INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA FLUMINENSE

PROGRAMA DE PÓS-GRADUAÇÃO EM SISTEMAS APLICADOS À ENGENHARIA E GESTÃO

EVERTON ALVES MIRANDA

SISTEMA MICROCONTROLADO EMBARCADO EM UNIDADE FLUTUANTE PARA MONITORAMENTO AUTOMÁTICO DE FITOPLÂNCTON E DE MÚLTIPLAS TEMPERATURAS

Campos dos Goytacazes, RJ

2019

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA FLUMINENSE

PROGRAMA DE PÓS-GRADUAÇÃO EM SISTEMAS APLICADOS À ENGENHARIA E GESTÃO

EVERTON ALVES MIRANDA

SISTEMA MICROCONTROLADO EMBARCADO EM UNIDADE FLUTUANTE PARA MONITORAMENTO AUTOMÁTICO DE FITOPLÂNCTON E DE MÚLTIPLAS TEMPERATURAS

Renato Gomes Sobral Barcellos
(Orientador)

Luiz Gustavo Lourenço Moura
(Coorientador)

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação do Instituto Federal de Educação, Ciência e Tecnologia Fluminense, no Curso de Mestrado Profissional em Sistemas Aplicados à Engenharia e Gestão (MPSAEG), como parte dos requisitos necessários à obtenção do título de Mestre em Sistemas Aplicados à Engenharia e Gestão.

Campos dos Goytacazes, RJ
Fevereiro de 2019

Biblioteca Anton Dakitsch
CIP - Catalogação na Publicação

M623s

Miranda, Everton Alves
Sistema microcontrolado embarcado em unidade flutuante para monitoramento automático de fitoplâncton e de múltiplas temperaturas / Everton Alves Miranda - 2019.
92 f.: il. color.

Orientador: Renato Gomes Sobral Barcellos
Coorientador: Luiz Gustavo Lourenço Moura

Dissertação (mestrado) -- Instituto Federal de Educação, Ciência e Tecnologia Fluminense, Campus Campos Centro, Curso de Mestrado Profissional em Sistemas Aplicados à Engenharia e Gestão, Campos dos Goytacazes, RJ, 2019.
Referências: f. 64 a 67.

1. Fotofluorescência da clorofila. 2. Estratificação da temperatura. 3. Fitoplâncton. 4. Sistema embarcado. 5. Monitoramento ambiental. I. Barcellos, Renato Gomes Sobral, orient. II. Moura, Luiz Gustavo Lourenço, coorient. III. Título.

Elaborada pelo Sistema de Geração Automática de Ficha Catalográfica da Biblioteca Anton Dakitsch do IFF com os dados fornecidos pelo(a) autor(a).

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA FLUMINENSE
PROGRAMA DE PÓS-GRADUAÇÃO EM SISTEMAS APLICADOS À ENGENHARIA E GESTÃO

EVERTON ALVES MIRANDA

SISTEMA MICROCONTROLADO EMBARCADO EM UNIDADE FLUTUANTE PARA MONITORAMENTO AUTOMÁTICO DE FITOPLÂNCTON E DE MÚLTIPLAS TEMPERATURAS

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação do Instituto Federal de Educação, Ciência e Tecnologia Fluminense, no Curso de Mestrado Profissional em Sistemas Aplicados à Engenharia e Gestão (MPSAEG), como parte dos requisitos necessários à obtenção do título de **Mestre** Sistemas Aplicados à Engenharia e Gestão.

Aprovado em 25 de fevereiro de 2019.

Banca Examinadora:

Renato Gomes Sobral Barcellos, Doutor em Geociência.
Instituto Federal de Educação, Ciência e Tecnologia Fluminense (Orientador)

Luiz Gustavo Lourenço Moura, Doutor em Engenharia de Sistemas e Computação.
Instituto Federal de Educação, Ciência e Tecnologia Fluminense (Co-orientador)

Milton Erthal Júnior, Doutor em Produção Vegetal.
Instituto Federal de Educação, Ciência e Tecnologia Fluminense

William da Silva Vianna, Doutor em Engenharia e Ciência dos Materiais.
Instituto Federal de Educação, Ciência e Tecnologia Fluminense

Julio César de Faria Alvim Wasserman, Doutor em Oceanologia
Universidade Federal Fluminense

À minha esposa Claudia Márcia e aos meus filhos Heitor e Ítalo, como um pequeno reconhecimento por todos os muitos ônus que lhes foram impostos no período da realização deste curso e, em especial, durante a elaboração deste trabalho.

AGRADECIMENTOS

Ao professor Renato Gomes Sobral Barcellos, pelo seu conhecimento, sua orientação intelectual, seu apoio e empenho na viabilização das condições necessárias para a elaboração deste trabalho.

Ao professor Luiz Gustavo Lourenço Moura, por atuar com coorientador deste trabalho.

Aos professores Mara de Menezes de Assis Gomes e Eliemar Campostrini, do Laboratório de Melhoramento Genético Vegetal da Universidade Estadual do Norte Fluminense Darcy Ribeiro (UENF), pelo apoio em um dos momentos mais cruciais da fase experimental dessa dissertação.

Ao Comitê Baixo Paraíba do Sul e Itabapoana e a AGEVAP, pelo apoio financeiro concedido através do Edital Nº 004/2018.

Aos meus gestores da TRANSPETRO Diego Firmino Pereira e Patrícia Duplan Pacheco, por sua postura ética, solidária e comprometida com o desenvolvimento profissional de sua equipe, e por apoiar e prover as condições logísticas, as quais viabilizaram o atendimento às exigências deste curso.

Ao meu amigo José Rogério dos Santos Neves, pelo apoio na manutenção dos equipamentos eletrônicos.

Aos meus amigos, Daniel Alves Gomes, Marcelo Ferreira Rangel e Diego Vidal Bueno, pelas frequentes conversas, dicas, instruções e inspirações.

A minha esposa pelo constante incentivo e inabalável persistência.

E finalmente, aos meus pais Jorge Miranda Cerqueira e Erenice Alves Miranda, não só por todas as orações e incentivos neste período, mas também pelo constante exemplo de caráter, coragem e garra durante toda a minha vida.

RESUMO

Introdução: O suprimento de água potável para consumo é baseado no tratamento de água bruta oriunda de mananciais frequentemente degradados, estando sujeitos à ocorrência de um crescimento acelerado do fitoplâncton, o qual, por sua vez, é influenciado pela variação de temperatura. Os produtos metabólicos decorrentes de algumas de suas espécies produzem toxinas capazes de causar distúrbios ao ser humano, afetando assim, a saúde pública. Esse crescimento acelerado provoca também, consequências negativas sobre a eficiência e o custo do tratamento da água. **Objetivo:** Desenvolver um sistema embarcado em unidade flutuante, para monitoramento de reservatórios e mananciais hídricos, visando à estimativa do volume do fitoplâncton, na camada superficial, e o monitoramento da temperatura em múltiplas profundidades. **Revisão Bibliográfica:** Os sistemas lagunares costeiros caracterizam-se em geral por baixas profundidades da coluna d'água onde ocorrem importantes processos geoquímicos como eutrofização, floração algal e ciclagem de nutrientes. A temperatura decai exponencialmente em função da profundidade, sendo um fator determinante na ocorrência destes processos. O monitoramento do crescimento de fitoplâncton *in situ* é caracterizado como uma problemática básica na ciência hidrológica. **Metodologia:** realização de uma pesquisa bibliográfica, sobre o desenvolvimento de sensores e de técnicas para estimativa de volume fitoplanctônico, seguida de uma etapa experimental envolvendo o desenvolvimento do sistema (hardware e software), a sua calibração e a realização dos testes de desempenho. **Resultados:** A comparação das especificações técnicas fundamentou a escolha do ESP8266 como microcontrolador e do DS18B20 como elemento sensor de temperatura. O sensor de fotofluorescência construído inspirou-se em orientações disponíveis na literatura. Foi modelado, projetado, construído e programado o protótipo do sistema embarcado. Durante a calibração do sensor construído, obteve-se um R^2 de 0,90. Os testes de campo foram realizados em dois eventos em locais e horários diferentes atingindo os resultados esperados. **Conclusão:** Foi realizado um projeto de desenvolvimento que materializou os conhecimentos em uma aplicação específica, a qual envolveu o desenvolvimento, a prática construtiva e a utilização do produto. O protótipo construído apresentou-se com um equipamento reprodutível e econômico, atendendo aos objetivos propostos. Este projeto terá continuidade, devido a sua aprovação pela AGEVAP (Edital Nº 004/2018) e ao já concedido auxílio financeiro.

Palavras-chave: Fotofluorescência da Clorofila, Estratificação da temperatura, Fitoplâncton, Sistema Embarcado, Monitoramento Ambiental.

ABSTRACT

Introduction: The supply of drinking water for consumption is based on the treatment of raw water. The latter comes from frequently degraded wellsprings, subject to the event of an accelerated growth of phytoplankton, which, in turn, is influenced by the temperature variation. The metabolic products derived from some of its species produce toxins capable of causing disruption to the human being, thus affecting public health. This accelerated growth also has negative consequences on the efficiency and cost of water treatment. **Objective:** To develop a floating unit for the monitoring of water reservoirs and wellsprings, aiming at the estimation of phytoplankton volume, in the surface layer, and temperature monitoring at multiple depths. **Bibliographical Review:** Coastal lagoon systems are generally characterized by low depths of the water column where important geochemical processes such as eutrophication, algal bloom and nutrient cycling occur. The temperature decreases exponentially as a function of depth, being a determining factor in the occurrence of these processes. The monitoring of *in situ* phytoplankton growth is characterized as a basic problem in hydrological science. **Methodology:** a bibliographic research on the development of sensors and techniques for estimating phytoplankton volume, followed by an experimental stage involving the development of the system (hardware and software), its calibration and performance tests. **Results:** Taking base the comparison of the technical specifications was choosed of the ESP8266 as the microcontroller and the DS18B20 as the temperature sensing element. The built-in photofluorescence sensor has been inspired by guidelines available in the literature. The prototype of the embedded system was modeled, designed, built and programmed. During the calibration of the constructed sensor, an R^2 of 0.90 was obtained. The field tests were carried out in two events at different times and places reaching the expected results. **Conclusion:** A development project was carried out that materialized the knowledge in a specific application, which involved the development, the constructive practice and the use of the product. The built prototype was presented with a reproducible and economical equipment, meeting the proposed objectives. This project will continue, due to its approval by AGEVAP (Call Notice No. 004/2018) and to the already granted financial aid.

Keywords: Chlorophyll Photofluorescence, Temperature Stratification, Phytoplankton, Embedded System, Environmental Monitoring.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE CÓDIGOS

LISTA DE SIGLAS

AES	*Advanced Encryption Standard*
AGEVAP	Associação Pró-Gestão das Águas da Bacia Hidrográfica do Rio Paraíba do Sul
BPMN	*Business Process Model and Notation*
bps	Bits por segundo
DIY	*do-it-yourself*
GPIO	*General Purpose Input/output*
GPS	*Global Positioning System*
HPLC	*High performance liquid chromatography*
I2C	*Inter-Integrated Circuit*
I2S	*inter-integrated circuit Sound*
IDE	*Integrated Development Environment*
IoT	*Internet of Things*
LED	*Light Emitting Diode*
LIDAR	*Light Detection And Ranging*
MIT	*Massachusetts Institute of Technology*
PCB	*Printed circuit board*
PUC	Pontifícia Universidade Católica
PWM	Modulação por largura de pulso (*Pulse-Width Modulation*)
OTA	*OVER THE AIR*
RF	*Radio frequency*
RGB	Vermelho (*Red*), Verde (*Green*) e Azul (*Blue*)
SPI	*Serial Peripheral Interface*
TKIP	*Temporal Key Integrity Protocol*
UART	*Universal Asynchronous Receiver/Transmitter*
URI	*Uniform Resource Identifier*
VCC	Volts de Corrente Contínua
WEP	*Wired Equivalent Privacy*
Wi-Fi	*Wireless Fidelity*
WPA/WPA2	*Wi-Fi Protected Access*

SUMÁRIO

1 INTRODUÇÃO

1.1 Contextualização

O gerenciamento de recursos é um dos principais assuntos relacionados à sustentabilidade para a sociedade. A Lei Federal 9433/97, que instituiu a Política Nacional de Recursos Hídricos, dota a água de valor econômico, caracterizando-a como um recurso finito e de domínio público, sendo as bacias hidrográficas as unidades de planejamento e gestão, tanto para ações de promoção, quanto de proteção (BRASIL, 1997).

O suprimento de água potável para consumo nos centros urbanos, geralmente é baseado no tratamento de água bruta oriunda de rios, lagos e represas. Estes mananciais sofrem com as ações antrópicas como: a ocupação irregular em seu entorno, o despejo de esgotos, a pesca predatória, a destruição da mata ciliar, dentre outras. O processo de eutrofização antrópico, há muito tempo, representa uma preocupação no que tange a conservação dos recursos hídricos, principalmente, quando se trata de abastecimento.

A eutrofização é um processo natural que ocorre em muitos mananciais. Sua ocorrência é devida a um nível excessivo de matéria orgânica na água, gerando impactos à biota aquática e danos à qualidade da água.

A eutrofização antrópica, está mais associada aos corpos hídricos localizados próximos as áreas urbanas, pois, nestes ambientes, a poluição difusa e os despejos industriais e residenciais agravam o problema. Na zona rural, o processo está relacionado às atividades agrícolas, associadas ao uso de agrotóxicos e fertilizantes (orgânicos ou sintéticos), os quais sofrem processo de lixiviação, gerando um aporte adicional de nutrientes (especialmente fósforo e nitrogênio) para as populações de algas (BRAGA *et al.*, 2005, p. 97; MOTA, 2003, p. 173).

Em muitos mananciais ocorre a predominância de grandes períodos de retenção da água, gerando um favorecimento ao acúmulo de matéria orgânica e sedimentos. Fato este que, ocasiona um crescimento acelerado da comunidade fitoplanctônica, provocando consequências negativas que incidem sobre a eficiência e o custo do tratamento da água para abastecimento. Além disso, os produtos metabólicos secretados por algumas espécies do fitoplâncton conferem odor e sabor à água. Outras produzem toxinas capazes de causar distúrbios gastrointestinais, respiratórios, neurológicos e alergênicos ao ser humano (BERNARDO, 1995, p. 31; BRAGA *et al.*, 2005, p. 97; BROOKE *et al.*, 2008; MOTA, 2003, p. 173).

A detecção do volume fitoplanctônico é necessária não só para descrever as tendências de longo prazo, mas também, para a implementação de medidas para proteger a saúde pública (CULLEN *et al.*, 1997).

Partindo do princípio de que a clorofila *a* é o principal pigmento fotossintético do fitoplâncton, a mesma vem sendo amplamente utilizada para estimar o volume de biomassa relacionada ao mesmo, sendo o método da medição por fotofluorescência, um dos mais aplicados para esta finalidade (INAG, 2009).

Os sistemas lagunares costeiros caracterizam-se, em geral, por um ambiente de baixa energia, baixas profundidades da coluna d'água e grande aporte de nutrientes provenientes das bacias hidrográficas. A temperatura varia em função da profundidade da coluna d'agua, se apresentando como um fator determinante, na cinética de processos geoquímicos, como a eutrofização, a floração algal e a ciclagem de nutrientes (BRAGA *et al.*, 2005, p. 75–76; 94).

1.2 Objetivos

O objetivo geral deste trabalho é desenvolver um sistema embarcado em unidade flutuante visando o monitoramento, tanto da fotofluorescência (relacionada ao volume do fitoplâncton) presente na camada superficial, quanto da temperatura em múltiplas profundidades de reservatórios e lagoas costeiras.

Como objetivos específicos têm-se:

- Construir um sensor ótico para medição da fotofluorescência relacionada à dimensão da comunidade fitoplanctônica;
- Construir um sistema de medição da temperatura em múltiplas profundidades;
- Desenvolver um sistema automático de coleta e armazenamento de dados relacionados ao volume do fitoplâncton e às temperaturas nos diferentes estratos de mananciais hídricos;
- Desenvolver um sistema embarcado em uma unidade flutuante que possa ser utilizada em deriva ou fundeada para monitoramento de reservatórios e lagoas costeiras;

- Demonstrar a utilização do sistema como uma ferramenta de monitoramento ambiental com domínio tecnológico nacional.

1.3 Justificativa

As técnicas de microscopia para a observação dos organismos fitoplanctônicos exigem diversos procedimentos prévios de preparação da amostra para a sua correta visualização, por isso, são tecnicamente exigentes, morosas e caras (INAG, 2009). Este processo demorado e dispendioso introduz tempo e atraso, colocando limites práticos sobre o número de medidas que podem ser obtidas em um determinado corpo hídrico. Além disso, são sujeitas a muitos erros de interpretação, assim como, às incertezas que podem ser introduzidas à medida que as amostras sofrem degradação durante o transporte para o laboratório (MULLER; CYBIS; RAYA-RODRIGUEZ, 2011; NG; SENFT-GRUPP; HEMOND, 2012).

Outra realidade é o fato de que conjuntos de dados de baixa densidade, obtidos através da amostragem manual, podem não capturar adequadamente a variabilidade espaciotemporal, a qual, por vezes, pode conter a chave para a compreensão dos processos biogeoquímicos nos corpos aquáticos (BEUTLER *et al.*, 2002; NG; SENFT-GRUPP; HEMOND, 2012).

Conforme Murphy *et al.* (2015), A natureza de baixo custo de um dispositivo de monitoramento permite que vários sensores possam ser implantados em uma grande área, com alta densidade espacial. Alternativamente, os eventos detectados por uma rede desses sensores poderiam informar a localização e o momento de cada amostragem para a análise dos dados.

Mediante ao exposto, o desenvolvimento de um sistema de medição e coleta automática de dados para ambientes aquáticos, o qual tenha a capacidade de associar o monitoramento da variação das temperaturas em múltiplas profundidades, à estimativa do volume fitoplanctônico e que, além disso, consiga incrementar a quantidade de amostras, sistematizando a distribuição das mesmas em relação ao período de campanha e aos intervalos de tempo adequados, apresenta-se como importante ferramenta de auxílio às pesquisas e aos estudos de gerenciamento de recursos hídricos.

2 REVISÃO BIBLIOGRÁFICA

2.1 Fitoplâncton e métodos para sua estimativa.

O fitoplâncton é composto por organismos unicelulares microscópicos com capacidade fotossintética que vivem em suspensão na coluna de água e que podem ser solitários ou estar agrupados na forma de colônias, apresentando diferentes requisitos e respostas a parâmetros físicos e químicos como: a luz, a temperatura, a alcalinidade e a concentração de nutrientes (INAG, 2009).

Esse crescimento excessivo das plantas aquáticas, em níveis capazes de causar interferências nos usos desejáveis do copo de água é chamado de eutrofização. A mesma provoca consequências negativas, tanto sobre a eficiência, quanto sobre o custo do tratamento da água para abastecimento, podendo chegar a uma situação de inviabilidade econômica, culminando até na perda do manancial (BROOKE *et al.*, 2008). Sua detecção é necessária não só para descrever a frequência de florescências e as tendências de longo prazo, mas também, para a proteção dos locais de aquicultura e para a implementação de medidas para proteger a saúde pública (CULLEN *et al.*, 1997).

A eutrofização natural é um processo bastante demorado, associado ao tempo de evolução dos ecossistemas, porém, devido à ocupação de atividades industriais, agrícolas ou zonas urbanas, este processo tem se acelerado. Essa eutrofização associada à intervenção humana é denominada eutrofização cultural ou acelerada (BRAGA *et al.*, 2005, p. 97) ou antrópica.

Devido à dependência da luz solar, o fitoplâncton otimiza o tempo de residência nos estratos superiores da coluna de água, através de diversas estruturas ou mecanismos. Apresenta ciclos de vida curtos (4/5 dias) e obtêm os nutrientes necessários para o seu desenvolvimento diretamente da coluna de água, caracterizando-se como um indicador biológico de alterações da concentração de nutrientes na coluna de água e de pressões associadas ao processo de eutrofização (INAG, 2009).

Os lagos localizados em regiões tropicais, devido à predominância de temperaturas mais elevadas e de maior irradiação solar, assim como, aqueles de pequena profundidade (devido à maior influência da referida irradiação), possuem uma maior tendência à eutrofização (BRAGA *et al.*, 2005, p. 97).

Conforme expressos em Brooke (BROOKE *et al*., 2008), a captação de água bruta destinada ao posterior tratamento para consumo humano é, frequentemente, realizada em mananciais caracterizados pela ocorrência de grandes períodos de retenção da água. Essa realidade favorece o acumulo de poluentes, e consequentemente, a aceleração do crescimento da comunidade fitoplanctônica. Os produtos metabólicos decorrentes de algumas espécies de fitoplâncton conferem odor e sabor à água, enquanto outras produzem toxinas capazes de causar distúrbios gastrointestinais, respiratórios, neurológicos e alergias ao ser humano. (BERNARDO, 1995, p. 31; BRAGA *et al.*, 2005, p. 97; MOTA, 2003, p. 173).

No fitoplâncton existem diversos pigmentos fotossintéticos, como as clorofilas *a*, *b* e *c*, os carotenos, entre outros. A clorofila *a* é o principal pigmento fotossintético de todos os organismos que realizam fotossíntese com libertação de oxigénio, sendo amplamente utilizada para estimar a biomassa fitoplanctônica nas águas doces superficiais (INAG, 2009).

Segundo Teixeira (TEIXEIRA, 1973), as primeiras tentativas para avaliar e medir a produção primária do fitoplâncton (neste caso, no ambiente marinho) foram feitas nas décadas de 1920 e 1930. No início da década de 1970, estes estudos apresentaram uma considerável expansão, sendo caracterizado como uma problemática básica na ciência marinha. Porém, neste mesmo período, os estudos referentes à produção primária ainda eram muito raros, quando se tratava de regiões internas (de natureza estuarina ou de complexo estuarino-lagunar), marginados por manguezais. O grau de variabilidade nessas regiões requer uma instrumentação flexível capaz de analisar, detectar e monitorar as substâncias de interesse, quase em tempo real (MASSERINI JR. *et al.,* 2017).

Conforme Kim *apud* (BARBOSA, 2003), a utilização de um LIDAR (*Light Detection And Ranging*), instalado em um helicóptero, para monitoramento da clorofila *a* no oceano, data de 1973. Um LIDAR é basicamente composto por um laser (como fonte emissora de luz), um sistema de detecção, o qual inclui um telescópio e equipamento optoeletrônico de alta sensibilidade para discriminação/detecção de fótons, assim como, um sistema de hardware e software para controle e sincronismo do conjunto microprocessado de aquisição de dados.

Muller, Cybis e Raya-Rodriguez (2011), relatam que a análise e a quantificação do fitoplâncton podem ser realizadas por diversos métodos, existindo métodos microscópicos, métodos moleculares, assim como, métodos que utilizam a fluorescência e métodos envolvendo contagem de partículas. Segundo os referidos autores, os métodos através de microscopia óptica estão entre os mais utilizados, destacando-se o método de Sedgwick-Rafter (SR) e o de Utermöhl, sendo que o primeiro apresenta um custo menor (pois não há necessidade de microscópio específico) e fornece resultados em menor período de tempo, no entanto, ambos os métodos estão sujeitos a muitos erros de interpretação e consomem um tempo considerável.

As técnicas de microscopia para a observação dos organismos fitoplanctônicos exigem diversos procedimentos prévios de preparação da amostra para a sua correta visualização, por isso, são tecnicamente exigentes, morosas e caras (INAG, 2009). Este processo é demorado e dispendioso, o que compromete a quantidade de medidas que podem ser obtidas em um determinado corpo hídrico, as quais podem ainda, ser impactada por fontes de incerteza introduzidas à medida que as amostras sofrem degradação durante o transporte para o laboratório (NG; SENFT-GRUPP; HEMOND, 2012). Além disso, conjuntos de dados de baixa densidade obtidos através da amostragem manual podem não capturar adequadamente a variabilidade espaço-temporal, a qual, por vezes, pode conter a chave para a compreensão dos processos biogeoquímicos nos corpos aquáticos (BEUTLER *et al.*, 2002; NG; SENFT-GRUPP; HEMOND, 2012).

A tomada de uma decisão dispendiosa de remediação (isto é, redução da carga de nutrientes ou cessação da poluição) torna-se mais fácil se as consequências da inação puderem ser previstas com base na análise quantitativa. Longas séries de medições quantitativas são, portanto, centrais para as decisões gerenciais (CULLEN *et al.*, 1997).

Barbosa (2003), à época de sua publicação, já defendia que o desenvolvimento de equipamentos e de metodologias que quantificassem a massa orgânica existente, assim como, a sua dinâmica *in vivo* era um requisito básico para a compreensão dos processos de produção natural de energia e equilíbrio ambiental. Sendo as técnicas óticas particularmente adequadas para esta tarefa (CULLEN *et al.*, 1997).

Puiu *et al.* (2015) demonstra que o nível de clorofila *a* na água é uma maneira direta de monitorar o crescimento de algas, configurando-se como um indicador indireto dos níveis de nutrientes.

Conforme expresso em INAG (2009), a determinação *in situ* da concentração da clorofila *a* pode ser efetuada recorrendo a três métodos: espectrofotometria, cromatografia líquida de alta eficiência (HPLC) e fluorometria.

Mckee, Ham e Jones (1997) reforçam que as medições ópticas têm vantagens significativas para o monitoramento ambiental, pois podem ser feitas diretamente na água circundante, sem a necessidade de extração de amostra ou consumo de reagentes, tendo requisitos de energia relativamente baixos e permitindo uma instrumentação mecanicamente robusta, sem partes móveis. Friedrichs *et al.* (2017), também enfatiza a vantagem da medição no campo em função eliminação da preparação da amostra.

Barbosa (2003) acrescenta que a fluorescência é percebida externamente ao organismo fotossintetizante, podendo ser detectada de maneira não invasiva e não destrutiva. Além disso, pode ser detectada à distância, permitindo assim, o uso de técnicas de sensoriamento remoto.

A fluorometria *in situ* é usada frequentemente na oceanografia para fornecer estimativas da biomassa do fitoplâncton. No entanto, o alto preço dos fluorímetros industrializados tornou-os indisponíveis para vários indivíduos e instituições (LEEUW; BOSS; WRIGHT, 2013).

A necessidade de cobrir grandes áreas nas campanhas de medição e o interesse geral na redução dos custos observacionais abrem a necessidade de desenvolver novas estratégias para este objetivo, as quais são fundamentais para lidar com projetos de pesquisa atuais e futuros. Por isso, o desenvolvimento de instrumentos de baixo custo torna-se um fator chave (AYMERICH *et al.*, 2014).

O desenvolvimento de instrumentação de baixo custo desempenha um papel fundamental nos estudos ambientais e representa um dos aspectos mais inovadores da pesquisa atual (MARCELLI *et al.*, 2014).

Os sistemas de monitoramento de longo prazo devem ser econômicos, por isso é importante desenvolver tecnologias de observação que forneçam informações úteis, quantitativas e acessíveis (CULLEN *et al.*, 1997).

A medição da fotofluorescência, mediante a sensibilização por LED com o adequado comprimento de onda, apresenta-se como um método adequado e relativamente barato para a interpretação do volume de clorofila *a* em corpos de água bruta (PUIU *et al.*, 2015).

2.2 Medição por fotofluorescência

Como em qualquer outro processo de troca de energia, a conversão de energia luminosa em energia química no processo fotossintético não é perfeita. Assim, nem todos os elétrons das moléculas de clorofila excitada são passados aos aceptores, retornando por isso, ao estado anterior à recepção da energia dos fótons. Neste momento, essa energia é dissipada em forma de calor ou luz (fluorescência) (CAMPOSTRINI, 2011). Na maioria dos casos, a emissão da fotofluorescência tem comprimento de onda maior que a radiação usada para sua excitação (SKOOG; HOLLER; NIEMAN, 2006, p. 322; 330).

Conforme Barbosa (2003), um grande número de fatores altera o processo da fotossíntese, por exemplo: temperatura e disponibilidade de luz, água ou nutrientes. A fluorescência é um fenômeno que compete com a conversão fotossintética no aproveitamento dos fótons absorvidos. Se o mecanismo de aproveitamento de fótons para a fotossíntese sofre alguma mudança, aumentando ou diminuindo sua eficiência, a intensidade da fluorescência também se altera. A fluorescência natural da clorofila *a*, a qual apresenta um pico em torno de 685nm, é um parâmetro utilizado para se estimar a quantidade de fitoplâncton presente em uma determinada amostra.

Os sistemas para sensoriamento remoto desenvolvidos para estes fins estão, geralmente, embarcados em satélites ou aeronaves, entretanto, outros equipamentos podem ser instalados em conjuntos de boias, navios, veículos submarinos autônomos ou veículos de superfície autônomos (BARBOSA, 2003; MARCELLI *et al.*, 2014). Paralelamente, existem sistemas que são utilizados para compor redes de informação ambiental (ALBALADEJO *et al.*, 2010; BARBOSA, 2003; NG; SENFT-GRUPP; HEMOND, 2012).

A detecção do sinal de fluorescência depende principalmente de três parâmetros: da intensidade da luz da fonte de excitação, do comprimento de onda de absorção relevante, da eficiência quântica do processo e da sensibilidade do detector (FRIEDRICHS e*t al.*, 2017).

Conforme Roesler e Barnard (2013), o uso de fluorômetros simples *in situ* para a obtenção de uma estimativa para a concentração de clorofila é uma das medições biológicas aquáticas mais comuns. Esses dispositivos dependem da excitação da molécula de clorofila usando uma fonte de luz azul e medindo a reemissão de fótons de luz vermelha no processo conhecido como fluorescência.

Uma das características mais atrativas da espectroscopia de fluorescência é sua sensibilidade, se comparada com a espectroscopia de absorção. O limite de detecção das técnicas fluorimétricas é, em geral, de uma a três ordens de magnitude menor do que as de espectrofotometria de absorção (BARBOSA, 2003; SKOOG; HOLLER; NIEMAN, 2006, p. 322).

Seu limite de detecção está, tipicamente, na escala de partes por bilhão. Outra vantagem acessória é a grande faixa de concentração do fluoróforo em que as medidas de intensidade guardam uma relação linear, simplificando assim o procedimento laboratorial de rotina.

Igualmente importante, é o fato de os métodos de espectroscopia por fluorescência serem muito mais seletivos (BARBOSA, 2003).

Equipamentos comerciais de detecção, no âmbito de sondas multi-paramétrica ou sensores de parâmetro único, dominam o mercado de detecção ambiental. O custo desses sensores pode ser proibitivo e, quando os sistemas de registro de medidas são incluídos na conta, os custos de compra podem ser proibitivos (MURPHY *et al.,* 2015).

Para avaliar remotamente a quantidade de clorofila *a* presente em reservatórios naturais de água, duas categorias de medições por fluorescência são empregadas: as técnicas passivas e as ativas. Os métodos passivos procuram medir as alterações da radiação eletromagnética proveniente da água, seja ela absorvida ou espalhada (sem fazer incidir sobre o alvo qualquer fonte artificial de radiação), e correlacioná-la à concentração de clorofila (BARBOSA, 2003). Porém, um elemento fotossintetizante, iluminado continuamente, irá emitir fluorescência de forma continua e a melhor maneira de obter informações a partida da emissão da fluorescência, é por meio da interpretação de seu comportamento cinético em função do tempo (CAMPOSTRINI, 2011).

Os métodos ativos, por sua vez, utilizam uma fonte de excitação luminosa artificial, a qual gera na amostra uma perturbação. Neste sistema, um detector registra a excitação gerada pela fonte artificial superposta à emissão natural. Este sinal de fundo poderá ser subtraído medindo-se com o mesmo detector a emissão em um instante próximo anterior (não submetido à fonte artificial), minimizando os efeitos de uma série dos fatores que alteram a composição da radiação retroespalhada desde a massa de água até o detector (BARBOSA, 2003).

Outra alternativa muito utilizada em fluorímetros comerciais é a utilização de feixe luminoso duplo, sendo o primeiro utilizado como referência, após passar por um atenuador. O segundo feixe, passa por um filtro para a adequação do comprimento de onda, sendo direcionado para promover a sensibilização da amostra, que por sua vez, emite fotofluorescência. Ambos os sinais luminosos são encaminhados a fotomultiplicadores e suas respectivas saídas vão para um amplificador diferencial. Ao final, obtém-se uma leitura resultante, a qual já possui uma compensação para possíveis flutuações na potência da fonte luminosa (SKOOG; HOLLER; NIEMAN, 2006, p. 330).

Esse método de excitação, por meio de iluminação artificial superposta à natural, é conhecido como indução de fluorescência modulada. Entretanto, um método de implementação mais simples, é o de indução de fluorescência não modulada. Nesse segundo método, o elemento fotossintetizante é rapidamente iluminado, após ter sido mantido em ambiente escuro durante um período de adaptação (CAMPOSTRINI, 2011).

Em termos gerais, algumas faixas espectrais são convenientemente escolhidas, de acordo com os espectros de absorção da água, da clorofila *a* e, eventualmente, de algum outro pigmento acessório importante. De acordo com esses espectros, é criado um modelo de irradiação correlacionando as faixas selecionadas e os pigmentos (BARBOSA, 2003).

O fluorímetro para medições *in vivo* não é comparável aos espectrofluorímetros padrões de laboratório, pois o comprimento de onda das excitações não pode ser ajustado por um monocromador, tampouco utilizam lâmpadas de alta pressão para a produção da radiação de excitação. Apesar disso, são bastante empregados, tendo em vista que não há a necessidade de realizar uma varredura completa dos diferentes comprimentos de onda de emissão, uma vez que os intervalos de comprimento de onda que estimulam a fluorescência da clorofila são bem conhecidos (HANELT, 2018).

Conforme (SKOOG; HOLLER; NIEMAN, 2006, p. 331), os fluorímetros comerciais podem utilizar diferentes tipos de emissores de luz. De um modo geral, eles precisam de uma fonte luminosa mais intensa que as utilizadas nas medições por absorção. Entre essas, as lâmpadas de vapor de mercúrio são comuns aos fluorímetros, enquanto os espectrofluorímetros costumam utilizar lâmpadas de xenônio de alta pressão (devido a requererem fonte de radiação contínua). Outra fonte de excitação luminosa possível são os lasers. Em todos esses casos, o consumo energético é um ponto de atenção.

O potencial para fazer medições ópticas *in situ* foi aprimorado pelos avanços nos diodos emissores de luz (LEDs), os quais estão disponíveis a um custo razoável e em uma ampla gama de comprimentos de onda (de 245 nm até a região infravermelha). Isso permite que as medições do espectro de fluorescência sejam feitas em um número substancial de comprimentos de onda de excitação. Nessa função, os LEDs oferecem várias vantagens em relação às fontes alternativas de luz de banda estreita. Em comparação com as lâmpadas de banda larga emparelhadas com monocromadores ou filtros, os LEDs geralmente trazem a vantagem de os sensores poderem usar uma única fonte de excitação e um detector de emissão com um filtro (escolhidos para as bandas de comprimento de onda otimizadas para o parâmetro de interesse). Em comparação com fontes de laser, os LEDs são menos dispendiosos e estão disponíveis em uma ampla gama de comprimentos de onda. Assim, a utilização de LEDs proporciona um menor uso de energia, uma maior eficiência, temperaturas de funcionamento mais baixas, menor tamanho e menor custo (NG; SENFT-GRUPP; HEMOND, 2012).

Utilizando-se esta tecnologia, as peças ópticas são muito mais fáceis de construir, geralmente usando apenas filtros ópticos, para selecionar os picos de emissão de fluorescência, e um arranjo de LEDs com picos de emissão discretos para os comprimentos de onda específicos para a excitação de interesse (HANELT, 2018; NG; SENFT-GRUPP; HEMOND, 2012).

2.3 Estratificação da temperatura em ambientes lagunares

Em função do calor específico da água, as variações naturais de temperatura nos meios aquáticos costumam ser brandas. Consequentemente, a biota aquática não é apta a sobreviver às grandes variações de temperatura. A penetração da luz nos meios aquáticos é essencial para a realização da fotossíntese, porém, na medida em que essa luminosidade é absorvida, é gerada uma transferência de calor, a qual decai de forma aproximadamente exponencial em função da profundidade (BRAGA *et al.*, 2005, p. 75–76).

Esse decaimento é devido à absorção por compostos orgânicos dissolvidos e pelas moléculas suspensas, as quais causam dispersão da luz influindo na distribuição vertical das algas. Como o calor é acumulado principalmente na superfície do lago, essa agua mais quente apresenta menor massa específica e o lago se estratifica termicamente (BERNARDO, 1995, p. 28).

Em um reservatório estratificado (Figura 1), o local de produção do oxigênio é o epilímnio, posicionado junto à superfície, praticamente coincidindo com a zona de luz (zona eufótica). O local de consumo para a decomposição da matéria orgânica é basicamente a região do fundo, a qual é denominada de hipolímnio. O perfil de concentração de oxigênio dissolvido ao longo da vertical possui características bastante semelhantes ao do perfil de temperatura (BRAGA *et al.*, 2005, p. 94; TORRES; GAMA; VILLAS BOAS, 2005, p. 25).

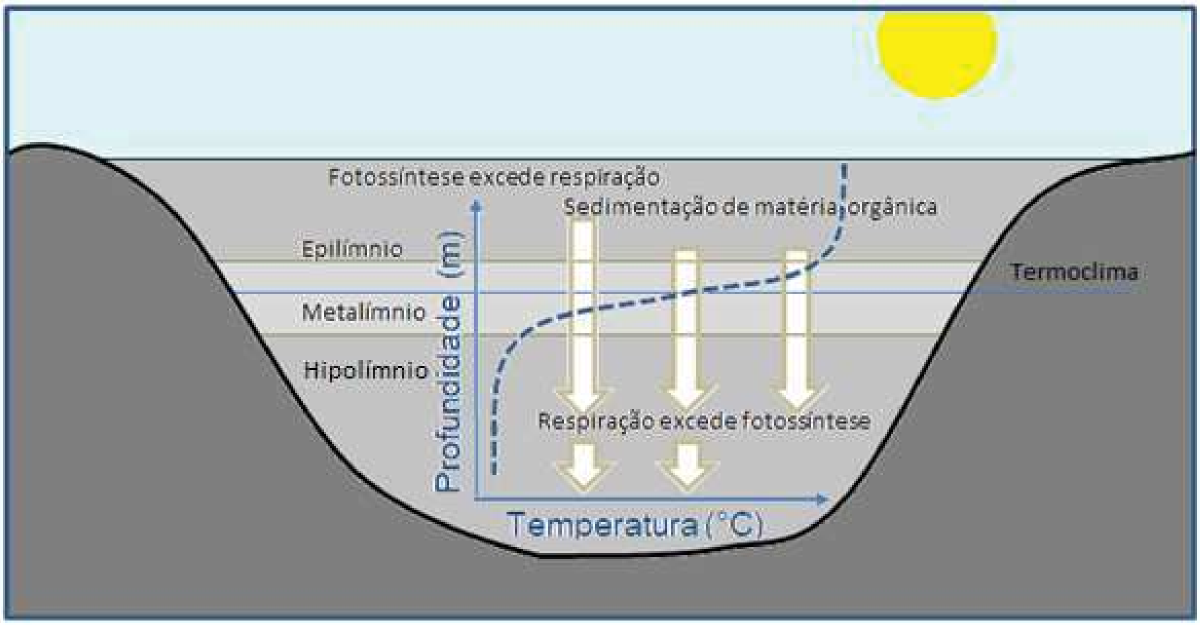


Figura 1 – Estratificação de temperatura em ambiente lagunar

Fonte: Adaptado de Braga *et al*. (2005, p. 75–76).

A região intermediária (entre o epilímnio e o hipolímnio), é denominada metalímnio ou termoclima e apresenta grande influência sobre o fitoplâncton, pois, pelo fato da massa específica da água ser menor, os organismos que estão em processo de sedimentação são freados quando alcançam essa região e aí ficam retidos. Esse fato explica a grande quantidade de algas nessa região (BERNARDO, 1995, p. 29).

> "A estratificação térmica em reservatórios é importante, pois a temperatura afeta todos os processos químico e biológicos que ocorrem no lago. A estabilidade induzida pela estratificação inibe os processos de transporte de calor e massa no reservatório, causando assim, problemas relativos à qualidade da água." (BRAGA *et al.*, 2005, p. 94).

Tanto as reações químicas, quanto a quantidade e o tipo de espécies presentes no meio aquático variam em função da elevação da temperatura, tendo em vista que o metabolismo dos organismos decompositores tende a se acelerar, aumentando a taxa de degradação da matéria orgânica (BRAGA *et al*., 2005, p. 89–90, 93).

As algas, por possuírem massa específica maior que a da água, tenderiam a decantar para o fundo do reservatório. Isso não acontece porque muitas dessas espécies possuem apêndices e adaptações que as ajudam a flutuar fazendo com que a força de atrito entre a água e a superfície das mesmas as mantém em suspensão. A referida força de atrito é função da viscosidade da água, a qual, por sua vez, sofre redução em função do aumento de sua temperatura. Essa realidade leva a uma maior sedimentação desses organismos, afastando-os da zona iluminada, acarretando redução do processo de fotossíntese (BERNARDO, 1995, p. 29; BRAGA *et al.*, 2005, p. 75–76; ODUM *et al*., 2007, p. 494).

Os sistemas lagunares costeiros caracterizam-se em geral por baixas profundidades da coluna d’água, sendo a temperatura, um fator determinante, tanto na ocorrência, quanto na cinética de importantes processos geoquímicos (como eutrofização, floração algal e ciclagem de nutrientes). Tendo em vista que a temperatura varia em função da profundidade da coluna d’agua (Figura 1). O desenvolvimento de sistemas para a determinação da variação das temperaturas em múltiplas profundidades, visando estudar os seus estratos e entender o seu comportamento ao longo da coluna, auxiliará as pesquisas e os estudos de gerenciamento de recursos hídricos.

3 MATERIAL E MÉTODOS

3.1 Pesquisa Bibliográfica

Os primeiros aspectos pesquisados foram os fundamentos teóricos relacionados às alterações de volume fitoplanctônico em corpos hídricos, assim como, à evolução do desenvolvimento dos sistemas de medição correlacionados, analisando as atuais tecnologias empregadas, identificando os prós e contras de cada opção e verificando os aspectos relevantes das implementações.

De forma coerente com o explicado em (PEREIRA; GALVÃO, 2014), antes da coleta dos títulos para a revisão sistemática, foi realizada uma pesquisa preliminar, buscando uma maior familiaridade com o tema, assim como, um melhor embasamento para a avaliação da viabilidade da revisão e para obtenção de uma melhor definição da questão de pesquisa.

As buscas de documentos científicos foram realizadas no repositório ScienceDirect. Os primeiros termos base para a formação da *string* de busca foram: avaliação, clorofila, fitoplâncton e fluorescência, utilizando-se também, seus respectivos thesaurus e radicais de busca. Em seguida, seguindo a mesma estratégia, foram adicionadas as expressões: "*in situ*" e "baixo custo", de forma a promover o refinamento do foco da pesquisa. O estudo bibliográfico culminou na elaboração e publicação de um artigo científico. O Quadro 1 detalha a estrutura das *strings* de busca utilizadas na pesquisa.

Em seguida, foram definidos os critérios de priorização e exclusão de documentos, privilegiando os artigos mais recentes ou com maior número de citações e excluindo aqueles nos quais o Abstract não demonstrou adequada aderência à pergunta de pesquisa. Os documentos selecionados foram analisados na íntegra, sendo realizada uma síntese do conteúdo, quanto aos aspectos relevantes associados ao problema de pesquisa. Esses detalhes serão apresentados na seção de resultados.

Quadro 1 – *Strings* de busca utilizadas nas pesquisas

Tentativa	***String* de busca**
A (Seleção Global)	(((Assessment*) OR (measurement) OR (metering) OR (mensuration) OR (sensor*)) AND ((fluoromet*) OR (fluorimet*)) AND (chlorophyll) AND ((phytoplankt*) OR (eutrophi*)))

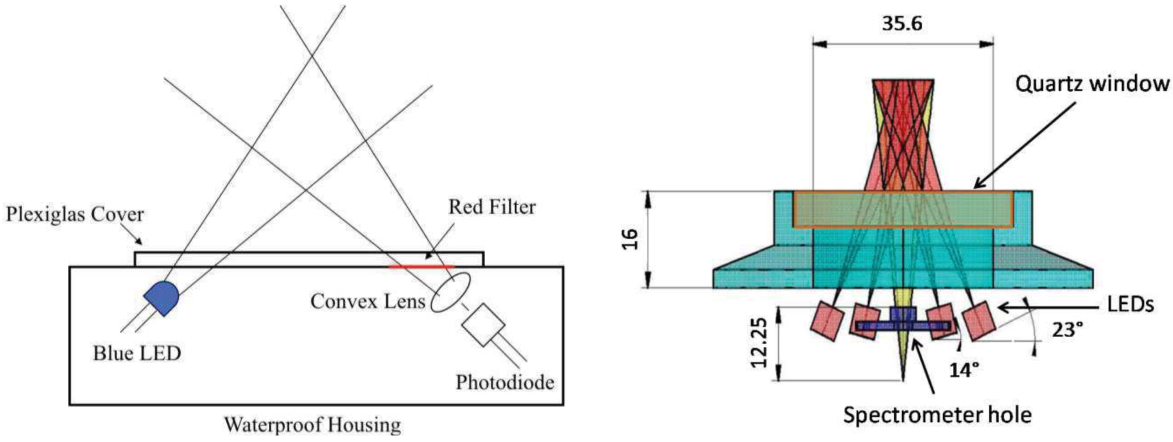
B	(((Assessment*) OR (measurement) OR (metering) OR (mensuration) OR (sensor*)) AND ((fluoromet*) OR (fluorimet*)) AND (chlorophyll) AND ((phytoplankt*) OR (eutrophi*)) AND (in situ))
C	(((Assessment*) OR (measurement) OR (metering) OR (mensuration) OR (sensor*)) AND ((fluoromet*) OR (fluorimet*)) AND (chlorophyll) AND ((phytoplankt*) OR (eutrophi*)) AND ((low cost) OR (low-cost) OR (cheap*) OR (Inexpens*) OR (affordable)))
D (Seleção Específica)	(((Assessment*) OR (measurement) OR (metering) OR (mensuration) OR (sensor*)) AND ((fluoromet*) OR (fluorimet*)) AND (chlorophyll) AND ((phytoplankt*) OR (eutrophi*)) AND ((low cost) OR (low-cost) OR (cheap*) OR (Inexpens*) OR (affordable)) AND (in situ))

Fonte: O Autor

3.2 Procedimentos Técnicos

A pesquisa foi desenvolvida tomando como base os recentes estudos sobre o desenvolvimento de sensores e técnicas para medição de parâmetros ambientais.

A mesma envolveu a construção de um sistema microcontrolado, dotado de sensores de temperatura e a construção de um sensor capaz de medir a fotofluorescência, visando estimar o volume fitoplanctônico de ambientes lagunares (Figura 2). Esse último foi inspirado nos moldes previamente propostos na literatura (LEEUW; BOSS; WRIGHT, 2013; PUIU *et al.*, 2015).



(a) (b)

Figura 2 – Sistemas de medição de fotofluorescência

Fontes: a) Leeuw; Boss; Wright (2013), b) Puiu *et al.* (2015)

O sistema embarcado, em sua primeira versão, consiste do referido sensor, um microcontrolador, um GPS, um sistema de armazenamento local de dados e um conjunto de sensores de temperatura, sendo delimitado conforme as seguintes especificações e premissas:

- Sistema flutuante específico para ambiente lagunar;
- Medição focada em biomassa fitoplanctônica via fotofluorescência;
- Estratificação da temperatura
- Adaptação de tecnologia de custo mais acessível para a construção e a operação;
- Coleta adicional dos valores de posição geográfica;
- Armazenamento local e transmissão de dados, a intervalos definidos;
- Utilização de hardware com arquitetura aberta;

A modelagem do sistema foi desenvolvida utilizando-se BPMN (*Business Process Model and Notation*).

A seleção do microcontrolador e dos sensores de temperatura foi realizada confrontando-se, tanto as experiências relatadas, quanto, as especificações (dos componentes) predominantes na literatura, contra alternativas tecnológicas mais recentes no mercado. Os parâmetros considerados, no caso do microcontrolador, foram: a capacidade de memória, a capacidade de processamento, a aptidão para comunicação, a quantidade de portas de entrada / saída de dados (GPIO), a facilidade de programação, a faixa de tensão de alimentação, a corrente e tensão de fornecimento, a existência de *bootloader* integrado, o tamanho e o custo de aquisição (à época desta publicação). No caso dos sensores de temperatura, foram consideradas as seguintes características: Faixa de medição, resolução, tipo de interligação, protocolo de comunicação, possibilidade de compartilhamento de canal e o custo de aquisição (à época desta publicação). A seção de resultados trará um maior detalhamento da análise realizada.

A construção mecânica do sensor foi desenvolvida nos laboratórios do curso de mecânica do IFF - CAMPUS CAMPOS CENTRO conforme o procedimento descrito no Apêndice A.

A calibração do sistema para a estimativa de volume fitoplanctônico foi realizada tomando como base amostras de extrato de clorofila (retirado de folhas de espinafre). As referidas amostras foram preparadas por maceração por intermédio de almofariz e pistilo, seguida da adequada diluição (em água destilada), obtendo concentrações de 100% (amostra integral), 75%, 50%, 25% e 12,5% da concentração original (vide Apêndice B), associadas a uma amostra de água destilada pura (0% de concentração). A curva de calibração será abordada em detalhes na seção de resultados.

Os resultados obtidos pelo sensor devem manter a correspondência proporcional com as concentrações das amostras utilizadas, assim como, aderência aos resultados obtidos através do uso de um fluorímetro comercial (Aquafluor).

Os testes de campo foram realizados em dois eventos distintos: o primeiro foi na Lagoa Limpa de Travessão, enquanto o segundo se deu em um tanque de criação de peixes, localizado também na localidade de Travessão de Campos (vide Seção 4.10 e Apêndice E). Devido à dificuldade logística de disponibilização de embarcação e condutor devidamente regularizados (fato que dificultou a operação de resgate da boia), a maior parte dos testes foi realizado com o conjunto fundeado, em ambos os eventos (ao contrário da expectativa inicial de realização integral dos testes em modo de deriva). Tendo em vista a continuidade do projeto, mediante a sua aprovação pela AGEVAP (Edital Nº 004/2018) e ao já concedido auxílio financeiro, a próxima etapa envolverá os testes em modo deriva e uma maior aplicabilidade dos dados oriundos no módulo GPS.

Na etapa atual do desenvolvimento, o teste de posicionamento do GPS foi realizado em trajeto rodoviário com amostragem de 5 minutos. Após a finalização do deslocamento, o sistema permaneceu energizado, de forma contínua, até o término da carga da bateria, apresentando-se também, como um teste de autonomia.

4 RESULTADOS E DISCUSSÃO

4.1 Aspectos da seleção de documentos

O diagrama de Venn apresentado na Figura 3 mostra que, considerando-se a base de dados pesquisada, foram encontradas 5631 ocorrências para a combinação de termos: avaliação, fluorescência, clorofila e fitoplâncton. Observa-se na Figura 4 que, após a adição do termo "*in situ*", as ocorrências foram reduzidas a 3582, o que equivale à, aproximadamente, 64% do total. Observa-se também, que após a adição simultânea dos termos "*in situ*" e "baixo custo", as ocorrências foram reduzidas a 578.

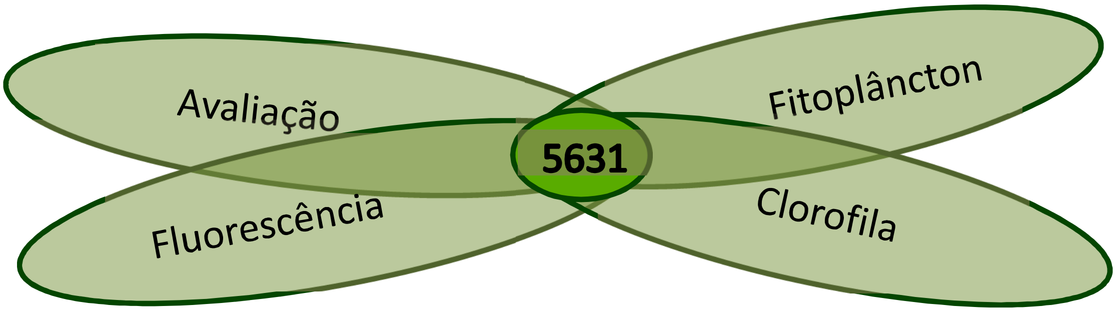


Figura 3 – Resultados da aplicação da *string* final (Seleção Global)

Fonte: O Autor

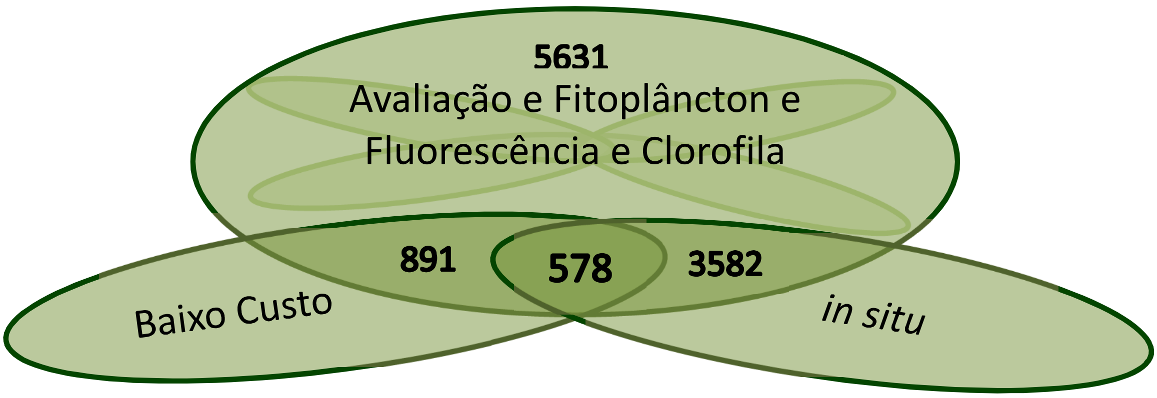


Figura 4 – Resultados da aplicação da string final (Seleção Específica)

Fonte: O Autor

Na Figura 5, considerando-se um corte temporal de 1980 até 2017, pode-se observar a tendência de comportamento das quantidades de publicações relacionadas às *strings* "A" (linha em azul) e "D" (linha em vermelho) mencionadas no Quadro 1. Em ambos os casos, pode-se verificar a continuidade da tendência de crescimento.

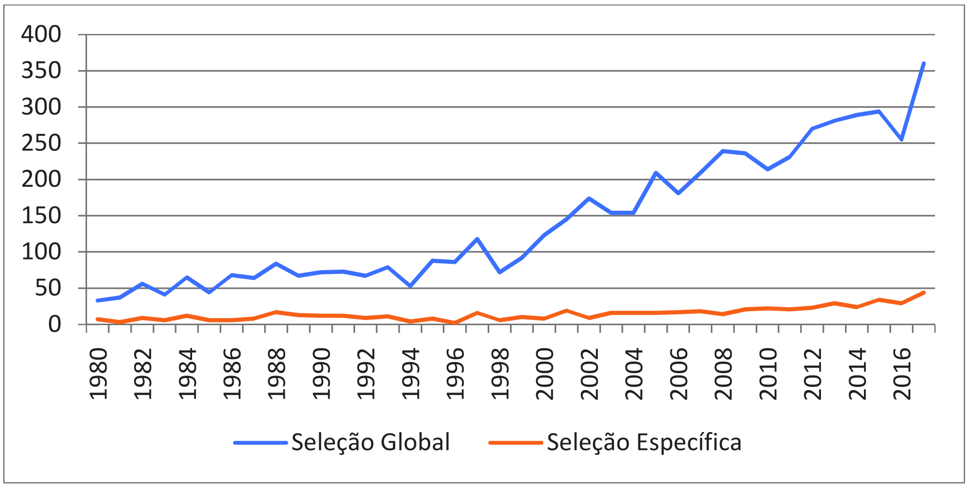


Figura 5 – Gráfico de tendência do número de publicações na base ScienceDirect

Fonte: O Autor

Com base na *string* "D" (seleção específica do Quadro 1 e da Figura 5) puderam ser identificados os periódicos mais usados para divulgação destes conhecimentos (Quadro 2).

Quadro 2 – Vinte veículos de divulgação mais relevantes

Título da Publicação	Nº de Artigos
Deep Sea Research Part B. Oceanographic Literature Review	70
Deep Sea Research Part II: Topical Studies in Oceanography	33
Journal of Marine Systems	31
Water Research	27
Progress in Oceanography	24
Journal of Experimental Marine Biology and Ecology	22
Marine Pollution Bulletin	22
Estuarine, Coastal and Shelf Science	20
Journal of Chromatography A	20
Harmful Algae	17
Remote Sensing of Environment	15
Journal of Great Lakes Research	14
Continental Shelf Research	13
Deep Sea Research Part I: Oceanographic Research Papers	10
Science of The Total Environment	10
Journal of Sea Research	9
Marine Chemistry	9
Aquaculture	8
Deep Sea Research Part A. Oceanographic Research	8
Ecological Engineering	6

Fonte: O Autor

Considerando-se os dados apresentados na Figura 3, observou-se a existência de um número expressivo de trabalhos relacionados à avaliação do fitoplâncton por fotofluorescência da clorofila (Quadro 1. Tentativa A = 5631 trabalhos). Analisando-se a Figura 5, para esse mesmo conjunto de termos (neste caso, identificados como “Seleção Global”), pôde-se perceber que, apesar das oscilações, existe uma tendência de aumento progressivo na quantidade de trabalhos relacionados.

A Figura 4 demonstra que mais da metade dos trabalhos encontrados (Quadro 2, Tentativa B = 3582) se relaciona à medição/estimativa *in situ*. Essa predominância exemplifica às tentativas de contornar as dificuldades relacionadas às medições realizadas em ambiente de laboratório, pois nas mesmas, se fazem necessários o acondicionamento, a preservação e o transporte das amostras (FRIEDRICHS *et al.*, 2017; INAG, I. P., 2011; NG; SENFT-GRUPP; HEMOND, 2012). Esses métodos apresentam considerável morosidade na obtenção do resultado da análise e custos elevados, os quais acabam contribuindo para a limitação do quantitativo de amostras a serem analisadas.

Analisando-se na Figura 5, a tendência identificada como “Seleção Específica” (compatível com a Tentativa D do Quadro 2), pôde-se identificar um crescimento expressivo no número de publicações relacionadas ao tema abordado, com um destaque especial para o período entre 2008 (14 publicações) e 2017 (44 publicações). Considerando-se apenas os dados relativos aos meses de janeiro e fevereiro de 2018, os quais não foram representados na Figura 5, identificaram-se 10 novas publicações, o que equivale a uma média bimestral 36% maior do que a das publicações de 2017.

4.2 Aspectos da revisão sistemática

A partir de uma revisão sistemática, foram encontradas publicações, as quais apresentaram diferentes características, vantagens e custos de construção, relacionados aos seus sistemas e implementações. O Quadro 3 resume os aspectos relevantes das principais publicações encontradas, seguindo uma ordem crescente de datas de publicação:

Quadro 3 – Principais descobertas da revisão sistemática

Autor(es) / Ano	Aspectos relevantes da publicação
(MCKEE; HAM; JONES, 1997)	Descrição de um instrumento submersível que faz medições simultâneas de fluorescência de clorofila, atenuação de feixe e dispersão angular, usando uma lanterna de xenon como fonte de luz e três sensores ópticos (fluorímetro, nefelômetro e transmissômetro), montados em uma carcaça submersível de 2 metros. O sistema dispara trens de pulso de espaçamento predeterminado (geralmente cinco pulsos por segundo, seguido por um período de pausa ajustável).
(BARBOSA, 2003)	Relato do desenvolvimento de métodos e algoritmos para a extração de informações de interesse no monitoramento ambiental, a partir de espectros gerados por um equipamento de *Light Detection And Ranging* (LIDAR), construindo uma ferramenta de suporte a estudos oceanográficos.
(NG; SENFT-GRUPP; HEMOND, 2012)	Apresentação de um detector ótico baseado em fluorescência, absorvência e dispersão, utilizando um microcontrolador Atmel ATmega (programável pela linguagem de programação do Arduino), cinco unidades LED (*Light Emitting Diode*) e uma lâmpada de deutério-tungstênio com comprimento de onda entre 185 e 1100 nm.
(LEEUW; BOSS; WRIGHT, 2013)	Apresentação de uma investigação sobre a construção de um fluorímetro *in situ*, de arquitetura aberta. O mesmo se propõe a ser facilmente reproduzível, a partir de um microcontrolador Arduino Duemilanove (ATmega328) e de componentes eletrônicos simples e acessíveis, permitindo a realização de coletas de medidas precisas da fluorescência *in situ* do fitoplâncton, a um custo de aproximadamente US$ 150.
(MARCELLI *et al.*, 2014)	Apresentação de uma aplicação para o monitoramento integrado e distribuído em ambiente marinho, especificamente desenvolvida para utilização em embarcações. O referido sistema mede temperatura, condutividade, clorofila *a* e clorofila orgânica dissolvida, possuindo um custo de implementação, à época, de aproximadamente 1.200 €.
(MURPHY *et al.*, 2015)	Descrição de um equipamento com capacidade de identificação de tendências de poluição e de uma implementação de sistema de alerta antecipado. O mesmo utiliza 5 lâmpadas de LED e um microcontrolador CC2511F32 (*Texas Instruments*) programado em uma variante da linguagem C, associado a um conversor analógico/digital de 12 bits. Em paralelo é utilizado um controlador Raspberry PI (modelo *b*). O custo aproximado foi de 250 € para os elementos óticos-eletrônico, convergindo para um custo total inferior a 650 €.
(PUIU *et al.*, 2015)	Apresentação do desenvolvimento de um sensor submersível baseado em espectroscopia de fluorescência para monitoramento, em tempo real, de importantes indicadores da qualidade da água (como clorofila *a*, óleo e material similar a proteína), em diferentes profundidades, focando no desenvolvimento de um sensor de baixo custo capaz de medir a fluorescência em um amplo intervalo espectral (200 nm-1100 nm). O sistema utilizou um microcontrolador PIC 18F2550 (microchip), possuindo uma interface gráfica desenvolvida no ambiente LabView.

(BUSCH *et al.*, 2016)	Análise de diferentes equipamentos desenvolvidos com características de baixo custo e baseados no paradigma do "faça você mesmo".
(LOCKRIDGE *et al.*, 2016)	Detalhamento de duas sondas para monitoramento simultâneo de temperatura e salinidade (condutividade) para ambiente marinho. A proposta tem foco no baixo custo, na flexibilidade e na maximização da personalização. O primeiro modelo apresentado consiste em uma sonda de deriva, utilizando a plataforma Arduino MEGA 2560 e um sistema de GPS (*Global Positioning System*) a um custo aproximado de US$ 300, enquanto o segundo modelo é um sistema fixo, utilizando a plataforma Arduino UNO a um custo de aproximadamente US$ 250.
(BARDAJI *et al.*, 2016).	Relato do projeto denominado KdUINO, o qual consiste de uma boia para medição da transparência da água. O sistema se baseia na utilização da plataforma Arduino MEGA, aproveitando as funcionalidades de uma câmera de vídeo para a captura da disponibilidade de luz em diferentes profundidades, em três canais (RGB), e interpretar a atenuação de luz em toda a coluna de água. O custo de construção manteve-se entre US$ 60 e US$ 100.
(MASSERINI JR. *et al.*, 2017)	Descrição do desenvolvimento de um analisador portátil da superfície da água do mar, destinado ao mapeamento da distribuição de nitrato, nitrito e amônia, os quais são considerados os três principais nutrientes inorgânicos nitrogenados nos sistemas costeiros. A abordagem utiliza um GPS e lâmpadas de flash xenon pulsadas. O equipamento pesa 45kg, utiliza três fluorímetros separados. O custo foi de, aproximadamente, US$ 20.000.
(FRIEDRICHS *et al.*, 2017)	Relato do projeto denominado SmartFluo, o qual consiste de um sistema, dotado de um tipo de sensor que aproveita as funcionalidades de um Smartphone para a captura, via câmera fotográfica, da parcela vermelha do sinal RGB, o qual é relativo aos valores de fotofluorescência da concentração de clorofila *a*. O mesmo custou aproximadamente 50 €, excluindo o smartphone. O projeto tem, como um de seus objetivos, incentivar os cidadãos a realizar observações ambientais com seus smartphones.

*Os valores financeiros citados são relativos às datas das respectivas publicações

Fonte: O Autor

Partindo dos dados analisados (Quadro 3), foi constatada uma grande heterogeneidade no que diz respeito à abrangência das diferentes implementações, existindo algumas caracterizadas por uma alta flexibilidade e/ou por uma grande quantidade de parâmetros monitoráveis (MARCELLI *et al.*, 2014; MASSERINI JR. *et al.*, 2017; MCKEE; HAM; JONES, 1997; NG; SENFT-GRUPP; HEMOND, 2012; PUIU *et al.*, 2015), enquanto outras, demonstraram considerável simplicidade, mantendo seu foco em alguns poucos parâmetros específicos, obtendo, por conseguinte, uma expressiva redução de custo (BARDAJI *et al.*, 2016;

LEEUW; BOSS; WRIGHT, 2013; LOCKRIDGE *et al.*, 2016). A existência dessas diferentes categorias de implementação dificultou a comparação entre o custo das mesmas.

No que diz respeito aos aspectos construtivos (apesar de que nem todas as publicações revelaram os detalhes de suas implementações), considerando-se as informações disponibilizadas, foi possível identificar alguns fatores predominantes quanto a determinados aspectos. O Quadro 4 os resume:

Quadro 4 – Aspectos construtivos predominantes na revisão sistemática

Aspecto	Predominância identificada
Quanto à utilização de microcontro-ladores	São relatadas arquiteturas, as quais, apesar de apresentarem alguma variação (entre versão ou modelo), demonstraram a predominância do Arduino/ATmega (BARDAJI *et al.*, 2016; LEEUW; BOSS; WRIGHT, 2013; LOCKRIDGE *et al.*, 2016; NG; SENFT-GRUPP; HEMOND, 2012), sobre outras opções como PIC (PUIU *et al.*, 2015) ou CC2511F32 (MURPHY *et al.*, 2015).
Quanto à fonte de excitação luminosa	Constatou-se a predominância da utilização de LED (LEEUW; BOSS; WRIGHT, 2013; MURPHY *et al.*, 2015, 2015; NG; SENFT-GRUPP; HEMOND, 2012), em detrimento de outras opções, como as lâmpadas de xênon (MASSERINI JR. *et al.*, 2017; MCKEE; HAM; JONES, 1997) ou as lâmpadas de deutério-tungstênio (NG; SENFT-GRUPP; HEMOND, 2012).
Quanto ao elemento sensor	Propostas mais flexíveis (e consequentemente, com maior custo de implementação) aplicaram fluorímetros comerciais (MASSERINI JR. *et al.*, 2017; MCKEE; HAM; JONES, 1997; NG; SENFT-GRUPP; HEMOND, 2012), enquanto outras utilizaram sensores de intensidade luminosa. (LEEUW; BOSS; WRIGHT, 2013; MURPHY *et al.*, 2015). Implementações mais recentes relataram o uso das funcionalidades de câmeras de vídeo para a captura da disponibilidade de luz em três canais (RGB) (BARDAJI *et al.*, 2016; FRIEDRICHS *et al.*, 2017).

Fonte: O Autor

As constatações acima podem ser justificadas, pois o Arduino caracteriza-se como uma solução de baixo custo, possuindo hardware e software de código aberto, o que facilita a instalação e manutenção do sistema. A existência de um *bootloader* integrado apresenta-se como um grande facilitador para sua utilização (ARDUINO, 2018a). Sua popularidade tem sido impulsionada, tanto por sua simplicidade, quanto pelos inúmeros sensores e bibliotecas disponíveis para a ampliação de suas capacidades básicas (ALVAREZ-CAMPANA *et al.*, 2017; ARDUINO, 2018b; WANG; YANG; CHEN, 2017).

A excitação luminosa utilizando LED oferece várias vantagens em relação às fontes alternativas de luz, estando disponível em uma ampla gama de comprimentos de onda, proporcionando um menor consumo de energia, uma maior eficiência, um menor aquecimento e uma concepção construtiva mais compacta, apresentando-se como um método adequado e barato para a interpretação do volume do fitoplâncton em corpos de água bruta (NG; SENFT-GRUPP; HEMOND, 2012; PUIU *et al.*, 2015). Outro aspecto relevante é que conforme (LEEUW; BOSS; WRIGHT, 2013), o alto preço dos fluorímetros industrializados, por vezes, torna-os inacessíveis a várias instituições. Essa realidade tem impulsionado a utilização de sensores de intensidade luminosa, montados em conjunto com filtros específicos para a seleção do desejado comprimento de onda.

4.3 Seleção do microcontrolador

Apesar da citada predominância dos sistemas de prototipagem Arduino (e suas variantes), entre os trabalhos acadêmicos analisados, foi pesquisada outras concepções tecnológicas mais recentes, porém menos consolidadas para atender a função de microcontrolador.

Baseando-se na análise prévia da quantidade de entradas e saídas necessárias para o projeto em voga, identificou-se o microcontrolador ESP8266-12 (NodeMCU) como uma alternativa a ser confrontada com as opções do Arduino.

O ESP8266 integra switches de antenas, RF *balun*, amplificador de potência, amplificador de baixa emissão de ruído, filtros e módulos de gerenciamento de energia, caracterizando-se como uma solução de Wi-Fi SoC (*System-On-Chip*) altamente integrada para atender às contínuas demandas por eficiência energética, design compacto e desempenho confiável na indústria da Internet das coisas (ESPRESSIF, 2018).

Os módulos desta família são identificados através de uma numeração, a qual varia entre ESP01 até ESP12 (NodeMCU) e, mais recentemente, o ESP32. Os módulos ESP01 e ESP10 vêm configuradas de fábrica como ponte serial Wi-Fi de forma a permitir sua interligação a microcontroladores (a exemplo do Arduino e do PIC), através de comunicação serial, fornecendo aos mesmos a capacidade de comunicação Wi-Fi. Entretanto, em função de que os referidos módulos já são um tipo de microcontrolador, torna-se mais vantajoso utilizá-las diretamente. Exceção a essa regra são os casos de pré-existência de uma aplicação muito complexa ou robusta, já implementada em outro microcontrolador, o qual se queira dotar de comunicação wireless.

A maioria dos módulos desta família necessitam de um conversor USB serial externo para que haja troca de informações entre computador e o módulo. Porém, o módulo NodeMCU (Figura 6) caracteriza-se como uma placa de desenvolvimento que já combina o chip ESP8266, uma interface usb-serial e um regulador de tensão de 3,3 VCC (NODEMCU, 2018). Essa realidade, aliada ao custo e à quantidade de porta de entrada/saída de dados, faz deste módulo a alternativa mais atraente da família para a aplicação aqui analisada.

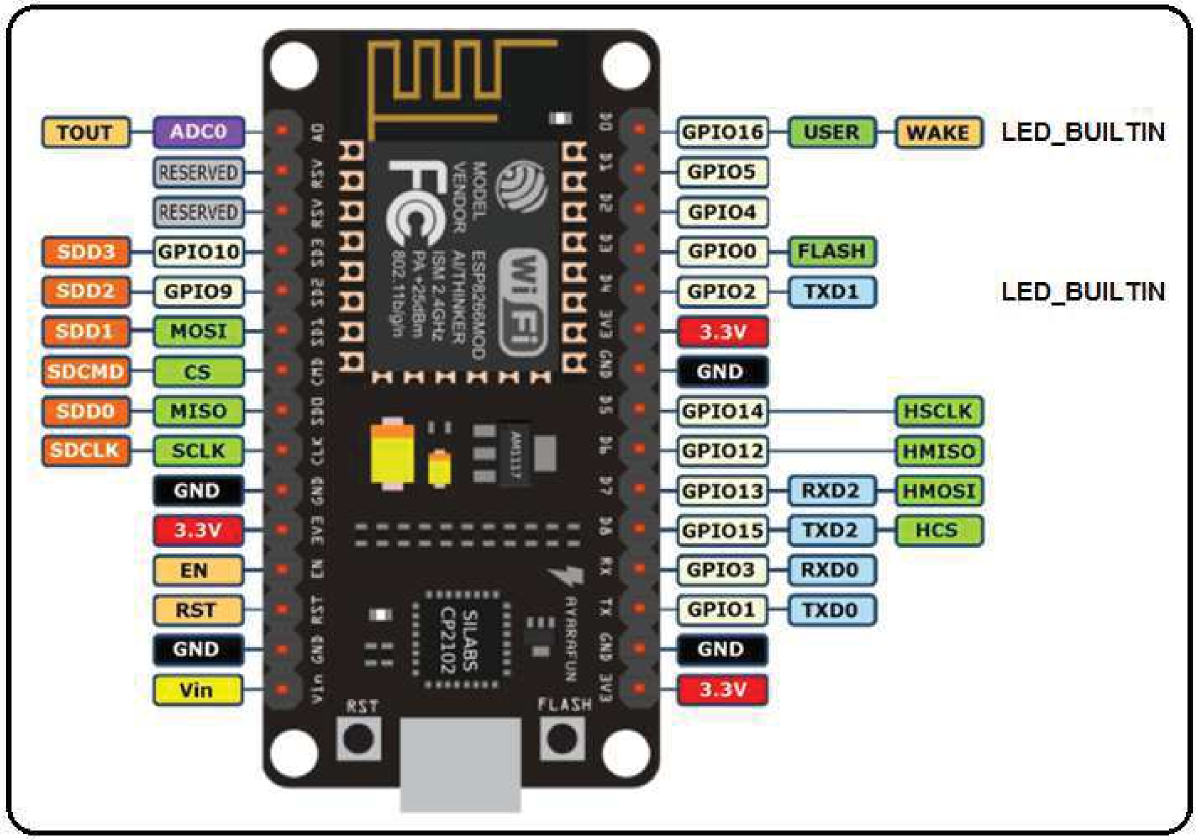


Figura 6 – Placa de desenvolvimento NodeMCU (ESP8266-12)

Fonte: Adaptado de (IOT BYTES, 2016)

A programação dos módulos desta família, geralmente, é feita através da comunicação via cabo micro-usb, utilizando-se uma das seguintes linguagens: Lua (LUA, 2017b), MicroPython (MICROPYTHON, 2017) ou uma variação da linguagem C mediante a utilização da já tradicional IDE do Arduino (ARDUINO, 2018a).

O Quadro 5 compara as especificações técnicas dos modelos Arduino mais utilizados, com as da versão 3 do NodeMCU.

Quadro 5 – Comparativo entre NodeMCU e ARDUINO

	ARDUINO UNO (NANO) ATmega328	NodeMCU v3 Lolin ESP8266-12F
Arquitetura	8 bits	RISC de 32 bits
Velocidade da CPU	16 MHz	80 MHz, possibilidade de 160 MHz
Memória RAM	2 KB	32 KBytes
Memória ROM	1 KB (EEPROM*)	64 KBytes
Protocolos de comunicação	UART, SPI, I2C (TWI*)	Wi-Fi 2.4 GHz integrado + UART, SPI, I2C (TWI), I2S
Porta de entrada / saída (10 bits)	14 discretas e 6 analógicas / (14 discretas e 8 analógicas*)	10 discretas e 1 analógica
Facilidade de programação	C modificado	Lua, MicroPython ou C modificado
Faixa de tensão de alimentação	7 – 12V / 6 – 20V (limites)	4,5 – 9V
Tensão de fornecimento	3,3 V @ 50mA ou 5 V @40mA	3,3 V
Existência de *bootloader* integrado	Sim	Sim
Tamanho (mm)	68,6 x 53,4 / (43,2 x 18,5*)	49 x 25,5
Custo de aquisição (R$)*	25,00 a 45,00 / (23,00 a 45,00*)	30,00 a 50,00

* Características do modelo NANO do Arduino, não válidas para o modelo UNO.

** À época desta publicação

Fonte: O Autor

Comparando as informações do Quadro 5, se pode perceber a superioridade das especificações do chip ESP8266 em relação ao chip ATmega328 (normalmente utilizados nos referidos modelos de Arduino). O ESP8266 opera a uma velocidade, no mínimo, 5 vezes maior, possui 16 vezes mais memória RAM e 64 vezes mais memória ROM. Outras vantagens residem na existência de Wi-Fi 2.4 GHz integrado, na adição do I2S entre os protocolos de comunicação já suportados pelo Arduino, e na possibilidade de utilização de três diferentes linguagens de programação (incluindo o tradicional C, já utilizada pelo Arduino).

A versão NodeMCU pode ser alimentada por tensões entre 5 – 9V, o que, apesar de ser uma faixa menor do que a do Arduino, não configura uma desvantagem significativa. A tensão de fornecimento se restringe à apenas 3,3V, a qual é compatível com a maioria dos periféricos, porém, caso haja necessidade de utilização de algum periférico que demande 5V (o qual poderia ser alimentado diretamente, no caso do uso de alguma versão do Arduino), será necessária a implementação de um circuito de alimentação externo.

Quanto à ocupação de espaço físico, o NodeMCU é vantajoso sobre o Arduino UNO, porém, possui dimensões ligeiramente maiores do que as do Arduino NANO.

Apesar de uma quantidade menor de GPIO, há no mercado diversos circuitos integrados que podem ser utilizados para expansão de entradas e saídas digitais. Cabe ressaltar ainda que, apesar da existência de apenas uma porta de entrada analógica, podem-se utilizar mais sensores interligados paralelamente a mesma porta, mediante uma rotina de multiplexação e utilização de uma porta GPIO para a alimentação sequenciada de cada sensor multiplexado (Vide Apêndice C).

Outras características relevantes do ESP8266 podem ser verificadas no Quadro 6:

Quadro 6 – Características adicionais do ESP8266

Característica	ESP8266
Ethernet	Wi-Fi nativo padrão 802.11b/g/n
Suporte à conexão	Até 5 TCP/IP e comunicação TCP e UDP
Modos de operação	*Access Point, Station e Access Point + Station*
Segurança do Wi-Fi	WPA/WPA2 e criptografia WEP/TKIP/AES
Atualização de firmware	Via interface UART ou OTA
Barramentos de comunicação	UART/SDIO/SPI/I2C/I2S
Temperatura de operação	De -40°C a 125°C

Fonte: O Autor

Assim, apesar de toda a consolidação e aceitação da arquitetura Arduino, tomando como base a análise realizada, optou-se pela utilização da plataforma de desenvolvimento NodeMCU. A principal motivação reside no fato da mesma possuir um tamanho bastante reduzido, uma grande quantidade de funcionalidades, apresentar capacidade de memória e de processamento superiores, possuir um módulo de comunicação Wi-Fi nativo (permitindo inclusive, atualização de firmware via *OVER THE AIR),* aliado a um baixo custo de aquisição, a uma grande facilidade de uso e de integração com sensores/atuadores existentes.

4.4 Seleção da linguagem de programação

Como já mencionado, a programação do NodeMCU (assim como, dos demais módulos da família ESP8266), pode ser realizada através de uma das seguintes linguagens: Lua (LUA, 2017b), MicroPython (MICROPYTHON, 2017) ou uma variação da linguagem C mediante a utilização da já tradicional IDE do Arduino (ARDUINO, 2018a).

Lua é uma linguagem de programação eficiente e leve, que permite programação procedural, programação orientada a objetos, programação funcional, programação orientada a dados e descrição de dados. Ela combina uma sintaxe procedural simples com poderosas construções para descrição de dados, com base em tabelas associativas e semântica extensível, sendo tipada dinamicamente (ou seja, o tipo das variáveis pode ser alterado durante a execução do programa) e executada via interpretação de *bytecodes* para uma máquina virtual baseada em registradores. A mesma possui gerenciamento automático de memória e coleta de lixo incremental, tornando-se ideal para configuração, desenvolvimento de *script* e prototipagem rápida (LUA, 2017a, 2017b).

Lua caracteriza-se como um software livre de código aberto, distribuída sob uma licença MIT (*Massachusetts Institute of Technology*), podendo ser usada para quaisquer propósitos, incluindo propósitos comerciais, sem qualquer custo ou burocracia (LUA, 2017a). Sendo inteiramente projetada, implementada e desenvolvida no Brasil, por uma equipe do Grupo de Tecnologia em Computação Gráfica da Pontifícia Universidade Católica do Rio de Janeiro. Atualmente, é desenvolvida no laboratório LabLua do Departamento de Informática da referida universidade (LUA, 2017b).

Python caracteriza-se como uma linguagem de programação de alto nível, de tipagem forte e dinâmica, interpretada, de script e orientada a objetos. O MicroPython, por sua vez, é uma implementação enxuta e eficiente do Python 3 que inclui um pequeno subconjunto das bibliotecas padrão Python e está otimizado para ser executado em microcontroladores e em ambientes com recursos restritos (RANGEL, 2017).

Apesar das interessantes características das linguagens de programação citadas, optou-se pela utilização da linguagem C, pois além de aproveitar a já bem conhecida IDE do Arduino (ARDUINO, 2018a), a mesma apresenta-se como uma linguagem madura e muito conhecida no ambiente dos programadores. Outro fator determinante foi a existência de uma grande quantidade, tanto de fóruns de discussão sobre esse assunto, quanto, de exemplos, bibliotecas e códigos já publicados e disponíveis para a arquitetura Arduino, os quais podem ser muito facilmente adaptados à nova plataforma de hardware escolhida.

Essa realidade facilitará a reprodução do sistema mantendo aderência aos paradigmas DIY (*Do It Yourself*).

4.5 Seleção dos sensores de temperatura.

O mercado disponibiliza uma ampla variedade de sensores de temperatura, a citar: LM35, DHT11, DHT22, NTC3950, DS18B20, entre outros (Figura 7).

Figura 7 – Sensores de temperatura analisados

Fonte: O Autor

Dentre essas principais opções de sensores, se optou pelo DS18B20, pois o mesmo possui uma faixa de medição ampla, resolução configurável, possibilidade de interligação com compartilhamento de canal e o custo de aquisição bastante acessível.

O DS18B20 é um termômetro digital que fornece medições de temperatura em Celsius, com resolução configurável (de 9 até 12 bits), faixa de aplicação entre -55 ° C a + 125 ° C (com precisão de pelo menos ± 2 ° C), possuindo precisão de ± 0,5 ° C (entre -10 ° C a + 85 ° C). O mesmo pode ser alimentado por tensões entre +3 e +5,5 VCC, possui função de alarme configurável para temperatura alta e baixa e seu tempo máximo de conversão analógica-digital é de 93,75 ms em resolução de 9 bits, porém, podendo atingir 750 ms em resolução de 12 bits (MAXIM INTEGRATED, 2018).

O sensor está disponível em diferentes formas e modelos, existindo 3 variações: SO de 8 pinos (150 mils), µSOP de 8 pinos e TO-92 de 3 pinos. A versão TO-92 possui uma opção de modelo encapsulado em uma caixa impermeável (Figura 8), permitindo instalação submersa (LUIJTEN, 2014; MAXIM INTEGRATED, 2018).

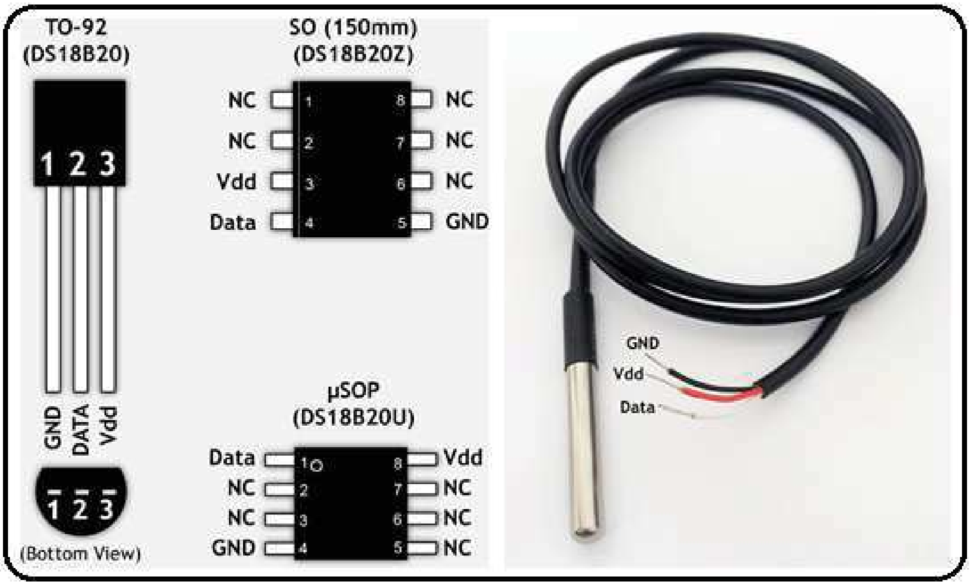


Figura 8 – Modelos de DS18B20

Fonte: Adaptado (LUIJTEN, 2014)

O DS18B20 utiliza o barramento de comunicação OneWire, podendo-se integrar vários sensores em paralelo (Figura 9), utilizando-se de ligação a três fios (*Normal Power Mode*) ou de apenas dois fios (*Parasite Power Mode*), sendo um fio, dedicado à comunicação (LUIJTEN, 2014; MAXIM INTEGRATED, 2018). No modo parasita, o sensor deriva de sua energia a partir da linha de dados sendo necessários apenas os fios de dados e de terra (LEARN OPENENERGYMONITOR, 2019).

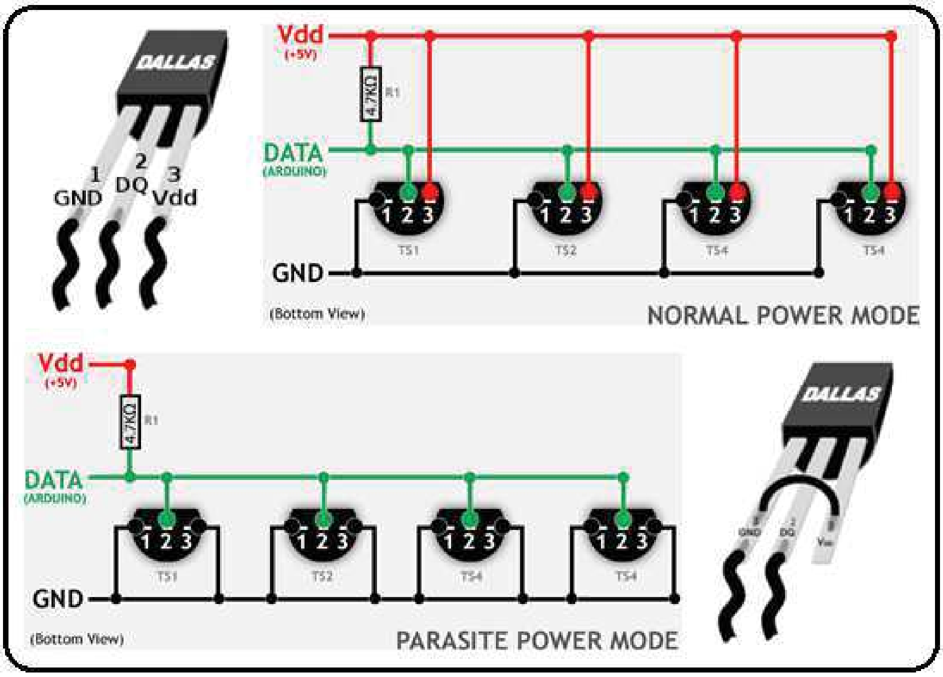


Figura 9 – Múltiplos DS18B20 ligados em paralelo

Fonte: Adaptado de (LEARN OPENENERGYMONITOR, 2019; LUIJTEN, 2014)

Para a utilização do barramento de comunicação OneWire, cada sensor possui um código de série único de 64 bits, o qual promove a identificação do mesmo, permitindo que vários sensores funcionem interligados a um único ponto de entrada digital do microcontrolador em uso (MAXIM INTEGRATED, 2018). O programa para a identificação do código de série do sensor e seu procedimento de utilização esta disponível no Apêndice D.

4.6 Modelagem de software e arquitetura de hardware utilizadas.

Conforme detalhado em Miranda *et. al.* (2017), foi analisada uma nova proposta de metodologia para a modelagem do sistema, porém ao final da análise, considerou-se a tradicional modelagem via BPMN, como mais vantajosa para este tipo de projeto. Mediante a utilização da mesma, foi gerado o modelo de processo apresentado na Figura 10.

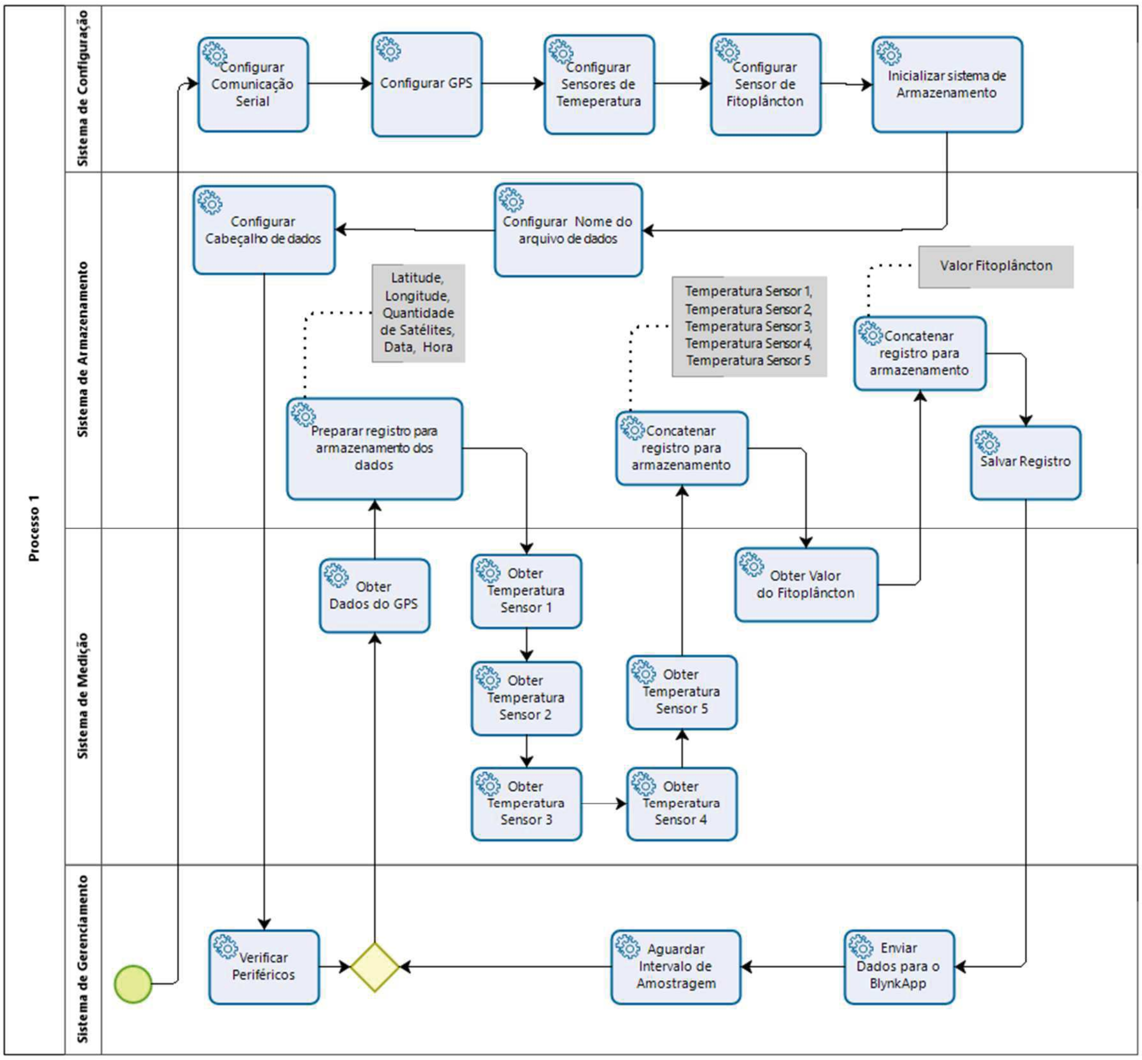


Figura 10 – Modelo de processos do sistema

Fonte: O Autor

A Figura 11 apresenta a representação o diagrama de blocos da arquitetura do hardware:

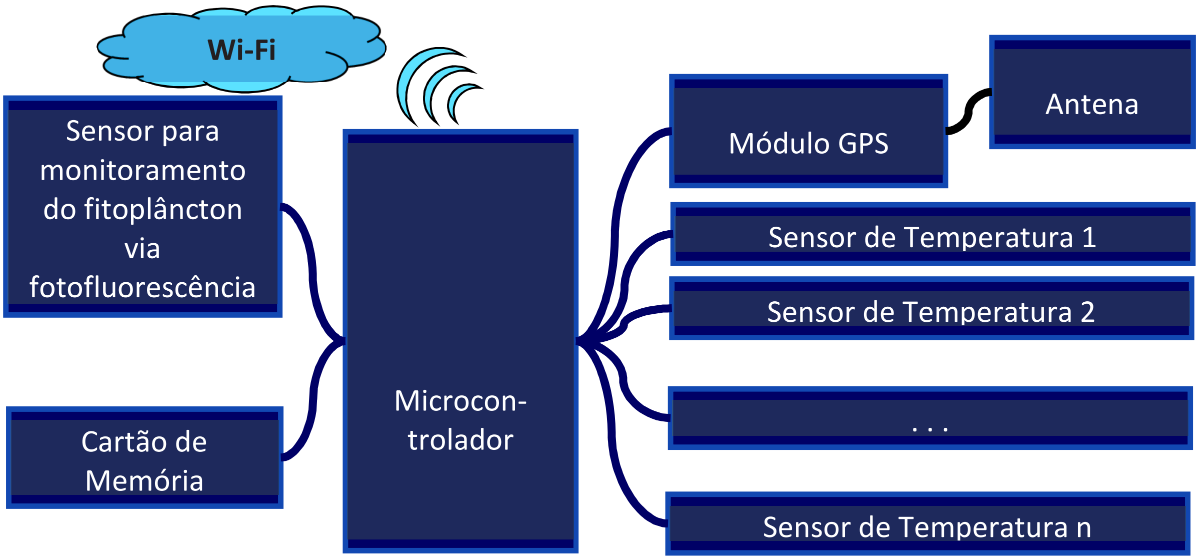


Figura 11 – Diagrama de blocos da arquitetura do hardware

Fonte: O Autor

4.7 Prototipação do hardware

O equipamento foi desenvolvido a partir de uma proposta reconhecida na literatura, a qual utiliza um microcontrolador Arduino Duemilanove e componentes eletrônicos simples (fotodiodo FDS 100, LED azul, Amplificador operacional, resistores de 5 MOhms e 10 Ohms e capacitor 10 pF), além de um filtro de luz (Rosco #19), uma lente convexa, Cartão SD e leitor de cartão SD (LEEUW; BOSS; WRIGHT, 2013).

O trabalho proposto tem como objetivo a integração do sistema de determinação de fitoplâncton embarcado numa boia de deriva a fim de realizar as determinações automaticamente das concentrações em lagoas costeiras e reservatórios de baixa energia.

Este diferencial amplia significativamente a área de cobertura de monitoramento através da utilização de múltiplas estações utilizando a deriva relativa à corrente do reservatório como força de deslocamento. Esta característica atrela os resultados da concentração de fitoplâncton ao horário e à localização determinados por GPS ao longo do dia.

Os resultados calculados do deslocamento permitem a inferência da velocidade e direção da corrente da lagoa e a produtividade fitoplanctônica no momento determinado. Para atender este perfil de produção de dados projetou-se um equipamento integrado cujos componentes são apresentados nos Quadros 7 e 8.

Quadro 7 – Composição e particularidades do hardware geral

Componente	Particularidade(s)
Microcontrolador NodeMCU (EPS 8266-12)	1 (uma) unidade
Sensor de Temperatura BS18B20	5 (cinco) unidades do modelo TO-92; Possibilidade de expansão a mais unidades; Ligação em modo normal (*Normal Power Mode*), vide Figura 9; Ligação OneWire ao pino D4 (GPIO2); Resistor de 4700 Ohms.
Módulo GPS	Ligação serial; TX ligado ao pino D3 (GPIO0, configurado como RX); RX ligado ao pino D2 (GPIO3, configurado como TX).
Modulo SD Card	Ligação aos pinos: • D5 (GPIO14 - HSCLK); • D6 (GPIO12 - HMISO); • D7 (GPIO13 - HMOSI); • D8 (GPIO15 – HCS).

Fonte: O Autor

Quadro 8 – Composição e particularidades do Sensor do fitoplâncton

Componente	Particularidade(s)
LED Azul	1 (uma) unidade Ligação da alimentação ao pino D1 (GPIO5)
LDR	1 (uma) unidade Alimentação em 3,3V Ligação da saída ao pino A0
Filtro de luz (gelatina)	Especificação: Rosco #19 Montado a frente do sensor de intensidade luminosa (LDR)
Resistor	1 (uma) unidade 10.000 Ohms

Fonte: O Autor

A Figura 12 apresenta o esquemático do circuito eletrônico, onde os 5 (cinco) DS18B20 (modelo TO-92) aparecem ligados (em modo normal) ao pino D4 (GPIO2) do NodeMCU e associados a um resistor de 4700 Ohms (entre a alimentação e o sinal de saída). O Módulo GPS utilizou ligação serial com seu pino TX ligado ao pino D3 (GPIO0, configurado como RX no NodeMCU) e RX ligado ao pino D2 (GPIO3, configurado como TX). O módulo do cartão SD está interligado aos seguintes pinos: D5 (GPIO14 - HSCLK), D6 (GPIO12 - HMISO), D7 (GPIO13 - HMOSI) e D8 (GPIO15 – HCS). O LED que compõe o sensor de fotofluorescência está alimentado pelo pino D1 (GPIO5), tendo seu outro borne ligado ao GND. Já O sensor de intensidade luminosa (LDR) é alimentado por um pino de 3,3V, tendo sua saída ligada à entrada analógica (A0) em modo divisor de tensão (com um resistor de 10k Ohms).

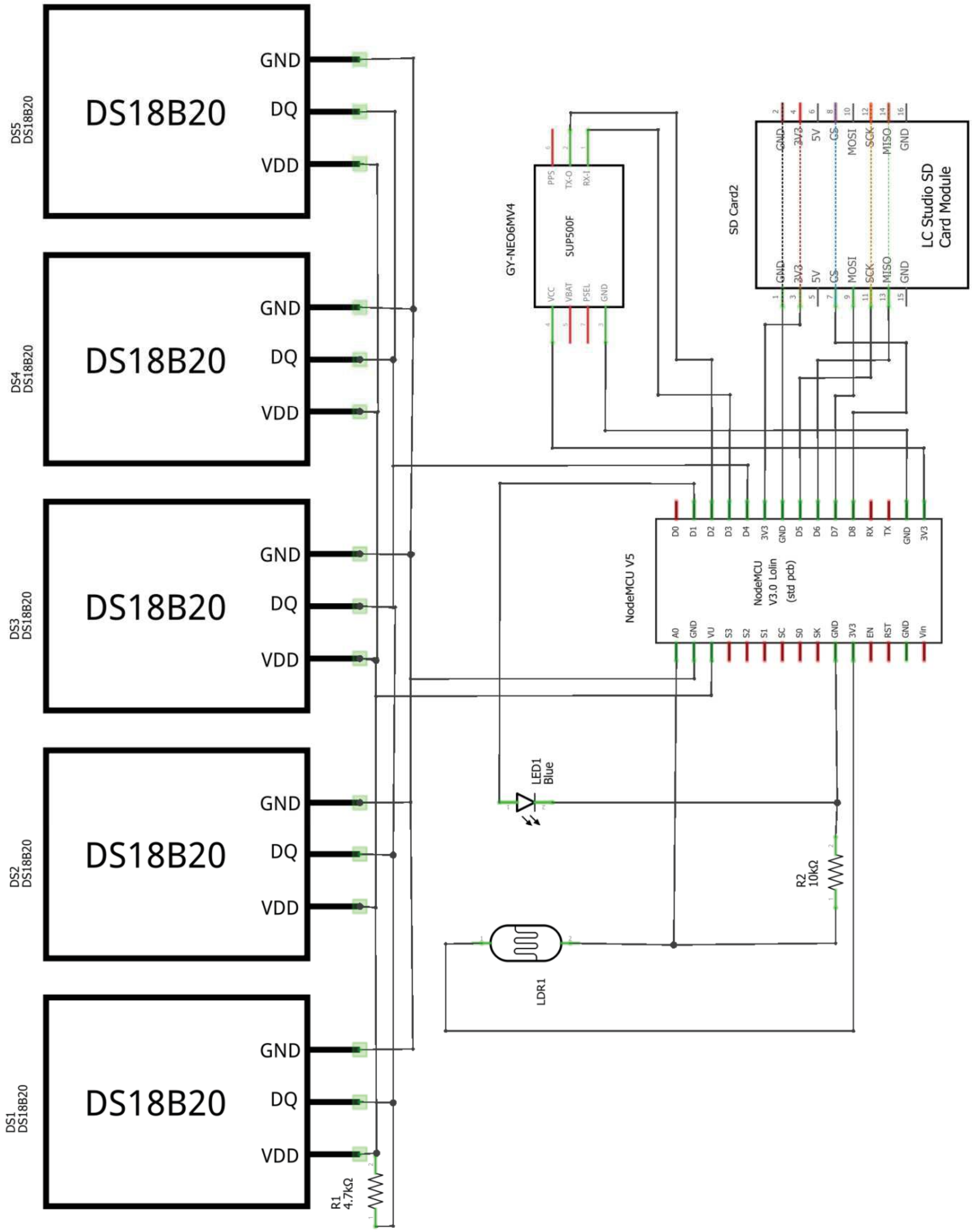


Figura 12 – Esquemático do protótipo

Fonte: O Autor

4.8 Programação do microcontrolador

O programa foi desenvolvido seguindo uma premissa de modularização do código, objetivando favorecer o entendimento e a retirada (ou a inclusão) de componentes. Os comentários foram registrados no idioma inglês e buscaram o maior detalhamento possível. As partes do código serão explicadas a seguir e sua versão completa está disponível no Apêndice F.

Para a utilização dos módulos comerciais (Cartão SD, GPS e Sensores DS18B20) e do aplicativo Blynk foram utilizadas as seguintes bibliotecas:

- SD Library
- SPI Library
- OneWire Library
- DallasTemperature Library
- TinyGPS++ Library
- SoftwareSerial Library
- ESP8266WiFi
- BlynkSimpleEsp8266

O programa desenvolvido se inicia com um mapeamento de portas visando à compatibilidade como a plataforma Arduino (Código 1)

Código 1 – Mapeamento de portas de I/O

```
//Mapping the digital ports for compliance to Arduino Architecture
#define D0   16
#define D1   5        //will be used for blue LED command
#define D2   4        //will be used like RX pin for GPS module communication
#define D3   0        //will be used like TX pin for GPS module communication
#define D4   2        //will be used like bus for OneWire communication with DS18B20 sensors
#define D5   14       //will be used like SCK pin for the SPI communication with SD Card module
#define D6   12       //will be used like MISO pin for the SPI communication with SD Card module
#define D7   13       //will be used like MOSI pin for the SPI communication with SD Card module
#define D8   15       //will be used like chipselect pin for SD Card module
#define D9   3
#define D10  1
```

Seguindo a premissa de modularização do código, a segunda parte do programa apresenta um conjunto de blocos, os quais são responsáveis pela definição das variáveis globais e das bibliotecas de cada periférico ou módulo de programação.

O Código 2 apresenta as configurações do módulo de cartão SD, iniciando pela definição de uma variável global, destinada a manipulação dos dados (os quais serão posteriormente gravados), seguida da inclusão das bibliotecas para utilização do referido módulo e da definição do pino para sua habilitação (D8).

Código 2 – Definições para o módulo de cartão SD

```
String logString;           // String for data handling

// Definitions for SDCard
#include <SPI.h>
#include <SD.h>              // SD.h for ESP8266
#define SD_CHIPSELECT D8
```

O módulo GPS comunica-se com o microcontrolador, através de comunicação serial. Nessa aplicação, optou-se por não utilizar os pinos TX/RX previamente configurados pelo fabricante, deixando-os disponíveis para a comunicação serial com o microcomputador responsável pela programação, permitindo assim, a utilização constante da interface USB (neste programa, configurada em 115200 bps) e do monitor serial da IDE do Arduino. A comunicação serial com o modulo GPS se efetivou através da utilização da biblioteca SoftwareSerial.h e da configuração de dois outros pinos digitais do microcontrolador como portas TX e RX.

O fragmento de código apresentado no Código 3 efetua as definições para o módulo GPS, iniciando pela inclusão das bibliotecas necessária ao seu funcionamento, seguida da definição das portas para comunicação serial no microcontrolador. No caso desse programa, foram configuradas as portas D2, como RX (a qual é interligada ao pino TX do Modulo GPS) e D3, como TX, que por sua vez, é interligada ao RX do referido módulo. Em seguida, é criada uma instância do objeto TinyGPSPlus (para acesso aos dados do GPS) e outra do objeto SoftwareSerial (para a comunicação). Por fim, é criada a variável global logString2, a qual será utilizada para preparação e envio de dados ao servidor Blynk.

Código 3 – Definições para o módulo GPS

```
// Definitions for GPS Module
#include <TinyGPS++.h>
#include <SoftwareSerial.h>
const int RXPin = D2, TXPin = D3;
TinyGPSPlus tinyGPS;                  // tinyGPSPlus object to be used throughout
SoftwareSerial GPS(RXPin, TXPin);     // Create a SoftwareSerial
String logString2;                    // String for GPS data to Blynk App
```

O segmento de código apresentado no Código 4 trata dos sensores DS18B20. Primeiramente, efetua a inclusão das bibliotecas necessárias ao funcionamento. Em seguida, é definindo o pino D4 como porta de comunicação, criando uma instância de canal de comunicação oneWire para os dispositivos, é efetuada a passagem da referência para a biblioteca DallasTemperature.h, em seguida é definindo o nome das instâncias dos sensores, associando-os aos seus respectivos endereços para o barramento oneWire. Esses endereços foram obtidos pela utilização do programa apresentado no Apêndice D e são específicos para os sensores utilizados nessa montagem. Por fim, são criadas as variáveis globais para as temperaturas e para a montagem e envio dos dados ao servidor Blynk.

Código 4 – Definições para os sensores DS18B20

```
// Definitions for DS18B20
#include <OneWire.h>
#include <DallasTemperature.h>
#define ONE_WIRE_BUS D4
OneWire oneWire(ONE_WIRE_BUS);      // Setup a oneWire instance to communicate with any
                                    // OneWire devices
DallasTemperature sensors(&oneWire);// Pass our oneWire reference to Dallas Temperature
//***  Here you must insert the addresses of all DS18B20 ***
DeviceAddress Thermometer01 = { 0x28, 0xFF, 0x9D, 0x27, 0x62, 0x16, 0x03, 0xAB }; // for 1st DS18B20
DeviceAddress Thermometer02 = { 0x28, 0xA8, 0x5B, 0xB5, 0x04, 0x00, 0x00, 0x88 }; //for 2nd DS18B20
DeviceAddress Thermometer03 = { 0x28, 0xFF, 0xA1, 0xC7, 0x60, 0x17, 0x05, 0x11 }; // for 3rd DS18B20
DeviceAddress Thermometer04 = { 0x28, 0xFF, 0x7D, 0x24, 0x30, 0x17, 0x04, 0x8A }; // for 4th DS18B20
DeviceAddress Thermometer05 = { 0x28, 0xFF, 0xF9, 0x34, 0x60, 0x17, 0x05, 0x43 }; // for 5th DS18B20
String Temperature0, Temperature1, Temperature2, Temperature3, Temperature4;
String logString3;                                  // String for Temperature data to Blynk App
```

Para melhor organização do programa, foram criadas *procedures* específicas para cada uma das ações. Essas *procedures* são:

- updateFileName( )
- printHeader( )
- beginSDcard( )
- getGPS_Data( )
- setDS18B20( )
- getDS18B20( )
- setPhyto( )
- getPhyto( )
- logData( )
- callBlynk()

Além das *procedures* padrão: setup( ) e loop( ).

A *procedure* updateFileName( ) é responsável pela definição de um novo nome para o arquivo de armazenamento no cartão de memória. A utilização dessa *procedure* garante que, a cada reinicialização do sistema, um novo arquivo seja criado, evitando que os arquivos anteriores sejam perdidos devido à sobrescrissão. O código 5 define os parâmetros para a *procedure*, a citar: LOG_FILE_PREFIX, composto por 6 (seis) caracteres, o qual irá compor o nome do arquivo a ser gravado; MAX_LOG_FILES, o qual variará de 1 a 99, permitindo a gravação de 99 arquivos consecutivamente diferentes; e LOG_FILE_SUFFIX, o qual especifica o tipo de arquivo a ser gravado (neste caso, o tipo é CSV). Por fim cria uma variável denominada logFileName, a qual armazenará o nome do arquivo a ser gravado.

Código 5 – Definições para *procedure* updateFileName( )

```
//Definitions for Procedure updateFileName()
#define LOG_FILE_PREFIX "SAEGvD"  // Name of the log file.
#define MAX_LOG_FILES 100          // Number of log files that can be made
#define LOG_FILE_SUFFIX "csv"      // Suffix of the log file
char logFileName[13];              // Char string to store the log file name
```

O Código 6 apresenta as definições relativas à *procedure* printHeader( ) e define o número de colunas do cabeçalho do arquivo CSV (COLUMN_COUNT) e os nomes de cada uma das colunas. Neste caso, são 13 colunas e seus respectivos nomes são: *"longitude", "latitude", " velocity", "satellites", "time", "date", "temp_0 (Celsius)”, "temp_1 (Celsius)”, "temp_2 (Celsius)”, "temp_3 (Celsius)”, "temp_4 (Celsius)”, "LDR" e "Volts".*

Código 6 – Definições para *procedure* printHeader( )

```
// Definitions for Procedure printHeader()
#define COLUMN_COUNT 13       // Setup the number of columns for the CSV file
char * log_col_names[COLUMN_COUNT] = {
"longitude", "latitude", " velocity ", "satellites", "date", "time", "temp_0 (Celsius)",
"temp_1 (Celsius)","temp_2 (Celsius)","temp_3 (Celsius)","temp_4 (Celsius)","LDR", "Volts"
};  // Setup the column title. Then, log_col_names will be printed at the top of the file.
```

O fragmento de código apresentado no Código 7 apresenta as definições relativas à *procedure* getPhyto( ) e define os pinos do microcontrolador, os quais serão responsáveis pelo acionamento de emissor de luz (D1) e pela leitura analógica do sensor de intensidade analógica (A0). A seguir, é definida a quantidade de leituras utilizadas para a extração de um valor médio a ser computado, assim como, o intervalo de tempo entre as medições. São criadas as variáveis para armazenamento do valor de tensão e seu equivalente digitalizado (em 10 bits).

Código 7 – Definições para *procedure* getPhyto( )

```
//Definitions for Procedure getPhyto()
#define BLUE_LED D1
#define LDR_PIN A0
const int SAMPLE_NUMBER = 20;   //for calculate the average
const int INTERVAL = 100;       //interval between sampling (miliseconds)
const int DIGITAL_MIN = 0;      //lowest value used in sensor calibration
const int DIGITAL_MAX = 1023;   //highest value used in sensor calibration
double voltage;
int read_LDR;
```

As próximas definições são para a *procedures* logData( ) e loop ( ). É definido o valor de ajuste de fuso horário para variável hora do GPS, neste caso específico, menos 3 (três) horas. Em seguida é configurado a o valor da frequência de amostragem em 5 segundo e definida uma variável que armazena uma variável de tempo relacionada à última execução do programa (Código 8).

Código 8 – Definições para as *procedures* logData( ) e loop( )

```
//Definitions for Procedure logData()
const int OFF_SET = -3;           // Factor for hour adjust from Greenwich time

//Definitions for Procedure loop()
#define LOG_RATE 300000          // Log every 5 minutes //***
unsigned long lastLog = 0;       // Global var to keep of last time we logged
```

Concluindo o bloco de definições, o Código 9 apresenta as definições necessárias para a utilização do aplicativo Blynk, o qual disponibiliza as informações relevantes em uma interface gráfica específica para os sistemas Androide e IOS. Foi definida a variável BLYNK_PRINT, e incluídas as bibliotecas ESP8266WiFi.h (que permite a comunicação WiFi do controlador) e BlynkSimpleEsp8266.h (que permite que o Aplicativo Blynk se comunique com o controlador específico deste projeto). Em seguida são configurados: o token de autorização, o qual é fornecido pelo aplicativo Blynk durante a criação de cada nova interface gráfica; a rede WiFi na qual o dispositivo será conectado e a respectiva senha de acesso à citada rede. Por fim, é criado um *timer* o qual controlará a frequência de transferência de dado entre o microcontrolador e o servidor do Blynk.

Código 9 – Definições para a utilização do Blynk App

```
//Definitions for Blynk App use
#define BLYNK_PRINT Serial
```

```
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
//Bellow insert your authorization token, which was gave
//on project creation in Blynk App.
char auth[] = "<CodigoDeAutorizaçãoGeradoNoSeuCadastroBlynk";        // Insert your token
char ssid[] = "<SuaRede>";                                           // Insert your network name
char pass[] = "<SuaSenha>";                                          // Insert your password
BlynkTimer timer;                                                    // For Blynk App refresh
```

Dando sequência ao programa, seguem os códigos para cada uma das *procedures*.

A primeira *procedure* codificada é updateFileName( ), a qual cria um novo nome para o arquivo a ser gravado no cartão. No caso do programa, o nome é SAEGvDxx.csv, onde os dois últimos caracteres anteriores ao ".csv" são um número sequencial que variará de 1 até 99. A cada inicialização do sistema é executado este procedimento, o qual realiza um teste de existência de um arquivo com mesmo nome, informando no monitor serial o resultado. Ao final, define o nome criado e informa no monitor serial (Código 10).

Código 10 – *Procedure* updateFileName( )

```
void updateFileName(){ // It will be used in beginSDcard() procedure
  //Create a new name for the log file for each run of the setup() procedure
  Serial.println("trying updateFileName.");
  int i = 0;
  for (; i < MAX_LOG_FILES; i++){
    memset(logFileName, 0, strlen(logFileName)); // Clear logFileName string
    // Set logFileName to "SAEGvDXX.csv":
    sprintf(logFileName, "%s%d.%s", LOG_FILE_PREFIX, i, LOG_FILE_SUFFIX);
    if (!SD.exists(logFileName)){ // If a file doesn't exist
      break;                  // Break out of this loop. We found our index
    }
    else{
      Serial.print(logFileName);
      Serial.println(" exists");  // Print a debug statement
    }
  }
  Serial.print("Current file name: ");
  Serial.println(logFileName);    // Print the file name for debug
}
```

A segunda *procedure* é a printHeader( ). A mesma promove o acesso ao arquivo previamente criado, escrevendo no mesmo, uma primeira linha, na qual consta o nome de cada uma das colunas (previamente definidos no vetor log_col_names). Essas colunas armazenarão as variáveis coletadas e permitirão uma fácil manipulação em aplicativos de planilha eletrônica (Código11).

Código 11 – *Procedure* printHeader( )

```
void printHeader(){  //prints each column names to the top of our log file
  //prints each column names to the top of our log file
  Serial.println("\nTrying printHeader.");
  File logFile = SD.open(logFileName, FILE_WRITE); // Open the log file
  if (logFile){     // If the log file opened, print our column names to the file
    int i = 0;
    for (; i < COLUMN_COUNT; i++){
      logFile.print(log_col_names[i]);
      if (i < COLUMN_COUNT - 1){  // If it's anything but the last column
        logFile.print(',');       // print a comma
      }
      else {                      // If it's the last column
        logFile.println();        // print a new line
      }
    }
    logFile.close();              // close the file
  }
}
```

O Código 12 apresenta a *procedure* beginSDcard( ), a qual será invocada durante o *setup*. A mesma promove um teste de inicialização do *SDCard,* seguido das chamadas às *procedures* updateFileName( ) e printHeader( ), previamente explicadas.

Código 12 – *Procedure* beginSDcard( )

```
void beginSDcard(){
  Serial.println("\n Setting up SD card.");
  // see if the card is present and can be initialized:
  if (!SD.begin(SD_CHIPSELECT)){
    Serial.println("Error initializing SD card.");
  }
  Serial.println(" SD card initialized.");
  updateFileName();    // Each time we start, create a new file, increment the number
  printHeader();       // Print a header at the top of the new file
}
```

As variáveis do GPS são as primeiras a serem armazenadas na *string* de manipulação (logString), a qual será gravada no arquivo do cartão SD. Todo armazenamento é realizado, mediante a conversão do tipo de dado original, de cada uma das diferentes variáveis lidas, para o tipo String. Em seguida, este texto é concatenando ao previamente existente, e, só ao final do agrupamento de todos os dados, a *string* de manipulação é, efetivamente, gravada no arquivo.

O Código 13 apresente a *procedure* getGPS_Data( ), a qual será responsável pela coleta dos dados do GPS. Apesar dos módulos GPS poderem fornecer um grande número de diferentes dados (como por exemplo: altitude, sentido de curso, etc.), nesta aplicação, optou-se por não utilizar todos os dados disponíveis. A programação limitou-se aos dados efetivamente pertinentes à aplicação.

Código 13 – *Procedure* getGPS_Data( )

```
162. void getGPS_Data(){
163.   int rightHour, rightDay;
164.   logString2 =  String(tinyGPS.location.lng(), 6) + ", " + String(tinyGPS.location.lat(), 6) + ", ";
165.   logString2 += String(tinyGPS.speed.kmph(), 1)+ ", " + String(tinyGPS.satellites.value()) + ", ";
166.   if (int(tinyGPS.time.hour()) <= (OFF_SET * (-1))) {
167.     rightHour = (int(tinyGPS.time.hour()) + 24 + OFF_SET);
168.     rightDay = (int(tinyGPS.date.day()) - 1);
169.   }
170.   else {
171.     rightHour = (int(tinyGPS.time.hour()) + OFF_SET) ;
172.     rightDay = int(tinyGPS.date.day());
173.   }
174.   logString2 += String(rightDay) + "/" + String(tinyGPS.date.month()) + "/" ;
175.   logString2 += String(tinyGPS.date.year()) + ", " +  String(rightHour) + ":";
176.   logString2 += String(tinyGPS.time.minute()) + ":" + String(tinyGPS.time.second())+ ", ";
177.   logString = logString2;
178. }
```

Esta *procedure* utiliza duas variáveis globais do tipo *string* (logString e logString2). A primeira armazenará, consecutivamente, todas as informações que deverão ser armazenadas no cartão de memória, e por isso, será utilizada em várias diferentes *procedures* (sempre recebendo dados adicionais, concatenados aos já existentes e separados por uma vírgula). A variável logString2 será utilizada para o envio dos mesmos dados ao servidor Blynk. Este envio será realizado pela *procedure* BlynkRun( ) no final da execução da *procedure* loop( ).

O código 13 utiliza funções nativas da biblioteca TinyGPS++.h, as quais retornam às informações disponibilizadas pelo módulo GPS. São criadas as variáveis locais rightHour, rightDay, as quais são utilizadas para armazenar, respectivamente, a hora e o dia corrigidos em função do fuso horário. Em seguida são lidos e armazenados os valores de longitude e latitude (ambos com seis casas decimais). Optou-se por armazenar também a informação de velocidade e o número de satélites visíveis ao módulo GPS (pois, um número muito reduzido pode deteriorar consideravelmente o resultado do processo de localização).

O valor a ser armazenado para data e hora da medição deve ser corrigido em função do fuso horário (vide linhas de 166 a 173). Atenção especial deve ser dada ao fato de que o código de correção criado é aplicável apenas à parte ocidental do meridiano de Greenwich. Para a aplicação à parte oriental, ajustes devem ser feitos ao referido código. Nessa aplicação foi desconsiderada a existência de horário de verão.

A *string* de manipulação possui campos únicos para armazenar, respectivamente, a data e a hora da medição. Por isso, o primeiro dado deve ser produzido através da junção do valor da leitura do dia (já corrigido em função do fuso horário), associada às leituras dos valores de mês e ano. Enquanto a informação de hora é representada pela junção do valor da leitura da hora (também corrigida em função do fuso horário), associada às leituras dos valores de minuto e segundo (desconsiderando sua parcela decimal).

Para a utilização dos sensores de temperatura escolhidos (DS18B20), foram criadas duas *procedures* (Código 14). A *procedure* setDS18B20( ) é invocada durante o *setup* e inicializa a instância do objeto DallasTemperature (denominada sensors) previamente definida (linha 56 do Código 4). Em seguida, configura a resolução de cada um dos sensores cujos endereços foram definidos no Código 4 (no caso, cinco unidades).

Código 14 – *Procedures* setDS18B20( ) e getDS18B20( )

```
void setDS18B20(){
  sensors.begin();                                  // Start up the DallasTemperature library
  sensors.setResolution(Thermometer01, 10);  // set the DS18b20 resolution to 10 bit for each one
  sensors.setResolution(Thermometer02, 10);
  sensors.setResolution(Thermometer03, 10);
  sensors.setResolution(Thermometer04, 10);
  sensors.setResolution(Thermometer05, 10);
}

void getDS18B20(){
  sensors.requestTemperatures();
  Temperature0 = String(sensors.getTempC(Thermometer01), 2);
  logString += Temperature0 + ", ";
  logString3 = "Temp0 = " + Temperature0 + " Celcius, \n";

  Temperature1 = String(sensors.getTempC(Thermometer02), 2);
  logString += Temperature1  + ", ";
  logString3 += "Temp1 = " + Temperature1 + " Celcius, \n";

  Temperature2 = String(sensors.getTempC(Thermometer03), 2);
  logString += Temperature2  + ", ";
  logString3 += "Temp2 = " + Temperature2 + " Celcius, \n";

  Temperature3 = String(sensors.getTempC(Thermometer04), 2);
  logString += Temperature3  + ", ";
  logString3 += "Temp3 = " + Temperature3 + " Celcius, \n";

  Temperature4 = String(sensors.getTempC(Thermometer05), 2);
  logString += Temperature4  + ", ";
  logString3 += "Temp4 = " + Temperature4 + " Celcius, \n, ";
}
```

A segunda *procedure* foi denominada getDS18B20( ). A mesma utiliza *procedures* e funções nativas da biblioteca DallasTemperature.h, atribuídas ao objeto sensors, assim como, mais seis variáveis globais (uma para cada valor de temperatura e uma para o conjunto de valores, os quais serão transmitidos ao servidor Blynk), além da variável logString. Primeiramente é invocada a *procedure* requestTemperatures( ). Em seguida, se armazena o valor de cada temperatura em sua respectiva variável, concatena-se o mesmo aos dados já agrupados na variável logString e monta-se a variável logString3 para apresentação no aplicativo Blynk.

Esse procedimento se repete para cada um dos outros sensores de temperatura. Ao final tem-se cada temperatura armazenada em sua respectiva variável, a variável logString3 com a coletânea de todas as temperaturas incluindo sua identificação, sua unidade de medida e um comando de salto de linha (para a separação entre as mesmas).

A *string* de armazenamento (logString) recebe todos os valores de temperatura, separados por vírgula. Este processo segue a metodologia já utilizada para o módulo GPS, ou seja, faz-se previamente, a conversão do tipo de dado obtido (através da invocação da função getTempC( )) para um dado do tipo String.

O Código 15 apresenta as *procedures* para a estimativa do volume fitoplanctônico.

Código 15 – *Procedure* setPhyto( ) e getPhyto( )

```
void setPhyto(){
  pinMode(LDR_PIN, INPUT);
  pinMode(BLUE_LED, OUTPUT);
}

void getPhyto() {
  Serial.println("Iniciando rotina de leitura do LDR");
   String phytoString = " ";                              //string just for debugging on serial monitor
   voltage = 0.0;
   read_LDR = 0;

   for (int i = 1; i <= SAMPLE_NUMBER / 2; i++){          //measurement before the excitation
     int read_LDR_I = map(analogRead(LDR_PIN),DIGITAL_MIN,DIGITAL_MAX,0,1023);
     double voltage_I = read_LDR_I * (3.3 / 1023);
     phytoString = String(i) + " --> Leitura digitalizada = " + String(read_LDR_I);
     phytoString += ", Leitura Analogica = " + String(voltage_I) + "Volts;";
     Serial.println(phytoString);
   }

   Serial.println("   ");
   digitalWrite(BLUE_LED, HIGH);
   delay(60000);  // 60 segundos para aquecimento do led

   for (int i = 1; i <= SAMPLE_NUMBER; i++){   //measurement during the excitation
     int read_LDR_I = map(analogRead(LDR_PIN),DIGITAL_MIN,DIGITAL_MAX,0,1023);
     double voltage_I = read_LDR_I * (3.3 / 1023);
     phytoString = String(i) + " --> Leitura digitalizada = " + String(read_LDR_I);
     phytoString += ", Leitura Analogica = " + String(voltage_I) + "Volts;";
```

```
242.      Serial.println(phytoString);
243.      voltage += voltage_I;                //accumulating for average
244.      read_LDR += read_LDR_I ;             //accumulating for average
245.      delay(INTERVAL);
246.    }
247.    voltage = (voltage/SAMPLE_NUMBER);            //average of voltage
248.    read_LDR = int(read_LDR/SAMPLE_NUMBER);   //average of read LDR
249.    phytoString = "Leitura digitalizada média = " + String(read_LDR);
250.    phytoString +=  ", Leitura Analogica Media = " + String(voltage) + "Volts;";
251.    Serial.println(phytoString + "\n");
252.
253.    digitalWrite(BLUE_LED, LOW);
254.    logString += String(read_LDR) + ", " + String(voltage, 2) + ", \n";
255.
256.    for (int i = 1; i <= SAMPLE_NUMBER / 2; i++){     //measurement after the excitation
257.      int read_LDR_I = map(analogRead(LDR_PIN),DIGITAL_MIN,DIGITAL_MAX,0,1023);
258.      double voltage_I = read_LDR_I * (3.3 / 1023);
259.      phytoString = String(i) + " -->  Leitura digitalizada = " + String(read_LDR_I);
260.      phytoString += ", Leitura Analogica = " + String(voltage_I) + "Volts;";
261.      Serial.println(phytoString);
262.    }
263.  }
```

A *procedure* setPhyto( ) é invocada durante o setup. Ela configura o pino de que receberá a tensão oriunda do sensor de luminosidade como um pino de entrada, assim como o pino para ativação do emissor de luz como um pino de saída. A seleção e declaração desses pinos foram realizadas previamente no Código 7 (linhas 80 e 81).

A *procedure* getPhyto( ) se encarrega da rotina de comando do emissor e leitura do receptor de luz. Nela é criada uma variável local do tipo String (phytoString), para o acompanhamento via monitor serial nas medições individualizadas, além de mais duas outras (restritas ao *loop* "for") para o armazenamento temporário de cada uma das leituras de intensidade luminosa. A variável read_LDR_I recebe o valor digitalizado da tensão (entre 0 e 1024), enquanto a variável voltage_I recebe o valor de read_LDR_I convertido para o valor equivalente de tensão (entre 0 e 3,3V) e a variável phytoString recebe o texto que será exibido no monitor serial. O código possui três *loops* do tipo "*for*", sendo o primeiro destinado a medições da intensidade luminosa, no momento anterior a excitação pelo LED (linhas entre 225 e 231), o segundo realiza a medição propriamente dita, enquanto o terceiro (linhas entre 256 e 262) realiza a medição após o desligamento de LED (de forma a avaliar o decaimento da luminosidade). O primeiro e o terceiro *loops* utilizam a metade da quantidade de interações do segundo *loop* (*loop* de medição).

Na linha 234 inicia-se o procedimento de medição propriamente dita. O pino digital referente ao comando do LED é energizado, aguarda-se 60 segundos para o aquecimento e estabilização do mesmo, e inicia-se o *loop* de medição. A cada execução do *loop* “for” é realizada a leitura do pino analógico (A0), convertendo-o para a faixa de correlação com o volume do fitoplâncton (linha 238) e armazenando-o na variável local read_LDR_I, em seguida, o valor é convertido para a o seu equivalente em Volts (linha 239) e armazenado na variável voltage_I. As linhas de 240 a 242 montam a montagem e a apresentação da *string* referente à respectiva interação, no monitor serial. Na continuação do *loop*, acumula-se o valor das variáveis locais voltage_I e read_LDR_I, nas variáveis globais voltage (linha 243) e read_LDR (linha 244), respectivamente. Ao final faz-se a contagem do intervalo entre as interações do *loop*.

Após o termino do *loop* é calculado o valor médio do total de medições definido previamente em SAMPLE_NUMBER. Concluindo o procedimento o emissor (LED) é desligado (linha 253) e os valores médios são adicionados à *string* de manipulação (linha 254), concluindo a mesma.

Quanto à correlação entre a leitura analógica e volume estimado do fitoplâncton, o código apresentado, até o presente momento, mantem o valor original da leitura, pois a correlação é parte do estudo subsequente ao atual. Entretanto, o valor obtido torna-se útil, na medida em que a sua alteração já é, proporcionalmente, representativa da alteração do volume, permitindo análises qualitativas.

O Código 16 apresenta a função logData( ), a qual é a única função criada neste projeto. Seu retorno é um bit que confirma o sucesso da execução da mesma.

Código 16 – Function logData( )

```
byte logData(){
  Serial.println("Trying logData.");
  File logFile = SD.open(logFileName, FILE_WRITE); // Open the log file
  logString = " ";
  if (logFile){
    getGPS_Data();
    getDS18B20();
    getPhyto();
    Serial.print(logString);
    logFile.print(logString);
    logFile.close();
    return 1; // Return success
  }
  return 0; // If we failed to open the file, return fail
}
```

O código inicia-se pela abertura do arquivo criado pela *procedure* updateFileName( ) e previamente armazenado no cartão SD. Caso o acesso para a edição do arquivo seja bem sucedido, vai iniciar-se a chamada às *procedures* na seguinte ordem: getGPS_Data( ), getDS18B20( ), getPhyto( ). Essas *procedures* efetuarão a leitura e processamento de todos os dados que serão armazenados cartão e exibidos no aplicativo Blynk.

Em seguida, é exibida no monitor serial a *string* a ser armazenada, efetua-se a devida escrita no arquivo e o seu consecutivo "fechamento".

A *procedure* callBlynk( ) é responsável por enviar os dados para o servidor do aplicativo Blynk. O Código 17 apresenta os comandos de invocação da *procedure* Blynk.virtualWrite( ), oriunda da biblioteca BlynkSimpleEsp8266 (a qual é específica para o microcontrolador utilizado). Cada invocação envia uma determinada informação para uma variável específica do servidor Blynk (V0 a V7), a qual é imediatamente exibida no aplicativo desenvolvido.

Código 17 – *Procedure* callBlynk( )

```
284. void callBlynk(){
285.   Blynk.virtualWrite(V0,"Longitude,  Latitude,   Alt, Sats");
286.   Blynk.virtualWrite(V0,logString2);   //Data from GPS module
287.   Blynk.virtualWrite(V0,logString3);   //Temperature from BS18B20
288.   Blynk.virtualWrite(V1,String(voltage,2));//Phytoplankton from 0 to 3.3 V
289.   Blynk.virtualWrite(V2,Temperature0); //Temperature on surface
290.   Blynk.virtualWrite(V3,Temperature1); //Temperature on 1st level
291.   Blynk.virtualWrite(V4,Temperature2); //Temperature on 2nd level
292.   Blynk.virtualWrite(V5,Temperature3); //Temperature on 3rd level
293.   Blynk.virtualWrite(V6,Temperature4); //Temperature on 4th level
294.   Blynk.virtualWrite(V7,read_LDR);     //Phytoplankton from 0 to 1023
295. }
```

Concluído o programa estão as *procedures* setup ( ) e loop( ), as quais são obrigatórias na arquitetura do microcontrolador escolhido.

O Código 18 se inicia apresentando a *procedure* setup( ), a qual é executada apenas uma vez, a cada inicialização do microcontrolador. A mesma é responsável por executar os comandos de configuração necessários ao funcionamento do programa. A linha 300 e 301 inicializam a comunicação com o monitor serial (em 115.200 bps) e com módulo GPS (em 9.600). No caso da substituição do NodeMCU por um microcontrolador de arquitetura Arduino, ambas a taxas de transferência devem ser 9.600 bps. As linhas 302 e 303 fazem as chamadas das *procedures* setDS18B20( ), setPhyto( ), as quais configuram os sensores DS18B20 e o sensor de fotofluorescência. A linha 304 invoca a *procedure* beginSDcard( ), a qual inicializa o cartão SD e invoca as *procedures* updateFileName( ) e printHeader( ). Por fim, são invocadas *procedures* nativas da biblioteca BlynkSimpleEsp8266. A *procedure* Blynk.begin(auth, ssid, pass) inicializa a comunicação com o servidor, a partir da autenticação, do nome da rede local e de sua respectiva senha. A *procedure* timer.setInterval(LOG_RATE, callBlynk) configura o intervalo entre as chamadas à *procedure* callBlynk( ).

Código 18 – *Procedures* setup( ) e loop( )

```
299. void setup(){
300.   Serial.begin(115200);          // Sets serial monitor speed and start
301.   GPS.begin(9600);               // Sets serial GPS speed and start
302.   setDS18B20();                  // Sets Up DS18B20 Sensors
303.   setPhyto();                    // Sets Up for the procedure getPhyto
304.   beginSDcard();                 // Tests the SD card and initiales the file for log
305.                                  // Here will be called the updateFileName() and
306.                                  // printHeader() procedures
307.   Serial.println("\n *** This is the sdCard_GPS_Phyto_06_rB Program ***");
308.   Blynk.begin(auth, ssid, pass);// Sets the initial Blynk App parameters
309.   timer.setInterval(LOG_RATE, callBlynk); // Sets the interval for Blynk App refresh
310. }
311.
312. void loop(){
313.   int a = millis();
314.   if ((lastLog + LOG_RATE) <= a){          // If it's been LOG_RATE milliseconds since the last log
315.     if (tinyGPS.location.isUpdated()){     // If the GPS data is vaild
316.       if (logData()){                      // Log the GPS data
317.         Serial.println("GPS logged.");     // Prints a debug message
318.         lastLog = millis();                // Updates the lastLog variable
319.       }
320.       else {                               // If we failed to log GPS, print an error, don't update lastLog
321.         Serial.println("Failed to log new data.");
322.       }
323.     }
324.     else { // If GPS data isn't valid, print a debug message
325.       Serial.print("No GPS data. Sats: ");
326.       Serial.println(tinyGPS.satellites.value());
327.     }
328.   }
329.   // If we're not logging, continue to "feed" the tinyGPS object:
330.   while (GPS.available()){
331.     tinyGPS.encode(GPS.read());
332.   }
333.   Blynk.run();
334.   timer.run(); // Initiates BlynkTimer
335. }
```

A *procedure* loop( ) é executada ciclicamente pelo microcontrolador. A mesma inicia-se verificando se o intervalo definido entre as execuções foi decorrido (linhas 313 e 314). Em seguida, verifica a validade das informações oriundas do GPS e depois invoca e verifica a execução da *procedure* logData( ). Caso ocorra alguma falha, é enviada uma mensagem ao monitor serial. As linhas entre 330 e 332 são responsáveis por coletar as informações do GPS. Por fim, são invocadas as *procedures* Blynk.run( ) e timer.run( ) para a atualização das informações no servidor Blynk.

O aplicativo criado para visualização (Figura 13) pode ser acessado de qualquer smartphone Android ou iOS. O mesmo recebe os dados do servidor Blynk e exibe as informações pertinentes.

Figura 13 – Telas do aplicativo Blynk

Fonte: O Autor

4.9 Calibração do sistema de estimativa do fitoplâncton

A verificação da proporcionalidade entre a diluição das amostras (utilizadas como padrão), comparando-as às leituras obtidas com o fluorímetro comercial (Aquafluor). A Figura 14 demonstra esta comparação, caracterizando as amostras manipuladas como válidas (R^2 = 0,99).

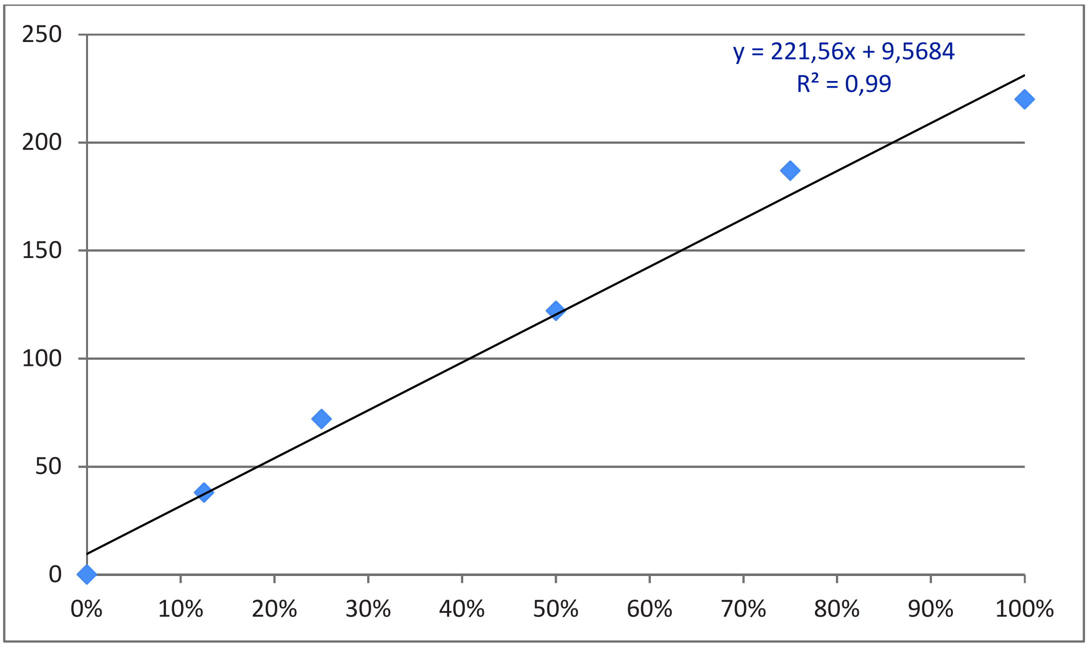


Figura 14 – Gráfico de dispersão das amostras padrão (em %) e das leituras do fluorímetro comercial

Fonte: O Autor

As mesmas amostras foram utilizadas na medição com o sistema desenvolvido, onde foram obtidos os resultados apresentados na Figura 15. Nota-se uma correlação entre os valores, com $R^2 = 0{,}87$.

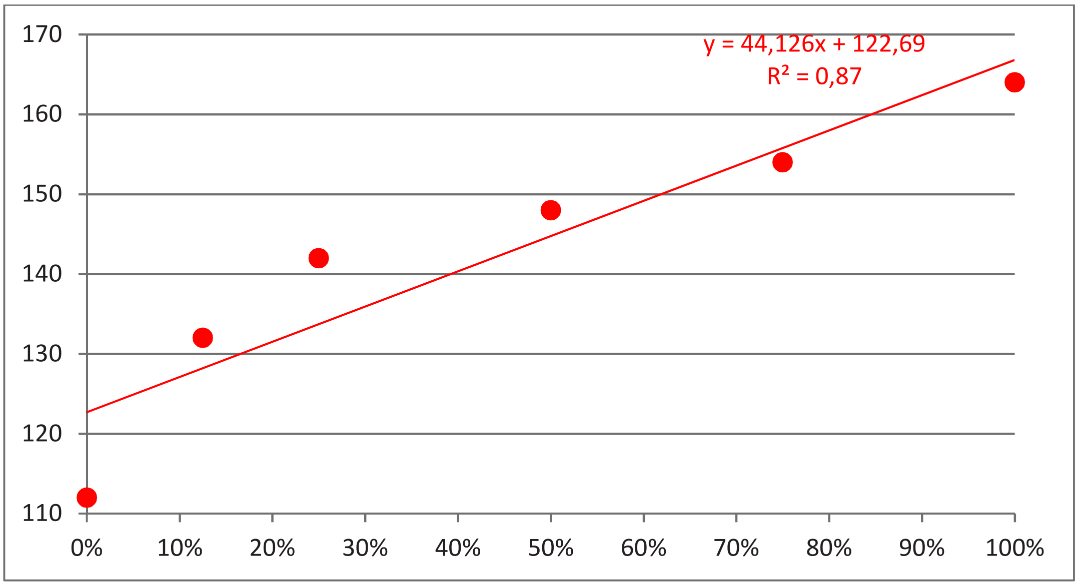


Figura 15 – Gráfico de dispersão das amostras padrão (em %) e das leituras do sistema desenvolvido

Fonte: O Autor

A comparação direta entre as leituras do sistema desenvolvido e as do fluorímetro comercial permitiu elaborar o gráfico de dispersão apresentado na Figura 16. Nele observa-se um R^2 de 0,90 (considerando-se as concentrações entre 0 e 100%), porém, considerando-se apenas as concentrações entre 12,5 e 100%, o R^2 cresce para 0,96.

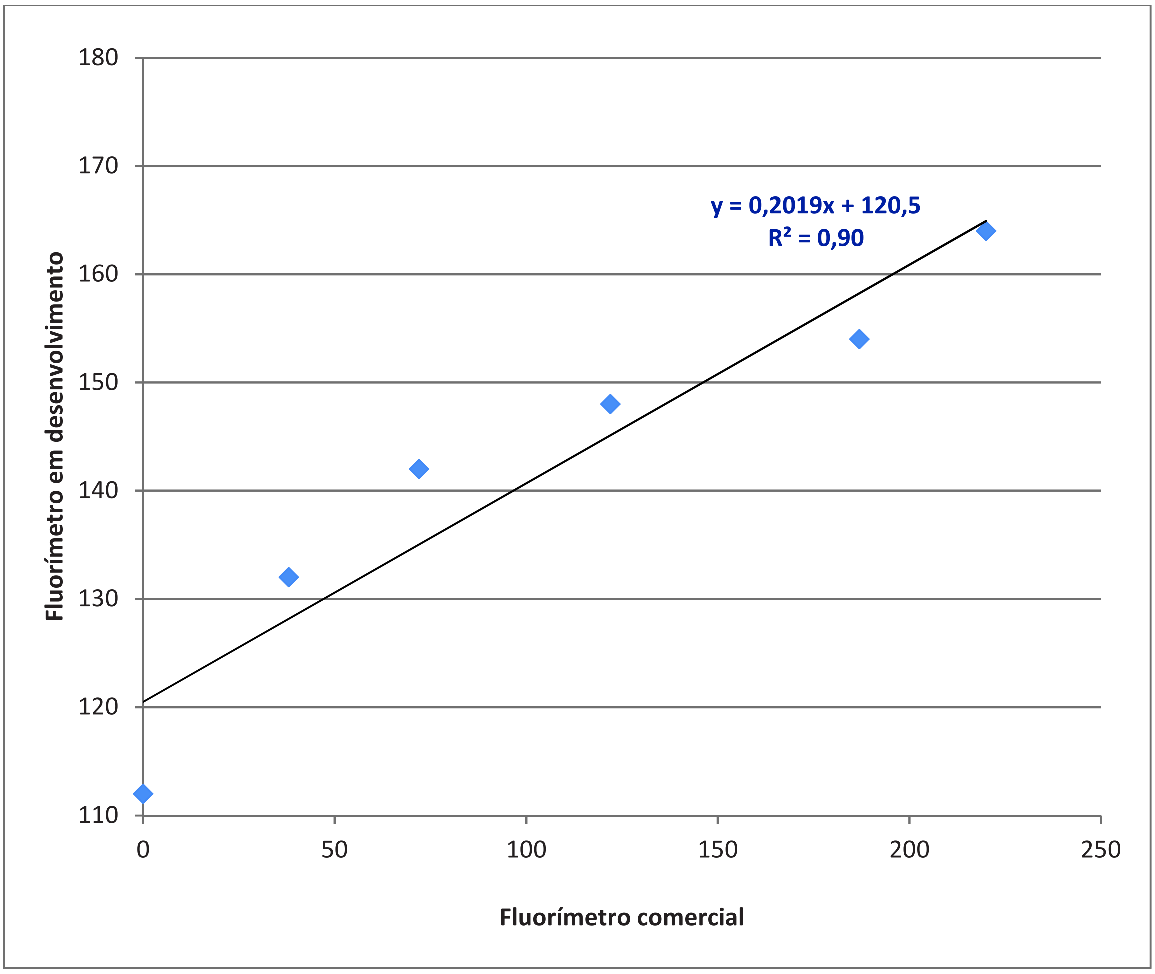


Figura 16 – Gráfico de dispersão entre as leituras do sistema desenvolvido e do fluorímetro comercial

Fonte: O Autor

Este fato sugere que o sensor desenvolvido apresenta uma maior variabilidade quando trabalhando com concentrações baixas.

Foram analisadas amostras naturais, oriundas de corpos hídricos distintos e obtidas durante os testes de campo (Apêndice E). Os valores lidos de cada conjunto de amostras (com suas respectivas legendas) foram sobrepostos ao gráfico da Figura 16, gerando a Figura 17.

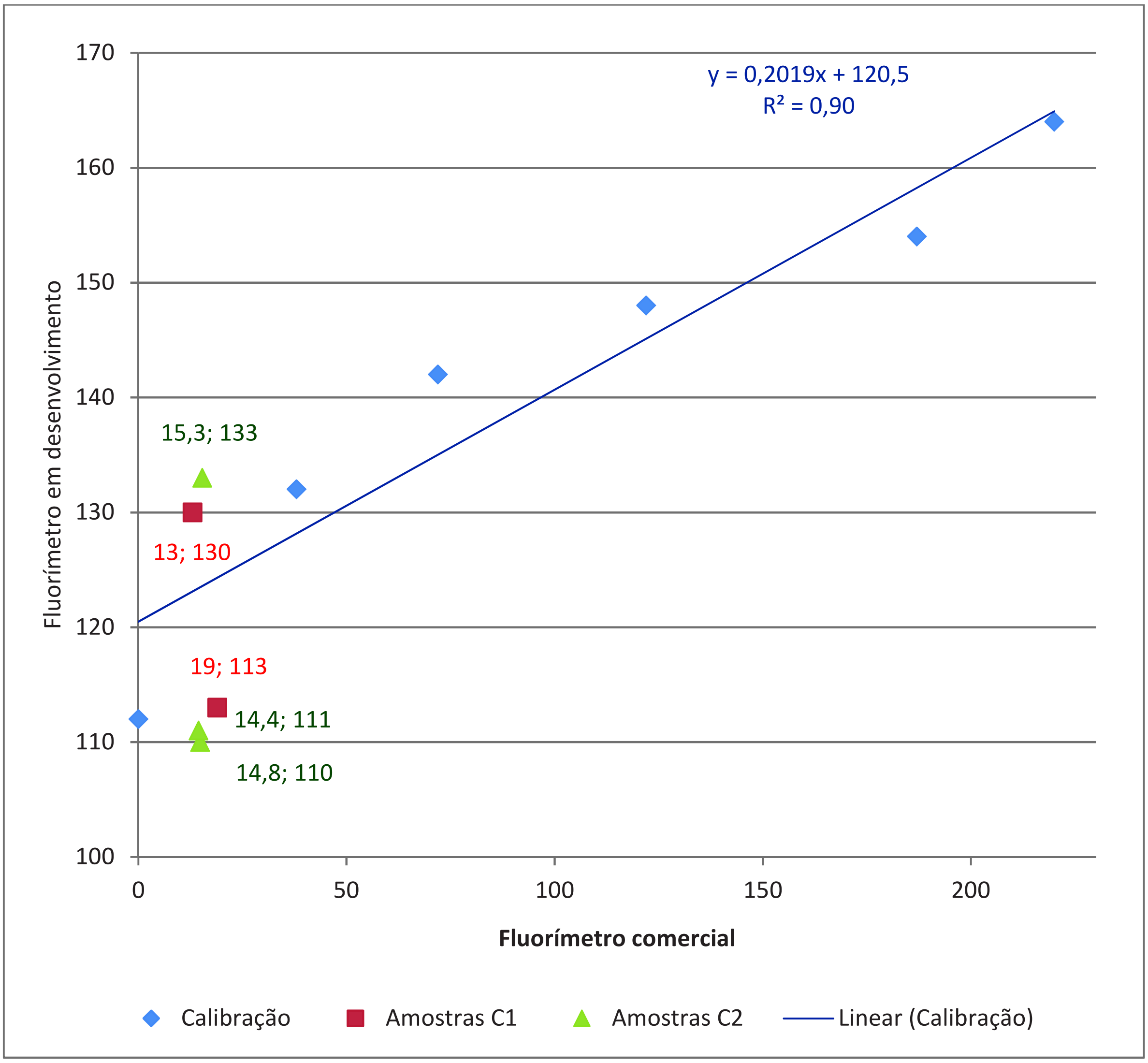


Figura 17 – Gráfico de dispersão entre as leituras do sistema desenvolvido e do fluorímetro comercial, incluindo amostras naturais

Fonte: O Autor

As duas amostras indicadas pelos quadrados são oriundas da atividade de campo 1 (C1), já as três indicadas por triângulos, são oriundas da atividade de campo 2 (C2).

Considerando a Equação 1, extraída da linha de tendência (apresentada no gráfico das Figuras 14 e 15), na qual a variável *x* representa o valor lido pelo Aquafluor, calcularam-se os valores esperados de leitura (para cada amostra natural), no sistema desenvolvido.

$$y = 0{,}2024x - 120{,}54 \qquad (1)$$

Comparando-se as leituras esperadas às leituras obtidas no sistema, foi possível calcular as diferenças absolutas e relativas. Os resultados estão expressos no Quadro 7.

Quadro 9 – Diferenças (absoluta e relativa) entre as leituras

	Leitura obtida no AquaFluor	Leitura Esperada do sistema	Leitura obtida no sistema	Diferença Absoluta	Diferença Relativa
C1_1	19,0	124,4	113,0	-11,3	-9%
C1_2	13,0	123,1	130,0	6,9	6%
C2_1	14,8	123,5	110,0	-13,5	-11%
C2_2	14,4	123,4	111,0	-12,4	-10%
C2_3	15,3	123,6	133,0	9,4	8%

Fonte: O Autor

Os resultados obtidos sugerem que o equipamento desenvolvido, quando utilizado em bancada (tal qual o Aquafluor), apresenta desempenho semelhante ao equipamento comercial, com uma margem de erro em torno de ±10%. No entanto, esta margem de erro é perfeitamente aceitável, se considerarmos o baixo custo do sistema, em relação aos instrumentos comerciais, a sua manutenção e a possibilidade de personalização, e potencialização do sistema ótico e sensor.

Os resultados das amostras mensuradas *in situ* apresentaram variações maiores, em relação ao equipamento comercial, sugerindo que há interferência da luminosidade no compartimento do sensor. No entanto, deve-se levar em consideração que esta diferença pode ser considerada baixa se relacionada à variabilidade característica dos sistemas hídricos naturais.

O projeto em questão abre a possibilidade da produção de um quantitativo maior de módulos a serem utilizados em uma pesquisa de campo ampliando o volume de dados em tempo real e a produção de dados complementares tais como posicionamento, hora que podem derivar a velocidade e direção da corrente do corpo hídrico estudado.

4.10 Testes de Campo

Antes da realização do teste, foi realizado um pré-teste de flutuabilidade/estabilidade ou um teste hidrodinâmico conforme pode ser observado na Figura 18. Posteriormente, foram realizados alguns ajustes no tamanho do flutuador, assim como, foi realizada a perfuração do suporte dos sensores de temperatura. Essa solução evitou a necessidade de aumento da massa de lastro para o conjunto, o que resultaria em um considerável aumento de peso do mesmo. O referido pré-teste foi realizado em ambiente marinho, na praia de Guaxindiba (São Francisco do Itabapoana – RJ).

Figura 18 – Pré-teste de flutuabilidade e protótipo da unidade flutuante

Fonte: O Autor

O primeiro evento de teste de campo foi realizado em 25/01/2019, entre 09h00min e 12h20min, na Lagoa Limpa de Travessão (Figura 19), nas coordenadas geográficas 21°37'27.8"S 41°21'51.4"W.

Figura 19 – Localização do primeiro evento de teste de campo (C1)

Fonte: O Autor

Durante alguns períodos, o protótipo foi deixado à deriva, porém seu deslocamento foi mantido dentro de curtos trajetos (devido à dificuldade de resgate do sistema, no caso de um maior afastamento), esse fato tornou as variações de posição pouco relevantes. A maior parte do período o sistema foi mantido fundeado em posição fixa. Foram recolhidas duas amostras para posterior teste em condições de laboratório (Figura 20).

Figura 20 – Primero teste de campo (C1): Modo deriva e modo fundeio

Fonte: O Autor

O segundo teste de campo foi realizado em 01/02/2019, entre 18h50min e 20h20min, em um tanque de criação de peixes, localizado nas coordenadas 21°35’35.7”S 41°18’58.7”W (Figura 21), na localidade de Travessão (Campos dos Goytacazes – RJ).

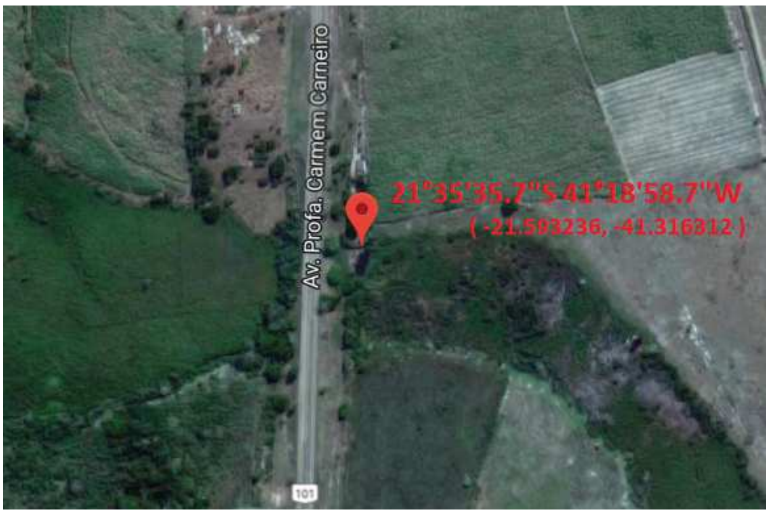


Figura 21 – Localização do segundo evento de teste de campo (C2)

Fonte: O Autor

O teste da integração do GPS foi realizado em trajeto rodoviário com a mesma taxa de amostragem prevista para a aplicação real (uma a cada 5 minutos). Paralelamente, foi realizado um teste de autonomia da bateria, a qual permitiu 20 horas e 10 minutos de medições contínuas. Mediante a isso, para campanhas de maior duração, será necessária a utilização de uma fonte que forneça a adequada autonomia.

A figura 22 demostra o resultado do teste do GPS. Os dados utilizados foram, apenas, os relacionados ao período de deslocamento (21h24min às 22h25min). Durante o restante do período, não houve deslocamento, tornando seus dados repetitivos e não relevantes.

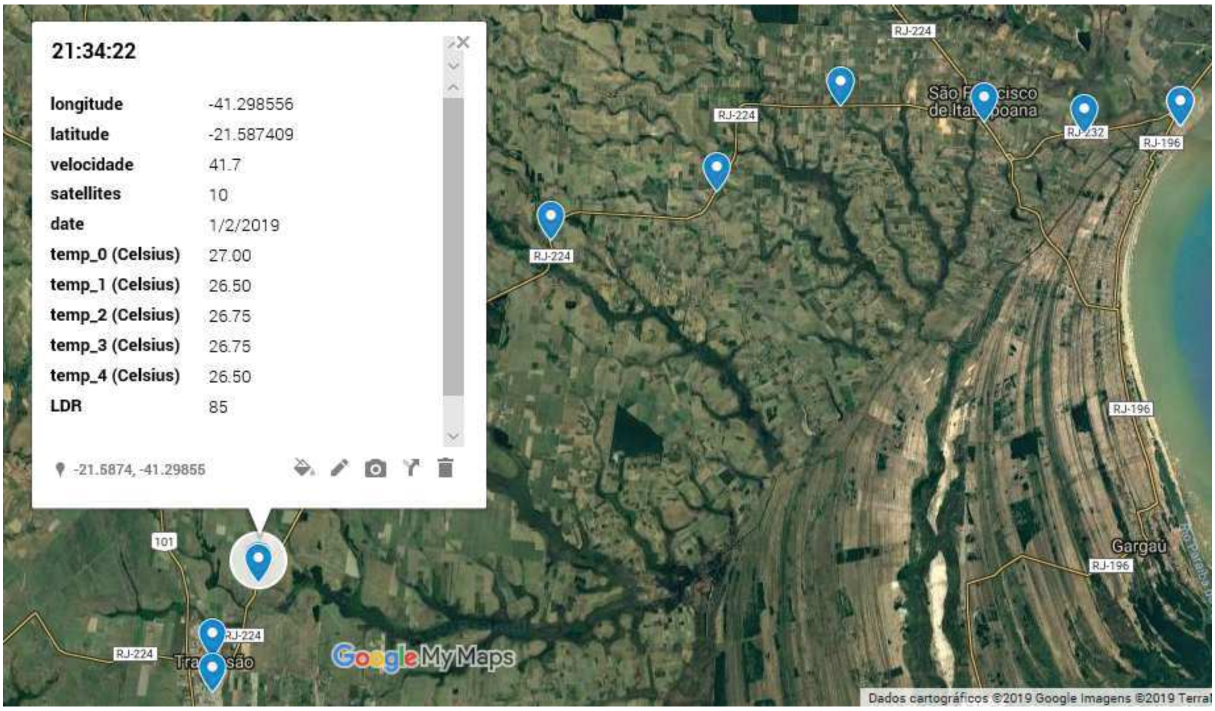


Figura 22 – Evidências do teste do GPS

Fonte: O Autor

A realização do segundo evento de teste de campo (C2), se deu em 01/02/2019, sendo estrategicamente escolhido, um horário que coincidia com uma grande variação luminosa e, consequente, sensível resfriamento da água. O gráfico da Figura 23 apresenta a variação das temperaturas em cada profundidade. Observa-se que a temperatura superficial e as temperaturas situadas a 15 e 30 cm de profundidade, sofreram uma perceptível redução, variando de 32,20 a 30,7 °C, enquanto, as temperaturas situadas a 45 e 60 cm de profundidade apresentaram uma estratificação mais estável com pequena oscilação, de no máximo 0,5 °C em torno do valor inicial.

Estes resultados demonstram que o sistema apresenta sensibilidade às variações de temperatura identificando, através de resultados com comportamentos semelhantes, três estratos diferenciados de temperatura: o primeiro, situado entre a faixa de 0 a 30 cm de profundidade, e caracterizado por comportamentos semelhantes de decréscimo da temperatura, o segundo, na faixa de 45 cm e o terceiro na faixa de 60 cm.

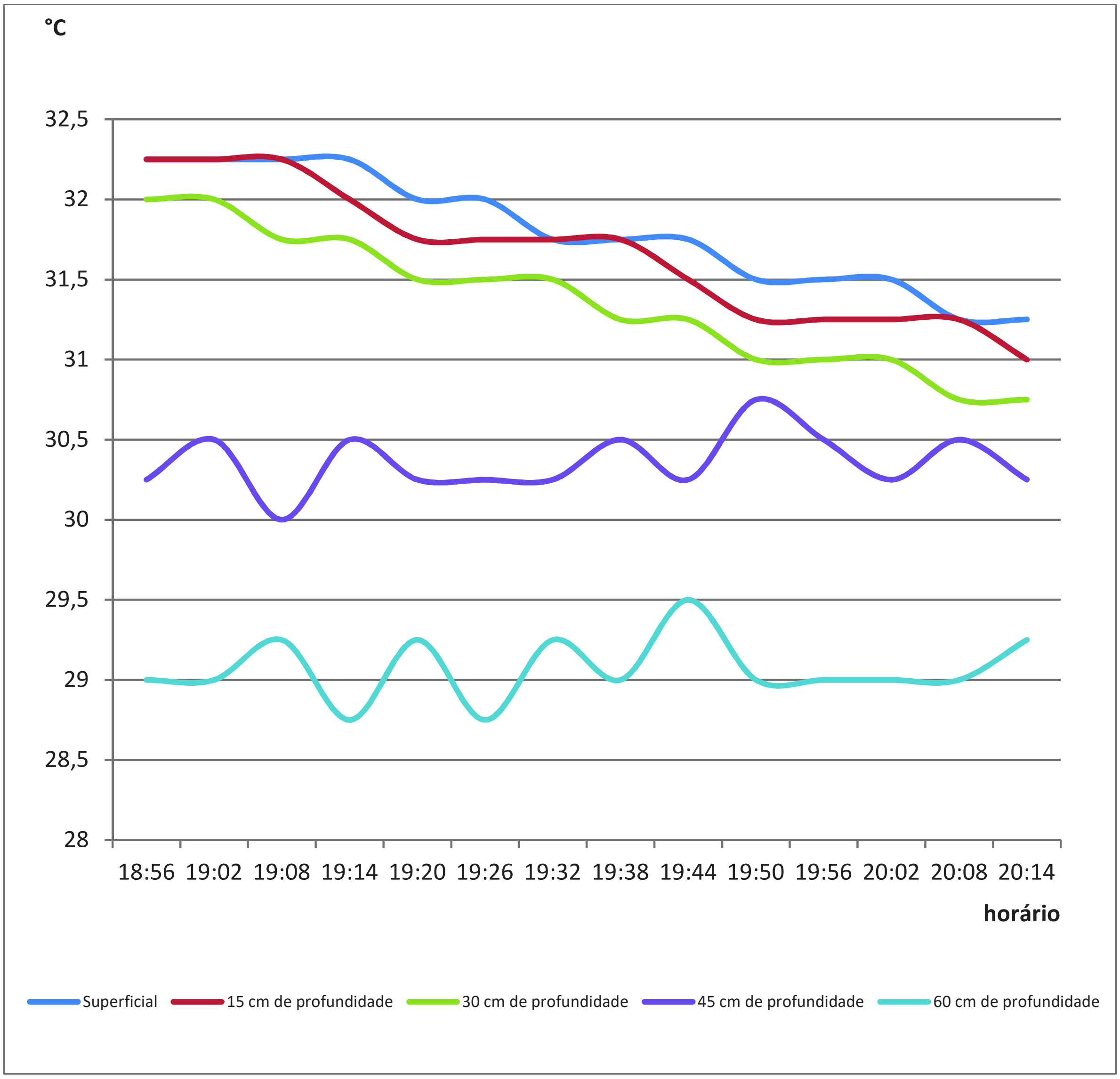


Figura 23 – Estratificação de temperatura entre a superfície e 60 cm de profundidade

Fonte: O Autor

Esses resultados demonstram a eficiência do sistema para medição das temperaturas em múltiplas profundidades. Essa constatação, aliada à possibilidade de personalização da disposição dos sensores de temperatura (quantidade, espaçamento e posicionamento), atesta a viabilidade do dispositivo para a referida finalidade.

No que tange aos resultados de fotofluorescência, verificou-se o adequado armazenamento das variações ao longo do período, as quais puderam ser comparadas às variações de temperatura ocorridas na região da instalação do sensor de fotofluorescência (Figura 24).

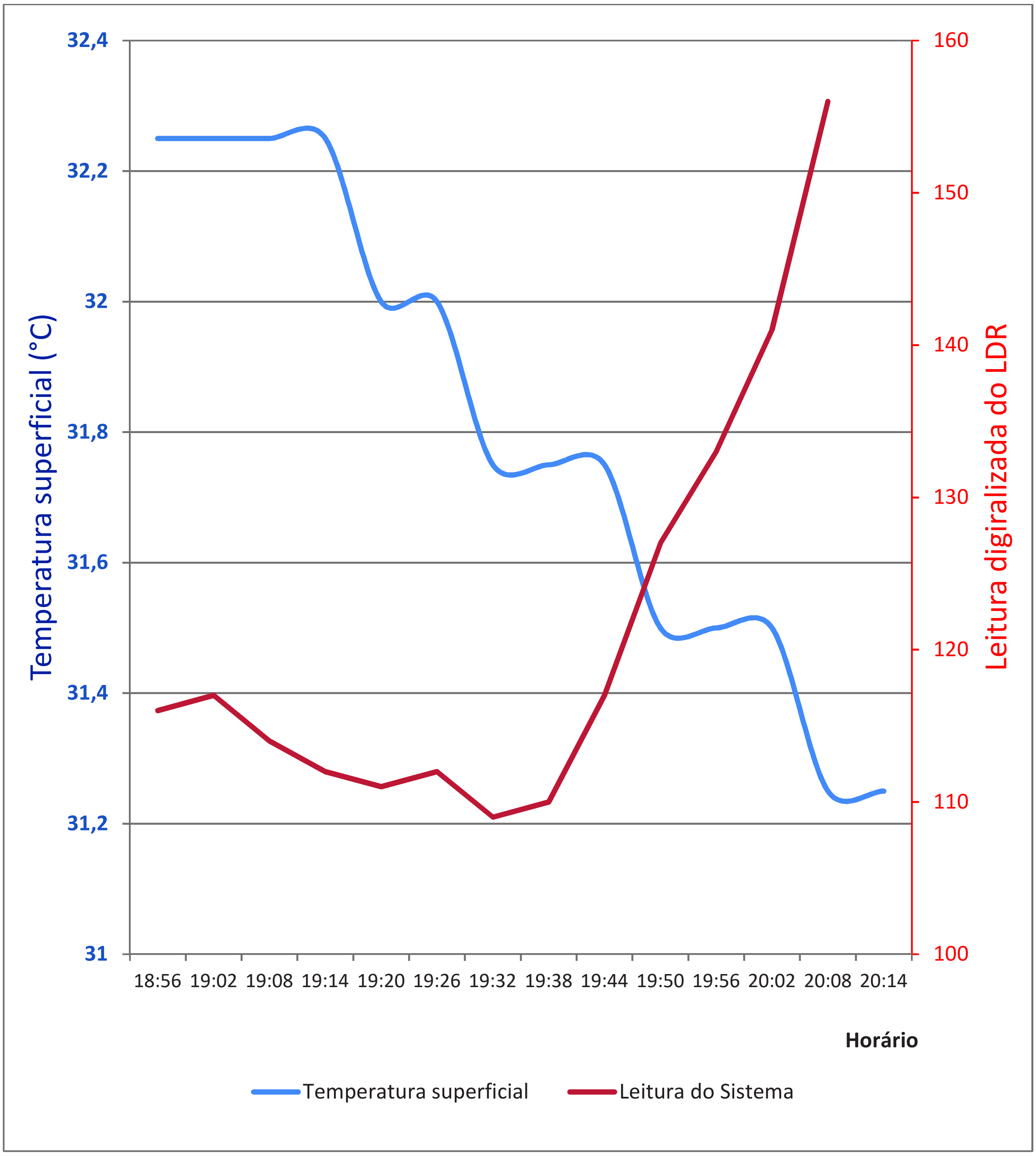


Figura 24 – Comparação entre a variação da temperatura no local de instalação do sensor e a sua medição (fluorescência) ao longo do período

Fonte: O Autor

4.11 Análise técnico-financeira

A Tabela 1 detalha o custo aproximado (à época desta publicação), dos principais componentes necessários para a construção da parte eletrônica do sistema.

Tabela 1 – Custo aproximado dos principais componentes eletrônicos.

Componente	Preço médio (R$)
NodeMCU V3	25,00
Módulo GPS c/antena	55,00
Módulo leitor de cartão SD	10,00
Cartão SD 8GB	15,00
Sensor de temperatura DS18B20	11,00
Power Bank Usb 28000mAh	45,00
Led 3w 3v Azul Royal	3,00
LDR 5 mm	1,00
Resistor 10k Ohms	1,00
Resistor 4k7 Ohms	1,00
TOTAL*	**167,00**

*Considerando-se uma única unidade do DS18B20. Para cinco unidades, custo de R$ 211,00.

Fonte: O Autor

Caso o sistema seja utilizado exclusivamente em modo fundeio, pode-se substituir o módulo GPS por um módulo de relógio de tempo real (RTC DS3231 – custo aproximado de R$ 10,00) o que reduziria o custo dos componentes eletrônico para, aproximadamente, R$122,00. A referida substituição acarretaria em uma redução do consumo energético e, possivelmente, admitiria uma fonte com menor potência, gerando assim, uma redução adicional de custo.

5 CONCLUSÕES

O objetivo principal da pesquisa em desenvolver um sistema embarcado em unidade flutuante, para monitoramento de reservatórios e mananciais hídricos, visando à medição da fotofluorescência, relacionada à estimativa do volume do fitoplâncton, assim como, o monitoramento da temperatura em múltiplas profundidades foi plenamente atingido.

A pesquisa bibliográfica realizada identificou que a quantidade de publicações relacionadas ao assunto abordado apresentou um crescimento consistente, principalmente quando se considera o período entre 2008 e 2017. Na análise dos aspectos mais relevantes das publicações, pode-se perceber a predominância da utilização da arquitetura Arduino (para a função de plataforma de desenvolvimento) e do uso de LED como fonte de excitação luminosa para a fotofluorescência. Quanto ao elemento sensor, as propostas mais flexíveis utilizaram fluorímetro comerciais, enquanto nas propostas mais dedicadas houve a predominância do uso de sensores de intensidade luminosa (tipo LDR).

Apesar da citada predominância da plataforma Arduino (e suas variantes) entre os trabalhos acadêmicos analisados, este trabalho buscou analisar também, outras concepções tecnológicas mais recentes, porém menos consolidadas. Nesta análise identificou-se o microcontrolador ESP8266-12 (plataforma NodeMCU) como uma alternativa mais interessante para a aplicação em questão, concluindo em sua utilização. Partindo dessa escolha, analisaram-se as três linguagens de programação disponíveis para esse microcontrolador, optando-se pelo uso de uma variante da linguagem C, a qual coincide com a já utilizada na arquitetura Arduino (aproveitando também a tradicional interface de programação do mesmo).

Quanto aos sensores de temperatura optou-se pela utilização do DS18B20, pois o mesmo atende simultaneamente aos requisitos de faixa de medição, sensibilidade e custo, além de possuir uma versão à prova d’água. Outro aspecto importante é o compartilhamento do canal de comunicação por vários sensores, economizando portas de I/O do microcontrolador e facilitando a expansão da quantidade de sensores do sistema.

Apresentou-se a modelagem de software e hardware, incluindo seus detalhes construtivos. Apresentou-se também, o código de programação desenvolvido, explicando-o em sua totalidade.

A construção do sensor de fotofluorescência esta detalhada no Apêndice A e sua calibração foi realizada efetuando-se medições em amostras específicas (as quais foram consideradas como amostras padrão), comparando-as às leituras obtidas com um fluorímetro comercial. A comparação direta entre as leituras do sistema desenvolvido e as do fluorímetro comercial permitiu elaborar um gráfico de dispersão, no qual se observa um R^2 de 0,90. Comparando o desempenho do sistema ao do referido fluorímetro, em medição de amostras naturais (extraídas de corpos hídricos reais), foi demonstrada a existência de uma margem de variação de aproximadamente 10%, a qual foi entendida como aceitável, tendo em vista a variabilidade característica dos sistemas hídricos naturais. Assim, atingiu-se o objetivo secundário de construir um sensor ótico para medição da fotofluorescência.

Os testes de campo ocorrem em sua maior parte com o conjunto fundeado, sendo realizados dois eventos em locais e horários distintos, utilizando uma frequência de uma amostragem a cada 5 minutos. Um terceiro teste teve foco na autonomia da bateria, demostrando que sua carga foi capaz de perdurar por 20 horas e 10 minutos de medições contínuas.

Durante o segundo evento de teste de campo (C2), foi estrategicamente escolhido, um horário que coincidia com uma grande variação luminosa e sensível resfriamento da água. Percebeu-se que que a temperatura superficial e as temperaturas situadas a 15 e 30 cm de profundidade, sofreram uma perceptível redução, enquanto, as temperaturas situadas a 45 e 60 cm de profundidade sofreram apenas pequenas oscilações (de no máximo 0,5 °C) em torno do valor inicial. Percebeu-se também que, comparando-se as variações de fluorescência com as da temperatura nas proximidades do sensor, ocorreu uma inversão de tendência por volta de 19h30min.

A realização deste projeto materializou os conhecimentos em uma aplicação específica, a qual envolveu o desenvolvimento, a prática construtiva e a utilização do produto. Os resultados obtidos demonstram a utilização do sistema para medição das temperaturas em múltiplas profundidades, assim como, para a medição da fluorescência. Atendendo assim, aos demais objetivos secundários.

Considera-se que os dados obtidos pelo sistema constituem um banco de dados relevante para análises de comportamento de corpos hídricos, os quais poderão servir como insumo para outros pesquisadores.

O sistema desenvolvido apresenta custo acessível e possibilidade de personalização (quanto a quantidade, espaçamento e posicionamento dos sensores de temperatura). Esses fatos sugerem a viabilidade do equipamento para o monitoramento da qualidade de água superficial, podendo atender a empresas de fornecimento de água, hidrelétricas, unidades de conservação ambiental, assim como, qualquer outra pesquisa que dependa de monitoramento geoquímico ambiental, podendo inclusive, ser aplicado com múltiplas unidades simultâneas, ampliando a cobertura em escalas de tempo e espaço diferenciadas.

As próximas etapas do desenvolvimento são: a calibração do sistema, mediante a utilização de padrão de Rodamina WT 400ppb, de forma a correlacionar a fluorescência relativa a uma estimativa de concentração (µg/L) de clorofila *in vivo* e a realização de teste de longa duração na Lagoa Limpa de Travessão. Essas próximas etapas estão previstas para ocorrer até dezembro de 2019, conforme projeto aprovado e auxílio financeiro já concedido pela AGEVAP, no Edital Nº 004/2018 (Seleção pública para concessão de auxílio financeiro para elaboração de trabalhos técnicos e científicos com recursos financeiros oriundos da cobrança pelo uso dos recursos hídricos na Região Hidrográfica Baixo Paraíba do Sul e Itabapoana).

Como possíveis trabalhos futuros, sugere-se a agregação de outros tipos de sensores ao sistema, o aprimoramento da estrutura flutuante e a aplicação do sistema no monitoramento de outros corpos hídricos (por exemplo: Lagoa de Campelo, Lagoa de Gargaú, Lagoa Feia e Lagoa de Carapebus).

REFERÊNCIAS BIBLIOGRÁFICAS

AGEVAP. **Bacia Hidrográfica do Rio Paraíba do Sul**. Disponível em: <http://www.ceivap.org.br/downloads/mapa2.jpg>. Acesso em: 17 jun. 2018.

ALBALADEJO, C. *et al*. Wireless Sensor Networks for Oceanographic Monitoring: A Systematic Review. **Sensors (Basel, Switzerland)**, v. 10, n. 7, p. 6948–6968, 19 jul. 2010.

ALVAREZ-CAMPANA, M. *et al*. Smart CEI Moncloa: An IoT-based Platform for People Flow and Environmental Monitoring on a Smart University Campus. **Sensors**, v. 17, n. 12, p. 2856, 8 dez. 2017.

ARDUINO. **Arduino - Software**. Disponível em: <https://www.arduino.cc/en/Main/Software>. Acesso em: 6 mar. 2018a.

ARDUINO. **Arduino - Home**. Disponível em: <https://www.arduino.cc/>. Acesso em: 18 ago. 2018b.

AYMERICH, I. F. *et al.* Analysis of discrimination techniques for low-cost narrow-band spectrofluorometers. **Sensors (Basel, Switzerland)**, v. 15, n. 1, p. 611–634, 30 dez. 2014.

BARBOSA, P. C. D. C. **Aplicações de Fluorescência Induzida por Laser em Monitoramento Ambiental**. Doutorado—Rio de Janeiro: Pontifícia Universidade Católica do Rio de Janeiro, 27 out. 2003.

BARDAJI, R. *et al.* Estimating the Underwater Diffuse Attenuation Coefficient with a Low-Cost Instrument: The KdUINO DIY Buoy. **Sensors**, v. 16, n. 3, p. 373, 15 mar. 2016.

BERNARDO, L. DI. **Algas e suas influências na qualidade das águas e nas tecnolgias de tratamento**. Rio de Janeiro: ABES - Associação Brasileira de Engenharia Sanitária e Ambiental, 1995.

BEUTLER, M. *et al.* A fluorometric method for the differentiation of algal populations in vivo and in situ. **Photosynthesis Research**, v. 72, n. 1, p. 39–53, 2002.

BRAGA, B. *et al.* **Introdução à engenharia ambiental: o desafio do desenvolvimento sustentável**. 2ª. ed ed. São Paulo: Pearson Prentice Hall, 2005.

BRASIL. **LEI No 9.433/97: A política nacional de recursos hídricos e sua implementação no Distrito Federal.,** 1997. Disponível em: <https://ceapg.fgv.br/sites/ceapg.fgv.br/files/u60/politica_nacional_dos_recursos_hidricos.pdf>. Acesso em: 3 mar. 2018

BROOKE, D. *et al.* **Algas e Seus Impactos em Sistemas de Tratamento de Águas para Abastecimento: Estudo de Caso Sistema Guarapiranga**. Disponível em: <https://www.brookepeig.com/downloads/Algas.pdf>. Acesso em: 3 mar. 2018.

BUSCH, J. A. *et al.* Citizen Bio-Optical Observations from Coast- and Ocean and Their Compatibility with Ocean Colour Satellite Measurements. **Remote Sensing**, v. 8, n. 11, p. 879, 25 out. 2016.

CAMPOSTRINI, E. **FLUORESCÊNCIA DA CLOROFILA *a*: CONSIDERAÇÕES TEÓRICAS E APLICAÇÕES PRÁTICAS**. Campos, RJ, 2011. Disponível em: <http://www.uenf.br/Uenf/Downloads/CENTRO_CCTA_1629_1112121492.pdf>. Acesso em: 7 fev. 2019.

CULLEN, J. J. *et al.* Optical detection and assessment of algal blooms. **Limnology and Oceanography**, v. 42, n. 5part2, p. 1223–1239, jul. 1997.

ESPRESSIF. **ESP8266EX Datasheet**, 2018. Disponível em: <https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf>. Acesso em: 6 mar. 2018

FRIEDRICHS, A. *et al*. SmartFluo: A Method and Affordable Adapter to Measure Chlorophyll a Fluorescence with Smartphones. **Sensors**, v. 17, n. 4, p. 678, 25 mar. 2017.

HANELT, D. 9 - Photosynthesis assessed by chlorophyll fluorescence. In: HÄDER, D.-P.; ERZINGER, G. S. (Eds.). . **Bioassays**. [s.l.] Elsevier, 2018. p. 169–198.

INAG, I. P. **Manual para a avaliação da qualidade biológica da água. Protocolo de amostragem e análise para o Fitoplâncton.** Instituto da Água, I.P., 2009. Disponível em: <https://www.apambiente.pt/dqa/assets/protocolo-de-amostragem-e-an%C3%A1lise-para-o-fitopl%C3%A2ncton.pdf>. Acesso em: 3 mar. 2018

INAG, I. P. **Manual para a avaliação da qualidade biológica da água. Guia de utilização da tabela de valores - guia normalizado de biovolumes e determinação do biovolume através de procedimentos laboratoriais.** Instituto da Água, I.P., 2011. Disponível em: <http://www.azores.gov.pt/NR/rdonlyres/4ADD2DC8-A054-4EA4-8D85-3FC9B666294A/584911/GuiaTabelaeDeterminaoBiovolumesV10Dez2011.pdf>. Acesso em: 3 mar. 2018

IOT BYTES. **NodeMCU PinoutIoT Bytes**, 5 abr. 2016. Disponível em: <https://iotbytes.wordpress.com/nodemcu-pinout/>. Acesso em: 6 mar. 2018

LEARN OPENENERGYMONITOR. **Temperature Sensing using DS18B20 Digital Sensors**. Disponível em: <https://learn.openenergymonitor.org/electricity-monitoring/temperature/DS18B20-temperature-sensing?redirected=true>. Acesso em: 6 fev. 2019.

LEEUW, T.; BOSS, E. S.; WRIGHT, D. L. In situ Measurements of Phytoplankton Fluorescence Using Low Cost Electronics. **Sensors**, v. 13, n. 6, p. 7872–7883, 19 jun. 2013.

LOCKRIDGE, G. *et al.* Development of a Low-Cost Arduino-Based Sonde for Coastal Applications. **Sensors**, v. 16, n. 4, p. 528, 13 abr. 2016.

LUA. **Lua: about**. Disponível em: <https://www.lua.org/about.html>. Acesso em: 13 mar. 2018a.

LUA. **A Linguagem de Programação Lua**. Disponível em: <https://www.lua.org/portugues.html>. Acesso em: 6 mar. 2018b.

LUIJTEN, H. **Tweaking4All.com - How to measure temperature with your Arduino and a DS18B20Tweaking4All.com**, 22 mar. 2014. Disponível em: <https://www.tweaking4all.com/hardware/arduino/arduino-ds18b20-temperature-sensor/>. Acesso em: 6 mar. 2018

MARCELLI, M. *et al.* Design and Application of New Low-Cost Instruments for Marine Environmental Research. **Sensors (Basel, Switzerland)**, v. 14, n. 12, p. 23348–23364, 5 dez. 2014.

MASSERINI JR., R. T. et al. A coastal surface seawater analyzer for nitrogenous nutrient mapping. **Continental Shelf Research**, v. 150, p. 48–56, 1 nov. 2017.

MAXIM INTEGRATED. **DS18B20 - Programmable Resolution 1-Wire Digital Thermometer**, 6 mar. 2018. Disponível em: <https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf>. Acesso em: 6 mar. 2018

MCKEE, D.; HAM, A. C.; JONES, K. An integrated submersible fluorometer/nephelometer/transmissometer: design and testing at sea. **Optics & Laser Technology**, v. 29, n. 1, p. 35–39, 1 fev. 1997.

MICROPYTHON. **Quick reference for the ESP8266 — MicroPython 1.9 documentation**. Disponível em: <http://docs.micropython.org/en/v1.9/esp8266/esp8266/quickref.html>. Acesso em: 6 mar. 2018.

MIRANDA, E. A. *et al.* **Um estudo da aplicação de Enterprise Information Systems Patterns à modelagem de sistemas de automação**. XIV Simpósio de Excelência em Gestão e Tecnologia. **Anais**...Resende, RJ: out. 2017Disponível em: <https://www.aedb.br/seget/arquivos/artigos17/5025123.pdf>. Acesso em: 3 mar. 2018

MOTA, S. **Introducão à engenharia ambiental**. Rio de Janeiro: ABES, 2003.

MULLER, C. C.; CYBIS, L. F.; RAYA-RODRIGUEZ, M. T. Validação do método de Sedgwick-Rafter para a quantificação do fitoplâncton. n. 186, p. 8, maio 2011.

MURPHY, K. *et al.* A low-cost autonomous optical sensor for water quality monitoring. **Talanta**, v. 132, p. 520–527, 15 jan. 2015.

NG, C.-L.; SENFT-GRUPP, S.; HEMOND, H. F. A multi-platform optical sensor for in situ sensing of water chemistry. **Limnology and Oceanography: Methods**, v. 10, n. 12, p. 978–990, 1 dez. 2012.

NODEMCU. **NodeMCU Documentation**. Disponível em: <https://nodemcu.readthedocs.io/en/master/>. Acesso em: 6 mar. 2018.

ODUM, E. P. *et al.* **Fundamentos de ecologia**. São Paulo: Thomson Learning, 2007.

PEREIRA, M. G.; GALVÃO, T. F. Etapas de busca e seleção de artigos em revisões sistemáticas da literatura. **Epidemiologia e Serviços de Saúde**, v. 23, n. 2, p. 369–371, jun. 2014.

PUIU, A. *et al.* Submersible Spectrofluorometer for Real-Time Sensing of Water Quality. **Sensors**, v. 15, n. 6, p. 14415–14434, 18 jun. 2015.

RANGEL, M. JOSE. **Using Micropython with NodeMCU (ESP8266)**Campos, RJ, 3 ago. 2017. Disponível em: <https://docs.google.com/presentation/d/13fgSk4bCCg4IHB8oaPsW98OREYp0HZZ8jAluIJpi3NU>. Acesso em: 6 mar. 2018

ROESLER, C. S.; BARNARD, A. H. Optical proxy for phytoplankton biomass in the absence of photophysiology: Rethinking the absorption line height. **Methods in Oceanography**, v. 7, p. 79–94, 1 set. 2013.

SKOOG, D. A.; HOLLER, F. J.; NIEMAN, T. A. **Princípios de análise instrumental.** Porto Alegre: Bookman, 2002. reimpressão, 2006.

TEIXEIRA, C. Introdução aos métodos para medir a produção primária do fitoplâncton marinho. **Boletim do Instituto Oceanográfico**, v. 22, p. 59–92, 1973.

TORRES, V. F. N.; GAMA, C. D. DA; VILLAS BOAS, R. C. **Engenharia ambiental subterrânea e aplicações**. Rio de Janeiro: CETEM/CNPq/CYTED, 2005.

WANG, J.-M.; YANG, M.-T.; CHEN, P.-L. Design and Implementation of an Intelligent Windowsill System Using Smart Handheld Device and Fuzzy Microcontroller. **Sensors**, v. 17, n. 4, p. 830, 11 abr. 2017.

APÊNDICE A:
Construção do sensor do fotofluorescência

Para a aplicação neste trabalho, o amostrador precisava possuir algumas características específicas, a citar:

1. Ser resistente à passagem de luz;
2. Possuir um adequado posicionamento e alinhamento entre emissor e receptor de luz (Figura A.1), no caso, posicionados em um ângulo de 90° entre si;
3. Ser facilmente reprodutível;

Figura 25 – Emissor e receptor de luz

Fonte: O Autor

Foi escolhido como material para a construção do invólucro, um eletroduto rígido de ½ polegada, o qual, por ser composto por um material plástico (PVC) de cor preta, atendeu bem ao primeiro requisito. Além disso, devido ao seu baixo custo e sua alta disponibilidade no mercado, acabou atendendo também o terceiro requisito.

Para completar o invólucro, foram instaladas dois caps de PVC, para tubulação hidráulica rosqueável. Essa concepção permite a desmontagem dessas partes após o processo de calibração (Figura A.2).

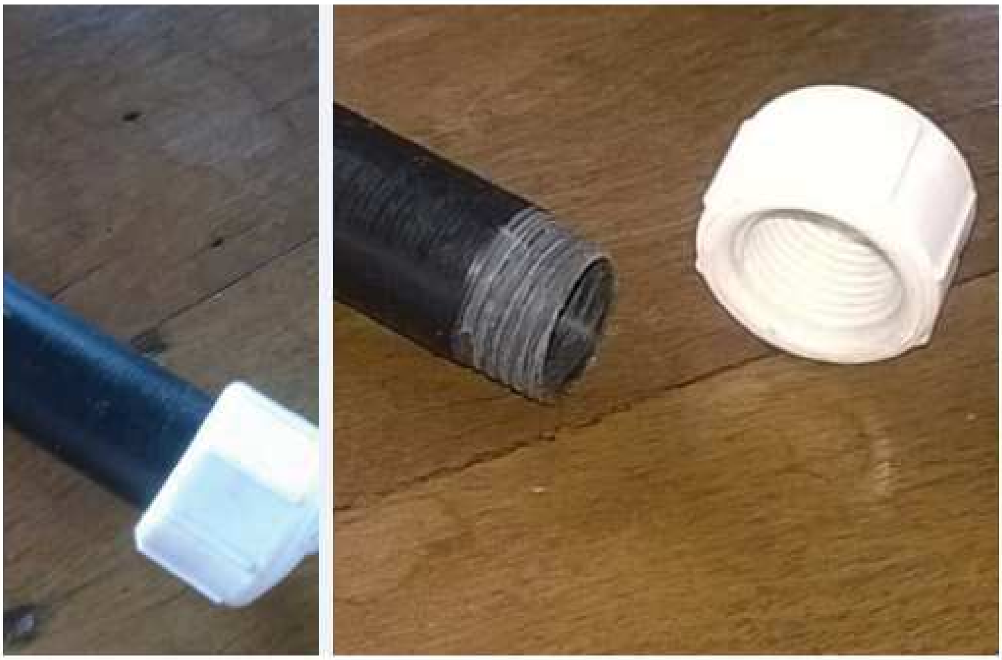


Figura 26 – Sistema desmontável do amostrador

Fonte: O Autor

O eletroduto, por vezes, já é fornecido com a adequada rosca em uma de suas extremidades. Porém, faz necessária a fabricação da rosca equivalente na outra extremidade. Essa atividade foi realizada utilizando-se uma tarraxa para tubos de PVC de dimensão compatível (Figura A.3). A fixação do tubo foi manual, utilizando-se uma lixa de granulometria 120.

Figura 27 – Abertura de rosca no amostrador

Fonte: O Autor

Para a fabricação dos furos alinhados em 90°, os quais receberam posteriormente o emissor e receptor de luz, foram testadas algumas metodologias. Na primeira tentativa, utilizou-se uma máquina fresadora industrial e um cabeçote divisor, porém, essa solução apresentou-se como inadequada, pois comprometia o objetivo de garantir a fácil reprodução do dispositivo.

A solução utilizada foi através da fixação e traçagem, mediante a utilização de elementos comuns aos ajustadores mecânicos, no caso: um bloco prismático (90°), um grampo de fixação e um graminho (Figura A.4).

Figura 28 – Elementos de traçagem para ajustagem mecânica

Fonte: O Autor

Utilizando-se esses elementos, foi possível realizar um traçado circunferencial, perfeitamente alinhado ao centro do duto (Figura A.5).

Através de uma furadeira de coluna, foram realizadas duas furações sobre a linha previamente traçada. Na primeira furação, apoiou-se o bloco prismático sobre sua base horizontal, enquanto que na segunda, virou-se o bloco em 90°, apoiando-o em sua lateral.

Figura 29 – Processo de traçagem de linha de referência e resultado da furação

Fonte: O Autor

Durante a montagem do amostrador, foram instalados o emissor de luz (LED Azul) e o receptor de luz (LDR). A frente desse último, foi montado um filtro de luz (ROSCO #19), conforme a seguinte sequência:

1. Posicionamento do LDR sobre o filtro e colagem (com cola de secagem rápida) através das laterais. Atenção especial deve ser dada para não deixar a cola penetrar entre os componentes ou sujar a parte frontal do LDR.

2. Retirada do excesso lateral do filtro;

3. Colagem (com cola de secagem rápida) do LED e do LDR, no interior dos furos, mantendo o alinhamento dos mesmos com os furos.

4. Aplicação de cola de silicone, de forma a calafetar o invólucro, evitando o vazamento de líquido pelas furações.

As Figuras A.6 e A.7 mostram, respectivamente, os passos da sequência e o amostrador montado.

Figura 30 – Sequência de montagem do emissor e receptor de luz

Fonte: O Autor

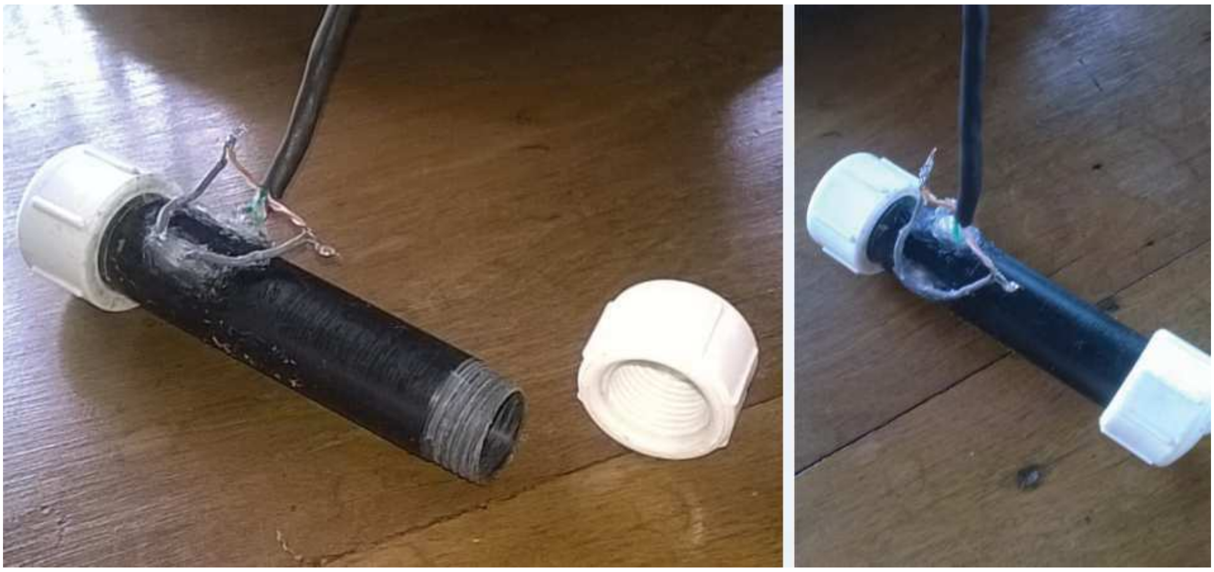


Figura 31 – Amostrador montado

Fonte: O Autor

Para concluir, foi realizado o isolamento elétrico dos fios, a pintura do conjunto com tinta preta (em spray) e a montagem de um suporte para tubo, o qual executou a função de fixador ao conjunto flutuante (Figura A.8).

Figura 32 – Conjunto pintado e montado no suporte

Fonte: O Autor

OBS: O modelo de suporte ideal é fabricado em PVC, porém, neste primeiro protótipo, foi utilizado um suporte metálico. Esta opção se deu devido a problemas de disponibilidade de mercado local, na data da construção.

Durante os eventos de calibração, um dos caps deve ser retirado para admissão da amostra, sendo posteriormente recolocado para a contenção da mesma. Quando da montagem no conjunto flutuante, ambos os caps devem ser retirados de forma a permitir a renovação da amostra, durante a campanha de monitoramento.

APÊNDICE B:
Procedimento para preparação de amostras padrão

A matéria prima para a criação das amostras é água destilada e folhas de espinafre.

A preparação da amostra acompanha os seguintes passos:

1. Separe 80 ml de água destilada em uma proveta graduada;
2. Selecione uma folha de espinafre de tamanho médio e aspecto sadio;
3. Retire a parte mais rígida da folha (talo);
4. Macere a folha com o auxílio de almofariz e pistilo;
5. Acrescente 20 ml de água destilada e misture;
6. Retire a parcela líquida da mistura e reserve em um recipiente à parte;
7. Repita os passos 4, 5 e 6 por mais duas vezes (reservando no mesmo recipiente);
8. Efetue a última maceração de forma a garantir a máxima extração possível;
9. Acrescente 10 ml de água destilada e misture;
10. Repita os passos 5 e 6;
11. Coe toda a parte líquida previamente reservada e reserve em outro recipiente;
12. Utilize os últimos 10 ml de água destilada para lavar o pistilo, o almofariz e o coador, acrescentando ao conteúdo do recipiente;
13. Coloque o extrato em uma proveta graduada e acrescente mais água destilada até obter um volume igual a 90 ml;
14. Misture para homogeneizar (vide Figura B.1)

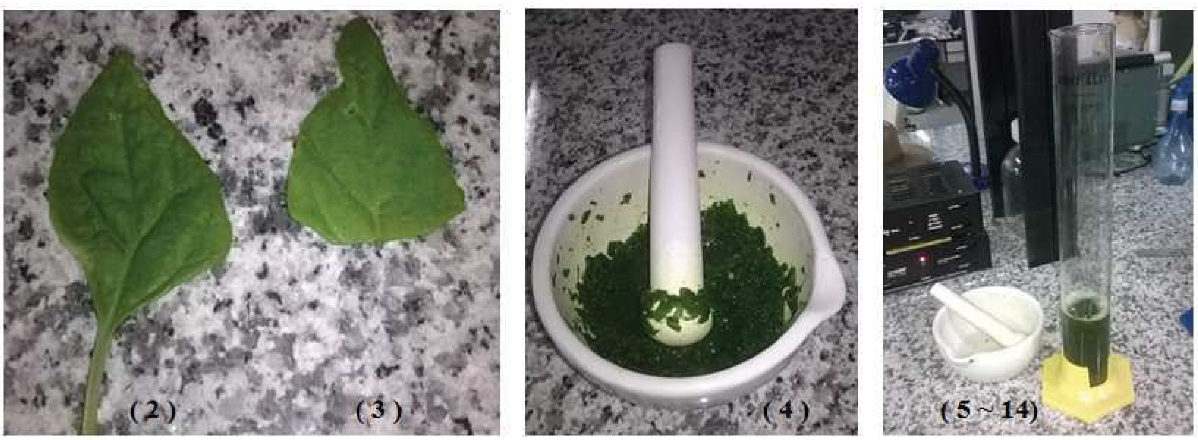


Figura 33 – Processo de maceração e separação de extrato de espinafre

Fonte: O Autor

OBS: Caso a concentração obtida seja considerada alta, acrescente mais água destilada até atingir um valor aceitável para a sua aplicação. Caso a concentração obtida seja considerada pobre, repita o procedimento, utilizando mais folhas de espinafre no passo 2. Ao final, separe 90 ml para a próxima etapa.

Os 90 ml de extrato oriundos do processo anterior será utilizado para a obtenção das amostras padrão de referência. Os próximos passos são referentes à diluição de uma parcela desta amostra para a obtenção de concentrações mais baixas (Figura B.2).

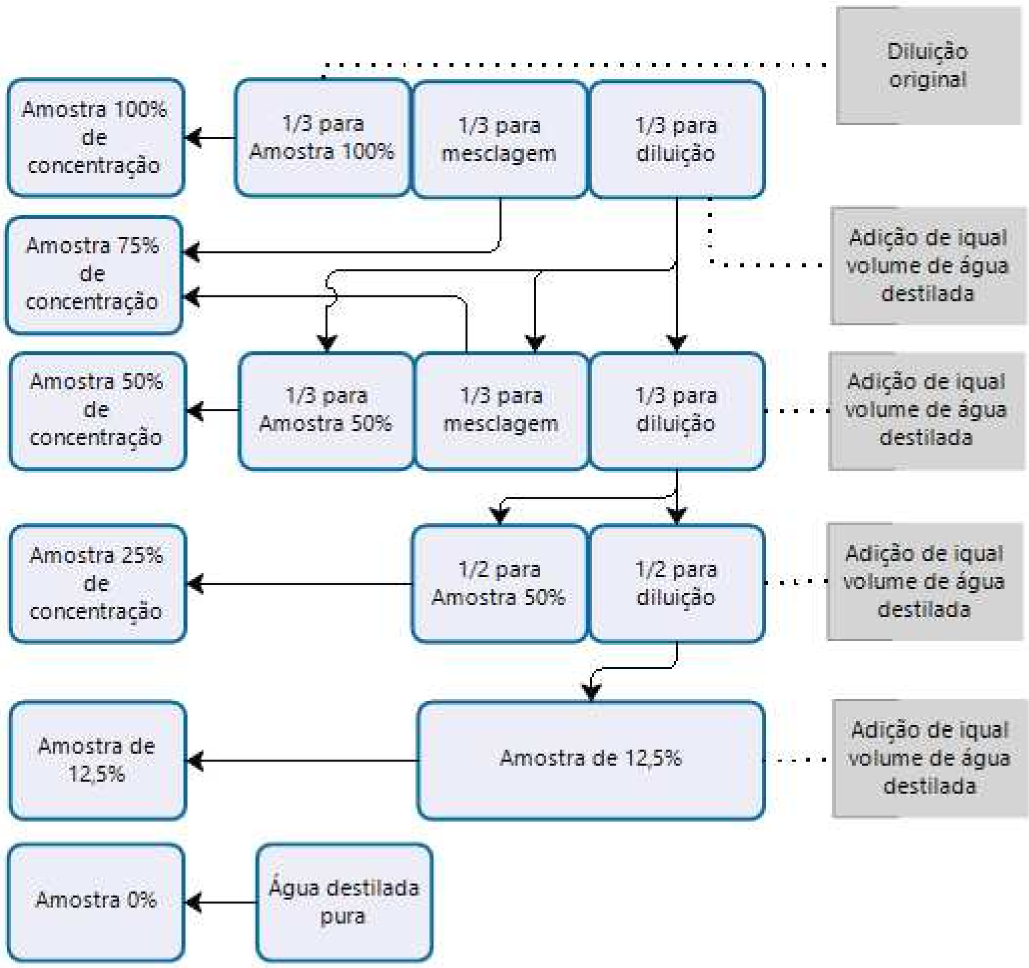


Figura 34 – Processo de diluição das amostras e obtenção de concentrações menores

Fonte: O Autor

1. Separe 30 ml como amostra com 100% da concentração;
2. Reserve 30 ml para a obtenção da amostra referente a 75%;
3. Reserve 30 ml para formar a amostra referente a 50% mediante ao acréscimo de 30 ml de água destilada;
4. Desta amostra (já diluída), separe 30 ml como referente a 50% de concentração;

5. Retire 30 ml do restante e acrescente-o aos 30 ml previamente reservados (concentração de 100%), formando assim, uma amostra de 60 ml em concentração 75%;
6. Dilua os 30 ml remanescentes, acrescentando de 30 ml de água destilada;
7. Desta amostra (diluída pela segunda vez), separe 30 ml como referente a 25% de concentração;
8. Repita o passo 6;
9. Desta amostra (diluída pela quarta vez), separe 30 ml como referente a 12,5% de concentração;
10. Reserve 30 ml de água destilada pura como amostra referente a 0% de concentração.

Ao final existirão amostras nas concentrações de 100% (integral), 75%, 50%, 25%, 12,5% e 0% as quais poderão ser utilizadas como padrão de proporcionalidade na verificação do comportamento das leituras dos instrumentos de medição.

OBS: Caso se deseje amostras mais com concentrações menores, deve-se continuar repetindo o processo de diluição para as concentrações relativas a 6,25%, 3,125%, etc.

APÊNDICE C:
Leitura de múltiplos valores analógica utilizando um único pino do microcontrolador

Por vezes, faz-se necessária a leitura de uma quantidade de valores analógicos maior do que a quantidade de entradas de um determinado microcontrolador. No caso do NodeMCU, está realidade se torna consideravelmente expressiva, pois entre os GPIO disponíveis, apenas a uma porta (A0) recebe valores analógicos.

A metodologia para sanar a citada dificuldade baseia-se na possibilidade de interligar e ler valores de mais de um sensor usando o mesmo pino de entrada analógica do microcontrolador.

Fisicamente, pode utilizar mais de um sensor interligado paralelamente a mesma porta, mediante uma rotina de multiplexação e utilização de uma GPIO discreta, para a alimentação sequenciada de cada sensor multiplexado.

A rotina deve energizar o sensor A, ler o seu valor analógico e, em seguida, desenergizar o referido sensor. Seguencialmente, este processo deve ser repido para cada sensor inteligado a referidaentrada analógica.

A Figura C.1, demostra uma montagem que exemplifica a leitura de três sensores, utilizando a porta A0 de um Micricontrolador NodeMCU.

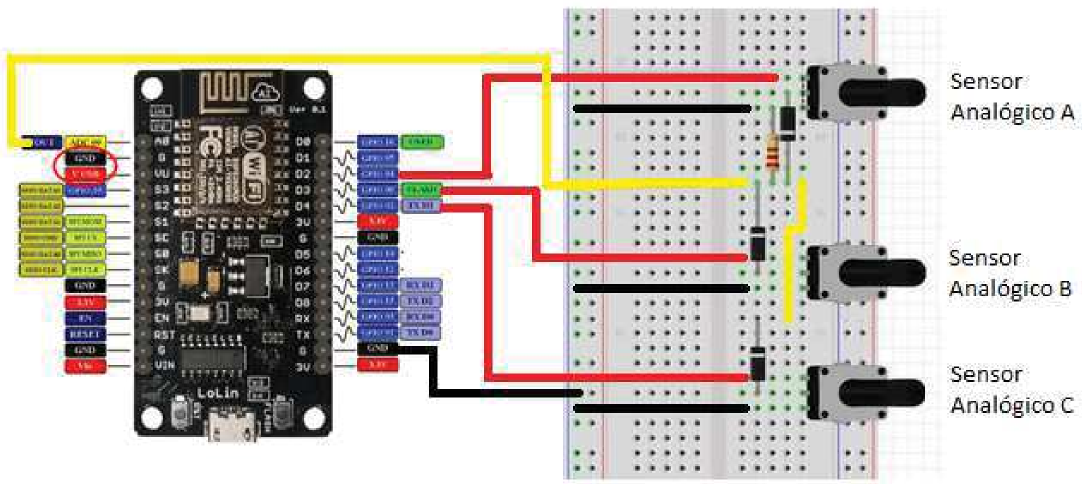


Figura 35 – Montagem para leitura múltipla em um único canal analógico

Fonte: O Autor

A motagem envolveu:

- Três sensores analógicos, que no caso, foram simulados por potenciômetros;
- Três diodos (um para cada sensor), os quais impedem o desvio de corrente pela interligação dos sensores paralelos que não estão em uso no momento;
- Um protoboard;
- Um Microcontrolador, no nosso caso, um NodeMCU;
- Fios para interligação;
- Um resistor de 10kΩ (opcional para o caso do NodeMCU).

As interligações são realizadas conforme a seguinte seguência:

- Interligue cada sensor de forma a extrair sua alimentação de uma porta discreta configurada como *output* (fios vermelhos, ligados às portas D2, D3 e D4);
- Caso o sensor necessite de interligação ao GND, execute-a normalmente;
- O terminal de saída de sinal de cada sensor deverá ser interligado em série com um diodo (de forma a permitir a pasagem de sua corrente);
- Todas as saídas dos diodos serão ligadas paralelamente entre sí, e essa junção, ligada em série com a entrada analógica do microcontrolador. Caso a aplicação utilize um Arduino, faz-se necessária a inerligação de um resistor *pull down* (10kΩ) entre o GND e o pino analógico. No caso da montagem utilizando um NodeMCU, em função da tensão de saída das portas discretas serem 3,3V, essa montagem trona-se opcional.

Considerando que a montagem utilizou potenciômetro, um dos pinos laterais deve ser insteligado ao pino de terra, enquanto o outro pino lateral deve ser interligado a uma porta discreta, a qual irá energizá-lo. O pino central será ligado ao diodo.

Os diodos são os responsáveis por isolar os circuitos dos sensores, pois os mesmos restringem a corrente a apenas uma direção, de forma impedir a interação entre os sensores em paralelo, e a garantir o correto funcionamento. Para multiplexar, apenas um sensor pode energizado por cada vez.

Considerando-se que, tanto o NodeMCU, quanto o Arduino, possuem conversores A/D de 10 bits, a resultado da tensão (entre 0 e 3,3V no NodeMCU ou 5V no Arduino) será digitalizado para um valor entre 0 e 1023.

O Código C.1 executa a leitura multipla, sendo aplicável tanto ao NodeMCU quanto ao Arduino. A estratégia utilizada imprime uma resistência adicional devido à resistência interna do diodo utilizado em cada sensor. Por isso, faz-se necessária a utilização da função MAP (não implementada nesta versão), a qual permite adequar a faixa de resposta resultante à faixa real. Assim, faz-se necessária uma calibração de cada sensor mediante sua comparação a um padrão de referência.

Código 19 – Leitura analógica múltipla em canal único.

```
/*
Multiple Analog inputs on one Analoge pin adapted from
 http://www.instructables.com/id/Multiple-Analog-Inputs-on-Only-One-Analoge-Pin/
  Show input of three sensors on one analog pin.
  The vaues of all sensors will be displayed in the serial monitor.
The circuit:
  * center pin of all Potentiometer/Sensors attached via a diode to analog input A0
  * one side pin (either one)of all potentiometers/sensors to ground
  * the pin of other side of all potentiometers/sensors to one discrete port (like a power source) of the MCU
  * pull-down resitor (10k) between analog pin A0 and ground (Only for Arduino)

About the origin version:
 Created by Vincent Verheggen
 based on:
  *example sketch analog input by David Cuartielles
  *http://www.arduino.cc/en/Tutorial/AnalogInput
  *http://www.instructables.com/id/ESP8266-with-Multiple-Analog-Sensors/
*/

//Mapping digital ports for making compatible with Arduino architecture
#define D0   16
#define D1   5
#define D2   4
#define D3   0
#define D4   2
#define D5   14
#define D6   12
#define D7   13
#define D8   15
#define D9   3
#define D10  1

int sensorPin = A0;   // select the input pin for the potentiometer
int enable1 = D2;     // enable reading sensor A
int enable2 = D3;     // enable reading sensor B
int enable3 = D4;     // enable reading sensor C

int sensorValue1 = 0;  // variable to store the value coming from sensor A
int sensorValue2 = 0;  // variable to store the value coming from sensor B
int sensorValue3 = 0;  // variable to store the value coming from sensor C

void setup() {
 Serial.begin(9600);
 // declare the enable and ledPin as an OUTPUT:
 pinMode(enable1, OUTPUT);
 pinMode(enable2, OUTPUT);
 pinMode(enable3, OUTPUT);
}
```

```
void loop() {
 // read the value from sensor A:
 digitalWrite(enable1, HIGH);
 delay(300);
 sensorValue1 = analogRead(sensorPin);
 Serial.println(sensorValue1);
 digitalWrite(enable1, LOW);
 delay(100);

 // read the value from sensor B:
 digitalWrite(enable2, HIGH);
 delay(300);
 sensorValue2 = analogRead(sensorPin);
 Serial.println(sensorValue2);
 digitalWrite(enable2, LOW);
 delay(100);

 // read the value from sensor C:
 digitalWrite(enable3, HIGH);
 sensorValue3 = analogRead(sensorPin);
 delay(300);
 Serial.println(sensorValue3);
 digitalWrite(enable3, LOW);
 Serial.println("--------------------------------------");
 delay(1000);
}
```

APÊNDICE D:
Utilização e reconhecimento de endereço de sensores DS18B20

A aplicação desenvolvida utilizou cinco unidades do sensor DS18B20 e um resistor 4,7 omhs, interligados a porta D4 (GPIO 2) de um microcontralodor NodeMCU. Na etapa apresesntada aqui, foi utilizado um cabo microUSB para alimentação elétrica e comunicação.

O reconhecimento dos endereços dos sensores DS18B20 foi realizado utilizando-se o programa ReconheceDS18B20 (Código D.1), o qual foi adaptado da referência citada no mesmo.

Código 20 – Detecção e reconhecimento de sensores DS18B20.

```
// Este programa procura pelos sensores no circuito e mostra o valor
// do endereço físico de cada sensor no Serial Monitor
// Adaptado do original --> Programa: Scan DS18B20, disponível em:
// http://www.arduinoecia.com.br/2013/04/sensor-de-temperatura-ds18b20-arduino.html

#include <OneWire.h>

// Conecte o pino central dos sensores ao Microcontrolador atravez do pino identificado abaixo
OneWire  ds(2);  //GPIO 2 --> D4 no NodeMCU

void setup(void)
{
 Serial.begin(9600);
 discoverOneWireDevices();
}

void discoverOneWireDevices(void)
{
 byte i;
 byte present = 0;
 byte data[12];
 byte addr[8];

 Serial.print("Procurando dispositivos DS18B20...\n\r");

 while(ds.search(addr))
 {
  Serial.print("\n\rEncontrado sensor \'DS18B20\' com endereco:\n\r");
  for( i = 0; i < 8; i++)
  {
   Serial.print("0x");
    if (addr[i] < 16)
    {
     Serial.print('0');

    }
   Serial.print(addr[i], HEX);

    if (i < 7)
    {
```

```
    Serial.print(", ");
   }

  }

  if ( OneWire::crc8( addr, 7) != addr[7])
  {
     Serial.print("CRC nao e valido!\n");
     return;
  }
 }
 Serial.print("\n\r\n\rFinal da verificacao.\r\n");
 ds.reset_search();
 return;
}

void loop(void)
{
 // Loop Vazio
}
```

A utilização do código segue o seguinte procedimento:

1. Interlique o todos os sensores DS18B20 na porta física indicada no código, neste caso específico, D4 (GPIO 2);
2. Efetue o carregamento do programa na placa NodeMCU;
3. Abra o monitor serial;
4. Aperte o botão de reset na placa;
5. Anote os endereços informados no monitor serial;

Cabe resaltar que este código também funciona na família arduino, bastando interligar os sensores em uma porta digital e associar o programa a essa porta.

Os sensores do tipo DS18B20 utilizam as bibliotecas de software OneWire.h e DallasTemperature.h. Nas aplicações desenvolvidas nessa pesquisa, foram criadas instâncias de objetos destas bibliotecas, configurado o atributo DeviceAddress (utilizando os endereços descobertos pelo procedimento anterior) e utilizando as seguintes funcionalidades dos objetos:

1. search( ): descobre e armazena o endereço do(s) sensor(es);
2. OneWire::crc8( ): Verifica a intergridade da mensagem enviada no barramento OneWire;
3. reset_search( ): desfaz o armazenamento realizado pela função search();
4. begin( ): Inicializa instâncias do tipo DallasTemperature;
5. setResolution( ): Configura a resolução de cada sensor;
6. requestTemperatures( ): Requisita o envio dos valores de temperatura do(s) sensor(es) via rede OneWire;
7. getTempC( ): Obtem o valor da temperatura medida em cada sensor da mensagem da rede;

APÊNDICE E:
Registros adicionais e evidências dos testes de campo

Os testes de campo foram realizados em dois eventos distintos.

O primeiro evento foi realizado na Lagoa Limpa de Travessão (21°37'27.8"S 41°21'51.4"W), em 25/01/2019, entre 09h00min e 12h20min. A mesma faz parte da Bacia Hidrográfica do Rio Paraíba do Sul (Figuras E.1), nas proximidades do rio Muriaé (Figura E.2).

Figura 36 – Bacia Hidrográfica do Rio Paraíba do Sul

Fonte: Adaptado de (AGEVAP, 2018).

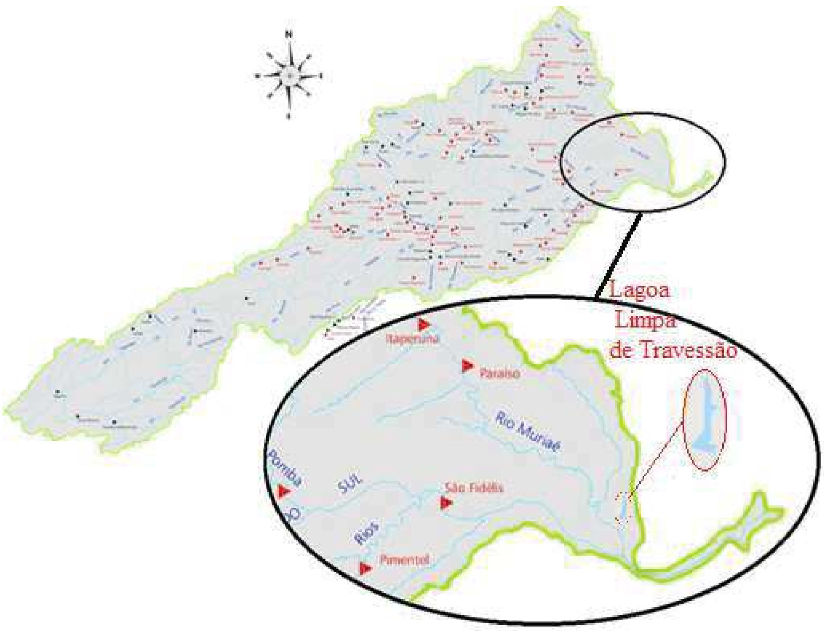


Figura 37 – Proximidades do Rio Muriaé

Fonte: Adaptado de (AGEVAP, 2018).

A referida lagoa encontra-se na área rural, sendo circundada por uma grande área utilizada para a agricultura, com predominância da cultura da cana-de-açúcar. O principal fator de degradação está relacionado ao uso de fertilizantes agrícolas. As principais utilizações são para a pesca, a recreação, e para a própria agricultura, podendo, no entanto, apresentar potencial de uso para fins de abastecimento. As figuras de E.3 a E.7 ilustram algumas atividades desse primeiro evento.

Figura 38 – Visão (esquerda e direita) do ambiente de teste nas coordenadas citadas

Fonte: O Autor

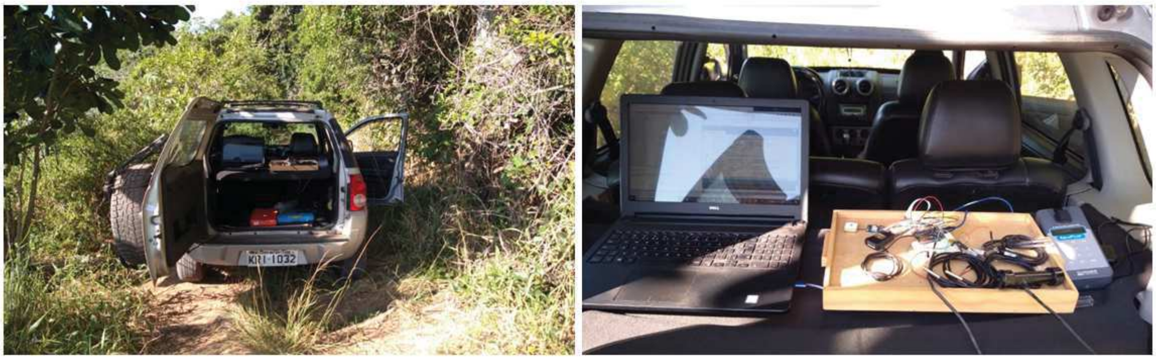


Figura 39 – Estrutura utilizada para e verificação de calibração

Fonte: O Autor

Figura 40 – Ajustes finais do sistema e início das medições

Fonte: O Autor

Figura 41 – Resgate do sistema flutuante em modo deriva

Fonte: O Autor

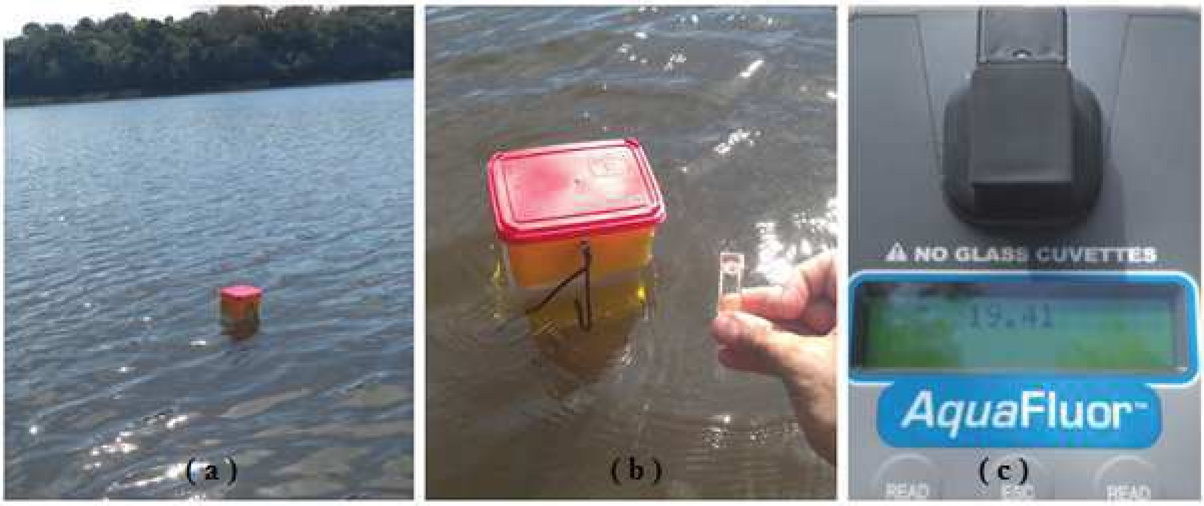


Figura 42 – Campanha de medição: (a) Sistema em deriva, (b) Sistema fundeado e recolhimento de amostra no local, (c) Medição da amostra com fluorímetro comercial

Fonte: O Autor

O segundo evento de teste de campo foi realizado em um tanque de criação de peixes (21°35’35.7”S 41°18’58.7”W), em 01/02/2019, entre 18h50min e 20h20min, na localidade de Travessão (Campos dos Goytacazes – RJ). Neste evento, toda a campana foi realizada com a unidade flutuante em modo fundeado. A figura E.8 ilustra algumas atividades do segundo evento de campo.

Figura 43 – Posicionamento do sistema flutuante em modo fundeio

Fonte: O Autor

APÊNDICE F:
Código de programa para o NodeMCU (versão completa)

Código 21 – Programa completo para o NodeMCU

```
/*****************************************************************************
   This program estimate the phytoplankton bulk and measure the temperature
 on multiple depths.  It uses SoftwareSerial to communicate with the GPS module
 on pins D2 and D3, and communicates over SPI to log that data to a SD card.
   It uses the TinyGPS++ library to parse the NMEA strings sent by the GPS
 module,  and records the interesting information - comma separated - in a
 newly created file on the SD card.
   At end it export the relevant data for Blynk App

 Resources:
  SD Library      (Built-in)
  SPI Library     (Built-in)
  OneWire Library (Built-in)
  DallasTemperatureLibrary(Built-in)
  TinyGPS++ Library - https://github.com/mikalhart/TinyGPSPlus/releases
  SoftwareSerial Library (Built-in)
  ESP8266WiFi      (Built-in after installation package_esp8266com_index.json)
  BlynkSimpleEsp8266 - https://github.com/blynkkk/blynk-library/releases

*****************************************************************************/

//Mapping the digital ports for compliance to Arduino Architecture
#define D0    16
#define D1    5         //will be used for blue LED command
#define D2    4         //will be used like RX pin for GPS module communication
#define D3    0         //will be used like TX pin for GPS module communication
#define D4    2         //will be used like bus for OneWire communication with DS18B20 sensors
#define D5    14        //will be used like SCK pin for the SPI communication with SD Card module
#define D6    12        //will be used like MISO pin for the SPI communication with SD Card module
#define D7    13        //will be used like MOSI pin for the SPI communication with SD Card module
#define D8    15        //will be used like chipselect pin for SD Card module
#define D9    3
#define D10   1

String logString;                    // String for data handling to SDCard

// Definitions for SDCard
#include <SPI.h>
#include <SD.h>                      // SD.h for ESP8266
#define SD_CHIPSELECT D8

// Definitions for GPS Module
#include <TinyGPS++.h>
#include <SoftwareSerial.h>
const int RXPin = D2, TXPin = D3;
TinyGPSPlus tinyGPS;                 // tinyGPSPlus object to be used throughout
SoftwareSerial GPS(RXPin, TXPin); // Create a SoftwareSerial
String logString2;                   // String for GPS data to Blynk App

// Definitions for DS18B20
```

```
#include <OneWire.h>
#include <DallasTemperature.h>
#define ONE_WIRE_BUS D4
OneWire oneWire(ONE_WIRE_BUS);              // Setup a oneWire instance to communicate with any
                                            // OneWire devices
DallasTemperature sensors(&oneWire);        // Pass our oneWire reference to Dallas Temperature
//***  Here you must insert the addresses of all DS18B20 ***
DeviceAddress Thermometer01 = { 0x28, 0xFF, 0x9D, 0x27, 0x62, 0x16, 0x03, 0xAB }; //for 1st DS18B20
DeviceAddress Thermometer02 = { 0x28, 0xA8, 0x5B, 0xB5, 0x04, 0x00, 0x00, 0x88 }; //for 2nd DS18B20
DeviceAddress Thermometer03 = { 0x28, 0xFF, 0xA1, 0xC7, 0x60, 0x17, 0x05, 0x11 }; //for 3rd DS18B20
DeviceAddress Thermometer04 = { 0x28, 0xFF, 0x7D, 0x24, 0x30, 0x17, 0x04, 0x8A }; //for 4th DS18B20
DeviceAddress Thermometer05 = { 0x28, 0xFF, 0xF9, 0x34, 0x60, 0x17, 0x05, 0x43 }; //for 5th DS18B20
String Temperature0, Temperature1, Temperature2, Temperature3, Temperature4;
String logString3;                          // String for Temperature data to Blynk App

//Definitions for Procedure updateFileName()
#define LOG_FILE_PREFIX "SAEGvD"            // Name of the log file.
#define MAX_LOG_FILES 100                   // Number of log files that can be made
#define LOG_FILE_SUFFIX "csv"               // Suffix of the log file
char logFileName[13];                       // Char string to store the log file name

// Definitions for Procedure printHeader()
#define COLUMN_COUNT 13                     // Setup the number of columns for the CSV file
char * log_col_names[COLUMN_COUNT] = {
  "longitude", "latitude", "velocity", "satellites", "date", "time", "temp_0 (Celsius)",
  "temp_1 (Celsius)","temp_2 (Celsius)","temp_3 (Celsius)","temp_4 (Celsius)", "LDR", "Volts"
};  // Sets the columns title. Then, log_col_names will be printed at the top of the file

//Definitions for Procedure getPhyto()
#define BLUE_LED D1
#define LDR_PIN A0
const int SAMPLE_NUMBER = 20;               //for calculate the average
const int INTERVAL = 100;                   //interval between sampling (miliseconds)
const int DIGITAL_MIN = 0;                  //lowest value used in sensor calibration
const int DIGITAL_MAX = 1023;               //highest value used in sensor calibration
double voltage;
int read_LDR;

//Definitions for Procedure logData()
const int OFF_SET = -3;                     // Factor for hour adjust from Greenwich time

//Definitions for Procedure loop()
#define LOG_RATE 300000                     // Log every 5 minutes
unsigned long lastLog = 0;                  // Global var to keep of last time we logged

//Definitions for Blynk App use
#define BLYNK_PRINT Serial
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
//Bellow insert your authorization token, which was gave
//on project creation in Blynk App.
char auth[] = "9b3e6dfab86d41a1a48e4f02576b901e"; // Insert your token
char ssid[] = "SuaRede";                          // Insert your network name
char pass[] = "SuaSenha";                         // Insert your password
BlynkTimer timer;                                 // For Blynk App refresh

///////////////////////////////////////////////////////////////
```

```

void updateFileName(){                                  // It will be used in beginSDcard() procedure
  //Create a new name for the log file for each run of the setup() procedure
  Serial.println("trying updateFileName.");
  int i = 0;
  for (; i < MAX_LOG_FILES; i++){
    memset(logFileName, 0, strlen(logFileName));         // Clear logFileName string
    // Set logFileName to "SAEGvCXX.csv":
    sprintf(logFileName, "%s%d.%s", LOG_FILE_PREFIX, i, LOG_FILE_SUFFIX);
    if (!SD.exists(logFileName)){                        // If a file doesn't exist
      break;                                             // Break out of this loop. We found our index
    }
    else{
      Serial.print(logFileName);
      Serial.println(" exists");                         // Print a debug statement
    }
  }
  Serial.print("Current file name: ");
  Serial.println(logFileName);                           // Print the file name for debug
}
/////////////////////////////////////////////////////////////////

void printHeader(){                              // It will be used in beginSDcard() procedure
                                                 //prints each column names to the top of our log file
  Serial.println("\nTrying printHeader.");
  File logFile = SD.open(logFileName, FILE_WRITE);// Open the log file
  if (logFile){                                  // If the log file opened, print our column names to the file
    int i = 0;
    for (; i < COLUMN_COUNT; i++){
      logFile.print(log_col_names[i]);
      if (i < COLUMN_COUNT - 1){                 // If it's anything but the last column
        logFile.print(',');                      // print a comma
      }
      else {                                     // If it's the last column
        logFile.println();                       // print a new line
      }
    }
    logFile.close();                             // close the file
  }
}
//////////////////////////////////////////////////////////////

void beginSDcard(){
  Serial.println("\n Setting up SD card.");
  // see if the card is present and can be initialized:
  if (!SD.begin(SD_CHIPSELECT)){
    Serial.println("Error initializing SD card.");
  }
  Serial.println(" SD card initialized.");
  updateFileName();                          // Each time we start, create a new file, increment the number
  printHeader();                             // Print a header at the top of the new file
}
///////////////////////////////////////////////////////////////

void getGPS_Data(){
  int rightHour, rightDay;
  logString2 =  String(tinyGPS.location.lng(), 6) + ", " + String(tinyGPS.location.lat(), 6) + ", ";
```

```
  logString2 += String(tinyGPS.tinyGPS.speed.kmph(), 1)+ ", " + String(tinyGPS.satellites.value()) + ", ";
  if (int(tinyGPS.time.hour()) <= (OFF_SET * (-1))) {
   rightHour = (int(tinyGPS.time.hour()) + 24 + OFF_SET);
   rightDay = (int(tinyGPS.date.day()) - 1);
  }
  else {
   rightHour = (int(tinyGPS.time.hour()) + OFF_SET) ;
   rightDay = int(tinyGPS.date.day());
  }
  logString2 += String(rightDay) + "/" + String(tinyGPS.date.month()) + "/" ;
  logString2 += String(tinyGPS.date.year()) + ", " +  String(rightHour) + ":";
  logString2 += String(tinyGPS.time.minute()) + ":" + String(tinyGPS.time.second())+ ", ";
  logString = logString2;
}
/////////////////////////////////////////////////////////////

void setDS18B20(){
  sensors.begin();                                     // Start up the DallasTemperature library
  sensors.setResolution(Thermometer01, 10);   // set the DS18b20 resolution to 10 bit for each one
  sensors.setResolution(Thermometer02, 10);
  sensors.setResolution(Thermometer03, 10);
  sensors.setResolution(Thermometer04, 10);
  sensors.setResolution(Thermometer05, 10);
}

void getDS18B20(){
  sensors.requestTemperatures();
  Temperature0 = String(sensors.getTempC(Thermometer01), 2);
  logString += Temperature0 + ", ";
  logString3 = "Temp0 = " + Temperature0 + " Celcius, \n";

  Temperature1 = String(sensors.getTempC(Thermometer02), 2);
  logString += Temperature1  + ", ";
  logString3 += "Temp1 = " + Temperature1 + " Celcius, \n";

  Temperature2 = String(sensors.getTempC(Thermometer03), 2);
  logString += Temperature2  + ", ";
  logString3 += "Temp2 = " + Temperature2 + " Celcius, \n";

  Temperature3 = String(sensors.getTempC(Thermometer04), 2);
  logString += Temperature3  + ", ";
  logString3 += "Temp3 = " + Temperature3 + " Celcius, \n";

  Temperature4 = String(sensors.getTempC(Thermometer05), 2);
  logString += Temperature4  + ", ";
  logString3 += "Temp4 = " + Temperature4 + " Celcius, \n, ";
}
/////////////////////////////////////////////////////////////

void setPhyto(){
  pinMode(LDR_PIN, INPUT);
  pinMode(BLUE_LED, OUTPUT);
}

void getPhyto() {
  Serial.println("Iniciando rotina de leitura do LDR");
   String phytoString = " ";                           //string just for debugging on serial monitor
```

```
    voltage = 0.0;
    read_LDR = 0;

    for (int i = 1; i <= SAMPLE_NUMBER / 2; i++){            //measurement before the excitation
     int read_LDR_I = map(analogRead(LDR_PIN),DIGITAL_MIN,DIGITAL_MAX,0,1023);
     double voltage_I = read_LDR_I * (3.3 / 1023);
     phytoString = String(i) + " --> Leitura digitalizada = " + String(read_LDR_I);
     phytoString += ", Leitura Analogica = " + String(voltage_I) + "Volts;";
     Serial.println(phytoString);
    }

    Serial.println(“   ”);
    digitalWrite(BLUE_LED, HIGH);
    delay(60000);                                            // 60 segundos para aquecimento do led

    for (int i = 1; i <= SAMPLE_NUMBER; i++){                //measurement during the excitation
     int read_LDR_I = map(analogRead(LDR_PIN),DIGITAL_MIN,DIGITAL_MAX,0,1023);
     double voltage_I = read_LDR_I * (3.3 / 1023);
     phytoString = String(i) + " --> Leitura digitalizada = " + String(read_LDR_I);
     phytoString += ", Leitura Analogica = " + String(voltage_I) + "Volts;";
     Serial.println(phytoString);
     voltage += voltage_I;                                   //accumulating for average
     read_LDR += read_LDR_I ;                                //accumulating for average
     delay(INTERVAL);
    }
    voltage = (voltage/SAMPLE_NUMBER);                       //average of voltage
    read_LDR = int(read_LDR/SAMPLE_NUMBER);                  //average of read LDR
    phytoString = "Leitura digitalizada média = " + String(read_LDR);
    phytoString +=  ", Leitura Analogica Media = " + String(voltage) + "Volts;";
    Serial.println(phytoString + "\n");

    digitalWrite(BLUE_LED, LOW);
    logString += String(read_LDR) + ", " + String(voltage, 2) + ", \n";

    for (int i = 1; i <= SAMPLE_NUMBER / 2; i++){            //measurement after the excitation
     int read_LDR_I = map(analogRead(LDR_PIN),DIGITAL_MIN,DIGITAL_MAX,0,1023);
     double voltage_I = read_LDR_I * (3.3 / 1023);
     phytoString = String(i) + " --> Leitura digitalizada = " + String(read_LDR_I);
     phytoString += ", Leitura Analogica = " + String(voltage_I) + "Volts;";
     Serial.println(phytoString);
    }
  }

  ////////////////////////////////////////////////////////////

  byte logData(){
   Serial.println("Trying logData.");
   File logFile = SD.open(logFileName, FILE_WRITE);          // Open the log file
   logString = " ";
   if (logFile){
    getGPS_Data();
    getDS18B20();
    getPhyto();
    Serial.print(logString);
    logFile.print(logString);
    logFile.close();
    return 1;                                                // Return success
```

```
  }
  return 0;                                         // If we failed to open the file, return fail
}
//////////////////////////////////////////////////////////////

void callBlynk(){
  Blynk.virtualWrite(V0,"Longitude,  Latitude,   Alt, Sats");
  Blynk.virtualWrite(V0,logString2);                //Data from GPS module
  Blynk.virtualWrite(V0,logString3);                //Temperature from BS18B20
  Blynk.virtualWrite(V1,String(voltage,2));         //Phytoplankton from 0 to 3.3 V
  Blynk.virtualWrite(V2,Temperature0);              //Temperature on surface
  Blynk.virtualWrite(V3,Temperature1);              //Temperature on 1st level
  Blynk.virtualWrite(V4,Temperature2);              //Temperature on 2nd level
  Blynk.virtualWrite(V5,Temperature3);              //Temperature on 3rd level
  Blynk.virtualWrite(V6,Temperature4);              //Temperature on 4th level
  Blynk.virtualWrite(V7,read_LDR);                  //Phytoplankton from 0 to 1023
}

//////////////////////////////////////////////////////////////

void setup(){
  Serial.begin(115200);                             // Sets serial monitor speed and start
  GPS.begin(9600);                                  // Sets serial GPS speed and start
  setDS18B20();                                     // Sets Up DS18B20 Sensors
  setPhyto();                                       // Sets Up for the procedure getPhyto
  beginSDcard();                                    // Tests the SD card and initiales the file for log
                                                    // Here will be called the updateFileName() and
                                                    // printHeader() procedures
  Serial.println("\n *** This is the sdCard_GPS_Phyto_06_rA Program ***");
  Blynk.begin(auth, ssid, pass);                    // Sets the initial Blynk App parameters
  timer.setInterval(LOG_RATE, callBlynk);           // Sets the interval for Blynk App refresh
}

void loop(){
  int a = millis();
  if ((lastLog + LOG_RATE) <= a){                // If it's been LOG_RATE milliseconds since the last log
    if (tinyGPS.location.isUpdated()){           // If the GPS data is vaild
      if (logData()){                            // Log the GPS data
        Serial.println("GPS logged.");           // Prints a debug message
        lastLog = millis();                      // Updates the lastLog variable
      }
      else {                             // If we failed to log GPS, print an error, don't update lastLog
        Serial.println("Failed to log new data.");
      }
    }
    else {                               // If GPS data isn't valid, print a debug message
      Serial.print("No GPS data. Sats: ");
      Serial.println(tinyGPS.satellites.value());
    }
  }
  // If we're not logging, continue to "feed" the tinyGPS object:
  while (GPS.available()){
    tinyGPS.encode(GPS.read());
  }
  Blynk.run();
  timer.run();                           // Initiates BlynkTimer
}
```