experiment on a super intensive farm in southern Bahia. The proposed LPWAN Quality of Service (QoS) and Received Signal Strength Indication (RSSI) were measured to assess the effect of the communication channel. Combined with the use of temporal diversity, almost 100% QoS has been achieved for all nodes, affirming the feasibility of LPWAN monitoring in modern farms using intensive systems.

**Keywords:** LPWAN. LoRa®. IoT. Monitoring. shrimp farming.



## LISTA DE FIGURAS

Figura 1	Fazenda Bahia Sul, localizada em Canavieira, Bahia. Fonte: Elaborado pelo autor. ....	22
Figura 2	Camadas do sistema baseado em LPWAN. Fonte: Elaborado pelo autor. ....	27
Figura 3	End device LoRa® NIT 21 LI. Fonte: (KHOMP, 2019b). ....	27
Figura 4	Gateway ITG200. Fonte: (KHOMP, 2019a). ....	29
Figura 5	Fluxograma do protocolo MQTT. Fonte: Elaborado pelo autor. ....	30
Figura 6	Fluxograma dos programas escritos em Python. Fonte: Elaborado pelo autor. ....	31
Figura 7	Posicionamento do gateway e dos pontos onde serão feitas as medidas. Fonte: Elaborado pelo autor. ....	32
Figura 8	PER e QoSe2e para cada nó do teste em campo. Fonte: Elaborado pelo autor. ....	35
Figura 9	Médias das RSSIs em forma de mapa de calor. Fonte: Elaborado pelo autor. ....	35
Figura 10	PER e QoSe2e para cada nó do teste em campo usando a diversidade temporal. Fonte: Elaborado pelo autor. ....	38
Figura 11	RSSIs médias em forma de mapa de calor usando a diversidade temporal. Fonte: Elaborado pelo autor. ....	38
Figura 12	Interior de uma estufa de carcinicultura vazia. Fonte: Elaborado pelo autor. ....	59
Figura 13	Interior de uma estufa de carcinicultura em operação. Fonte: Elaborado pelo autor. ....	59
Figura 14	Fermentação dos bioflocos antes de adicioná-los à água. Fonte: Elaborado pelo autor. ....	60
Figura 15	Torre para <i>Internet</i> via satélite. Fonte: Elaborado pelo autor. ....	60
Figura 16	Foto do <i>end-device LoRa®</i> durante as medidas. Fonte: Elaborado pelo autor. ....	61





## LISTA DE ABREVIATURAS E SIGLAS

ONU	Organização das Nações Unidas.....	19
ANATEL	Agência Nacional de Telecomunicações.....	19
FENACAM	Feira Nacional de Camarão .....	20
WSSV	White Spot Syndrome Virus .....	20
LPWAN	Low Power Wide Area Networks .....	21
LoRaWAN®	Long Range Wide Area Networks .....	21
RSSI	Received Signal Strength Indication.....	22
LoRa®	Long Range .....	22
MQTT	Message Queuing Telemetry Transport .....	22
SF	Spread Factor .....	23
QoS	Quality of Service.....	25
IoS	Internet of Shrimp.....	25
CSS	Chirp Spread Spectrum.....	26
ISM	Industrial, Scientific and Medical radio bands.....	28
IoT	Internet of Things .....	31
e2e	end-to-end.....	31
PER	Packet Error Rate .....	33



## SUMÁRIO

<b>1 INTRODUÇÃO</b> .....	19
1.1 OBJETIVO GERAL .....	22
1.2 OBJETIVOS ESPECÍFICOS .....	22
1.3 DELIMITAÇÃO DO TRABALHO .....	23
<b>2 MATERIAL E MÉTODOS</b> .....	25
2.1 NECESSIDADES E RESTRIÇÕES DO SISTEMA .....	25
2.2 CAMADAS FÍSICAS DISPONÍVEIS PARA LPWAN .....	25
2.3 DESCRIÇÃO DO SISTEMA .....	26
2.3.1 Camada de sensoriamento: End-devices.....	27
2.3.2 Camada de rede: gateway .....	29
2.3.3 Camada de Aplicação: Servidor .....	30
2.4 TESTE DE CAMPO E VALIDAÇÃO .....	31
<b>3 RESULTADOS</b> .....	35
<b>4 DISCUSSÃO</b> .....	37
4.1 RSSI, QOS, PER E A PERFORMANCE DO SISTEMA LPWAN.....	37
4.2 DIVERSIDADE TEMPORAL .....	37
<b>5 CONCLUSÃO</b> .....	41
<b>REFERÊNCIAS</b> .....	43
<b>APÊNDICE A – Códigos Python</b> .....	49
<b>APÊNDICE B – Figuras extras</b> .....	59



## 1 INTRODUÇÃO

Considerando um cenário de crescimento econômico modesto, de acordo com a ONU, a população mundial deve crescer para quase 9,73 bilhões em 2050, aumentando a demanda agrícola em cerca de 50% em comparação a 2013. O crescimento da renda nos países de baixa e média renda aceleraria uma transição dietética para maior consumo de carne, frutas e legumes, em relação ao de cereais, exigindo mudanças proporcionais na produção e aumentando a pressão sobre os recursos naturais (FAO, 2017).

Para atender à demanda, a agricultura em 2050 precisará produzir quase 50% mais alimentos, rações e biocombustível do que em 2012. Para suprir essa procura, desde a década de 1980, praticamente todo o aumento na quantidade de peixe consumido vem da aquicultura, que ultrapassou o crescimento da população e se tornou a indústria de produção de alimentos que mais cresce no mundo segundo a pesquisa da FAO(2016).

A produtividade da aquicultura tem aumentado devido à intensificação dos métodos de produção. Na Ásia, a aquicultura tradicional em pequena escala (em que várias espécies de carpas com comportamentos complementares de alimentação foram armazenadas em viveiros fertilizados) deu lugar à produção de peixes e crustáceos que dependem fortemente, se não exclusivamente, de rações. Os principais impulsionadores têm aumentado os preços das terras e os altos preços pagos pelos peixes de criação, o que torna os alimentos acessíveis (BEVERIDGE, 2013).

O potencial do Brasil para o desenvolvimento da aquicultura é de tal magnitude que se for eficientemente explorado, o país pode vir a competir pela liderança na produção mundial desse setor, pois dispõe de 1.000.000 ha de áreas já apropriadas, e conta com excelentes condições edafoclimáticas (aquilo que é relativo ao clima e ao solo), infra-estruturais e um amplo mercado interno; Todos estes elementos são suficientemente sólidos e atrativos, para viabilizar essa exploração (ROCHA, 2018).

Entretanto, o atual crescimento da aquicultura tem aumentado ainda mais a demanda por novas áreas de criação, e simultaneamente novas áreas propícias tem se tornado escassas e de difícil acesso, visto que muitas delas estão sendo protegidas por órgãos governamentais ou já são ocupadas por outros piscicultores, que ainda não usam sistemas mais produtivos. Problemas desse tipo em outras partes do mundo

tem estimulado outras formas de criação de animais marinhos, como é o caso das fazendas de criação de peixe em mar aberto na Noruega, em que o limite de espaço não é um problema (BJELLAND et al., 2015).

Já no setor da carcinicultura (área da aquicultura relacionada à criação de camarão) Brasileira, como visto na Feira Nacional de Camarão (FENACAM), realizada todos os anos nos meses de novembro na cidade de Natal, RN, os carcinicultores tem optado pela criação intensiva de camarão para reduzir as áreas ocupadas das fazendas e aumentar a sua produtividade. O método de criação intensiva combina o uso de bioflocos, que insere à água do viveiro conjuntos de microrganismos que tornam a alta densidade de camarão possível, com o uso de estufas que isolam os tanques de criação do meio externo, protegendo os camarões das variações de temperatura e, de doenças como a mancha branca (White Spot Syndrome Virus - WSSV), que tem afetado fortemente a produção de camarão internacional desde 2003, com o pico em 2011 (ROCHA, 2018). Sistemas intensivos conseguem produzir de 200 a 600 camarões por metro cúbico, que se comparado com os sistemas extensivos, que chegam a no máximo 6 a 10 camarões por metro cúbico, são muito mais eficientes e ocupam menos área.

Mesmo com o uso desses dois métodos citados acima em criações intensivas, a criação de camarão em alta densidade é um grande desafio devido ao grande número de variáveis independentes que afetam o sistema como um todo. Temperatura, oxigenação, amônia, nitrito, nitrato, pH e condutividade são fatores que influenciam diretamente a qualidade da água e o bem estar do camarão, e precisam ser monitoradas constantemente (ROCHA, 2018).

A existência de fatores variáveis implicam a necessidade de soluções e ferramentas tecnológicas que permitam aos carcinicultores observar e monitorar o ambiente aquático dos tanques de criação de maneira consistente. Atualmente, o monitoramento depende de equipamentos manuais e exames laboratoriais, que podem levar horas ou até dias para darem o resultado. Assim, fica clara a necessidade de soluções tecnológicas que monitorem e atuem de forma automática, para tornar o futuro desta atividade, que tem sofrido diversos problemas, viável e bem sucedido.

IoT refere-se a um paradigma de rede em que dispositivos físicos, como uma lixeira por exemplo, são integrados a microcontroladores e conectados à *Internet*, facilitando o acesso e a troca de dados com humanos. Mas esses dispositivos geralmente são alimentados por baterias e não suportam se comunicar usando protocolos como WI-FI, que são complexos e demandam muita energia. As aplicações atuais da IoT

incluem nós de detecção independentes e simples para automação residencial, automóveis, sensores industriais e agrícolas e atuadores (RAZA et al., 2017).

Além de serem pequenos e alimentados por bateria, esses dispositivos IoT, também chamados de dispositivos finais (derivado do inglês *end-devices*), são frequentemente distribuídos por grandes áreas geográficas, restritos a rigorosos orçamentos de energia e se comunicando com taxas de dados relativamente baixas. Essas características sugerem que não é ideal conectar dispositivos IoT usando tecnologias de curto alcance como WI-FI ou redes celulares tradicionais, mas sim empregar protocolos de rádio de baixa potência e longo alcance. O conceito de *Low Power Wide Area Networks* (LPWAN) implica em uma tecnologia de comunicação emergente que complementam as redes sem fio existentes, abordando diretamente os requisitos exclusivos dos dispositivos de IoT e que se baseia em uma arquitetura semelhante às redes celulares (RAZA et al., 2017).

As LPWANs exploram as bandas industriais, científicas e médicas (sigla derivada do inglês ISM), não licenciadas, e fornecem grandes áreas de cobertura e baixo consumo de energia nos dispositivos finais, utilizando esquemas de modulação eficientes e ciclos intermitentes de transmissão/recepção, com sensibilidade típica de recepção tão baixa quanto -130 dBm a -150 dBm. O baixo consumo de energia e as amplas áreas de cobertura das LPWANs são alcançadas a um custo de taxas de dados muito baixas, normalmente da ordem de alguns kilobytes por segundo (kbps). No entanto, dispositivos IoT típicos precisam apenas trocar pequenas quantidades de dados em intervalos esporádicos, geralmente enviados para um servidor IoT central através de nós de *gateway* em uma topologia estrela, tornando menos importante a alta taxa de dados (RAZA et al., 2017).

As LPWANs são altamente relevantes para sistemas de monitoramento dos viveiros de camarão que permitem o acesso do usuário a dados como oxigenação, amônia e etc. Portanto, este estudo foi focado no teste de uma solução de monitoramento em uma fazenda de camarão, fazendo o uso da tecnologia LPWAN. A solução proposta faz uso dos aparelhos destinados a IoT da Khomp®, um *gateway* e um *end-device*, que fornecem uma interface de comunicação sem fio, de longo alcance e econômica para captar as variáveis da temperatura, umidade e RSSI (do inglês *Received signal strength indication*). Juntamente com o sistema intensivo que faz o uso de estufas e bioflocos, vários desses *end-devices* podem ser conectados em uma topologia estrela para formar uma LPWAN das variáveis dos viveiros, estabelecendo um conceito

chamado "Internet of Shrimp"(IoS). O sistema proposto foi testado em uma fazenda de criação de camarão de larga escala que faz o uso de sistemas intensivos de criação, no sul da Bahia, mais precisamente na cidade de Canavieiras, como visto na Figura 1 e nas Figuras extras no apêndice B no fim deste trabalho.



Figura 1 – Fazenda Bahia Sul, localizada em Canavieira, Bahia. Fonte: Elaborado pelo autor

## 1.1 OBJETIVO GERAL

É o objetivo desse trabalho avaliar experimentalmente o QoS (em termos da probabilidade de erro de pacotes) dos aparelhos de comunicação sem fio que usam a modulação *LoRa*® e o protocolo *LoRaWAN*®, em uma fazenda de criação de camarão que faz o uso de sistemas intensivos através de estufas e bioflocos em 13 viveiros.

## 1.2 OBJETIVOS ESPECÍFICOS

- Desenvolver um *script* para acessar o *broker* que contém os pacotes enviado pelo *end-device*, via protocolo MQTT, para que se tenha acesso a RSSI, temperatura e umidade, para então, armazená-las.



- Desenvolver um *script* que trace os gráficos da RSSI, temperatura e umidade dos arquivos gerados pelo primeiro *script*.
- Investigar o impacto de diversidade temporal no desempenho do sistema.
- Testar na prática o uso dos aparelhos da Khomp®, extraindo os dados da recepção dos sinais para os pontos mais perto e mais longe de cada tanque de camarão até o *gateway*.
- Participar da Feira Nacional de Camarão para assim melhor analisar o estado da arte do cenário nacional em relação às tecnologias empregadas no cultivo de camarão.
- Avaliar a viabilidade da tecnologia *LoRa*®/*LoRaWAN*® na carcinicultura moderna.

### 1.3 DELIMITAÇÃO DO TRABALHO

Avaliar a interferência de diversos *end-devices* distribuídos dentro dos viveiros está fora do escopo deste trabalho, visto o recurso limitado de apenas um aparelho disponível. Além de contar com apenas um *end-device*, não foi possível setar as configurações, como por exemplo, o *Spreading Factor* (SF). Isso limita o trabalho, mas ainda está dentro dos objetivos gerais. Outro ponto limitante deste trabalho é a falta de possíveis pontos de instalação do *gateway* para um melhor dimensionamento de rede, visto que na fazenda a única forma de acesso à internet é via satélite, contando com apenas um ponto que tem um cabo de rede disponível.



## 2 MATERIAL E MÉTODOS

### 2.1 NECESSIDADES E RESTRIÇÕES DO SISTEMA

O alcance é um requisito importante para o conceito de IoS. Em fazendas de produção extensiva, em que as altas densidades de camarão por metro cúbico não são exploradas, a extensão máxima do começo ao fim dos viveiros de camarão passam de 5 Km. Já nas fazendas de produção intensivas essa extensão não passa de centenas de metros. Como esse trabalho foca nos processos intensivos, que são os que prometem atender à grande demanda futura de frutos do mar, foi constatado que a extensão máxima dos viveiros da fazenda Bahia Sul, que é a maior fazenda de sistemas intensivos do Brasil e onde o teste foi ministrado, é de 566 m, indicando assim, o alcance que a rede *LoRaWAN*<sup>®</sup> deve fornecer.

Por mais que seja extremamente importante monitorar os parâmetros das águas dos viveiros, não é necessário que essas medidas sejam monitoradas a cada segundo, ou mesmo a cada minuto. Variáveis como temperatura e oxigenação são extraídas a cada 6 horas, manualmente, e outras variáveis como amônia, nitrito, nitrato são informadas a cada 3 dias pelos laboratórios locais que coletam a água e levam para a análise. Portanto, monitorar os dados na escala de minutos seria um avanço muito grande em relação ao monitoramento manual.

Por a fazenda Bahia Sul se localizar distante da cidade, o acesso à internet no local se torna restrito e possível apenas por sinal de satélite. Para isso, a fazenda conta com uma antena que recebe o sinal de satélite e encaminha para uma das casas dos proprietários, que fica próximo dos viveiros, através de cabo de rede, limitando o número de possíveis pontos para instalar o *gateway*, mas tornando-o um local interessante ao mesmo tempo, pois fica a uma distância de menos de 20 metros das estufas de criação.

### 2.2 CAMADAS FÍSICAS DISPONÍVEIS PARA LPWAN

Várias tecnologias como *SigFox*<sup>®</sup>, *Weightless*<sup>®</sup>, *LoRa*<sup>®</sup> e *Ingeniu* estão competindo dentro do cenário de LPWAN (RAZA et al., 2017).

Todas essas tecnologias suportam tranquilamente a cobertura de centenas de metros de extensão dos viveiros intensivos de camarão. As

suas diferenças estão no tipo de modulação usada nas camadas físicas, que afetam o tamanho da mensagem, banda e taxa de transmissão. As tecnologias como *SigFox*® e *Weightless*® tem severas restrições quanto ao tamanho das mensagens, com a *SigFox*® sendo limitada com mensagens de apenas 12 Bytes e *weightless*® com mensagens de 20 Bytes (RAZA et al., 2017). Com essas restrições, essas tecnologias não são necessariamente aptas para o conceito de IoS, visto que, potencialmente, existem muitas variáveis de medição. No entanto, devido a folga de tempo que existe entre uma medição e outra, pode-se optar por fazer o uso dessas tecnologias transmitindo as inúmeras variáveis em tempos diferentes, porém isso pode acarretar em uma menor durabilidade da bateria.

Apesar do *Ingenu*® não ter problemas com o tamanho das mensagens, ele possui um alto consumo de energia comparada com as outras tecnologias, além de não estar disponível no Brasil (ADELANTADO et al., 2017).

A quarta tecnologia LPWAN considerada neste estudo é o *LoRa*®, que usa modulação *Chirp Spread Spectrum* (CSS), oferece uma carga útil de até 250 bytes, possui taxas de dados de até 37,5 kbps e consumo de energia muito baixo (RAZA et al., 2017).

Um recurso exclusivo do *LoRa*® é o parâmetro *Spreading Factor* (SF), que fornece um grau adicional de liberdade no design do *end-device*. O SF pode ser escolhido como uma troca entre a área de cobertura, taxas de dados e tamanho do pacote de rádio (RAZA et al., 2017; ADELANTADO et al., 2017). A tecnologia *LoRa*® também foi identificada como a melhor candidata para aplicações similares na agricultura (ADELANTADO et al., 2017; TALAVERA et al., 2017), e será a tecnologia considerada neste trabalho, em conjunto com o protocolo de comunicação *LoRaWAN*®. Ademais, vale ressaltar que a tecnologia não depende de um operador de rede comercial, podendo ser implantada pelo próprio usuário.

### 2.3 DESCRIÇÃO DO SISTEMA

Três camadas podem ser identificadas em um sistema de monitoramento típico baseado em LPWAN. A primeira camada é a de sensoriamento que consiste nos *end-devices* representados pelos dispositivos que são distribuídos geograficamente/espacialmente (TALAVERA et al., 2017). A segunda camada é a camada de rede e consiste em um dispositivo centralizado indicado como *gateway*. Todos os *end-devices*

na camada de sensoriamento se comunicam com esse *gateway* em uma topologia estrela. A terceira camada nessa representação é a de aplicação, que normalmente apresenta um servidor e um banco de dados que funciona como um *back-end* e *front-end* do sistema para apresentação dos dados ao usuário. O sistema LPWAN desenvolvido no presente estudo está em conformidade com essa arquitetura e inclui as mesmas três camadas, como pode ser visto na Figura 2.

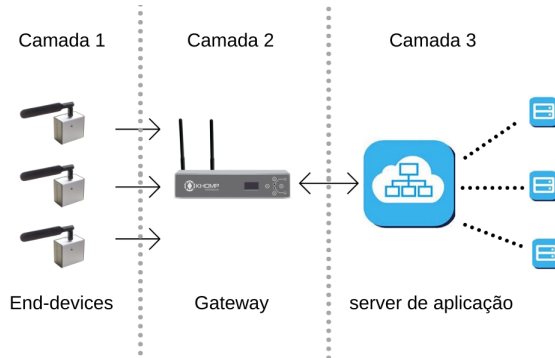


Figura 2 – Camadas do sistema baseado em LPWAN. Fonte: Elaborado pelo autor.

### 2.3.1 Camada de sensoriamento: End-devices.



Figura 3 – End device LoRa® NIT 21 LI. Fonte: (KHOMP, 2019b).

O *end-device* LoRa® NIT 21 LI usado nesse trabalho é um módulo autônomo operado por bateria, baseado em microcontrolador e

desenvolvido pela Khomp®), na linha de IoT, para atuar como o dispositivo de percepção da camada 1 da rede LPWAN e fornecer os dados de temperatura, umidade e RSSI (KHOMP, 2019b). Cada *end-device* possui um ID exclusivo e é um dispositivo de transmissão básico no sistema LPWAN, também conhecido como nó na terminologia.

O alcance efetivo da comunicação, o tamanho máximo da carga útil e o tempo no ar são determinados no *LoRa*® pelo fator de espalhamento (SF), largura de banda e potência de transmissão (AUGUSTIN et al., 2016). Para uma largura de banda e potência de transmissão fixa, o SF define o tamanho máximo de uma mensagem de rádio em bytes, tempo no ar e sensibilidade do receptor. Os valores típicos de SF variam de SF7 a SF12. As mensagens enviadas com um SF7 gastam menos tempo no ar, requerem receptores sensíveis e podem ter um tamanho de mensagem útil de até 250 bytes, enquanto as mensagens enviadas com o SF12 passam mais tempo no ar, requerem receptores menos sensíveis e têm um tamanho de mensagem útil de até 60 bytes. (AUGUSTIN et al., 2016; RAZA et al., 2017; ADELANTADO et al., 2017).

Para modulações LPWAN que operam na banda livre ISM sub-gigahertz, tempo no ar e as sub bandas de frequência são reguladas em diferentes regiões do mundo pelos órgãos de telecomunicações responsáveis (ADELANTADO et al., 2017; RAZA et al., 2017). No casos dos aparelhos da Khomp®, o órgão que exerce essa função é a ANATEL.

Na união europeia (UE), a modulação *LoRa*® usa a banda ISM de 868 MHz com um *duty cycle* permitido de 1%, o que fornece um tempo de transmissão no ar máximo de 36 s/h (ADELANTADO et al., 2017). O tempo no ar para cada transmissão na modulação *LoRa*® depende do tamanho de cada pacote, largura de banda, tamanho do cabeçalho e SF (AUGUSTIN et al., 2016) e pode ser calculado usando as ferramentas de software fornecidas pelo fabricante do chip de modulação *LoRa*®.

O *end-device LoRa*® da *Khomp*® usado neste trabalho é programado para usar SF10 com uma banda de 125 kHz, potência transmitida de 14 dBm e uma antena de ganho de 5 dBi. Esses parâmetros já são setados pelo fabricante e não se sabe como alterados. Apesar disso restringir o trabalho, como já comentado nas limitações do projeto, ainda é possível avaliar a viabilidade da rede *LoRaWAN*®, pois com um SF10 tem-se um balanço entre sensibilidade de recepção, tempo no ar e o tamanho de mensagens.

### 2.3.2 Camada de rede: gateway



Figura 4 – Gateway ITG200. Fonte: (KHOMP, 2019a).

O *gateway* funciona como um nó centralizado para todos os *end-devices*, é capaz de rodar um servidor de rede LoRaWAN® interno, e é responsável por encaminhar todas as mensagens recebidas dos nós com IDs autorizados para um servidor. O ITG200 (KHOMP, 2019a) foi usado como o *gateway* nesse trabalho, vindo equipado com uma antena com ganho de 5 dBi adequada para receber dados dos nós por meio do link de rádio LoRa® e uma entrada para cabo de rede para se ter conexão com a Internet e transmitir os dados recebidos ao servidor.

O ITG200 suporta o protocolo Eclipse Mosquitto MQTT (LIGHT, 2017) para enviar dados pela Internet. O MQTT é um protocolo baseado em *publish/subscribe* usado em aplicações *Machine to Machine* (M2M) e IoT, funciona sobre o protocolo TCP/IP e possui dois elementos, um cliente e um *broker* (HUNKELER et al., 2008).

Qualquer dispositivo que suporte um protocolo MQTT é um cliente MQTT, e um cliente pode ser um *subscriber* (que está recebendo dados) ou um *publisher* (que está produzindo dados). Um *client* se conecta a um *broker* MQTT, que é o aplicativo central no protocolo. O *broker* é responsável por conectar e manter conexões com todos os *clients* e por receber dados dos *clients*. As mensagens trocadas no MQTT são fornecidas com um campo adicional indicado como o *topic*. Esse campo torna o MQTT escalável e mais versátil, permitindo que o *broker* faça a filtragem e o encaminhamento de mensagens com base nesse valor. Isso é possível porque os assinantes do MQTT assinam um *topic* em vez dos dados de um editor específico, o que significa que o *broker* é responsável por garantir que os dados marcados com um *topic* específico sejam encaminhados a todos os *subscribers* desse *topic* (HUNKELER et al., 2008).

### 2.3.3 Camada de Aplicação: Servidor

Um computador pessoal com acesso à Internet recebeu a função de servidor na camada de aplicação. O servidor foi responsável por receber e armazenar dados localmente em seu disco rígido e por apresentar dados de temperatura, umidade, RSSI e pacotes perdidos aos usuários finais. Para isso, dois *scripts* foram desenvolvidos: um *client* assinante do broker público test.mosquitto.org (ECLIPSE™, 2019) e que salva os dados em um arquivo e uma interface gráfica para poder-se visualizar os dados, ambos em *Python*.

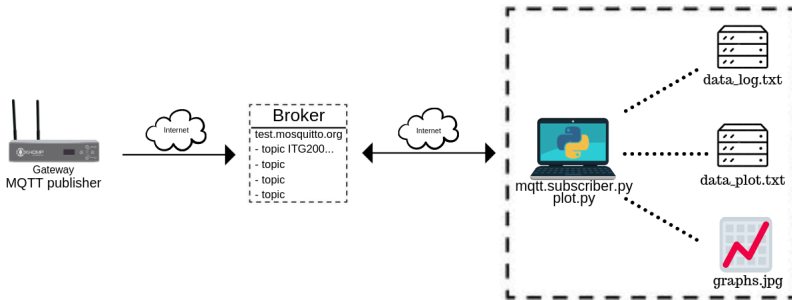


Figura 5 – Fluxograma do protocolo MQTT. Fonte: Elaborado pelo autor.

Fazendo-se o uso da biblioteca *Paho* (PYTHON, 2019), disponível em *Python*, foi possível conectar-se ao broker e se inscrever no *topic* - ITG200/0000F8033201a70a/measurement - para onde o *gateway* envia os dados. Depois de acessado esses dados, o *script* salva-os em um arquivo .txt, que posteriormente será usado pelo outro *script* para gerar os gráficos na interface. A Figura 5 ilustra como que o protocolo MQTT foi usado em todo o processo para a obtenção dos dados, enquanto que a Figura 6 ilustra o diagrama de fluxo dos 2 programas escritos em *Python* para ter-se acesso aos dados do *broker* e para traçar os gráficos, respectivamente.

A mensagem de *uplink* enviada pelo *end-device* possui um campo de *timestamp* que indica a hora que a mensagem foi enviada, com isso, é possível identificar quais pacotes foram perdidos, visto que o *end-device* manda uma mensagem por minuto. Com a finalidade de ilustrar os pacotes perdidos, foi atrelado um valor de -200 dBm para cada um deles, como pode ser visualizado na Figura 6. Ambos os códigos estão



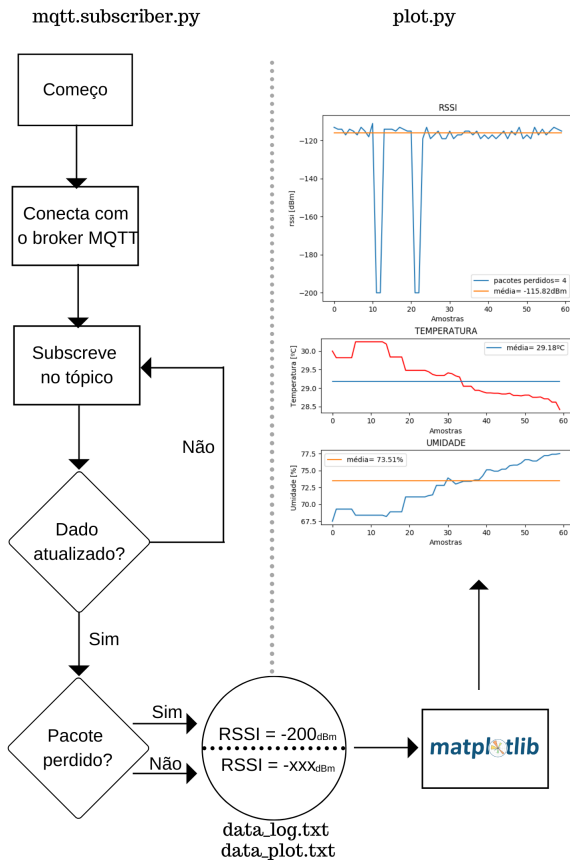


Figura 6 – Fluxograma dos programas escritos em Python. Fonte: Elaborado pelo autor.

comentados e anexados no apêndice A no fim deste trabalho.

## 2.4 TESTE DE CAMPO E VALIDAÇÃO

A funcionalidade do sistema LPWAN foi primeiro testada e verificada em uma série de experimentos realizados em campo. Posteriormente, foi realizado um extenso teste de campo em uma fazenda

comercial de camarão no sul da Bahia, com o objetivo de estudar a viabilidade do uso do LPWAN baseado em LoRa® para monitoramento dos parâmetros de temperatura, umidade e RSSI dos viveiros intensivos de camarão. Um *gateway* e um *end-device* foram usados no teste de campo. A posição do *gateway* está ilustrada na Figura 7, juntamente com as posições onde as medidas foram feitas.



Figura 7 – Posicionamento do gateway e dos pontos onde serão feitas as medidas. Fonte: Elaborado pelo autor.

O *gateway* encontra-se a uma altura de 1 metro no interior da casa, indicada pelo localizador em vermelho. Os *end-devices* foram colocados a uma altura de 2 m no interior das estufas nos 23 pontos em azul. Esses pontos representam o ponto mais perto e o mais longe de cada viveiro de camarão. Com isso, é possível ter noção do intervalo do valor da RSSI e QoS (na forma de taxa de sucesso na transmissão dos pacotes) para cada tanque. A escolha de colocá-los no interior das estufas deve-se pela intenção de ver qual o efeito que as estruturas metálicas da cobertura e outros obstáculos, como aeradores, chafarizes, etc, causam no sinal transmitido pelo *end-device*.

Foram extraídos 60 pacotes de cada ponto ilustrado em azul, acumulando um tempo de medição total de 23 horas, realizado em um período de 5 dias. Durante os testes foi possível observar a temperatura, umidade e RSSI de cada pacote enviado, como ilustrado na Figura 6 no lado superior direito.

Um aspecto importante do experimento foi avaliar o QoS e a

RSSI da implementação proposta da LPWAN. A União Internacional das Telecomunicações - Setor de Normalização das Telecomunicações (ITU-T), define QoS como um indicador de satisfação percebido ou experimentado por um usuário de um serviço (IVERSEN et al., 2013) e também propõe QoS *end-to-end* e QoS baseada em camadas de um sistema de comunicação. De acordo com RAZA et al. (2017), tecnologias LPWAN existentes fornecem QoS inexistente ou limitado. DUAN et al. (2011) também destaca a falta de uma definição genérica de QoS em IoT e LPWANs e propõe QoS baseada em camadas para cada camada das três camadas do sistema baseado em LPWAN. Os parâmetros que podem ser usados para avaliar a QoS da rede de uma LPWAN incluem perda de pacotes, atraso e largura de banda (DUAN et al., 2011).

Um outro estudo também propõe a perda de pacotes ou a taxa de erros de pacotes (do inglês *Packet Error Rate* - PER) como um dos parâmetros de destino para avaliar a QoS no LPWAN baseado em LoRa® (PETRIĆ et al., 2016). Portanto, o PER *end-to-end* (e2e) é usado neste estudo para definir a QoS e2e e é calculado com base em:

$$PER_{e2e} = (1 - QoS_{e2e}); \quad (2.1)$$

$$QoS_{e2e} = \frac{\text{Número de mensagens não corrompidas}}{\text{Número de mensagens enviadas}}. \quad (2.2)$$

O número de pacotes transmitidos por um nó e o número total correspondente de pacotes não corrompidos recebidos no servidor são, portanto, os únicos parâmetros necessários para avaliar a QoS e2e de um nó. A QoS e2e de todos os *end-devices* foi calculado com base nos arquivos adquiridos durante o experimento para avaliar o desempenho geral do conceito IoS no ambiente de criação de camarão marinho.

Para se ter uma noção de quanto o sinal transmitido é atenuado por todo o ambiente das estufas, o valor da RSSI também é considerado e avaliado no capítulo seguinte. Esse valor é calculado pelo ITG200 e faz parte do pacote enviado ao *broker*. Esse valor pode ser tão baixo quanto -140 dBm, que é o limite da sensibilidade do *gateway* ITG200. Quando a RSSI é menor que -140 dBm, o *gateway* não recebe o pacote e não atualiza o *broker*.



### 3 RESULTADOS

A Figura 8 mostra, para os 60 pacotes enviados por cada nó, os parâmetros de QoS e2e e PER e2e. Já na Figura 9, temos o valores médios para as RSSIs em forma de mapa de calor.

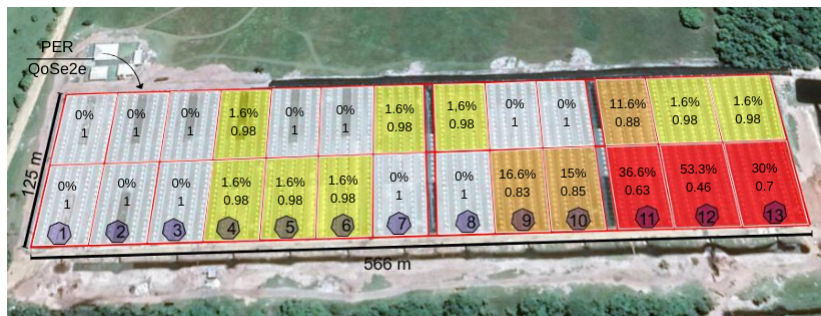


Figura 8 – PER e QoS e2e para cada nó do teste em campo. Fonte: Elaborado pelo autor.

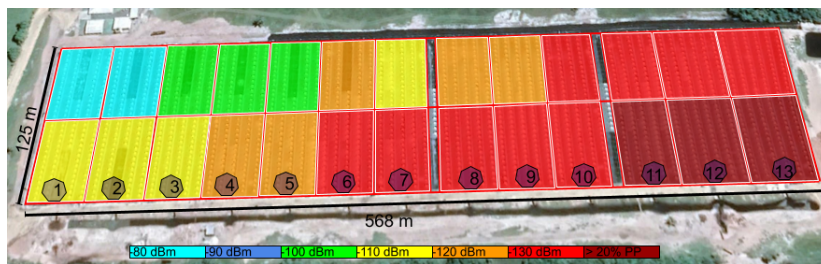


Figura 9 – Médias das RSSIs em forma de mapa de calor. Fonte: Elaborado pelo autor.

Considerando que as medidas foram feitas em cada uma das duas metades de cada tanque, é notável que 75% das metades dos tanques obtiveram um QoS e2e maior que 90%. Os outros 25%, representado pelos tanques mais distantes e com cores em laranja e vermelho, mostram um mau QoS com um mínimo de 46% no Tanque 12, sinalizando uma grande perda de pacotes no percurso.

Na Figura 9, pode-se ver a grande atenuação que o sinal sofre ao longo do percurso no interior das estufas. A partir do Tanque 6, e mesmo a uma distância de apenas 280m, o sinal já está na zona vermelha, indicando sinais com RSSIs inferiores a -130 dBm. As metades dos tanques na cor vermelho escuro representam os viveiros que tiveram uma RSSI inferior a -130 dBm e PER e superior a 20%, que notadamente são os mesmos 3 tanques que tiveram um péssimo desempenho na Figura 8.

## 4 DISCUSSÃO

### 4.1 RSSI, QOS, PER E A PERFORMANCE DO SISTEMA LPWAN.

Os nós mais distantes da rede LPWAN não demonstraram bons resultados no teste de campo. Em um estudo realizado em uma fazenda de salmão em alto mar, onde testou-se a rede LoRaWAN<sup>®</sup>, conseguiu-se um QoS e de acima de 90% para uma distância maior de 2Km com linha de visada (HASSAN et al., 2019). Isso mostra a forte atenuação que a estufa e todo o ambiente causa no sinal. Pode-se notar também que essa atenuação gerada pelas estufas é maior que a atenuação gerada no interior de cidades, onde em um estudo realizado em um ambiente urbano, em que também fazia-se o teste da rede LoRaWAN<sup>®</sup>, obteve-se um resultado de PER de 5.3% (PETÄJÄJÄRVI et al., 2016).

### 4.2 DIVERSIDADE TEMPORAL

O monitoramento das variáveis aquáticas dos tanques de camarão aceita uma latência razoavelmente grande, possibilitando o uso de diversidade temporal, na forma de repetição de pacotes. Ao replicar os pacotes enviados ao *gateway*, há uma melhora na recepção da informação, pois os pacotes sofrem efeitos diferentes do canal sem fio.

Como discutido em SANTOS (2019), o uso dessa diversidade ajuda o sistema na recepção dos pacotes transmitidos, mas levanta o ponto de que caso seja feita muitas replicações destes pacotes, o sistema pode a vir a se sobrecarregar e aumentar a interferência entre os pacotes de outros *end-devices*, além de consumir uma maior energia.

Por outro lado, trabalhos como RAZA et al. (2017) e Augustin et al. (2016) realizaram um estudo sobre colisões e interferência a partir de simulações computacionais. Os resultados mostram que, mesmo sob interferência, dispositivos *LoRa*<sup>®</sup> conseguem se comunicar com uma baixa probabilidade de erro, desde que a densidade de nós ou o tráfego não sejam demasiadamente grandes.

Com o intuito de usar a diversidade temporal nesse trabalho, e observar os seus efeitos, foi introduzido no *script* `mqtt.subscriber.py` linhas de código para que ao invés de armazenar os dados dos pacotes direto no arquivo `.txt`, fosse feita uma comparação da melhor RSSI entre dois pacotes consecutivos e eleito o que apresentasse a maior RSSI, para assim, armazená-lo no arquivo. Caso um dos pacotes fosse perdido, o

*script* armazenava o único que tinha sido recebida pelo *gateway*. Caso os dois pacotes fossem perdidos, é atribuído o valor de - 200 dBm àquela RSSI. Novas medidas foram feitas apenas nas partes mais distantes dos tanques 9, 10, 11, 12 e 13, pois foram os nós mais afetados. Para cada um desses nós, extraiu-se mais 120 medidas que resultaram em 60 pacotes armazenado nos arquivos após a execução do *script*.

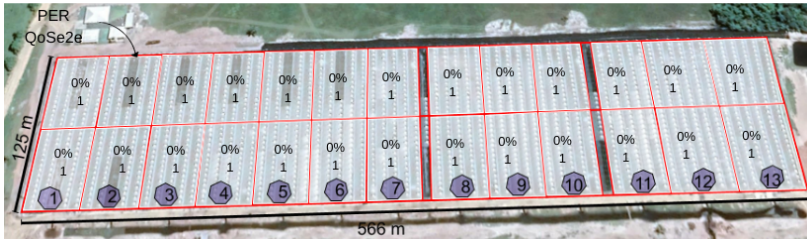


Figura 10 – PER e QoS e2e para cada nó do teste em campo usando a diversidade temporal. Fonte: Elaborado pelo autor.

Na Figura 10, pode-se constatar os resultados do QoS e2e e PER e2e da rede LPWAN. Note que todos os nós usados no teste de campo tiveram um QoS de 100%. Já na Figura 11, constata-se como que ficaram as RSSIs médias fazendo-se o uso da diversidade temporal. É possível notar que, comparando com a Figura 9, houve um aumento nas RSSIs médias, o que indica que, a diversidade temporal realmente pode ajuda a solucionar o problema de perda de pacotes e aumentar a RSSI média.

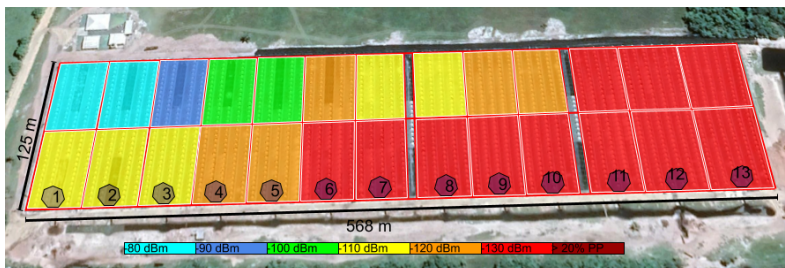


Figura 11 – RSSIs médias em forma de mapa de calor usando a diversidade temporal. Fonte: Elaborado pelo autor.

Outras técnicas como diversidade espacial (SANTOS, 2019), ele-



var os aparelhos para melhorar a linha de visada, SF maiores e etc, também melhorariam os resultados do teste de campo, mas por limitação imposta pelos aparelhos, não foi possível utilizar dois *gateways*, por questão de custo, alterar o SF dos *end-devices* ou instalar o *gateway* em um ponto mais alto. Mas, foi possível solucionar o problema e constatar a viabilidade da rede LPWAN apenas fazendo-se o uso da diversidade temporal para melhorar QoS, PER e RSSI.



## 5 CONCLUSÃO

Os resultados do experimento afirmam a viabilidade do uso da rede LPWAN e seu potencial em aplicações de monitoramento das variáveis aquáticas dentro dos viveiros intensivos de criação de camarão. Especificamente, foi alcançado um QoS de praticamente 100% para todos os pontos de coleta de dados de dentro das estufas de camarão, usando a diversidade temporal, provando que LPWAN baseada em LoRa® é promissor para o monitoramento do ambiente aquático de camarões em aplicações da piscicultura moderna.

Essa harmonia nas características de desempenho representa uma boa base para integrar LPWANs com as estufas de criação intensiva de camarão para se realizar o conceito de IoS. Em conclusão, isso faz do conceito IoS uma escolha razoável como tecnologia de rádio para aplicações de monitoramento das variáveis aquáticas, além de ser eficiente em termos de energia e escalável.

Como uma extensão deste trabalho, é planejado testar a interferência de diversos *end devices* espalhados por vários nós ao longo das estufas de criação. A devida disposição do *gateway*, assim como o potencial uso de múltiplos *gateways* para explorar diversidade espacial, também são pontos importantes de serem investigados.

Por fim, é importante fazer um teste de maior duração, para aumentar a relevância estatística dos dados, uma vez que devido às restrições de tempo o número de medidas realizadas foi bastante limitado. Porém, mesmo reconhecendo a reduzida significância estatística dos testes realizados neste trabalho, é evidente que a tecnologia é bastante promissora para a aplicação em questão.



## REFERÊNCIAS

- ADELANTADO, F. et al. Understanding the limits of lorawan. *IEEE — IEEE Communications Magazine* ( Volume: 55, Issue: 9 , Sept. 2017 ), p. 34 – 40, 2017. ISSN 17169467. <<https://ieeexplore.ieee.org/abstract/document/8030482>>. Acessado em 27/11/2019.
- AUGUSTIN, A. et al. A study of lora: long range and low power networks for the internet of things sensors. *Sensors (Switzerland)*, 2016. <<https://doi.org/10.3390/s16091466>>.
- BEVERIDGE, M. Meeting the food and nutrition needs of the poor: the role of fish and the opportunities and challenges emerging from the rise of aquaculture. *Journal of Fish Biology* 83(4):1067-1084, 2013.
- BJELLAND, H. et al. Exposed aquaculture in norway. *IEEE — OCEANS 2015 - MTS/IEEE Washington*, 2015. ISSN 15798843. <<https://doi.org/10.23919/OCEANS.2015.7404486>>.
- DUAN, R. et al. A qos architecture for iot. *IEEE — International Conference on Internet of Things and 4th International Conference on Cyber, Physical and Social Computing*, Dalian, China, 2011. ISSN 12526147. <<https://doi.org/10.1109/iThings/CPSCom.2011.125>>. Acessado em 25/11/2019.
- ECLIPSE<sup>TM</sup>. *Eclipse Mosquitto<sup>TM</sup> - An open source MQTT broker*. 2019. <<https://www.mosquitto.org/>>. Acessado em 30 nov. 2019.
- FAO. The future of food and agriculture. 2017. <<http://www.fao.org/publications/fofa/en/>>. Acessado em 05/08/2019.
- HASSAN, W. et al. Internet of fish: Integration of acoustic telemetry with lpwan for efficient real-time monitoring of fish in marine farms. *ELSEVIER — Computers and Electronics in Agriculture*, Volume 163, 2019. ISSN 104850. <<https://www.sciencedirect.com/science/article/pii/S0168169918314121>>. Acessado em 05/11/2019.
- HUNKELER, U. et al. Mqtt-s — a publish/subscribe protocol for wireless sensor networks. *3rd International Conference on*

*Communication Systems Software and Middleware* — IEEE, 2008. ISSN 10072012. <<https://doi.org/10.1109/COMSWA.2008.4554519>>. Acessado em 25/11/2019.

IVERSEN, A. et al. Definitions of terms related to quality of service. rapport 32/2013 nofima. 2013.

KHOMP. *ITG 200 Gateway de telemetria que capta e envia dados dos sensores*. 2019. <<https://www.khomp.com/pt/produto/itg-200/>>. Acessado em 27 nov. 2019.

KHOMP. *NIT 21 LI – Endpoint IoT transmissor Indoor, com sensor de temperatura e umidade integrado na placa (onboard)*. 2019. <<https://www.khomp.com/pt/produto/endpoint-lora/>>. Acessado em 27 nov. 2019.

LIGHT, R. Mosquitto: server and client implementation of the mqtt protocol. j.open sour. softw. 2. 2017. <<https://doi.org/10.21105/joss.00265>>. Acessado em 25/11/2019.

PETRIĆ, T. et al. Measurements, performance and analysis of lora fabian, a real-world implementation of lpwan. *IEEE* — 27th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC), pp. 1–7., 2016. ISSN 2326-8301. <<https://doi.org/10.1109/PIMRC.2016.7794569>>. Acessado em 25/11/2019.

PETÄJÄJÄRVI, J. et al. Evaluation of lora lpwan technology for remote health and wellbeing monitoring. *IEEE* — International Symposium on Medical Information and Communication Technology (ISMICT), 2016. ISSN 2326-8301. <<https://ieeexplore.ieee.org/document/7498898>>. Acessado em 25/11/2019.

PYTHON. *paho-mqtt 1.5.0*. 2019. <<https://www.pypi.org/project/paho-mqtt/>>. Acessado em 30 set. 2019.

RAZA, U. et al. Low power wide area networks: an overview. *IEEE Commun. Surv. Tutorials* 19, 855–873., 2017. <<https://doi.org/10.1109/COMST.2017.2652320>>. Acessado em 05/08/2019.

ROCHA, I. Cultivo do camarão marinho: Atividade socialmente justa, ambientalmente responsável e, economicamente importante, de forma especial para o meio rural da região nordeste. *ABCCAM*,

2018. <<https://abccam.com.br/2015/05/cultivo-do-camarao-marinho-atividade-socialmente-justa-ambientalmente-responsavel-e-economicamente-importante-de-forma-especial-para-o-meio-rural-da-regiao-nordeste-2/>>. Acessado em 05/08/2019.

SANTOS, E. Análise do impacto de técnicas de diversidade na comunicação entre dispositivos lora/lorawan. UFSC, 2019. <<https://repositorio.ufsc.br/handle/123456789/196225>>. Acessado em 16/08/2019.

TALAVERA, J. et al. Review of iot applications in agro-industrial and environmental fields. *comput. electron. agric.* 142, 283–297. 2017. <<https://doi.org/10.1016/j.compag.2017.09.015>>. Acessado em 20/11/2019.





## APÊNDICE A - Códigos Python



*Script mqtt.subscriber.py; Programa usado para extrair os dados do broker.*

```

1 # Developed by: Cassiano Candido.
2 # Contact: candidocassiano2@gmail.com
3 import paho.mqtt.client as mqtt
4 import json
5 import datetime
6 import time
7
8 # parametros
9 broker = "mqtt.eclipse.org"
10 broker_port = 1883
11 keep_alive_broker = 60
12 subscribe_topic = "itg200/0000f8033201a70a/measurement"
13
14 # inicializa MQTT:
15 client = mqtt.Client()
16 now = datetime.datetime.now()
17
18 count = 0
19 stak = 0
20
21 def init_broker_conection():
22     print("[STATUS] Inicializando MQTT...")
23     client.on_connect = on_connect
24     client.on_message = on_message
25     client.connect(broker, broker_port, keep_alive_broker)
26     client.loop_start()
27     global count
28     while count < 120:
29         time.sleep(60)
30         print("[STATUS] Amostras coletadas. Encerrando conexao
31             ...")
32         client.disconnect()
33         print("[STATUS] Disconectado. Programa encerrado.")
34         client.loop_stop()
35
36 # Extrai parametros da mensagem recebida no broker
37 def get_message_informations(message):
38     parameters_dict = json.loads(message)
39     return parameters_dict
40
41 # Callback - conexao ao broker realizada
42 def on_connect(client, userdata, flags, rc):
43     print("[STATUS] Conectado ao Broker. Resultado de
44         conexao: " + str(rc))
45     # topic subscribe
46     client.subscribe(subscribe_topic)
47
48 # Callback - mensagem recebida do broker

```

```

49 def on_message(client, userdata, msg):
50     print("[MSG RECEBIDA] Topico: " + msg.topic)
51     received_message = get_message_informations(str(msg.
52     payload))
53     rssi = str(received_message[2][u'v'])
54     temp = str(received_message[3][u'v'])
55     humi = str(received_message[4][u'v'])
56     stamp= time.gmtime(float(str(received_message[0][u'bt'])
57    [:-6] + '.' + str(received_message[0][u'bt'])[:-6:]))
58     minu = int(stamp.tm_min)
59     print("RSSI=" + rssi + " dBw" + " temperature = " +
60     temp + " C" + " humidity = " + humi + " %")
61     file = open("log.txt", "a")
62     file.write("Sensor 01: " + str(stamp.tm_hour) + ":" +
63     str(stamp.tm_min) + ":" + str(stamp.tm_sec) + " RSSI ="
64     + rssi + "dBw" + " temperature = " + temp + "*C" + "
65     humidity= " + humi + "%" + "\n" )
66     file.close()
67     file = open("log_plot_datas.txt", "a")
68     global count
69     global stak
70     if count == 0:
71         stak = minu - 1
72         stakt = stak + 1
73         if stakt == 60:
74             file.write(str(count) + "," + str(stamp.tm_min) + "," +
75             rssi + "," + temp + "," + humi + "\n")
76             file.close()
77             count += 1
78         elif minu == stakt:
79             file.write(str(count) + "," + str(stamp.tm_min) + "," +
80             rssi + "," + temp + "," + humi + "\n")
81             file.close()
82             count += 1
83         else:
84             dif = minu - stakt
85             for i in range(dif):
86                 file.write(str(count) + "," + "7777" + "," + "-200" + "
87                 , " + temp + "," + humi + "\n")
88                 count += 1
89             file.write(str(count) + "," + str(stamp.tm_min) + "," +
90             rssi + "," + temp + "," + humi + "\n")
91             count += 1
92             file.close()
93             stak = minu
94
95 # programa principal:
96 if __name__ == '__main__':
97     init_broker_conection()

```

*Script plot.py; Programa usado para traçar os gráficos.*

```

# Developed by: Cassiano Candido.
2 # Contact: candidocassiano2@gmail.com

4 import matplotlib.pyplot as plt
import numpy as np

6
x, y, z, t, u = np.loadtxt('log_plot_datas.txt', delimiter =
',', unpack= True)
8 k = len(x)
m = 0
10 mt = 0
mu = 0
12 c = 0
zdois = []
14 tdois = []
udois = []
16 for i in range(k):
    if z[i] != -200:
18         c += 1
                mt = mt + t[i]
20         mu = mu + u[i]
                m = m + z[i]
22 r = m/c
rt = mt/c
24 ru = mu/c
for i in range(k):
26     zdois.append(r)
        tdois.append(rt)
28     udois.append(ru)

30 fig1 = plt.figure()
fig2 = plt.figure()
32
    rssi = fig1.add_subplot(111)
34 temp = fig2.add_subplot(211)
umid = fig2.add_subplot(212)
36
    rssi.plot(x, z)
38 rssi.plot(x, zdois, label='m dia= ' + str("{:.2f}".format(r
)))
temp.plot(x, t, color="r")
40 temp.plot(x, tdois, label='m dia= ' + str("{:.2f}".format(
rt)))
umid.plot(x, u)
42 umid.plot(x, udois, label='m dia= ' + str("{:.2f}".format(
ru)))

44 rssi.legend()
temp.legend()
46 umid.legend()

```

```

48 rssi.set_title("RSSI")
temp.set_title("TEMPERATURA")
50 umid.set_title("UMIDADE")

52 rssi.set_xlabel("Amostras")
rssi.set_ylabel("rssi [dBm]")
54
56 temp.set_xlabel("Amostras")
temp.set_ylabel("Temperatura [ C ]")

58 umid.set_xlabel("Amostras")
umid.set_ylabel("Umidade [%]")
60
62 fig2.subplots_adjust(hspace=0.5)
64
fig1.savefig("rssi_plot.png")
fig2.savefig("temp_umid_plot")
plt.show()

```

*Script* mqtt.subscriber.diversidade.py; Programa usado para acessar o *broker* aplicando a diversidade temporal.

```

1 # Developed by: Cassiano Candido.
# Contact: candidocassiano2@gmail.com
3
import paho.mqtt.client as mqtt
5 import json
import datetime
7 import time

9 # parametros
broker = "mqtt.eclipse.org"
11 broker_port = 1883
keep_alive_broker = 60
13 subscribe_topic = "itg200/0000f8033201a70a/measurement"

15 # inicializa MQTT:
client = mqtt.Client()
17 now = datetime.datetime.now()

19 count = 0
cd = 1
21 stak = 0
pum = 0
23 pdois = 0

25 def init_broker_conection():
print("[STATUS] Inicializando MQTT...")
27 client.on_connect = on_connect

```

```

client.on_message = on_message
29 client.connect(broker, broker_port, keep_alive_broker)
client.loop_start()
31 global count
while count < 60:
33     time.sleep(60)
print("[STATUS] Amostras coletadas. Encerrando conexao
...")
35 client.disconnect()
print("[STATUS] Desconectado. Programa encerrado.")
37 client.loop_stop()

39 # Extrai parametros da mensagem recebida no broker
41 def get_message_informations(message):
    parameters_dict = json.loads(message)
43     return parameters_dict

45 # Callback - conexao ao broker realizada
def on_connect(client, userdata, flags, rc):
47     print("[STATUS] Conectado ao Broker. Resultado de
conexao: " + str(rc))
    # topic subscribe
49     client.subscribe(subscribe_topic)

51 # Callback - mensagem recebida do broker
def on_message(client, userdata, msg):
53     print("[MSG RECEBIDA] Topico: " + msg.topic)
received_message = get_message_informations(str(msg.
payload))
55     rssi = str(received_message[2][u'v'])
stamp= time.gmtime(float(str(received_message[0][u'bt'])
[: -6] + '.' + str(received_message[0][u'bt'])[-6:]))
57     minu = int(stamp.tm_min)
global cd, pum, pdois, count, stak
59     file = open("log_dup.temp_datas.txt", "a")
if count == 0:
61         stak = minu - 1
stakt = stak + 1
63     if stakt == 60:
        if cd == 1:
65             pum = rssi
cd = 2
67         else:
            pdois = rssi
69             if pdois < pum:
                file.write(str(count) + "," + "2" + "," +
str(pdois) + "\n")
71             print("pacote = " + "2" + "   RSSI = " + str(
pdois) + " dBw")
cd = 1
73             count += 1

```

```

75         else:
76             file.write(str(count) + "," + "1" + "," +
str(pum) + "\n")
77             print("pacote = "+ "1" + "  RSSI =" + str(
pum) + " dBw")
78             cd = 1
79             count += 1
80     elif minu == stakt:
81         if cd == 1:
82             pum = rssi
83             cd = 2
84         else:
85             pdois = rssi
86             if pdois < pum:
87                 file.write(str(count) + "," + "2" + "," +
str(pdois) + "\n")
88                 print("pacote = "+ "2" + "  RSSI =" + str(
pdois) + " dBw")
89                 cd = 1
90                 count += 1
91             else:
92                 file.write(str(count) + "," + "1" + "," +
str(pum) + "\n")
93                 print("pacote = "+ "1" + "  RSSI =" + str(
pum) + " dBw")
94                 cd = 1
95                 count +=1
96     else:
97         dif = minu - stakt
98         div = dif/2
99         if cd == 1:
100             if div == 0:
101                 file.write(str(count) + "," + "2" + "," +
rssi + "\n")
102                 print("pacote =2" + "  RSSI =" + rssi + "
dBw")
103                 count += 1
104             else:
105                 for i in range(div):
106                     file.write(str(count) + "," + "0" + "," +
+ "-200" + "\n")
107                     print("pacote = "+ "perdido" + "  RSSI =
" + "-200" + " dBw")
108                     count += 1
109                     pum = rssi
110                     cd = 2
111             else:
112                 file.write(str(count) + "," + "1" + "," + str(
pum) + "\n")
113                 print("pacote = "+ "1" + "  RSSI =" + str(pum) +
" dBw")
114                 count += 1

```



```

115         for i in range(div):
            file.write(str(count) + "," + "0" + "," + "
-200" + "\n" )
            print("pacote = " + "0" + "   RSSI = " + "-200"
+ " dBw")
117             count += 1
                pum = rssi
119                 cd = 2
                    stak = minu
121                     file.close()

123 # programa principal:
124 if __name__ == '__main__':
125     init_broker_conection()

```

*Script* plot.diversidade.py; Programa usado para traçar os gráficos usando a diversidade temporal.

```

1 # Developed by: Cassiano Candido.
# Contact: candidocassiano2@gmail.com
3
4 import matplotlib.pyplot as plt
5 import numpy as np
6
7 x, y, z, t, u = np.loadtxt('13.1_log_plot_datas.txt',
8                             delimiter=',',unpack= True)
9
10 zdiv = []
11 xx = []
12 c = 0
13 s = 0
14 m = 0
15 pp = 0
16 sos = 0
17 ppp = 0
18 zc = 0
19 zdois = []
20 zdivd = []
21
22 for i in range(0,len(z),2):
23     if z[i] > z[i+1]:
24         zdiv.append(z[i])
25         xx.append(c)
26         c += 1
27     else:
28         zdiv.append(z[i+1])
29         xx.append(c)
30         c += 1
31
32 for i in range(len(z)):

```

```

33     if z[i] != -200:
34         s += 1
35         m = m + z[i]
36     else:
37         pp += 1
38 r = m/s
39 for i in range(len(zdiv)):
40     if zdiv[i] != -200:
41         sos += 1
42         zc = zc + zdiv[i]
43     else:
44         ppp += 1
45 rm = zc/sos
46
47 for i in range(len(z)):
48     zdois.append(r)
49
50 for i in range(len(zdiv)):
51     zdivd.append(rm)
52
53 fig1 = plt.figure()
54 fig2 = plt.figure()
55
56 rssi = fig1.add_subplot(111)
57 rssid = fig2.add_subplot(111)
58
59 rssi.plot(x, z, label="pacotes perdidos= " + str(pp))
60 rssi.plot(x, zdois, label='m dia= ' + str("{:.2f}".format(r
61 ))+"dBm")
62 rssid.plot(xx, zdiv, label = "pacotes perdidos= " + str(ppp)
63 )
64 rssid.plot(xx, zdivd, label="m dia= " + str("{:.2f}".format
65 (rm)) + "dBm")
66
67 rssi.legend()
68 rssid.legend()
69
70 rssi.set_title("RSSI")
71 rssid.set_title("DIVERSIDADE TEMPORAL")
72
73 rssi.set_xlabel("Amostras")
74 rssi.set_ylabel("rssi [dBm]")
75
76 rssid.set_xlabel("Amostras")
77 rssid.set_ylabel("rssi [dBm]")
78
79 fig1.savefig("rssi_plot.png")
80 fig2.savefig("diversidadetemp.png")
81 plt.show()

```

## APÊNDICE B – Figuras extras





Figura 12 – Interior de uma estufa de carcinicultura vazia. Fonte: Elaborado pelo autor.



Figura 13 – Interior de uma estufa de carcinicultura em operação. Fonte: Elaborado pelo autor.



Figura 14 – Fermentação dos bioflocos antes de adicioná-los à água.  
Fonte: Elaborado pelo autor.



Figura 15 – Torre para *Internet* via satélite. Fonte: Elaborado pelo autor



Figura 16 – Foto do *end-device* LoRa® durante as medidas. Fonte: Elaborado pelo autor.