Universidade Federal de Uberlândia
Faculdade de Engenharia Elétrica
Bacharelado em Engenharia Elétrica

VICTOR BORGES FELICÍSSIMO

DESENVOLVIMENTO DE UM SISTEMA DE MONITORAMENTO DE ÁREA DE CUSTO REDUZIDO EMPREGANDO O MICROCONTROLADOR ESPRESSIF ESP32 E O MÓDULO GLOBAL POSITIONING SYSTEM NEO-M8N

Uberlândia
2026

Universidade Federal de Uberlândia
Faculdade de Engenharia Elétrica
Bacharelado em Engenharia Elétrica

VICTOR BORGES FELICÍSSIMO

DESENVOLVIMENTO DE UM SISTEMA DE MONITORAMENTO DE ÁREA DE CUSTO REDUZIDO EMPREGANDO O MICROCONTROLADOR ESPRESSIF ESP32 E O MÓDULO GLOBAL POSITIONING SYSTEM

Trabalho de Conclusão de Curso apresentada ao Programa de Graduação em Engenharia Elétrica da Universidade Federal de Uberlândia como parte dos requisitos para obtenção do título de Graduação em Engenharia Elétrica.

Orientador: Dr. Ernane Antônio Alves Coelho

Uberlândia
2026

VICTOR BORGES FELICÍSSIMO

DESENVOLVIMENTO DE UM SISTEMA DE MONITORAMENTO DE ÁREA DE CUSTO REDUZIDO EMPREGANDO O MICROCONTROLADOR ESPRESSIF ESP32 E O MÓDULO GLOBAL POSITIONING SYSTEM

Trabalho de Conclusão de Curso apresentado ao Programa de Graduação em Engenharia Elétrica da Universidade Federal de Uberlândia como parte dos requisitos para obtenção do título de Graduação em Engenharia Elétrica.

Uberlândia, 12 de junho de 2026

BANCA EXAMINADORA

Prof.: Dra. Gabriela Vieira Lima

Prof.: Dr. Gustavo Brito de Lima

Uberlândia

2026

Dedico este trabalho aos meus pais, Irenilda Borges e Wesley Felicíssimo, que trabalharam e se empenharam muito para que eu pudesse chegar até aqui.

AGRADECIMENTOS

Agradeço, primeiramente, a Deus por me conceder saúde e força para seguir em frente todos os dias, e por nunca ter abandonado a mim e aos meus pais após um período desafiador de pandemia que perdurou durante dois anos.

À minha mãe, Irenilda Borges, por ter me criado com amor e educação, ensinando-me a ser respeitoso e a tratar a todos com igualdade, cumprimentando cada pessoa que encontro em meu caminho, sempre sem distinções e com um sorriso no rosto.

Ao meu pai, Wesley Felicíssimo, por me ensinar o valor do trabalho e a compreender que a vida não é tão simples, mostrando-me a importância de valorizar o meu próprio suor, assim como ele sempre valorizou o dele para proporcionar conforto e momentos de lazer para mim e à minha mãe.

Ao meu professor, Dr. Ernane Alves Coelho, por aceitar o meu pedido de orientação neste trabalho e por me auxiliar em todos os momentos em que precisei de atenção e apoio, estando sempre presente durante o desenvolvimento deste projeto.

Aos professores Gabriela Vieira Lima e Gustavo Brito de Lima, por aceitarem participar da banca examinadora e por suas valiosas contribuições, que enriqueceram e aprimoraram o resultado final deste trabalho.

Aos meus amigos, Mateus Flausino Conceição e Otávio Sousa Siqueira, por não permitirem que eu desistisse dos estudos durante o período de pandemia e diante das dificuldades enfrentadas na faculdade, além de compartilharem comigo momentos de lazer e alegria, entre um açaí, um espetinho, partidas de videogame e chamadas de voz.

À minha namorada, Lara Wyne, por estar ao meu lado nas horas mais difíceis e por me dar o impulso fundamental para concluir esta etapa da minha vida, ajudando-me a gerenciar as responsabilidades sem que eu me sentisse sobrecarregado.

“Não há nada nobre em ser superior ao seu semelhante. A verdadeira nobreza é ser superior ao seu antigo eu.”

-W.L. Sheldon

RESUMO

O presente Trabalho de Conclusão de Curso (TCC) propõe o desenvolvimento de um sistema embarcado de baixo custo para a medição de áreas e o monitoramento de desempenho em implementos agrícolas. O projeto utiliza o microcontrolador ESP32 e um módulo receptor GPS para registrar a área coberta, a distância percorrida e as horas trabalhadas, oferecendo uma alternativa acessível às soluções comerciais de alto custo. A arquitetura fundamenta-se na conectividade nativa Wi-Fi e Bluetooth do ESP32, permitindo a comunicação sem fio em regiões rurais remotas, desprovidas de rede móvel, e viabilizando o resgate dessas informações por meio de uma interface web dedicada. Motivada pela dificuldade em aferir a produtividade individual em regimes cooperativos, a solução visa mitigar a remuneração desigual ao fornecer registros precisos dos trajetos e das atividades operacionais. Como resultado, o sistema viabiliza análises precisas e tomadas de decisão fundamentadas em dados diretamente no campo, promovendo a democratização e a modernização das práticas agrícolas para pequenos produtores e prestadores de serviços autônomos.

Palavras-chave: ESP32; GPS; Monitoramento Agrícola; NEO-M8N; Interface Web.

ABSTRACT

This Undergraduate Thesis proposes the development of a low-cost embedded system for area measurement and performance monitoring in agricultural implements. The project utilizes the ESP32 microcontroller and a GPS receiver module to record the covered area, distance traveled, and hours worked, offering an accessible alternative to high-cost commercial solutions. The architecture is based on the native Wi-Fi and Bluetooth connectivity of the ESP32, allowing wireless communication in remote rural areas—devoid of mobile networks—and enabling the retrieval of this information through a dedicated web interface. Motivated by the difficulty in assessing individual productivity in cooperative arrangements, the solution aims to mitigate unequal remuneration by providing accurate records of routes and operational activities. As a result, the system enables accurate analyses and data-driven decision-making directly in the field, promoting the democratization and modernization of agricultural practices for small-scale farmers and autonomous service providers.

Keywords: ESP32; GPS; Agricultural Monitoring; NEO-M8N; Web Interface.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AP	*Access Point* (Ponto de Acesso)
CSV	*Comma-Separated Values* (Valores Separados por Vírgula)
ESP32	Microcontrolador da Espressif
GLONASS	*Global'naya Navigatsionnaya Sputnikovaya Sistema* (Sistema Global de Navegação por Satélite)
GNSS	*Global Navigation Satellite System* (Sistema Global de Navegação por Satélite)
GPIO	*General Purpose Input/Output* (Entrada e Saída de Propósito Geral)
GPS	*Global Positioning System* (Sistema de Posicionamento Global)
HDOP	*Horizontal Dilution of Precision* (Diluição Horizontal da Precisão)
HTML	*HyperText Markup Language* (Linguagem de Marcação de Hipertexto)
HTTP	*HyperText Transfer Protocol* (Protocolo de Transferência de Hipertexto)
IBGE	Instituto Brasileiro de Geografia e Estatística
IDE	*Integrated Development Environment* (Ambiente de Desenvolvimento Integrado)
IoT	*Internet of Things* (Internet das Coisas)
LNA	*Low Noise Amplifier* (Amplificador de Baixo Ruído)
MAPA	Ministério da Agricultura e Pecuária
NMEA	*National Marine Electronics Association* (Associação Nacional de Eletrônica Marinha)
ONU	Organização das Nações Unidas
PIB	Produto Interno Bruto
RF	Radiofrequência
RMSE	*Root Mean Square Error* (Erro Quadrático Médio)

RTK	*Real Time Kinematic* (Cinemático em Tempo Real)
SIG	Sistemas de Informações Geográficas
SPE	Secretaria de Política Econômica
SPIFFS	*SPI Flash File System* (Sistema de Arquivos em Memória Flash SPI)
TTFF	*Time To First Fix* (Tempo para a Primeira Fixação)
UART	*Universal Asynchronous Receiver-Transmitter* (Transmissor-Receptor Assíncrono Universal)
UFU	Universidade Federal de Uberlândia
UHF	*Ultra High Frequency* (Frequência Ultra Alta)
UTC	*Coordinated Universal Time* (Tempo Universal Coordenado)
VHF	*Very High Frequency* (Frequência Muito Alta)
Wi-Fi	*Wireless Fidelity* (Fidelidade Sem Fio)

Sumário

1 INTRODUÇÃO

De acordo com o Ministério da Agricultura e Pecuária (MAPA) do Governo Federal, o setor agropecuário apresentou crescimento de 12,2% no primeiro trimestre de 2025 em comparação ao último trimestre de 2024, impulsionando o avanço da economia nacional e consolidando-se como o principal destaque do Produto Interno Bruto (PIB). Nesse mesmo período, a economia brasileira registrou crescimento de 1,4% em relação ao último trimestre de 2024 e de 2,9% em relação ao primeiro trimestre do mesmo ano, segundo dados do Instituto Brasileiro de Geografia e Estatística (IBGE, 2025). Ainda, segundo a Secretaria de Política Econômica (SPE) do Ministério da Fazenda, a expectativa é de que o setor agropecuário represente cerca de 6,3% do PIB em 2025, mesmo diante de um cenário de juros elevados, consumo das famílias em patamares altos e pressões inflacionárias que dificultam o cumprimento de metas fiscais.

Esse desempenho evidencia a relevância da agricultura no contexto nacional e internacional, visto que o acesso à alimentação adequada é uma das principais preocupações globais. A Organização das Nações Unidas (ONU) para a Alimentação e a Agricultura (FAO, 2024) destaca que a agricultura é uma das indústrias mais significativas do mundo, não apenas pela sua função essencial de garantir a segurança alimentar, mas também por seu papel estratégico no comércio internacional. Embora quase todos os países produzam alimentos, alguns se consolidam como grandes exportadores líquidos, como é o caso do Brasil, que ocupa posição de destaque nas exportações globais de grãos e carnes, fortalecendo sua economia e atraindo investimentos para a ampliação da produção.

Nesse cenário, observa-se que o aumento da produtividade agrícola depende não apenas da expansão de áreas cultiváveis, mas também da redução de perdas ao longo da cadeia produtiva, causadas por fatores como manejo inadequado, limitações de monitoramento e baixa adoção de tecnologias. Uma das estratégias fundamentais para mitigar essas perdas é o aperfeiçoamento dos maquinários, visando minimizar perdas de grãos durante a colheita, por exemplo, especialmente na plataforma de corte e em outros componentes da colheitadeira. Além disso, o setor tem investido constantemente em inovações como a modernização de implementos, o desenvolvimento de fertilizantes mais eficientes e a utilização de sementes

geneticamente melhoradas, fatores que contribuem para elevar a eficiência e sustentar o crescimento do agronegócio.

Nesse contexto, a agricultura de precisão surge como uma alternativa estratégica, sendo compreendida como um conjunto de técnicas que permitem o gerenciamento localizado dos cultivos, otimizando a utilização de insumos e aumentando a eficiência produtiva. Entre as principais ferramentas utilizadas pela agricultura de precisão destacam-se o Sistema Global de Navegação por Satélite (GNSS), os Sistemas de Informações Geográficas (SIG), pilotos automáticos e máquinas capazes de realizar aplicações de insumos em taxas variáveis, reduzindo custos e impactos ambientais (MOLIN *et al*., 2015). Nesse contexto, tais ferramentas possibilitam a otimização das rotas de operação no campo, contribuindo para a redução do consumo de combustível e do tempo de trabalho, além de diminuir o desperdício de insumos e promover uma utilização mais eficiente dos recursos naturais (EMBRAPA, 2018). Como exemplo dessas tecnologias, a Figura 1 apresenta a interface do monitor agrícola Gen 4 Universal 4240 da John Deere, no qual é possível visualizar informações relevantes para a operação em campo, como o mapa da área a ser percorrida de forma automática, a região já trabalhada, a velocidade de operação, entre outros parâmetros operacionais. Além dessas informações exibidas na tela principal, o monitor disponibiliza outras páginas que apresentam dados adicionais relacionados ao desempenho da máquina e ao acompanhamento das atividades agrícolas.

Figura 1 - Monitor Gen 4 Universal 4240

Fonte: (JOHN DEERE, 2025)

Como mencionado, o monitoramento de áreas, a demarcação de fronteiras e as aplicações em agricultura de precisão são realizados por meio do GNSS, que se refere a qualquer constelação de satélites que possibilita o posicionamento em tempo real e a navegação. Com isso, vale salientar que o sistema mais utilizado no mundo ainda é o NAVSTAR-GPS, desenvolvido e controlado pelo Departamento de Defesa dos Estados Unidos, que opera com 31 satélites. Porém, além dele, destacam-se outros sistemas globais como o GLONASS (Rússia), o GALILEU (União Europeia) e o BEIDOU (China), que também operam regionalmente.

Contudo, o posicionamento GNSS padrão oferece uma precisão na ordem de metros, o que se mostra insuficiente para aplicações agrícolas que demandam maior exatidão. A acurácia do sinal é determinada pelo tempo de propagação da luz entre o satélite e o receptor, podendo ser degradada por interferências atmosféricas e obstáculos no percurso. Para mitigar essas limitações, surgiram os serviços de correção, como o Cinemático em Tempo Real (RTK) e o Estático Rápido (ER), que fornecem precisão centimétrica, e em alguns casos, até milimétrica, com alta confiabilidade, mesmo diante de interferências físicas. O principal problema, entretanto, é o elevado custo desses sistemas comerciais de correção alinhados ao

GNSS, o que restringe sua adoção por pequenos e médios produtores, bem como por prestadores de serviços autônomos (PIRES *et al*., 2004).

1.1 Motivação

Diante desse cenário de dispositivos para agricultura de precisão que possuem um elevado custo para pequenos produtores e autônomos, surge como alternativa o desenvolvimento de soluções de custo reduzido, baseando-se na utilização de microcontroladores modernos e módulos de navegação acessíveis. É o caso do microcontrolador ESP32 da Espressif, que se destaca por sua boa capacidade de processamento, e recursos intrínsecos de conectividade Wi-Fi e Bluetooth, além do baixo custo. Tais características o tornam ideal para projetos que exigem coleta, armazenamento e transmissão de dados sem fio. Associado a esse microcontrolador, existe no mercado o módulo de navegação NEO-M8N, que se apresenta como uma excelente opção, permitindo a recepção simultânea de até três sistemas GNSS (como GPS, Galileo e GLONASS), o que garante um posicionamento de boa precisão mesmo em condições de sinal fracas, além de apresentar um bom custo-benefício. Juntos, esses componentes são capazes de atender às demandas de monitoramento de áreas agrícolas com custo significativamente reduzido.

A integração desses dois componentes possibilita o desenvolvimento de um sistema de monitoramento de áreas de baixo custo. Assim, este trabalho propõe o desenvolvimento de um sistema de monitoramento de áreas agrícolas de baixo custo, cujo núcleo é a integração do microcontrolador ESP32 com o módulo GPS NEO-M8N. Essa combinação possibilita o cálculo de áreas em hectares e a disponibilização dos dados em tempo real por meio de uma interface web acessível via Wi-Fi, facilitando o acompanhamento e o registro das operações agrícolas. Com isso, a pesquisa busca, então, demonstrar que soluções acessíveis podem democratizar o uso de tecnologias de georreferenciamento, facilitando o acesso à modernização das práticas agrícolas em propriedades de todos os tamanhos, principalmente para pequenos produtores, e prestadores de serviços autônomos.

1.2 – Objetivos

1.2.1 – Objetivo Geral

Este trabalho objetiva o desenvolvimento de um sistema para medição de área (em hectares), distância percorrida e tempo de operação, direcionado a pequenos produtores e operadores de maquinários agrícolas. A pesquisa justifica-se pela necessidade de mensurar a produtividade individual em regimes cooperativos, visando mitigar a remuneração desigual e a ineficiência operacional decorrente de divisões igualitárias de rendimento. Complementarmente, a solução busca otimizar o manejo agrícola e promover a sustentabilidade, auxiliando no controle de custos mediante a redução do consumo de combustível, do desperdício de insumos e do tempo ocioso de máquina.

1.2.2 – Objetivo Específico

Para atingir o objetivo principal de desenvolver um GPS agrícola de custo reduzido, os objetivos específicos incluem:

- pesquisar e fundamentar teoricamente conceitos de georreferenciamento e estudar sobre o posicionamento por satélite;
- avaliar diferentes modelos de módulos GPS disponíveis no mercado, identificando seus pontos fortes, limitações e melhor compatibilidade com o microcontrolador ESP32;
- desenvolver códigos em C++ para extração, armazenamento e envio dos dados coletados pelo módulo GPS;
- desenvolver filtros para redução de ruídos característicos no GPS;
- desenvolver uma interface web (Web Server) embarcada no ESP32 para visualização dos dados de posicionamento, área calculada, tempo de operação e *status* do sistema;

- realizar testes e simulações com o modelo proposto, visando sua verificação, validação e calibração;
- implementar e validar métodos para o cálculo de área percorrida, distância e tempo de trabalho em operações agrícolas.

Ao final, espera-se consolidar uma boa fundamentação teórica e prática sobre o uso de sistemas GPS para monitoramento agrícola, além de obter um modelo funcional e acessível que possa ser utilizado por operadores de máquinas agrícolas, pequenos produtores e prestadores de serviço no campo, contribuindo para otimização da produtividade e transparência na medição de áreas trabalhadas.

1.3 – Estrutura do Trabalho

A estrutura deste trabalho foi organizada em cinco capítulos, conforme descritos a seguir:

1. Introdução (este capítulo);
2. Fundamentação Teórica;
3. Metodologia Utilizada;
4. Resultados;
5. Conclusões.

O Capítulo 1 – Introdução tem como objetivo delimitar e apresentar o objeto de estudo, justificar sua relevância e contextualizar ao leitor o projeto a ser desenvolvido. Além disso, busca situá-lo quanto à organização do texto, fornecendo uma visão geral do conteúdo abordado nos capítulos subsequentes.

O Capítulo 2 – Fundamentação Teórica busca validar o projeto ao apresentar pesquisas e estudos prévios, consistindo na seleção e organização de teorias, conceitos e trabalhos que dão suporte ao desenvolvimento acadêmico. No caso do presente trabalho, serão abordados temas como: conceitos de georreferenciamento e GPS; tecnologias de sensoriamento aplicadas à agricultura de precisão; estudos que utilizaram o ESP32 em sistemas embarcados; definições relacionadas à agricultura

de precisão, medição de área e produtividade agrícola; além da comparação com tecnologias comerciais já existentes.

O Capítulo 3 – Metodologia Utilizada descreve em detalhes os procedimentos adotados para a obtenção dos resultados, incluindo o desenvolvimento do protótipo proposto, cujo objetivo é desempenhar funções semelhantes às de um GPS agrícola comercial, fornecendo métricas de área, tempo e outros indicadores de desempenho.

O Capítulo 4 – Resultados apresenta, de forma organizada, os dados obtidos a partir da aplicação da metodologia descrita no capítulo anterior. À medida que os resultados são expostos, são discutidas suas implicações e ressaltadas conclusões parciais relacionadas ao objeto de estudo.

Por fim, o Capítulo 5 – Conclusões realiza o fechamento do trabalho, sintetizando os principais achados e evidenciando as contribuições da pesquisa, tanto em termos acadêmicos quanto práticos, destacando os benefícios potenciais para a sociedade.

2 FUNDAMENTAÇÃO TEÓRICA

A agricultura de precisão tem se tornado cada vez mais presente no meio rural em razão de sua capacidade de aumentar a produtividade agrícola e otimizar o uso dos recursos naturais. Essa abordagem está presente tanto em implementos modernos quanto pode ser adaptada a equipamentos mais antigos, possibilitando ganhos adicionais no desenvolvimento das culturas. Trata-se de uma técnica estratégica que busca gerenciar de forma localizada os cultivos, permitindo a aplicação de insumos de maneira inteligente, dosando-os com precisão nos pontos de maior necessidade. Com isso, promove-se a redução de custos e, consequentemente, a diminuição dos impactos ambientais (MOLIN *et al*, 2015).

Nesse contexto, o uso de tecnologias de georreferenciamento assume papel central, destacando-se o Sistema Global de Navegação por Satélite, responsável por viabilizar o posicionamento geoespacial e a coleta de dados em tempo real. Essa tecnologia é aplicada em diversas etapas da produção agrícola, como no mapeamento de áreas, no monitoramento de implementos e no controle de operações mecanizadas, desde a fase de plantio até a colheita. Entretanto, a acurácia do posicionamento pode ser comprometida por fatores atmosféricos e obstáculos físicos. Para superar essas limitações, empregam-se métodos de correção diferencial, como o RTK, capazes de garantir precisão centimétrica. Contudo, tais recursos elevam consideravelmente o custo de aquisição, restringindo seu acesso a pequenos produtores (EMBRAPA, 2018).

Diante desse cenário, o desenvolvimento de soluções acessíveis baseadas em microcontroladores e módulos GPS de baixo custo se apresenta como alternativa viável. O ESP32, desenvolvido pela Espressif, destaca-se pela combinação de baixo custo, conectividade Wi-Fi e Bluetooth integradas, além de boa capacidade de processamento, o que o torna ideal para aplicações embarcadas em ambientes rurais (ESPRESSIF SYSTEMS, 2022). Associado a ele, o módulo Ublox NEO-M8N oferece recepção simultânea de múltiplos sistemas GNSS, garantindo precisão satisfatória a um custo reduzido (UBLOX, 2020).

Assim, a fundamentação teórica deste trabalho se baseia na articulação entre os conceitos de agricultura de precisão, sistemas de posicionamento por satélite e

tecnologias embarcadas de baixo custo, que fornecem sustentação científica e técnica para o desenvolvimento do sistema proposto.

2.1 História do Sistema Global de Navegação por Satélite

Em 1947, após a Segunda Guerra Mundial, teve início o período histórico denominado Guerra Fria, marcado por um conflito político e ideológico entre duas potências da época: os Estados Unidos da América, de orientação capitalista, e a ex-União Soviética, de orientação socialista. Esse confronto perdurou até 1991 e caracterizou-se pela polarização mundial em dois grandes blocos, cada um alinhado aos interesses políticos de uma dessas nações. Embora o risco de um confronto militar direto fosse uma ameaça constante, a Guerra Fria foi caracterizada por disputas indiretas que envolveram espionagem, corrida armamentista (chegando ao ápice das armas nucleares), avanços tecnológicos e a busca pela supremacia geopolítica, resultando em significativos impactos no desenvolvimento global.

Diante do cenário dessa guerra ideológica, destacou-se também a corrida espacial, que representou uma demonstração de poder político, militar, econômico, tecnológico e científico. As duas grandes potências investiram tempo e dinheiro em pesquisas aeroespaciais a fim de demonstrar quem seria o primeiro a explorar o espaço. Com isso, em 4 de outubro de 1957, dez anos após o início da guerra, houve um anúncio soviético sobre o lançamento do satélite Sputnik I, se tornando o primeiro satélite artificial do mundo, que acabou por impactar o Estados Unidos ao superar seu satélite Vanguard que estava em desenvolvimento, e perdendo a liderança tecnológica e posição de maior potência do mundo. Menos de um mês depois, a União Soviética lançou *o* Sputnik II tripulado pela cadela Laika, demonstrando seus esforços para melhorar sua tecnologia espacial e fragilizando os EUA. Nesse contexto, essa rivalidade acelerou ainda mais o desenvolvimento tecnológico de comunicação, rastreamento, e navegação por satélite.

Foi justamente o rastreamento do Sputnik I que deu origem ao conceito fundamental por trás da navegação por satélite. Os cientistas por trás do lançamento, ao monitorar o sinal de rádio transmitido pelo satélite, perceberam que a frequência do sinal mudava constantemente devido ao Efeito Doppler. Essa mudança de

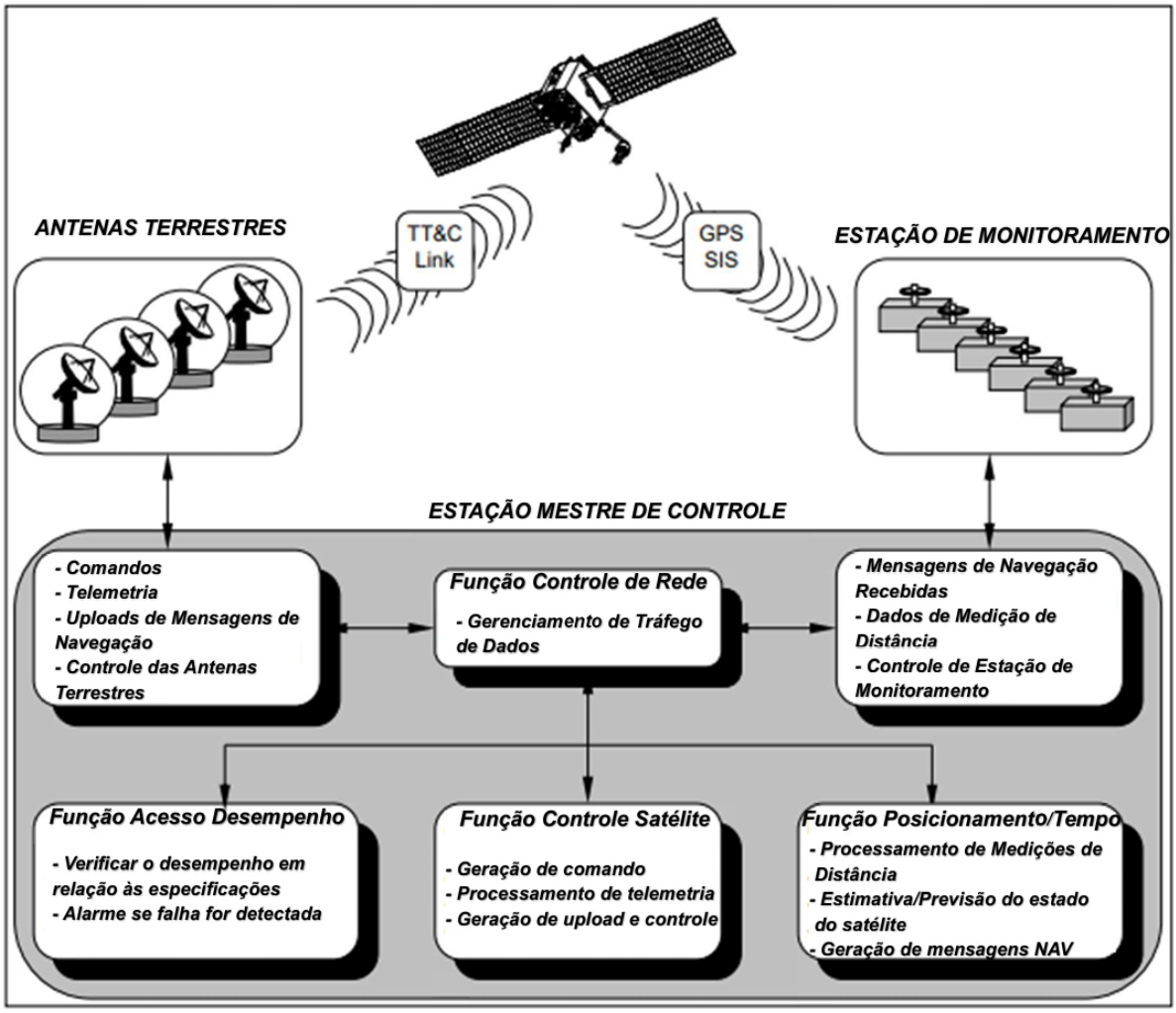
frequência permitiu-lhes mapear a órbita terrestre com precisão, pois ao receber os pulsos de rádio, eles puderam aprender sobre a densidade da atmosfera e testar métodos de rádio e ópticos de rastreamento orbital. Logo, essa descoberta demonstrou o seguinte conceito: se a posição de um observador na Terra fosse conhecida, seria possível determinar a órbita de um satélite. Invertendo essa lógica, se a órbita exata de um satélite fosse conhecida, sua posição na Terra poderia ser determinada. Essa constatação permitiu que os americanos rastreassem os pulsos de rádio do Sputnik usando o efeito Doppler, um método que eventualmente levou ao desenvolvimento do Sistema de Navegação por Satélite da Marinha, também conhecido como Transit. Sendo assim, estava lançada a base para o futuro Sistema de Posicionamento Global.

No início da década de 1970, o Departamento de Defesa (DoD) queria garantir a disponibilidade de um sistema de navegação por satélite robusto e estável. Adotando as ideias anteriores de cientistas da Marinha, o DoD decidiu usar satélites para apoiar o sistema de navegação proposto. Sendo assim, seguiram em frente com essa ideia e lançaram seu primeiro satélite, o Sistema de Navegação com Tempo e Alcance (NAVSTAR), em 1978, conhecido como Global Positioning System (GPS). Esse sistema possui, até hoje, uma constelação que inclui 24 satélites, cada um viajando em uma órbita circular de 12 horas a pouco mais de 20.000 quilômetros acima da Terra. Os satélites estão posicionados de forma que seis sejam observáveis em quase 100% do tempo de qualquer ponto da Terra. Cada um dos satélites são equipados com relógios atômicos redundantes e rastreado por uma rede de controle terrestre. Cada satélite transmite sua posição e hora em intervalos regulares, e esses sinais são interceptados por receptores GPS terrestres. Além disso, ao receber sinais de múltiplos satélites, o receptor calcula o tempo que cada sinal levou para chegar e, por meio de um processo de trilateração, determina sua localização exata na Terra em longitude e latitude. Vale salientar ainda, que esse sistema oferece dois níveis de serviço: o Standard Positioning Service (SPS), aberto ao uso civil, e o Precise Positioning Service (PPS), um sinal criptografado e mais preciso, restrito ao uso militar dos EUA e de nações aliadas.

A Figura 2 ilustra o funcionamento do sistema de comunicação entre os satélites, as estações de monitoramento em solo e os receptores GNSS. Nesse processo, os satélites transmitem continuamente sinais contendo informações de

posicionamento e tempo, que são captados pelas antenas receptoras. Esses dados são então processados, permitindo a obtenção de informações como latitude, longitude, horário e outros parâmetros de navegação, fundamentais tanto para aplicações civis quanto militares.

Figura 2 - “Global Positioning System Standard Positioning Service Performance Standard, 5th Edition, April 2020”, U.S. Government



Fonte: (Adaptado de UNITED STATES, 2020)

Enfim, de forma resumida, os receptores GNSS são capazes de detectar, decodificar e processar sinais transmitidos pelos satélites. Estes enviam códigos de alcance em diferentes portadoras de radiofrequência, permitindo que a posição seja determinada com variados graus de precisão, dependendo do receptor e do pós-processamento. Além disso, uma rede global de receptores GNSS permanentes

fornece dados que permitem estudos de alta precisão, como os movimentos das placas tectônicas, os deslocamentos associados a terremotos e a orientação da Terra. Atualmente, além do GPS (EUA) e do GLONASS (Rússia), outros sistemas globais estão em operação, como o Galileo (União Europeia) e o BeiDou (China), ampliando a cobertura e a precisão do posicionamento em escala mundial.

2.2 Radiofrequência em Sistema Global de Navegação por Satélite (GNSS)

Existem diversos serviços que operam em determinadas bandas de frequência, variando desde aplicações civis, como televisão e telefonia móvel, até aplicações militares. Cada serviço possui um espectro eletromagnético bem definido, com a finalidade de prevenir interferências entre os serviços e garantir sua máxima eficiência de uso. Esse espectro se estende aproximadamente de 30 kHz a 300 GHz, sendo internacionalmente dividido em faixas ou bandas.

Uma antena transmissora pode irradiar ondas eletromagnéticas capazes de percorrer grandes distâncias, atendendo a diferentes serviços. Os satélites, por sua vez, são sistemas mais robustos, capazes de transmitir sinais a distâncias superiores a 19.000 km da superfície terrestre. Em ambos os casos, os mecanismos de propagação das ondas variam significativamente, dependendo da frequência de operação e das condições ambientais. Nesse contexto, as ondas que se propagam pelas camadas da ionosfera são denominadas ondas ionosféricas ou celestes; aquelas que se propagam na baixa atmosfera e na troposfera recebem o nome de ondas troposféricas; já as ondas que se propagam próximas à superfície da Terra são chamadas de ondas terrestres, subdivididas em ondas diretas, ondas refletidas pelo solo e ondas de superfície.

No caso das ondas transmitidas por satélites, as chamadas ondas celestes sofrem forte influência dos fenômenos de refração e absorção, uma vez que atravessam diversas camadas atmosféricas — ionosfera, termosfera, mesosfera, estratosfera e troposfera —, cada qual contribuindo para a alteração do sinal. Dessa forma, múltiplos mecanismos de propagação podem atuar simultaneamente, sendo os mais relevantes: absorção, espalhamento, reflexão, refração, difração, múltiplo percurso, cintilação, desvanecimento e dispersão em frequência. Assim, em estudos

de propagação, é essencial identificar quais mecanismos predominam em determinado cenário e quais apresentam menor relevância (Vasconcelos, Lorenço – 2017).

De modo geral, os fatores que determinam os mecanismos presentes em uma situação específica estão relacionados às condições do meio e à frequência de operação. Em meios ionizados, como a ionosfera, deve-se considerar os efeitos de absorção e refração. Em transmissões próximas à superfície terrestre, por outro lado, a reflexão torna-se indispensável. Além disso, o comprimento de onda influencia diretamente a ocorrência de difração em obstáculos presentes ao longo do trajeto de propagação.

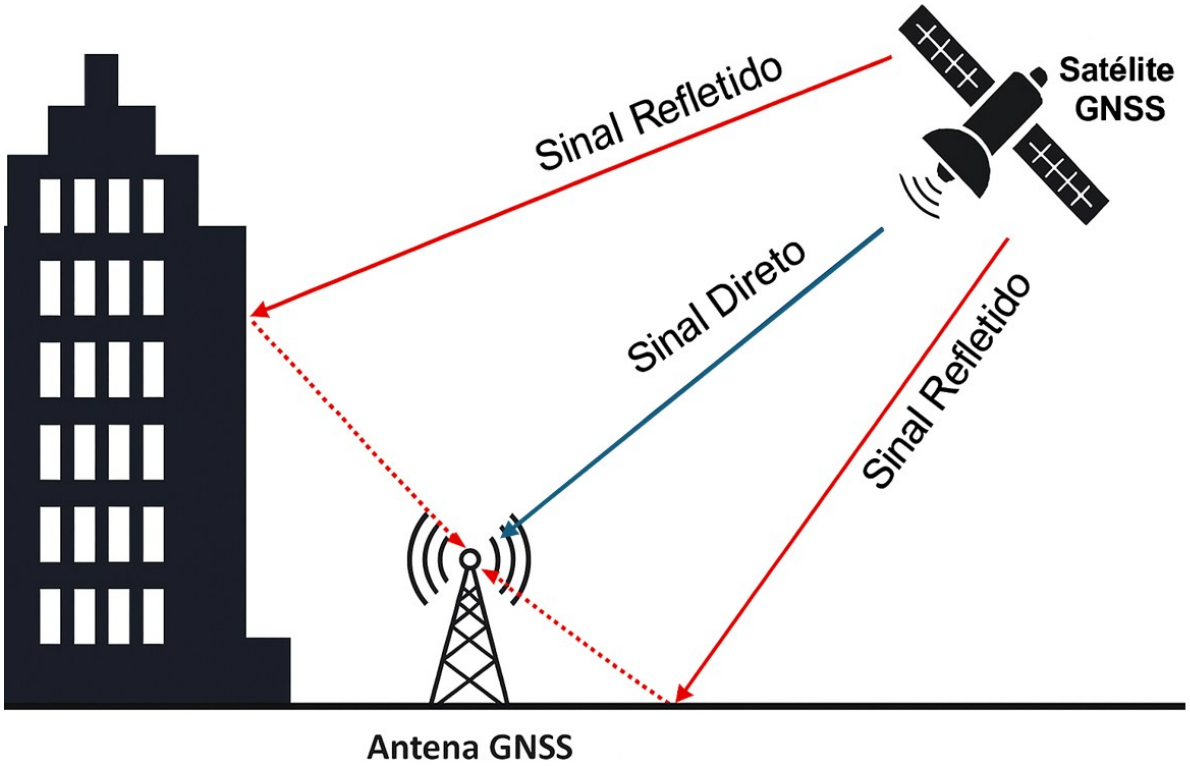
No caso do GNSS, normalmente é necessário rastrear quatro ou mais satélites para calcular as coordenadas de latitude e longitude do receptor, uma vez que os mecanismos de propagação podem comprometer a precisão do sinal. A partir das informações transmitidas por radiofrequência, aplica-se o algoritmo da trilateração para determinar a posição do receptor na superfície terrestre. No entanto, esses sinais frequentemente sofrem perturbações, como refração, absorção, reflexão e difração. Um dos fenômenos mais críticos é o multicaminho, no qual o sinal chega ao receptor por trajetórias múltiplas devido a reflexões em superfícies próximas.

A principal fonte de erro no posicionamento e navegação via GPS é a refração ionosférica, responsável por provocar o atraso do código e o avanço da fase da onda portadora. Além disso, o bloqueio ou a atenuação do sinal por obstáculos físicos — que no contexto rural incluem relevo acidentado, áreas de mata e densos dosséis vegetativos — reduz drasticamente a precisão da leitura. Para mitigar essas limitações de cobertura, sistemas modernos fundem os dados do receptor com sensores complementares, como magnetômetros e unidades de medição inercial (IMUs), assegurando orientação contínua e estimativa de trajetória mesmo sob perda temporária de sinal do satélite. Entretanto, a exigência de *hardware* adicional para essa fusão sensorial encarece a solução, representando um desafio limitante para pequenos produtores e evidenciando a necessidade do desenvolvimento de sistemas de monitoramento que priorizem a acessibilidade econômica.

A propagação por multicaminho merece atenção especial, pois ocorre quando um ou mais sinais são refletidos em diferentes superfícies antes de alcançar a antena

receptora. Essas superfícies podem estar dispostas horizontal, vertical ou inclinadamente, como ruas, edifícios e veículos. O tipo de material também influencia o comportamento do sinal, podendo gerar atenuação ou distorções distintas. Estudos demonstram, por exemplo, que receptores circundados por alumínio apresentam desempenho diferente daqueles cercados por madeira (SEEBER, 2003). A Figura 3 ilustra esse fenômeno, no qual os sinais dos satélites GPS sofrem múltiplas reflexões e processamentos antes de alcançar a antena receptora.

Figura 3 - Demonstração do Multicaminho em GNSS.



Fonte: (O autor, 2025)

2.3 Protocolo de Comunicação do Sistema Global de Navegação por Satélite (GNSS)

A comunicação entre os satélites e os receptores ocorre por meio de sinais de radiofrequência, como mencionado anteriormente. No entanto, é necessário um protocolo padronizado para a abstração e interpretação desses dados, sendo esse o NMEA, sigla para *National Marine Electronics Association* (Associação Nacional de Eletrônica Marinha). Embora amplamente utilizado no contexto do Sistema de

Posicionamento Global (GPS), o protocolo NMEA surgiu muito antes da criação do próprio GPS, tendo sido instituído em 1957 por um grupo de revendedores de equipamentos eletrônicos marítimos, com o objetivo de melhorar a comunicação e padronizar as interfaces entre fabricantes e usuários de dispositivos náuticos.

Atualmente, o NMEA é reconhecido como o formato padrão de dados adotado por praticamente todos os fabricantes de receptores GNSS, desempenhando papel semelhante ao padrão ASCII no mundo da computação digital, uma vez que ambos utilizam caracteres de texto. Cada sentença NMEA é limitada a 82 caracteres, o que garante compatibilidade entre diferentes equipamentos e *softwares*.

O principal objetivo do padrão NMEA é assegurar a interoperabilidade entre dispositivos e programas, permitindo que desenvolvedores criem aplicações compatíveis com uma ampla gama de receptores, sem a necessidade de projetar interfaces específicas para cada modelo. Sem esse padrão, o desenvolvimento e a manutenção de *softwares* de posicionamento seriam processos mais complexos, demorados e custosos.

Apesar de sua padronização, o NMEA pode gerar certa confusão para os desenvolvedores, pois não existe apenas um tipo de mensagem, mas diversas, cada qual com formato e finalidade específicos. Assim como há receptores GNSS com diferentes capacidades e constelações, também existem vários tipos de sentenças NMEA, cada uma voltada a uma função específica. Todas as sentenças do padrão possuem uma sequência de cinco caracteres, sendo que as duas primeiras letras indicam a constelação de satélites utilizada (GP para GPS, GL para GLONASS e GN para múltiplas constelações), e as três letras seguintes indicam o tipo da mensagem. Esses dados podem ser transmitidos por diferentes interfaces de comunicação, como RS-232, USB, Bluetooth, Wi-Fi, UHF, entre outras.

Estruturalmente, as sentenças NMEA iniciam-se com o caractere “$” e são constituídas por campos de comprimento variável delimitados por vírgulas. O fechamento da cadeia de dados é indicado por um asterisco (*), que antecede o *checksum* — um código de verificação de integridade gerado pela operação lógica 'ou exclusivo' (XOR) aplicada a todos os caracteres situados entre os delimitadores inicial e final (GPS WORLD, 2020).

Para uma melhor compreensão da estrutura dos dados, a Figura 4 apresenta um exemplo de mensagem NMEA gerada por um receptor GPS RTK:

Figura 4: Sentença NMEA

```
$GPGGA,181908.00,3404.7041778,N,07044.3966270,W,4,13,1.00,495.144,M,29.200,M,0.10,0000*40
```

Fonte: (O autor, 2026)

Como já dito, todas as mensagens NMEA começam com o caractere “$”, e cada campo de dados é separado por vírgulas. Sendo assim, no caso apresentado:

- GP indica que se trata de uma posição proveniente do GPS;

- 181908.00 representa o horário UTC no formato horas, minutos e segundos (18:19:08 UTC);

- 3404.7041778 indica a latitude no formato DDMM.MMMMM, em que o número de casas decimais é variável;

- N indica que a latitude pertence ao hemisfério norte;

- 07044.3966270 representa a longitude no formato DDDMM.MMMMM, também com número variável de casas decimais;

- W indica longitude oeste;

- 4 representa o indicador de qualidade da solução, sendo 1 para coordenada não corrigida, 2 para coordenada diferencialmente corrigida (exemplo: WAAS, DGPS), 4 para coordenada fixa RTK (precisão em centímetros) e 5 para coordenada RTK *float* (precisão em decímetros);

- 13 indica o número de satélites utilizados no cálculo da posição;

- 1.00 representa a diluição horizontal da precisão (HDOP);

- 495.144 corresponde à altitude da antena em relação ao nível médio do mar;

- M indica que a unidade de medida da altitude é o metro;

- 29.200 representa a separação geoidal, que deve ser subtraída da altitude da antena para se obter a altura acima do elipsoide (HAE);

- M indica novamente a unidade da separação geoidal;

- 0.10 representa a idade da correção diferencial, caso existente;

- 0000 corresponde ao identificador da estação base de correção;

- *40 corresponde aos dígitos de verificação (*checksum*).

A mensagem $GPGGA é uma das sentenças mais utilizadas do padrão NMEA, pois fornece informações fundamentais sobre o posicionamento e a qualidade do sinal recebido. Existem, contudo, outras sentenças NMEA alternativas e complementares, como RMC, GSA, GSV e VTG, que oferecem dados adicionais sobre tempo, velocidade, direção e *status* dos satélites, possibilitando uma análise mais completa e precisa do desempenho do sistema GNSS (GPS WORLD, 2020).

2.3 Aplicações do Sistema Global de Navegação por Satélite (GNSS)

As aplicações do Sistema Global de Navegação por Satélite (GNSS) baseiam-se nos sinais transmitidos pelos satélites para executar suas funções com elevada eficiência e precisão, especialmente na determinação da posição geográfica. Em essência, tais aplicações estão fortemente relacionadas ao georreferenciamento, que consiste em associar o sistema de coordenadas interno de um mapa digital ou de uma fotografia aérea a um sistema de coordenadas geográficas reais da superfície terrestre. Dessa forma, um mapa ou imagem digital georreferenciada está vinculada a um sistema de coordenadas conhecido, permitindo seu usuário identificar com precisão a localização espacial de qualquer ponto representado, o que é essencial para diversas aplicações que serão exemplificadas ao decorrer desse tópico. Além disso, a presença dessas informações espaciais no arquivo digital possibilita a realização de análises geográficas básicas, como a identificação das coordenadas de um ponto com um simples clique, o cálculo de distâncias e áreas, bem como a extração de informações complementares com praticidade e alta precisão.

Em decorrência do parágrafo anterior, é importante salientar que essas aplicações abrangem diversos setores da economia, desde a agricultura e a indústria automotiva até a aviação e a defesa, como citado em tópicos anteriores, podendo ser classificadas em cinco categorias principais:

1. Localização: determinação precisa da posição de um ponto ou objeto na superfície terrestre;
2. Navegação: definição da melhor rota entre dois locais distintos;
3. Rastreamento: monitoramento contínuo do deslocamento de um objeto em tempo real;
4. Mapeamento: levantamento e geração de mapas detalhados de áreas específicas;
5. Cronometragem: medição extremamente precisa do tempo, em escala de bilionésimos de segundo.

Se tratando da exemplificação das aplicações, no setor automotivo, o GNSS é aplicado tanto para localização e navegação quanto no desenvolvimento de veículos autônomos, como os veículos da Tesla, em que o rastreamento contínuo do ambiente é fundamental para a segurança e prevenção de colisões. Nesse contexto, a precisão buscada é em nível de faixa, o que difere da precisão centimétrica exigida em aplicações agrícolas baseado no *Real Time Kinematic*. Como os veículos circulam por diversas regiões, os serviços de correção baseados em RTK tornam-se menos eficientes, sendo preferíveis os métodos PPP (Precise Point Positioning), disponíveis globalmente.

Na agricultura, o GNSS é amplamente utilizado para otimizar operações de semeadura, fertilização e colheita. Por meio do posicionamento e rastreamento de maquinários agrícolas, é possível realizar atividades com maior eficiência e menor desperdício de insumos. Além disso, o sistema pode ser empregado para mapear propriedades rurais, contribuindo para o planejamento das rotas e o manejo adequado do solo.

A combinação entre GNSS e Sistemas de Navegação Inercial (INS), conhecida como fusão sensorial, é essencial em ambientes urbanos densos, onde os sinais de satélite podem sofrer bloqueios ou reflexões. Essa integração com sensores como IMUs, LiDAR e fotogrametria garante redundância e continuidade do posicionamento, mesmo em situações de perda temporária do sinal.

Por outro lado, a indústria de defesa prioriza a resiliência do sinal. Aplicações militares exigem soluções de posicionamento capazes de manter a precisão mesmo em ambientes contestados, sujeitos a interferências intencionais (*jamming* e *spoofing*). Para isso, são adotadas tecnologias de mitigação de interferências, redundância de sistemas e consciência situacional, que permitem identificar e reagir a sinais maliciosos que possam comprometer a navegação.

Embora cada setor apresente necessidades e requisitos distintos, os princípios fundamentais do GNSS permanecem os mesmos: determinar com precisão onde o usuário está e como ele se desloca no espaço, de forma confiável e contínua. Essas aplicações demonstram a versatilidade e a importância do GNSS como uma das tecnologias fundamentais para o avanço da automação, da precisão operacional e da conectividade global (NOVATEL, 2021).

2.4 Agricultura de Precisão

A introdução das técnicas de Agricultura de Precisão (AP) no Brasil ocorreu no final da década de 1990, impulsionada por pesquisas pioneiras desenvolvidas em universidades públicas e na Empresa Brasileira de Pesquisa Agropecuária (Embrapa). Nesse contexto, o reconhecimento da variabilidade espacial da produtividade e das propriedades químicas do solo constituiu um dos principais pilares para a consolidação da AP no país.

Nas últimas duas décadas e meia, o avanço do conhecimento científico nacional possibilitou progressos expressivos em diversas áreas correlatas, como o manejo de doenças, a robótica agrícola, o sensoriamento remoto e proximal (aéreo e orbital), além do desenvolvimento de plataformas digitais, *softwares* de gestão agrícola e ferramentas de apoio à decisão voltadas ao produtor rural e às empresas prestadoras de serviço.

Ainda, mais recentemente, o panorama da pesquisa científica em Agricultura de Precisão tem sido analisado por meio de estudos bibliométricos e análises de redes científicas em escala global, com destaque para a participação de países europeus, em especial, a Itália. Paralelamente, levantamentos internacionais foram conduzidos com o intuito de identificar os padrões de adoção da AP em diferentes regiões do mundo, incluindo o contexto brasileiro, em razão da relevância do país como um dos principais produtores agrícolas globais.

Nesse cenário, a adoção de boas práticas de manejo agrícola, como o plantio direto, a rotação de culturas e o uso de corretivos orgânicos, associada aos investimentos em tecnologias adequadas, caracteriza o sistema de Agricultura de Precisão como um modelo essencial para a intensificação agrícola sustentável. Esse modelo visa reduzir perdas, melhorar a aplicabilidade de insumos e aumentar a eficiência operacional. De forma geral, a AP pode ser definida como uma estratégia de manejo orientada pela variabilidade espacial e temporal, cujo objetivo é otimizar o uso de recursos, elevar a produtividade e a qualidade dos produtos agrícolas, ampliar a lucratividade e promover a sustentabilidade ambiental.

Essa abordagem integra um conjunto de tecnologias convergentes, que combinam sensores, sistemas de informação geográfica (SIG), sistemas de posicionamento por satélite, processamento de dados e máquinas inteligentes, permitindo o gerenciamento preciso da variabilidade espaço-temporal. Trata-se, portanto, de uma nova e contínua revolução agrícola, fundamentada na tecnologia da informação, que busca personalizar o manejo agrícola de forma coerente e holística, explorando as variações espaciais e temporais das culturas e das condições ambientais.

Como reflexo dessa evolução, a expansão global da Agricultura de Precisão impulsionou a sinergia entre os Sistemas Globais de Navegação por Satélite, o sensoriamento inteligente e a Internet das Coisas (IoT). Esse arranjo tecnológico estabeleceu um novo paradigma produtivo, tornando a gestão agrícola mais sustentável, eficiente e fundamentada na tomada de decisão baseada em dados.

A contribuição do GNSS para o surgimento e a consolidação dos princípios da AP é inegável. Inicialmente, sua importância se evidenciou pela facilidade de georreferenciar dados de campo, permitindo a coleta e o registro espacial de

informações agronômicas com coordenadas de latitude e longitude, o que possibilita a rastreabilidade e a aplicação localizada de insumos, conforme as análises de solo. Com o avanço das tecnologias de posicionamento, surgiram aplicações ainda mais relevantes, como a orientação automatizada de máquinas agrícolas para a execução de operações específicas, tais como semeadura, adubação, pulverização e colheita.

Os sistemas de direção automática chegaram ao Brasil na virada do século, marcando um avanço expressivo no uso do GNSS no setor agrícola. Associada ao GPS-RTK, essa tecnologia foi avaliada em experimentos de plantio mecanizado de cana-de-açúcar, demonstrando que a precisão do paralelismo entre as passadas foi aproximadamente cinco vezes superior à obtida por meio do direcionamento manual.

Entretanto, a ionosfera brasileira apresenta condições particularmente severas de cintilação ionosférica, fenômeno que pode provocar perda de sinal e interrupções em operações mecanizadas durante períodos de alta atividade solar. Embora existam estratégias de mitigação capazes de reduzir parcialmente os efeitos adversos, é essencial monitorar as condições ionosféricas e evitar a execução de operações agrícolas críticas em momentos de forte cintilação, como foi discutido brevemente no tópico 2.2.

Enfim, nos últimos anos, a popularização dos receptores GNSS de múltiplas constelações tem proporcionado maior redundância e precisão, especialmente em ambientes com visibilidade limitada dos sinais. Além disso, o uso combinado de diferentes sistemas, como GPS com Beidou/Compass e Galileo, aliado à modernização dos sinais de comunicação, tende a ampliar ainda mais o desempenho e a robustez da navegação por satélite em aplicações de Agricultura de Precisão, minimizando erros de posicionamento que, embora toleráveis em determinadas operações, podem ser substancialmente reduzidos (MOLIN, 2021).

2.4 Agricultura de Precisão em Maquinários Agrícolas

A indústria de maquinários agrícolas tem sido uma das principais impulsionadoras da adoção da Agricultura de Precisão. Com o avanço contínuo da tecnologia, tornou-se cada vez mais comum a incorporação de métodos e sistemas embarcados em colheitadeiras, tratores, pulverizadores autopropelidos e drones, com

o objetivo de gerenciar a variabilidade espacial da produção e otimizar a eficiência operacional, conforme evidenciado na evolução apresentada no tópico anterior.

Atualmente, as máquinas agrícolas apresentam um elevado nível de tecnologia embarcada, incluindo diversos sistemas de sensores e atuadores capazes de realizar a aplicação localizada de insumos, além de fornecerem informações valiosas e de alta resolução espacial, que contribuem para o manejo preciso e eficiente das lavouras, reduzindo desperdícios e promovendo o uso racional de recursos.

Diante do exposto nos parágrafos anteriores, destaca-se a introdução dos monitores de produtividade em colheitadeiras, que representou um marco para o estabelecimento de uma base de conhecimento sobre a variabilidade espacial das culturas agrícolas no Brasil. Globalmente, esses dispositivos surgiram no mercado no início da década de 1990, inicialmente aplicados em colheitadeiras de grãos. No contexto nacional, uma contribuição significativa foi o desenvolvimento de um sistema de pesagem contínua do produto colhido, composto por um subtanque inserido no reservatório de grãos da colheitadeira, apoiado em células de carga capazes de medir a massa das sementes, contribuindo para a definição do rendimento da colheita de grãos por hectare. O mapeamento dessa produtividade, por sua vez, proporcionou importantes avanços no gerenciamento das operações mecanizadas de colheita.

No caso da cana-de-açúcar, inicialmente não existiam ferramentas adequadas para medir a variabilidade espacial da produtividade, o que retardou a adoção de tecnologias de manejo localizado nessa cultura. No entanto, os custos associados às operações mecanizadas, que representam a maior parcela do custo de produção, impulsionaram as usinas no desenvolvimento da adoção de inovações voltadas ao aumento da eficiência operacional. Entre essas inovações destacam-se o uso de sistemas GNSS de alta precisão (RTK e correção via satélite por assinatura), tecnologias de condução assistida, sistemas de gestão de frotas e telemática, elementos que contribuíram para consolidar a adoção tecnológica nos maquinários agrícolas.

O mercado nacional de equipamentos agrícolas é amplamente representado por pulverizadores, distribuidores, semeadoras de plantio direto e colheitadeiras. Diversos controladores nacionais de taxa variável foram desenvolvidos, enquanto empresas globais adquiriram companhias do setor, promovendo a integração entre

tecnologias mecânicas nacionais e eletrônicos importados, como sistemas GNSS, controladores e módulos de distribuição de sementes. Os equipamentos mais comuns no mercado atualmente atendem às demandas de aplicação em taxa variável de calcário, fósforo, potássio, nitrogênio e sementes. O controle de seções para pulverizadores, distribuidores e semeadoras também se tornou uma prática comum, sendo que, mais recentemente, sistemas de controle de bicos e válvulas de modulação por largura de pulso (PWM) foram introduzidos no mercado, aprimorando ainda mais a precisão operacional.

Vale dizer que nos últimos anos, observa-se um aumento expressivo no uso de veículos aéreos não tripulados (VANTs), que se tornaram plataformas versáteis de sensoriamento remoto, permitindo o acoplamento de sensores (como câmeras e sistemas LiDAR) e de equipamentos complementares, como pulverizadores ou dispersores de agentes biológicos, denominados Drones. Esses estudos, ainda recentes no Brasil, têm sido conduzidos com sensores de imagem de diferentes resoluções espectrais, ampliando as possibilidades de análise.

Apesar da contínua evolução e da crescente incorporação de tecnologias no maquinário agrícola, visando ao aprimoramento do manejo, ao aumento da produtividade e à redução de desperdícios, o elevado custo de aquisição desses equipamentos ainda permanece como uma barreira significativa para grande parcela dos produtores rurais. Essa conjuntura econômica, contudo, tem favorecido o surgimento de um modelo de negócio emergente baseado na prestação de serviços especializados. Nesse contexto, empreendedores realizam investimentos em máquinas e tecnologias de alto valor para atender às demandas de terceiros, ampliando e democratizando o acesso às tecnologias agrícolas no campo.

Embora o investimento inicial seja vultoso, observa-se uma tendência natural de redução de preços motivada pela depreciação e maturação do mercado. O caso dos drones de pulverização exemplifica essa dinâmica: populares desde 2020, modelos que antes custavam R$ 200.000,00 com capacidade de 20 litros foram superados por equipamentos atuais que oferecem o dobro do desempenho pela metade do custo. Contudo, o montante ainda é impeditivo para muitos agricultores, que acabam recorrendo a métodos convencionais, como bombas de arrasto manuais, desprovidas de qualquer tecnologia embarcada.

Segundo levantamento conduzido pela Empresa Brasileira de Pesquisa Agropecuária (Embrapa) e publicado por Bolfe *et al.* (2020), 67,1% dos agricultores apontam o elevado custo de aquisição de equipamentos e aplicativos como a principal barreira para a adoção de tecnologias no campo. Esse fator supera, inclusive, desafios estruturais históricos, como a precariedade da conectividade em áreas rurais. Tal cenário alimenta a percepção de que as inovações tecnológicas são restritas a produtores com alta capacidade de investimento, agravando a exclusão de pequenos e médios agricultores, que compõem a base do setor produtivo, do acesso a ferramentas modernas de automação e manejo.

Apesar de as grandes empresas concentrarem a produção de equipamentos de alto custo, existem alternativas acessíveis que podem contribuir significativamente para o trabalho do produtor rural. Entre elas, destacam-se o uso de ferramentas digitais gratuitas, como planilhas e aplicações baseadas em inteligência artificial, além de soluções desenvolvidas por pequenas empresas e startups que buscam oferecer produtos de qualidade a preços reduzidos. Essas empresas utilizam tecnologias de baixo custo disponíveis no mercado, como plataformas baseadas em Arduino, ESP ou Raspberry Pi, para o desenvolvimento de *hardwares* capazes de desempenhar funções avançadas, como sistemas de geolocalização (GPS) e monitoramento em campo, a um custo muito inferior ao dos equipamentos comerciais. Dessa forma, tornam-se uma alternativa viável para produtores que desejam modernizar suas operações, otimizando tempo, reduzindo desperdícios e melhorando o manejo agrícola. Por exemplo, um trabalho em hidroponia desenvolveu *hardware* de baixo custo para aquisição de dados usando ESP8266-01 e sensores comuns, reduzindo significativamente custo associado à automação agrícola. Da mesma forma, artigos recentes sobre Agricultura Digital enfatizam que a democratização do IoT e da automação exige soluções acessíveis, confiáveis e duráveis, adequadas ao contexto de produtores com restrição de recursos.

Por fim, esse movimento de soluções de baixo custo se alinha também à tendência de que as tecnologias digitais, mesmo que inicialmente de alto valor, tendem a se baratear com o tempo, graças à evolução dos componentes, maior escala de produção e maior concorrência. Assim, ao combinar plataformas mais acessíveis com *softwares* livres ou de baixo custo e parcerias ou serviços locais, abre-se caminho para que médias e pequenas propriedades também introduzam a automação e o

monitoramento inteligente no manejo, reduzindo desperdícios, otimizando insumos e melhorando a eficiência operacional.

2.5 Microcontroladores e Sistemas Embarcados

Antes de abordar especificamente os microcontroladores, é importante destacar que eles se diferenciam significativamente dos microprocessadores, tanto em termos de arquitetura de *hardware* quanto de aplicação.

Um microprocessador (MPU) é um único chip semicondutor que integra as principais unidades funcionais de um computador, como a unidade aritmética e lógica, a unidade de controle e os mecanismos de comunicação com memória e dispositivos de entrada e saída. Ele atua como a Unidade Central de Processamento (CPU), sendo frequentemente descrito como o “cérebro” de um sistema computacional. No entanto, por não possuir periféricos integrados — como temporizadores, conversores analógico-digitais ou memórias dedicadas —, o microprocessador depende de componentes externos para compor um sistema funcional completo, limitando-se à execução de operações lógicas e aritméticas definidas por *software*. Ainda assim, trata-se de um componente essencial em computadores pessoais e diversos dispositivos eletrônicos, pois interpreta e executa instruções provenientes de dispositivos de entrada, como teclados, mouses, câmeras e microfones.

Ao longo das décadas, os microprocessadores evoluíram de forma significativa, tornando-se menores, mais eficientes e mais potentes, o que possibilitou sua ampla aplicação em produtos que variam de *smartphones* a eletrodomésticos. Essa evolução teve início no começo da década de 1970, com o lançamento do Intel 4004 — o primeiro microprocessador comercial —, que estabeleceu as bases para os avanços posteriores da tecnologia computacional.

Em contraste, um microcontrolador pode ser definido como um computador completo integrado em um único chip, frequentemente denominado *system-on-a-chip* (SoC). Enquanto um computador de mesa é formado por vários componentes distintos que trabalham em conjunto dentro do gabinete, o microcontrolador concentra todos os elementos essenciais de um sistema computacional em um único circuito

integrado. Assim como qualquer computador, ele possui elementos fundamentais, definidos em:

- Unidade Central de Processamento (CPU): Frequentemente chamada de "cérebro" do sistema, é o núcleo responsável por buscar, decodificar e executar as instruções do programa, além de controlar o fluxo de dados e as operações aritméticas e lógicas.
- Memória: Os microcontroladores integram dois tipos distintos de memória:
    - Memória de Programa (Não Volátil/Flash): Armazena o *firmware* (o código a ser executado), retendo as informações mesmo sem energia.
    - Memória de Dados (Volátil/RAM): Armazena variáveis e dados temporários gerados durante a execução. Ao contrário da Flash, seu conteúdo é perdido quando o sistema é desligado.
- Periféricos e Interfaces: São os módulos que permitem ao microcontrolador interagir com o mundo externo. Variam conforme a aplicação, mas geralmente incluem:
    - Entrada/Saída (E/S): Pinos GPIO, temporizadores (*timers*) e contadores.
    - Conversores de Sinal: Analógico-Digital (ADC) e Digital-Analógico (DAC).
    - Comunicação: Protocolos de interface serial (como UART, SPI, I2C), USB ou Ethernet.
    - Interfaces de Usuário: Controladores para *displays* LCD ou teclados.

Essa elevada integração torna o microcontrolador um dispositivo compacto, eficiente e especialmente projetado para executar tarefas dedicadas em sistemas embarcados, geralmente sem a necessidade de sistemas operacionais complexos. Tais características o tornam ideal para aplicações que exigem processamento em tempo real, como controle de motores, servomecanismos, aquisição de dados por sensores e comunicação entre dispositivos.

A importância dos microcontroladores está diretamente associada à crescente demanda por sistemas embarcados no mercado atual, que necessitam de sistemas computacionais compactos, de baixo custo e projetados para atender a funções específicas. Um exemplo amplamente conhecido é a plataforma Arduino, que oferece placas de desenvolvimento completas e genéricas. No entanto, muitos de seus recursos podem ser desnecessários para aplicações específicas. Nesse contexto, o uso direto de um microcontrolador permite o desenvolvimento de placas personalizadas, contendo apenas os periféricos exigidos pela aplicação, o que reduz custos, otimiza o desempenho e aumenta a eficiência do sistema.

A indústria automotiva exemplifica de forma clara essa realidade. Veículos modernos podem conter mais de 50 microcontroladores, com uma média de 20 a 25 unidades dedicadas a subsistemas como controle do motor, sistemas de freio ABS, *airbags* e diversos sensores. Além do setor automotivo, esses dispositivos estão amplamente presentes em eletrônicos de consumo, eletrodomésticos inteligentes e sistemas de comunicação (WOLF, 2012).

Diferentemente da indústria automotiva convencional, onde os sistemas embarcados priorizam a segurança viária e o entretenimento, no setor agrícola o foco recai sobre a eficiência operacional e a agricultura de precisão. Tratores, colheitadeiras e pulverizadores modernos utilizam redes complexas, padronizadas pela norma ISO 11783 (ISOBUS), para integrar a unidade de potência aos implementos agrícolas. Com essa norma integralizada aos implementos e maquinários, torna-se possível que o trator de uma marca se comunique com o implemento (semeadora) de outra. Nesse contexto, o sistema de posicionamento global (GNSS) deixa de ser apenas uma ferramenta de localização para atuar como um sensor primário de controle, permitindo funções como o piloto automático e o corte automático de seções. No entanto, o alto custo dessas soluções comerciais proprietárias cria uma barreira para pequenos produtores, abrindo uma lacuna tecnológica que pode ser preenchida por arquiteturas de baixo custo baseadas em microcontroladores de 32 bits, como o ESP32, integrados a módulos GNSS de dupla frequência

Um requisito crítico em muitas dessas aplicações é a robustez, visto que os microcontroladores devem operar sob condições extremas que computadores convencionais não suportariam. Por exemplo, o controle eletrônico de um motor de

trator pode exigir funcionamento em temperaturas que variam de –34 °C até valores superiores a 125 °C, devido ao calor gerado pelo próprio sistema.

Esses dispositivos constituem o núcleo dos sistemas embarcados, definidos como sistemas computacionais integrados a produtos maiores e projetados para executar funções dedicadas. Diferentemente de computadores de uso geral, os sistemas embarcados não são destinados à programação direta pelo usuário final, mas à execução eficiente de tarefas específicas, interagindo com o ambiente por meio de sensores e atuadores.

Embora compartilhem princípios dos sistemas computacionais tradicionais, os sistemas embarcados apresentam grande diversidade. Cada aplicação impõe requisitos particulares de desempenho, consumo energético e restrições físicas, demandando combinações específicas de *hardware* e *software*. A literatura especializada costuma classificá-los segundo critérios como tamanho, conectividade, requisitos temporais e nível de interação com o usuário.

No mercado atual, três grandes famílias de microcontroladores se destacam pela ampla utilização: PIC, da Microchip; Intel MCS, da Intel; e AVR, da Atmel, hoje Microchip (A Microchip Technology anunciou a aquisição da Atmel em janeiro de 2016). Entre elas, a família Atmel AVR tornou-se especialmente popular por servir de base às placas Arduino, amplamente difundidas entre estudantes e entusiastas devido à sua simplicidade, acessibilidade e vasta disponibilidade de material didático. Por esse motivo, essas plataformas são frequentemente recomendadas como primeiro contato com sistemas embarcados. Apesar disso, os microcontroladores não se limitam a aplicações educacionais, sendo amplamente empregados em contextos profissionais, como prototipagem, robótica, automação industrial, sistemas automotivos e soluções voltadas à Internet das Coisas (IoT).

Além dessas famílias tradicionais, o mercado atual conta com uma ampla diversidade de fabricantes, como NXP, Arm e Intel, que oferecem centenas de variantes de microcontroladores, desde modelos de uso geral para entusiastas até soluções altamente especializadas destinadas a aplicações industriais e comerciais.

2.6 Plataforma Arduino UNO e ESP32

Os microcontroladores constituem o núcleo das plataformas de prototipagem eletrônica e podem ser classificados, principalmente, de acordo com a largura do barramento de dados e a arquitetura de memória empregada. Em relação ao tamanho dos dados, os microcontroladores são tradicionalmente divididos em três categorias principais.

Os microcontroladores de 8 bits são capazes de manipular apenas 8 bits de dados por ciclo de *clock*. Embora apresentem desempenho limitado, destacam-se pelo baixo consumo de energia e pelo custo reduzido, sendo amplamente utilizados em aplicações simples e sistemas de controle básico.

Os microcontroladores de 16 bits oferecem frequências de *clock* mais elevadas, maior capacidade de memória e desempenho aproximadamente duas vezes superior ao dos dispositivos de 8 bits, atendendo a aplicações intermediárias que demandam maior capacidade de processamento.

Já os microcontroladores de 32 bits apresentam elevada velocidade de processamento e grande capacidade de cálculo, sendo indicados para aplicações mais complexas, ainda que, em geral, apresentem maior consumo energético.

Além da largura de dados, os microcontroladores também podem ser diferenciados quanto à sua arquitetura interna, a qual influencia diretamente o desempenho do sistema. Dispositivos baseados na arquitetura Von Neumann utilizam um único barramento para instruções e dados, o que limita a execução a uma operação por vez. Embora essa abordagem simplifique o projeto do *hardware*, ela impõe restrições de desempenho em aplicações mais exigentes. Por outro lado, microcontroladores baseados na arquitetura Harvard dispõem de barramentos separados para instruções e dados, permitindo acessos simultâneos e, consequentemente, maior eficiência e desempenho. Essa arquitetura é amplamente adotada em microcontroladores modernos.

Nesse contexto, destaca-se a plataforma Arduino, amplamente utilizada no desenvolvimento de projetos eletrônicos e sistemas embarcados. Trata-se de uma plataforma de código aberto que combina *hardware* e *software* de forma acessível, sendo especialmente popular em ambientes educacionais e de prototipagem rápida.

Entre suas versões mais conhecidas está o Arduino UNO e o Nano, baseados no microcontrolador ATmega328P.

O Arduino UNO possui 14 pinos de entrada e saída digitais — dos quais seis podem ser utilizados como saídas PWM —, seis entradas analógicas, um oscilador de 16 MHz, interface USB, conector de alimentação, interface ICSP e botão de reset. Esses recursos tornam a placa autossuficiente para o desenvolvimento de projetos básicos, bastando conectá-la a um computador ou a uma fonte externa de alimentação. Por ser baseado em um microcontrolador de 8 bits da família AVR, com arquitetura Harvard, o Arduino UNO é especialmente indicado para introdução à eletrônica, controle básico e aplicações educacionais.

Entretanto, o avanço das aplicações embarcadas, especialmente aquelas que demandam maior capacidade de processamento e conectividade, impulsionou o uso de soluções mais robustas. Nesse cenário, destaca-se o ESP32, desenvolvido pela empresa Espressif Systems.

O ESP32 é um microcontrolador de 32 bits, equipado com processador dual-core e projetado especialmente para aplicações de Internet das Coisas (IoT). Diferentemente do Arduino UNO, o ESP32 integra nativamente conectividade Wi-Fi e Bluetooth, o que o torna uma solução altamente versátil para sistemas conectados. Além disso, o dispositivo oferece elevado poder de processamento, ampla quantidade de periféricos e suporte a múltiplas linguagens de programação, como C/C++, Python, Lua, MicroPython e JavaScript.

Devido à sua conectividade, o ESP32 é amplamente empregado em aplicações de automação residencial, sistemas de monitoramento remoto, dispositivos vestíveis (*wearables*), sensores inteligentes e soluções de rastreamento. Projetos como despertadores inteligentes, sensores de fumaça, dispositivos de monitoramento de ativos e câmeras de segurança ilustram a versatilidade dessa plataforma.

Dessa forma, enquanto o Arduino UNO se destaca pela simplicidade, baixo custo e facilidade de aprendizado, o ESP32 configura-se como uma evolução natural para aplicações que demandam maior capacidade de processamento, conectividade e escalabilidade. Ambas as plataformas, portanto, são relevantes e complementares no contexto do desenvolvimento de sistemas embarcados. Além de uma arquitetura de *hardware* mais robusta, o ESP32 apresenta elevada flexibilidade do ponto de vista

de *software*, oferecendo suporte a diferentes linguagens de programação, como C/C++ (compatível com a IDE do Arduino), Python (por meio do MicroPython), Lua e JavaScript, o que permite ao desenvolvedor selecionar a ferramenta mais adequada de acordo com a complexidade e os requisitos da aplicação.

A fim de sintetizar as distinções técnicas entre o ESP32 e as placas Arduino Uno e Nano, a Tabela 1 apresenta um comparativo direto das principais especificações. A análise desses dados é fundamental para subsidiar a escolha da plataforma mais adequada, considerando os requisitos de processamento, conectividade e eficiência energética do projeto proposto.

Tabela 1 - Comparativo entre ESP32 e Arduino UNO

Característica	ESP32 (DevKit V1)	Arduino Uno R3
Processador	Xtensa Dual-Core 32-bit LX6	Microcontrolador ATmega328P 8-bit
***Clock* (Velocidade)**	Ajustável entre 80 MHz e 240 MHz	16 MHz
Memória Flash	Geralmente 4 MB (até 16 MB)	32 KB (0.5 KB usado pelo Bootloader)
RAM	520 KB (interna)	2 KB (Uno)
GPIOs	Até 34 GPIOs	14 (Digitais) + 6 (Analógicos/Digitais)
Interfaces de Comunicação	3xUART, 3xSPI, 2xI2C, 2xI2S, CAN	1xUART, 1xSPI, 1xI2C
Conectividade Sem Fio	Wi-Fi 802.11 b/g/n, Bluetooth v4.2 e v5.0 BLE	N/A (Requer módulos externos)
Modos de Economia de Energia	Alto com Wi-Fi ativo / Baixíssimo em *Deep Sleep*	Modos básicos (Idle, *Power-down*)
Ambientes de Desenvolvimento	Arduino IDE, ESP-IDF, MicroPython	Arduino IDE, PlatformIO
Linguagens de Programação	C/C++, MicroPython	Arduino (C/C++)
Tensão de Operação (Lógica)	3.3 V (Não tolerante a 5V)	5 V (Padrão TTL)
Conversor A/D (ADC)	12-bit (18 canais)	10-bit (6 Canais)
Observações	Mais flexível devido à conectividade Wi-Fi/ Bluetooth e maior capacidade de processamento e memória	Menos flexível em termos de *hardware*, mas com grande comunidade e muitas bibliotecas

Além das características já descritas, o ESP32 apresenta recursos avançados que o diferenciam de microcontroladores convencionais. O núcleo de processamento do ESP32 é baseado na arquitetura Xtensa LX6 de 32 bits, que oferece desempenho de até 600 DMIPS, conforme descrito no DataSheet, possibilitando a execução simultânea de algoritmos complexos de Processamento Digital de Sinais (DSP) e pilhas de comunicação robustas, sem comprometer a responsividade do sistema.

Com o objetivo de maximizar a eficiência energética em aplicações voltadas à Internet das Coisas (IoT), o ESP32 incorpora um coprocessador ULP (*Ultra-Low-Power*). Esse módulo permite que o sistema opere em modo de suspensão profunda (*Deep Sleep*), com consumo de corrente da ordem de microampères, enquanto continua monitorando sensores e periféricos essenciais**.** O processador principal é ativado apenas quando condições previamente definidas são atendidas, o que contribui de forma significativa para a extensão da vida útil da bateria.

2.7 Módulos de GPS de Baixo Custo

O GPS, conforme mencionado em tópicos anteriores, é um sistema de navegação por satélite que utiliza uma constelação de satélites em órbita terrestre para determinar, com elevada precisão, a posição de um receptor em qualquer ponto do planeta. Para que isso seja possível, o sistema é constituído por três segmentos principais:

- Segmento Espacial: A constelação GPS é composta por, no mínimo, 24 satélites distribuídos em órbitas a aproximadamente 20.200 km de altitude, garantindo que, em qualquer local da Terra, pelo menos seis satélites estejam visíveis simultaneamente;
- Segmento de Controle: Responsável pelo monitoramento contínuo dos satélites, pela correção de suas órbitas e pela atualização dos dados transmitidos, assegurando a precisão, a confiabilidade e a integridade do sistema; e

- Segmento do Usuário: Compreende os receptores GPS, ou seja, os dispositivos capazes de captar os sinais dos satélites, como *smartphones*, navegadores automotivos e módulos utilizados em plataformas embarcadas, como Arduino e ESP32, permitindo o cálculo da posição geográfica.

O funcionamento do GPS baseia-se no princípio da trilateração, que ocorre em três etapas fundamentais:

- Transmissão: cada satélite transmite continuamente informações sobre sua posição orbital e o instante exato em que o sinal foi emitido;
- Cálculo da distância: o receptor mede o tempo decorrido entre a transmissão e a recepção do sinal. Como as ondas de rádio se propagam à velocidade da luz, esse intervalo de tempo permite calcular a distância entre o satélite e o receptor;
- Determinação da posição: para obter uma posição bidimensional (latitude e longitude), são necessários sinais de, no mínimo, três satélites. Para uma posição tridimensional (incluindo altitude), são exigidos quatro ou mais satélites, o que também possibilita a correção do erro do relógio interno do receptor.

Nesse contexto, diversos módulos GPS destacam-se pela ampla utilização em projetos de eletrônica embarcada, devido ao baixo custo, facilidade de integração e desempenho satisfatório. Entre suas principais características técnicas, destacam-se:

- Sensibilidade: alta sensibilidade de recepção, tipicamente em torno de –161 dBm, permitindo operação confiável mesmo em ambientes com sinais fracos;
- Precisão: precisão horizontal média de aproximadamente 2,5 metros em condições ideais de recepção;
- Comunicação: interface de comunicação serial do tipo UART, com taxa de transmissão padrão de 9600 bps, compatível com níveis lógicos de 3,3 V e 5 V, facilitando a conexão com microcontroladores;
- Consumo de energia: corrente típica de cerca de 45 mA em operação normal, com modo de economia de energia que reduz o consumo para aproximadamente 11 mA; e

- Funcionalidades adicionais: presença de bateria de backup, que possibilita inicializações rápidas (*hot start*) em torno de 1 segundo, além de LED indicador de *status*, que sinaliza quando o módulo estabelece fixação com os satélites.

Diante desse cenário, destacam-se diversos módulos GPS disponíveis no mercado para aplicações em prototipagem com Arduino e ESP32, os quais serão analisados e comparados posteriormente neste trabalho. Entre os principais modelos, podem ser citados: SIM7600SA; Skylab SKM53; Beitian BN-180; Beitian BN-220; Beitian BN-880 / BN-880Q; NEO-6M (com antena externa); NEO-7M; NEO-M8N; NEO-M8P; e o ZED-F9P.

2.7.1 SIM7600SA

O SIM7600SA é um módulo de comunicação 4G LTE Cat-1 que integra conectividade de dados móveis e serviços de posicionamento GNSS, sendo indicado para aplicações que exigem comunicação remota, localização em tempo real e conexão via rede celular. Suas configurações se resumem em:

- Tecnologia: 4G LTE Cat-1;
- Velocidade: até 10 Mbps (*download*) e 5 Mbps (*upload*);
- Suporte GNSS: integrado (GPS);
- Padrões: LTE-FDD e LTE-TDD;
- Interface: UART (RX/TX);
- Formato: LCC1;
- Compatibilidade: séries SIM5320/SIM5360 e SIM7600/SIM7600-H (facilita migração de 3G para 4G);
- Aplicações: IoT, telemetria, rastreamento, sistemas embarcados conectados;
- Observação: requer uso de SIM Card; e
- Preço: ~ R$180,00 - R$529,00.

2.7.2 Skylab SKM53

O SKM53 é um módulo GPS compacto com antena embutida, baseado no chipset MediaTek 3339, adequado para aplicações em ambientes com baixa visibilidade de sinal. De acordo com o fabricante é um módulo que possui boa estabilidade e desempenho em cânions urbanos e áreas com folhagem densa, permitindo fácil integração em sistemas embarcados.

- Chipset: MediaTek 3339;
- Sensibilidade de rastreamento: –165 dBm;
- Interface: UART (RX/TX) – conector de 6 pinos;
- Antena: integrada;
- Aplicações: rastreamento veicular, navegação portátil, localizadores pessoais;
- Preço: ~ R$79,00.

2.7.3 Beitian BN-880Q

O BN-800Q é um módulo GNSS multiconstelação com bússola eletrônica integrada, indicado para aplicações que exigem posicionamento e orientação, sendo adequado para aplicações de navegação e controle de movimento, por apresentar alta flexibilidade de comunicação, boa precisão e suporte a aplicações de navegação dinâmica.

- Chipset: M10050;
- Constelações: GPS, BDS, Galileo, QZSS, SBAS;
- Precisão horizontal: 1,5 m CEP;
- Frequência de atualização: 0,25 a 18 Hz;
- Interface: UART (TTL), 4800 a 921600 bps;
- Bússola: QMC5883L integrada;
- Consumo: ~50 mA @ 5 V;
- Preço: ~ R$167,00.

2.7.4 NEO-6M (com antena)

Módulo GPS clássico baseado no chipset u-blox 6, sendo uma solução simples e econômica, mas limitada quando comparada a módulos de multiconstelação.

- Canais: 50;
- Tempo de aquisição: < 1 s (TTFF);
- Interface: UART (nível lógico 3,3 V);
- Características: supressão de interferências e multipath;
- Preço: ~ R$56,00.

2.7.5 NEO-M8N

Módulo GNSS multiconstelação de alto desempenho da u-blox, capaz de oferecer alta precisão e robustez, sendo ideal para aplicações industriais e automotivas.

- Constelações: GPS, Galileo, BeiDou, GLONASS (até 3 simultâneas);
- Frequência de atualização: até 18 Hz;
- Interfaces: UART, I²C, SPI;
- Recursos: proteção contra *spoofing*, memória flash, geofencing;
- Tensão: 2,7 a 3,6 V;
- Preço: ~ R$114,00.

2.7.6 NEO-M8P (RTK)

Módulo GNSS com suporte a Real-Time Kinematic para precisão centimétrica, indicada para aplicações que exigem posicionamento de altíssima precisão.

- Precisão: nível de centímetros (com estação base);
- Modos: *base* e *rover;*
- Aplicações: agricultura de precisão, VANTs, robótica autônoma;
- Recursos: *survey-in*, linha de base móvel (MB);
- Preço: ~ R$730,00.

2.7.7 ZED-F9P

Módulo GNSS multibanda de última geração da u-blox, com RTK e PPP-RTK, representando o estado da arte em posicionamento GNSS, com altíssima precisão e confiabilidade.

- Tecnologia: RTK multibanda e PPP-RTK;
- Precisão: centimétrica;
- Dimensões: 17 × 22 × 2,4 mm;
- Aplicações: automação industrial, UAVs, máquinas agrícolas;
- Preço: ~ R$1.000,00.

Diante dos diversos módulos disponíveis no mercado, anteriormente mencionados, torna-se necessária a realização de uma análise comparativa que destaque seus principais aspectos construtivos e de *software*, a fim de auxiliar na escolha do módulo mais adequado para a aplicação proposta. Dessa forma, nas Tabelas 2, 3 e 4 são apresentadas comparações relacionadas ao suporte a diferentes constelações de satélites, sensibilidade de recepção, interfaces de comunicação, precisão de posicionamento, recursos adicionais, faixa de preço e outras considerações relevantes no processo de seleção do módulo GNSS mais apropriado.

Tabela 2 - Comparação entre Módulos GNSS quanto ao Suporte de Constelações e Sensibilidade

Módulo	GNSS Suportado	Sensibilidade (dBm)	Interfaces
SIM7600SA	GPS, GLONASS, BeiDou, LTE CAT1	-165 (Rastreamento)	UART, USB, LTE
Skylab SKM53	GPS (MTK3339)	-165 (Rastreamento)	UART
Beitian BN-880Q	GPS, GLONASS, Galileo, BeiDou, QZSS, SBAS	-167 (Rast.), -148 (Cold)	UART (TTL)
NEO-6M	GPS (u-blox 6), SBAS	-161 (Rastreamento)	UART (3.3V)
NEO-M8N	GPS, GLONASS, Galileo, BeiDou, QZSS, SBAS	-167 (Rast.), -148 (Cold)	UART, I2C, SPI
NEO-M8P	GPS, GLONASS + RTK (L1)	-161 (Navegação)	UART, I2C, SPI
ZED-F9P	GPS, GLONASS, Galileo, BeiDou (Multibanda) + RTK	-167 (Navegação)	UART, I2C, SPI

Tabela 3 - Comparação entre Módulos GNSS quanto a Precisão, Recursos Extras e Preço

Módulo	Precisão	Recursos Especiais	Preço (R$)
SIM7600SA	~2,5 m	Modem 4G integrado (10/5 Mbps); GNSS multiconstelação	R$ 180,00 a R$ 529,00
Skylab SKM53	~2,5 m	Antena cerâmica integrada; Conector 6 pinos (2.0 mm)	R$60,00 a R$120,00
Beitian BN-880Q	~2,5 m	Chipset M10; Bússola QMC5883L; Atualização 18Hz;	R$130,00 a R$220,00
NEO-6M	~2,5 m	50 canais; baixo consumo	R$35,00 a R$80,00
NEO-M8N	~2,5 m	Recepção simultânea de 3 constelações; AssistNow	R$90,00 a R$200,00
NEO-M8P	1–2 cm (RTK)	GNSS com RTK integrado	R$600,00 a R$1000,00
ZED-F9P	1 cm (RTK)	Multibanda; RTK e PPP-RTK	R$1000,00 a R$2500,00

Quadro 1 - Considerações Finais dos Módulos GNSS

Módulo	Considerações Finais
SIM7600SA	Monitoramento remoto. Permite transmissão de dados via rede 4G, sendo adequado para aplicações de rastreamento e telemetria em áreas rurais.
Skylab SKM53	Compacto e de baixo consumo. Indicado para aplicações de rastreamento simples e dispositivos embarcados de pequeno porte.
Beitian BN-880Q	Desempenho aprimorado. Receptor GNSS multiconstelação com bússola integrada, adequado para aplicações de navegação e veículos em movimento.
NEO-6M	Econômico. Amplamente utilizado em projetos acadêmicos e iniciantes, porém limitado ao sistema GPS.
NEO-M8N	Alta performance. Receptor GNSS multiconstelação com melhor precisão padrão (sem RTK), amplamente utilizado em aplicações de navegação e agricultura de precisão de baixo custo.
NEO-M8P	Precisão centimétrica. Receptor GNSS com tecnologia RTK, indicado para aplicações que exigem alta precisão, como mapeamento e agricultura de precisão.
ZED-F9P	Profissional. Receptor GNSS multibanda com tecnologia RTK, considerado referência para aplicações de alta precisão em agricultura de precisão, geodésia e topografia.

A análise comparativa entre os módulos GNSS supracitados teve como objetivo fundamentar a seleção do componente mais adequado à arquitetura proposta, considerando o equilíbrio entre custo, precisão posicional, eficiência energética e facilidade de integração com o ESP32. Para além dos parâmetros nominais apresentados nos *datasheets*, julgou-se essencial incorporar resultados de testes empíricos, realizados de forma independente e divulgados em plataformas como o youtube, bem como relatos da comunidade especializada acerca do desempenho prático desses sensores em cenários reais de aplicação.

Com base na correlação entre os dados técnicos e no estudo comparativo conduzido por iforce2d (2021), no qual se avaliou a estabilidade da precisão entre diferentes gerações de módulos GNSS, definiu-se o módulo u-blox NEO-M8N como a escolha mais apropriada para este projeto. Tal decisão se justifica pelo seu destacado custo-benefício, pelo suporte simultâneo a múltiplas constelações de satélites e pela elevada estabilidade de sinal em comparação com modelos de entrada, assegurando a confiabilidade requerida para a aplicação agrícola proposta.

2.8 Integração ESP32 e GPS para Agricultura

Com base nessas premissas, a arquitetura da solução proposta foi estruturada utilizando o microcontrolador ESP32 integrado ao módulo GNSS u-blox NEO-M8N. Essa escolha fundamenta-se principalmente em critérios de custo-benefício, facilidade de integração entre os dispositivos, disponibilidade de conectividade Wi-Fi nativa do microcontrolador ESP32 e no desempenho satisfatório do módulo GNSS em aplicações de posicionamento e navegação. Além disso, o NEO-M8N apresenta suporte a múltiplas constelações de satélites, o que contribui para uma maior confiabilidade e precisão na obtenção das coordenadas geográficas. A Figura 5 apresenta a placa de prototipagem utilizada no desenvolvimento do sistema.

Figura 5 - Placa de Desenvolvimento ESP32 de 30 Pinos

Fonte: (O autor, 2026)

O módulo NEO-M8N destaca-se por sua estabilidade posicional. Enquanto módulos de entrada, como o NEO-6M, apresentam variações (*drift*) que podem atingir

até 10 metros, além de oscilações significativas na altitude, o NEO-M8N mantém uma dispersão de dados muito mais concentrada em torno da posição real, proporcionando maior confiabilidade nas medições.

Outro fator determinante é o suporte a múltiplas constelações simultaneamente. O NEO-M8N é capaz de rastrear até três sistemas GNSS ao mesmo tempo, incluindo GPS (EUA), GLONASS (Rússia), BeiDou (China), Galileo (União Europeia), QZSS (Japão) e SBAS (EGNOS, WAAS, MSAS, GAGAN). Essa característica é particularmente relevante em ambientes agrícolas, onde podem existir obstruções parciais do céu por árvores, edificações ou relevo, aumentando a disponibilidade de satélites e a robustez do posicionamento.

De acordo com análises comparativas realizadas por usuários e especialistas, o NEO-M8N é frequentemente classificado como a melhor opção em termos de “melhor custo-benefício” para aplicações que demandam boa precisão, mas que não justificam o investimento elevado de módulos RTK, como o ZED-F9P.

O módulo também oferece frequência de atualização configurável de 1 Hz a 10 Hz, permitindo maior taxa de aquisição de dados quando necessário. Em termos de comunicação, é compatível com múltiplas interfaces, incluindo UART (9600 ou 38400 bps), I²C e SPI, possibilitando integração direta com plataformas como Arduino, ESP8266, ESP32 e Raspberry Pi.

Adicionalmente, o módulo NEO-M8N oferece suporte à tecnologia A-GPS (Assisted GPS), por meio do serviço AssistNow, que possibilita a redução do tempo necessário para obtenção do primeiro posicionamento (*Time To First Fix – TTFF*) através do uso de dados auxiliares previamente armazenados ou fornecidos pela rede.

O módulo também possui, em muitas versões comerciais, antena cerâmica integrada do tipo patch, além de um conector u.FL (ou IPEX) que permite a utilização de antena externa ativa, aumentando a flexibilidade de instalação e melhorando a recepção do sinal em ambientes com maior obstrução.

Algumas placas baseadas no NEO-M8N incluem ainda memória não volátil externa (EEPROM ou Flash) para o armazenamento de configurações do receptor entre

reinicializações. Além disso, o módulo apresenta alta sensibilidade de recepção, característica essencial para a operação em locais com visibilidade parcial do céu, como áreas urbanas ou regiões com vegetação densa.

A Figura 6 apresenta o módulo NEO-M8N utilizado neste trabalho, destacando a antena cerâmica integrada.

Figura 6 - Módulo GNSS NEO-M8N e Antena

Fonte: (CURTO CIRCUITO, 2026)

Assim, a combinação entre compatibilidade multiconstelação, maior confiabilidade de fixação, baixo consumo energético e facilidade de uso com ESP32 torna o conjunto ESP32 + NEO-M8N especialmente adequado para aplicações em ambientes rurais e projetos de sistemas embarcados voltados ao monitoramento e automação, atendendo de forma equilibrada aos requisitos técnicos propostos neste trabalho.

3. METODOLOGIA

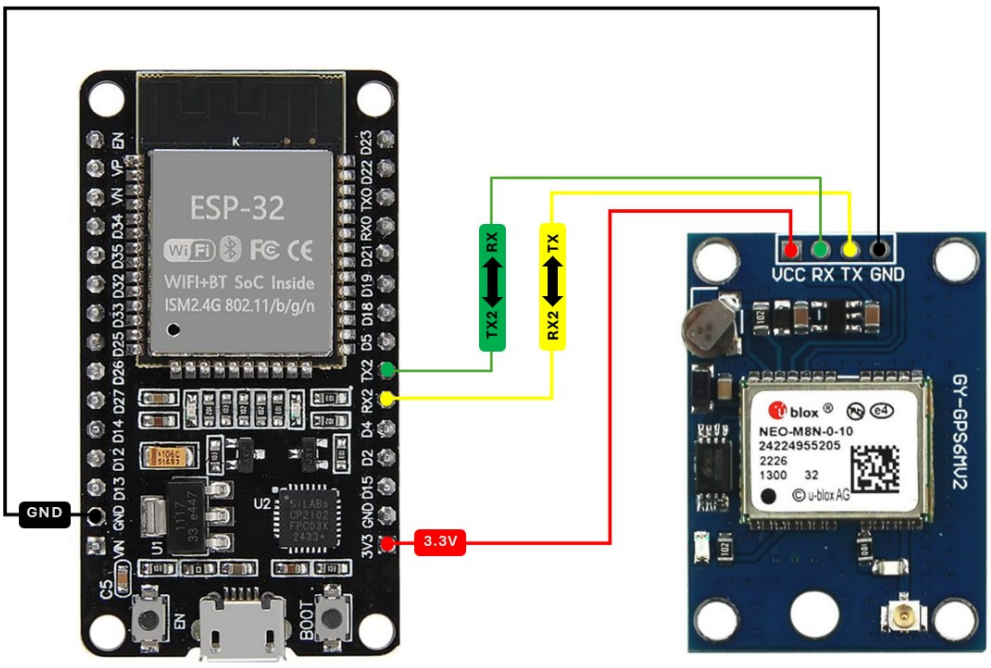
Com base nos conceitos teóricos fundamentados anteriormente, este capítulo detalha a operacionalização e os procedimentos técnicos utilizados no desenvolvimento do Sistema de Monitoramento de Área de Baixo Custo. A arquitetura proposta utiliza o microcontrolador ESP32 integrado ao módulo GNSS NEO-M8N para a aquisição de dados geoespaciais em tempo real. O fluxo metodológico compreende o processamento embarcado das informações, o armazenamento estruturado em sistema de arquivos (formato CSV) e, por fim, o tratamento dos dados via linguagem Python. Esta etapa final é responsável pelo cálculo preciso da distância percorrida, do tempo de operação e da área coberta, permitindo a geração de mapas interativos para visualização georreferenciada. Nesse contexto, este capítulo apresenta a arquitetura do sistema, descreve o programa desenvolvido, seus princípios de funcionamento e o modo de utilização, evidenciando as etapas de coleta, processamento, armazenamento e visualização dos dados.

A Tabela 5 detalha a configuração dos pinos do microcontrolador ESP32 utilizados para a alimentação do módulo GNSS NEO-M8N e para o tráfego de sinais. A interface de dados fundamenta-se no protocolo de comunicação serial assíncrona SCI (*Serial Communication Interface*), utilizando os periféricos UART (*Universal Asynchronous Receiver/Transmitter*) internos do ESP32. O respectivo esquema de interconexão elétrica está representado na Figura 7.

Tabela 4 - Esquema de Ligação do ESP32 e NEO-M8N

Pino ESP32	**Pino NEO-M8N**	**Função**	**Fio**
3.3V	VCC	Alimentação positiva (3.3V)	Vermelho
GND	GND	Referência (Terra)	Preto
GPIO 16 (RX2)	TX	Transmissão de dados NMEA	Amarelo
GPIO 17 (TX2)	RX	Recepção de comandos (Opcional)	Verde

Figura 7 - Esquema de ligação elétrica entre ESP32 e o módulo NEO-M8N



Fonte: (O autor, 2026)

3.1 Arquitetura Geral do Sistema

A arquitetura do sistema desenvolvido segmenta-se em duas fases distintas: a aquisição de dados via sistema embarcado e o pós-processamento em ambiente computacional . Na etapa inicial, o módulo NEO-M8N transmite as sentenças do protocolo NMEA (tais como $GPGGA e $GPRMC) via interface de comunicação serial (SCI). Esses dados são interceptados pelo microcontrolador ESP32 (versão de 30 pinos) através da GPIO 16 (RX2), vinculada ao periférico UART2, transportando variáveis geoespaciais críticas como latitude, longitude, tempo universal (UTC), velocidade e curso. Para a decodificação eficiente e estruturada dessas mensagens, utiliza-se a biblioteca TinyGPS++, selecionada por sua alta robustez e reduzido *overhead* computacional em aplicações GNSS de tempo real.

Inicialmente, o módulo GNSS NEO-M8N capta os sinais provenientes das diferentes constelações de satélites e transmite os dados por meio de comunicação serial do tipo UART. Nessa comunicação, o pino TX do módulo é utilizado para a transmissão das sentenças NMEA, que contêm informações configuradas

previamente pelo usuário, como latitude, longitude, horário e outros parâmetros de navegação.

O ESP32 realiza o processamento primário dessas informações, organizando os dados recebidos e tornando-os legíveis para posterior armazenamento. Em seguida, os pontos geográficos podem ser registrados em arquivos nos formatos .txt ou .csv. Alternativamente, caso um computador esteja conectado ao sistema, esses dados também podem ser visualizados por meio do *Serial Monitor* do *software* Arduino IDE, possibilitando sua posterior manipulação em ambiente computacional.

Na etapa seguinte, o arquivo gerado é transferido para um computador por meio de conexão Wi-Fi, Bluetooth ou via cabo. Os dados são então processados por um script em Python, responsável pelo cálculo da distância total percorrida, do tempo de operação e da área coberta, além da geração de um mapa interativo por meio da biblioteca Folium, que permite a visualização georreferenciada dos pontos em navegadores web. Alternativamente, os pontos geográficos podem ser importados para ferramentas como o Google Earth, possibilitando a visualização do trajeto percorrido sobre imagens de satélite.

3.2 Aquisição dos Dados GNSS pelo NEO-M8N

Para determinar a posição de uma pessoa ou de uma máquina por meio de um módulo GNSS, é necessário seguir um conjunto de etapas e processos específicos. Conforme apresentado neste capítulo, o primeiro passo consiste na utilização do módulo NEO-M8N, responsável pela obtenção dos dados provenientes do protocolo NMEA, os quais fornecem informações sobre a posição geográfica, além de outros dados relevantes para o usuário.

Como discutido no capítulo anterior, o módulo NEO-M8N foi selecionado devido à sua eficiência aliada a um excelente custo-benefício, quando comparado a soluções concorrentes disponíveis no mercado. Esse módulo se destaca por permitir a recepção simultânea de até três sistemas GNSS, como GPS e Galileo combinados com BeiDou ou GLONASS, reconhecendo múltiplas constelações ao mesmo tempo e oferecendo alta precisão de posicionamento, mesmo em ambientes urbanos densamente povoados ou em condições de sinal degradado.

Com o objetivo de proporcionar um posicionamento ainda mais rápido e preciso, o NEO-M8N também oferece suporte à integração com sistemas de aumento, tais como QZSS, GAGAN e IMES, além de WAAS, EGNOS e MSAS. O módulo dispõe, ainda, de recursos avançados, como proteção de integridade das mensagens, *geofencing* e detecção de *spoofing*, bem como interfaces configuráveis, o que facilita sua adaptação às necessidades específicas de diferentes aplicações.

Com o intuito de entender melhor os sistemas de aumento mencionados anteriormente, esses serão descritos a seguir:

-Sistema de Aumento Baseado em Satélite (SBAS)

Esses sistemas complementam os dados GNSS com informações adicionais de correção e integridade transmitidas via satélite. Tais dados permitem que os receptores GNSS melhorem significativamente a precisão do posicionamento. Além disso, os satélites SBAS podem ser utilizados como satélites adicionais para navegação, aumentando a disponibilidade do sistema. Os principais sistemas SBAS suportados são GAGAN, WAAS, EGNOS e MSAS.

-Sistema de Satélites Quasi-Zenith (QZSS):

O QZSS é um sistema regional de navegação por satélite que transmite sinais GPS L1 C/A adicionais para a região do Pacífico, abrangendo principalmente o Japão e a Austrália. Os módulos da série NEO-M8 são capazes de receber e rastrear esses sinais simultaneamente aos sinais GPS convencionais, resultando em maior disponibilidade e confiabilidade do posicionamento, especialmente em ambientes urbanos com grande concentração de edifícios. O sinal L1-SAIF fornecido pelo QZSS pode ser habilitado por meio de mensagens de configuração GNSS.

-Sistema Japonês de Mensagens Internas (IMES):

O IMES é utilizado para fornecer informações de localização em ambientes internos por meio de transmissores de baixa potência que emitem sinais semelhantes aos do GPS. Os módulos NEO-M8 podem ser configurados para receber e demodular

esses sinais, possibilitando a estimativa de posição em ambientes onde o sinal GNSS convencional é limitado.

-GPS Diferencial (D-GPS):

De acordo com a especificação RTCM 10402.3, o uso de D-GPS permite melhorar significativamente a precisão do posicionamento GPS. A implementação do protocolo RTCM nos módulos NEO-M8 oferece suporte às mensagens RTCM versão 2.3, possibilitando a aplicação de correções diferenciais.

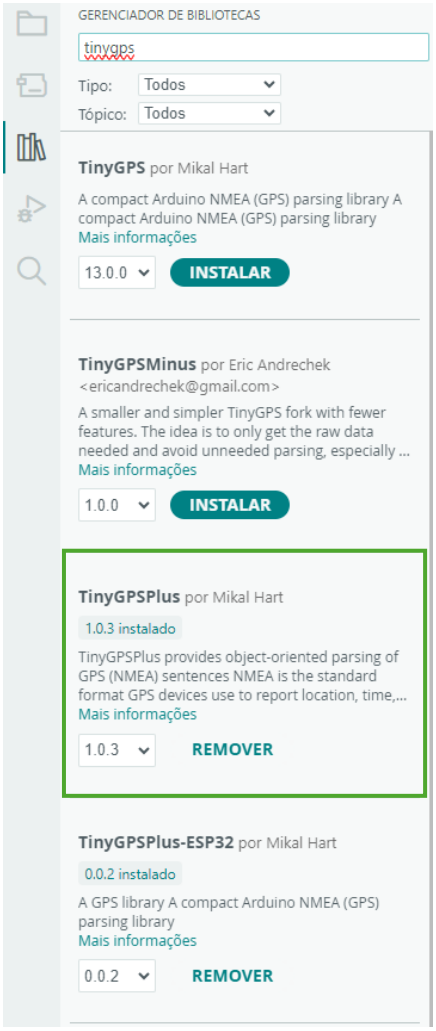
3.2.1 TinyGPS++

Diante da ampla gama de benefícios oferecidos por esse módulo GNSS, torna-se necessária a utilização da biblioteca TinyGPSPlus para a obtenção dos dados requeridos pelo usuário, uma vez que ela é responsável por realizar a análise sintática orientada a objetos das sentenças GPS no formato NMEA. Conforme já mencionado, o NMEA é o padrão utilizado por dispositivos GPS para o reporte de informações como posição geográfica, horário, altitude, velocidade, entre outras.

A biblioteca TinyGPSPlus destaca-se por ser compacta, robusta e amplamente difundida, sendo capaz de interpretar as sentenças NMEA mais comuns, como GGA e RMC. Além disso, ela pode ser personalizada para a extração de dados provenientes de qualquer sentença compatível com o padrão NMEA, o que amplia significativamente sua flexibilidade de aplicação.

Como etapa inicial para a sua utilização, é necessário realizar a instalação da biblioteca na IDE do Arduino, a qual é plenamente compatível com o microcontrolador ESP32. Nesse processo, ao pesquisar pela biblioteca no gerenciador de bibliotecas, deve-se atentar para a seleção da versão correta, denominada TinyGPSPlus, desenvolvida por Mikal Hart, como pode ser observado na Figura 8.

Figura 8 - Biblioteca TinyGPSPlus para obtenção dos dados NMEA



Fonte: (O autor, 2026)

Essa biblioteca recebe os dados no protocolo padrão NMEA e é capaz de processá-los, extraindo de forma estruturada as informações de interesse do usuário, como latitude, longitude, data, hora, entre outros parâmetros relevantes. Dessa forma, ela abstrai a complexidade da interpretação das sentenças NMEA, disponibilizando os dados já decodificados para uso direto na aplicação.

Sendo assim, a fim de compreender o seu funcionamento, a Figura 9 apresenta um exemplo básico disponibilizado pela própria biblioteca, acompanhado de comentários explicativos no código, com o objetivo de facilitar o entendimento de sua lógica de operação e funcionalidade.

Figura 9 - Exemplo básico de como funciona a TinyGPSPlus

BasicExample.ino

```ino
#include <TinyGPSPlus.h>   // Inclui a biblioteca TinyGPSPlus para decodificar sentenças NMEA

// Fluxo de dados NMEA de exemplo (simula um GPS real)
const char *gpsStream =
  "$GPRMC,045103.000,A,3014.1984,N,09749.2872,W,0.67,161.46,030913,,,A*7C\r\n" // Sentença RMC (posição, velocidade, data)
  "$GPGGA,045104.000,3014.1985,N,09749.2873,W,1,09,1.2,211.6,M,-22.5,M,,0000*62\r\n" // Sentença GGA (fix, satélites, altitude)
  "$GPRMC,045200.000,A,3014.3820,N,09748.9514,W,36.88,65.02,030913,,,A*77\r\n" // Nova posição com maior velocidade
  "$GPGGA,045201.000,3014.3864,N,09748.9411,W,1,10,1.2,200.8,M,-22.5,M,,0000*6C\r\n" // GGA correspondente
  "$GPRMC,045251.000,A,3014.4275,N,09749.0626,W,0.51,217.94,030913,,,A*7D\r\n" // Outra leitura RMC
  "$GPGGA,045252.000,3014.4273,N,09749.0628,W,1,09,1.3,206.9,M,-22.5,M,,0000*6F\r\n"; // Última sentença GGA

// Cria o objeto TinyGPSPlus responsável por decodificar os dados do GPS
TinyGPSPlus gps;

void setup()
{
  Serial.begin(115200); // Inicializa a comunicação serial a 115200 bps
  // Percorre toda a string gpsStream caractere por caractere
  while (*gpsStream)
    // Envia cada caractere para o parser do TinyGPSPlus
    // Quando uma sentença NMEA válida é totalmente decodificada, retorna true
    if (gps.encode(*gpsStream++))
      displayInfo(); // Exibe as informações do GPS
        //gps.encode() retorna true quando uma sentença completa válida foi processada
        //gpsStream++ avança para o próximo caractere
  Serial.println(F("Done.")); // Indica fim do processamento
}

void loop()
{
}

void displayInfo()
{
  Serial.print(F("Location: ")); // Cabeçalho de localização

  // Verifica se a localização é válida
  if (gps.location.isValid())
  {
    Serial.print(gps.location.lat(), 6); // Imprime latitude com 6 casas decimais
    Serial.print(F(","));                // Separador
    Serial.print(gps.location.lng(), 6); // Imprime longitude com 6 casas decimais
  }
  else
  {
    Serial.print(F("INVALID")); // Caso não haja fix válido
  }
  Serial.println(); // Quebra de linha no monitor serial
}
```

Fonte: (O autor, 2026)

O código apresentado tem como objetivo demonstrar o funcionamento da biblioteca TinyGPSPlus na decodificação de sentenças GPS no formato NMEA, sem a necessidade de um receptor GNSS físico, ou seja, sem a necessidade do módulo para obtenção dos dados. Para isso, é utilizado um fluxo de dados simulado, composto por sentenças NMEA do tipo RMC e GGA, que representam informações típicas fornecidas por um módulo GPS real, que são as sentenças contidas em “*gpsStream”.

Inicialmente, a biblioteca TinyGPSPlus é incluída no projeto, possibilitando a interpretação das sentenças NMEA. Em seguida, é criada uma string contendo múltiplas sentenças GPS concatenadas, que simulam dados recebidos via

comunicação serial. Cada sentença representa uma leitura distinta de posicionamento e termina com caracteres de fim de linha, reproduzindo fielmente o comportamento de um módulo GNSS em operação real.

Durante a execução da função setup(), o código percorre toda a string de dados simulados caractere por caractere. Cada caractere é enviado ao método encode() da biblioteca TinyGPSPlus, que funciona como um analisador sintático incremental. Esse método armazena temporariamente os caracteres recebidos até que uma sentença NMEA completa e válida seja identificada. Quando isso ocorre, o método retorna um valor verdadeiro, indicando que os dados internos do objeto GPS foram atualizados com sucesso.

Sempre que uma sentença válida é processada, a função displayInfo() é chamada. Essa função acessa os dados decodificados armazenados no objeto TinyGPSPlus e verifica a validade da informação de localização. Caso a posição seja válida, são exibidos no monitor serial os valores de latitude e longitude, já convertidos para o formato decimal, prontos para uso em aplicações de navegação, mapeamento ou registro de trajetórias. Caso contrário, é indicada a ausência de um posicionamento válido.

Por fim, após o processamento completo de todas as sentenças simuladas, o código encerra sua execução, uma vez que todo o fluxo ocorre dentro da função setup(). A função loop() permanece vazia, pois este exemplo tem caráter exclusivamente demonstrativo e não realiza leitura contínua de dados, como ocorreria em uma aplicação com um módulo GNSS real conectado ao microcontrolador.

Diante do exposto, torna-se necessário apresentar como será realizada a utilização da biblioteca TinyGPSPlus em conjunto com o microcontrolador ESP32 e o módulo GNSS NEO-M8N, de modo a exemplificar o processo de aquisição dos dados que serão empregados no desenvolvimento do projeto. Esses dados são obtidos por meio de comunicação serial via UART2 do ESP32 (TX2-GPIO17, RX2-GPIO16), conectada ao módulo GNSS, visando a transmissão e recepção das informações de posicionamento.

Sendo assim, a Figura 10 apresenta um exemplo de aplicação que ilustra a utilização do ESP32 em conjunto com o módulo GNSS para a aquisição e o processamento dos dados de localização.

Figura 10 - Exemplo do TinyGPS+ para ser utilizado com o NEO-M8N

DeviceExample.ino

```ino
#include <TinyGPSPlus.h>      // Inclui a biblioteca TinyGPSPlus para decodificar sentenças NMEA
#include <SoftwareSerial.h>  // Inclui a biblioteca SoftwareSerial para criar uma serial por software
static const int RXPin = 17, TXPin = 16; // Define o pino de recepção (RX) e transmissão (TX) da comunicação serial com o GPS
static const uint32_t GPSBaud = 4800; // Define a taxa de comunicação do módulo GPS (baud rate)
TinyGPSPlus gps; // Cria o objeto TinyGPSPlus, responsável por interpretar os dados NMEA
SoftwareSerial ss(RXPin, TXPin); // Cria o objeto SoftwareSerial para comunicação com o módulo GPS
void setup()
{
  Serial.begin(115200); // Inicializa a serial principal (USB) para depuração
  ss.begin(GPSBaud);    // Inicializa a serial por software na taxa do GPS
}
void loop()
{
  while (ss.available() > 0)   // Enquanto houver dados disponíveis vindos do GPS
  {
    // Lê um caractere da serial do GPS e envia para o parser TinyGPSPlus
    // Quando uma sentença NMEA completa e válida é processada,
    // o método encode() retorna true
    if (gps.encode(ss.read()))
      displayInfo(); // Exibe as informações decodificadas
  }
  // Verificação de segurança:
  // Se após 5 segundos poucos caracteres foram recebidos,
  // assume-se que o GPS não está conectado corretamente
  if (millis() > 5000 && gps.charsProcessed() < 10)
  {
    Serial.println(F("No GPS detected: check wiring."));
    while(true); // Trava o programa
  }
}
void displayInfo()
{
  Serial.print(F("Location: ")); // Cabeçalho de localização

  // Verifica se a posição GPS é válida (fix obtido)
  if (gps.location.isValid())
  {
    Serial.print(gps.location.lat(), 6); // Imprime latitude com 6 casas decimais
    Serial.print(F(","));                // Separador
    Serial.print(gps.location.lng(), 6); // Imprime longitude com 6 casas decimais
  }
  else
  {
    Serial.print(F("INVALID")); // Indica ausência de posição válida
  }
  Serial.println(); // Quebra de linha no Monitor Serial
}
```

Fonte: (O autor, 2026)

O código apresentado na Figura 10 tem como finalidade realizar a aquisição e o processamento de dados de posicionamento provenientes de um módulo GNSS NEO-M8N, utilizando o microcontrolador ESP32 e a biblioteca TinyGPSPlus. Inicialmente, são incluídas as bibliotecas TinyGPSPlus e SoftwareSerial, onde a primeira é responsável por interpretar as sentenças NMEA recebidas do módulo GNSS, enquanto a segunda permite a criação de uma porta serial adicional por *software*, utilizada para a comunicação com o receptor GPS. Em seguida, são definidos os pinos de recepção e transmissão da comunicação serial,

correspondentes às GPIOs 17 (RX) e 16 (TX) do ESP32, bem como a taxa de comunicação do módulo, configurada em 4800 bps.

Na sequência, é instanciado o objeto TinyGPSPlus, que atua como o analisador sintático dos dados NMEA, e o objeto SoftwareSerial, responsável por estabelecer o canal de comunicação serial com o módulo GNSS.

Durante a execução da função setup(), a comunicação serial principal é inicializada para fins de depuração e visualização dos dados no monitor serial, enquanto a serial por *software* é iniciada na mesma taxa configurada para o módulo GNSS. Após essa etapa, o sistema encontra-se apto a receber e processar os dados de posicionamento.

No laço principal loop(), o código verifica continuamente se há dados disponíveis provenientes do módulo GNSS. Enquanto houver caracteres disponíveis na serial, cada caractere é lido individualmente e enviado ao método encode() da biblioteca TinyGPSPlus. Esse método realiza a análise incremental das sentenças NMEA e retorna um valor verdadeiro sempre que uma sentença completa e válida é reconhecida. Nesse momento, os dados internos do objeto GPS são atualizados e a função displayInfo() é chamada para exibir as informações decodificadas.

Continuando, o código implementa uma verificação de segurança para identificar possíveis falhas de comunicação com o módulo GNSS. Caso, após um intervalo de cinco segundos, um número mínimo de caracteres tenha sido processado, o sistema assume que o módulo não foi detectado corretamente e exibe uma mensagem de erro no monitor serial, interrompendo a execução do programa.

Por fim, a função displayInfo() é responsável por acessar os dados de localização armazenados no objeto TinyGPSPlus. Nessa função, é verificada a validade da posição obtida e, caso o posicionamento seja válido, os valores de latitude e longitude são impressos no monitor serial em formato decimal, com seis casas decimais. Caso contrário, é indicada a ausência de um posicionamento válido.

3.2.2 Outros benefícios do NEO-M8N

Adicionalmente, o NEO-M8N possui memória flash interna, permitindo atualizações de *firmware*, característica que o torna especialmente adequado para aplicações industriais e automotivas. O módulo também conta com um amplificador de baixo ruído (LNA) frontal adicional, que facilita a integração com antenas externas, e com um filtro SAW frontal, responsável por proporcionar maior imunidade a interferências eletromagnéticas.

No que se refere ao desempenho, o módulo apresenta uma precisão horizontal mínima de aproximadamente 2,5 metros, valor considerado adequado para o projeto a ser desenvolvido, uma vez que os maquinários envolvidos são robustos e essa margem de erro não compromete o resultado final. Além disso, o sistema opera com uma taxa de atualização mínima de 5 Hz e máxima de 10 Hz, o que atende satisfatoriamente às necessidades da aplicação, considerando que os equipamentos operam a velocidades que não ultrapassam 10 km/h.

O módulo NEO-M8N também dispõe da função de odômetro, a qual fornece informações sobre a distância percorrida em solo (em metros) com base na posição e na velocidade Doppler obtidas pela solução de navegação. Para cada distância calculada desde a última reinicialização do odômetro, é estimado um valor com precisão de 1-sigma. A distância total percorrida é mantida e armazenada na memória BBR, possibilitando sua utilização em aplicações de registro de dados.

O recurso de *data logging* permite o armazenamento contínuo de informações de posição, velocidade e tempo em uma memória flash SQI integrada, com capacidade mínima de 16 Mbit. Adicionalmente, é possível registrar a distância do odômetro, e os dados armazenados podem ser posteriormente transferidos do receptor para análise adicional ou para conversão em ferramentas de mapeamento. Para mais detalhes sobre essas funcionalidades, recomenda-se a consulta à u-blox 8 / u-blox M8 *Receiver Description*, incluindo a *Protocol Specification*.

Enfim, os dados são lidos periodicamente pelo microcontrolador e convertidos para um formato estruturado, permitindo seu armazenamento e posterior processamento. A taxa de atualização do módulo é configurada de acordo com a

necessidade da aplicação, garantindo resolução espacial adequada para trajetos agrícolas.

3.3 Tratamento dos Dados

Após a aquisição dos dados fornecidos pelo módulo GNSS, torna-se fundamental a aplicação de métodos de filtragem, a fim de aumentar a exatidão das informações de posicionamento. Isso se deve ao fato de que o sistema GNSS apresenta uma margem de erro típica da ordem de até 2,5 metros, podendo ser ainda maior, dependendo de fatores como a quantidade de satélites visíveis, a qualidade do sinal recebido e as condições do ambiente de operação.

Diversos fenômenos podem degradar a precisão do posicionamento, destacando-se o efeito de multicaminho, causado pela reflexão dos sinais GNSS em superfícies como edificações, estruturas metálicas e até torres de rádio ou televisão. Essas reflexões fazem com que o sinal chegue ao receptor por diferentes trajetos, introduzindo atrasos e erros no cálculo da posição, conforme discutido em capítulos anteriores.

Além disso, é importante estabelecer uma taxa adequada de aquisição dos dados, definida de acordo com a aplicação. No contexto deste projeto, considerando que uma máquina agrícola opera a velocidades máximas da ordem de 10 km/h, a coleta de dados a cada segundo torna-se desnecessária e pode resultar em armazenamento redundante, além de aumentar o número de escritas na memória Flash. Assim, a definição de um intervalo de amostragem apropriado contribui para a eficiência do sistema e para a preservação da memória do microcontrolador.

Dessa forma, a filtragem dos dados GNSS é realizada com base em critérios mínimos de qualidade, tais como número de satélites utilizados no posicionamento, valor de HDOP (*Horizontal Dilution of Precision*) e intervalo de tempo entre amostras consecutivas. A adoção desses critérios permite a obtenção de informações mais confiáveis e adequadas para os cálculos e análises propostas no projeto. Sendo assim, na sequência serão apresentados os principais métodos empregados para a melhoria da precisão dos dados de posicionamento.

3.3.1 Condição de Validação do Sinal

A condição de validação do sinal consiste na implementação de uma estrutura condicional (if) no código-fonte, inserida no interior da função void loop(), a qual é executada continuamente pelo microcontrolador. Essa função é responsável por verificar, em tempo real, a chegada de novos dados provenientes do módulo GNSS. A partir disso, estabelece-se um conjunto de critérios que regulam quais informações de posicionamento podem ser consideradas válidas e, portanto, utilizadas pelo sistema, como pode ser visto na Figura 11.

Figura 11 - Condição de Validação do Sinal

```
void loop() {

  // Lê dados do GPS por até 1 segundo
  unsigned long start = millis();
  unsigned long teste=millis();
  static unsigned long time=0;

  while ((SerialGPS.available()>0) && (millis() - start < 1000)) {
    gps.encode(SerialGPS.read());
  }

  //Se houver atualização válida dos dados GPS, exibe e salva
  if (gps.location.isUpdated() && gps.location.isValid() && (gps.satellites.value() > 3) && (gps.hdop.hdop()<2)) {

    printGPSDistancia(); //Calcula Distancia
    salvarDadosGPS();    // Salva no SPIFFS
    salvarVariaveis();    // Salvar variavel
  }

  server.handleClient();  // Processa conexões dos clientes para tráfego
}
```

Fonte: (O autor, 2026)

Essa estrutura condicional tem como principal objetivo assegurar a confiabilidade dos dados de posicionamento, evitando o processamento e o armazenamento de informações imprecisas ou inconsistentes. Para que o código interno seja executado, quatro condições lógicas devem ser satisfeitas simultaneamente, conforme descrito a seguir.

a) gps.location.isUpdated()

Essa condição verifica se novos dados de localização foram recebidos desde a última leitura realizada pelo sistema. Dessa forma, garante-se que o microcontrolador esteja trabalhando apenas com informações atualizadas, evitando o processamento repetido de um mesmo ponto geográfico.

b) gps.location.isValid()

Essa verificação confirma se a posição obtida pelo módulo GNSS é válida, ou seja, se o receptor já conseguiu estabelecer um *fix* de posicionamento. Enquanto o módulo não dispõe de informações suficientes dos satélites, os valores de latitude e longitude são considerados inválidos, o que impede o uso de coordenadas inconsistentes ou inexistentes nos cálculos do sistema.

c) gps.satellites.value() > 3

Essa condição avalia se o número de satélites utilizados no cálculo da posição é superior a três. Considera-se esse limite, pois com menos de quatro satélites o posicionamento tende a apresentar baixa precisão, mesmo com o processo de triangulação realizado pelo módulo GNSS. A partir da utilização de quatro ou mais satélites, torna-se possível calcular latitude, longitude e tempo com maior confiabilidade, reduzindo significativamente os erros de localização.

d) gps.hdop.hdop() < 2

O HDOP representa a qualidade geométrica da distribuição dos satélites no plano horizontal. Valores de HDOP entre 1 e 2 indicam boa precisão, enquanto valores superiores a 2 já refletem uma degradação significativa da exatidão do posicionamento, podendo resultar em erros superiores a 2,5 metros. Embora o valor ideal de HDOP seja menor ou igual a 1, para garantir um funcionamento adequado durante a inicialização a frio do módulo GNSS, adotou-se como critério o valor inferior a 2. Dessa forma, o sistema assegura que apenas posições com qualidade geométrica satisfatória sejam aceitas.

Em conjunto, essas condições permitem que apenas dados de posicionamento atualizados, válidos e com boa qualidade geométrica sejam processados e armazenados, aumentando a confiabilidade das informações utilizadas ao longo do projeto.

3.3.2 Limiar Mínimo Baseado na Velocidade

A filtragem do ruído do sinal GPS, por meio da aplicação de um limiar mínimo de deslocamento baseado na velocidade, constitui uma técnica fundamental em sistemas de navegação e monitoramento, especialmente quando se busca garantir que a deriva estática (*drift*) não comprometa os cálculos de distância percorrida e área trabalhada. Essa abordagem tem como principal objetivo evitar o acúmulo de pequenas variações de posição que não representam um deslocamento real do módulo GNSS.

Esse fenômeno ocorre porque, mesmo quando um maquinário agrícola encontra-se estacionado, o receptor GPS continua fornecendo coordenadas que apresentam pequenas oscilações. Tais variações são decorrentes de fatores como erros atmosféricos, efeito de multicaminho e limitações inerentes ao próprio receptor. Caso essas variações sejam interpretadas como movimento real, o sistema passa a somar deslocamentos fictícios, resultando em valores superestimados de distância total e área calculada, o que compromete a confiabilidade e a transparência dos dados obtidos.

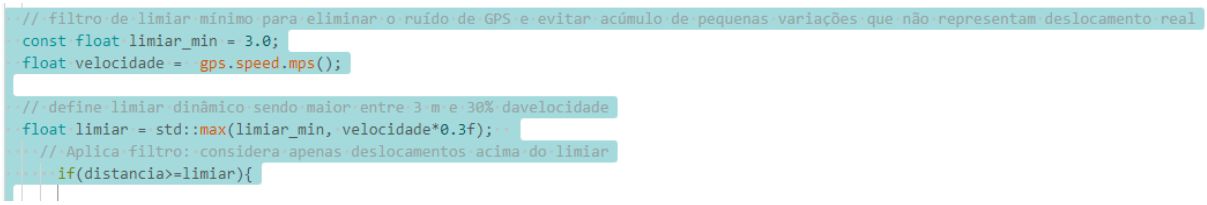
A técnica de filtragem adotada funciona como uma “trava lógica” baseada na dinâmica do movimento. O algoritmo executado no ESP32 realiza inicialmente a leitura da velocidade instantânea fornecida pelo módulo GNSS NEO-M8N, obtida por meio da biblioteca TinyGPS++, utilizando a função gps.speed.kmph() ou gps.speed.mps(). Em seguida, define-se um limiar mínimo de velocidade, considerado realista para a operação agrícola, por exemplo, 0,5 km/h ou 0,2 m/s.

A partir desse critério, o funcionamento do sistema ocorre da seguinte forma: quando a velocidade medida é superior ao limiar estabelecido, o sistema entende que o veículo encontra-se em deslocamento real e, portanto, utiliza as coordenadas recebidas para o cálculo da distância percorrida e da área trabalhada. Por outro lado, quando a velocidade é igual ou inferior ao limiar, as variações de latitude e longitude são desconsideradas, sendo tratadas como ruído eletrônico ou interferência do sinal GNSS.

Dessa forma, essa estratégia de filtragem contribui significativamente para o aumento da robustez e da precisão do sistema, garantindo que apenas

deslocamentos efetivos sejam considerados nos cálculos finais. Esse procedimento é fundamental em aplicações agrícolas, nas quais a confiabilidade dos dados de posicionamento é um fator essencial para a correta análise das áreas percorridas. A Figura 12 apresenta um trecho do código que demonstra a implementação do limiar mínimo como mecanismo de filtragem.

Figura 12 - Limiar Mínimo Baseado na Velocidade



Fonte: (O autor, 2026)

Inicialmente, define-se um limiar mínimo fixo de 3 metros (limiar_min), que estabelece a menor variação de posição aceitável para que um deslocamento seja considerado válido. Assim, apenas deslocamentos superiores a esse valor são contabilizados como movimento efetivo do equipamento. Conforme ilustrado na Figura 11, variações inferiores a esse limiar são atribuídas ao ruído inerente ao sinal GNSS, que pode ocorrer mesmo quando o receptor se encontra em repouso. Esse fenômeno está associado a limitações do próprio receptor, à geometria dos satélites e a interferências externas, como multipercurso (multipath) e condições atmosféricas.Em seguida, a velocidade instantânea do sistema é obtida por meio da função gps.speed.mps(), que fornece a velocidade em metros por segundo (m/s). Esse parâmetro é fundamental para permitir a adaptação dinâmica do critério de filtragem de acordo com o regime de movimento do equipamento.

Na sequência, é definido um limiar dinâmico de deslocamento, calculado como o maior valor entre o limiar mínimo fixo (3 metros) e 30% da velocidade instantânea. Essa estratégia possibilita que o sistema apresente maior tolerância ao ruído em situações de baixa velocidade e maior rigor em condições de deslocamento mais elevadas, ajustando automaticamente o critério de aceitação do movimento.

Dessa forma, o filtro adapta-se às diferentes condições de operação da máquina, evitando a superestimação da distância percorrida em baixas velocidades e

a subestimação em velocidades mais altas. Por fim, a verificação principal do filtro é realizada, de modo que os cálculos que se sucedem sejam feitos de forma eficaz para o salvamento dos dados.

3.3.3 Filtro de Média Móvel

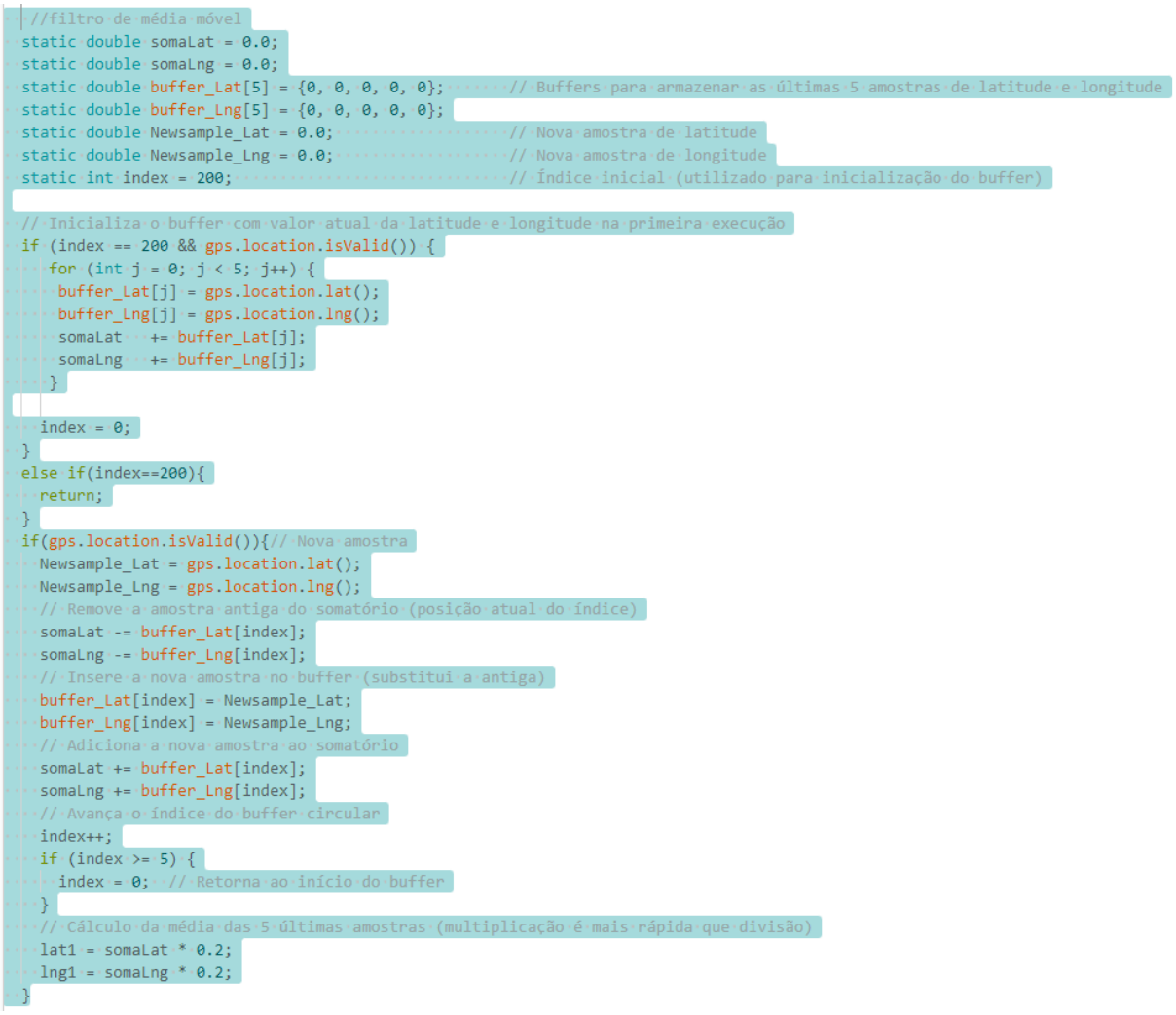
O Filtro de Média Móvel é uma técnica de processamento digital de sinais amplamente empregada em sistemas embarcados com o objetivo de reduzir ruídos e suavizar oscilações em séries temporais de dados. No contexto deste projeto, esse filtro desempenha um papel fundamental no tratamento das coordenadas fornecidas pelo módulo GNSS, as quais apresentam pequenas variações aleatórias mesmo quando o receptor se encontra em estado estacionário.

Do ponto de vista técnico, o filtro de média móvel pode ser classificado como um filtro passa-baixas, uma vez que atenua componentes de alta frequência associadas ao ruído do sinal, preservando as variações de baixa frequência relacionadas ao deslocamento real do equipamento. Seu funcionamento baseia-se na aplicação de uma janela deslizante sobre os dados amostrados: ao invés de considerar apenas a última coordenada lida, o sistema calcula a média aritmética de um conjunto de amostras recentes.

Para tanto, adota-se um *buffer* circular para o gerenciamento das leituras recentes de latitude e longitude, permitindo a atualização sequencial dos dados com baixo custo computacional. Essa técnica viabiliza a aplicação de um filtro de média móvel voltado à redução de ruídos e instabilidades do sinal GNSS, refinando os dados antes do cálculo de área e deslocamento. Cabe destacar que o atraso inerente ao aumento da janela do filtro é perfeitamente tolerável no contexto agrícola, visto que a baixa velocidade de operação do maquinário mitiga impactos indesejados no rastreamento.

Dessa forma, a aplicação do filtro de média móvel resulta em uma posição média mais estável e representativa do deslocamento real do equipamento, aumentando a confiabilidade das informações utilizadas nas etapas subsequentes do sistema, como pode ser visto na Figura 13.

Figura 13 - Filtragem por Média Móvel



```
//filtro de média móvel
static double somaLat = 0.0;
static double somaLng = 0.0;
static double buffer_Lat[5] = {0, 0, 0, 0, 0};      // Buffers para armazenar as últimas 5 amostras de latitude e longitude
static double buffer_Lng[5] = {0, 0, 0, 0, 0};
static double Newsample_Lat = 0.0;                  // Nova amostra de latitude
static double Newsample_Lng = 0.0;                  // Nova amostra de longitude
static int index = 200;                             // Índice inicial (utilizado para inicialização do buffer)

// Inicializa o buffer com valor atual da latitude e longitude na primeira execução
if (index == 200 && gps.location.isValid()) {
    for (int j = 0; j < 5; j++) {
      buffer_Lat[j] = gps.location.lat();
      buffer_Lng[j] = gps.location.lng();
      somaLat   += buffer_Lat[j];
      somaLng   += buffer_Lng[j];
    }

    index = 0;
}
else if(index==200){
    return;
}
if(gps.location.isValid()){// Nova amostra
    Newsample_Lat = gps.location.lat();
    Newsample_Lng = gps.location.lng();
    // Remove a amostra antiga do somatório (posição atual do índice)
    somaLat -= buffer_Lat[index];
    somaLng -= buffer_Lng[index];
    // Insere a nova amostra no buffer (substitui a antiga)
    buffer_Lat[index] = Newsample_Lat;
    buffer_Lng[index] = Newsample_Lng;
    // Adiciona a nova amostra ao somatório
    somaLat += buffer_Lat[index];
    somaLng += buffer_Lng[index];
    // Avança o índice do buffer circular
    index++;
    if (index >= 5) {
      index = 0;  // Retorna ao início do buffer
    }
    // Cálculo da média das 5 últimas amostras (multiplicação é mais rápida que divisão)
    lat1 = somaLat * 0.2;
    lng1 = somaLng * 0.2;
}
```

Fonte: (O autor, 2026)

O funcionamento do filtro de média móvel implementado no código ocorre da seguinte forma. Inicialmente, são declaradas as variáveis responsáveis por armazenar a soma acumulada das últimas amostras de latitude e longitude, denominadas static double somaLat e somaLng. Essas variáveis permitem o cálculo da média de maneira mais eficiente, pois evitam a necessidade de recalcular a soma completa das amostras a cada nova leitura. O uso do modificador static é fundamental para garantir que os valores armazenados não sejam reinicializados a cada execução da função loop(), preservando o histórico necessário para o funcionamento do filtro.

Os vetores buffer_Lat[5] e buffer_Lng[5] atuam como *buffers* circulares, sendo responsáveis por armazenar as últimas cinco amostras válidas de latitude e longitude obtidas pelo receptor GNSS. Cada posição desses vetores corresponde a uma leitura

consecutiva fornecida pelo módulo. À medida que novas amostras são inseridas, os valores mais antigos são automaticamente substituídos, mantendo sempre um conjunto atualizado das medições mais recentes.

As variáveis NewSample_Lat e NewSample_Lng armazenam temporariamente as novas amostras de latitude e longitude obtidas diretamente do módulo GNSS antes de sua inserção no *buffer* circular.

A variável static int index é responsável por controlar o processo de inicialização e o armazenamento das primeiras amostras de dados fornecidas pelo módulo NEO-M8N, não devendo ser confundida com o índice do vetor utilizado nos laços de repetição, representado pela variável j. Para esse controle, foi adotado o valor inicial 200, utilizado apenas como um marcador de inicialização, indicando que o *buffer* ainda não foi completamente preenchido com valores válidos de posição.

Esse mecanismo evita que dados inválidos ou incompletos sejam utilizados no cálculo da média durante as primeiras execuções do sistema. Caso valores nulos, como 0, fossem inicialmente inseridos no *buffer*, o resultado da média móvel tenderia a valores próximos de zero, comprometendo a coerência das coordenadas calculadas. Dessa forma, o *buffer* é inicializado com valores próximos à posição real obtida no momento do primeiro fix do GNSS.

Após a etapa de inicialização e organização do *buffer*, a variável index passa a assumir valores entre 0 e 4, controlando o funcionamento do *buffer* circular responsável pelo armazenamento das cinco amostras mais recentes. O valor 200 volta a ser atribuído apenas quando o sistema é reiniciado, como ocorre no desligamento e religamento do módulo, garantindo que o processo de inicialização seja executado novamente e evitando o uso de valores inconsistentes no início da operação.

A condição if (index == 200 && gps.location.isValid()) verifica simultaneamente se o sistema ainda se encontra na fase de inicialização do filtro e se já existe uma posição GNSS válida disponível. Somente após a obtenção de um fix confiável o processo de inicialização é realizado. Nesse momento, por meio de um laço for, o *buffer* é preenchido com o mesmo valor inicial de latitude e longitude, enquanto as somas acumuladas são calculadas corretamente desde o início. Essa estratégia evita

transientes indesejados no filtro e garante que a primeira média calculada represente uma posição coerente.

Após essa etapa, o índice é ajustado para zero, passando a apontar para a primeira posição do *buffer* e habilitando o funcionamento normal do filtro. Caso o receptor GNSS ainda não possua uma posição válida, o código interrompe temporariamente o processamento, impedindo o uso de dados inconsistentes.

Na etapa seguinte, novas coordenadas são capturadas nas variáveis NewSample_Lat e NewSample_Lng. Antes de inserir a nova amostra no *buffer*, o algoritmo remove da soma acumulada o valor antigo armazenado na posição atual do índice. Em seguida, a nova amostra substitui o valor mais antigo no *buffer* e é adicionada à soma acumulada, mantendo o somatório sempre atualizado sem a necessidade de recalcular todas as amostras.

Após a atualização do *buffer*, o índice é incrementado e o processo continua de forma circular. Por fim, realiza-se o cálculo da média móvel das últimas cinco amostras. Para isso, utiliza-se o fator 0,2 (equivalente a 1/5) em substituição à operação de divisão, com o objetivo de reduzir o custo computacional e aumentar velocidade do cálculo da média

As variáveis lat1 e lng1 representam, portanto, a posição filtrada, sendo utilizadas posteriormente nos cálculos de distância percorrida, tempo de operação e área trabalhada, contribuindo para aumentar a confiabilidade dos resultados obtidos pelo sistema.

3.4 Cálculo da Distância entre Latitudes e Longitudes

O cálculo da distância entre dois pontos geográficos por microcontroladores, considerando a curvatura da Terra, pode ser realizado por meio da fórmula de Haversine. Essa formulação é amplamente empregada em aplicações de navegação e em sistemas de posicionamento global, pois permite estimar a distância real percorrida sobre a superfície terrestre a partir das coordenadas geográficas de latitude e longitude (SINNOTT, 1984).

Embora a Terra seja um corpo aproximadamente esférico e de grandes dimensões, para curtas distâncias pode-se, em determinadas situações, adotar aproximações baseadas em geometria plana. Entretanto, à medida que a distância entre os pontos aumenta, os erros associados a esse modelo tornam-se mais evidentes. Em geral, métodos baseados em aproximações planas passam a apresentar erros significativos para distâncias superiores a aproximadamente 30 km, tornando necessário o emprego de modelos que considerem a geometria esférica da Terra.

A geometria esférica baseia-se na trigonometria esférica, área da matemática que estuda as relações entre ângulos e lados definidos sobre a superfície de uma esfera. Nesse contexto, os chamados círculos máximos (great circles) representam os menores caminhos entre dois pontos sobre a superfície esférica, sendo fundamentais para cálculos de navegação. A fórmula de Haversine deriva desses princípios e pode ser interpretada como uma reformulação da lei esférica dos cossenos, apresentando maior estabilidade numérica, especialmente para ângulos e distâncias pequenas.

Historicamente, a função haversine foi amplamente utilizada na navegação marítima e aérea, pois sua formulação facilitava a realização de cálculos manuais em uma época anterior ao uso disseminado de calculadoras e computadores digitais. O emprego de funções trigonométricas específicas permitia simplificar operações matemáticas repetitivas, tornando os cálculos mais eficientes e confiáveis em aplicações práticas (WEISSTEIN, 2023).

Dessa forma, conforme apresentado por Weisstein (2023), a fórmula de Haversine pode ser expressa conforme apresentado a seguir. Considere dois pontos na superfície terrestre:

- Ponto 1: latitude φ_1, longitude λ_1
- Ponto 2: latitude φ_2, longitude λ_2

As coordenadas devem ser expressas em radianos.

1. Diferenças angulares:

$$\Delta\varphi = \varphi_2 - \varphi_1 \quad (1)$$

$$\Delta\lambda = \lambda_2 - \lambda_1 \tag{2}$$

2. Função haversine:

$$a = \sin^2\left(\frac{\Delta\varphi}{2}\right) + \cos(\varphi_1)\cos(\varphi_2)\sin^2\left(\frac{\Delta\lambda}{2}\right) \tag{3}$$

3. Ângulo central:

$$c = 2 \cdot \arctan 2\left(\sqrt{a}, \sqrt{1-a}\right) \tag{4}$$

4. Distância:

$$d = R \cdot c \tag{5}$$

onde:

- d representa a distância entre os dois pontos;

- R corresponde ao raio médio da Terra, aproximadamente 6.371 km ou 6.371.000 m.

Dessa forma, consolidando os conceitos de geometria esférica apresentados, utiliza-se a Fórmula de Haversine para a determinação da distância ortodrômica entre dois pontos na superfície terrestre:

$$d = 2R \cdot \arcsin\left(\sqrt{\sin^2\left(\frac{\Delta\varphi}{2}\right) + \cos(\varphi_1)\cos(\varphi_2)\sin^2\left(\frac{\Delta\lambda}{2}\right)}\right) \tag{6}$$

Diante a fórmula necessária para o cálculo entre as distâncias de dois pontos de latitude, por exemplo, é importante salientar que no desenvolvimento de sistemas embarcados, a otimização de recursos computacionais é uma premissa fundamental. Nesse contexto, a biblioteca TinyGPSPlus já se integra uma implementação nativa e otimizada deste cálculo, o qual optou-se pela utilização do método distanceBetween() para obter a distância entre dois pontos. Essa abordagem evita a redundância de

código e assegura uma gestão eficiente da memória do ESP32, descartando a necessidade de declarar funções trigonométricas complexas manualmente.

A Figura 14 apresenta o trecho do *firmware* responsável pela execução deste cálculo:

Figura 14 - Distância entre dois pontos utilizando "distanceBetween"

```
if((millis()-lastMillis > 3000UL) && gps.location.isValid() && initial==1){

 if(lat2 == 0.0 && lng2 ==0.0)
 {
   lat2 = lat1;
   lng2 = lng1;
   return;
 }

 else{

     distancia = TinyGPSPlus::distanceBetween(
     lat1,
     lng1,
     lat2,
     lng2
     );

     course = TinyGPSPlus::courseTo(
     lat1,
     lng1,
     lat2,
     lng2
     );

     courseCard = TinyGPSPlus::cardinal(course);
```

Fonte: (O autor, 2026)

3.5 Armazenamento de Dados

Observa-se que tanto o módulo GNSS NEO-M8N quanto o microcontrolador ESP32 dispõem de memória *Flash* para armazenamento de dados. Entretanto, optou-se pela utilização da memória *Flash* do ESP32, uma vez que ela permite o armazenamento local das informações adquiridas e a posterior transferência desses dados por meio da interface Wi-Fi integrada ao próprio dispositivo, dispensando a necessidade de *hardware* adicional.

O ESP32 possibilita o armazenamento permanente de dados, os quais podem ser empregados para diversas finalidades em sistemas embarcados, como nomes de redes e senhas Wi-Fi, contadores, parâmetros de configuração e outras informações geradas durante a operação do sistema. Nesse contexto, o microcontrolador disponibiliza uma área específica de sua memória *Flash* que pode ser dedicada exclusivamente a essa finalidade. Os dados armazenados nessa memória são preservados mesmo após reinicializações do sistema ou interrupções no fornecimento de energia elétrica, o que o torna interessante para a realização do *download* do arquivo gerado ao término de um dia de trabalho, por exemplo.

Uma limitação inerente à memória *Flash* refere-se ao número finito de ciclos de escrita suportados. Embora os dados possam ser lidos indefinidamente, a maioria dos dispositivos *Flash* é projetada para suportar aproximadamente entre 100.000 a 1.000.000 operações de gravação, o que exige cuidados quanto à frequência de escrita, a fim de garantir maior vida útil da memória.

Nesse contexto, o SPIFFS, acrônimo de *SPI Flash File System* (Sistema de Arquivos em Flash com Interface Periférica Serial), destaca-se como uma solução adequada para o gerenciamento de arquivos na memória Flash do ESP32. Trata-se de um sistema de arquivos leve, amplamente utilizado em microcontroladores da família ESP, que possibilita a criação, leitura, escrita e exclusão de arquivos diretamente na memória Flash interna.

Uma das principais vantagens do SPIFFS é que os arquivos armazenados permanecem intactos mesmo quando um novo *firmware* é gravado na memória Flash do microcontrolador, desde que a partição destinada ao sistema de arquivos não seja alterada. O espaço disponível para armazenamento depende do modelo da placa e da configuração de partições selecionada na IDE do Arduino. Por exemplo, na placa ESP32 Dev Module é possível dispor de aproximadamente 1,5 MB de memória dedicada ao SPIFFS, valor que pode ser verificado e ajustado por meio das opções de partição disponíveis na IDE após a seleção correta da placa.

O SPIFFS é um sistema de arquivos otimizado para memórias Flash do tipo NOR, sendo especialmente indicado para o armazenamento de páginas web, arquivos de configuração (como JSON) e registros de dados (*logs*), dispensando a necessidade de utilização de cartões SD. Apesar de compartilhar a memória Flash interna com o

firmware do dispositivo e não oferecer suporte a diretórios hierárquicos reais, o SPIFFS destaca-se por sua simplicidade, baixo consumo de memória RAM e adequação a aplicações embarcadas de pequeno e médio porte.

No sistema desenvolvido, a gravação na memória Flash não ocorre a cada leitura do módulo GNSS, pois isso reduziria significativamente a vida útil da memória. Em vez disso, os dados operacionais essenciais, como distância total percorrida, tempo de operação e área calculada, são atualizados em intervalos controlados ou ao final do percurso. Dessa forma, reduz-se consideravelmente o número de ciclos de escrita na memória Flash. Considerando que dispositivos Flash do ESP32 suportam tipicamente entre 10^5e 10^6ciclos de gravação, essa estratégia permite estimar uma vida útil de vários anos de operação, mesmo em aplicações agrícolas com uso diário.

Considerando, por exemplo, uma atualização de dados a cada 5 segundos durante uma jornada de trabalho de 10 horas, o sistema realizaria aproximadamente 7.200 operações de gravação por dia. Admitindo uma memória Flash com capacidade de 100.000 ciclos de escrita por setor, a vida útil teórica seria inferior a 14 dias caso todas as gravações ocorressem repetidamente no mesmo setor da memória. Entretanto, como o sistema de arquivos SPIFFS implementa mecanismos de distribuição de desgaste (*wear leveling*), as gravações são distribuídas ao longo de diferentes setores da memória, aumentando significativamente a vida útil efetiva do dispositivo.

Para utilizar as funcionalidades do SPIFFS no ESP32, é necessário inicialmente incluir as bibliotecas correspondentes no código da aplicação. Essas bibliotecas permitem acessar as rotinas de inicialização, leitura, escrita e gerenciamento de arquivos na memória Flash do microcontrolador. A inclusão dessas bibliotecas é apresentada na Figura 15.

Figura 15 - Inclusão das bibliotecas FS.h e SPIFFS.h

```
#include <WiFi.h>                // Biblioteca principal para Wi-Fi no ESP32
#include <WiFiAP.h>           // Biblioteca para configurar o ESP32 como ponto de acesso (Access Point)
#include <TinyGPS++.h>           // Biblioteca para processar dados NMEA do GPS
#include <HardwareSerial.h>      // Permite usar múltiplas portas seriais no ESP32 (Serial1, Serial2)
#include <WebServer.h>         // Inclui a biblioteca WebServer para criar servidor web
#include <FS.h>                  // Biblioteca de File System
#include <SPIFFS.h>              // Biblioteca para usar SPIFFS no ESP32
#include <algorithm>             //Para usar max

#define RX2 16                   // Pino RX para comunicação com GPS (GPIO16)
#define TX2 17                   // Pino TX (não usado neste caso, mas necessário)
#define GPS_BAUD 9600           // Taxa de comunicação do GPS

TinyGPSPlus gps;                 // Instância do GPS
HardwareSerial SerialGPS(2);    // Instância da Serial2 (UART2) para o módulo GPS

// Credenciais do ponto de acesso (Access Point)
const char *ssid = "Maquina01";
const char *password = "Maquina01";

// Nomes arquivos no SPIFFS
const char *historicoGPS = "/historico.csv";
const char *dadosGPS = "/estado.dat";

//Variaveis para calculo de distancia percorrida
static float distanciaTotal = 0.0;
static float tempoOp = 0.0;
static float hectare = 0.0;
float distanciaPercorrida=0.0;
```

Fonte: (O autor, 2026)

Após a inclusão das bibliotecas necessárias, o próximo passo consiste em inicializar o sistema de arquivos SPIFFS no código. Essa inicialização é fundamental, pois permite que o ESP32 reconheça e monte a área da memória Flash destinada ao armazenamento de arquivos. Esse procedimento normalmente é realizado na função setup(), logo no início da execução do programa.

A inicialização do SPIFFS é feita por meio do comando SPIFFS.begin(). Caso o sistema de arquivos ainda não exista ou esteja corrompido, é possível habilitar a formatação automática da memória Flash, garantindo que o sistema seja montado corretamente. Uma vez inicializado, o SPIFFS passa a estar disponível para operações de leitura e escrita, conforme pode ser visto na Figura 16.

Figura 16 - Inicializar arquivos SPIFFS

```
51  void setup() {
52    Serial.begin(115200);        // Inicializa o monitor serial
53  //  Serial.println("Iniciando GPS NEO-M8N...");
54
55    SerialGPS.begin(GPS_BAUD, SERIAL_8N1, RX2, TX2);  // Inicializa Serial2 com configuração padrão
56
57  //  Serial.println("Configurando Ponto de Acesso..."); // Inicializa o ponto de acesso Wi-Fi
58
59  // Inicializa o SPIFFS
60  //  Serial.println("Inicializando SPIFFS...");
61    if (!SPIFFS.begin(true)) {
62      Serial.println("Erro ao montar SPIFFS.");
63      return;
64    }
65
66    if (SPIFFS.exists("/estado.dat")) {
67      lerVariaveis();
68    }
69
70    // Configura o ESP32 como Access Point
71  //  Serial.println("Configurando Ponto de Acesso...");
72    if (!WiFi.softAP(ssid, password)) {
73      log_e("Falha ao criar ponto de acesso");
74      while (true);  // Loop infinito em caso de falha
75    }
```

Fonte: (O autor, 2026)

Com o sistema de arquivos devidamente inicializado, torna-se possível verificar a existência de arquivos previamente armazenados. Essa verificação é especialmente útil em aplicações que necessitam recuperar dados salvos em execuções anteriores, como configurações ou estados do sistema. Caso o arquivo exista, ele pode ser aberto para leitura; caso contrário, um novo arquivo pode ser criado.

Em seguida, o código pode realizar operações de gravação de dados na memória Flash. A gravação normalmente é utilizada para armazenar informações que devem ser preservadas mesmo após o desligamento do dispositivo, como registros de operação, dados de sensores ou parâmetros calculados durante a execução do sistema. Os dados podem ser armazenados em diferentes formatos, sendo o formato texto amplamente utilizado por sua simplicidade e facilidade de interpretação.

Da mesma forma, o SPIFFS permite a leitura dos dados previamente gravados, possibilitando que o sistema recupere informações armazenadas e continue sua operação a partir do último estado salvo. Esse recurso é essencial para garantir continuidade e confiabilidade em sistemas embarcados que operam por longos períodos ou estão sujeitos a interrupções de energia.

Por fim, o sistema de arquivos SPIFFS também disponibiliza funcionalidades para o gerenciamento de arquivos, como a remoção de registros antigos ou a

substituição de arquivos existentes, permitindo manter a memória Flash organizada e evitar o uso excessivo do espaço de armazenamento. Após a conclusão das operações de leitura ou escrita, os arquivos devem ser devidamente fechados, a fim de garantir a integridade dos dados gravados. Esse procedimento pode ser observado na Fig 17.

Figura 17 - Funcionalidades do SPIFFS

```
296 void salvarVariaveis() {
297   static unsigned long saveMillis = 0;
298
299   if (millis() - saveMillis > 60000UL) {
300     saveMillis = millis(); // Atualiza o tempo para o próximo salvamento
301
302     // --- Salvar Distância ---
303     File file = SPIFFS.open("/estado.dat", FILE_WRITE);
304     if (!file) {
305       Serial.println("Erro ao abrir estado.dat");
306     } else {
307       file.println(distanciaTotal, 8);
308       file.println(tempoOp, 8);
309       file.println(hectare, 8);
310       file.close();
311       Serial.println("Estados Salvos!");
312     }
313   }
314 }
315
316
317 void lerVariaveis() {
318
319   // --- Ler Dados ---
320   File file = SPIFFS.open("/estado.dat", FILE_READ);
321   if (!file) {
322     Serial.println("Erro ao abrir estado.dat para leitura!");
323     return;
324   }
325   if(file.available()){
326     distanciaTotal=file.readStringUntil('\n').toFloat();
327     tempoOp=file.readStringUntil('\n').toFloat();
328     hectare=file.readStringUntil('\n').toFloat();
329     file.close();
330   }
331   else {
332     Serial.println("Erro ao abrir estado.dat para leitura.");
333   }
334 }
```

Fonte: (O autor, 2026)

A utilização de formatos distintos de arquivo para exportação e para uso interno no sistema foi adotada de forma intencional, visando atender a diferentes requisitos de armazenamento, processamento e interoperabilidade dos dados.

O formato CSV (Comma-Separated Values) foi empregado para a exportação dos dados, pois se trata de um padrão amplamente difundido e de fácil interpretação por diferentes plataformas e ferramentas computacionais. Arquivos CSV podem ser

abertos e analisados diretamente em *softwares* como Microsoft Excel, LibreOffice Calc, MATLAB, Python, R e sistemas de informação geográfica (SIG), o que facilita a visualização, o pós-processamento e a validação dos dados coletados. Além disso, por ser um formato textual simples e estruturado em colunas, o CSV permite a organização clara das informações de latitude, longitude, tempo, distância e área, tornando-o ideal para compartilhamento e análise externa.

A Figura 18 apresenta um trecho do código responsável pela exportação dos arquivos no formato .csv.

Figura 18 - Exportação do arquivo em .csv

```
453  void handleDownload() {
454    if (!SPIFFS.exists("/historico.csv")) {
455      server.send(404, "text/csv", "Arquivo não encontrado.");
456      return;
457    }
458
459    File file = SPIFFS.open("/historico.csv", FILE_READ);
460    server.streamFile(file, "text/csv");
461    file.close();
462  }
```

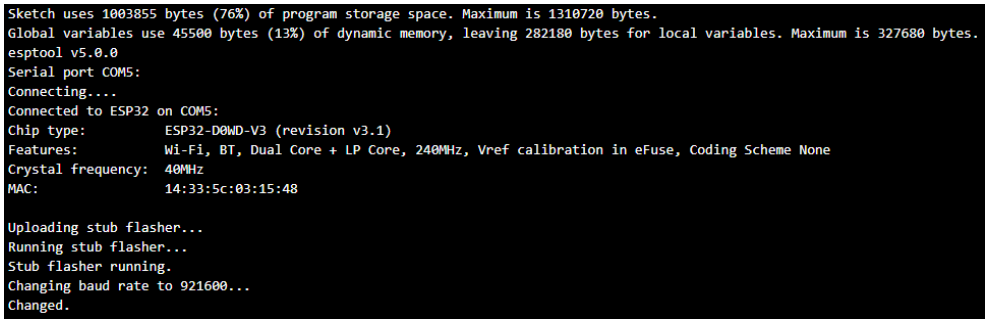
Fonte: (O autor, 2026)

Por outro lado, o formato .dat foi adotado para uso interno do sistema, principalmente para o armazenamento do estado de operação do dispositivo, como distância total percorrida, tempo de funcionamento e área calculada em hectares. Esse formato é utilizado de maneira mais enxuta e controlada, sem a necessidade de cabeçalhos ou estruturas tabulares complexas, o que reduz o volume de dados gravados e minimiza o número de operações de escrita na memória Flash do microcontrolador. Dessa forma, o uso do arquivo .dat contribui para a preservação da vida útil da memória Flash, uma vez que limita a frequência de gravações.

Adicionalmente, o arquivo .dat permite operações de leitura e escrita mais rápidas durante a inicialização do sistema, possibilitando a recuperação eficiente das variáveis essenciais após reinicializações ou eventuais quedas de energia. Assim, enquanto o formato .csv atende ao requisito de portabilidade e análise externa dos dados, o formato .dat mostra-se mais adequado para a persistência interna do estado do sistema, garantindo maior eficiência, robustez e confiabilidade à aplicação.

Destaca-se ainda que o código desenvolvido no ambiente da Arduino IDE para o ESP32, após o processo de compilação, ocupa aproximadamente 1,003 MB de memória de programa, enquanto a memória destinada ao armazenamento do *firmware* é de cerca de 1,31 MB. Dessa forma, aproximadamente 76% da memória destinada ao *firmware* encontra-se utilizada, restando cerca de 24% disponíveis, conforme apresentado na Figura 19.

Figura 19 - Memória de Programa utilizada para o código fornecido ao ESP32



Fonte: (O autor, 2026)

Ressalta-se que a memória Flash total do ESP32 é tipicamente de 4 MB, sendo organizada em diferentes partições destinadas ao *firmware*, ao sistema e ao armazenamento de dados. No sistema desenvolvido, o espaço remanescente para o *firmware* é de aproximadamente 0,31 MB, permitindo eventuais expansões futuras do código.

Além disso, cerca de 1,5 MB da memória Flash são reservados para o sistema de arquivos SPIFFS, responsável pelo armazenamento dos dados operacionais gerados pelo dispositivo, os quais são registrados em arquivos no formato .csv para posterior *download* e processamento em computador.

3.6 Transferência de Dados

A extração dos dados gerados pelo sistema foi concebida para ocorrer de forma totalmente sem fio, eliminando a necessidade de conexões físicas ou da remoção de cartões de memória em campo. Essa funcionalidade é fundamental para a viabilidade

do projeto, especialmente em regiões rurais remotas, onde a cobertura de telefonia móvel é limitada ou inexistente. Nesse contexto, optou-se pela utilização do ESP32 devido à sua capacidade nativa de comunicação Wi-Fi embarcada, característica já mencionada ao longo deste trabalho.

Para possibilitar a visualização e o *download* dos dados coletados, foi implementada uma interface de comunicação na qual o ESP32 opera simultaneamente como ponto de acesso (*Access Point*) e servidor Web embarcado. Dessa forma, o próprio dispositivo cria uma rede Wi-Fi local, permitindo que *smartphones*, *tablets* ou *notebooks* se conectem diretamente ao sistema, sem a necessidade de roteadores ou infraestrutura externa.

Após estabelecer a conexão com a rede criada pelo ESP32, o usuário pode acessar, por meio de um navegador web, uma página hospedada no próprio microcontrolador. Nessa interface, são disponibilizadas as informações coletadas pelo módulo GPS, bem como a opção de *download* dos arquivos armazenados no sistema de arquivos interno.

A Figura 20 apresenta as bibliotecas empregadas para viabilizar essa funcionalidade.

Figura 20 - Bibliotecas para Transferência de Dados

```
#include <WiFi.h>                 // Biblioteca principal para Wi-Fi no ESP32
#include <WiFiAP.h>               // Biblioteca para configurar o ESP32 como ponto de acesso (Access Point)
#include <WebServer.h>            // Inclui a biblioteca WebServer para criar servidor web
#include <TinyGPS++.h>            // Biblioteca para processar dados NMEA do GPS
#include <HardwareSerial.h>       // Permite usar múltiplas portas seriais no ESP32 (Serial1, Serial2)
#include <FS.h>                   // Biblioteca de File System
#include <SPIFFS.h>               // Biblioteca para usar SPIFFS no ESP32
#include <algorithm>              //Para usar max
```

Fonte: (O autor, 2026)

3.6.1 Configuração do Ponto de Acesso (*Access Point*)

O módulo ESP32 disponibiliza uma API Wi-Fi compatível com o protocolo IEEE 802.11 b/g/n, oferecendo suporte a diferentes modos de operação. Entre eles, destacam-se: o modo estação (STA), no qual o ESP32 atua como cliente e se conecta a um ponto de acesso existente; o modo *Access Point* , também denominado *Soft-AP*,

em que o próprio ESP32 cria uma rede Wi-Fi e permite a conexão de outros dispositivos; além do suporte a diferentes mecanismos de segurança, como WPA2 e WPA3, e funcionalidades de varredura de redes disponíveis.

No modo AP, o ESP32 é configurado como um Ponto de Acesso capaz de receber conexões de dispositivos clientes (estações), fornecendo uma rede Wi-Fi própria. Nesse modo de operação, também é possível hospedar um servidor HTTP ou HTTPS embarcado no próprio microcontrolador, o que o torna particularmente adequado para aplicações em campo. Considerando que o objetivo do projeto é permitir o *download* dos arquivos de forma totalmente sem fio, essa configuração mostra-se ideal, pois elimina a necessidade de infraestrutura externa de rede.

Com o intuito de garantir conectividade em qualquer localidade, o ESP32 foi configurado para operar no modo *Access Point* por meio das bibliotecas <WiFi.h> e <WiFiAP.h>. Dessa forma, o microcontrolador cria uma rede Wi-Fi local privativa, permitindo que o operador conecte dispositivos como *smartphones*, *tablets* ou *notebooks* diretamente ao sistema, sem depender de roteadores ou acesso à internet. As credenciais de acesso (SSID e senha) são definidas estaticamente no *firmware*, assegurando controle e segurança da conexão estabelecida.

É apresentado pela Figura 21 o trecho do código empregado para viabilizar a configuração do ponto de acesso.

Figura 21 - Configuração do Ponto de Acesso (AP)

```
// Configura o ESP32 como Access Point
//  Serial.println("Configurando Ponto de Acesso...");
  if (!WiFi.softAP(ssid, password)) {   //SSID e Password já definidos
    log_e("Falha ao criar ponto de acesso");
    while (true);  // Loop infinito em caso de falha
  }

 // Obtém e imprime o endereço IP do ponto de acesso
  IPAddress myIP = WiFi.softAPIP();
  Serial.print("Endereço IP do AP: ");
  Serial.println(myIP);

  server.on("/", handleRoot);  // Configura rota "/" no servidor web
  server.on("/reset", handleReset); // Configura rota para resetar dados
  server.on("/resetAll", handleResetAll); // Configura rota para resetar dados
  server.on("/DownloadMaquina01.csv", handleDownload); // Configura rota para fazer download de dados

  // Inicia o servidor web
  server.begin();
//  Serial.println("Servidor iniciado!");

}
```

Fonte: (O autor, 2026)

O trecho de código apresentado é responsável por configurar o ESP32 no modo *Access Point* e exibir o endereço IP da rede criada.

A função “WiFi.softAP(ssid, password)” inicializa o ESP32 no modo *Soft-AP*, permitindo que o próprio microcontrolador crie uma rede Wi-Fi local. Nesse contexto, o “ssid” corresponde ao nome da rede Wi-Fi que será disponibilizada, enquanto o “password” define a senha de acesso (com no mínimo 8 caracteres para autenticação WPA2).

A função retorna true caso o ponto de acesso seja criado com sucesso e false em caso de falha.

Em caso de erro na inicialização da rede, o comando “log_e()” registra uma mensagem de erro no console (nível *error*), permitindo o diagnóstico do problema. Em seguida, a instrução “while (true);” mantém o sistema em *loop* infinito, interrompendo a execução do programa. Essa estratégia é adotada para evitar que o sistema continue operando sem conectividade Wi-Fi, uma vez que a comunicação sem fio é uma funcionalidade essencial para a extração dos dados no projeto.

Após a criação bem-sucedida do ponto de acesso, a função “WiFi.softAPIP()” retorna o endereço IP atribuído ao ESP32 na rede criada. Por padrão, esse endereço

costuma ser 192.168.4.1, que será utilizado pelo operador para acessar o servidor web embarcado por meio de um navegador.

Por fim, o endereço IP é exibido no Monitor Serial, permitindo que o usuário identifique corretamente o endereço necessário para acessar a interface web do sistema.

3.6.1 Servidor Web Embarcado

Após a conexão de um dispositivo móvel, como notebook ou smartphone, à rede criada pelo ESP32 no modo *Access Point*, o usuário passa a ter acesso à interface web do sistema.

A interface de usuário e a disponibilização dos arquivos foram implementadas por meio da biblioteca <WebServer.h>, que permite ao ESP32 atuar como um servidor HTTP embarcado, processando requisições enviadas pelo navegador. O fluxo de dados ocorre de forma integrada ao sistema de arquivos interno (SPIFFS), vistos anteriormente, onde os relatórios gerados são armazenados.

Por meio do servidor web, o sistema disponibiliza um endereço IP local, geralmente 192.168.4.1, que pode ser digitado no navegador. Ao acessar esse endereço, o usuário visualiza uma página HTML hospedada no próprio ESP32, a partir da qual é possível realizar o *download* dos relatórios de produtividade diretamente para o dispositivo conectado.

De forma simplificada, um servidor web pode ser entendido como um sistema responsável por armazenar e fornecer páginas da web. Ele mantém arquivos como documentos HTML e recursos associados (imagens, folhas de estilo CSS, fontes e outros elementos). Quando um usuário realiza uma solicitação por meio do navegador, o servidor envia esses arquivos como resposta.

Esse processo ocorre utilizando o Protocolo de Transferência de Hipertexto (HTTP), que define como as informações são requisitadas e transmitidas na rede. Assim, ao acessar o endereço IP do ESP32, o navegador envia uma requisição HTTP

ao microcontrolador, que responde disponibilizando a página e os arquivos solicitados, como pode ser visto no trecho da Figura 22.

Figura 22 - Criação do Servidor Web

```
// Obtém e imprime o endereço IP do ponto de acesso
 IPAddress myIP = WiFi.softAPIP();
 Serial.print("Endereço IP do AP: ");
 Serial.println(myIP);

 server.on("/", handleRoot);  // Configura rota "/" no servidor web
 server.on("/reset", handleReset); // Configura rota para resetar dados
 server.on("/resetAll", handleResetAll); // Configura rota para resetar dados
 server.on("/DownloadMaquina01.csv", handleDownload); // Configura rota para fazer download de dados

 // Inicia o servidor web
 server.begin();
//  Serial.println("Servidor iniciado!");

}
```

Fonte: (O autor, 2026)

Esse trecho de código é responsável por configurar as rotas do servidor web embarcado no ESP32 e iniciar o funcionamento do servidor HTTP após o endereço IP ser acessado por um dispositivo cliente. Cada rota define como o sistema deve responder quando o usuário acessa um determinado endereço por meio do navegador.

Essas rotas funcionam como *endpoints*, permitindo o controle do ESP32 diretamente pela URL. Na prática, está sendo implementado um servidor web local, com características semelhantes a um servidor REST, porém totalmente independente de conexão com a internet.

A função server.on() recebe dois parâmetros, o qual o primeiro corresponde ao caminho da URL (*endpoint*) requisitado pelo cliente. Por outro lado, o segundo é a função de tratamento (*handler*), que será executada automaticamente quando aquele endpoint for acessado.

Essa função de tratamento é definida no próprio código-fonte e é responsável por gerar a resposta HTTP enviada ao navegador, normalmente na forma de uma página HTML ou de um arquivo para *download*.

A rota raiz do servidor é definida pelo caractere /, representando a página principal do sistema. Ela é acessada quando o usuário digita apenas o IP do ESP32 no navegador, por exemplo:

http://192.168.4.1/

A função handleRoot() geralmente é responsável por: Enviar a página HTML da interface do sistema; exibir informações gerais de operação; e disponibilizar botões para *download* de arquivos ou reset de dados, conforme pode ser visto na Figura 23.

Figura 23 - Função handleRoot configurando dados e comandos

```
// Função chamada quando acessar a página "/"
void handleRoot() {

  // Cria conteúdo HTML em uma string
  String html = "<!DOCTYPE html>";
  html += "<html lang='pt-br'>";
  html += "<head>";
  html += "<meta charset='UTF-8'>";
  html += "<meta name='viewport' content='width=device-width, initial-scale=1.0'>";
  html += "<meta http-equiv='refresh' content='5'>";
  html += "<title>Máquina 01 - Dados GPS</title>";
  html += "<style>";
  html += "body { font-family: Arial; background-color: #f0f2f5; padding: 20px; }";
  html += "h1 { text-align: center; color: #333; }";
  html += "div.container { max-width: 600px; margin: auto; background: white; padding: 20px; border-radius: 10px; box-shadow: 0 0 10px rgba(0,0,0,0.1); }";
  html += "p { font-size: 1.1em; color: #555; }";
  html += "strong { color: #000; }";
  html += "button, a.button { padding: 10px 20px; font-size: 16px; text-decoration: none; border: none; border-radius: 5px; display: inline-block; margin: 5px 0; }";
  html += "button { background-color: #f0ad4e; color: white; }";
  html += "a.button { background-color: #5bc0de; color: white; }";
  html += ".btn-reset-all { background-color: #c9302c; color: white; }";   // estilo do reset all
  html += ".spaced { margin-top: 30px; display: block; }";              // cria espaço extra
  html += "</style>";
  html += "</head><body><div class='container'>";
  html += "<h1>Máquina 01 - Dados GPS NEO-M8N</h1>";

  // Dados GPS
  html += "<p><strong>Satélites:</strong> " + String(gps.satellites.value()) + "</p>";
  html += "<p><strong>Precisão:</strong> " + String(gps.hdop.hdop()) + "</p>";
  html += "<p><strong>Velocidade (km/h):</strong> " + String(gps.speed.kmph(), 2) + "</p>";
  html += "<p><strong>Data:</strong> " + String(gps.date.day()) + "/" + String(gps.date.month()) + "/" + String(gps.date.year()) + "</p>";
  html += "<p><strong>Hora (UTC-3):</strong> " + String(gps.time.hour() - 3) + ":" + String(gps.time.minute()) + ":" + String(gps.time.second()) + "</p>";
  html += "<p><strong>Curso:</strong> " + String(courseCard) + "</p>";
  html += "<p><strong>Distância percorrida (km):</strong> " + String(distanciaPercorrida, 3) + "</p>";
  html += "<hr>";  // Ponto de separação visual antes dos dados "a partir de distancia percorrida"

  html += "<p><strong>Hectare:</strong> " + String(hectare) + "</p>";
  html += "<p><strong>Distância Total (km):</strong> " + String(distanciaTotal, 3) + "</p>";
  html += "<p><strong>Tempo decorrido (min):</strong> " + String(tempoOp) + "</p>";

  // Botão de reset parcial
  html += "<form action='/reset' method='get'>";
  html +=   "<button type='submit'>Resetar Dados</button>";
  html += "</form>";
```

Fonte: (O autor, 2026)

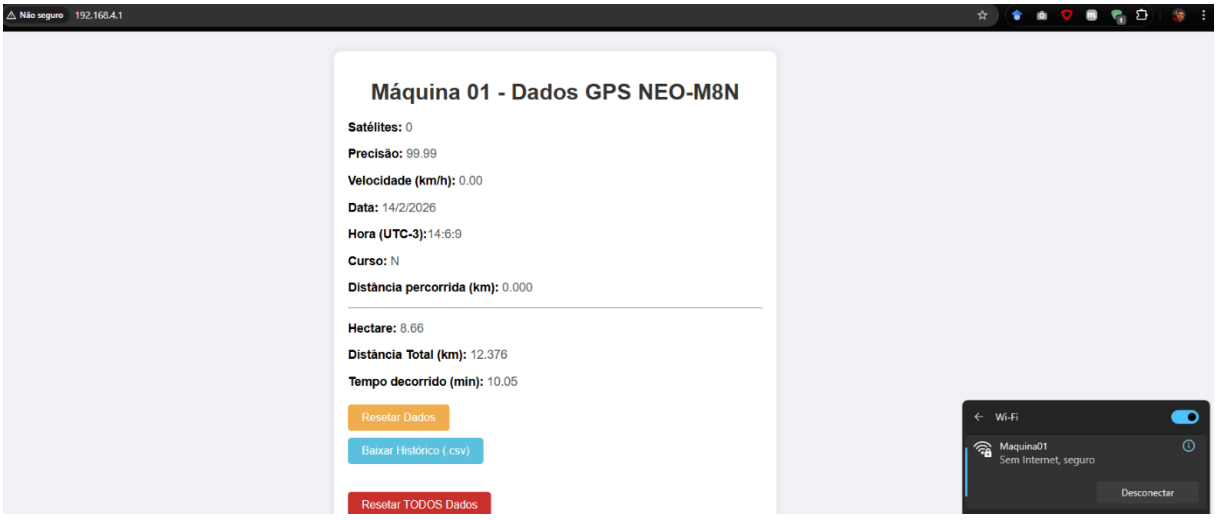
A rota “server.on("/reset", handleReset)” cria um endpoint destinado à redefinição de informações do sistema. Ao acessar http://192.168.4.1/reset, a função handleReset() é executada. Essa funcionalidade é normalmente utilizada para: Zerar variáveis relacionadas ao percurso; reiniciar contadores; e limpar dados temporários sem apagar arquivos permanentes armazenados no sistema.

A rota “server.on("/DownloadMaquina01.csv", handleDownload);” é responsável por definir o endpoint que realizará o download do arquivo CSV armazenado no sistema de arquivos interno. Ao acessar http://192.168.4.1/DownloadMaquina01.csv, o navegador inicia automaticamente o *download* do arquivo. A função handleDownload() é responsável por localizar o arquivo, configurar os cabeçalhos HTTP adequados e transmitir os dados ao cliente.

Por fim, a instrução server.begin() inicia efetivamente o servidor HTTP no ESP32. A partir desse momento, o microcontrolador passa a: Escutar requisições HTTP; Processar as rotas configuradas; e permitir a interação do usuário por meio do navegador.

Dessa forma, o ESP32 opera como um servidor web autônomo, viabilizando o controle do sistema e a extração de dados de maneira local, segura e sem dependência de infraestrutura externa, conforme mostra a Figura 24.

Figura 24 - Interface Web do sistema desenvolvido



Fonte: (O autor, 2026)

3.7 Processamento em Python

Após as etapas de aquisição, filtragem, transferência e *download* dos dados, torna-se necessário realizar o processamento das informações coletadas, a fim de

extrair indicadores relevantes ao usuário. Entre os principais parâmetros analisados estão a distância total percorrida pelo maquinário, a área efetivamente trabalhada (em hectares) e o tempo total de operação diária.

Essa etapa é desenvolvida por meio de scripts implementados na linguagem Python, responsáveis pela leitura do arquivo no formato CSV, organização dos dados em estruturas apropriadas (como *dataframes*), processamento das variáveis e geração dos resultados consolidados. O script realiza o cálculo da distância total percorrida, do tempo de trabalho e da área coberta, além de possibilitar a geração de um mapa interativo para visualização do trajeto realizado.

A distância percorrida é obtida pela soma das distâncias entre pares consecutivos de coordenadas geográficas (latitude e longitude), empregando fórmulas de cálculo geodésico adequadas à curvatura da Terra, como a fórmula de Haversine.

O tempo total de operação é determinado a partir da diferença entre os instantes inicial e final registrados pelo sistema GNSS, considerando apenas os períodos em que o equipamento esteve efetivamente em funcionamento.

Para o cálculo da área trabalhada, os pontos geográficos coletados são tratados como vértices de um polígono georreferenciado. A partir desses vértices, aplica-se um método matemático apropriado para determinar a área delimitada pelo trajeto, permitindo a estimativa da superfície coberta pelo implemento agrícola. Esse cálculo é diretamente relacionado à largura de trabalho da máquina e ao deslocamento registrado ao longo do percurso.

No que se refere ao cálculo das horas trabalhadas, realiza-se a identificação do período correspondente ao dia de operação e a soma dos intervalos em que o dispositivo esteve ativo, resultando no total de horas produtivas registradas.

Dessa forma, o processamento computacional converte as coordenadas geográficas brutas em informações estratégicas e interpretáveis, possibilitando ao usuário uma análise objetiva e quantitativa do desempenho operacional do maquinário em campo.

A partir desse tratamento prévio dos dados, apresenta-se a seguir um trecho do código desenvolvido em linguagem Python, utilizando a plataforma *Visual Studio Code* (VS Code), responsável pela geração de um mapa interativo construído com

base nos dados já processados. Esse mapa permite a visualização espacial do trajeto percorrido, facilitando a interpretação dos resultados e a validação das métricas calculadas.

A Figura 25 ilustra a implementação do algoritmo desenvolvido no ambiente de desenvolvimento integrado (IDE) *Visual Studio Code*. O script, escrito em linguagem Python, utiliza a biblioteca Folium para o processamento das coordenadas coletadas, resultando na geração de uma interface cartográfica interativa para a visualização das rotas agrícolas.

Figura 25 - Trecho em Python para Criar Mapa e Tratar Dados

```python
import pandas as pd
import folium
from folium.plugins import Draw, MeasureControl

# === 1. Ler o CSV (delimitador ";")
df = pd.read_csv(r"C:\Users\victo\OneDrive\Área de Trabalho\TCC\1-TESTES_CARRO\TESTE15\DownloadMaquina01 (5).csv", sep=";")

# === 2. Segmentar os trechos (0,0 indica início/fim de trecho)
segments = []
current = []
for _, row in df.iterrows():
    if row["Lat"] == 0.0 and row["Lon"] == 0.0:
        if current:
            segments.append(pd.DataFrame(current))
            current = []
    else:
        current.append(row)
if current:
    segments.append(pd.DataFrame(current))

# === 3. Criar o mapa centralizado nos pontos reais
all_real = pd.concat(segments, ignore_index=True)
center = [all_real["Lat"].mean(), all_real["Lon"].mean()]
m = folium.Map(location=center, zoom_start=16,
               tiles="https://server.arcgisonline.com/ArcGIS/rest/services/World_Imagery/MapServer/tile/{z}/{y}/{x}",
               name = "Satelite",
               attr='Esri World Imagery', control_scale=True)

# Adiciona outras opções de mapa
folium.TileLayer('OpenStreetMap', name='Padrão').add_to(m)
folium.TileLayer('CartoDB positron', name='Claro').add_to(m)
folium.LayerControl().add_to(m)


# === 4. Adicionar ferramenta de medida
m.add_child(MeasureControl(position='topright', primary_length_unit='meters', primary_area_unit='hectares'))

m.add_child(folium.LatLngPopup())

# === 5. Adicionar ferramenta de desenho
draw = Draw(
    export=False,
    position="topleft",
    draw_options={
        'polyline': True,
        'polygon': True,
        'circle': False,
        'rectangle': True,
        'marker': True,
        'circlemarker': False
```

Fonte: (O autor, 2026)

O trecho do código apresenta inicialmente o arquivo CSV, que é carregado por meio da função pd.read_csv(), considerando o delimitador ponto e vírgula (“;”), conforme definido na etapa de exportação dos dados pelo sistema embarcado. O conteúdo é armazenado em um *DataFrame*, estrutura que permite o tratamento eficiente das informações georreferenciadas, como latitude, longitude, data, hora, velocidade, distância total e área trabalhada.

Na sequência, é realizada a segmentação dos trajetos. O critério adotado considera que registros contendo latitude e longitude iguais a 0,0 indicam o início ou o término de um trecho operacional. Assim, o algoritmo percorre todas as linhas do *DataFrame* e agrupa os pontos válidos em listas temporárias, que posteriormente são convertidas em novos *DataFrames*, formando uma lista de segmentos independentes. Essa abordagem permite representar separadamente diferentes períodos de operação ou deslocamentos distintos do maquinário.

Para a criação do mapa, todos os segmentos válidos são concatenados a fim de determinar o ponto médio das coordenadas, utilizado como centro inicial da visualização. O mapa é então instanciado por meio da classe folium.Map(), com nível de zoom ajustado para 16, adequado à visualização em escala de campo agrícola. Como camada principal, foi utilizada a base cartográfica “Esri World Imagery”, que fornece imagens de satélite de alta resolução. Além disso, foram adicionadas camadas alternativas, como OpenStreetMap e CartoDB Positron, permitindo ao usuário alternar entre diferentes estilos de visualização por meio do controle de camadas (*LayerControl*).

Com o intuito de ampliar a interatividade, foi incorporada a ferramenta de medição (*MeasureControl*), configurada para exibir comprimentos em metros e áreas em hectares. Também foi adicionada a funcionalidade de exibição de coordenadas geográficas ao clicar no mapa (*LatLngPopup*), facilitando análises complementares.

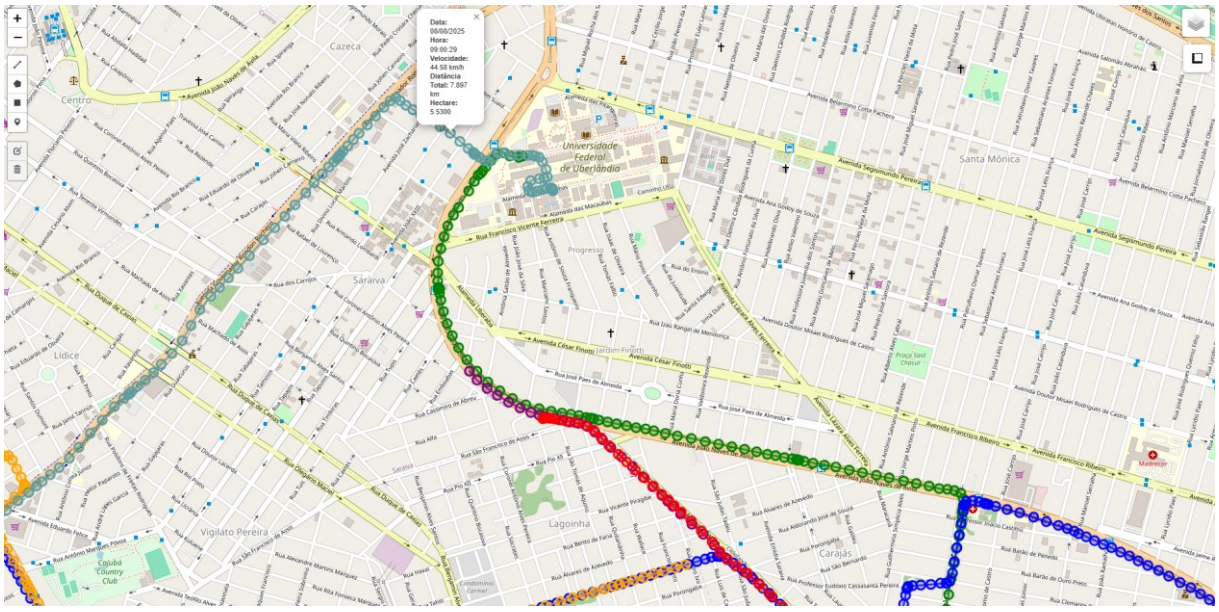
Adicionalmente, foi implementada a ferramenta de desenho (*Draw*), permitindo que o usuário trace linhas, polígonos, retângulos e marcadores diretamente sobre o mapa. Essa funcionalidade possibilita a delimitação manual de áreas, comparação entre trajetos calculados automaticamente e regiões desenhadas pelo operador, bem como análises exploratórias adicionais.

Os trechos segmentados são então representados graficamente por meio de objetos PolyLine, cada qual com uma cor distinta para facilitar a diferenciação visual. Para cada ponto pertencente ao trecho, é adicionado um marcador circular (*CircleMarker*) contendo uma janela informativa (*popup*). Essa janela apresenta dados relevantes, como data, hora, velocidade instantânea, distância total acumulada e área trabalhada até aquele ponto específico.

Por fim, o mapa interativo é exportado em formato HTML por meio da função m.save("mapa.html"). O arquivo gerado pode ser aberto em qualquer navegador, permitindo a navegação, inspeção dos dados e utilização das ferramentas interativas sem a necessidade de ambiente Python ativo.

Dessa forma, o processamento computacional transforma os dados brutos coletados pelo sistema embarcado em uma representação geoespacial intuitiva e interativa, possibilitando ao usuário uma análise clara e objetiva do desempenho operacional do maquinário em campo, como é possível ver na Figura 26.

Figura 26 - Multipercursos realizados com o NEO-M8N e o ESP32



Fonte: (O autor, 2026)

3.8 Considerações Finais

Diante o exposto ao longo deste capítulo, verifica-se que a metodologia adotada contempla de forma estruturada todas as etapas necessárias para o desenvolvimento do Sistema de Monitoramento de Área de Baixo Custo, desde a aquisição do sinal GNSS até a geração de relatórios e mapas interativos.

Inicialmente, a integração entre o módulo GNSS NEO-M8N e o microcontrolador ESP32 possibilita a captura confiável dos dados de posicionamento por meio do protocolo NMEA, utilizando comunicação serial UART. Em seguida, são aplicadas técnicas de validação e filtragem, como critérios de qualidade do sinal, limiar mínimo baseado na velocidade e filtro de média móvel, que tiveram o objetivo de reduzir ruídos e aumentar a confiabilidade das coordenadas utilizadas nos cálculos.

Posteriormente, os dados são armazenados na memória Flash do ESP32 por meio do sistema de arquivos SPIFFS, garantindo persistência das informações e viabilizando sua extração por meio de um servidor web embarcado operando em modo *Access Point*. Essa solução elimina a dependência de infraestrutura externa de rede, tornando o sistema adequado para aplicações em ambientes rurais.

Por fim, o processamento em ambiente computacional, realizado em Python, permite transformar as coordenadas brutas em indicadores operacionais relevantes, tais como distância percorrida, área trabalhada e tempo efetivo de operação. A geração de mapas interativos consolida os resultados de forma visual e acessível, ampliando o potencial de análise e validação dos dados obtidos.

Dessa forma, a metodologia proposta demonstra-se coerente, tecnicamente fundamentada e alinhada aos objetivos estabelecidos neste trabalho, evidenciando a viabilidade da implementação de um sistema de monitoramento agrícola de baixo custo, robusto e aplicável à realidade do campo.

4 RESULTADOS

Concluída a etapa de desenvolvimento, abrangendo desde a definição da arquitetura de aquisição de sinais até a implementação da visualização em mapas interativos, torna-se indispensável proceder à validação experimental do sistema, por meio da realização de testes controlados. Essa fase tem por finalidade verificar a robustez dos algoritmos de filtragem empregados, bem como evidenciar as contribuições da linguagem Python na organização, processamento e análise dinâmica dos dados coletados.

Em consonância com o objetivo central deste trabalho, que consiste no desenvolvimento de um sistema de monitoramento de área de baixo custo baseado no microcontrolador Espressif ESP32 e no módulo GNSS NEO-M8N, foi conduzida uma análise comparativa de desempenho. Tal procedimento envolveu o confronto entre as distâncias estimadas pelo protótipo e aquelas obtidas por meio do *software* Google Earth Pro, além da comparação com os valores registrados pelo odômetro veicular. Essa abordagem constituiu a principal métrica de validação, permitindo aferir a precisão, a confiabilidade e a aplicabilidade do dispositivo em condições reais de operação.

4.1 Validação da Comunicação Serial sem filtros

Inicialmente, faz-se necessária a aferição do módulo GNSS NEO-M8N, com o objetivo de verificar se o desempenho observado está em conformidade com as especificações fornecidas pela u-blox, especialmente no que se refere à taxa de atualização, que pode variar entre 5 Hz e 10 Hz. Essa verificação é fundamental para que tais parâmetros sejam posteriormente explorados em favor da estratégia de filtragem de dados, visando aumentar a confiabilidade das informações processadas pelo sistema.

Antes da obtenção dos resultados, propõe-se, portanto, a validação progressiva do projeto, conduzida de forma sistemática e etapa por etapa, até que se atinjam dados consistentes e adequados à aplicação proposta. Essa abordagem justifica-se pelo fato de que, na ausência de filtros ou critérios de condicionamento e parada, os

dados seriam transmitidos continuamente, mesmo quando desnecessário, comprometendo a eficiência do processamento e o desempenho do sistema.

Assim, nesta fase preliminar, são avaliados três parâmetros principais: a taxa de atualização das mensagens GNSS, o erro de leitura e o tempo necessário para a fixação do sinal (*Time to First Fix* – TTFF), com o objetivo de validar a comunicação serial e assegurar a integridade das informações recebidas.

A avaliação de desempenho do sistema foi conduzida sob condições de céu predominantemente encoberto, simulando cenários operacionais reais em que a cobertura de nuvens pode elevar a atenuação dos sinais GNSS. A escolha desse ambiente justifica-se pela necessidade de submeter o protótipo a situações de menor relação sinal-ruído (SNR), garantindo um maior rigor à validação experimental e à confiabilidade dos indicadores obtidos. As condições atmosféricas vigentes durante a campanha de testes estão documentadas na Figura 27, que ilustra o cenário de baixa visibilidade celeste descrito.

Figura 27 - Validação do NEO-M8N em clima nublado

Fonte: (O autor, 2026)

4.1.1 Taxa de Atualização e Tempo de Fixação do Sinal

Para a verificação da taxa de atualização, será utilizado um código simplificado baseado em um dos exemplos disponibilizados pela biblioteca TinyGPS++. A partir da medição do intervalo de tempo entre mensagens consecutivas, expresso em milissegundos, será possível determinar experimentalmente a frequência de atualização em Hertz (Hz). A Figura 28 ilustra a captura do monitor serial durante o ensaio, evidenciando os intervalos medidos e a respectiva taxa de atualização obtida.

Figura 28 - Obtenção de dados para verificação do tempo de atualização

```
73
74    if (gps.date.isValid() && gps.time.isValid())
75    {
76      Serial.print("Hora (UTC): ");
77      Serial.print(gps.time.hour());
78      Serial.print(":");
```

Monitor Serial ×

Não conectado. Selecione uma placa e uma porta para conectar automaticamente.

```
11:51:01.164 -> Sistema GNSS - ESP32 + NEO-M8N
11:51:01.164 -> GPS iniciado...
11:52:03.395 -> ------------------------------------------------
11:52:03.395 -> Latitude: -18.935404
11:52:03.395 -> Longitude: -48.245840
11:52:03.395 -> Velocidade (km/h): 0.04
11:52:03.395 -> Satélites: 4
11:52:03.395 -> Hora (UTC): 14:52:2
11:52:03.395 -> Tempo de sistema (ms): 63257
11:52:03.395 -> ------------------------------------------------
11:52:03.395 ->
11:52:03.495 -> ------------------------------------------------
11:52:03.495 -> Latitude: -18.935404
11:52:03.532 -> Longitude: -48.245840
11:52:03.532 -> Velocidade (km/h): 0.04
11:52:03.532 -> Satélites: 4
```

Fonte: (O autor, 2026)

Primeiramente, de acordo com a Figura 28, verifica-se que o sistema foi iniciado às 11h51min01,164s, enquanto a primeira fixação válida do sinal GNSS ocorreu às 11h52min03,395s.

O tempo para a primeira fixação (*Time to First Fix* – TTFF) foi determinado pela diferença entre os instantes de inicialização e de obtenção do sinal válido, resultando em 62,231 segundos. Assim, o módulo levou aproximadamente 62,23 segundos (≈ 1 min e 2,23 s) para estabelecer a fixação do sinal GNSS.

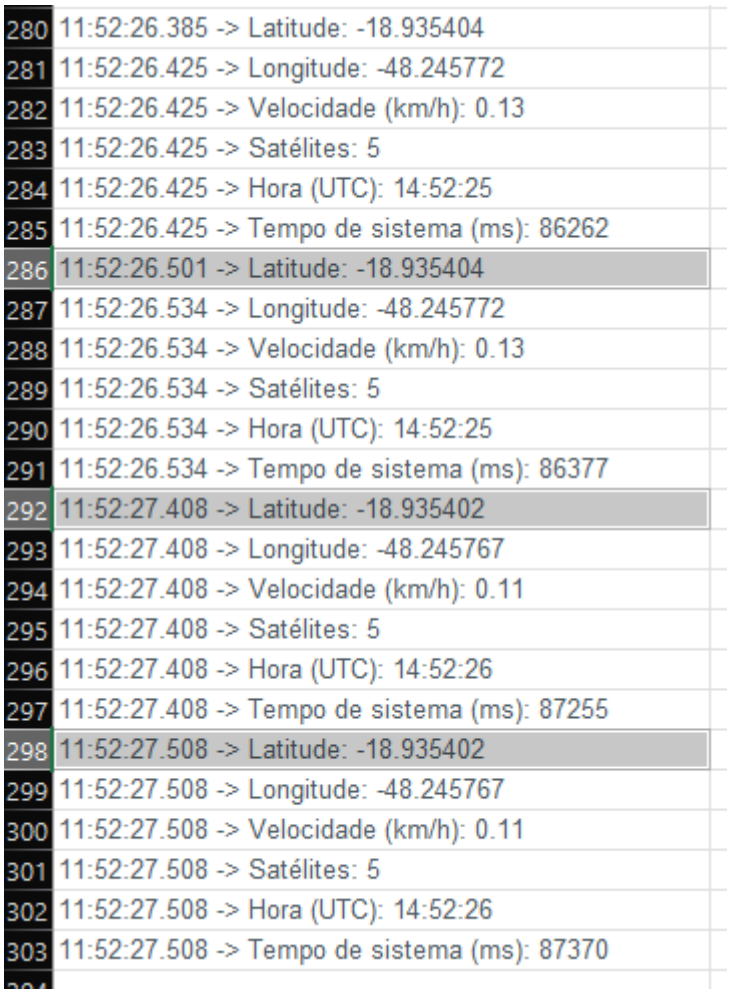
O valor obtido experimentalmente encontra-se acima da faixa típica especificada pelo fabricante para condições de inicialização a frio (*cold start*), a qual, em condições ideais de recepção e céu aberto, situa-se em torno de até 30 segundos.

Entretanto, é importante destacar que o ensaio foi realizado em ambiente real, sob condições atmosféricas não ideais, o que pode impactar diretamente o tempo de aquisição inicial do sinal GNSS.

Para a aplicação proposta neste trabalho, um tempo de inicialização da ordem de 62 segundos não compromete a operacionalidade do sistema. No contexto das atividades agrícolas, há um intervalo natural entre a partida do maquinário e o início efetivo das operações, período no qual o motor atinge a temperatura adequada de funcionamento e ocorre a correta lubrificação dos componentes mecânicos. Assim, o tempo adicional necessário para a fixação do sinal GNSS ocorre paralelamente aos procedimentos operacionais preliminares, não impactando de forma significativa a dinâmica de utilização do equipamento.

Em segundo lugar, para a análise da taxa média de atualização, foi considerada uma amostra composta por 50 registros obtidos por meio do monitor serial. Os dados coletados foram exportados para o Microsoft Excel, onde se realizou a comparação dos intervalos de tempo entre mensagens consecutivas, com o objetivo de determinar quantas atualizações são realizadas em um intervalo de 1 segundo, conforme apresentado na Figura 29.

Figura 29 - Dados do módulo exportado para o Excel



Fonte: (O autor, 2026)

Verifica-se que o sistema apresenta aproximadamente 3 atualizações por segundo, valor inferior ao esperado, considerando que uma configuração de 5 Hz implicaria ao menos 5 atualizações no mesmo intervalo de tempo.

Essa diferença está relacionada à estrutura do código implementado, especificamente à utilização da condição gps.location.isUpdated(). Essa função retorna verdadeiro apenas quando o objeto de localização recebe uma nova informação válida, proveniente do processamento completo de uma sentença NMEA que contenha dados de posição (como RMC ou GGA). Dessa forma, a exibição das informações, como latitude e longitude, ocorre somente quando há atualização efetiva desses dados.

Assim, caso a biblioteca não identifique alteração ou nova sentença válida no momento da verificação, não haverá nova exibição no monitor serial, ainda que o módulo esteja transmitindo dados continuamente. Portanto, a taxa observada não caracteriza uma limitação intrínseca do módulo GNSS, mas sim uma consequência do critério de atualização adotado no código para apresentação das informações.

Para a aplicação proposta, entretanto, essa taxa de atualização mostra-se adequada. Considerando que uma máquina agrícola em operação se desloca, em média, a velocidades inferiores a 10 km/h (aproximadamente 2,7 m/s), uma taxa de 3 Hz implica em uma atualização de posição a cada ~0,33 s, o que corresponde a um deslocamento aproximado de 0,9 m entre amostras consecutivas. Tal resolução espacial é suficiente para o monitoramento da área nas condições previstas de uso.

Adicionalmente, a fim de evitar a geração excessiva de registros em regiões muito próximas, o que poderia resultar em sobrecarga desnecessária de memória, são aplicados os filtros descritos no capítulo anterior. Esses mecanismos contribuem tanto para o refinamento da estimativa de posição quanto para a redução do armazenamento redundante de dados, preservando a eficiência do sistema e prevenindo o consumo excessivo de memória.

4.1.2 Erro de leitura

Com base na Figura 29, observa-se também um aspecto relevante relacionado à velocidade indicada pelo módulo GNSS. Durante os testes, foram registradas velocidades inferiores a 0,15 km/h, mesmo com o equipamento completamente estático no momento da coleta. Nota-se, portanto, que o valor de velocidade raramente assume exatamente 0 km/h, ainda que o sistema esteja em repouso.

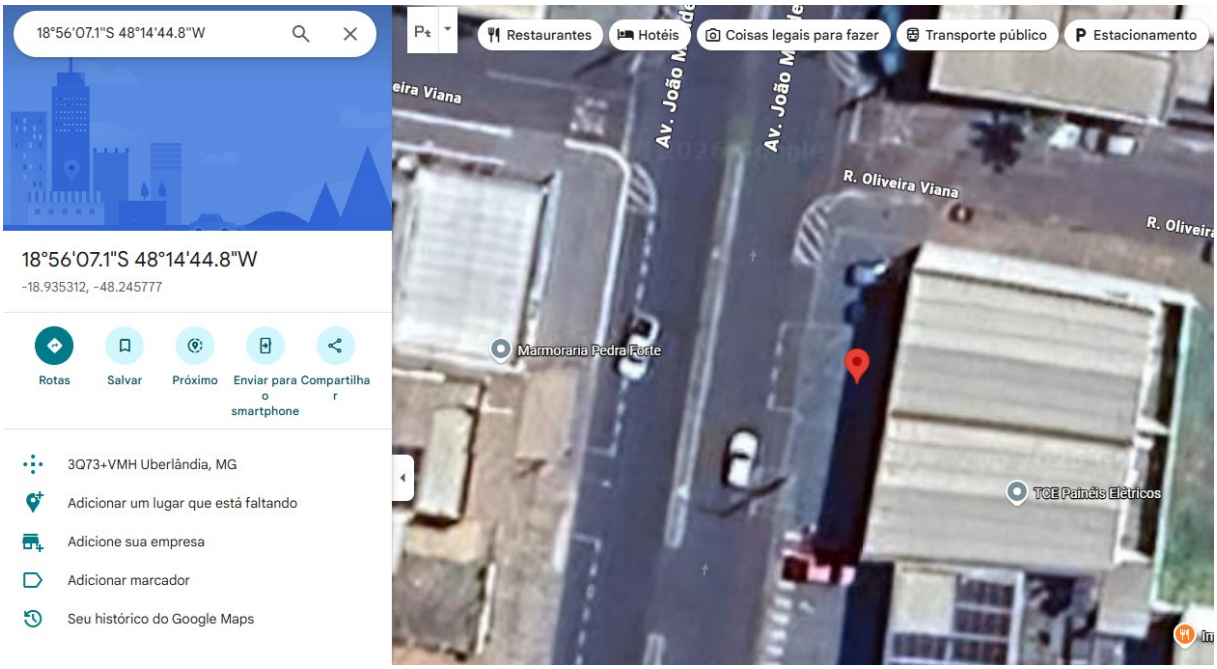
Esse comportamento está associado às pequenas variações nas leituras de posição, decorrentes de ruídos inerentes ao sistema GNSS, tais como multipercurso, imprecisões na solução de posicionamento e variações no cálculo doppler da velocidade. Essas oscilações geram deslocamentos residuais mínimos entre amostras consecutivas, resultando em uma velocidade diferente de zero.

Embora o erro observado seja reduzido, ele evidencia a necessidade da aplicação de filtros, conforme descrito anteriormente, a fim de minimizar variações espúrias e aumentar a confiabilidade dos dados registrados.

Dessa forma, a seguir será apresentada, com o auxílio do Google Earth Pro, uma análise do erro espacial correspondente ao teste realizado com 50 amostras, permitindo quantificar, em metros, a dispersão das medições obtidas. Ressalta-se que a coleta foi iniciada no instante da primeira fixação do sinal, ou seja, ainda durante a fase inicial de estabilização da solução de posicionamento.

A caracterização da área experimental e a respectiva localização geográfica utilizada como referência para a aquisição de dados são apresentadas na Figura 30.

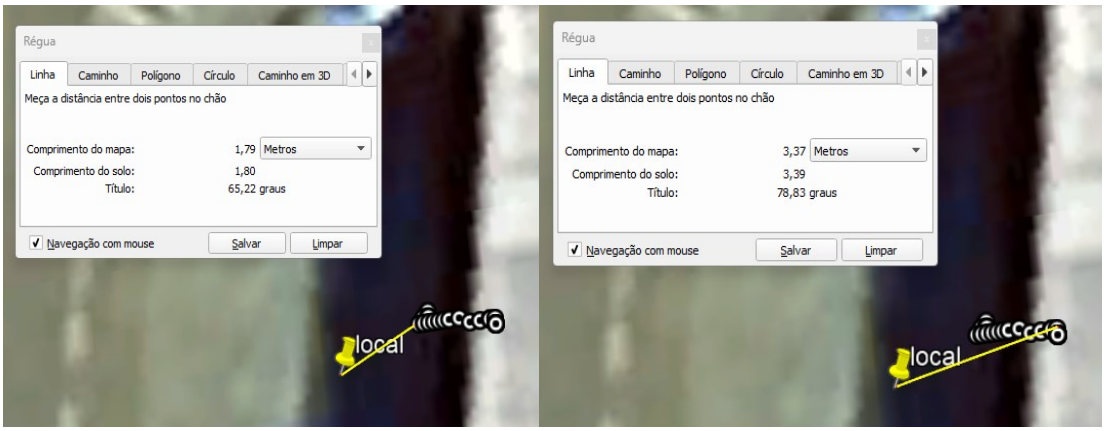
Figura 30 - Definição do local de coleta de dados.



Fonte: (O autor, 2026)

Os dados georreferenciados, sobrepostos ao local de referência dos testes, estão ilustrados na Figura 31. A representação evidencia a proximidade das amostras em relação ao ponto real de coleta, consolidando a validação experimental da trajetória monitorada.

Figura 31 - Erro máximo e mínimo com base no local de coleta de dados. (A) Erro Mínimo. (B) Erro Máximo



(A) (B)

Fonte: (O autor, 2026)

Observa-se que, nas primeiras 50 amostras, correspondentes a aproximadamente 25 segundos após a inicialização, o sistema apresentou erro máximo de 3,37 metros, valor que pode ser considerado significativo para medições estáticas.

Destaca-se que esse erro máximo ocorreu nos instantes iniciais de operação, especialmente nos primeiros 10 segundos, período no qual o número de satélites disponíveis variava entre 3 e 6, indicando que a solução de posicionamento ainda se encontrava em fase de consolidação.

À medida que o número de satélites aumentou e a solução GNSS passou a se estabilizar, a dispersão dos pontos reduziu-se para aproximadamente 2 metros de distância em relação ao ponto de referência, conforme a Figura 32.

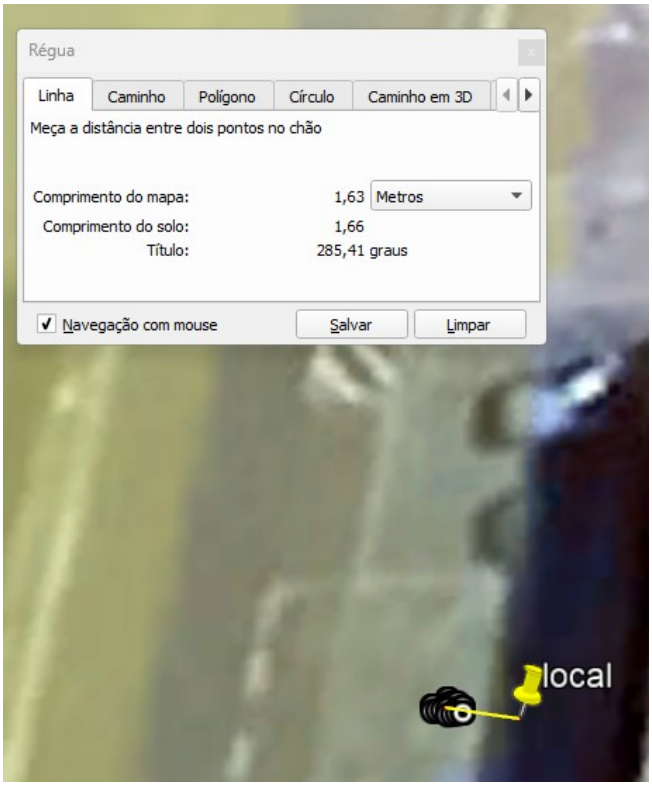
Figura 32 - Dados coletados para análise no Google Earth Pro

LATITUDE	LONGITUDE	SATÉLITES	HORAS (UTC)
-18.935300	-48.245747	3	15:8:51
-18.935300	-48.245747	4	15:8:51
-18.935301	-48.245746	4	15:8:52
-18.935301	-48.245746	5	15:8:52
-18.935301	-48.245746	5	15:8:53
-18.935301	-48.245746	6	15:8:53
-18.935301	-48.245746	6	15:8:54
-18.935301	-48.245746	6	15:8:54
-18.935301	-48.245746	6	15:8:55
-18.935301	-48.245746	6	15:8:55
-18.935300	-48.245746	6	15:8:56
-18.935300	-48.245746	6	15:8:56
-18.935300	-48.245747	6	15:8:57
-18.935300	-48.245747	6	15:8:57
-18.935300	-48.245747	6	15:8:58
-18.935300	-48.245747	7	15:8:58
-18.935300	-48.245749	7	15:8:59
-18.935300	-48.245749	7	15:8:59
-18.935300	-48.245751	7	15:9:0
-18.935300	-48.245751	7	15:9:0
-18.935299	-48.245753	7	15:9:1
-18.935299	-48.245753	7	15:9:1
-18.935299	-48.245755	7	15:9:2
-18.935299	-48.245755	7	15:9:2
-18.935299	-48.245756	7	15:9:3
-18.935299	-48.245756	7	15:9:3
-18.935299	-48.245757	7	15:9:4
-18.935299	-48.245757	7	15:9:4
-18.935299	-48.245758	7	15:9:5
-18.935299	-48.245758	7	15:9:5
-18.935299	-48.245759	7	15:9:6
-18.935299	-48.245759	7	15:9:6
-18.935299	-48.245760	7	15:9:7
-18.935299	-48.245760	7	15:9:7
-18.935299	-48.245760	7	15:9:8
-18.935299	-48.245760	7	15:9:8
-18.935299	-48.245761	7	15:9:9
-18.935299	-48.245761	7	15:9:9

Fonte: (O autor, 2026)

Na sequência, analisam-se as últimas 50 amostras, correspondentes a aproximadamente 1 minuto e 45 segundos após a inicialização do sistema, momento em que a solução de posicionamento já se encontrava estabilizada, antes de completar 2 minutos de funcionamento contínuo, conforme apresentado na Figura 33.

Figura 33 - Erro após estabilização do sistema



Fonte: (O autor, 2026)

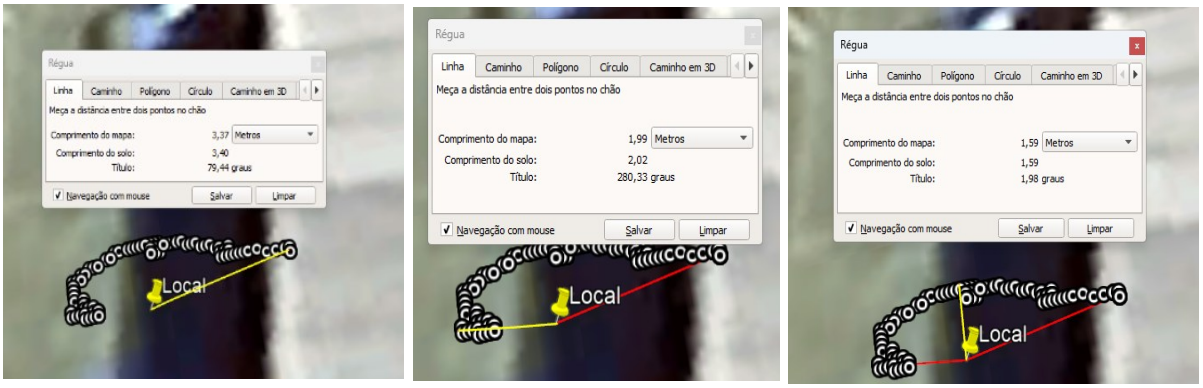
Verifica-se que, nesse intervalo, os dados passaram a se concentrar de forma mais significativa em torno de um único ponto, evidenciando redução da dispersão e maior estabilidade da solução GNSS. O erro máximo observado diminuiu para aproximadamente 1,63 metros, indicando melhoria substancial na confiabilidade das medições.

A localização de referência foi obtida por meio do Google Maps em dispositivo móvel, cuja precisão estimada durante o teste era de aproximadamente 4 metros, conforme indicado pelo próprio aplicativo. Assim, os resultados obtidos mostram-se compatíveis tanto com a incerteza da referência adotada quanto com as especificações apresentadas no datasheet do fabricante, que indicam erros horizontais da ordem de alguns metros para condições padrão de operação.

Para analisar a estabilização do sinal GNSS, realizou-se um ensaio com 250 amostras durante os dois primeiros minutos de operação. Esta abordagem visa documentar o comportamento do receptor logo após o início da coleta, evidenciando a redução da dispersão das coordenadas ao longo do tempo. O mapeamento dessa

evolução está detalhado na Figura 34, que apresenta a trajetória das amostras desde o instante zero até a estabilização do posicionamento.

Figura 34 - Demonstração da dispersão das coordenadas desde a inicialização do receptor até aproximadamente dois minutos de operação. (A) Erro de posicionamento do ponto real até a extremidade direita do conjunto de pontos. (B) Erro de posicionamento do ponto real até a extremidade



(A) (B) (C)

Fonte: (O autor, 2026)

Observa-se que os dados coletados apresentam maior dispersão nos instantes iniciais, evoluindo gradualmente até atingir uma região de concentração mais definida, correspondente ao período de estabilização da solução de posicionamento. Esse comportamento evidencia a necessidade de aguardar aproximadamente 2 minutos após a energização do módulo para a realização de medições mais confiáveis.

Dessa forma, reforça-se a importância da aplicação de critérios de filtragem, tais como a definição de um número mínimo de satélites, o estabelecimento de um limiar máximo para o HDOP, a utilização de filtro de média móvel e a adoção de um limiar mínimo de deslocamento baseado na velocidade. Esses mecanismos contribuem para aumentar a robustez e a precisão na coleta e no armazenamento dos dados.

Para além da análise qualitativa, a validação do sistema requer uma verificação quantitativa rigorosa. Nesse sentido, calculou-se a distância geodésica entre cada coordenada amostrada e o ponto de referência real utilizando a fórmula de Haversine, adotando-se o raio médio terrestre de 6.371.000 metros. A partir desses desvios individuais, foram extraídos indicadores estatísticos fundamentais: o erro médio, o desvio padrão e o Erro Quadrático Médio (RMSE), visando mensurar a acurácia e a precisão do posicionamento. O processamento desses dados foi realizado em planilha

eletrônica (Microsoft Excel), cujos resultados e métricas consolidadas estão detalhados na Figura 35.

Figura 35 - Cálculo de distância entre os pontos utilizando Haversine

LATITUDE	LONGITUDE	LAT.O	LONG.O	Dist. Haversine (m)
-18.935300	-48.245747	-18.935312	-48.245777	3.43
-18.935300	-48.245747	-18.935312	-48.245777	3.43
-18.935301	-48.245746	-18.935312	-48.245777	3.48
-18.935301	-48.245746	-18.935312	-48.245777	3.48
-18.935301	-48.245746	-18.935312	-48.245777	3.48
-18.935301	-48.245746	-18.935312	-48.245777	3.48
-18.935301	-48.245746	-18.935312	-48.245777	3.48
-18.935301	-48.245746	-18.935312	-48.245777	3.48
-18.935301	-48.245746	-18.935312	-48.245777	3.48
-18.935301	-48.245746	-18.935312	-48.245777	3.48
-18.935300	-48.245746	-18.935312	-48.245777	3.52
-18.935300	-48.245746	-18.935312	-48.245777	3.52
-18.935300	-48.245747	-18.935312	-48.245777	3.43
-18.935300	-48.245747	-18.935312	-48.245777	3.43
-18.935300	-48.245747	-18.935312	-48.245777	3.43
-18.935300	-48.245747	-18.935312	-48.245777	3.43
-18.935300	-48.245749	-18.935312	-48.245777	3.23
-18.935300	-48.245749	-18.935312	-48.245777	3.23
-18.935300	-48.245751	-18.935312	-48.245777	3.04
-18.935300	-48.245751	-18.935312	-48.245777	3.04
-18.935299	-48.245753	-18.935312	-48.245777	2.91
-18.935299	-48.245753	-18.935312	-48.245777	2.91
-18.935299	-48.245755	-18.935312	-48.245777	2.73
-18.935299	-48.245755	-18.935312	-48.245777	2.73
-18.935299	-48.245756	-18.935312	-48.245777	2.64
-18.935299	-48.245756	-18.935312	-48.245777	2.64
-18.935299	-48.245757	-18.935312	-48.245777	2.55

Fonte: (O autor, 2026)

Verifica-se que as distâncias calculadas em metros entre o ponto de referência e os dados inicialmente adquiridos apresentam coerência com a análise visual realizada no Google Earth Pro, validando a consistência dos resultados obtidos.

Na sequência, são apresentados os dados correspondentes às últimas amostras coletadas, acompanhados dos respectivos cálculos de erro médio, desvio padrão e erro quadrático médio, permitindo uma comparação direta entre o comportamento

inicial e o período posterior à estabilização do sistema. A Figura 36 mostra os valores de Erro Médio, Desvio Padrão e Erro Médio Quadrático.

Figura 36 - Valores de Erro Médio, Desvio Padrão e Erro Médio Quadrático

-18.935305	-48.245789	-18.935312	-48.245777		1.48
-18.935305	-48.245789	-18.935312	-48.245777		1.48
-18.935305	-48.245790	-18.935312	-48.245777		1.57
-18.935305	-48.245790	-18.935312	-48.245777		1.57
-18.935305	-48.245791	-18.935312	-48.245777		1.67
-18.935305	-48.245791	-18.935312	-48.245777		1.67
-18.935305	-48.245792	-18.935312	-48.245777		1.76
-18.935305	-48.245792	-18.935312	-48.245777		1.76
-18.935304	-48.245792	-18.935312	-48.245777		1.81
-18.935304	-48.245792	-18.935312	-48.245777		1.81
-18.935304	-48.245793	-18.935312	-48.245777		1.90
-18.935304	-48.245793	-18.935312	-48.245777		1.90
-18.935304	-48.245794	-18.935312	-48.245777		2.00
-18.935304	-48.245794	-18.935312	-48.245777		2.00
-18.935304	-48.245795	-18.935312	-48.245777		2.09
-18.935304	-48.245795	-18.935312	-48.245777		2.09
-18.935303	-48.245795	-18.935312	-48.245777		2.14
-18.935303	-48.245795	-18.935312	-48.245777		2.14
-18.935303	-48.245795	-18.935312	-48.245777		2.14
-18.935303	-48.245795	-18.935312	-48.245777		2.14
-18.935303	-48.245795	-18.935312	-48.245777		2.14
-18.935303	-48.245795	-18.935312	-48.245777		2.14
				Erro Médio:	2.15
				Desvio Padrão:	0.48
				Errro Quadr. Médio:	1.49

Fonte: (O autor, 2026)

Constata-se que, após o período de estabilização, as distâncias em relação ao ponto de referência tornam-se menores e mais concentradas, evidenciando a redução da dispersão dos dados. Os valores calculados de erro médio, desvio padrão e RMSE confirmam quantitativamente essa melhoria na precisão do sistema, corroborando a análise visual previamente apresentada.

4.2 Impacto da Bateria de Backup e Hot Start

O módulo u-blox NEO-M8N utilizado no projeto integra uma bateria de backup interna, a qual se mostrou fundamental para a dinâmica dos testes realizados, conforme será demonstrado a seguir.

De acordo com o fabricante, o módulo pode apresentar os seguintes comportamentos de inicialização:

- Cold Start: ocorre na primeira ativação do dia, quando não há dados orbitais previamente armazenados. Nessas condições, o tempo médio de fixação é de aproximadamente 30 segundos em ambiente com céu aberto.
- Hot Start: ocorre quando as efemérides permanecem armazenadas na memória volátil, alimentada pela bateria interna. Nessa situação, reinicializações rápidas, como após troca da bateria do protótipo ou desligamentos breves, resultam em tempo de fixação de aproximadamente 1 segundo.

Com base nessas características, o sistema apresentou maior prontidão operacional quando comparado a receptores de entrada que não possuem bateria de backup, garantindo a continuidade da coleta de dados mesmo diante de interrupções momentâneas na alimentação do microcontrolador ESP32.

Nos testes realizados para validação desse comportamento, observou-se que, após um desligamento completo do sistema por cerca de 10 segundos, o tempo de reinicialização permaneceu em aproximadamente 1 segundo, conforme especificado pelo fabricante, como ilustrado na Figura 37.

Figura 37 - Tempo de reinicialização após 10 segundos desligado

```
12:27:48.377 -> Sistema GNSS - ESP32 + NEO-M8N
12:27:48.377 -> GPS iniciado...
12:27:49.411 -> ------------------------------------------------
12:27:49.411 -> Latitude: -18.935277
12:27:49.411 -> Longitude: -48.245569
12:27:49.411 -> Velocidade (km/h): 6.32
12:27:49.411 -> Satélites: 4
12:27:49.411 -> Hora (UTC): 15:27:48
12:27:49.411 -> Tempo de sistema (ms): 2055
12:27:49.411 -> ------------------------------------------------
12:27:49.411 ->
```

Fonte: (O autor, 2026)

Após um período de aproximadamente 20 segundos com o sistema totalmente desligado, o tempo de reinicialização aumentou para pouco mais de 3 segundos. Esse

comportamento indica que a bateria de backup ainda mantém parcialmente os dados necessários para uma inicialização rápida (warm start), conforme demonstrado na Figura 38 apresentada adiante.

Figura 38 - Tempo de reinicialização após 20 segundos desligado

```
12:30:44.158 -> Sistema GNSS - ESP32 + NEO-M8N
12:30:44.158 -> GPS iniciado...
12:30:47.396 -> ------------------------------------------------
12:30:47.396 -> Latitude: -18.935331
12:30:47.396 -> Longitude: -48.245374
12:30:47.396 -> Velocidade (km/h): 6.69
12:30:47.396 -> Satélites: 4
12:30:47.432 -> Hora (UTC): 15:30:46
12:30:47.432 -> Tempo de sistema (ms): 4260
12:30:47.432 -> ------------------------------------------------
```

Fonte: (O autor, 2026)

Considerando uma primeira inicialização e, posteriormente, um desligamento completo do sistema por aproximadamente 2 minutos (sem qualquer alimentação externa), observou-se que o tempo de reinicialização foi de 51 segundos, valor cerca de 21 segundos superior ao especificado no datasheet do fabricante para determinadas condições.

Dessa forma, pode-se considerar que, após aproximadamente 2 minutos sem alimentação, o módulo passa a se comportar de maneira semelhante a uma inicialização a frio (*cold start*), indicando que a bateria de backup não consegue manter integralmente os dados necessários por esse intervalo de tempo. Esse comportamento pode ser visualizado na Figura 39.

Figura 39 - Tempo de reinicialização após 2 minutos desligado

```
12:24:11.880 -> Sistema GNSS - ESP32 + NEO-M8N
12:24:11.880 -> GPS iniciado...
12:25:02.395 -> ------------------------------------------------
12:25:02.395 -> Latitude: -18.935327
12:25:02.395 -> Longitude: -48.245585
12:25:02.395 -> Velocidade (km/h): 0.35
12:25:02.395 -> Satélites: 4
12:25:02.435 -> Hora (UTC): 15:25:1
12:25:02.435 -> Tempo de sistema (ms): 51528
12:25:02.435 -> ------------------------------------------------
12:25:02.435 ->
```

Fonte: (O autor, 2026)

4.3 Validação da Comunicação Serial com filtros

Inicialmente, o sistema foi avaliado operando sem a aplicação de filtros, com o objetivo de analisar seu comportamento em condições originais de funcionamento e verificar se o desempenho observado estava de acordo com as especificações técnicas apresentadas no datasheet do fabricante. Os resultados indicaram que o módulo opera dentro dos padrões previamente mencionados, confirmando a conformidade com as características técnicas informadas.

A partir dessa constatação, tornou-se necessária a realização de novos testes com a aplicação dos filtros propostos no capítulo metodológico, a fim de avaliar o ganho em estabilidade, confiabilidade e eficiência no armazenamento dos dados. Os filtros implementados consistem em:

- Validação da solução de posicionamento: os dados somente são considerados quando o módulo GNSS indica posição válida, sendo realizado o registro apenas quando o número de satélites visíveis é igual ou superior a quatro, garantindo maior robustez na solução de posicionamento.
- Limiar mínimo de deslocamento baseado na velocidade: o armazenamento de novos pontos ocorre apenas quando há deslocamento significativo. Em velocidades mais elevadas, o intervalo de atualização pode ser reduzido (por exemplo, de 5 s para 3 s), desde que o veículo tenha percorrido ao menos 3

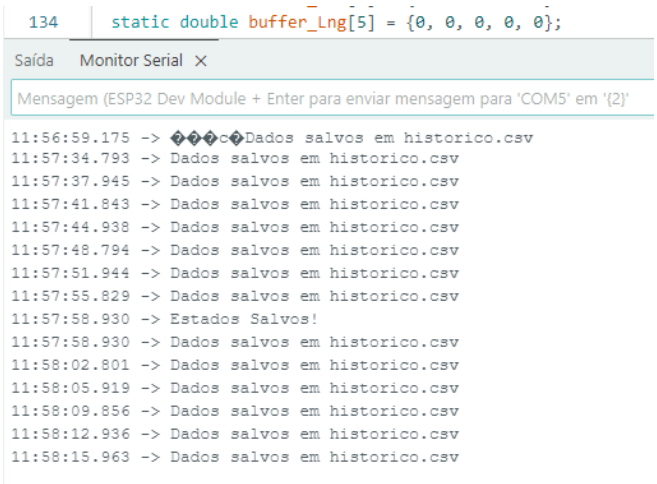
metros ou aproximadamente 30% da velocidade estimada, evitando redundância de registros.

- Filtro de média móvel: aplicação de média aritmética móvel sobre as amostras coletadas, com o objetivo de suavizar oscilações e reduzir ruídos inerentes ao posicionamento GNSS.

Para assegurar a comparabilidade dos resultados, os testes com filtros foram realizados na mesma localização anteriormente utilizada nos testes sem filtragem, permitindo uma análise coerente entre os dois cenários.

Na versão do sistema com implementação de filtros, a visualização e a exportação de dados passaram a ser gerenciadas integralmente por um servidor web nativo no ESP32, prescindindo da dependência do monitor serial. Essa arquitetura foi adotada para otimizar o consumo de memória e o ciclo de processamento da CPU, mitigando o *overhead* causado por sucessivas chamadas de depuração serial (Serial.print). Assim, mantiveram-se apenas as mensagens críticas de log referentes à persistência de estados, garantindo maior estabilidade ao *firmware*. A interface de monitoramento resultante é apresentada na Figura 40.

Figura 40 - Monitor Serial ao obter informações do ESP32



Fonte: (O autor, 2026)

Para a obtenção dos dados, basta conectar-se à rede Wi-Fi gerada pelo ESP32 e realizar o *download* do arquivo no formato .csv disponibilizado pelo sistema. Posteriormente, esse arquivo pode ser importado para o Google Earth Pro, permitindo

a análise espacial dos pontos coletados e a verificação da dispersão das coordenadas registradas. A Figura 41 apresenta a interface do sistema, na qual é possível visualizar o botão destacado em azul claro destinado ao *download* do arquivo.

Figura 41 - Interface Web para *download* do arquivo gerado

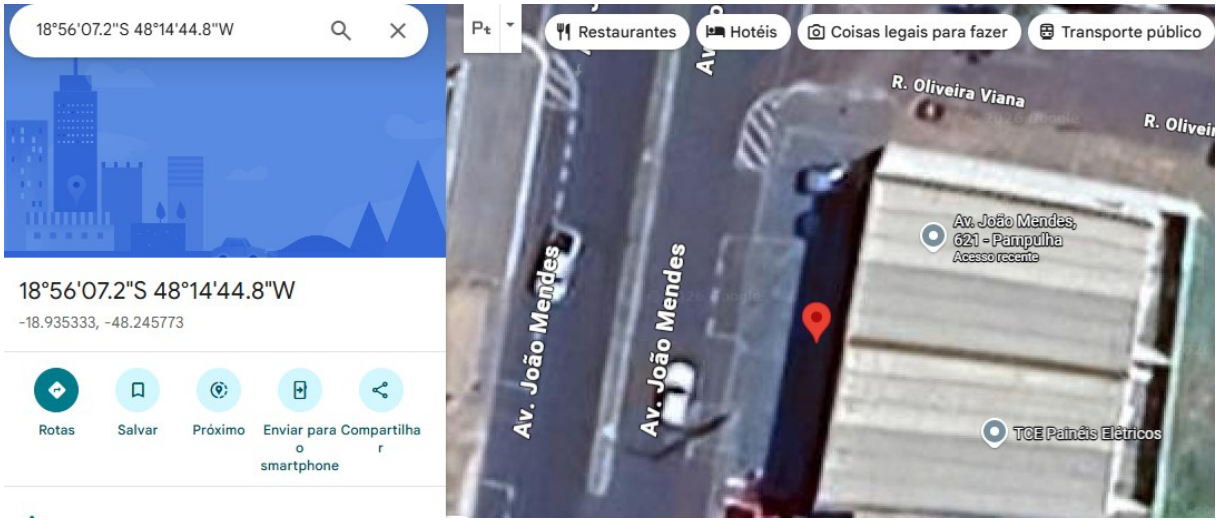
Fonte: (O autor, 2026)

Considerando o sistema com filtros, uma das condições estabelecidas para o armazenamento dos dados é que o valor de precisão, representado pelo HDOP, seja inferior a 2. Durante o período de estabilização, observou-se um valor inicial de 5.8, indicando que o sistema somente iniciaria o salvamento após a redução desse parâmetro para níveis considerados adequados.

Para fins de validação e comparação com os dados filtrados do módulo GNSS NEO-M8N, utilizou-se a localização obtida por um *smartphone*, registrando as coordenadas no mesmo ponto de análise. Ressalta-se que o dispositivo móvel apresentou uma precisão estimada de aproximadamente 4 metros, servindo como um referencial qualitativo de comparação, visto que era o recurso disponível para a aferição em campo. Cabe destacar que os *smartphones* empregam sistemas de posicionamento híbrido, que combina o sinal GNSS com o mapeamento de redes Wi-

Fi e torres de celular (A-GPS), para otimizar a estimativa de localização. As coordenadas obtidas pelo dispositivo móvel são apresentadas na Figura 42.

Figura 42 : Localização do GPS do Smartphone



Fonte: (O autor, 2026)

Com os dados obtidos pelo módulo associado ao ESP32, foi registrada uma amostra composta por 32 pontos ao longo de aproximadamente 2 minutos de operação do GPS sob condição estática, conforme pode ser observado na Figura 43.

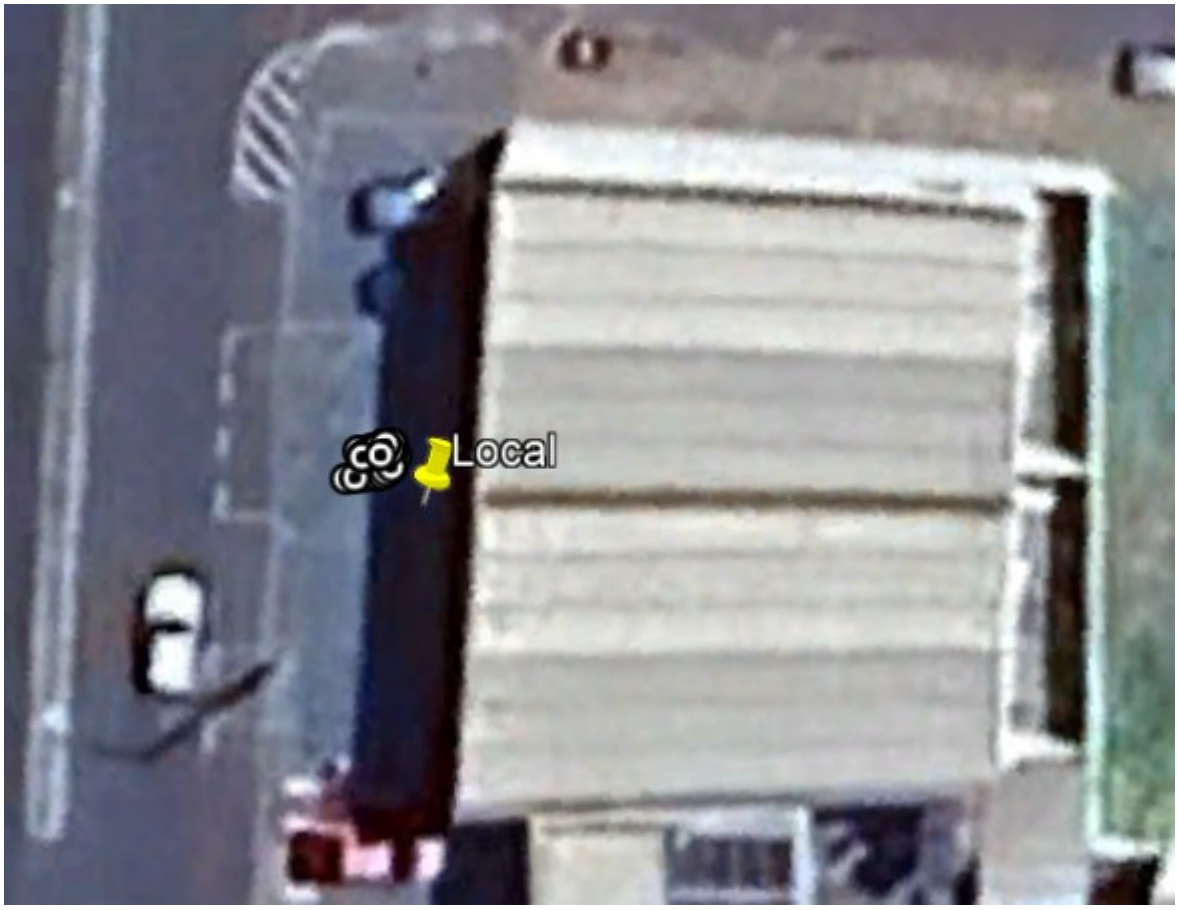
Figura 43 - Amostra de dados obtidos durante período de 2 minutos

	Lon	DistPercorrida	DistTotal	Hectare	Sat	Vel(km/h)	HDOP	Data	Hora	Tempo(min)
2	-48.245791	0	0	0	12	1.63	1.03	25/02/2026	12:00:20	0
3	-48.24579	0	0	0	12	0.02	1.06	25/02/2026	12:00:24	0
4	-48.24579	0	0	0	12	0.02	1.03	25/02/2026	12:00:27	0
5	-48.24579	0	0	0	12	0.65	1.03	25/02/2026	12:00:31	0
6	-48.245788	0	0	0	12	1.02	1.02	25/02/2026	12:00:34	0
7	-48.245787	0	0	0	12	0.02	0.74	25/02/2026	12:00:38	0
8	-48.245786	0	0	0	12	0.06	1.02	25/02/2026	12:00:41	0
9	-48.245785	0	0	0	12	0.04	1.02	25/02/2026	12:00:45	0
10	-48.245785	0	0	0	12	0.02	1.02	25/02/2026	12:00:48	0
11	-48.245785	0	0	0	12	0.04	1.02	25/02/2026	12:00:52	1
12	-48.245784	0	0	0	12	0.02	1.02	25/02/2026	12:00:55	1
13	-48.245783	0	0	0	12	0.02	1.02	25/02/2026	12:00:58	1
14	-48.245783	0	0	0	12	0.09	0.74	25/02/2026	12:01:02	1
15	-48.245782	0	0	0	12	0.04	1.06	25/02/2026	12:01:05	1
16	-48.24578	0	0	0	12	0.04	0.74	25/02/2026	12:01:09	1
17	-48.24578	0	0	0	12	0.24	0.74	25/02/2026	12:01:12	1
18	-48.24578	0	0	0	12	0.04	0.8	25/02/2026	12:01:16	1
19	-48.245781	0	0	0	12	0.02	0.8	25/02/2026	12:01:19	1
20	-48.245781	0	0	0	12	0.19	0.8	25/02/2026	12:01:23	1
21	-48.24578	0	0	0	12	0.06	1.19	25/02/2026	12:01:26	1
22	-48.245779	0	0	0	12	0.02	1.19	25/02/2026	12:01:30	1
23	-48.24578	0	0	0	12	0.04	0.8	25/02/2026	12:01:33	1
24	-48.245781	0	0	0	12	0.09	0.74	25/02/2026	12:01:37	1
25	-48.245781	0	0	0	12	0.06	0.74	25/02/2026	12:01:40	1
26	-48.245781	0	0	0	12	0.09	0.74	25/02/2026	12:01:44	1
27	-48.245783	0	0	0	12	0.02	1.02	25/02/2026	12:01:47	1
28	-48.245784	0	0	0	12	0.04	1.02	25/02/2026	12:01:51	2
29	-48.245786	0	0	0	12	0	0.74	25/02/2026	12:01:54	2
30	-48.24579	0	0	0	12	0	0.8	25/02/2026	12:01:58	2
31	-48.245793	0	0	0	12	0.02	0.8	25/02/2026	12:02:01	2
32	-48.245796	0	0	0	12	0.04	0.8	25/02/2026	12:02:05	2
33										

Fonte: (O autor, 2026)

Ao plotar os dados no Google Earth Pro, observou-se uma menor dispersão das coordenadas de latitude e longitude, evidenciando maior concentração dos pontos em uma região específica. Para fins comparativos, foi adicionado um marcador em amarelo representando a posição registrada pelo GPS do smartphone, como pode ser visto na Figura 44.

Figura 44 - Demonstração da dispersão dos dados com o GPS do Smartphone

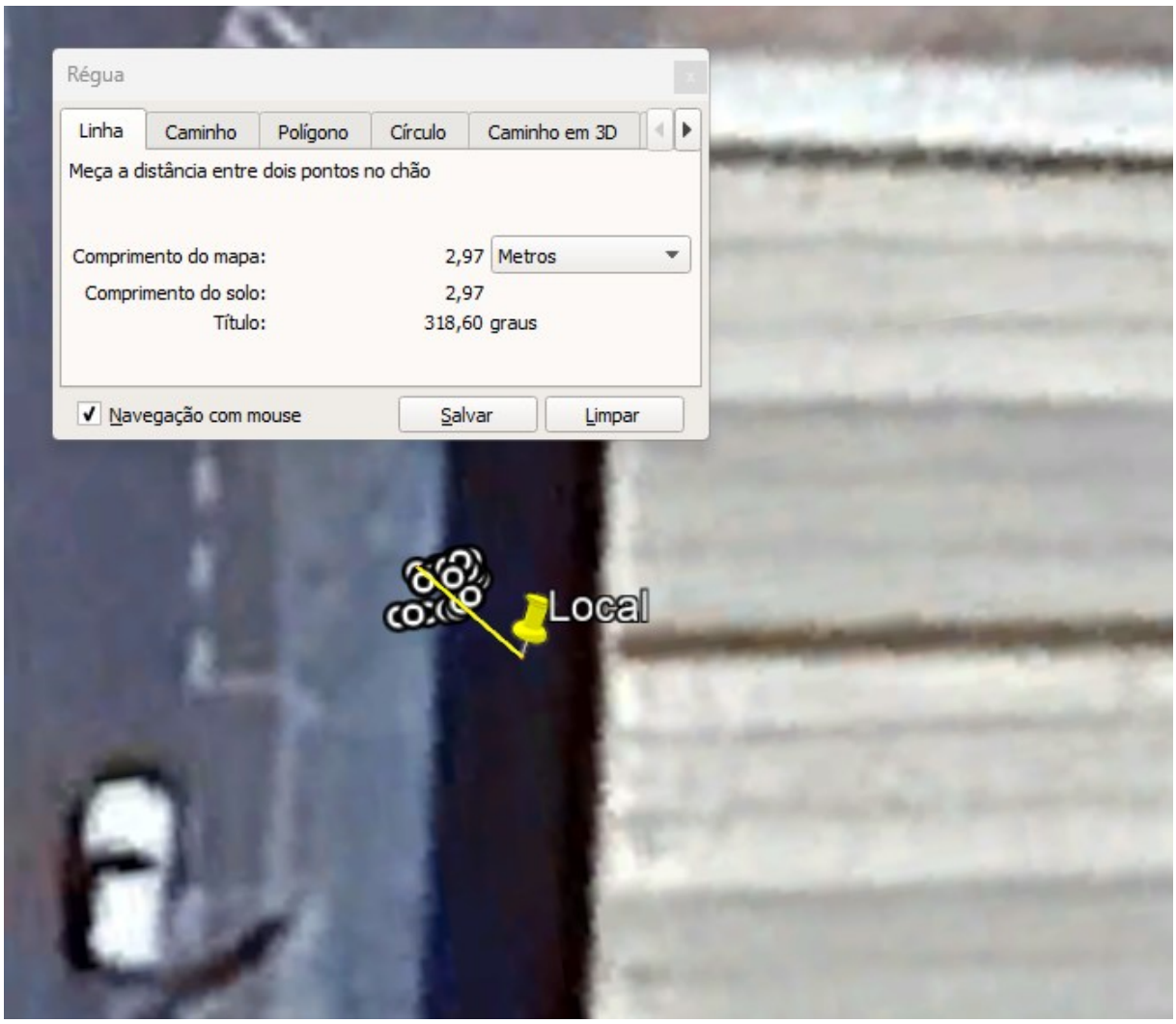


Fonte: (O autor, 2026)

Utilizando a ferramenta de medição do Google Earth Pro, verificou-se que a distância máxima entre o ponto de referência (GPS do smartphone) e os pontos obtidos pelo módulo NEO-M8N foi de aproximadamente 2,97 metros. Observa-se que, após a estabilização do HDOP abaixo de 2, os dados passaram a apresentar comportamento mais consistente e concentrado.

A aplicação dos filtros permitiram uma trajetória mais coesa e compacta, reduzindo drasticamente a dispersão espacial observada nos dados brutos. Esse comportamento valida a importância do critério de filtragem adotado para assegurar a confiabilidade das medições de área e distância, como evidenciado na análise comparativa da Figura 45.

Figura 45 - Indicação da distância máxima apresentada pelos dados em comparação como GPS do smartphone



Fonte: (O autor, 2026)

Com base nos dados coletados, procedeu-se ao cálculo do erro médio de distanciamento em relação ao ponto de referência, utilizando a fórmula de Haversine para determinação da distância geodésica entre coordenadas. Também foram calculados o desvio padrão e o erro quadrático médio (RMSE), a fim de quantificar estatisticamente a dispersão dos dados, como será visto na figura adiante.

Antes de demonstrar a figura, outro aspecto relevante refere-se à quantidade de amostras registradas em 2 minutos. Enquanto o sistema sem filtros registrava aproximadamente três amostras por segundo (totalizando cerca de 250 registros), o sistema com filtros registrou apenas 35 amostras no mesmo intervalo, uma vez que,

em condição estática e com velocidade inferior a 1 km/h, não havia deslocamento significativo para justificar novos registros.

Tal comportamento reitera que a filtragem é essencial para mitigar o armazenamento de dados ruidosos, otimizando o uso da memória interna do ESP32 e conferindo maior fluidez à análise de pós-processamento. A Figura 46 sintetiza os indicadores de Erro Médio, Desvio Padrão e RMSE, evidenciando que o tratamento algorítmico resulta em métricas significativamente mais confiáveis e representativas da realidade de campo.

Figura 46 - Apresentação do Erro Médio, Desvio Padrão e Erro Quadrático Médio

	Lat	Lon		LAT.O	LONG.O		Dist. Haversine (m)
2	-18.935314	-48.245791		-18.935333	-48.245773		2.84
3	-18.935315	-48.24579		-18.935333	-48.245773		2.68
4	-18.935316	-48.24579		-18.935333	-48.245773		2.60
5	-18.935316	-48.24579		-18.935333	-48.245773		2.60
6	-18.935315	-48.245788		-18.935333	-48.245773		2.55
7	-18.935314	-48.245787		-18.935333	-48.245773		2.58
8	-18.935314	-48.245786		-18.935333	-48.245773		2.52
9	-18.935314	-48.245785		-18.935333	-48.245773		2.46
10	-18.935315	-48.245785		-18.935333	-48.245773		2.37
11	-18.935316	-48.245785		-18.935333	-48.245773		2.27
12	-18.935316	-48.245784		-18.935333	-48.245773		2.22
13	-18.935314	-48.245783		-18.935333	-48.245773		2.36
14	-18.935313	-48.245783		-18.935333	-48.245773		2.46
15	-18.935313	-48.245782		-18.935333	-48.245773		2.42
16	-18.935313	-48.24578		-18.935333	-48.245773		2.34
17	-18.935314	-48.24578		-18.935333	-48.245773		2.24
18	-18.935314	-48.24578		-18.935333	-48.245773		2.24
19	-18.935315	-48.245781		-18.935333	-48.245773		2.17
20	-18.935316	-48.245781		-18.935333	-48.245773		2.07
21	-18.935316	-48.24578		-18.935333	-48.245773		2.03
22	-18.935317	-48.245779		-18.935333	-48.245773		1.89
23	-18.935317	-48.24578		-18.935333	-48.245773		1.93
24	-18.935318	-48.245781		-18.935333	-48.245773		1.87
25	-18.93532	-48.245781		-18.935333	-48.245773		1.67
26	-18.93532	-48.245781		-18.935333	-48.245773		1.67
27	-18.935321	-48.245783		-18.935333	-48.245773		1.70
28	-18.935322	-48.245784		-18.935333	-48.245773		1.68
29	-18.935322	-48.245786		-18.935333	-48.245773		1.83
30	-18.935322	-48.24579		-18.935333	-48.245773		2.17
31	-18.935322	-48.245793		-18.935333	-48.245773		2.43
32	-18.935322	-48.245796		-18.935333	-48.245773		2.71
33							
34						Erro Médio:	2.27
35						Desvio Padrão:	0.33
36						Errro Quadr. Médio:	1.50

Fonte: (O autor, 2026)

A análise comparativa demonstrou que, com a utilização dos filtros, os dados apresentaram maior concentração espacial. O erro médio obtido foi de

aproximadamente 2,27 metros, valor levemente superior ao observado no teste sem filtros (2,15 metros). Contudo, o desvio padrão apresentou valor inferior, indicando menor variabilidade das medições e maior estabilidade dos dados.

Além disso, a distância máxima registrada entre o ponto de referência e os dados obtidos com filtros foi de 2,84 metros, correspondente ao primeiro ponto válido após a estabilização do HDOP. No teste sem filtros, a distância máxima foi de 3,52 metros, representando uma diferença aproximada de 0,68 metros.

Ressalta-se que ambos os testes foram realizados considerando um período de estabilização de aproximadamente 2 minutos após a inicialização do sistema.

Portanto, embora o módulo GNSS apresente desempenho adequado mesmo sem filtragem, a aplicação dos filtros mostrou-se essencial para:

- Reduzir a dispersão dos dados;
- Diminuir o armazenamento redundante;
- Preservar a memória do sistema embarcado;
- Melhorar a organização e confiabilidade das informações para análise posterior.

Assim, conclui-se que a implementação dos filtros não apenas melhora a estabilidade do sistema, mas também otimiza seu funcionamento para aplicações práticas em campo.

4.4 Avaliação do Cálculo de Distância

Diante da análise comparativa entre os dados obtidos com aplicação de filtros e aqueles provenientes do sistema operando com dados brutos, torna-se necessária a avaliação do cálculo de distância, considerado um dos principais pilares do presente projeto.

Para essa etapa, optou-se por utilizar os dados provenientes do sistema com filtragem ativa, uma vez que essa é a configuração efetivamente proposta para utilização em campo. A aplicação dos filtros visa reduzir redundâncias no armazenamento, minimizar dispersões excessivas e garantir maior coerência espacial

entre os pontos registrados, proporcionando maior confiabilidade ao cálculo da distância percorrida.

A validação do cálculo foi realizada em dois cenários distintos: em condições climáticas normais e em condições adversas, aproveitando-se um período de chuva intensa para analisar o comportamento do módulo durante uma tempestade. Essa abordagem permitiu verificar não apenas a precisão do algoritmo de cálculo de distância em situação ideal, mas também sua robustez frente a interferências ambientais.

4.4.1 Avaliando Cálculo de Distância em Condições Normais

Após a validação comparativa dos dados obtidos com e sem a aplicação dos filtros implementados, constatou-se que os critérios de filtragem (validação de posição, número mínimo de satélites, limiar de HDOP, média móvel e limiar mínimo baseado na velocidade) promovem melhoria significativa na estabilidade e na confiabilidade das medições.

Dessa forma, optou-se por avaliar o desempenho do sistema em condições normais de operação, realizando um percurso superior a 10 km com o intuito de verificar a precisão do cálculo da distância percorrida.

Antes do início do trajeto, realizou-se o reset dos dados por meio do servidor web embarcado, garantindo que as medições registradas correspondessem exclusivamente ao percurso analisado, conforme pode ser visto na Figura 47.

Figura 47 - Reset dos Dados Totais do Arquivo

Fonte: (O autor, 2026)

Com o GPS devidamente resetado, definiu-se a utilização de parâmetros externos para comparação dos resultados, sendo eles:

- O odômetro (*trip*) do veículo;
- A medição do trajeto via Google Earth Pro.

A adoção de múltiplos parâmetros de referência conferiu maior robustez à análise comparativa dos resultados. No estágio inicial do percurso, o hodômetro parcial (*trip*) do veículo registrava a marca de 11,5 km, servindo como base para a validação da distância acumulada pelo protótipo, conforme documentado na Figura 48.

Figura 48 - Quilometragem antes de iniciar o trajeto

Fonte: (O autor, 2026)

Durante a execução do percurso, identificou-se a presença de torres de radiotransmissão em áreas adjacentes ao trajeto. Esse fator é tecnicamente relevante, pois tais infraestruturas podem atuar como fontes de interferência eletromagnética (EMI) ou causar obstruções físicas, resultando na degradação momentânea da precisão do posicionamento. Embora o sistema fundamente-se exclusivamente em sinais orbitais, a proximidade com grandes massas metálicas pode induzir o fenômeno de multicaminho (multipath), afetando a relação sinal-ruído (SNR) das mensagens capturadas pelo módulo NEO-M8N. A Figura 49 ilustra a localização dessas estruturas no cenário experimental.

Figura 49 - Demonstração da presença de torres de sinal que interferem no sinal obtido pelo NEO-M8N

Fonte: (O autor, 2026)

Ao término do percurso, com duração aproximada de 14 minutos, o odômetro parcial do veículo indicou uma distância total percorrida de 11,4 km, conforme documentado na Figura 50. Este valor de referência foi estabelecido a partir da variação absoluta registrada no odômetro entre os instantes inicial e final do ensaio:

- Valor inicial: 11,5 km
- Valor final: 22,9 km

Figura 50 – Quilometragem ao final do trajeto

Fonte: (O autor, 2026)

Por meio do servidor web do sistema embarcado, o módulo GPS registrou:

- Tempo decorrido: 15 minutos e 07 segundos;
- Distância percorrida: 10,792 km.

Observa-se, portanto, uma discrepância considerável em relação ao valor registrado pelo odômetro parcial do veículo. Tal divergência justifica a inclusão de uma terceira fonte de referência para fins de validação cruzada e maior precisão na análise. A Figura 51 ilustra os dados compilados pelo ESP32, os quais são disponibilizados via servidor web através da interface de rede configurada em modo ponto de acesso.

Figura 51 - Dados finais registrados pelo Servidor Web

Fonte: (O autor, 2026)

Considerando que o *trip* do veículo pode apresentar imprecisões, especialmente devido a alterações nas dimensões dos pneus, optou-se por validar o trajeto utilizando o Google Earth Pro.

Ressalta-se que o veículo utilizava pneus 175/70R14, enquanto o padrão recomendado para o modelo do carro seria de 185/70R14. Essa diferença altera o diâmetro efetivo do conjunto roda/pneu, impactando diretamente na leitura do velocímetro e do odômetro, podendo introduzir erro na medição da distância real percorrida.

Nesse contexto, as coordenadas consolidadas pelo sistema embarcado foram exportadas e, subsequentemente, processadas em *software* especializado para a reconstituição e análise técnica da trajetória. A Figura 52 ilustra o mapeamento resultante, permitindo a visualização espacial do percurso e a validação do comportamento do protótipo em campo.

Figura 52 - Demonstração do trajeto percorrido no Google Earth Pro

Fonte: (O autor, 2026)

Conforme apresentado na figura correspondente, mesmo com a presença de torres de sinal ao longo do percurso, não foram observadas flutuações significativas no traçado do caminho registrado, o que leva a verificar a robustez do módulo escolhido para ser utilizado no projeto.

A distância medida pelo Google Earth Pro foi de 10,835 km, conforme pode ser visto na Figura 53.

Figura 53 - Distância marcada pelo Google Earth Pro

Fonte: (O autor, 2026)

Comparando-se os valores obtidos:

- Google Earth Pro: 10,835 km
- Módulo GPS: 10,792 km

A diferença absoluta foi de 43 metros, correspondendo a um erro percentual aproximado de 0,4%.

Esse resultado demonstra elevada concordância entre o sistema desenvolvido e a medição realizada via *software* de referência. Observa-se, inclusive, que o módulo GNSS apresentou resultado mais próximo do valor obtido no Google Earth do que o *trip* do veículo, cuja leitura pode ter sido influenciada pela diferença nas dimensões dos pneus.

Considerando que o teste foi realizado em ambiente urbano, com presença de torres de transmissão, edificações e outros possíveis fatores de interferência, os

resultados obtidos mostram-se satisfatórios. Em aplicações rurais, onde há menor incidência de obstáculos verticais e interferências eletromagnéticas, espera-se desempenho igual ou superior ao observado neste ensaio.

Dessa forma, conclui-se que o sistema desenvolvido apresenta precisão adequada para aplicações em campo, atendendo aos objetivos propostos no projeto.

4.4.2 Avaliando Cálculo de Distância em Momento de Chuva

Além de avaliar o funcionamento do módulo GNSS NEO-M8N em condições climáticas normais ou parcialmente nubladas, como foi o caso mencionado anteriormente, torna-se relevante analisar seu desempenho em períodos chuvosos. Embora máquinas agrícolas, por exemplo, normalmente não operem sob chuva intensa devido à redução da eficiência operacional e ao possível desperdício de insumos durante a plantabilidade ou colheita, a análise em condições adversas permite compreender as limitações do sistema de posicionamento.

Para a verificação da acurácia do sistema quanto à distância, adotou-se o hodômetro parcial do veículo como base comparativa. A leitura inicial do percurso está documentada na Figura 54. O ensaio foi conduzido sob condições de nebulosidade, sem ocorrência de chuva, garantindo um ambiente estável para a recepção das sentenças NMEA pelo módulo NEO-M8N, apesar da redução na relação sinal-ruído (SNR) típica de climas nublados.

Figura 54 - Quilometragem do tempo inicial do trajeto

Fonte: (O autor, 2026)

Adicionalmente, o sistema web foi iniciado com os dados previamente resetados, garantindo que a coleta da distância percorrida ocorresse a partir de um estado inicial zerado, conforme a Figura 55.

Figura 55 - Tempo inicial apresentado pelo sistema web

Fonte: (O autor, 2026)

É importante observar que as condições climáticas variaram durante o ensaio: o início ocorreu sob céu nublado, seguido por um período de chuva intensa. Tal fenômeno atmosférico é relevante para a análise, pois a umidade elevada pode interferir na relação sinal-ruído (SNR) das sentenças NMEA. O registro fotográfico da Figura 56 evidencia a baixa visibilidade e a intensidade da chuva durante essa etapa do trajeto.

Figura 56 - Visibilidade da via utilizando o módulo GNSS em período de chuva intensa

Fonte: (O autor, 2026)

A conclusão da trajetória experimental foi documentada pela marcação de 308,4 km no odômetro do veículo (Figura 57). Este dado serve como o limite superior para a determinação da variação de deslocamento utilizada na validação do sistema embarcado.

Figura 57 - Quilometragem do tempo final do trajeto

Fonte: (O autor, 2026)

Com base nos dados coletados, a distância de referência fornecida pelo odômetro foi de 13,9 km. Em contrapartida, o módulo NEO-M8N registrou uma distância acumulada de 20 km, evidenciando uma discrepância significativa de 6,1 km, como pode ser visto na Figura 58. Tal variação é justificada pela instabilidade da solução de posicionamento sob chuva forte, condição que compromete a integridade do sinal e gera *outliers* geográficos. Esses pontos ruidosos, ao serem integrados no cálculo da distância total, resultam em um somatório artificialmente superior ao deslocamento efetivo do veículo.

Figura 58 - Tempo final apresentado pelo sistema web

Fonte: (O autor, 2026)

Diante da Figura 58, observa-se que o módulo GNSS sofreu influência significativa das condições climáticas adversas, registrando aproximadamente 6 km adicionais durante o período de chuva. De acordo com os dados obtidos no arquivo .csv, o percurso teve início às 16h20min, em condições climáticas inicialmente estáveis, conforme ilustrado na Figura 59.

Figura 59 - Trajeto inicial registrado pelo módulo

Lat	Lon	DistPerco	DistTotal	Hectare	Sat	Vel(km/h)	HDOP	Data	Hora	Tempo(min)
-18.9353	-48.2457	0	0	0	11	0.26	1.18	21/02/2026	16:19:50	0
-18.9353	-48.2457	0	0	0	11	0.04	1.18	21/02/2026	16:19:54	0
-18.9353	-48.2457	0	0	0	11	0.04	1.18	21/02/2026	16:19:57	0
-18.9353	-48.2457	0	0	0	11	0.07	1.18	21/02/2026	16:20:01	0
-18.9353	-48.2457	0	0	0	11	0.06	1.18	21/02/2026	16:20:04	0
-18.9353	-48.2457	0	0	0	11	0.07	1.18	21/02/2026	16:20:08	0
-18.9353	-48.2457	0	0	0	12	0.04	1.14	21/02/2026	16:20:11	0
-18.9353	-48.2457	0	0	0	12	0.15	1.14	21/02/2026	16:20:15	0
-18.9353	-48.2457	0	0	0	12	3.93	1.14	21/02/2026	16:20:18	0
-18.9353	-48.2458	0.006	0.006	0	12	13.61	1.23	21/02/2026	16:20:22	0
-18.9351	-48.2458	0.021	0.021	0.01	12	25.41	1.18	21/02/2026	16:20:25	0
-18.9349	-48.2457	0.045	0.045	0.03	12	42.95	1.18	21/02/2026	16:20:29	0
-18.9345	-48.2457	0.08	0.08	0.06	11	50.28	1.18	21/02/2026	16:20:32	0
-18.934	-48.2456	0.127	0.127	0.09	11	55.93	1.25	21/02/2026	16:20:36	0
-18.9335	-48.2455	0.173	0.173	0.12	12	57.5	1.25	21/02/2026	16:20:39	0
-18.933	-48.2454	0.23	0.23	0.16	12	57.13	1.15	21/02/2026	16:20:43	1
-18.9325	-48.2453	0.279	0.279	0.2	12	54.23	1.15	21/02/2026	16:20:46	1
-18.9321	-48.2452	0.37	0.37	0.26	12	54.13	0.9	21/02/2026	16:20:50	1
-18.9316	-48.2452	0.415	0.415	0.29	12	50.97	0.92	21/02/2026	16:20:53	1
-18.9312	-48.2451	0.457	0.457	0.32	12	40.87	0.88	21/02/2026	16:20:57	1
-18.9309	-48.245	0.497	0.497	0.35	12	27.82	0.89	21/02/2026	16:21:00	1
-18.9306	-48.245	0.521	0.521	0.36	12	17.87	0.95	21/02/2026	16:21:04	1
-18.9305	-48.2449	0.54	0.54	0.38	12	20.45	0.89	21/02/2026	16:21:07	1
-18.9306	-48.2447	0.555	0.555	0.39	12	28.45	0.89	21/02/2026	16:21:11	1
-18.9306	-48.2444	0.58	0.58	0.41	12	33.93	0.89	21/02/2026	16:21:14	1
-18.9307	-48.2441	0.612	0.612	0.43	12	29.02	0.91	21/02/2026	16:21:18	1
-18.9307	-48.2439	0.643	0.643	0.45	12	14.89	0.93	21/02/2026	16:21:21	1
-18.9307	-48.2438	0.662	0.662	0.46	12	14.04	0.89	21/02/2026	16:21:25	1
-18.9307	-48.2436	0.668	0.668	0.47	12	22.19	0.88	21/02/2026	16:21:28	1
-18.9308	-48.2434	0.683	0.683	0.48	12	33.65	0.89	21/02/2026	16:21:32	1
-18.9308	-48.2431	0.71	0.71	0.5	12	37.04	0.89	21/02/2026	16:21:35	1
-18.9309	-48.2427	0.745	0.745	0.52	12	27.21	0.88	21/02/2026	16:21:39	1
-18.9309	-48.2426	0.795	0.795	0.56	12	3.15	0.93	21/02/2026	16:21:42	2
-18.9309	-48.2425	0.801	0.801	0.56	12	19.33	0.93	21/02/2026	16:21:46	2
-18.9307	-48.2425	0.816	0.816	0.57	12	32.52	0.88	21/02/2026	16:21:49	2
-18.9304	-48.2424	0.842	0.842	0.59	12	30.71	0.88	21/02/2026	16:21:53	2
-18.9302	-48.2423	0.873	0.873	0.61	12	35	0.88	21/02/2026	16:21:56	2

Fonte: (O autor, 2026)

Cabe destacar que, durante o trajeto, houve uma parada no acostamento devido à intensidade da chuva, a qual comprometeu significativamente a visibilidade. O veículo permaneceu estacionado por alguns minutos. Nesse intervalo, verificou-se elevada oscilação nos dados do GPS, incluindo o registro de velocidades incompatíveis com a realidade, atingindo valores superiores a 440 km/h. Esse comportamento evidencia a ocorrência de erros significativos de posicionamento,

possivelmente associados às condições atmosféricas adversas e às limitações do sinal GNSS, conforme ilustrado na Figura 60.

Figura 60 - Dados oscilatórios obtidos pelo NEO-M8N

-18.9085	-48.2201	7.507	7.507	5.25	10	20.48	0.93	21/02/2026	16:30:59	10.6
-18.9084	-48.2201	7.526	7.526	5.27	9	18.85	1.16	21/02/2026	16:31:02	10.6
-18.9083	-48.22	7.54	7.54	5.28	12	18.83	0.89	21/02/2026	16:31:05	10.6
-18.9082	-48.2198	7.557	7.557	5.29	10	24.15	1.26	21/02/2026	16:31:09	10.6
-18.9081	-48.2197	7.594	7.594	5.32	9	12.39	1.82	21/02/2026	16:31:15	10.6
-18.908	-48.2195	7.617	7.617	5.33	8	22.02	1.36	21/02/2026	16:31:20	11.64
-18.9078	-48.219	7.668	7.668	5.37	7	36.08	1.47	21/02/2026	16:31:24	11.64
-18.9074	-48.219	7.704	7.704	5.39	9	52.26	1.05	21/02/2026	16:31:27	11.64
-18.9069	-48.2191	7.758	7.758	5.43	12	72.82	0.85	21/02/2026	16:31:30	11.64
-18.9061	-48.2192	7.842	7.842	5.49	10	85.99	1.44	21/02/2026	16:31:34	11.64
-18.9054	-48.2193	7.916	7.916	5.54	8	91.21	1.58	21/02/2026	16:31:37	11.64
-18.9047	-48.2196	7.999	7.999	5.6	9	102.34	1.91	21/02/2026	16:31:41	11.64
-18.9031	-48.22	8.132	8.132	5.69	10	135.53	1.22	21/02/2026	16:31:45	11.64
-18.9017	-48.2205	8.284	8.284	5.8	10	163.48	1.47	21/02/2026	16:31:49	11.64
-18.9	-48.2209	8.561	8.561	5.99	8	180.53	1.91	21/02/2026	16:31:55	11.64
-18.8984	-48.2212	8.74	8.74	6.12	7	226.83	1.75	21/02/2026	16:32:06	11.64
-18.8905	-48.2225	9.627	9.627	6.74	8	305.19	1.36	21/02/2026	16:32:15	11.64
-18.8833	-48.2228	10.364	10.364	7.25	8	347.21	1.12	21/02/2026	16:32:19	11.64
-18.8804	-48.2228	10.749	10.749	7.52	9	389.75	1.74	21/02/2026	16:32:24	12.64
-18.8751	-48.2227	11.34	11.34	7.94	11	410.24	1.66	21/02/2026	16:32:30	12.64
-18.8717	-48.2228	11.72	11.72	8.2	7	409.13	1.99	21/02/2026	16:32:35	12.64
-18.8638	-48.2233	12.601	12.601	8.82	10	426.48	1.66	21/02/2026	16:32:38	12.64
-18.862	-48.2234	12.803	12.803	8.96	9	440.5	1.95	21/02/2026	16:32:46	12.64
-18.8699	-48.2237	13.687	13.687	9.58	6	3	1.93	21/02/2026	16:39:14	19.54
-18.9034	-48.2242	17.406	17.406	12.18	9	3.54	1.39	21/02/2026	16:39:18	19.54
-18.9033	-48.2242	17.406	17.406	12.18	9	5.46	1.08	21/02/2026	16:39:21	19.54
-18.9033	-48.2242	17.412	17.412	12.19	7	5	1.86	21/02/2026	16:39:25	19.54
-18.9033	-48.2241	17.419	17.419	12.19	8	4.17	1.57	21/02/2026	16:39:45	19.54
-18.9031	-48.224	17.44	17.44	12.21	8	1.8	1.57	21/02/2026	16:39:49	19.54
-18.9031	-48.224	17.44	17.44	12.21	9	1.46	1.34	21/02/2026	16:39:52	19.54
-18.9031	-48.224	17.44	17.44	12.21	6	1.41	1.94	21/02/2026	16:40:10	19.54

Fonte: (O autor, 2026)

O trajeto também foi analisado por meio do *software* Google Earth Pro, permitindo a visualização espacial dos pontos registrados. Inicialmente, os dados apresentaram coerência com o percurso realizado, conforme a Figura 61.

Figura 61 - Trajeto inicial do percurso pelo Google Earth

Fonte: (O autor, 2026)

Entretanto, nos momentos de maior intensidade da chuva, observou-se maior dispersão dos pontos geográficos registrados, evidenciando deslocamentos inconsistentes em relação à rota real percorrida, conforme destacado na área demarcada na Figura 62.

Figura 62 - Pontos flutuantes obtidos pelo módulo GNSS

Fonte: (O autor, 2026)

Este tópico tem como objetivo apresentar uma análise do comportamento do módulo GNSS NEO-M8N sob condições climáticas adversas, como tempestades com precipitação intensa. Verificou-se que a chuva pode comprometer significativamente a precisão do posicionamento, ocasionando erros de distância, velocidade e localização. Dessa forma, este estudo complementa a avaliação do sistema, evidenciando suas limitações operacionais em cenários reais de uso.

Na Figura 63 observa-se um trecho em que ocorre uma interrupção na obtenção dos dados de posicionamento. Esse comportamento está associado à presença de uma torre de comunicação nas proximidades, previamente identificada, a qual possivelmente interferiu na recepção do sinal GNSS. Além disso, as condições climáticas desfavoráveis, caracterizadas por céu nublado, podem ter contribuído para a degradação da qualidade do sinal e para a ocorrência de interferências durante o processo de aquisição dos dados.

Figura 63 - Influência da antena de sinal no módulo GNSS

Fonte: (O autor, 2026)

Vale salientar que, em testes realizados sob condições climáticas normais, não foi observada influência significativa dessa estrutura sobre o funcionamento do sistema. Entretanto, no presente experimento, conduzido sob condições atmosféricas adversas, verificou-se impacto perceptível na qualidade do posicionamento. Ao trafegar nas proximidades dessa antena, no mesmo trecho anteriormente percorrido, observou-se flutuação abrupta do ponto geográfico demarcado em vermelho, além de instantes em que o sistema deixou temporariamente de registrar a posição.

Tal comportamento pode ser atribuído à combinação de dois fatores: as condições climáticas fortemente nubladas, com ocorrência de chuva intensa, e possíveis interferências eletromagnéticas provenientes da antena de transmissão.

Esses elementos contribuíram para a degradação momentânea da qualidade do sinal GNSS, refletindo diretamente na estabilidade e precisão das medições obtidas.

4.5 Avaliação do Cálculo de Área (hectares)

O cálculo da área está diretamente relacionado ao cálculo da distância percorrida, uma vez que o maquinário agrícola opera em metros lineares e possui largura fixa de trabalho, não sendo possível ampliá-la durante a operação. Assim, a estimativa da área trabalhada pode ser obtida a partir do produto entre a distância percorrida e a largura útil do implemento.

Como exemplo, considere uma plantadeira de 10 linhas, com espaçamento de 0,45 m entre linhas, resultando em uma largura total de 4,5 m. Nesse caso, essa largura deve ser parametrizada no código para que a área seja calculada automaticamente. Supondo que a plantadeira percorra 10.792 metros lineares, a área trabalhada será:

$$Área = Distância \times Largura = 10.000 \times 4{,}5 = 45.000 \text{ m}^2 \quad (7)$$

Convertendo para hectares:

$$45.000 \text{ m}^2 = 4{,}5 \text{ ha}$$

Dessa forma, ao definir previamente a largura do implemento no sistema, a área pode ser calculada simplesmente multiplicando-se esse parâmetro pela distância acumulada. Ressalta-se que, em condições normais de operação, o maquinário não percorre duas vezes o mesmo trajeto, pois isso representaria retrabalho e desperdício operacional.

No presente projeto, foi adotada uma largura de trabalho de 7 metros, correspondente a uma plataforma de colheitadeira de aproximadamente 23 pés, considerada de pequeno porte no contexto da agricultura mecanizada, conforme pode ser observado na Figura 64.

Figura 64 - Definição do tamanho do equipamento

```
// Função para calcular distância percorrida
void printGPSDistancia() {
  static unsigned long lastMillis=0;      //guardar valor entre chamadas. Tem que ser static
  static unsigned long lastTime=millis();
  static float distancia = 0.0;          //guardar valor inicial
  static float course = 0.0;
  static int i = 0;
  static int tamanhoPlat=7; //<----------------------------------------------------------------------

  const float limiar_min = 3.0;          // mínimo de 3 metros
  float velocidade = gps.speed.mps();    // melhor usar m/s
```

Fonte: (O autor, 2026)

O cálculo da área é realizado internamente na mesma função responsável pelo cálculo da distância percorrida, conforme ilustrado na Figura 65. Essa abordagem foi adotada porque a referida função já incorpora filtros de validação, destinados a evitar o registro indevido de pontos quando o equipamento permanece parado.

O filtro implementado impede o armazenamento repetitivo de coordenadas praticamente idênticas, reduzindo o acúmulo de erro e evitando a superestimação da distância percorrida e, consequentemente, da área calculada.

Figura 65 - Cálculo do hectare percorrido.

```
    // Aplica filtro: considera apenas deslocamentos acima do limiar
      if(distancia>=limiar){

        distanciaPercorrida=distanciaPercorrida+(distancia*(0.001));
        distanciaTotal=distanciaTotal+(distancia*(0.001));

        hectare=hectare+((distancia*tamanhoPlat)*0.0001);

        lat2=lat1;
        lng2=lng1;
      }

    }
```

Fonte: (O autor, 2026)

Dessa forma, considerando o exemplo apresentado adiante, ao multiplicar a distância percorrida, 10.792 metros, pela largura operacional de 7 metros definida no escopo do código, obtém-se uma área total de 75.544 m². Convertendo esse valor para hectares (1 hectare = 10.000 m²), resulta-se em aproximadamente 7,55 hectares.

Verifica-se, portanto, que o valor calculado pelo sistema apresenta coerência com a área efetivamente trabalhada, evidenciando a consistência do método adotado para estimativa de área com base na distância percorrida e na largura operacional do implemento, como pode ser visto na Figura 66.

Figura 66 - Dados obtidos e calculados pelo código implementado no ESP32

Fonte: (O autor, 2026)

Conforme evidenciado na imagem anterior, os resultados obtidos indicam que o método de cálculo implementado apresenta coerência prática.

Entretanto, recomenda-se a realização de testes prolongados em condições reais de operação, utilizando o sistema embarcado em maquinário agrícola durante jornadas extensas de trabalho, a fim de validar sua estabilidade, confiabilidade e robustez ao longo do tempo.

De forma preliminar, os resultados indicam que o sistema está apto a atender pequenos produtores ou prestadores de serviço em fase inicial de atuação, especialmente aqueles que não dispõem de recursos financeiros para investimento em sistemas profissionais de agricultura de precisão.

4.6 Aquisição de Dados de Diferentes Trechos

O código desenvolvido permite a aquisição e o armazenamento de dados referentes a diferentes trechos de operação. Isso ocorre porque, ao desligar o ESP32, entende-se que o maquinário também foi desligado, encerrando-se assim um ciclo diário de trabalho. Em um cenário real, essa interrupção normalmente representa o término de um dia útil de operação, sendo retomado apenas no dia seguinte.

Dessa forma, torna-se necessário distinguir os trechos correspondentes a dias distintos de trabalho. Para isso, foi implementado um “marco zero”, responsável por sinalizar o reinício das medições após cada desligamento do sistema. Esse recurso possibilita identificar com precisão o rendimento operacional diário. A partir da imagem do servidor web apresentada na Figura 67, será possível compreender melhor essa lógica de funcionamento.

Figura 67 - Demonstração dos dados temporários e fixos

Fonte: (O autor, 2026)

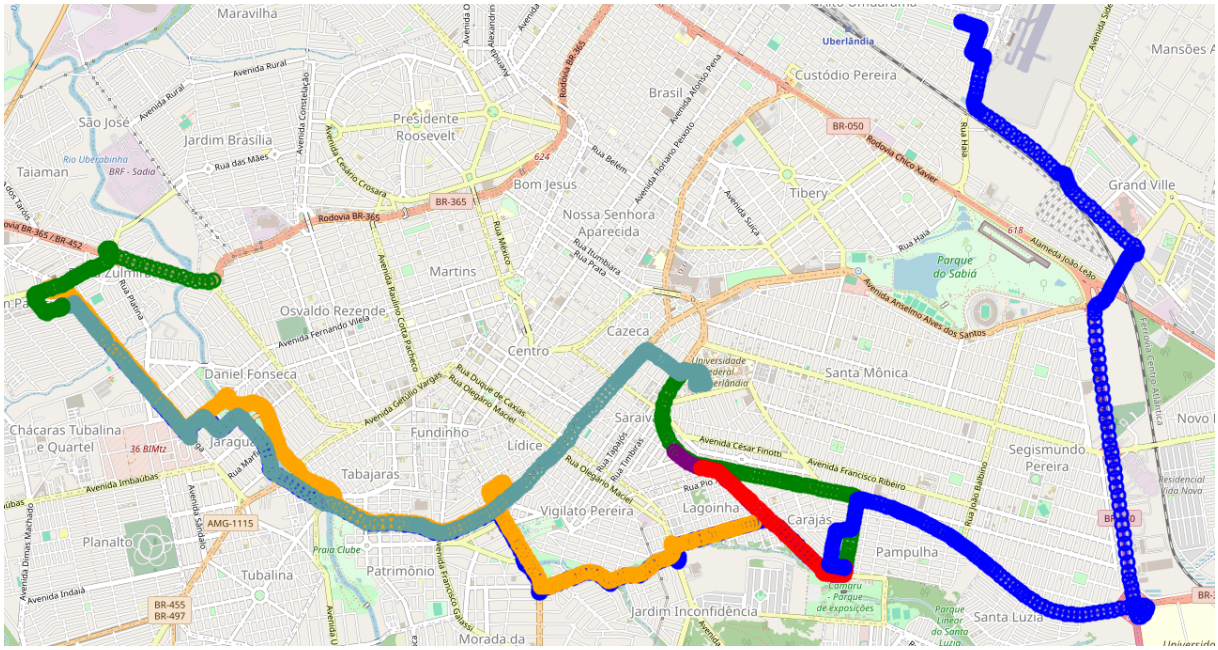
Observa-se, na interface do servidor web, a presença de uma linha divisória separando as informações de hectare, distância total e tempo decorrido das demais variáveis exibidas.

Essa separação foi implementada com o objetivo de diferenciar os dados referentes ao dia corrente daqueles acumulados ao longo de todo o período de utilização do sistema. Assim, as informações apresentadas na parte superior correspondem exclusivamente ao desempenho do dia específico (por exemplo, distância percorrida no dia X), enquanto o campo de distância total representa o valor acumulado desde o início da operação do sistema.

O mesmo princípio se aplica ao total de hectares trabalhados, permitindo ao operador verificar, ao final do expediente, uma estimativa básica da área efetivamente coberta, obtida pela multiplicação da distância percorrida pela largura operacional de 7 metros, conforme descrito anteriormente.

Para uma análise mais detalhada, os dados podem ser exportados no formato .csv e posteriormente importados para o Google Earth Pro ou para o código desenvolvido em Python, possibilitando a visualização completa dos trajetos percorridos e uma avaliação mais aprofundada do desempenho operacional do sistema, conforme ilustrado na Figura 68, cujos resultados serão discutidos em tópico posterior.

Figura 68 - Apresentação dos multipercursos fornecidos pelo código no ESP32



Fonte: (O autor, 2026)

Portanto, a definição de um marco zero torna-se fundamental para segmentar corretamente os períodos de trabalho, permitindo separar dias distintos de operação ou identificar momentos em que o equipamento foi desligado. Esse mecanismo contribui para uma organização mais precisa dos dados e para uma análise mais confiável do rendimento diário do sistema.

4.7 Análise do Servidor Web

Conforme apresentado nas figuras anteriores, o servidor web permite a visualização em tempo real de diversos parâmetros fornecidos pelo módulo u-blox

NEO-M8N, tais como número de satélites conectados, precisão estimada, velocidade instantânea, distância percorrida e área trabalhada (hectares).

Destacam-se como variáveis principais a distância total percorrida e o hectare acumulado em determinado trajeto, por representarem diretamente o desempenho operacional do maquinário. A interface também diferencia os dados referentes ao percurso atual, iniciados após a energização do sistema, dos dados cumulativos, que correspondem à soma de todos os trechos anteriormente registrados.

Além disso, o sistema dispõe de três funcionalidades principais por meio de botões dedicados:

- Resetar dados parciais: reinicia apenas os dados do percurso atual (distância, tempo decorrido e demais informações do dia);
- Resetar todos os dados: exclui completamente o arquivo de armazenamento, apagando o histórico acumulado;
- Baixar históricos: permite o *download* do arquivo de dados por qualquer dispositivo conectado à rede Wi-Fi configurada no sistema, onde o tempo de *download* vai depender do tamanho do arquivo gerado.

Tais recursos garantem agilidade na interação com o protótipo em ambiente real. Como evidenciado na Figura 69, a interface do servidor web embarcado dispõe de botões de comando específicos, permitindo que o operador execute funções de inicialização, salvamento e exportação de dados diretamente de um dispositivo móvel conectado ao ponto de acesso do ESP32.

Figura 69 - Botões existentes no servidor web

Fonte: (O autor, 2026)

A interface apresentada realiza atualização automática a cada 5 segundos, intervalo definido com o objetivo de reduzir o consumo de processamento e memória do ESP32, mantendo ao mesmo tempo uma visualização suficientemente dinâmica das informações, conforme pode ser visto na Figura 70.

Esse tempo de atualização pode ser facilmente alterado por meio da modificação de uma única linha no código-fonte, permitindo adequação conforme a necessidade do usuário.

Optou-se por uma interface simples e funcional, priorizando eficiência e baixo consumo de recursos computacionais. Dessa forma, evitou-se o uso de elementos gráficos excessivos ou layouts complexos que poderiam comprometer o desempenho do microcontrolador, especialmente considerando as limitações de memória e processamento do ESP32.

Figura 70 - Parte do código existente em HTML demonstrando o refresh

```
String html = "<!DOCTYPE html>";
html += "<html lang='pt-br'>";
html += "<head>";
html += "<meta charset='UTF-8'>";
html += "<meta name='viewport' content='width=device-width, initial-scale=1.0'>";
html += "<meta http-equiv='refresh' content='5'>";
html += "<title>Máquina 01 - Dados GPS</title>";
html += "<style>";
html += "body { font-family: Arial; background-color: #f0f2f5; padding: 20px; }";
html += "h1 { text-align: center; color: #333; }";
html += "div.container { max-width: 600px; margin: auto; background: white; padding: 20px; border-radius: 10px; box-shadow: 0 0 10px rgba(0,0,0,0.1); }";
html += "p { font-size: 1.1em; color: #555; }";
html += "strong { color: #000; }";
html += "button, a.button { padding: 10px 20px; font-size: 16px; text-decoration: none; border: none; border-radius: 5px; display: inline-block; margin: 5px 0; }";
html += "button { background-color: #f0ad4e; color: white; }";
html += "a.button { background-color: #5bc0de; color: white; }";
html += ".btn-reset-all { background-color: #c9302c; color: white; }";   // estilo do reset all
html += ".spaced { margin-top: 30px; display: block; }";               // cria espaço extra
html += "</style>";
html += "</head><body><div class='container'>";
html += "<h1>Máquina 01 - Dados GPS NEO-M8N</h1>";

// Dados GPS
html += "<p><strong>Satélites:</strong> " + String(gps.satellites.value()) + "</p>";
html += "<p><strong>Precisão:</strong> " + String(gps.hdop.hdop()) + "</p>";
html += "<p><strong>Velocidade (km/h):</strong> " + String(gps.speed.kmph(), 2) + "</p>";
html += "<p><strong>Data:</strong> " + String(gps.date.day()) + "/" + String(gps.date.month()) + "/" + String(gps.date.year()) + "</p>";
html += "<p><strong>Hora (UTC-3):</strong> " + String(gps.time.hour() - 3) + ":" + String(gps.time.minute()) + ":" + String(gps.time.second()) + "</p>";
html += "<p><strong>Curso:</strong> " + String(courseCard) + "</p>";
html += "<p><strong>Distância percorrida (km):</strong> " + String(distanciaPercorrida, 3) + "</p>";
html += "<hr>";  // Ponto de separação visual antes dos dados “a partir de distancia percorrida”

html += "<p><strong>Hectare:</strong> " + String(hectare) + "</p>";
html += "<p><strong>Distância Total (km):</strong> " + String(distanciaTotal, 3) + "</p>";
html += "<p><strong>Tempo decorrido (min):</strong> " + String(tempoOp) + "</p>";

// Botão de reset parcial
html += "<form action='/reset' method='get'>";
html +=   "<button type='submit'>Resetar Dados</button>";
html += "</form>";

// Link para download
html += "<a class='button' href='/DownloadMaquina01.csv'>Baixar Histórico (.csv)</a>";
```

Fonte: (O autor, 2026)

No trecho denominado *content* do código HTML da figura 70, é possível modificar tanto o tempo de atualização automática da página (*refresh*) quanto personalizar completamente a estrutura visual da interface, caso se deseje uma apresentação mais elaborada.

Contudo, o foco do sistema não está na estética, mas sim na funcionalidade e na praticidade operacional. A solução desenvolvida permite acesso, visualização e *download* dos dados sem a necessidade de conexão física do ESP32 a um notebook ou computador.

Essa característica torna o sistema especialmente adequado para aplicações em campo, onde o acesso à internet convencional é limitado ou inexistente. Assim, a comunicação via rede local (*Access Point*) facilita significativamente o processo de aquisição e extração de dados em ambientes rurais ou de difícil acesso.

4.8 Python Para Visualização dos Trajetos

Conforme descrito no capítulo de Metodologia, a linguagem Python mostrou-se uma ferramenta extremamente eficiente para a visualização dos diferentes trajetos realizados ao longo de vários dias de operação, seja em atividades de colheita ou plantio. Considerando que o *hardware* é desligado ao final de cada jornada de trabalho e reiniciado no dia seguinte, torna-se fundamental identificar com precisão o momento de início e término de cada trecho, permitindo determinar o tempo efetivo de operação, a distância percorrida e a área trabalhada em hectares.

Diferentemente dos dados temporários apresentados no servidor web, o processamento em Python possibilita uma análise mais estruturada e detalhada. Utilizando o Visual Studio Code como ambiente de desenvolvimento, foi possível integrar a linguagem Python às bibliotecas pandas e folium, permitindo a leitura do arquivo .csv, a realização de cálculos e a geração de mapas interativos. Essa abordagem evita o processamento excessivo no ESP32, preservando memória e desempenho do microcontrolador.

Para a representação cartográfica dos dados, foram implementadas três modalidades de visualização (tilesets) através da biblioteca Folium: Satélite, Padrão (OpenStreetMap) e Claro. Essa variedade permite ao operador alternar entre diferentes níveis de detalhamento conforme a necessidade da análise. A Figura 70 ilustra a interface utilizando a camada de Satélite, que facilita a correlação das coordenadas coletadas com a morfologia e as infraestruturas reais do terreno monitorado.

Figura 71 - Mapa tipo Satélite fornecido pela biblioteca Folium

Fonte: (O autor, 2026)

Como mencionado anteriormente, a imagem apresentada refere-se ao mapa no modo satélite, cuja finalidade é proporcionar uma visualização mais realista da área trabalhada. Esse modo facilita a identificação visual do terreno e permite a utilização de ferramentas da biblioteca empregada no sistema, como o desenho manual de polígonos para estimativa da área total em hectares.

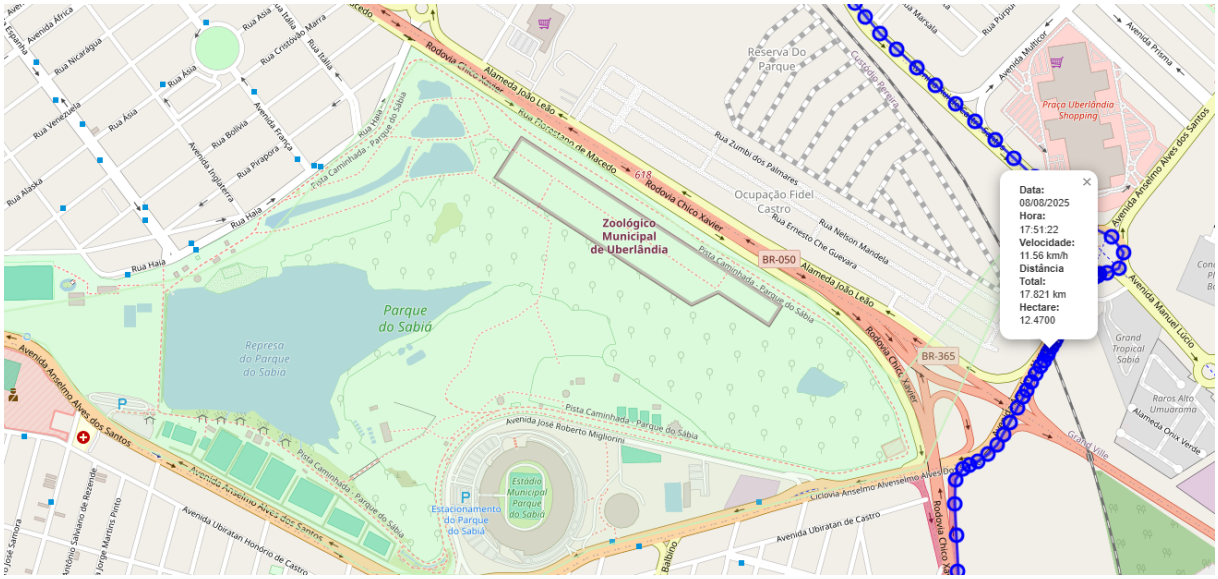
Dessa forma, torna-se possível comparar a área estimada visualmente com aquela calculada pelo sistema GPS, conforme ilustrado na Figura 72. Nessa figura, observa-se uma área demarcada em verde e, no canto superior direito, um campo de informações que apresenta o tamanho da área tanto em hectares quanto em acres.

Figura 72 - Demarcação de área no mapa interativo fornecido pela biblioteca Folium

Fonte: (O autor, 2026)

O mapa no modo padrão apresenta as informações cartográficas convencionais, oferecendo equilíbrio entre detalhamento e clareza visual. Essa visualização permite analisar o trajeto percorrido sem comprometer a identificação das vias ou áreas do mapa, conforme pode ser visto na Figura 73.

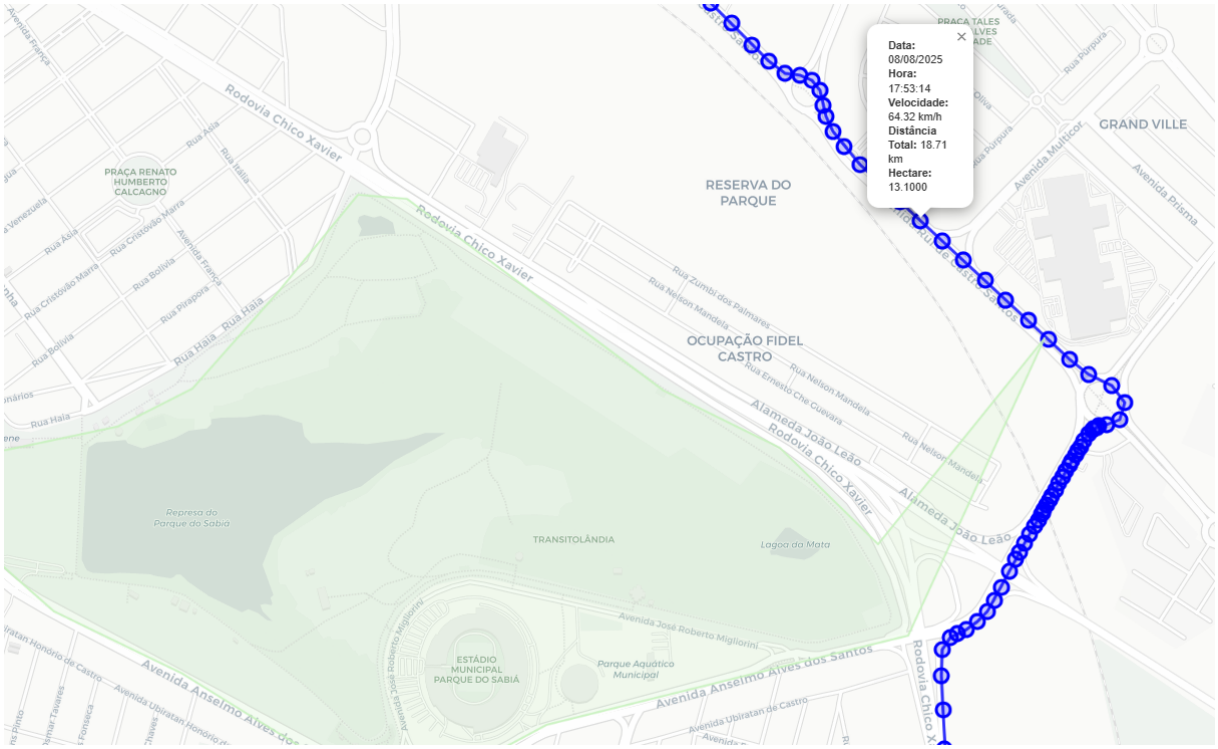
Figura 73 - Mapa tipo Padrão fornecido pela biblioteca Folium



Fonte: (O autor, 2026)

O mapa no modo claro tem como objetivo destacar de forma mais evidente o trajeto percorrido, reduzindo os elementos visuais do plano de fundo e direcionando a atenção do observador exclusivamente para o percurso realizado, conforme ilustrado na Figura 74.

Figura 74 - Mapa tipo Claro fornecido pela biblioteca Folium



Fonte: (O autor, 2026)

A biblioteca Folium, em conjunto com a linguagem Python, permite a criação de diversos elementos de visualização, como marcadores personalizados, linhas representando trajetos, pop-ups informativos e indicadores de início de percurso. Esses recursos possibilitam uma análise mais detalhada de cada trecho percorrido, assim como ilustrado na Figura 75.

Figura 75 - Pop-ups de início contendo informações do início do trecho

Fonte: (O autor, 2026)

Além da marcação do ponto inicial, é possível destacar o ponto final de cada trajeto e, subsequentemente, indicar o início de um novo trecho, facilitando a separação visual entre diferentes dias ou períodos de operação, como pode ser visualizado na Figura 76.

Figura 76 - Pop-ups de fim contendo informações do fim do trecho

Fonte: (O autor, 2026)

Portanto, a utilização do Python como ferramenta complementar ao projeto amplia significativamente as possibilidades de análise e validação dos dados. Diferentemente da importação direta para o Google Earth Pro, onde os dados brutos não passam por filtragem prévia, o uso do Python permite organizar, segmentar e apresentar as informações de maneira estruturada.

Com o Folium, é possível configurar o mapa para exibir início e fim de percurso, distância percorrida, área trabalhada em hectares, tempo de operação, datas e horários, tornando a análise mais objetiva e confiável.

Enquanto no Google Earth Pro é necessário realizar etapas adicionais, como ajustar a formatação do arquivo CSV, configurar separadores e definir corretamente as colunas de latitude e longitude, no Python basta executar o código previamente desenvolvido e indicar o caminho do arquivo baixado. O mapa é então gerado automaticamente conforme programado, facilitando a visualização e interpretação dos

dados, especialmente em aplicações voltadas à verificação de área colhida, tempo de operação e produtividade diária.

4.9 Limitações Identificadas

Diante dos resultados obtidos, observa-se que diversos fatores influenciam significativamente a qualidade dos dados adquiridos pelo módulo GNSS NEO-M8N. Entre eles, destacam-se as condições climáticas, evidenciando-se melhor desempenho em situações de céu aberto ou parcialmente nublado. Em contrapartida, condições de forte nebulosidade e chuva intensa impactaram de maneira expressiva a estabilidade e a precisão das medições, conforme verificado nos testes realizados.

Adicionalmente, constatou-se a presença de ruído mesmo com o veículo em repouso, caracterizado por oscilações de velocidade inferiores a 1 km/h. Esse comportamento indica a necessidade de aprimoramento dos filtros implementados, a fim de tornar o sistema mais robusto. Como possíveis melhorias, sugerem-se: aumento do tempo mínimo para validação das amostras; ampliação do tamanho do *buffer* do filtro de média móvel; redução do limite máximo de HDOP para valores mais restritivos, como 1,5; e ajuste do limiar mínimo dinâmico baseado na velocidade, podendo-se considerar 60% da velocidade medida ou um deslocamento mínimo de 4 metros. Tais ajustes tendem a aumentar a confiabilidade dos dados, especialmente em aplicações agrícolas, nas quais o maquinário opera predominantemente em trajetórias lineares, com exceção das manobras de cabeceira.

Verificou-se também que antenas de transmissão de rádio ou telefonia podem influenciar negativamente a recepção dos sinais GNSS. Conforme observado na avaliação realizada sob condição de chuva, houve flutuação significativa do ponto geográfico ao trafegar nas proximidades de uma antena, indicando possível interferência eletromagnética. Soma-se a isso o ambiente urbano, onde a presença de edificações, veículos e viadutos favorece a ocorrência de multipercurso (reflexão do sinal), contribuindo para oscilações e degradação momentânea da precisão do posicionamento.

5 CONCLUSÃO

O presente trabalho teve como objetivo o desenvolvimento e a validação de um sistema de monitoramento de distância percorrida e cálculo de área em hectares utilizando o módulo GNSS NEO-M8N aliado ao microcontrolador ESP32, com aplicação voltada ao contexto agrícola.

A partir dos testes realizados em diferentes condições ambientais, incluindo céu aberto, nublado, ambiente urbano e condições adversas como chuva intensa, foi possível avaliar o comportamento do sistema quanto à precisão, estabilidade e confiabilidade das medições. Observou-se que, em condições climáticas favoráveis, o módulo apresentou desempenho satisfatório, com erro percentual inferior a 1% quando comparado a medições realizadas por meio do Google Earth Pro, demonstrando coerência e consistência nos resultados obtidos.

A aplicação de critérios de filtragem, tais como número mínimo de satélites, limitação de HDOP, média móvel e limiar mínimo de deslocamento baseado na velocidade, mostrou-se fundamental para a redução de ruídos e flutuações, tornando o sistema mais robusto e adequado para aplicações práticas em campo. Verificou-se que, sem a aplicação desses filtros, a dispersão inicial e os erros acumulados poderiam comprometer significativamente a confiabilidade do cálculo de distância e, consequentemente, da área trabalhada.

Em contrapartida, constatou-se que condições atmosféricas severas e ambientes urbanos com obstáculos e possíveis efeitos de multipercurso podem impactar negativamente a qualidade do sinal GNSS, gerando discrepâncias nas medições. Tais limitações evidenciam que, embora o sistema apresente desempenho satisfatório para aplicações de baixo custo e menor exigência de precisão, ele não substitui sistemas agrícolas profissionais com correção diferencial (RTK), que operam com precisão centimétrica.

Quanto ao cálculo de área, verificou-se que o modelo baseado na multiplicação da distância percorrida pela largura do implemento agrícola apresentou resultados coerentes com a prática operacional, especialmente em cenários de deslocamento linear e sem sobreposição de trajetos.

Portanto, conclui-se que o sistema desenvolvido atende ao propósito proposto, configurando-se como uma solução de baixo custo e viável para pequenos produtores ou prestadores de serviço que necessitam de uma estimativa confiável de distância e área trabalhada, sem a necessidade de investimento em equipamentos agrícolas de alto valor agregado.

Como trabalhos futuros, sugere-se:

- Aplicação prática contínua em maquinário agrícola por períodos prolongados para validação estatística ampliada;
- Realização de uma placa de circuito impresso para melhor acomodação do sistema;
- Alocar uma fonte que contenha regulador de tensão para fornecer a energia necessária para o sistema; e
- Inclusão de um módulo de Cartão Micro SD para o salvamento dos dados.
- Implementação do Filtro Kalman.

Dessa forma, o presente trabalho contribui ao demonstrar que soluções baseadas em GNSS de baixo custo, quando associadas a técnicas adequadas de filtragem e processamento, podem alcançar resultados satisfatórios para aplicações reais no setor agrícola.

REFERÊNCIAS

ADVANCED NAVIGATION. Georeferencing. Disponível em: https://www.advancednavigation.com/glossary/georeferencing/. Acesso em: 29 out. 2025.

ARDUINO. Arduino UNO Rev3 – Technical specifications. Disponível em: https://docs.arduino.cc/hardware/uno-rev3/. Acesso em: 29 out. 2025.

ARM. Embedded system design. Disponível em: https://www.arm.com/glossary/embedded-system-design. Acesso em: 29 out. 2025.

CURTO CIRCUITO. Módulo GPS NEO-M8N. Disponível em: https://curtocircuito.com.br/modulo-gps-neo-m8n.html. Acesso em: 29 out. 2025.

ESPRESSIF SYSTEMS. ESP32 Datasheet. Disponível em: https://documentation.espressif.com/esp32_datasheet_en.pdf. Acesso em: 29 out. 2025.

ESPRESSIF SYSTEMS. *ESP32-WROOM-32 Datasheet.* Disponível em: https://documentation.espressif.com/esp32-wroom-32_datasheet_en.pdf. Acesso em: 29 nov. 2025.

FOLIUM. Folium: Python Data Visualization Library. Disponível em: https://python-visualization.github.io/folium/. Acesso em: 15 dez. 2025.

FRONTIERS IN PHYSICS. Artigo científico. 2022. Disponível em: https://www.frontiersin.org/journals/physics/articles/10.3389/fphy.2022.1071539/full. Acesso em: 29 out. 2025.

GPS WORLD. What exactly is GPS NMEA data? Disponível em: https://www.gpsworld.com/what-exactly-is-gps-nmea-data/. Acesso em: 25 jan. 2026.

IBM. What is a microcontroller? Disponível em: https://www.ibm.com/think/topics/microcontroller. Acesso em: 15 fev. 2026.

IFORCE2D. GPS module comparison: NEO-6M, NEO-M8N, NEO-M8P, ZED-F9P. 2021. Disponível em: https://www.youtube.com/watch?v=abMoRJ-1ynY. Acesso em: 05 jul. 2025.

JOHN DEERE. Monitor Universal Gen 4 4240. [S. l.], 2025. Disponível em: https://www.deere.com.br/assets/images/region-3/products/technology-products/precision-ag-technology/displays-and-receivers-and-rtk/gen-4-4240-universal/r4f024678_monitor_gen_4240_large_f9dd0dba56aa6b8e695fabbf23d2e408fcd5adc0.jpg. Acesso em: 3 set. 2025.

MERCADO LIVRE. Módulo ESP32-WROOM-32D WiFi Bluetooth para IoT. Disponível em: https://www.mercadolivre.com.br/modulo-esp32-wroom-32d-wifi-bluetooth-para-iot/p/MLB2053067268. Acesso em: 16 mar. 2026.

NASA. Global Positioning System (GPS). Disponível em: https://www.nasa.gov/directorates/somd/space-communications-navigation-program/gps/. Acesso em: 29 out. 2025.

NASA. Sputnik and the dawn of the space age. Disponível em: https://www.nasa.gov/history/sputnik/sputorig.html. Acesso em: 15 fev. 2026.

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY (NIST). How do you measure your location using GPS? Disponível em: https://www.nist.gov/how-do-you-measure-it/how-do-you-measure-your-location-using-gps. Acesso em: 15 fev. 2026.

NOVATEL. What are Global Navigation Satellite Systems (GNSS)? Disponível em: https://novatel.com/tech-talk/an-introduction-to-gnss/what-are-global-navigation-satellite-systems-gnss. Acesso em: 15 fev. 2026.

PYTHON SOFTWARE FOUNDATION. Python Documentation. Disponível em: https://www.python.org/doc/. Acesso em: 2025.

RANDOM NERD TUTORIALS. ESP32 Flash Memory. Disponível em: https://randomnerdtutorials.com/esp32-flash-memory/. Acesso em: 29 out. 2025.

RANDOM NERD TUTORIALS. ESP32 Web Server – Beginner's Guide. Disponível em: https://randomnerdtutorials.com/esp32-web-server-beginners-guide/. Acesso em: 29 out. 2025.

RANDOM NERD TUTORIALS. ESP32 Wi-Fi Manager (AsyncWebServer). Disponível em: https://randomnerdtutorials.com/esp32-wi-fi-manager-asyncwebserver/. Acesso em: 29 out. 2025.

RESEARCHGATE. Prototype of Automatic Tractor Control Navigation System Using ESP32 Microcontroller. Disponível em: https://www.researchgate.net/publication/374339383_Prototype_of_Automatic_Tractor_Control_Navigation_System_Using_ESP_32_Microcontroller. Acesso em: 29 out. 2025.

UNIVERSIDADE ESTADUAL PAULISTA (UNESP). Revista Energia na Agricultura. Disponível em: https://revistas.fca.unesp.br/index.php/energia/article/view/3630/2651. Acesso em: 29 out. 2025.

UNIVERSIDADE ESTADUAL PAULISTA (UNESP). Trabalho acadêmico. Disponível em: https://www.inscricoes.fmb.unesp.br/upload/trabalhos/201652417328.pdf. Acesso em: 29 out. 2025.

UNIVERSIDADE FEDERAL DE UBERLÂNDIA (UFU). Aplicação Android com recursos GPS. Uberlândia: UFU, [s.d.]. Disponível em: https://repositorio.ufu.br/bitstream/123456789/26147/3/AplicacaoAndroidRecurso.pdf. Acesso em: 29 out. 2025.

UNIVERSIDADE FEDERAL DE UBERLÂNDIA (UFU). Comportamento de modelos de cálculo. Uberlândia: UFU, [s.d.]. Disponível em:

https://repositorio.ufu.br/bitstream/123456789/18133/4/ComportamentoModelosCalculo.pdf. Acesso em: 29 out. 2025.

UNIVERSIDADE FEDERAL DE UBERLÂNDIA (UFU). Projeto de uma placa de desenvolvimento. Uberlândia: UFU, [s.d.]. Disponível em: https://repositorio.ufu.br/bitstream/123456789/36672/3/ProjetoPlacaDesenvolvimento.pdf. Acesso em: 29 out. 2025.

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO (UFES). Trabalho de Conclusão de Curso. São Mateus: UFES, [s.d.]. Disponível em: https://fisica.saomateus.ufes.br/sites/fisica.saomateus.ufes.br/files/field/anexo/tcc_denis_apolonio_santana.pdf. Acesso em: 29 out. 2025.

UNIVERSIDADE FEDERAL DO RIO GRANDE (FURG). Trabalho acadêmico. Rio Grande: FURG, [s.d.]. Disponível em: https://repositorio.furg.br/bitstream/handle/1/9890/GPS%20Vermelho%20-%20com%20capa.pdf. Acesso em: 29 out. 2025.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL (UFRGS). Trabalho acadêmico. Porto Alegre: UFRGS. Disponível em: https://lume.ufrgs.br/bitstream/handle/10183/230016/001131652.pdf. Acesso em: 29 out. 2025.

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ (UTFPR). Anais de congresso. 2018. Disponível em: https://repositorio.utfpr.edu.br/jspui/bitstream/1/14617/1/PB_COENC_2018_2_15.pdf. Acesso em: 29 out. 2025.

UNITED STATES. Department of Defense. Global Positioning System: Standard Positioning Service (SPS) Performance Standard. 5. ed. Washington, DC: U.S. Department of Defense, abr. 2020. Disponível em: https://www.gps.gov/sites/default/files/2025-07/2020-SPS-performance-standard.pdf. Acesso em: 25 nov. 2026.

WEISSTEIN, Eric W. Haversine Formula. MathWorld – Wolfram Research. Disponível em: https://mathworld.wolfram.com/Haversine.html. Acesso em: 15 jan. 2026.