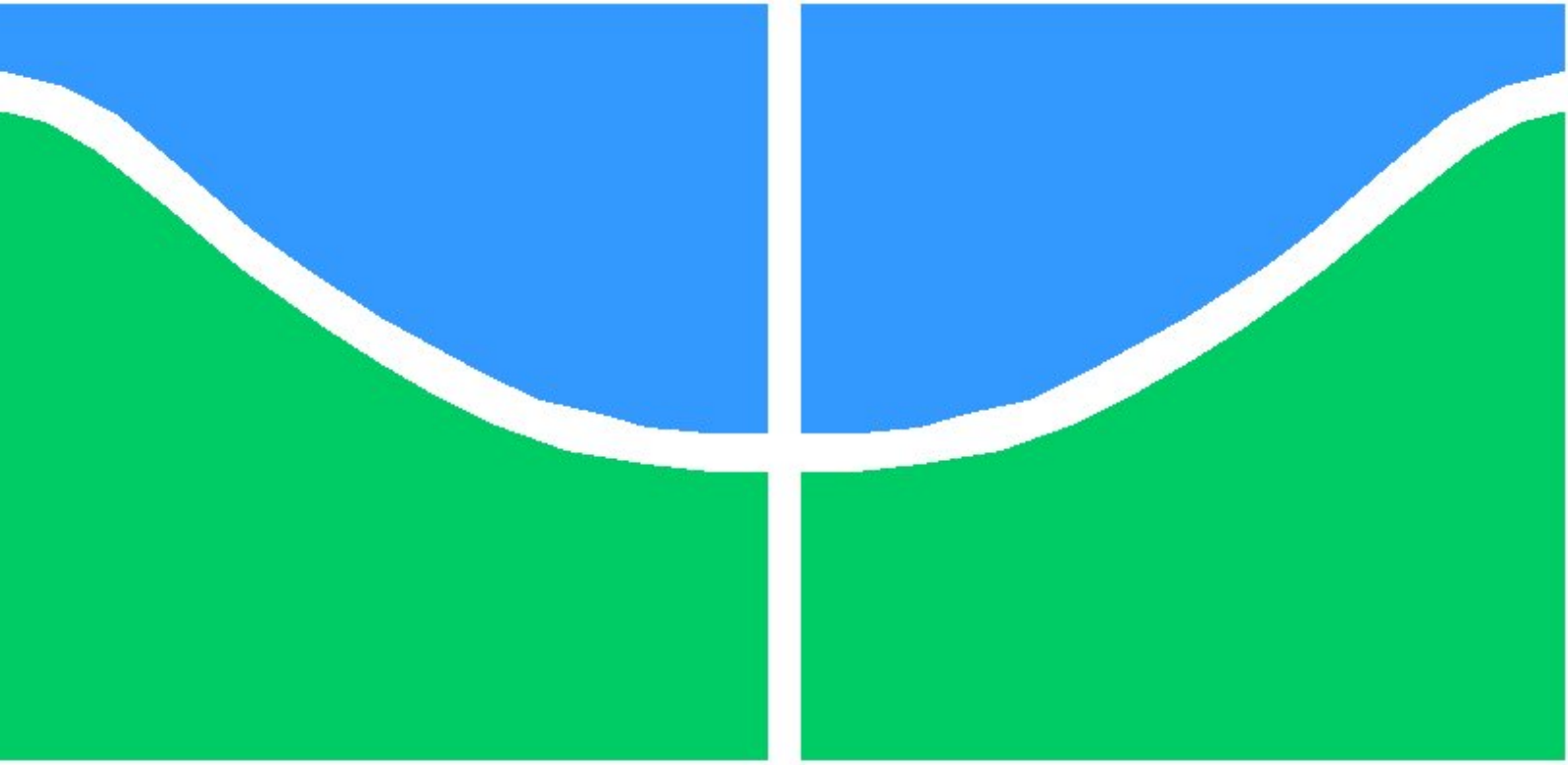

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Engenharia Eletrônica

Protótipo de uma plataforma móvel baseada em Android para monitoramento de parâmetros de qualidade da água do lago Paranoá

Autor: Felipe Duarte Machado e Lucas Hélion Santana de Souza

Orientador: Dr. Daniel Mauricio Muñoz Arboleda

Brasília, DF

2015

Felipe Duarte Machado e Lucas Hélion Santana de Souza

Protótipo de uma plataforma móvel baseada em Android para monitoramento de parâmetros de qualidade da água do lago Paranoá

Monografia submetida ao curso de graduação em (Engenharia Eletrônica) da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em (Engenharia Eletrônica).

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Dr. Daniel Mauricio Muñoz Arboleda

Coorientador: Dr. Manoel Fernando Tenório

Brasília, DF

2015

Felipe Duarte Machado e Lucas Hélion Santana de Souza

Protótipo de uma plataforma móvel baseada em Android para monitoramento de parâmetros de qualidade da água do lago Paranoá/ Felipe Duarte Machado e Lucas Hélion Santana de Souza. – Brasília, DF, 2015-

93 p. : il. (algumas color.) ; 30 cm.

Orientador: Dr. Daniel Mauricio Muñoz Arboleda

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2015.

1. Veiculo Autônomo de Superfície. 2. Estimação de parâmetros da qualidade da água. I. Dr. Daniel Mauricio Muñoz Arboleda. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Protótipo de uma plataforma móvel baseada em Android para monitoramento de parâmetros de qualidade da água do lago Paranoá

CDU 02:141:005.6

Felipe Duarte Machado e Lucas Hélion Santana de Souza

Protótipo de uma plataforma móvel baseada em Android para monitoramento de parâmetros de qualidade da água do lago Paranoá

Monografia submetida ao curso de graduação em (Engenharia Eletrônica) da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em (Engenharia Eletrônica).

Trabalho aprovado. Brasília, DF, 20 de junho de 2015:

Dr. Daniel Mauricio Muñoz Arboleda
Orientador

Dr. André Murilo de Almeida Pinto
Convidado 1

Dr. Gerardo Antonio Idrobo Pizo
Convidado 2

Brasília, DF
2015

Resumo

A água do planeta sofre constantes transformações, se renova e é reutilizada. Uma das principais transformações que a água sofreu no último século é a crescente contaminação, problema que afeta especialmente as grandes áreas urbanas e zonas litorâneas. Em razão do aumento da importância da água para a segurança de populações e devido ao risco de contaminação decorrente das atividades humanas surge a necessidade de um controle mais rígido para o abastecimento público. Neste cenário, destaca-se a importância do monitoramento da qualidade da água para a gestão dos recursos hídricos. O presente trabalho apresenta uma proposta de desenvolvimento de um protótipo de uma plataforma móvel aquática controlada remotamente com o objetivo de possibilitar a obtenção de parâmetros relevantes na estimação da qualidade da água de forma remota. Como prova de conceito foi escolhido escolhido apenas um desses parâmetros, em particular, foram aferidas medições da temperatura da água. No desenvolvimento do trabalho são investigados os parâmetros para estimar a qualidade da água, princípios de um ASV (Autonomous Surface Vehicle), utilização de sensores embarcados em um dispositivo móvel com sistema operacional Android, filtragem de sinais através do filtro media móvel, microcontrolador Arduino ADK e integração de aplicações utilizando um *webservice*. Como resultados foram obtidos um aplicativo para plataforma Android que acessa alguns sensores do aparelho *smartphone*, além de comunicar-se com um microcontrolador via USB o aplicativo também comunica-se com um *webservice* implementado para este trabalho. Além do aplicativo e do *webservice* foi implementado um programa controlador interfaciado, escrito em linguagem C#, que se comunica com o aplicativo através do *webservice* e por fim foi obtido um protótipo para plataforma móvel. Finalmente, foram realizados testes da plataforma no lago Paranoá, verificando o comportamento do algoritmo de navegação, coletando amostras reais da temperatura da água do lago e validando os dados coletados pelo aplicativo Android, assim como dos dados armazenados na solução *webservice*.

Palavras-chaves: ASV. Android. Arduino ADK. Webservice. Navegação por posição.

Abstract

Water of planet undergoes constant transformation, being renewed and reused. At last century, one of the major transformations that water has suffered is the increasing contamination, problem that especially affects large urban areas and coastal areas. A hard control of water is more necessary in reason of importance increasing of water for the safety of people and risk of contamination from human activity. In this scenario, it highlights the importance of water quality monitoring for management of water resources. This work presents a proposal to develop a prototype of an aquatic mobile platform remotely controlled in order that can obtain water parameters, propolsing as proof of concept, the measurement of one water parameter. On development of work, parameters are investigated to estimate water quality, principles of ASV (Autonomous Surface Vehicle), use of embedded sensors at mobile device with Android operating system, signal filtering through mobile average filter, Arduino ADK microcontroller and application's integration using webservices. As results, were obtained an application for Android Plataform that reads some sensors of smartphone device, gets communication with Arduino ADK microcontroller through USB connection and gets communication with a webservice developed for this work. In addition, were obtained a graphical user interface writed in C# language (using .NET Plataform) that communicates to Android application through webservice. Finally, were obtained a aquatic mobile plataform prototype.

Key-words: ASV. Android. Arduino ADK. Web service. Position navigation.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

ADK	Acessory Development Kit
ANA	Agencia Nacional das águas
API	Application Programming Interface
ASV	Autonomous Surface Vehicles
DBO5,20	Demanda Bioquímica de Oxigênio
GPS	Global Positioning System
HTTP	Hypertext Transfer Protocol
IAP	Índice de Qualidade da Água Bruta para fins de Abastecimento Público
IB	Índice de Balneabilidade
IET	Índice de Estado Trófico
IMU	Inertial Measurement Unit
IQA	Índice de Qualidade da Água
ISTO	Índice de Substâncias Tóxicas e Organolépticas
IVA	Índice de Qualidade da Água para a Proteção da Vida Aquática
NED	North East Down
pH	potencial Hidrogeniônico
PID	Proporcional Integral Derivativo
SOAP	Simple Object Access Protocol
UDDI	Universal Description Discovery and Integration
URL	Uniform Resource Locator
XML	Extensible Markup Language
WSDL	Web Services Description Language

Sumário

1 Introdução

Neste capitulo introdutório estão apresentados a descrição do problema a ser atacado, justificativa do trabalho, objetivos que almeja-se alcançar, metodologia utilizada no desenvolvimento do trabalho e organização do documento.

1.1 Contextualização

A água é um recurso essencial para várias atividades humanas, como consumo doméstico, irrigação, transporte, lazer e uso industrial, e sua disponibilidade e qualidade estão diretamente relacionadas com o bem estar da população, a manutenção do meio ambiente e uma série de atividades econômicas como a agricultura e pecuária, industria alimentícia, turismo entre outros. Assim, a degradação da água traz uma série de impactos ambientais (perda da biodiversidade), sociais (racionamento e falta de água para uso doméstico, aumento de doenças de veiculação hídrica) e econômicos (aumento de custo de tratamento e abastecimento, perda de produtividade nas industrias) que tornam explícitos a relevância da água, em termos quantitativo e qualitativos.

Segundo a Agência Nacional de Águas (ANA, 2014) o Brasil possui 12% da disponibilidade de água doce superficial do mundo. Apesar do aspecto quantitativo, o aspecto qualitativo representa um fator limitante para o aproveitamento da água, uma vez os diversos usos da água requerem requisitos de qualidade que podem ou não serem atendidos, o que torna claro que conhecer a qualidade da água é um fator primordial para sua gestão. Desse modo, o monitoramento e avaliação das águas superficiais são fatores essenciais para uma gestão adequada dos recursos hídricos, permitindo a análise e caracterização das bacias hidrográficas.

1.2 Descrição do Problema

A avaliação da qualidade das águas superficiais brasileiras possui alguns fatores de dificuldade como: as proporções continentais do Brasil, a ausência de redes estaduais de monitoramento em alguns Estados, além da heterogeneidade dos métodos de análise e avaliação nas redes de monitoramento existentes (ANA, 2014).

A avaliação da qualidade da água incluem a análise de parâmetros físicos (turbidez, temperatura, resíduo total, condutividade elétrica), químicos (oxigênio dissolvido, pH, demanda bioquímica de oxigênio, alcalinidade total), biológicos (coliformes tolerantes, clorofila, fitoplâncton) e nutricionais (fósforo total, nitrogênio total). Essas análises reque-

rem expedições de corpo técnico dotados de sondas e amostradores que coletam amostras e as avaliam em laboratório (ANA, 2012). De modo geral, a avaliação de corpos d'água é realizado com auxilio de índices de qualidade da água que, utilizando diversos parâmetros citados, sintetizam em um único número várias informações. Porém, neste processo de síntese perde-se informação sobre o comportamento dos parâmetros analisados.

Cada tipo de utilização da água requer requisitos de qualidade distintos. A prática de esportes e atividades realizadas na água como natação, remo, entre outros, requerem uma boa qualidade da água no que diz repeito a poluentes que podem transmitir doenças. Em determinadas áreas do lago Paranoá a pratica desses tipos de esportes e outras atividades de lazer são bastante difundidas, como por exemplo, na Península dos ministros e Ermida Dom Bosco e pier o da Asa Norte, de modo que é necessário atender requisitos mínimos de qualidade nessas regiões para que tais atividades possam ser realizadas com segurança.

1.3 Justificativa

O processo de avaliação de águas é constituído, geralmente por três etapas: amostragem, armazenamento e análise das amostras e estudo dos fatores intervenientes em sua gestão. Sendo assim, há diversas desvantagens nesse processo que o tornam relativamente lento e dispendioso, tais como: (1) A necessidade de deslocamento de recursos humanos para os locais de coleta de amostras, amiúde distantes e de difícil acesso; (2) Dificuldade da aquisição de um amplo conjunto de dados necessários para a investigação da evolução temporal dos parâmetros desejados; (3) Durante a fase de armazenamento, a integridade da amostra pode não ser preservada devido a possíveis atividades biológicas, o que impede a realização de análises em condições reais, o que minimiza a representatividade dos valores obtidos (ANA, 2012).

A ideia da utilização de uma plataforma móvel de baixo custo ajuda a superar diversas dificuldades apresentadas. Todo o processo de coleta de dados é automatizado. Além de possibilitar medições de alguns parâmetros em tempo real de uma forma remota, a possibilidade de expandir as funcionalidades dessa plataforma acrescentando outros tipos de tecnologias para atender a mais necessidades como monitoramento e mapeamento de regiões, por exemplo. Dessa forma o uso de uma plataforma móvel com capacidade de comunicação remota torna-se uma solução apropriada para resolver o problema de monitoramento da qualidade da água do lago Paranoá.

1.4 Objetivos

O objetivo deste trabalho é desenvolver um protótipo de uma plataforma móvel que possibilite a estimação de alguns parâmetros da qualidade da água de determinadas zonas superficiais do Lago Paranoá, como prova de conceito neste trabalho foi escolhido apenas um parâmetro físico da água, a temperatura. Essa plataforma deve movimentar-se de modo autônomo na água, sendo controlada por uma estação remota.

1.4.1 Objetivos específicos

- Viabilizar a comunicação entre a plataforma móvel e a estação remota, de modo que o usuário na estação remota possa enviar comandos e receber informações da plataforma móvel;

- Implementar um programa controlador interfaciado que possa permitir a visualização da localização e trajetória da plataforma móvel, e o envio de comandos por parte do usuário;

- Implementar um filtro para o processo de aquisição de dados na leitura dos sensores embarcados no dispositivo Android, de modo a obter uma medição mais próxima do valor real;

- Projeto e desenvolvimento eletromecânico de um protótipo de plataforma móvel que permita a sua navegação através do GPS.

1.5 Metodologia

O presente trabalho usa conceitos das ciências aplicadas para o desenvolvimento da plataforma móvel proposta e utiliza uma metodologia experimental baseada na observação direta de um parâmetro físico (temperatura) da água do Lago Paranoá. No desenvolvimento deste trabalho foram utilizadas as metodologias *top-down* e *bottom-up* para fins de organização das fases do projeto. No início, utilizou-se a metodologia *top-down* para uma análise do problema e modelagem da arquitetura do sistema proposto.

No desenvolvimento do sistema proposto, utilizou-se a metodologia *bottom-up*, que consiste no desenvolvimento, teste e validação da arquitetura final em seus módulos menores que compõem o sistema.

As fases de desenvolvimento para o Trabalho de Conclusão de Curso são as seguintes:

- Na primeira fase realizou-se o estudo do estado da arte do problema apresentado, estimação da qualidade da água, tipos de ASV's (Veiculo Autônomo de Superfície) existentes, aplicações e tecnologias associadas aos ASV's, sistemas inerciais e orientação, filtragem para amostras de sinais, plataforma e ambiente de desenvolvimento Android, microcontrolador para comunicação com Android e Web Services;

- Na segunda fase foi realizado a configuração da IDE *Eclipse* para o desenvolvimento do *webservice*. Em seguida a configuração da IDE *Android Studio* para desenvolvimento em plataforma Android e por fim a configuração do *Microsoft Visual Studio 2013* para a implementação da interface gráfica do usuário;

- Na terceira fase foi realizada a implementação do *webservice* em paralelo com a implementação do aplicativo Android e com o programa controlador interfaciado, afim de estabelecer a comunicação para o fluxo de dados do sistema proposto. Nesta fase, durante a implementação tanto do aplicativo como o programa controlador focou-se apenas na comunicação;

- Na quarta fase a partir da comunicação estabelecida, foi implementado no aplicativo Android a aquisição de dados provenientes dos sensores embarcados no mesmo (acelerômetro, giroscópio, magnetômetro e bússola digital) e no *webservice* foi implementado um método para o armazenamento desses dados recebidos em um arquivo texto. Em seguida, foi também implementado um filtro media móvel usando o *Matlab* para filtragem desses dados provenientes dos sensores embarcados no dispositivo Android afim de eliminar ruído do processo de medição dos sensores embarcados no dispositivo Android. Esse filtro foi testado utilizando os dados guardados no arquivo texto gerado pelo *webservice*. E por fim apos sua validação, esse mesmo filtro foi implementado no aplicativo Android;

- Na quinta fase a implementação foi realizada no microcontrolador, Arduino ADK, onde foi implementado a aquisição de dados provenientes do sensor de temperatura e a comunicação USB entre o dispositivo Android e o microcontrolador. Em seguida foi implementado o controle dos motores através do driver dos motores (Ponte H);

- Na sexta fase foi construído o protótipo da plataforma móvel proposta. Para tal foi utilizado uma estrutura e motores de um modelo comercial de barco elétrico de controle remoto onde será instalado o sensor de temperatura, o driver de motor, microcontrolador e dispositivo Android;

- Na sétima fase foi realizado testes com o sistema desenvolvimento, com o intuito de validar a solução proposta;

1.6 Organização do Trabalho

Após o capitulo introdutório onde o projeto e seu enquadramento são descritos. No capitulo 2 são apresentados os levantamentos bibliográficos realizados. Inicialmente é apresentado um estudo realizado sobre a estimação da qualidade da água. Em seguida são apresentados os índices e parâmetros relevantes para a medição da qualidade da água. Depois é apresentado um levantamento realizado sobre ASV's já desenvolvidos. Em seguida é realizado um estudo sobre o filtro media móvel. Depois um estudo realizado sobre as plataformas e periféricos que serão utilizadas no desenvolvimento do projeto como o Android, plataforma Arduino-ADK, Web Services, ponte H e o sensor de temperatura.

No capitulo 3 é apresentado o sistema proposto para obtenção de parâmetros de qualidade da água e como esse sistema será implementado no que diz respeito a materiais utilizados, plataformas de desenvolvimento, ferramentas necessárias para o desenvolvimento do projeto. Em um primeiro momento é apresentado a solução proposta e a arquitetura do sistema proposto e em seguida as implementações.

No capitulo 4 são apresentados os resultados parciais com testes realizados ao longo do desenvolvimento do projeto e os resultados obtidos. E no capitulo 5 conclui-se e discuti-se o trabalho realizado.

2 Fundamentação Teórica

Neste capitulo de levantamento do estudo da arte e revisão da literatura estão apresentados os resultados desta fase do projeto que constituiu em reunir informações sobre aspectos relevantes ao desenvolvimento do projeto. Foram estudados parâmetros para medição da qualidade de água, alguns tipos de ASV's (Veiculo Autônomo de Superfície), aplicações e tecnologias associadas aos ASV's, sistemas inerciais e orientação, filtragem de sinais utilizando filtro media móvel, plataforma de desenvolvimento *Open Source* Android, microcontrolador para comunicação com Android, Plataforma .NET, *WebServices*, sensor de temperatura DS18B20 e ponte H.

2.1 Estimação da qualidade da água

Segundo a ANA (ANA, 2014), as informações sobre diversos parâmetros físicos-químicos da água podem ser sintetizados utilizando índices de qualidade da água. Tais índices visam informar a população e orientar as ações de planejamento e gestão da qualidade da água, pois permitem resumir várias informações em apenas um número. De acordo com a ANA, os principais índices utilizados no Brasil são: Índice de Qualidade da Água (IQA), Índice de Qualidade da Água Bruta para fins de Abastecimento Público (IAP), Índice de Estado Trófico (IET), Índice de Contaminação por Tóxicos, Índice de Balneabilidade (IB) e Índice de Qualidade da Água para a Proteção da Vida Aquática (IVA). A seguir são apresentados os indices IQA e IAP que são mais relevantes para estimar a qualidade da água (ANA, 2014).

2.1.1 Índice de Qualidade da Água (IQA)

O Índice da Qualidade da Água (IQA) visa avaliar a qualidade da água bruta utilizada no abastecimento público. Ela foi desenvolvida em 1970, pela National Sanitation Fundation, nos Estados Unidos, e hoje é o principal índice de qualidade da água utilizado no Brasil. O IQA utiliza nove parâmetros: oxigênio dissolvido, coliformes termotolerantes, potencial hidrogeniônico (pH), demanda bioquímica de oxigênio (DBO5,20), temperatura, nitrogênio total, fósforo total, turbidez e resíduo total. Tais parâmetros são em sua maioria indicadores de contaminação causada pelo lançamento de esgotos domésticos (ANA, 2014).

O IQA é um número de 0 a 100, calculado pelo produtório ponderado dos noves parâmetros, de acordo com a formula (2.1).

$$IQA = \prod_{i=1}^{n} {q_i}^{w_i} \tag{2.1}$$

Onde:

q_i é a qualidade do *i-ésimo* parâmetro, isto é, um número entre 0 e 100, obtido pelo respectivo gráfico de qualidade, em função de sua concentração ou medida;

w_i é o peso correspondente ao *i-ésimo* parâmetro;

n número total de parâmetros.

De acordo com a ANA, o peso w_i de cada parâmetro foi definido em função da sua importância para a conformação global da qualidade da água. Estes parâmetros estão dispostos na tabela (1).

Tabela 1: Pesos dos parâmetros de qualidade da água (ANA, 2014)

Parâmetro de qualidade da água	**Peso**
Oxigênio dissolvido	0.17
Coliformes termotolerantes	0.15
Potencial hidrogeniônico (pH)	0.12
Demanda bioquímica de Oxigênio (DBO5,20)	0.10
Temperatura	0.10
Nitrogênio total	0.10
Fósforo total	0.10
Turbidez	0.08
Resíduo total	0.08

Segundo a ANA, cada parâmetro possui um valor de qualidade, que é extraído de seu gráfico de qualidade em função de sua medida ou concentração como mostra a figura (1).

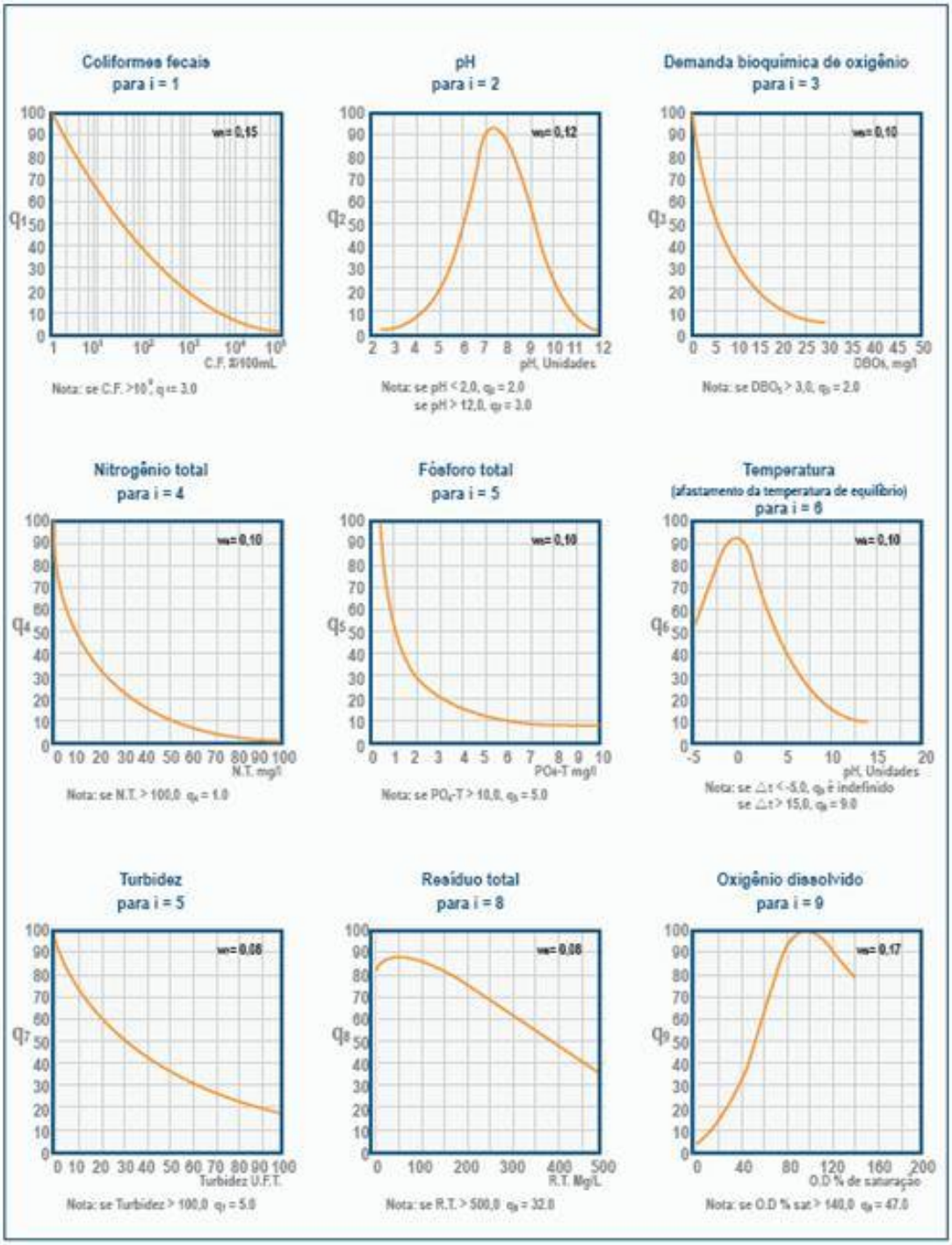

Figura 1: Curvas medias dos valores de qualidade de cada parâmetro (ANA, 2014).

De acordo com a ANA, as faixas de valores do IQA são classificadas de acordo com a tabela (2).

Tabela 2: Classificação das faixas de valores do IQA seperados por estados (ANA, 2014).

AL, MG, MT, PR, RJ, RN, RS	BA, CE, ES, GO, MS, PB, PE, SP	**Classificação**
91 – 100	80 – 100	Ótima
71 – 90	52 – 79	Boa
51 – 70	37 – 51	Razoável
26 – 50	20 – 36	Ruim
0 – 25	0 – 19	Péssima

2.1.2 Descrição dos Parâmetros do IQA

- Oxigênio Dissolvido

As águas poluídas por esgotos apresentam baixa concentração de oxigênio. Enquanto que águas limpas apresentam concentrações de oxigênio dissolvido mais elevadas, geralmente superiores a 5mg/L, exceto se houverem condições naturais que causem baixos valores deste parâmetro.

As águas eutrofizadas (ricas em nutrientes) podem apresentar concentrações de oxigênio superiores a 10 mg/L, situação conhecida como supersaturação.

- Coliformes termotolerantes

As bactérias coliformes termotolerantes ocorrem no trato intestinal de animais de sangue quente e são indicadoras de poluição por esgotos domésticos. Sua presença em grandes números indicam a possibilidade da existência de microrganismos patogênicos que são responsáveis pela transmissão de doenças de veiculação hídrica.

- Potencial Hidrogeniônico

A Resolução CONAMA 357 estabelece que para a proteção da vida aquática o pH deve estar entre 6 e 9. Alterações nos valores de pH também podem aumentar o efeito de substâncias químicas que são tóxicas para os organismos aquáticos, tais como os metais pesados.

- Demanda Bioquímica de Oxigênio (DBO5,20)

A Demanda Bioquímica de Oxigênio representa a quantidade de oxigênio necessária para oxidar a matéria orgânica presente na água através da decomposição microbiana aeróbia. A DBO5,20 é a quantidade de oxigênio consumido durante 5 dias em uma temperatura de 20 °C. A ocorrência de altos valores deste parâmetro causa uma diminuição

dos valores de oxigênio dissolvido na água, o que pode provocar mortandades de peixes e eliminação de outros organismos aquáticos.

- Temperatura

A temperatura influência parâmetros físico-químicos da água como a tensão superficial e a viscosidade. Além disso, os organismos aquáticos são afetados por temperaturas fora de seus limites de tolerância térmica, causando impactos sobre seu crescimento e processo de reprodução.

- Nitrogenio Total

O nitrogênio pode ocorrer nas formas de nitrogênio orgânico, amoniacal, nitrito e nitrato, nos corpos d'água. Os nitratos são tóxicos aos seres humanos, e em altas concentrações causa uma doença chamada metahemoglobinemia infantil, que é letal para crianças.

As fontes de nitrogênio para os corpos d'água são diversos, sendo o lançamento de esgotos sanitários e efluentes industriais uma das principais fontes. A drenagem pluvial, em áreas agrícolas, da água das chuvas em solos que receberam fertilizantes é uma fonte de nitrogênio, tal como a drenagem de águas pluviais em áreas urbanas.

- Fósforo Total

O fósforo é um importante nutriente para os processos biológicos e seu excesso pode causar a eutrofização das águas.

Os esgotos domésticos, pela presença dos detergentes super fosfatados e da própria matéria fecal, destacam-se entre as fontes de fósforo. Além disso, os efluentes industriais das indústrias de fertilizantes, alimentícias, laticínios, frigoríficos e abatedouros são fontes significativas de fósforo.

- Turbidez

A turbidez indica o grau de atenuação que um feixe de luz sofre ao atravessar a água, devido à absorção e espalhamento da luz causada pelos sólidos em suspensão na água. A erosão dos solos e o escoamento pluvial estão entre as principais fontes de turbidez.

A alta turbidez afeta a preservação dos organismos aquáticos, o uso industrial da água e as atividades de lazer. Além disso, o aumento da turbidez faz com que uma quantidade maior de produtos químicos sejam usados nas estações de tratamento de águas, aumentando os custos de tratamento.

- Resíduo Total

O resíduo total é a matéria que permanece após a evaporação, secagem ou calcinação da amostra de água dado um determinado período, sob certa temperatura.

Resíduos sólidos depositados nas águas podem causar danos à visa aquática, uma vez que destroem os organismos que vivem nos sedimentos e servem de alimento para outros organismos, além de danificar os locais de desova de peixes. Além disso, podem causar o assoreamento das águas, que gera problemas para a navegação e aumenta o risco de enchentes.

2.1.3 Índice de Qualidade da Água Bruta para fins de Abastecimento Público

O Índice de Qualidade da Água Bruta para fins de Abastecimento Público (IAP) foi desenvolvido por um grupo técnico formado por integrantes da CETESB, SABESP, institutos de pesquisas e universidades, e é composto, basicamente, por dois índices: o Índice de Qualidade da Água (IQA) e o Índice de Substâncias Tóxicas e Organolépticas (ISTO).

O ISTO é um índice composto por parâmetros que avaliam a presença de substâncias tóxicas (teste de mutagenicidade, potencial de formação de trihalometanos, cádmio, chumbo, cromo total, mercúrio e níquel) e por parâmetros que avaliam a qualidade organoléptica da água (fenóis, ferro, manganês, alumínio, cobre e zinco). De acordo com a ANA, são estabelecidas curvas de qualidade, representadas através das variáveis potencial de formação de trihalometanos e metais, que atribuem ponderações que variam de 0 a 1, para cada parâmetro pertencente a ISTO. Tais curvas foram construídas usando dois níveis de qualidade, associam o valores numéricos 1 e 0.5, respectivamente ao limite inferior e ao limite superior. Uma vez medidos os valores para o potencial de formação de trihalometanos, se o valor for menor que o limite inferior, as águas atendem aos padrões de potabilidade da Portaria 518/04 do Ministério da Saúde e são adequadas para o consumo humano. Caso o valor seja maior que o limite inferior e menor limite superior, as águas atendem aos padrões de qualidade da classe 3 da Resolução CONAMA 357/05 e são adequadas para o tratamento convencional. Caso o valor seja maior que o limite superior, as águas não atendem aos padrões de qualidade da classe 3 da Resolução CONAMA 357/05 e não devem ser submetidas apenas ao tratamento convencional (ANA, 2014)

O IAP é um número entre 0 e 100, calculado através da equação (2.2).

$$IAP = IQA \times ISTO \tag{2.2}$$

De acordo com a ANA, as faixas de valores do IAP são classificadas de acordo com a tabela (3)

Tabela 3: Classificação das faixas de valores do IQA (ANA, 2014).

Valor do IAP	**Qualificação**
80 – 100	Ótima
52 – 79	Boa
37 – 51	Regular
20 – 36	Ruim
< 19	Péssima

2.2 ASV's - Veículos Autônomos de Superfície

Veículos Autônomos de Superfície (ASV) são veículos que operam na superfície de água sem tripulação. A utilização deste tipo de veiculo possibilita a execução de tarefas completamente autônoma e independente. Dentre suas aplicações pode-se listar operações de segurança, monitoramento, exploração de oceanos, aquisição de dados marinhos, entre outros.

2.2.1 Levantamento de ASV's existentes

Na tabela (4) e na figura (2) são apresentados alguns ASV's estudados durante o levantamento do estado da arte. Com base neste levantamento foi realizada uma analise sobre os ASV's que recaiu sobre as aplicações, estrutura mecânica, propulsão, navegação, sistema embarcados e baterias.

Tabela 4: ASV's: modelos e características (ALMEIDA, 2012).

	Modelo/Fabricante	**Aplicação**
1	UMV-H / Yamaha	Busca e Resgate
2	Spartan / US Navy	Militar
3	Ocean Explorer / UOS (Verginia - USA)	Oceanografia e Vigilância
4	ROAZ II / LSA - ISEP (INESC)	Oceanografia e Vigilância
5	Springer / University of Plymouth (UK)	Recolha de dados ambientais

Figura 2: ASV's da tabela (4), respectivamente (ALMEIDA, 2012).

2.2.2 Estrutura mecânica

Segundo (ALMEIDA, 2012) alguns ASV's analisados nesta etapa foram implementados modificando ou adaptando embarcações já existentes, tais adaptações foram

realizadas no intuito de equipar os ASVs conforme a sua aplicação. No entanto a maioria dos ASV's estudos foram criados e não adaptados, criando assim uma necessidade de definir o tipo de estrutura mecânica ou casco mais adequado para determinada aplicação levando em consideração ambiente externo, funcionalidades e tecnologias envolvidas. A partir disso é possível destacar duas estruturas básicas de casco: casco de deslocamento e casco planantes.

Cascos de deslocamento são comuns em embarcações que não têm que navegar a velocidades elevadas, são sustentados pelo peso da água que fazem deslocar. Já os cascos planantes geralmente são mais estáveis em relação a embarcações de casco de deslocamento e possuem uma navegação mais suave em ambientes mais agitados.

2.2.3 Propulsão

Em relação à propulsão, nota-se que os tipos mais comuns são os propulsores com hélices e jato híbrido (Water-jet). Geralmente os propulsores com hélices quando comparados aos propulsores a jato hibrido, são usados quando não deseja-se velocidades elevadas, são mais eficientes e mais econômicos no entanto mais lentos e permitem menor manobrabilidade e também tem uma menor densidade de potência (relação potência/volume). É importante ressaltar que estes propulsores estão associados a motores que podem ser de combustão ou elétricos (ALMEIDA, 2012).

2.2.4 Sistemas Embarcados em ASV's

Um sistema embarcado é um sistema microprocessado no qual o computador é completamente encapsulado ou dedicado ao dispositivo ou sistema que ele controla, por definição um sistema embarcado recebe entradas de sensores as processa e gera uma saída. Adicionalmente, um sistema embarcado está submetido a restrições de portabilidade (tamanho e peso), alto desempenho e baixo consumo de potência (ALMEIDA, 2012).

Um sistema embarcado realiza um conjunto de tarefas predefinidas, geralmente com requisitos específicos. Já que o sistema é dedicado a tarefas específicas, através de engenharia pode-se otimizar o projeto reduzindo tamanho, recursos computacionais e custo do produto.

O sistema embarcado é um dos pontos mais cruciais no desenvolvimento de um ASV. Neste sistema onde esta localizado todo o sistema de controle e navegação do ASV, aquisição de dados dependendo da aplicação e todo os restantes sistemas tecnológicos estão implementados.

2.2.5 Navegação

A navegação de um ASV é normalmente efetuada complementando sistemas de posicionamento como sistemas GPS, sistemas inerciais e sistemas para desvio de obstáculos. (ALMEIDA, 2012)

2.2.5.1 Sistemas de Posicionamento

Em meio aquático os sistemas de posicionamento são normalmente baseados em GPS. A precisão dessa localização depende do tipo de sistema GPS implementado. Com o GPS simples, economicamente mais acessíveis, é possível obter precisão na ordem dos 3 a 4 metros. Utilizando um sistema de GPS diferenciais, com custo maior, é possível atingir níveis de precisão mais elevados, normalmente na ordem de centímetros (BARONI, 2004).

2.2.5.2 Sistemas Inerciais

Os sistemas inerciais também conhecidos como IMU (Inertial Measurement Unit), têm como finalidade estimar a posição, orientação, velocidade e aceleração do veículo através de sensores como giroscópios, acelerômetros, bússolas e inclinômetros. Obtendo os dados sobre a velocidade e localizações anteriores é possível integrar e estimar a localização atual (NOGUEIRA, 2014).

2.2.5.3 Sistema de desvio de obstáculos

O sistema de desvio de obstáculos são baseados em sensores que permitem a detecção de um objeto e auxilie no sistema de controle do ASV, tais como câmeras, lasers, radares, etc. A partir desses sensores complementados com técnicas de desvio de obstáculos, obtêm-se uma ampla variedade de algoritmos de navegação e controle, podendo apenas realizar o contorno do obstaculo ou podem realizar uma analise mais complexa da situação e identificar formas mais eficientes de efetuar o desvio.

2.2.6 Bateria

Normalmente as baterias utilizadas para este tipo de aplicação são baterias de íons de lítio ou baterias de polímeros de lítio. A tabela (5) mostra uma comparação entre esses tipos de baterias e outras baterias comumente usadas tais como baterias de ácido chumbo e baterias de níquel.

Tabela 5: Comparação de diferentes tipos de baterias (ALMEIDA, 2012).

Tipo de Bateria	**Potência específica** (W/Kg)	**Eficiência** (%)	**Ciclos de vida**
Ácido chumbo	120 - 180	70 - 92	500 - 800
Ni-MH	150 - 400	66	1000
Íons de lítio	1800 - 3000	94	1200
Polímeros de lítio	3000+	97	500 - 1000

Observa-se que que as baterias de íon de lítio e polímero de lítio possuem uma potência específica e eficiência maior sendo, portanto, apropriados para a aplicação em ASV's.

2.3 Filtragem

Os sistemas práticos estão sujeitos a ruídos e perturbações aleatórias que podem dificultar os procedimentos da análise e da identificação. Para contornar estes problemas tem-se recorrido a utilização de filtros, a fim de se capturar somente os sinais com as dinâmicas de interesse presentes no sistema (NOGUEIRA, 2014).

Filtrar um sinal é deixar passar pelo sistema a informação de interesse e bloquear a informação indesejada. Filtros podem ser aplicados não somente para diminuir a influência do ruído do processo, mas também para "suavizar" o resultado de não homogeneidade de misturas, turbulências ou fluxos não uniformes.

Neste trabalho foram analisados dois tipos de filtros lineares, o filtro Kalman e o filtro de média móvel. Simulações numéricas foram realizadas com cada filtro e após uma avaliação expertimental na plataforma Android foi escolhido implementar o filtro de média móvel dado que apresentou menor custo computacional. A seguir é apresentado o filtro que será utilizados na navegação e no controle da plataforma robótica.

2.3.1 Media Móvel

O filtro de média móvel é obtido calculando-se a média de um conjunto de valores, sempre se adicionando um novo valor ao conjunto e se descartando o mais velho. Não é apenas uma média de um conjunto isolado de valores (NOGUEIRA, 2014). O filtro de média móvel é representado pela equação (2.3).

$$y[k] = \frac{1}{N} \sum_{j=0}^{N-1} x[k-j] \tag{2.3}$$

Onde:

k é o tempo atual ou índice dos vetores utilizados;

N é o numero de amostras;

$x[k-j]$ representa o conjunto dos valores a serem somados;

$y[i]$ é o sinal filtrado;

2.4 Plataforma de Desenvolvimento Android

O Android é uma plataforma de desenvolvimento para aplicativos móveis como o *smarthphone* que conta com uma interface rica e repleta de diversas aplicações já instaladas e ainda um ambiente de desenvolvimento bastante poderoso, inovador e flexível (LECHETA, 2013)

Sua arquitetura é bastante flexível e pode integrar aplicações nativas com aplicações desenvolvidas. Um aspecto relevante do Android é que seu sistema operacional foi baseado no kernel 2.6 do Linux, que é responsável por gerenciar a memória, os processos, threads e a segurança dos arquivos e pastas, alem de redes e drivers (LECHETA, 2013).

A escolha da plataforma Android justifica-se visto que o Android é a primeira plataforma para aplicações móveis completamente livre e de código aberto (*Open Source*), o que representa uma grande vantagem para sua evolução, uma vez que diversos programadores do mundo podem podem contribuir para melhorar a plataforma. Além disso o sistema Android já possui um protocolo de comunicação (*Android Open Accessory*) que permite a interação entre hardware USB e dispositivos Android, o que é um dos requisitos da arquitetura do sistema proposto.

2.4.1 Arquitetura

A arquitetura da plataforma Android é dividida em várias camadas: Applications, Application Framework, Libraries e Android Runtime e por fim Linux Kernel (ANDROID, 2015).

Na camada Applications, está localizada uma lista de aplicações padrões que incluem um cliente de e-mail, programa de SMS, calendário, mapas, navegador, gerenciador de contatos, e outros que serão desenvolvidos pela comunidade, sendo todas essas aplicações escritas na linguagem Java.

Já na camada Application Framework estão os componentes que permitirão com que novas estruturas sejam utilizadas para futuras aplicações, enfatizando a reutilização de código. Os seguintes componentes fazem parte desta camada:

- Um rico e extensível conjunto de componentes gráficos que pode ser utilizado para construir uma aplicação, bem como listas, grids, caixas de textos, botões, e até um navegador web embutido;

- Provedores de conteúdo que habilitam às aplicações acessar dados de outras aplicações (como os Contatos, por exemplo) ou compartilhar seus próprios dados;
- Gerenciador de recursos que proveem acesso a recursos não-codificados como strings, gráficos, e arquivos de layout;
- Um gerenciador de notificação que permite que todas as aplicações exibam mensagens de alerta personalizáveis na barra de status;
- Um gerenciador de atividade que gerencia o ciclo de vida das aplicações e permite controlar os recursos previamente alocados, sendo que caso eles não estejam sendo mais utilizados, os mesmos são desalocados para liberar memória;

A camada logo abaixo é subdivida no grupo das bibliotecas (libraries) e o ambiente de execução (runtime) da plataforma Android, composto pelas bibliotecas padrão e pela máquina virtual denominada Dalvik. No primeiro grupo estão as bibliotecas escritas em C/C++, que são compostas por uma coleção de bibliotecas que são utilizadas pela plataforma Android (ANDROID, 2015). Estas bibliotecas são:

- Biblioteca de sistema C: é uma implementação da biblioteca C padrão (libc), otimizada para dispositivos que suportam a plataforma Linux (embbeded- linux);
- Bibliotecas de Mídias: as bibliotecas suportam execução e gravação da maioria dos formatos de áudio e vídeo, bem como exibição de imagens, incluindo MPEG4, H.264, MP3, AAC, AMR, JPG, e PNG;
- Gerenciador de Superfície: gerencia o acesso ao display do dispositivo e camadas de gráficos 2D e 3D de múltiplas aplicações;
- LibWebCore: uma moderna engine de navegador web que turbina tanto o navegador da plataforma Android e um outro navegador qualquer desenvolvido;
- SGL: uma engine de gráficos 2D;
- 3D libraries: uma implementação baseada na especificação OpenGL ES 1.0, a qual utiliza tanto aceleração de hardware 3D e um avançado e otimizado software para renderização de modelos tridimensionais;
- FreeType: renderização em formatos bitmaps e vetoriais de fontes;
- SQLite: uma ponderosa e leve engine de banco de dados relacional disponível para todas as aplicações;

Na base, está localizado o kernel Linux, que para a Android será a versão 2.6, fornecendo serviços do núcleo do sistema como segurança, gerenciamento de memória, gerenciamento de processos, pilhas de redes, etc. A Figura (3) mostra os principais componentes da arquitetura do Android e a Tabela (6) apresenta as principais API's (Application Programming Interface) (ANDROID, 2015).

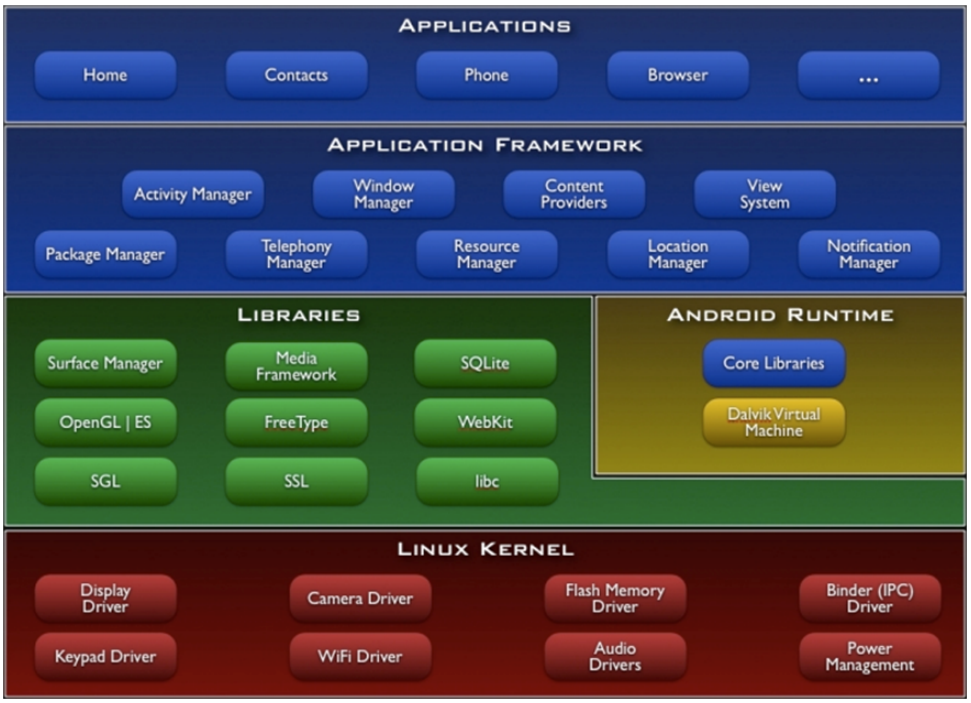

Figura 3: Camadas da arquitetura do Android (ANDROID, 2015).

Tabela 6: Principais API's do android (ANDROID, 2015).

Pacote	**Descrição**
android.util	Contém varias classes utilitárias
android.os	Contém serviços referentes ao sistema operacional
android.graphics	Pacote principal dos recursos gráficos
android.text	Suporte para ferramentas de processamento de texto
android.database	API's para comunicação com banco de dados SQLite
android.content	API's de acesso a dados no dispositivos
android.view	O pacote com os principais componentes da interface gráfica
android.widget	Contem widgets prontos (gerenciadores de layout)
android.app	API's de alto nivel referentes ao modelo de aplicação
android.provider	API's para padrões de provedores de conteúdo
android.telephony	API's para interagir com funcionalidades de telefonia
android.webkit	API's para conteudos de context box

2.4.2 Activity

As peças que compõem (não necessariamente todas ao mesmo tempo) uma aplicação em Android são: Activity, IntentReceiver, Service e ContentProvider. Uma Activity (ou Atividade) representa uma simples tela em branco e pode ser criada estendendo a classe Activity. As aplicações reais, como por exemplo, uma lista de contatos, um organizador pessoal, um localizador geográfico, etc, serão compostas de várias Actividades. Ou seja, cada tela seria um Activity (BOEHMER, 2012).

Para melhorar o entendimento de como funciona uma Activity no Android a figura (4) mostra o ciclo de vida de uma Activity no Android.

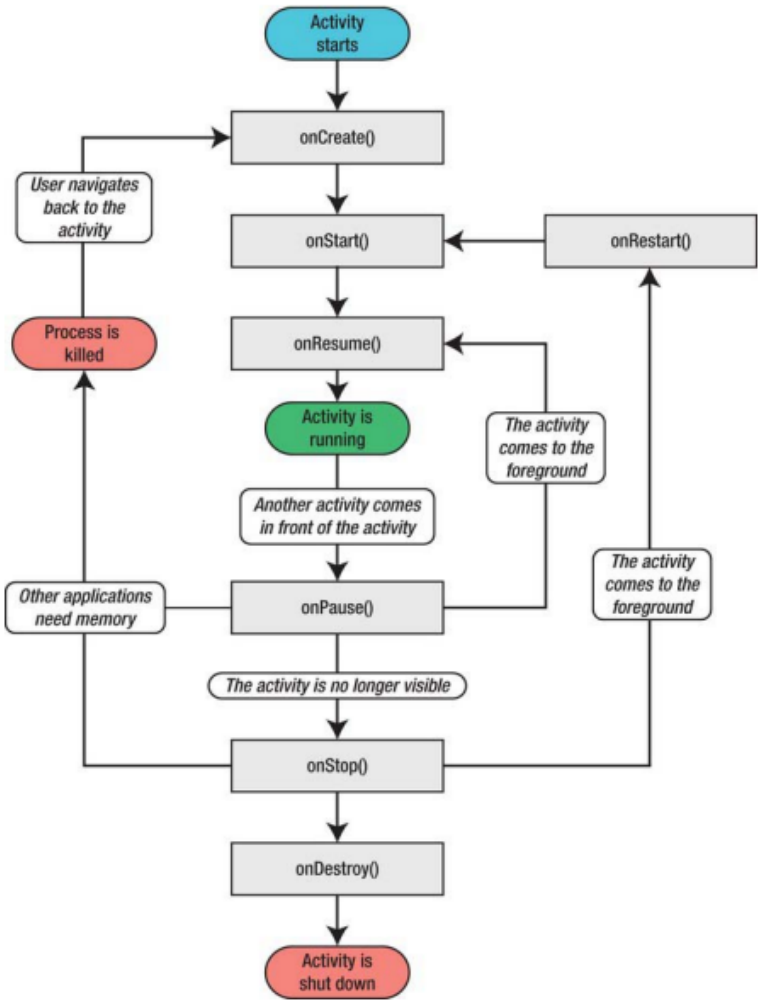

Figura 4: Ciclo de vida de uma Activity no Android (BOEHMER, 2012).

2.4.3 Interface Gráfica

Os principais componentes que fazem parte da interface gráfica com o usuário (GUI) da plataforma Android foram denominados de Views e Viewgroups (ANDROID, 2015).

Um View é representado pela classe android.view.View. Este componente nada mais é do que uma estrutura de dados que representa uma área retangular limitada da tela do dispositivo. A plataforma vem equipada com vários widgets já implementados prontos para o desenvolvedor utilizar em suas aplicações (ANDROID, 2015).

O outro componente é representado pela classe android.view.Viewgroup. Um Viewgroup pode ser considerado um layout, isto é, um container de Views (e também de Viewgroups) que pode ser utilizado para criar estruturas mais complexas, ricas e robustas. Um Viewgroup é a classe base para outros layouts (LinearLayout, RelativeLayout, AbsoluteLayout, etc). No momento em que o método setContentView() é chamado, o mesmo recebe uma referência para o nó raiz da árvore que representa o layout a ser mostrado. Sendo assim, o mesmo informa a seus filhos para que eles possam se desenhar, validar, etc. A Figura (5) mostra a árvore que representa a hierarquia formada por Views e Viewgroups (ANDROID, 2015).

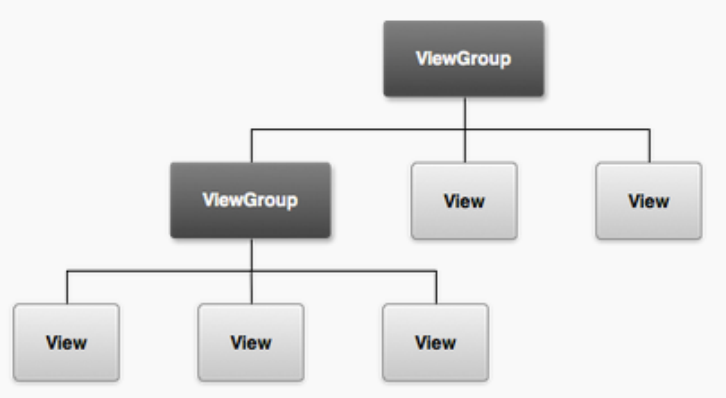

Figura 5: Hierarquia de Views e Viewsgroups no Android (ANDROID, 2015).

2.4.4 Sensores Embarcados em dispositivos Android

A maioria dos dispositivos com Android têm sensores embarcados que medem o movimento, a orientação e diversas condições ambientais. Estes sensores são capazes de fornecer dados brutos com alta precisão e exatidão, e são úteis para monitorar movimento tridimensional ou posicionamento tridimensional do dispositivo, ou monitorar mudanças no ambiente ambiente perto de um dispositivo. Por exemplo, um aplicativo de viagens pode utilizar o sensor de campo geomagnético e acelerômetro para implementar uma bússola (ANDROID, 2015).

A plataforma Android suporta três grandes categorias de sensores:

- **Sensores de movimento:** Estes sensores medem forças de aceleração e rotação ao longo de três eixos. Esta categoria inclui acelerômetros, sensores de gravidade, giroscópios e sensores de vetor de rotação.

- **Sensores ambientais:** Estes sensores medem vários parâmetros ambientais, tais como temperatura ambiente, pressão do ar, iluminação, umidade, etc. Esta categoria inclui barómetros, fotômetros e termômetros.
- **Sensores de Posição:** Estes sensores medem a posição física de um dispositivo. Esta categoria inclui sensores de orientação e magnetômetros.

A estrutura de sensores no Android permite o acesso a muitos tipos de sensores. Alguns desses sensores são baseados em hardware e alguns são baseados em software. Sensores baseados em hardware são componentes físicos embarcados em um dispositivo com Android. Tais sensores retiram os seus dados diretamente de propriedades ambientais específicas, tais como aceleração, força do campo geomagnético, ou rotação. Sensores baseados em software não são dispositivos físicos, embora eles imitam sensores baseados em hardware. Os sensores baseados em software são derivados de um ou mais sensores com base em hardware e são chamados sensores virtuais ou sensores sintéticos. O sensor de aceleração linear e o sensor de gravidade são exemplos de sensores com base em software. A tabela (7) resume os sensores que são suportados pela plataforma Android.

Tabela 7: Tipos de sensores suportados pela plataforma Android (ANDROID, 2015).

Sensor	Tipo	Descrição
TYPE_ACCLEROMETER	Hardware	Mede aceleração em m/s^2 aplicada nos três eixos (x,y,z)
TYPE_AMBIENT_TEMPERATURE	hardware	Mede temperatura ambiente em graus Celcius
TYPE_GRAVITY	Software ou Hardware	Mede gravidade em m/s^2 aplicada nos três eixos (x,y,z)
TYPE_GYROSCOPE	Hardware	Mede taxa de rotação em rad/s nos três eixos (x,y,z)
TYPE_LIGHT	Hardware	Mede nível de luz ambiente em lx
TYPE_LINEAR_ACCELERATION	Software ou Hardware	Mede a aceleração em m/s^2 retirando a gravidade
TYPE_MAGNETIC_FIELD	Hardware	Mede o campo geomagnético em mT nos três eixos (x,y,z)
TYPE_ORIENTATION	Software	Monitora a posição em relação ao referencial da Terra (Norte magnético)
TYPE_PRESSURE	Hardware	Mede a pressão do ar ambiente em hPa ou $mbar$
TYPE_PROXIMITY	Hardware	Mede a proximidade de um objeto em cm em relação ao dispositivo
TYPE_RELATIVE_HUMIDITY	Hardware	Mede a humidade ambiente relativa em porcetagem (%)
TYPE_ROTATION_VECTOR	Software ou Hardware	Mede a orientação atraves dos três elementos do vetor de rotação
TYPE_TEMPERATURE	Hardware	Mede a temperatura do dispositivo em graus Celcius.

O Android possui um *framework* para acessar esses sensores. Esse *framework* faz parte do pacote *android.hardware* (ANDROID, 2015) e inclui as seguintes classes e interface:

- **SensorManager:** Classe para criar uma instância do serviço de sensor. Essa classe fornece vários métodos para acessar e listar sensores, registrar e cancelar o registro dos *listeners* de evento do sensor, e aquisição de informações de sensores. Essa classe também fornece várias constantes de sensores que são usados para relatar a precisão do sensor, estabelecer taxas de aquisição de dados, e calibrar sensores.
- **Sensor:** Classe para criar uma instância de um sensor específico. Essa classe fornece vários métodos que permitem determinar as capacidades de um sensor.
- **SensorEvent:** O sistema usa essa classe para criar um objeto de evento do sensor, que fornece informações sobre um evento de sensor. Um objeto de evento do sensor

inclui as seguintes informações: os dados brutos do sensor, o tipo de sensor que gerou o evento, a precisão dos dados, e a data e hora do evento.

- **SensorEventListener:** Interface para criar dois métodos de retorno de chamada que recebem notificações (eventos do sensor) quando o valor do sensor muda ou quando a precisão do sensor muda.

Os sensores embarcados dentro de um dispositivo Android podem variar de dispositivo para dispositivo além de também poder variar entre versões do Android. Isso deve-se ao fato de que esses sensores foram introduzidos ao longo de vários lançamentos da plataforma e com a evolução dos dispositivos.

2.4.4.1 Serviço de Localização

Para utilizar o GPS no Android utiliza-se um *framework* pertencente ao pacote *android.location* diferente do utilizado nos sensores (*android.hardware*).

O Android fornece APIs pertencentes ao pacote *android.location* para trabalhar com o sistema de localização (ANDROID, 2015). O serviço de localização que o Android fornece, para a aquisição da localização do dispositivo, possui dois tipos de provedores:

- **GPS_PROVIDER:** provedor para obter dados do hardware do GPS;
- **NETWORK_PROVIDER:** provedor para obter dados por triangulação de antenas;

O NETWORK_PROVIDER retorna o primeiro resultado de forma mais rápida que o GPS_PROVIDER. Este provedor utiliza a triangulação de antenas da operadora para descobrir a localização do usuário, esse método não possui uma boa precisão e a posição retornada pode ter vários metros de diferença (ANDROID, 2015).

O GPS_PROVIDER possui uma boa precisão, mas pode demorar para retornar o primeiro resultado, que pode variar de 30 segundos a alguns minutos, no entanto esse provedor consome mais bateria que o NETWORK_PROVIDER além disso para se obter melhores resultados é melhor estar em um ambiente aberto visto que lugares cobertos afetam o sinal (ANDROID, 2015).

2.5 Plataforma .NET

A plataforma .NET é um *framework* desenvolvido pela Microsoft que oferece uma gama de ferramentas para o desenvolvimento e execução de sistemas e aplicações, o que inclui uma grande variedade de bibliotecas de classes, o Common Language Runtime

e as linguagens de programação C# e Visual Basic. Uma das principais vantagens da utilização da plataforma .NET é a padronização da mesma, que permite que uma aplicação .NET possa ser executada em qualquer sistema que possua a *framework* instalada (MICROSOFT, 2015a).

2.5.1 Windows Forms Application

Windows Forms é uma biblioteca de classe gráfica da plataforma .NET, que permite a construção de ricas interfaces gráficas para desktops. Basicamente, uma aplicação Windows Form (figura (6)) é um formulário em branco acrescido de controles que permitem o processamento e a manipulação de dados (MICROSOFT, 2015b).

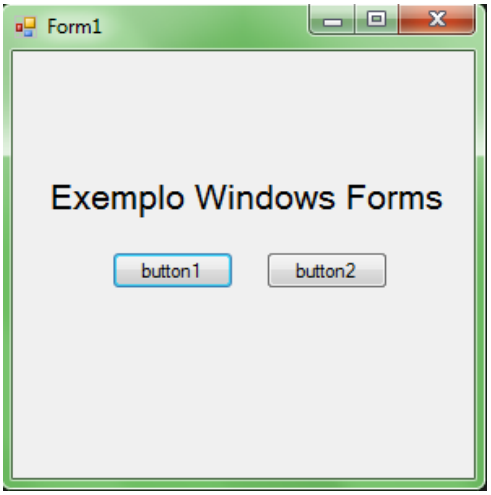

Figura 6: Exemplo de uma aplicacao Windows Form.

Um controle é um objeto que possui seu próprio conjunto de propriedades, métodos e eventos que o tornam mais adequado a determinada finalidade da aplicação. O Windows Forms já possui uma série de controles como: botões, caixas de textos, combos, caixas de checagem entre outros, como ilustrado na figura (7). Além disso, o Windows Forms oferece suporte para criação de controles através da classe *UserControl*.

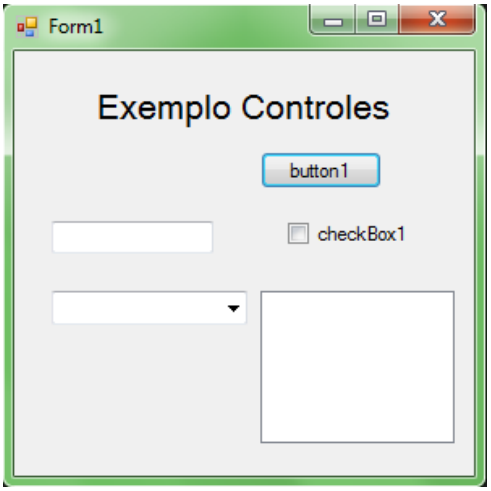

Figura 7: Exemplo de controle no Windows Form.

Uma aplicação Windows Forms é uma aplicação orientada a eventos. Desse modo, ao manipular os controles, eles disparam eventos. A aplicação responde a esses eventos (acionados pelos controles) e os processam assim que eles são disparados. Um exemplo de disparo de evento é o clique de botão. Ao clicar um botão, um evento é disparado e a aplicação executa uma determinada ação ao processar o evento (MICROSOFT, 2015b).

O Windows Forms oferece ainda uma série de classes para manipulação de dados de forma inteligente como o controle *DataGridView* e o componente *BindingSource*. O controle *DataGridView* permite a exibição de dados de forma tabular, permitindo uma poderosa customização nas exibições. O componente *BindingSource* permite a conexão direta entre uma fonte de dados (com uma propriedade) e um controle na aplicação (MICROSOFT, 2015b).

A escolha do Windows Forms no desenvolvimento da interface gráfica do usuário justifica-se visto que a plataforma já oferece uma gama de ferramentas e controlas que simplifica o desenvolvimento do front-end da aplicação. Ao mesmo tempo é possível desenvolver controles e templates personalizados em linguagem C#, o que torna a aplicação flexível e escalável. Além disso, a plataforma .NET já possui suporte para implementação de WebServices, que é um dos requisitos da arquitetura do sistema proposto.

2.6 WebServices

WebServices são aplicações modulares que tem como principal característica a utilização de uma forma padrão e universal de trocar informações, o que permite, por exemplo, que aplicações construidas em diferentes linguagens de programação possam se comunicar. Essas aplicações utilizam XML (Extensible Markup Language) em conjunto com o protocolo HTTP (Hypertext Transfer Protocol) e o padrão SOAP (Simple Object

Access Protocol) para trafegar informações (HADDAD, 2014). Mais especificamente, o acesso é via HTTP, de modo que as informações são enviadas em envelopes SOAP, que são descritos em XML, conforme demonstrado na figura (8).

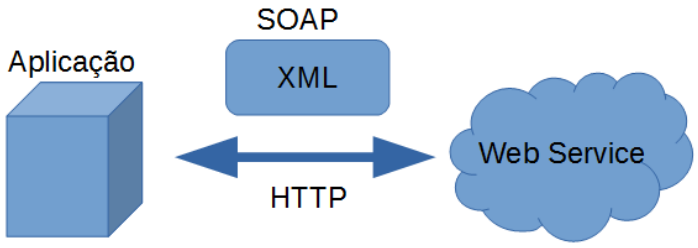

Figura 8: Exemplo de comunicação utilizando um *webservice*.

Para que o tráfego de informação entre um *webservice* e uma aplicação seja viabilizado, é necessário a aplicação saiba o modo de funcionamento do *webservice*, o que é realizado através do WSDL (Web Service Description Language), que é um documento (escrito em XML) que contém a descrição do funcionamento do WebService (MACORATTI, 2014).

A escolha de WebServices justifica-se visto que permitem a comunicação entre aplicações escritas em diferentes linguagens. Por exemplo, um aplicação escrita em linguagem C# pode comunicar-se com uma aplicação escrita em Java, através de WebServices. Isso ocorre porque independente da linguagem o qual a aplicação foi construída, a informação é transmitida em SOPA via HTTP.

2.7 Plataforma Arduino ADK

2.7.1 ADK - Acessory Development Kit

O Accessory Development Kit (ADK) junto com protocolo *Open Accessory* são padrões de implementação criados pela Google Inc. que visam padronizar a comunicação de dispositivos Android com hardware externo. Especificamente, o ADK é uma implementação de referência do protocolo padrão *Open Accessory* de uma placa de desenvolvimento micro-controladora (ANDROID, 2014).

Para que a comunicação entre um dispositivo externo e o Android ocorra alguns requisitos devem ser atendidos. O primeiro requisito é que o dispositivo externo deve possuir um controlador USB adequado. Este controlador permite a funcionalidade geral do USB, mas também permite o chamado modo acessório. O modo acessório permite que um dispositivo Android que não possui recursos de hospedeiro USB para comunicação com dispositivos externos, passe a atuar como parte de um hospedeiro USB. Além disso,

a especificação do padrão *Open Accessory* estipula que o hospedeiro USB deve fornecer energia para o barramento USB e poder enumerar os dispositivos conectados. Atendendo as especificações USB 2.0, o dispositivo externo deve fornecer uma alimentação de $5V$ e $500mA$ para o dispositivo Android (BOEHMER, 2012).

O ADK fornece um *firmware* para uma placa de desenvolvimento, que vem na forma de um conjunto de códigos fonte, bibliotecas e um *sketch* de modelo na sintaxe do arduino. O *firmware* fornecido é utilizado na enumeração do barramento USB e para que a placa de desenvolvimento seja reconhecida pelo dispositivo Android como um dispositivo compatível com o modo acessório.

2.7.2 Arduino ADK

O Arduino ADK (figura (9)) é uma placa de desenvolvimento microcontroladora baseada no *AtMega2560*, da série Arduino compatível com o padrão de referência ADK. A placa Arduino ADK possui um conector de alimentação de corrente contínua e um conector USB (tipo A) utilizado para conectar um dispositivo Android. Além disso, há um conector USB (tipo B) utilizado na programação e depuração da placa ADK (ARDUINO, 2014).

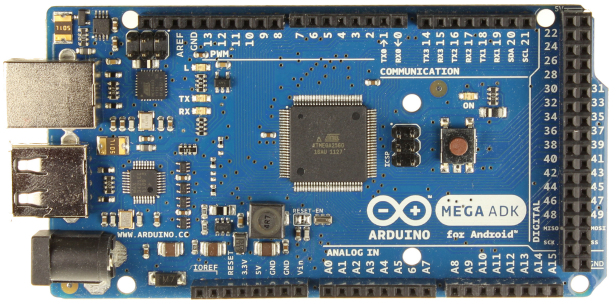

Figura 9: Arduino ADK (ARDUINO, 2014).

No arduino ADK, o circuito integrado *MAX3421e* é utilizado no interfaceamento USB entre a placa ADK e o dispositivo Android. Este modelo de Arduino possui 54 pinos digitais (dos quais 15 pinos podem ser utilizados no modo pwm), 16 pinos de entrada analogica, 4 UARTS e um oscilador de 16 MHz (BOEHMER, 2012).

A comunicação com o Android é realizada com a implementação de duas bibliotecas na programação do Arduino ADK: a biblioteca *USBHostShield* e a biblioteca *AndroidAccessory*.

A biblioteca *USBHostShield* foi, originalmente, criada com objetivo inicial de criar um padrão do comunicação USB da placa Arduino com outros dispositivos, mas que com algumas modificações que a tornou compatível com o padrão ADK, viabilizando a

comunicação entre o Arduino dotado do circuito integrado *MAX3421E* e o dispositivo Android.

A biblioteca *AndroidAccessory* é a responsável por implementar o protocolo Open Accessory. Essa implementação é realizada através da classe *AndroidAccessory*. Essa classe possui seis atributos: manufacturer, model, description, version, URI e serial. Os atributos manufacturer, model e version são os mais importantes pois eles são utilizados na aplicação Android, para verificar se a mesma está se comunicando com a placa ADK correta. Além disso, o atributo URI é utilizado pelo sistema Android para procurar um aplicativo adequado para o acessório, caso não haja nenhum instalado, podendo inclusive buscar um aplicativo na web.

O método *powerOn* da classe *AndroidAccessory* é o método que inicia o protocolo de conexão. A instanciação do objeto *AndroidAccessory* adequadamente e a invocação do método *powerOn* já são o suficiente para tornar o Arduino ADK reconhecível como dispositivo ADK.

A classe *AndroidAccessory* prover métodos de leitura e escrita de mensagens. A Google define uma mensagem como um array de 3 bytes (figura (10)). O primeiro byte define o tipo de comando a ser lido ou enviado. O segundo byte define o alvo do comando. O terceiro byte define o valor do comando.

1-Byte	2-Byte	3-Byte
Command 0xF	Target 0xF	Value 0xF

Figura 10: Protocolo de comunicação entre Arduino ADK e Android (BOEHMER, 2012).

O envio e leitura de strings pode ser implementado adaptando o protocolo da Google (figura (11)). O primeiro byte continua sendo o tipo do comando. O segundo byte continua a ser o alvo. O terceiro byte indica o tamanho da mensagem de texto, enquanto que os bytes sucessivos são a string em si.

1-Byte	2-Byte	3-Byte	N-Bytes
Command 0xF	Target 0xF	Checksum N	Message ...

Figura 11: Protocolo de comunicação adaptado para mensagens de texto (BOEHMER, 2012).

A escolha do Arduino ADK justifica-se visto que essa plataforma implementa o *Android Development Kit*, o que permite que a mesma possa interagir, via conexão USB, com um dispositivo Android.

2.7.2.1 PWM - Modulação por Largura de Pulso

PWM é uma técnica de transmissão de informação através de um transmissor digital. Nessa técnica uma saída digital produz uma onda quadrada, de modo que a informação é transmitida através da largura do pulso que está em nível alto em contraste com o nível baixo da onda. A porcentagem da largura do pulso (nível alto) em relação ao nível baixo do período da onda é chamado *duty cycle*, conforme ilustrado na figura (12) (SPARKFUN, 2015).

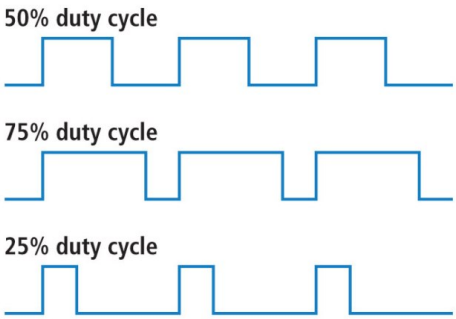

Figura 12: Exemplos de *duty cycle* (SPARKFUN, 2015).

Uma das vantagens do PWM é a possibilidade de simular um sinal analógico, utilizando um dispositivo digital. Tal simulação permite o controle de alguns dispositivos eletrônicos controlados por sinais analógicos (como motores elétricos), por meio de um sistema digital. Desse forma um sistema digital pode processar digitalmente as informações e ainda controlar (por meio do PWM) determinado atuador.

2.7.2.2 UART - Universal Asynchronous Receiver/Transmitter

UART é uma portal serial que permite a comunicação de dispositivos, como computadores, impressoras, microcontroladores. a transmissão em série envolve o envio de um bit por ciclo. O número total de bits transmitidos por segundo é a chamada taxa de transmissão. O recíproco da taxa de transmissão é o tempo de bit, que é tempo gasto para o envio de um único bit. Cada UART possui um registro de controle de velocidade transmissão, que é utilizado para selecionar a taxa de transmissão. Cada dispositivo é capaz de criar seu próprio clock serial, com uma frequência de transmissão de, aproximadamente, igual ao clock serial com o dispositivo o qual está se comunicando. Um frame é a menor unidade completa de uma transmissão serial. Na figura temos ilustrado a transmissão de sinal ao longo do tempo em uma porta serial, mostrando cada frame, o que inclui o bit de início (igual a 0), 8 bits de informação (o bit menos significativo é o primeiro), e um bit de parada (igual a 1). A informação da taxa ou largura de banda é definida como a quantidade de dados ou informação útil transmitida por segundo (VALVANO, 2015).

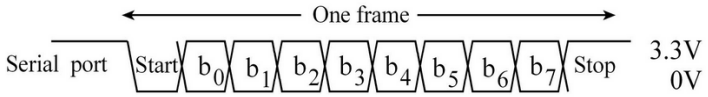

Figura 13: Exemplo de transmissão serial (VALVANO, 2015).

Na figura (13), observa-se que 10 bits são enviados para cada byte. Desse modo, a largura de banda do canal serial (em bytes por segundo) é a taxa de transmissão (em bits por segundo) dividido por 10. Nesse trabalho, utilizou-se um pino do Arduino para implementar o sinal PWM que faz o controle de velocidade da plataforma móvel.

2.8 Ponte H

O driver escolhido para o acionamento dos motores foi um Módulo driver *Ponte H* (figura (14)), que possui o chip L298N como referência. Este driver de motor é capaz de controlar o sentido de giro do motor e, além disso, a velocidade do mesmo. Segundo especificação do fabricante, o driver deve fazer o acionamento de um motor DC com tensão de operação de, no máximo, 30 V (HUINFINITO, 2015a).

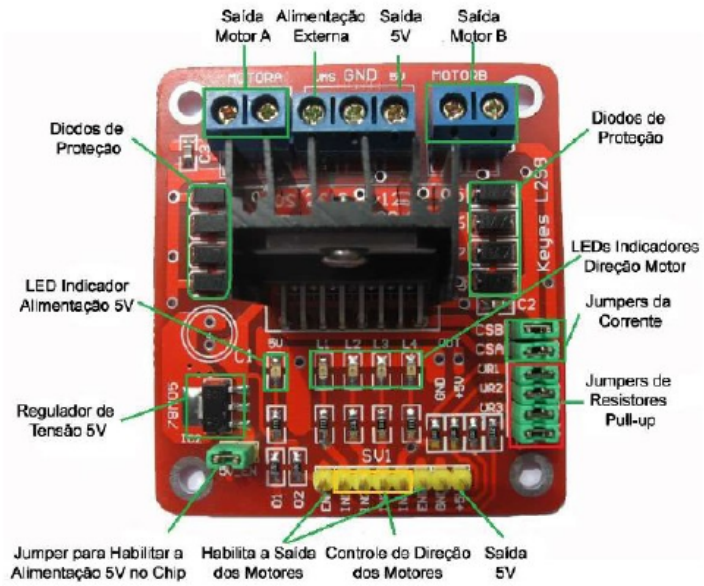

Figura 14: Modulo Driver Ponte H (HUINFINITO, 2015a).

O chip L298N é o circuito integrado no qual este driver foi desenvolvido, e trata-se de um circuito integrado com dupla ponte H. Este é capaz de operar em altas tensões e até mesmo altas correntes, e foi projetado, de modo, a aceitar níveis lógicos TTL e trabalhar também com unidades cargas indutivas como relés, solenoides, motores DC e motores de passo. Duas entradas de habilitação são fornecidas para ativar ou desativar o dispositivo,

independentemente da entrada de sinais. Os emissores dos transístores inferiores de cada ponte são ligados entre si e os correspondentes terminais externos podem ser utilizados para a conexão de uma resistência de detecção externa. Uma entrada adicional da fonte é fornecida de modo que a funcionalidade permanece a mesma em tensões admissíveis a níveis lógicos. A pinagem de driver está descrito na tabela (8) (ST, 2015).

Tabela 8: Descrição dos terminais do driver do motor (ST, 2015).

Terminal	**Tipo**	**Descrição**
VMS e GND	-	Conexão para fonte de alimentação externa (6V a 35V)
ENA	Entrada	Controle de saída do motor: o estado "baixo (0V)" desativa o MOTOR A
IN1	Entrada	Controle de direção do MOTOR A
IN2	Entrada	Controle de direção do MOTOR A
ENB	Entrada	Controle de saída do motor: o estado "baixo (0V)" desativa o MOTOR B
IN3	Entrada	Controle de direção do MOTOR B
IN4	Entrada	Controle de direção do MOTOR B
MOTOR A	Saída	Saída para o MOTOR A
VMS e GND	-	Conexão para fonte de alimentação externa (6V a 35V)
ENA	Entrada	Controle de saída do motor: o estado "baixo (0V)" desativa o MOTOR A
IN1	Entrada	Controle de direção do MOTOR A
IN2	Entrada	Controle de direção do MOTOR A
ENB	Entrada	Controle de saída do motor: o estado "baixo (0V)" desativa o MOTOR B
IN3	Entrada	Controle de direção do MOTOR B
IN4	Entrada	Controle de direção do MOTOR B
MOTOR A	Saída	Saída para o MOTOR A

Basicamente, o funcionamento deste driver consiste no uso de uma dupla ponte H, conforme figura (15). A ponte H, é uma forma de efetuar o controle de um motor de passo, ou motor de corrente contínua sem a necessidade do uso de recursos mecânicos de chaveamento, tais como chaves e relés.

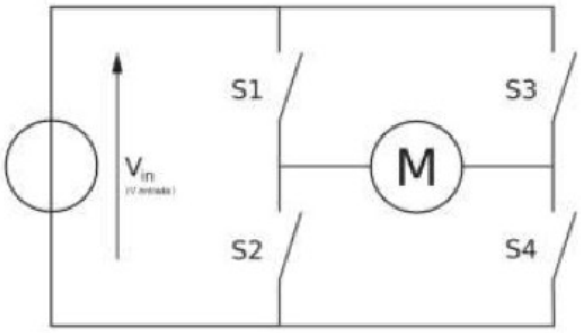

Figura 15: Ponte H (ST, 2015).

Para efetuar o controle do sentido da corrente, como indicado na figura(15), é necessário ativar as chaves S1 e S4, caso o sentido desejado seja o oposto, é necessário ativar as chaves S2 e S3. De modo a automatizar esse processo, o chip L298N é constituído internamente conforme figura (16) (ST, 2015).

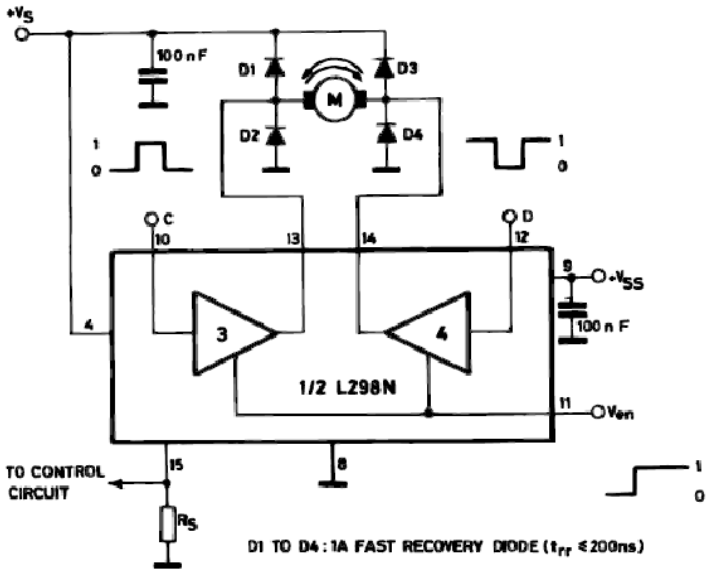

Figura 16: Esquema interno do L298N (ST, 2015).

De acordo com o esquemático do chip, pode-se observar que ao invés de chaves, a regulagem da corrente é feita através de diodos, que ao receber os sinais C e D, que são respectivamente os pinos 10 e 12 do componente, os mesmos são enviados devidamente pelos amplificadores operacionais e efetuam a passagem ou o bloqueio da corrente através do motor. Dessa forma o controle do mesmo torna-se uma tarefa simplificada. É válido observar que para que haja rotação, é necessário ativar uma determinada tensão enable, essa tensão fornecida ao pino 11, é capaz de fazer, por exemplo, uma parada rápida, caso seja necessário, pois esta funciona como acionador dos amplificadores operacionais, caso não haja tensão de entrada neste pino não haverá tensão de saída nos mesmos (ST, 2015).

Outro aspecto relevante da ponte H, é o fato de que os diodos internos possuem conexões que facilitam a funcionalidade, como pode ser observado nos diodos D1 e D3, estes estão ligados a um capacitor carregado, enquanto os diodos D2 e D4, estão ligados ao GND. Dessa forma a corrente irá fluir similarmente ao modelo com chaves de entre D1 e D4 ou entre D2 e D3, determinando assim a continuidade no giro do motor. Dessa forma, o controle é efetuado conforme tabela (9) (ST, 2015).

Tabela 9: Funcionamento do driver de motor (ST, 2015).

Entradas		**Funções**
EN = 1	C = 1; D = 0;	Gira no sentido horário
EN = 1	C = 0; D = 1;	Gira no sentido anti-horário
EN = 1	C = D	Interrupção do funcionamento do Motor
EN = 0	C = x; D = x	Não há giro do motor

A escolha do Módulo *L298N* justifica-se visto que esse módulo implementa uma

dupla ponte H, de modo que os dois motores da plataforma podem ser controlados utilizando somente uma unidade do módulo. Além disso, esse módulo permite o controle de velocidade por PWM.

2.9 Sensor de temperatura DS18B20

O sensor DS18B20 (figura (17)) foi o escolhido para fazer a medição da temperatura da água. O DS18B20 é um sensor de temperatura digital que opera na faixa de temperatura de -55ºC até +125ºC e com precisão de 0, 5ºC (na faxa de -10ºC a 85ºC). Além disso, ele oferece resolução de 9 bits a 12 bits, configuráveis.

Figura 17: Sensor de temperatura DS18B20 (HUINFINITO, 2015b).

O sensor de temperatura DS18B20 possui três terminais: um terminal de alimentação, um terminal de referência e um terminal de dados (figura (18)).

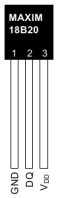

Figura 18: Pinagem sensor de temperatura (DALLAS, 2015).

A escolha do sensor DS18B20 justifica-se visto que ele atende ao range de temperatura do lago Paranoá. Além disso esse sensor já vêm a prova d'água, ideal para a aplicação.

3 Desenvolvimento

Neste capítulo é apresentada uma descrição geral do sistema proposto para monitoramento da temperatura da água do Lago Paranoá. São apresentados detalhes das implementações realizadas levando em consideração aspectos como materiais utilizados, arquiteturas, plataformas de desenvolvimento e ferramentas.

3.1 Solução Proposta

De forma geral, a solução proposta é um veículo autônomo de superfície (ASV) dotado de sensores embarcados em um dispositivo móvel com sistema operacional Android e um sensor para aquisição de dados de um parâmetro de qualidade da água (temperatura) e que se comunica enviando dados e recebendo comandos de uma estação remota controlada por um usuário, através de uma rede 3G.

A figura (19) mostra um esquemático da solução geral proposta.

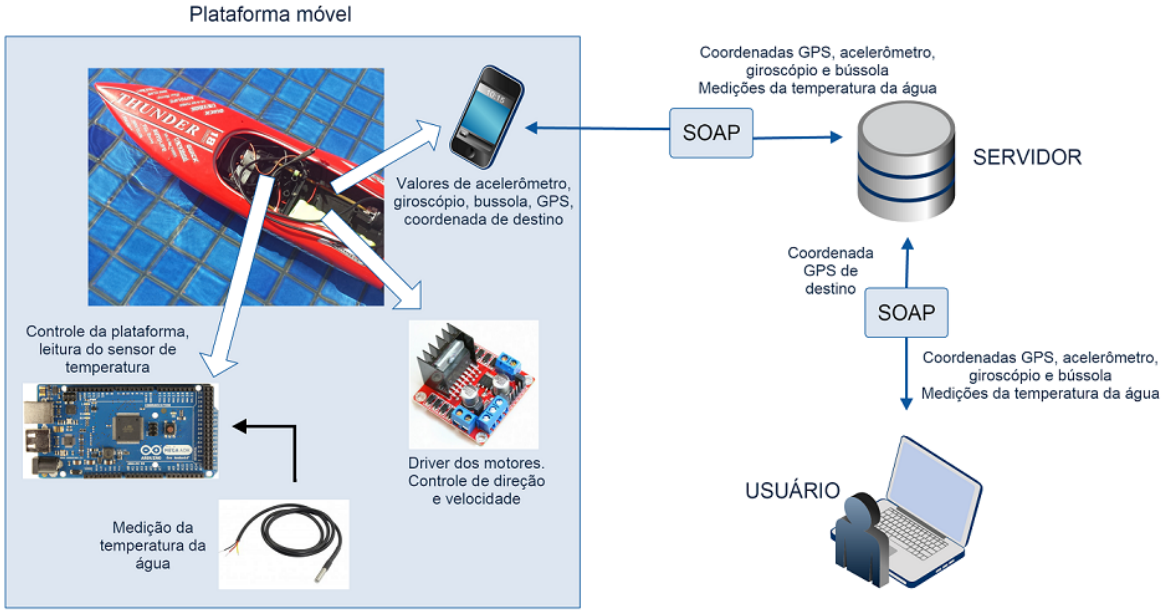

Figura 19: Esquemático do sistema proposto.

A plataforma ASV conta com os seguintes componentes de hardware: (a) Um aparelho smartphone Android que usa comunicação 3G e faz a leitura dos seguintes sensores embarcados: GPS, bussola digital, acelerômetro, giroscópio e magnetômetro, pois tais sensores são interessantes para navegação em robótica; (b) Kit Arduino ADK para controle dos motores da plataforma e coleta e processamento dos dados do sensor de temperatura,

além da comunicação via USB com o dispositivo Android; (c) Duas Bateria NI-MH 9.6V 600mAh.

A interface de usuário será implementada em um PC comercial e permite indicar o ponto de destino desejado e visualizar os dados de qualidade da água. O Servidor realiza a integração via HTTP com o ASV e o PC de forma a armazenar os dados referentes ao tempo de coleta, coordenada e valores das medições do parâmetros da água.

Como caso de estudo foi escolhida apenas a temperatura como parâmetro de estimação da qualidade da água. Esta escolha se justifica considerando o alto custo de aquisição das sondas de medição de pH e oxigênio dissolvido, por exemplo, os quais requerem de um processo de calibração constante.

Como ambiente de estudo escolheu-se estimar os parâmetros de qualidade da água do lago Paranoá, delimitando a área de atuação da plataforma no pier da asa norte, tal como mostrado na figura (20).

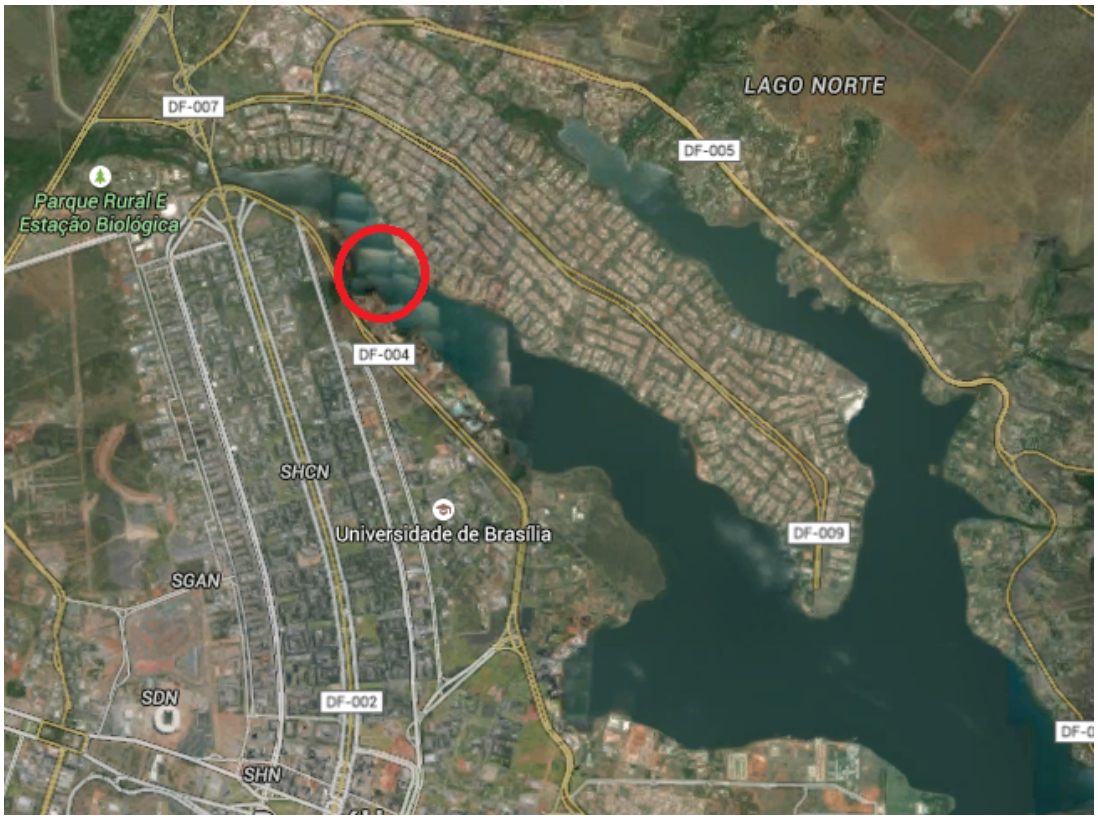

Figura 20: Zona de atuação da plataforma no lago Paranoa.

Neste trabalho o foco foi viabilizar a comunicação entre a plataforma móvel e um usuário na estação remota, de modo que este usuário possa visualizar o valor do parâmetro medido e enviar comandos para a plataforma móvel, através da interface gráfica do usuário. Entretanto não foi realizado um estudo da variação de temperatura das águas superficiais do Lago Paranoá.

A implementação da comunicação através de um *webservice* será feita apenas para integrar as aplicações Android e a interface gráfica do usuário e para armazenar dados relevantes ao escopo do trabalho. É importante salientar que neste trabalho não serão implementado os aspectos de segurança da informação referente a essa comunicação.

O sistema de navegação da plataforma será simplificado, de maneira que não serão implementados algoritmos de navegação sofisticados como: desvio de obstáculos, detecção de tombamento, detecção de baixa profundidade e encalhamento, análise da trajetória ótima e estabilização espacial completa da plataforma. A plataforma ficará limitada a atuar dentro de uma zona escolhida do lago Paranoá a pelo menos 20 metros de distância (considerando erros de medição e estimação do GPS) das margens do lago.

Na prototipação da plataforma foi utilizado um modelo comercial de barco elétrico, de modo que não foi desenvolvido uma estrutura eletro-mecânica para a plataforma móvel. Além disso, não foi realizado um estudo da cinemática da plataforma móvel dentro do ambiente ideal ou real.

3.1.1 Arquitetura do Sistema

Na figura (21) é apresentado uma arquitetura geral da proposta de solução. O controlador escolhido foi o Arduino ADK. Tal escolha se justifica principalmente porque o Arduino ADK implementa o padrão de referência ADK criado pela Google, e por isso permite a comunicação com dispositivos Android.

Como dispositivo de comunicação, escolheu-se um aparelho smartphone. Tal escolha justifica-se principalmente por três motivos. O primeiro motivo é pelo fato dos smartphones já possuírem uma série de sensores tais como GPS, acelerômetro, magnetômetro, giroscópio e entre outros que podem ser utilizados na navegação e eliminam a necessidade de aquisição de sensores externos. O segundo motivo é pela possibilidade de comunicação entre dispositivos Androids e dispositivos externos através do protocolo Open Accessory. O terceiro motivo é possibilidade da utilização da cobertura de rede 3G na comunicação com um servidor remoto, o que dispensa a utilização de transceptores de radio-frequência externos.

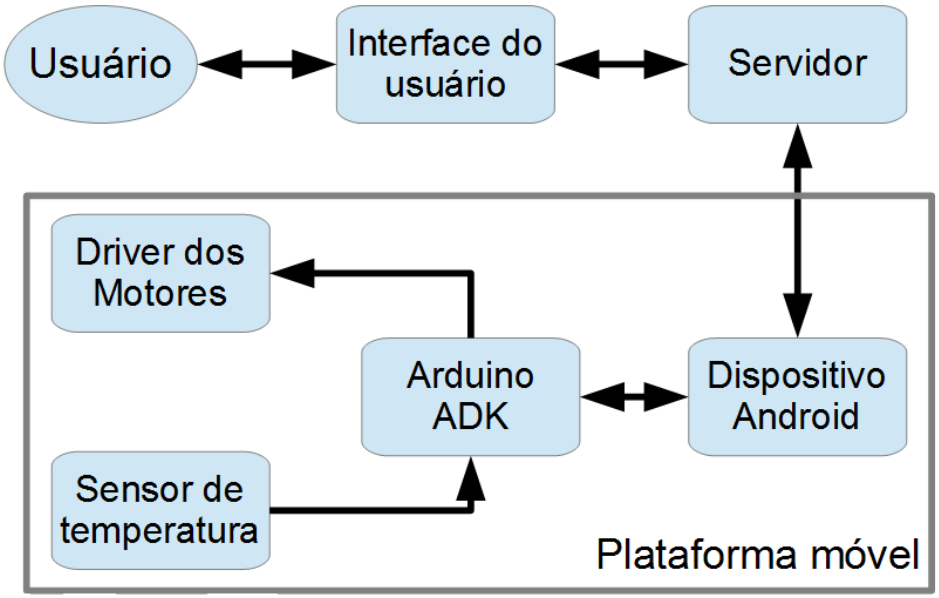

Figura 21: Diagrama de comunicação do sistema proposto.

No diagrama de comunicação é possível verificar o fluxo de informações na arquitetura escolhida. O usuário envia um comando ao servidor, através da interface do usuário. O dispositivo Android faz a leitura do comando no servidor, e envia requisições ao Arduino ADK e realiza o acionamento do driver dos motores e a leitura do sensor de temperatura. No caso da leitura do sensor, o Arduino ADK envia o valor ao dispositivo Android, que por sua vez envia a informação ao servidor. Finalmente, o usuário acessa o valor do sensor através da interface do usuário.

3.1.2 Interface Gráfica do Usuário

A interface Gráfica do Usuário é a responsável por permitir que o usuário envie comandos remotamente à plataforma móvel e que o usuário acesse as informações contidas no servidor referentes a leitura realizadas pelos sensores conectados a plataforma ADK, conforme figura (22).

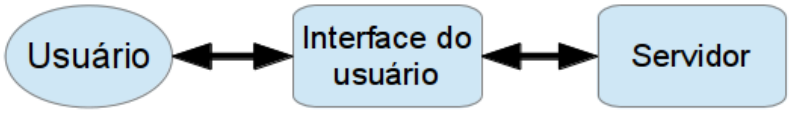

Figura 22: Diagrama de comunicação da interface do usuário.

No diagrama de comunicação da interface gráfica é possível verificar o fluxo de informações. O usuário envia comandos ao servidor e faz a leitura de informações do servidor através da interface gráfica do usuário.

3.1.3 Servidor

O servidor é responsável por permitir a comunicação entre o aparelho móvel que possui o Android e o usuário em uma estação remota, tal como mostra a figura (23). Além de realizar a comunicação, o servidor salva os dados recebidos da plataforma ASV.

Figura 23: Diagrama de comunicação do servidor.

3.1.4 Android

O Android é responsável pela comunicação com o webservice, leitura dos sensores embarcados no mesmo (GPS, acelerômetro, giroscópio, magnetômetro e bussola) e pela comunicação com a plataforma ADK, conforme figura (24). Alem disso o dispositivo Android e responsável pela navegação da plataforma controlando quando acionar o driver dos motores.

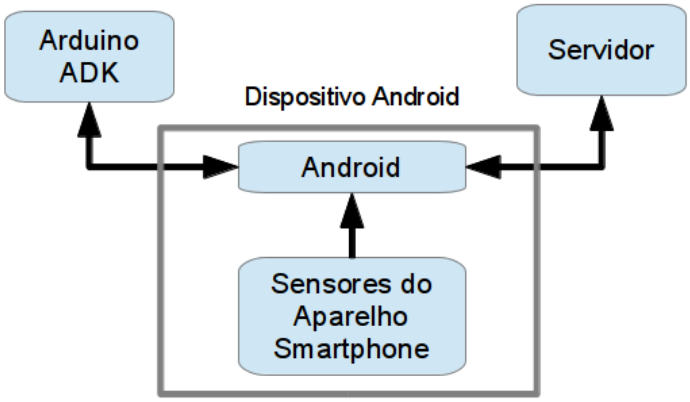

Figura 24: Diagrama de comunicação do Android.

No diagrama de comunicação do Android é possível verificar o fluxo de informações no dispositivo Android. O dispositivo Android faz a leitura dos comandos no servidor, e faz a comunicação com o Arduino ADK enviando dados de controle e recebendo dados provenientes do sensor de temperatura. Além disso, o dispositivo Android possui sensores embarcados, que são lidos, e tais informações são enviadas ao servidor.

3.1.5 Arduino ADK

O Arduino ADK e o responsável por fazer o controle do driver dos motores (Ponte H) da plataforma a partir de um comando recebido do Android, fazer a leitura do sensor

e envia-las ao Android, conforme a figura (25).

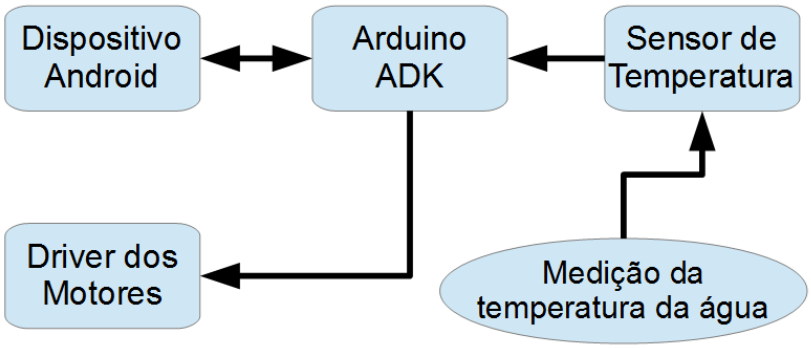

Figura 25: Diagrama de comunicação do Arduino ADK.

No diagrama de comunicação é possível verificar o fluxo de informações no Arduino ADK. O Arduino recebe comandos do dispositivo Android e, a partir de tais comandos, faz o acionamento dos motores através do driver dos motores. Adicionalmente, realiza a leitura do sensor de temperatura constantemente.

3.1.6 Navegação

A navegação da plataforma móvel é, totalmente, orientada por posição. Desse modo, todas as tomadas de decisão e critério de parada do algoritmo de navegação são baseados nas informações da posição da plataforma e da coordenada de destino. Na figura (26) está descrita, em forma de fluxograma, o algoritmo de navegação da plataforma móvel.

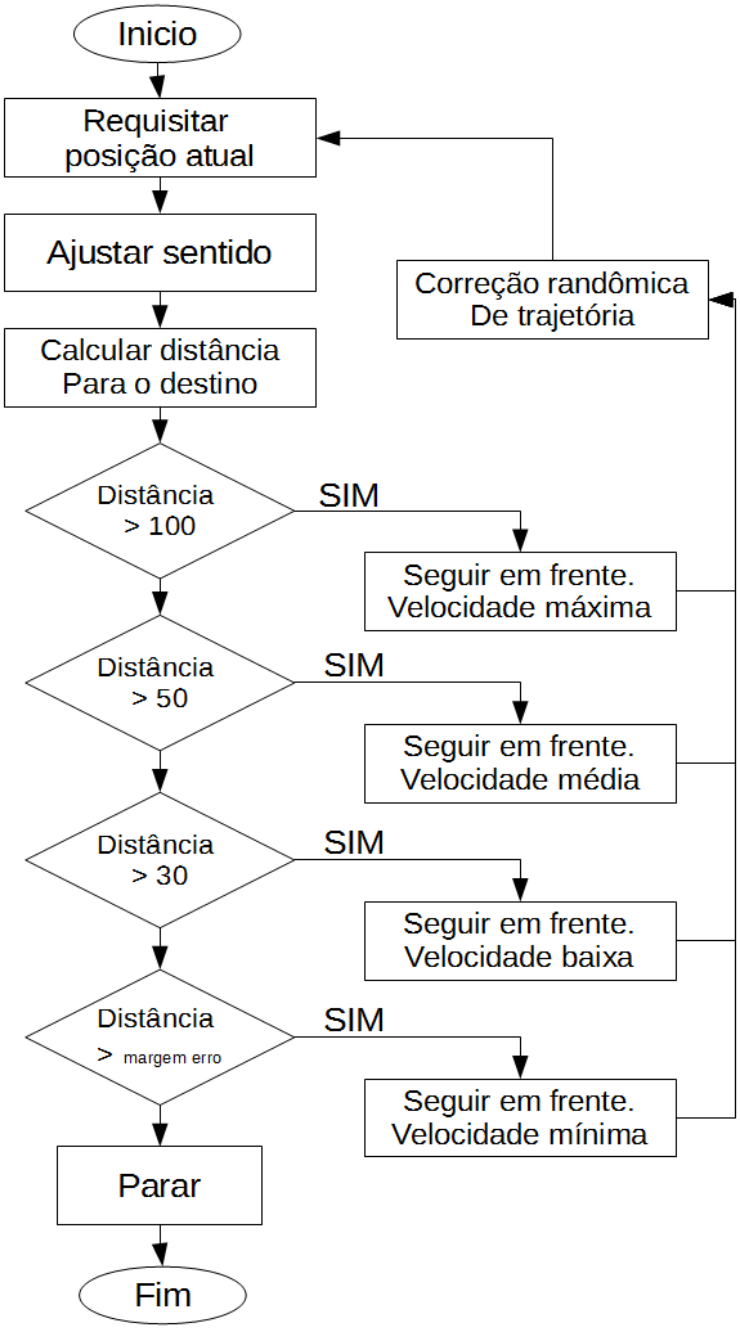

Figura 26: Fluxograma do algoritmo de navegação.

Ao iniciar o algoritmo, o primeiro procedimento é pegar a *posição atual*. O objetivo desse procedimento é obter as coordenadas geográficas da plataforma móvel. Essa tarefa é realizada pelo sistema Android, que, via software faz uma requisição ao hardware de GPS e como resultado obtém a posição atual (em coordenadas geográficas) da plataforma móvel.

O segundo procedimento do algoritmo é *ajustar sentido*. O objetivo desse procedimento é direcionar a plataforma para a coordenada de destino. Tal procedimento foi

solucionado com a descrição de uma sub-rotina ilustrada na figura (27).

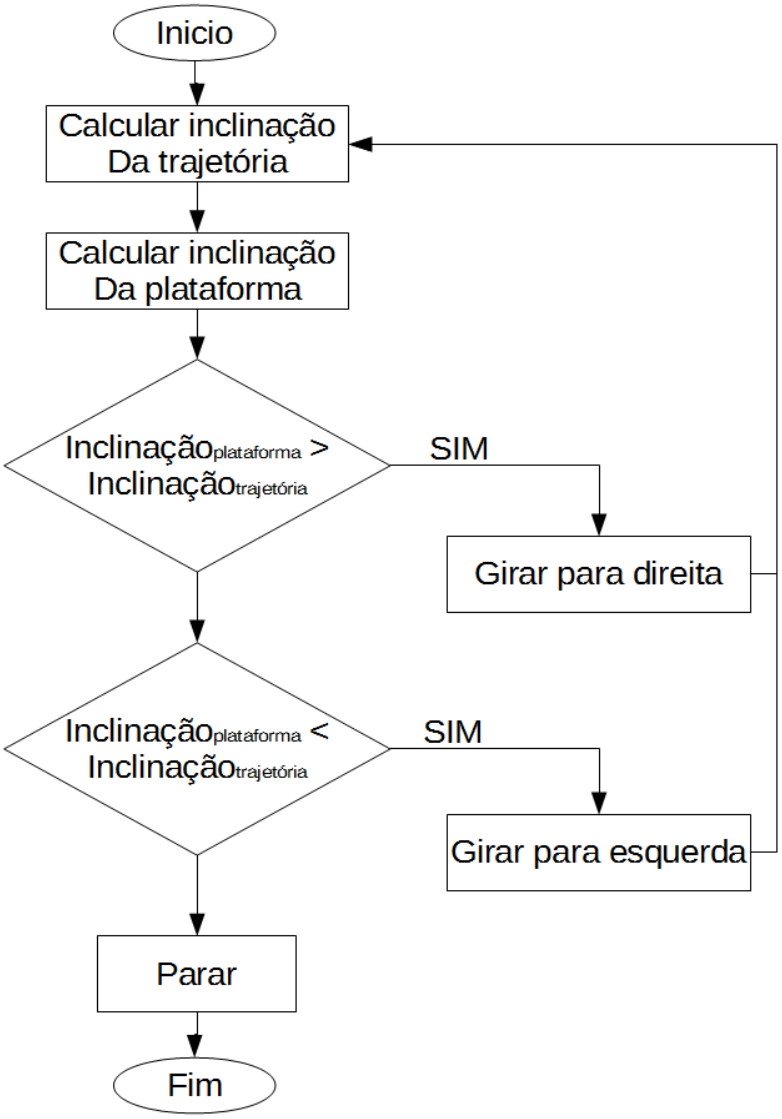

Figura 27: Fluxograma da sub-rotina ajustar sentido.

O primeiro problema de direcionar a plataforma móvel é descobrir para qual direção ela está apontada. Esse problema foi solucionado do seguinte modo: com a plataforma parada, pega-se a posição da mesma; em seguida aciona-se plataforma para seguir em frente durante um segundo; após um segundo, pega-se a posição da mesma, conforme figura (28).

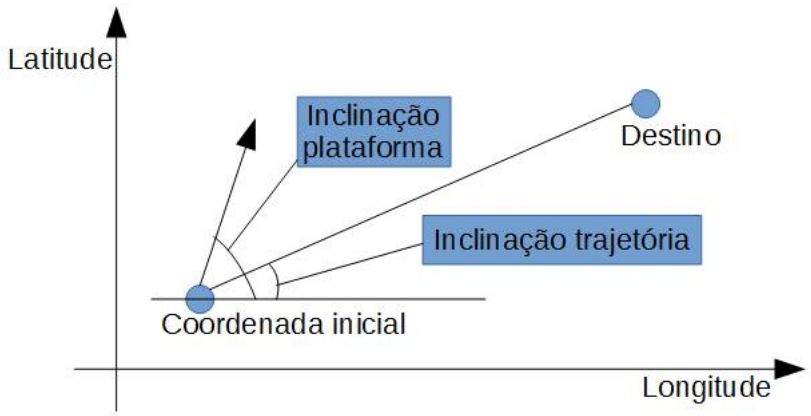

Figura 28: Método de calculo da inclinação da plataforma.

Através das coordenadas do ponto de partida e da posição da plataforma é possível calcular (equação 3.1) a inclinação da reta entre as duas coordenadas, que é, aproximadamente, a inclinação da plataforma em relação ao plano da superfície (no caso, o Lago Paranoá).

$$inclinacao = \arctan\left(\frac{latitude_{posicaofinal} - latitude_{posicaoinicial}}{longitude_{posicaofinal} - longitude_{posicaoinicial}}\right) \tag{3.1}$$

Utilizando o mesmo princípio da inclinação da plataforma, é possível calcular a inclinação da trajetória descrita pela linha reta que conecta a coordenada atual da plataforma à coordenada de destino (figura (29)).

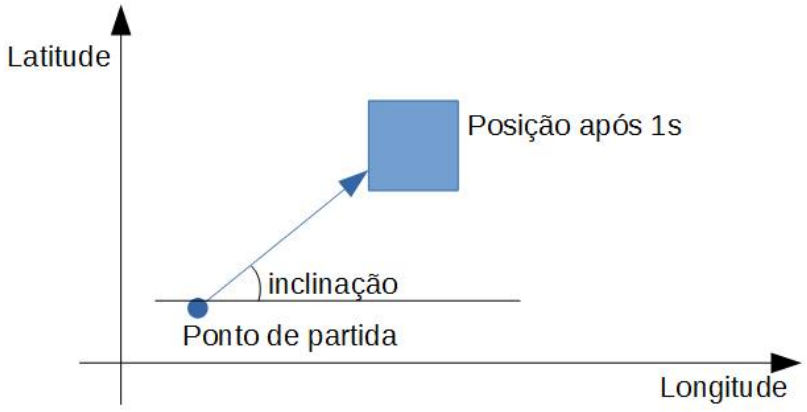

Figura 29: Método de calculo da inclinação da trajetória.

Caso a inclinação da plataforma seja maior que a inclinação da trajetória, a plataforma deve girar à direita. Caso a inclinação da plataforma seja menor que a inclinação da trajetória, a plataforma deve girar a esquerda. Finalmente, caso a inclinação da plataforma seja próxima (considerando somente a margem de erro de 20 metros) à inclinação

da trajetória, conclui-se que a plataforma está na direção correta, e a sub-rotina de ajuste de sentido é finalizado.

O terceiro procedimento do algoritmo de navegação é o cálculo da distância. O objetivo dessa etapa é obter a distância entre a plataforma e a coordenada de destino. A velocidade da plataforma, mensurada pelo duty-cycle do PWM, é definida de acordo com distância. A distância é obtida calculando-se a magnitude da reta que conecta a coordenada da posição atual à coordenada de destino, conforme a equação 3.2 (VENESS, 2015). Caso a distância seja maior que 100 metros, a plataforma deve seguir em frente com velocidade máxima (duty-cycle igual a 255). Caso a distância seja menor que 100 metros e maior que 50 metros, a plataforma deve seguir em frente com velocidade média (duty-cycle igual a 150). Caso a distância seja menor que 50 metros e maior que 30 metros, a plataforma deve seguir em frente com velocidade baixa (duty-cycle igual a 50). Caso a distância seja menor que 30 metros e maior que a margem de erro (10 metros), a plataforma deve seguir em frente com velocidade mínima (duty-cycle igual a 20). Caso a distância esteja dentro da margem de erro, a plataforma deve parar, e o algoritmo é finalizado. .

$$distancia = \arccos(\sin(lat_{inicial})\sin(lat_{final}) + \cos(lat_{inicial})cos(lat_{final})cos(lng_{inicial} - lng_{final})) \tag{3.2}$$

Em quaisquer dos casos em que a distância é maior que a margem de erro, o algoritmo executa uma sub-rotina de correção randômica de trajetória, ilustrada na figura (30). O objetivo dessa correção randômica é retirar o algoritmo de navegação de trajetórias divergentes em virtude de possíveis falhas da sub-rotina que ajusta o sentido da plataforma móvel.

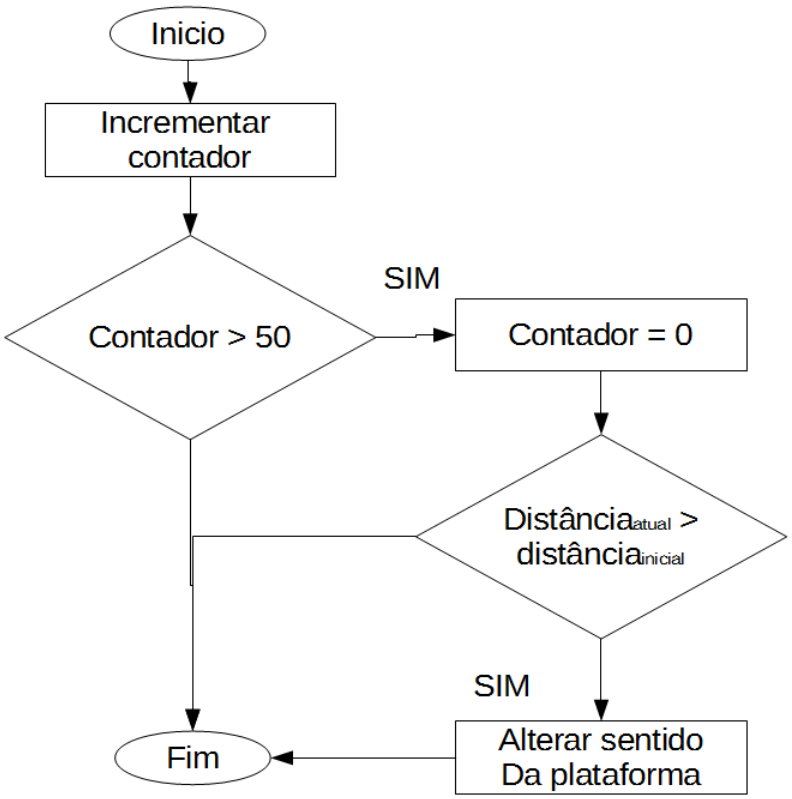

Figura 30: Fluxograma da sub-rotina de correção randômica.

A sub-rotina de correção possui um contador, o qual é incrementado sempre que a sub-rotina é executada. Quando contador é incrementado mais de 50 vezes, ele é zerado e o algoritmo verifica se a distância (distância entre posição da plataforma e o destino) está aumentando. Caso a distância esteja aumentando, a direção da plataforma é alterada randomicamente.

3.2 Implementações

3.2.1 Implementação da interface gráfica

A interface foi implementada utilizando um notebook da Samsung modelo NP275E4E processador AMD E1-1500 APU 1.48GHz, Memoria RAM 4GB e sistema operacional Windows 8.1 e ambiente de desenvolvimento Microsoft Visual Studio 2013 utilizando linguagem C#.

O programa desenvolvido possui basicamente três funções: Apresentar as coordenadas atuais da plataforma móvel, definir a coordenada de destino da plataforma e apresentar as medições realizadas ao usuário.

A apresentação das coordenadas atuais e definição das coordenadas de destino

foram implementadas através do GMapControl da library GMap.NET que é uma API de desenvolvimento open-source feita para ASP.NET para utilização de mapas. A partir dessa API foi importado para dentro do programa, um mapa para controle e visualização da plataforma móvel, sendo necessário apenas sua inicialização e configuração, conforme codigo a seguir.

```
1 /** Inicializar Maps e Configura mapa */
2 gMapControl.MapProvider = GMap.NET.MapProviders.
     BingHybridMapProvider.Instance;
3 GMap.NET.GMaps.Instance.Mode = GMap.NET.AccessMode.ServerOnly;
```

Para comunicação com o webservice e controle da plataforma através do dispositivo Android, foi necessário adicionar uma referencia web ao projeto do Visual Studio. Em seguida foi realizada a configuração do programa para consumir o web service e realizar requisições no mesmo. Essas requisições são realizadas constantemente, a cada 1 segundo, com o intuito de atualizar os valores provenientes dos sensores embarcados no dispositivo Android assim como os valores do sensor de temperatura, para o usuário.

```
1     /** variavel para comunicacao com servidor */
2        webServiceTCC.ComunicacaoPortTypeClient servidor = new
            webServiceTCC.ComunicacaoPortTypeClient();
```

Depois de adicionar a *Service Reference* (endereço do webservice) ao projeto do Visual Studio e configura-lo para consumir o webservice, foram implementados métodos para realizarem requisições ao webservice afim de implementar a comunicação entre eles. O código a seguir ilustra uma dessas requisições.

```
1    /** servidor.() --> () = metodo Dentro do webservice */
2    servidor.envioPosicaoFinal("n", LatitudeFinal, LongitudeFinal
        );
```

A definição da posição de destino a partir do mapa importado foi implementada dentro do *evento* de click do botão esquerdo do mouse sobre o mapa, conforme mostra o código a seguir.

```
1     private void gMapControl_MouseClick(object sender,
         MouseEventArgs e)
2        {
3            /** Pega Coordenadas do click do mouse */
4            if (e.Button == System.Windows.Forms.MouseButtons.
                Left)
5            {
6                /** Remove marcador anterior */
7                gMapControl.Overlays.Remove(markersOverlay2);
```

```
8                   gMapControl.ReloadMap();
9
10                  /** pega latitude clicada no mapa */
11                  latitude_final = gMapControl.FromLocalToLatLng(e.
                        X, e.Y).Lat;
12                  longitude_final = gMapControl.FromLocalToLatLng(e
                        .X, e.Y).Lng;
13
14                  /**add marcador */
15                  addMArcador_posFinal(latitude_final,
                        longitude_final);
16
17                  /** Atualizar Txtbox */
18                  txtLatFinal.Text = Convert.ToString(
                        latitude_final);
19                  txtLngFinal.Text = Convert.ToString(
                        longitude_final);
20
21              }
22          }
```

3.2.2 Implementação do webservice

Para criar o servidor web foi utilizado um NetBook da Acer modelo KAV10 processador 2x Intel(R) Atom(TM) CPU N80 1.66GHZ, Memoria RAM 1GB, sistema operacional Ubuntu 14.04.1 LTS e plataforma de desenvolvimento Eclipse IDE for Java EE Developers.

No que tange a implementação do servidor foi utilizado o Apache Tomcat que é um servidor web Java e implementou-se um web service utilizando o framework Xfire que é um framework *Open Source* escrito em Java que facilita a construção de webservices. Utilizando o Xfire a implementação do web service é bastante simplificada. Em um primeiro momento configura-se o arquivo web.xml da aplicação web para adicionar o servlet do Xfire. Em seguida basta apenas definir uma interface e uma classe que a implementa e por fim declarar essas classes no arquivo de configuração services.xml. A partir disso é possível publicar o web services utilizando o Apache Tomcat.

O WebService tem como função viabilizar a comunicação entre o dispositivo Android e o programa controlador, a partir disso no web service apos sua configuração, são implementados vários métodos para que o dispositivo Android e o programa controlador possam realizar requisições no servidor alterando alguma variável ou pegando algum dado. O código a seguir mostra todas as interfaces que foram implementadas no webservice.

```
1     /** metodo q pega dados do celular */
2        public String pegaDados_Celular(String sensor_temp,
            String lat, String lng, String alt,

            String aceX, String aceY, String aceZ, String giroX,
            String giroY, String giroZ,
                                                          String
             magX, String magY, String magZ, String azi, String pit
            , String rol);
3
4        /** metodo controle do barco pela GUI */
5        public String envioPosicaoFinal(String n, String
            lat_final, String lng_final);
6        public String controle(String action);
7
8        /** metodos atualizados da GUI */
9        public String getLatitude_GUI();
10       public String getLongitude_GUI();
11       public String getAltitude_GUI();
12       public String getAceX_GUI();
13       public String getAceY_GUI();
14       public String getAceZ_GUI();
15       public String getGiroX_GUI();
16       public String getGiroY_GUI();
17       public String getGiroZ_GUI();
18       public String getMagX_GUI();
19       public String getMagY_GUI();
20       public String getMagZ_GUI();
21       public String getAzi_GUI();
22       public String getPit_GUI();
23       public String getRol_GUI();
24       public String getSensorTemp_GUI();
```

O *webservice* além de promover a comunicação ele guarda os dados recebidos do dispositivo Android em um arquivo texto. Para guardar esses dados e não interferir na comunicação, caso o processo estivesse ocupado escrevendo no arquivo, o método para escrita do arquivo foi implementado em uma *Thread* diferente. O código a seguir mostra o método para escrita do arquivo texto.

```
1  /** escreve no arquivo criado */
2        public void escreverTXT(String sensor_temp,String lat,
            String lng, String alt,
```

```
            String aceX, String aceY, String aceZ,  String giroX,
            String giroY, String giroZ,

            String magX, String magY, String magZ, String azi,
            String pit, String rol)
3       {
4               /** cria arquivo de texto e escreve */
5               try {
6                       arquivo_coord = new BufferedWriter(new
                           FileWriter("/home/fmachado/Documentos/
                           Dados"+".txt",true));
7                       arquivo_coord.write("Sensor:"+sensor_temp
                           +" "+"GPS:"+lat+";"+lng+";"+alt+" "+

                           "Acelerometro:"+aceX+";"+aceY+";"+aceZ+
                           " "+

                           "Giroscopio:"+giroX+";"+giroY+";"+giroZ
                           +" "+

                           "Magnetometro:"+magX+";"+magY+";"+magZ+
                           " "+"Posicao de Referencia da Terra:"+
                           azi+";"+pit+";"+rol+"\n");
8                       arquivo_coord.close();
9                       verificaTXT = true;
10              }catch (IOException e) {
11                      e.printStackTrace();
12              }
13      }
```

3.2.3 Implementações com o Android

Para criar o projeto do Android foi utilizado o *Android Studio* que é uma IDE oficial para desenvolvimento de aplicativos Android, baseado na IntelliJ IDEA. Para a realização dos testes de comunicação, leitura de dados de sensores embarcados, sensor de temperatura e comunicação com o microcontrolador foi utilizado um aparelho da Samsung Galaxy S4 Active modelo GT-I9295 com a versão 4.4.2 do Android.

3.2.3.1 Sensores embarcados no dispositivo Android

Para obter dados de um sensor no Android foi implementado uma instância do *SensorManager* e depois foi utilizado o método *sensorManager.getDefaultSensor()*, conforme o código a seguir.

```
1 private SensorManager sensorManager;
2 private Sensor gyroscope;
3
4 sensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
5 gyroscope = sensorManager.getDefaultSensor(Sensor.TYPE_GYROSCOPE)
    ;
```

Os sensores utilizados e seus respectivos tipos estão descritos na tabela (10).

Tabela 10: Relação dos sensores utilizados e seus respectivos tipos.

Sensor	**Tipo**
Acelerômetro	TYPE_LINEAR_ACCELEROMETER
Giroscópio	TYPE_GYROSCOPE
Magnetômetro	TYPE_MAGNETIC_FIELD
Posicao em relacao ao referencial da Terra	TYPE_ORIENTATION

Depois de obter o objeto sensor foi iniciado o monitoramento através do método *registerListener*, conforme o código a seguir:

```
1 /**
2 us = microsegundos
3 * SENSOR_DELAY_NORMAL       = delay 200.000 us.
4 * SENSOR_DELAY_UI             = delay 60.000 us.
5 * SENSOR_DELAY_GAME          = delay 20.000 us.
6 * SENSOR_DELAY_FASTEST       = delay 0.0 us.
7 */
8 sensorManager.registerListener(this,gyroscope,SensorManager.
    SENSOR_DELAY_NORMAL);
```

Em seguida para receber os valores gerados pelos sensores, a classe precisa implementar a interface *SensorEventListener* que possui dois métodos: *onAccuracyChanged* e *onSensorChanged*.

- **onAccuracyChanged:** chamado para informar que a precisão do sensor mudou;
- **onSensorChanged:** chamado toda vez que tiver uma informação no sensor para que a aplicação leia os valores do sensor;

O código a seguir mostra esses métodos e como são obtido dados do giroscópio embarcado no dispositivo Android:

```
1    float[] giroscopio = new float[3]; /** array para dados
        giroscopio */
2
3    @Override
4    public void onSensorChanged(SensorEvent event) {
5        if(event.sensor.getType() == Sensor.TYPE_GYROSCOPE){
6            giroscopio[0] = event.values[0]; /** eixo x */
7                giroscopio[1] = event.values[1]; /** eixo y */
8            giroscopio[2] = event.values[2]; /** eixo z */
9        }
10   }
11   @Override
12   public void onAccuracyChanged(Sensor sensor, int accuracy) {
        }
```

Os dados provenientes dos outros três sensores foram implementados da mesma forma que os dados do sensor giroscópio foram obtidos.

No caso do GPS é necessário criar uma classe que implementa a interface *android.location.LocationListener*. A partir da implementação da interface *LocationListener*, basta chamar o método *requestlocationUpdates(provedor, tempoMinimo, distanciaMinima, LocationListener)*.

```
1    private LocationManager lm;
2
3    lm = (LocationManager) getSystemService(LOCATION_SERVICE);
4    lm.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0,
        this);
```

Os parâmetros do método *requestlocationUpdates(provedor, tempoMinimo, distanciaMinima, LocationListener)* estão na tabela (11), para iniciar o monitoramento do GPS.

Tabela 11: Parâmetros do método LocationManager.requestlocationUpdates

Parâmetro	**Descrição**
String provedor	GPS_PROVIDER ou NETWORK_PROVIDER
long tempoMinimo	Tempo em milissegundos para receber novas coordenadas
float distanciaMinima	Distância mínima em metros para receber novas coordenadas
LocationListener listener	Implementação da interface LocationListener

Ha vários métodos nessa interface, embora foi utilizado apenas o método *onLo-*

cationChanged(location), esse método é chamado automaticamente pelo Android sempre que a localização GPS for alterada.

```
1     Double latitude;
2     Double longitude;
3     Double altitude;
4
5     public void onLocationChanged(Location locat) {
6         latitude = locat.getLatitude();
7         longitude = locat.getLongitude();
8         altitude = locat.getAltitude();
9     }
```

3.2.3.2 Comunicação com servidor

O Android não possui nenhuma API nativa para acessar um *webservices*, no entanto pode-se usar qualquer outra biblioteca e incorpora-la a seu projeto. Vale ressaltar que essa biblioteca para consumir o *webservices* tem que ser leve e compacta, pois a aplicação vai executar em um dispositivo móvel, que tem seu poder de processamento reduzido comparado a um computador convencional.

A partir disso foi escolhido o *framework ksoap2*, que é uma biblioteca leve para consumir *webservices*, especialmente construída para dispositivos móveis com menor capacidade de processamento.

Para realizar a requisição SOAP foi definido o método a ser consultado, como esse método tem vários argumentos foi necessario adiciona-los ao objeto SOAP, conforme código a seguir:

```
1     public String envio(String sensor_temp, String Lat, String
         Lng, String Alt,
2                         String aceX, String aceY, String aceZ,
3                         String giroX, String giroY, String giroZ,
4                         String magX, String magY, String magZ,
5                         String azi, String pit, String rol)
                             throws IOException,
                             XmlPullParserException {
6         /** Namespace e nome para o objeto SOAP */
7         String namespace = "http://xfire.codehaus.org/Comunicacao
             ";
8         String name = "pegaDados_Celular";
9         String method = "Comunicacao";
10        SoapObject soap = new SoapObject(namespace, name);
11
```

```
12        /** Adiciona os parametros para a layout */
13        soap.addProperty("sensor_temp", sensor_temp);
14         soap.addProperty("lat", Lat);
15         soap.addProperty("lng", Lng);
16         soap.addProperty("alt", Alt);
17         soap.addProperty("aceX", aceX);
18         soap.addProperty("aceY", aceY);
19         soap.addProperty("aceZ", aceZ);
20         soap.addProperty("giroX",giroX);
21         soap.addProperty("giroY",giroY);
22         soap.addProperty("giroZ",giroZ);
23         soap.addProperty("magX", magX);
24         soap.addProperty("magY", magY);
25         soap.addProperty("magZ", magZ);
26         soap.addProperty("azi", azi);
27         soap.addProperty("pit", pit);
28         soap.addProperty("rol", rol);
```

Onde a string **namespace** é o serviço criado do *webservices*, e a string **name** é o método que vai ser chamado no *webservices*. Para isso, o nome do método e as variáveis dos parâmetros precisam ser exatamente iguais aos que foram definidos no *WSDL* do *webservices*.

Em seguida foi criado o elemento envelope da mensagem SOAP, conforme o código a seguir:

```
1         SoapSerializationEnvelope envelope = new
             SoapSerializationEnvelope(SoapEnvelope.VER11);
2         envelope.setOutputSoapObject(soap);
```

A partir do envelope criado, o *framework ksoap2* vai gerar o XML correspondente, sendo esse XML o envelope da mensagem SOAP. Com esse envelope foi criado o objeto do tipo *HttpTransport* para preparar a requisição HTTP e em seguida pegar o resultado, conforme código a seguir:

```
1         /** Cria o HttpTransport para enviar os dados (SOAP) */
2         HttpTransportSE httpTransport = new HttpTransport(url);
3
4         /** Faz a requisicao */
5         httpTransport.call(method, envelope);
6
7         /** Recupera o resultado */
8         Object resposta = envelope.getResponse();
```

3.2.3.3 Comunicaçao com Arduino ADK

O sistema Android disponibiliza duas bibliotecas que auxiliam no reconhecimento de dispositivos externos: *UsbAccessory* e *UsbManager*. A biblioteca *UsbAccessory* define a classe *BroadcastReceiver* que é um receptor de transmissão, que auxilia na comunicação do Android com dispositivos externos, utilizando os métodos *openAccessory* e *closeAccessory* para abrir e fechar a comunicação.

No método *openAccessory* o método do serviço USB é *delegate* (tipo de variável que guarda o endereço de um método), de modo, a obter um *FileDescriptor* para o dispositivo externo. O *FileDescriptor* gerencia os canais de entrada e saída que serão utilizados na comunicação com o dispositivo. Já o método *closeAccessory* é responsável por fechar todas as conexões abertas restantes para o dispositivo externo.

```
1     private UsbManager myUsbManager;
2     private UsbAccessory myUsbAccessory;
3     private ParcelFileDescriptor myAdkParcelFileDescriptor;
4     private FileInputStream myAdkInputStream;
5     private FileOutputStream myAdkOutputStream;
6
7     private void OpenUsbAccessory(UsbAccessory acc){
8         myAdkParcelFileDescriptor = myUsbManager.openAccessory(
              acc);
9         if(myAdkParcelFileDescriptor != null){
10
11            myUsbAccessory = acc;
12            FileDescriptor fileDescriptor =
                  myAdkParcelFileDescriptor.getFileDescriptor();
13            myAdkInputStream = new FileInputStream(fileDescriptor
                  );
14            myAdkOutputStream = new FileOutputStream(
                  fileDescriptor);
15
16            Thread thread = new Thread(runnableReadAdk);
17            thread.start();
18        }
19    }
20
21    private void closeUsbAccessory(){
22
23        if(myAdkParcelFileDescriptor != null){
24            try {
25                myAdkParcelFileDescriptor.close();
```

```
26              } catch (IOException e) {
27                  e.printStackTrace();
28              }
29          }
30
31          myAdkParcelFileDescriptor = null;
32          myUsbAccessory = null;
33      }
```

Através da comunicação USB o aplicativo Android recebe uma string (protocolo Google adaptado, figura (10)) que contém os valores do parâmetro lido pela placa ADK.

3.2.4 Implementações com o Arduino ADK

3.2.4.1 Sensor DS18B20

Fisicamente, a leitura desse sensor pelo Arduino ADK foi viabilizada conectando-se o terminal de referência ao pino GND, conectando o terminal de alimentação ao pino 5V, e conectando o terminal de dados ao pino 2. Além disso, conectou-se um resistor de pull-up de 4,7 *Kohm* entre os terminais de alimentação e de dados, conforme a figura (31).

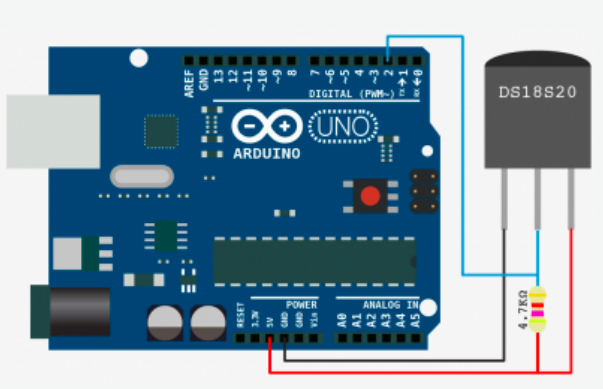

Figura 31: Exemplo de conexões entre o sensor DB18S20 e um Arduino UNO (BILDR, 2011).

A figura (31) exemplifica as conexões necessárias do sensor DS18B20 com um Arduino UNO. As mesmas conexões foram utilizadas neste trabalho com o Arduino ADK.

Via software, a leitura do sensor DS18B20 foi implementada com o auxílio da biblioteca OneWire, que permite, através de um pino digital do Arduino a leitura, modo serial, de uma palavra binária.

No código a seguir mostra-se como foi feita a inclusão da biblioteca *OneWire.h* e a declaração das variáveis necessárias para capturar os dados do sensor.

```
/** Include para sensor de Temperatura */
#include <OneWire.h>

/** DS18S20 Signal pin on digital 2 */
int DS18S20_Pin = 2;
/** Temperature chip i/o */
OneWire ds(DS18S20_Pin);
```

Foi utilizada a função *getTemp()* que foi importada de (BILDR, 2011), para capturar os dados do sensor.

```
float getTemp(){
 byte data[12];
 byte addr[8];

 if ( !ds.search(addr)) {
   ds.reset_search();
   return -1000;
 }

 if ( OneWire::crc8( addr, 7) != addr[7]) {
   Serial.println("CRC is not valid!");
   return -1000;
 }

 if ( addr[0] != 0x10 && addr[0] != 0x28) {
   Serial.print("Device is not recognized");
   return -1000;
 }

 ds.reset();
 ds.select(addr);
 ds.write(0x44,1);

 byte present = ds.reset();
 ds.select(addr);
 ds.write(0xBE);

 for (int i = 0; i < 9; i++) {
  data[i] = ds.read();
 }

```

```
32  ds.reset_search();
33  byte MSB = data[1];
34  byte LSB = data[0];
35
36  float tempRead = ((MSB << 8) | LSB);
37  float TemperatureSum = tempRead / 16;
38
39  return TemperatureSum;
40 }
```

3.2.4.2 Comunicação com Android e com o driver de motor

Utilizando o protocolo para strings, criou-se uma string para comunicação com o dispositivo Android.

O Arduino ADK precisa constantemente pegar os dados provenientes do sensor de temperatura DS18B20, envia-los ao dispositivo Android, receber dados provenientes do dispositivo Android e acionar o driver de motor (Ponte H). A partir disso foi necessário implementar no Arduino ADK *Threads* diferentes. Thread é uma forma de um processo dividir a si mesmo em duas ou mais tarefas que podem ser executadas de forma concorrente.

Em uma *Thread* o Arduino inicializa (*"start"*) a outra *Thread*, em seguida captura os valores do sensor e envia para o dispositivo Android, conforme o código a seguir:

```
1  /** Verifica se o Celular esta conectado */
2    if (acc.isConnected()){
3          /** start na outra thread */
4      if(readAdkControle.shouldRun())
5          readAdkControle.run();
6
7      /** Pega valor do sensor de temperatura 8/
8      float temp = getTemp();
9
10     char msg[8];
11
12     /** transforma float para char* */
13     dtostrf(temp,8,4,msg);
14
15     /** envia para Celular */
16     acc.write(msg, 8);
17   }
```

E na outra *Thread* o Arduino ADK fica aguardando comandos provenientes do dispositivo Android e quando recebe algum dado verifica e aciona ou não os motores, conforme o código a seguir:

```
void niceCallback(){
    int len = acc.read(read_cel, sizeof(read_cel), 1);
    if (len > 0) {
      /** transformando PWM recebido em int */
      char pwm[] = {read_cel[1],read_cel[2],read_cel[3]};
      velocidade = atoi(pwm);
      delay(200);
         if(read_cel[0] == 'a'){
            velocidade_motor(velocidade);
            digitalWrite(IN1,HIGH);
            digitalWrite(IN2,LOW);
            digitalWrite(IN3,HIGH);
            digitalWrite(IN4,LOW);
         }
      /** Liga os dois motores para "curva1" */
       else if(read_cel[0] == 'b'){
           velocidade_motor(velocidade);
           digitalWrite(IN1,HIGH);
           digitalWrite(IN2,LOW);
           digitalWrite(IN3,LOW);
           digitalWrite(IN4,HIGH);
        }
       /** Liga os dois motores para "Curva2" */
       else if(read_cel[0] == 'c'){
          velocidade_motor(velocidade);
          digitalWrite(IN1,LOW);
          digitalWrite(IN2,HIGH);
          digitalWrite(IN3,HIGH);
          digitalWrite(IN4,LOW);
        }
       /** Liga os dois motores para tras */
       else if(read_cel[0] == 'd'){
           velocidade_motor(velocidade);
           digitalWrite(IN1,LOW);
           digitalWrite(IN2,HIGH);
           digitalWrite(IN3,LOW);
           digitalWrite(IN4,HIGH);
        }
```

```
39          /** Desliga os dois motores */
40          else{
41              digitalWrite(IN1,LOW);
42              digitalWrite(IN2,LOW);
43              digitalWrite(IN3,LOW);
44              digitalWrite(IN4,LOW);
45          }
46      }
47  }
```

A comunicação entre o Arduino ADK e o driver dos motores (Ponte H) é feita através de pinos digitais. A tabela 12 mostra as conexões realizadas. O controle de velocidade dos motores através do driver de motor é implementado utilizando-se um sinal PWM, isto é, um sinal digital com largura de pulso modulada no terminal EN. Dessa forma a velocidade do motor será proporcional à largura de pulso do sinal.

Tabela 12: Funcionamento do driver de motor.

Arduino ADK	**Driver dos motores**
Pino 8	IN1
Pino 9	IN2
Pino 10	IN3
Pino 11	IN4
Pino 12	ENA
Pino 13	ENB

3.2.5 Implementação do Filtro

3.2.5.1 Média móvel

O filtro média móvel foi implementado primeiramente utilizando o *Matlab*, em seguida foi implementado dentro do dispositivo Android. Para tal, criou-se uma função chamada media_movel() que recebe dois parâmetros: N e medidas. O parâmetro N define o número de amostras que serão utilizadas para o calculo da cada média. Já o parâmetro medidas é um vetor que contém as amostras que serão filtradas. O código a seguir mostra a implementação desse filtro no *Matlab*.

```
1  tempo_f=length(medidas);
2
3  N = 30; % Parametro da media movel
4
5  for i=1:tempo_f-N
6      t(i) = i;
7      y(i) = medidas(i);
```

```
8       soma = 0.0;
9       for j=i:(i+N-1)
10          soma = soma + medidas(j);
11      end
12      medias(i) = soma/N;
13  end;
```

3.2.6 Implementação da plataforma móvel

A etapa de prototipação é a responsável por reunir os componentes do projeto em uma estrutura eletro-mecânica, resultando no protótipo da plataforma móvel. Nessa etapa foram utilizados os seguintes materiais: Arduino ADK, barco elétrico, driver de motor, baterias e dispositivo Android.

O barco elétrico é o principal componente dessa etapa, pois todos os componentes ficam dispostos dentro do barco. Foi escolhido um modelo de barco com dois motores elétricos, com estrutura confeccionada em plástico, controlado por rádio-frequência, com as seguintes dimensões: 45,7 cm de comprimento e 13,5 cm de largura. A justificativa para escolha desse modelo foi o baixo custo do mesmo. O barco (e suas respectivas dimensões) está ilustrado na figura (32).

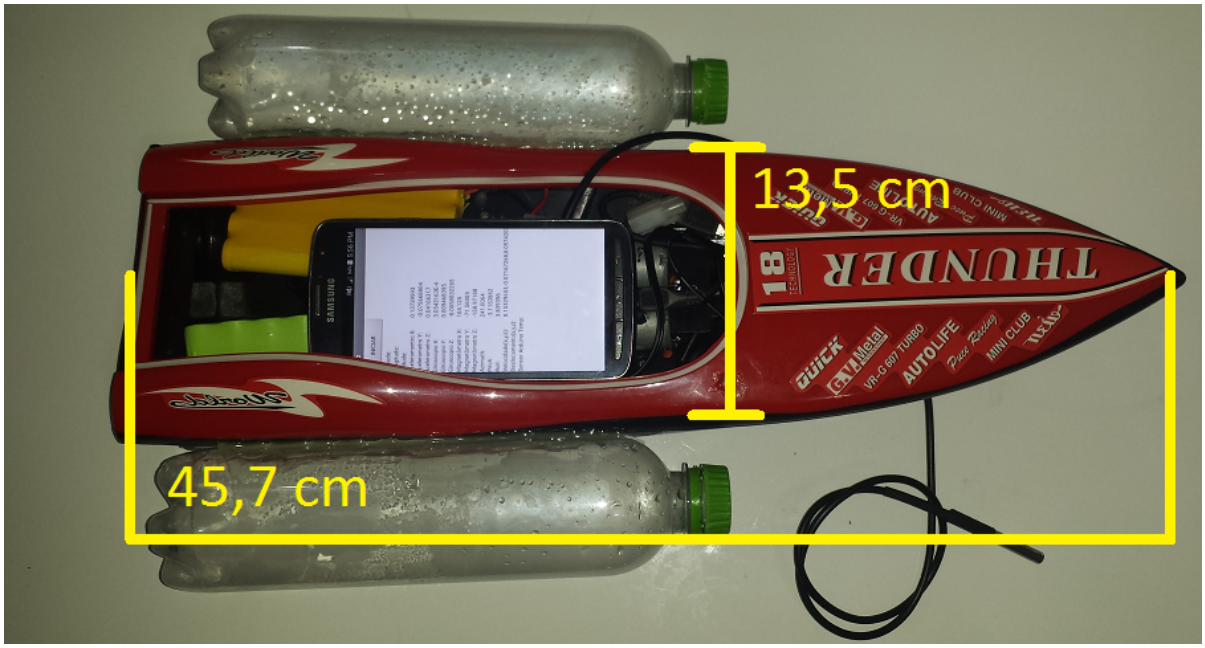

Figura 32: Dimensões do barco elétrico.

Foram utilizadas duas baterias na prototipação da plataforma móvel: uma bateria dedicada à alimentação dos motores e uma bateria dedicada à alimentação do Arduino ADK. A bateria dedicada aos motores é uma bateria de Ni-MH com tensão contínua de

9,6 V com carga de 600 mAh. A bateria dedicada ao Arduino é um modelo de chumbo com tensão contínua de 12 V com carga de 1,3 Ah. O critério de escolha da bateria foi ponderada pelo fator do menor custo desde que atendesse as demandas (especificadas nos capítulos anteriores).

A disposição dos componentes no barco está ilustrada na figura (33).

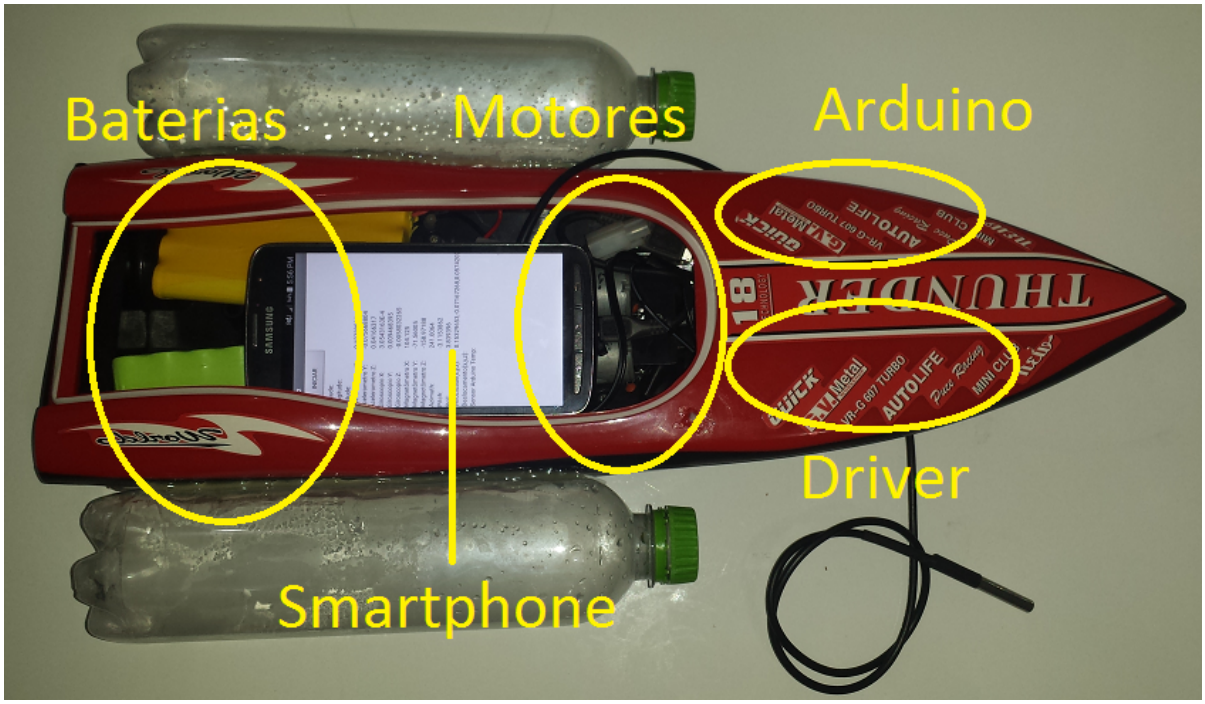

Figura 33: Disposição dos componentes no barco elétrico.

A disposição dos componentes foi realizada seguindo dois critérios: o balanceamento do peso na plataforma e proximidade de componentes. Dessa forma concluiu-se que o Arduino e driver deveriam estar próximos, visto que há diversas conexões entre os dois. Além disso o driver deveria estar próximo dos motores, pois todas as conexões dos motores são direcionados ao driver. Por essas razões, o Arduino e o driver foram colocados na ponta do barco, lado a lado, próximos aos motores. Visto que as baterias são os componentes mais pesados, foram colocados simetricamente no fundo do barco. Os motores são fixados (figura (34)) na estrutura, e por isso não havia liberdade na disposição do motores. O aparelho *smartphone* foi colocado sobre os motores no centro do barco, próximo ao Arduino ADK, pois a única conexão do *smartphone* é com o Arduino.

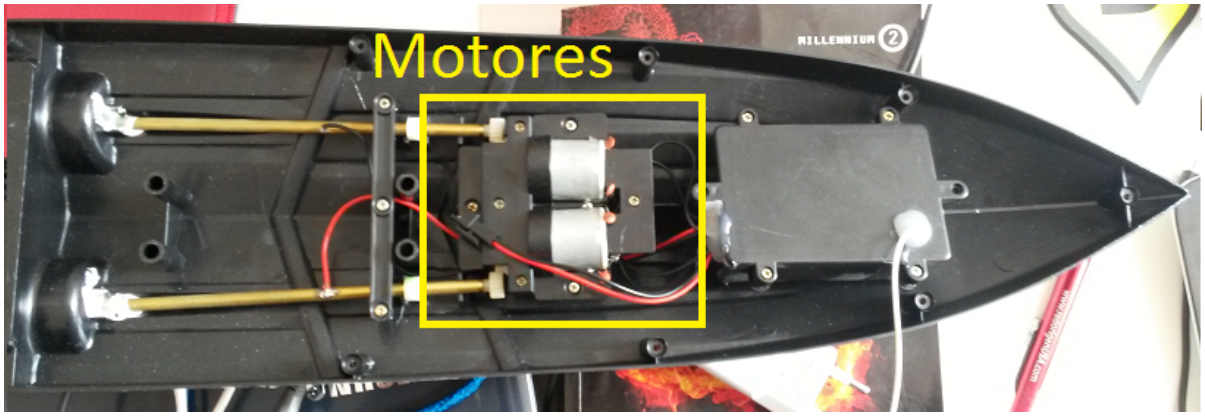

Figura 34: Disposição dos motores na estrutura do barco elétrico.

3.3 Dificuldades encontradas na construção da plataforma móvel

Nesta seção são descritas as dificuldades encontradas na construção do protótipo da plataforma móvel. A construção do protótipo envolveu diversos problemas de engenharia que requisitaram soluções ágeis, de modo a não prejudicar a execução e conclusão do trabalho.

O primeiro problema detectado na construção do protótipo foi a verificação de falhas na vedação da estrutura do protótipo. Esse problema foi observado no primeiro teste do propótipo. Nesse teste o protótipo foi colocado em uma piscina com o objetivo de verificar a vedação do mesmo. Ao colocar o protótipo na piscina, observou-se que após um minuto havia água dentro da estrutura. Para solucionar esse problema foram aplicadas filetes de massa plástica nas partes removíveis com uma camada de resina, de modo a vedar a estrutura.

Durante os testes em campo verificou-se um problema na estabilidade do protótipo. Devido a quantidade e peso dos componentes (das baterias principalmente) o protótipo ficou pesado e mostrou-se bastante instável na água. Para solucionar esse problema foi instalado duas garrafas no protótipo (um em cada lateral), com o objetivo de ajustar a estabilidade.

No projeto do algoritmo de navegação da plataforma, o primeiro problema enfrentado foi a determinação e ajuste do sentido do protótipo da plataforma móvel. Verificou-se que era necessário corrigir a direção da plataforma, de modo que a ponta do propótipo aponte para coordenada de destino. Como solução vislumbrou-se utilizar a bússola digital disponível no dispositivo Android para realizar tal ajuste. Entretanto verificou-se que a bússola digital apresentou-se sensível a oscilações. Uma segunda solução seria a aquisição de um hardware de bússola digital que atendesse aos requisitos do projeto. Porém essa solução demandaria recursos financeiros que não estavam disponíveis. A solução efetivamente implementada foi a utilização do GPS para realizar o ajuste de direção.

4 Resultados

Neste capitulo são apresentados os resultados obtidos ao longo do desenvolvimento do projeto assim como os resultados obtidos com a implementação da solução proposta.

4.1 Resultados Obtidos

4.1.1 Testes realizados utilizando a plataforma Android

Foram desenvolvidos cinco aplicativos no intuito de realizar testes no âmbito de comunicação com a plataforma ADK, com o servidor e leitura de sensores do aparelho. A partir desses cinco aplicativos foi desenvolvido o aplicativo final.

4.1.1.1 Aplicativo teste01_gps

Essa aplicação foi desenvolvida com o objetivo de testar a aquisição de coordenadas de GPS utilizando o hardware do aparelho. A interface desse aplicativo encontra-se na figura (35).

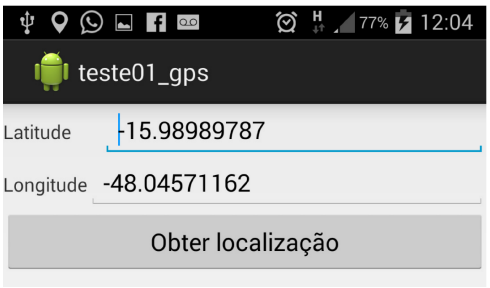

Figura 35: Aplicativo teste01_gps.

O usuário deve iniciar o monitoramento de GPS através do Button “Obter localização”. Neste momento o aplicativo faz a requisição ao hardware do aparelho e retorna as coordenadas nas TextView “Latitude” e “Longitude”.

4.1.1.2 Aplicativo gps1

Esse aplicativo foi desenvolvido com o objetivo de testar a aquisição de coordenadas de GPS utilizando dois diferentes provedores de GPS e testar a API do Google para utilizar o GoogleMaps. A interface do aplicativo encontra-se na figura (36).

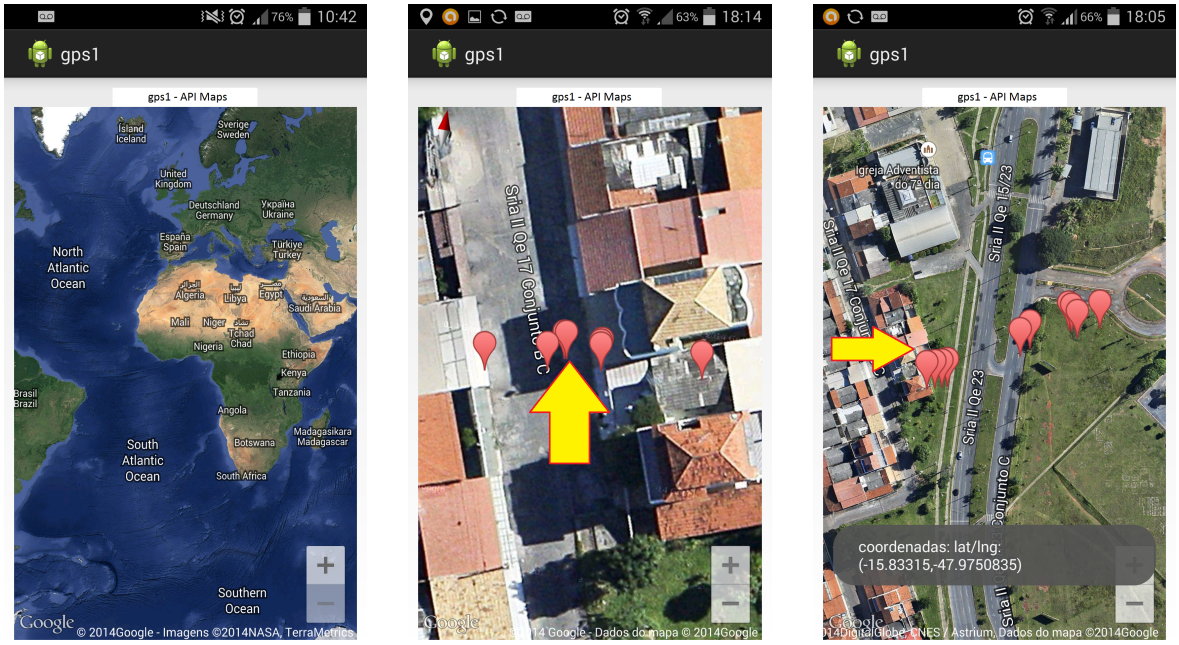

Figura 36: Aplicativo gps1.

Na figura (36) à esquerda mostra a tela incial do aplicativo, a imagem do meio mostra a tela do aplicativo utilizando o provedor GPS_PROVIDER que utiliza o hardware de GPS contido no aparelho, a seta amarela indica posição real. A imagem a direita mostra a tela do aplicativo utilizando o provedor NETWORK_PROVIDER que utiliza a triangulação de antenas, a seta amarela indica posição real.

Analisando a figura (36) conclui-se que o GPS_PROVIDER ofereceu coordenadas com mais acurácia do que o NETWORK_PROVIDER.

4.1.1.3 Aplicativo Listar_sensores

Essa aplicação foi implementada com o objetivo de listar todos os sensores disponíveis do aparelho. A interface do aplicativo encontra-se na figura (37).

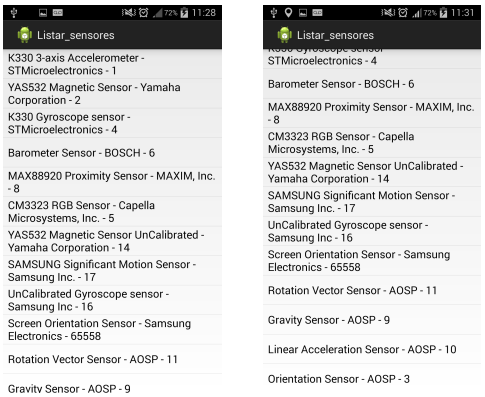

Figura 37: Aplicativo Lista_sensores.

4.1.1.4 Aplicativo Sensores

Essa aplicação foi implementada afim de testar a aquisição de dados de um sensor. Neste exemplo foi utilizado o acelerômetro, no entanto para utilizar outro sensor basta trocar o tipo do mesmo.

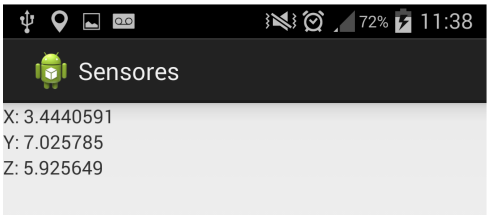

Figura 38: Aplicativo Sensores.

4.1.1.5 Aplicativo TCC

O aplicativo TCC foi implementado com intuito de testar a aquisição dos sensores que serão utilizados para navegação (GPS, acelerômetro, giroscópio e bússola) alem de testar a comunicação com o servidor. A interface do aplicativo encontra-se na imagem a esquerda da figura (39).

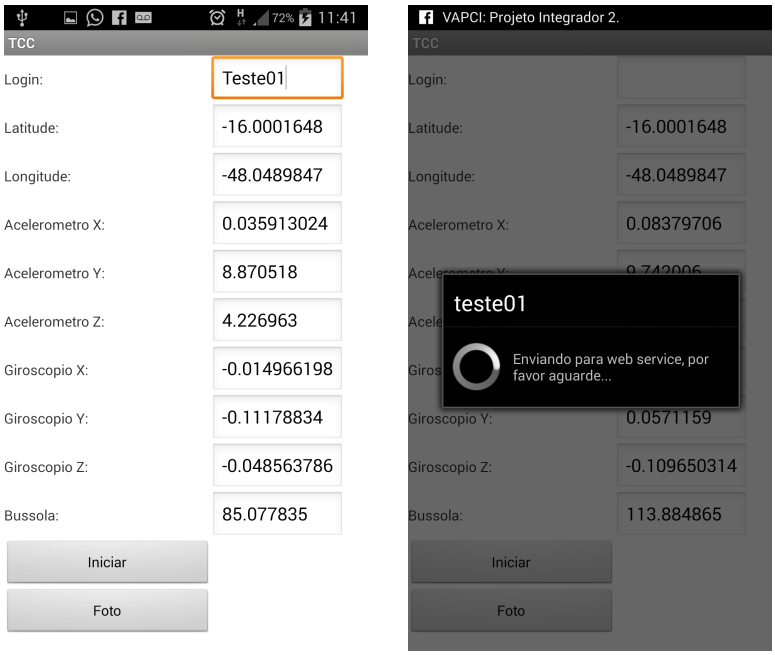

Figura 39: Aplicativo TCC.

Ao iniciar o aplicativo ele realiza a leitura repetidamente dos sensores: GPS, acelerômetro, giroscópio e bussola. Quando o Button "Iniciar" é acionado pelo usuário o aplicativo inicia uma conexão com o servidor e envia um pacote de informações para o mesmo, como mostra a imagem a direita da figura (39). Esse pacote de informações in-

cluem as últimas coordenadas (latitude, longitude) retornadas pelo GPS, data e hora e um identificador que deve ser inserido pelo usuário na TextView "Login".

4.1.2 Leitura dos sensores embarcados com aplicação do filtro média móvel

Os testes realizados a seguir foram feitos com o seguinte procedimento: o aparelho *smartphone* foi colocado sobre uma mesa plana e sem vibrações. Em seguida, utilizando o aplicavo TCC, o sistema Android fez a leitura dos sensores embarcados e enviou os dados para o servidor, via rede 3G, durante 1 minuto. Ao final desse período, os dados foram analisados estatisticamente e submetidos a filtragem com o objetivo de atenuar ruídos. A primeira análise ocorreu com o acelerômetro. Foram adquiridas 103 amostras de cada eixo do acelerômetro. No eixo x as amostras variaram em torno do valor de -0,01 m/s^2 dentro de uma faixa com picos de -0,06 m/s^2 à 0,03 m/s^2 com desvio padrão de 0,0171, conforme figura (40).

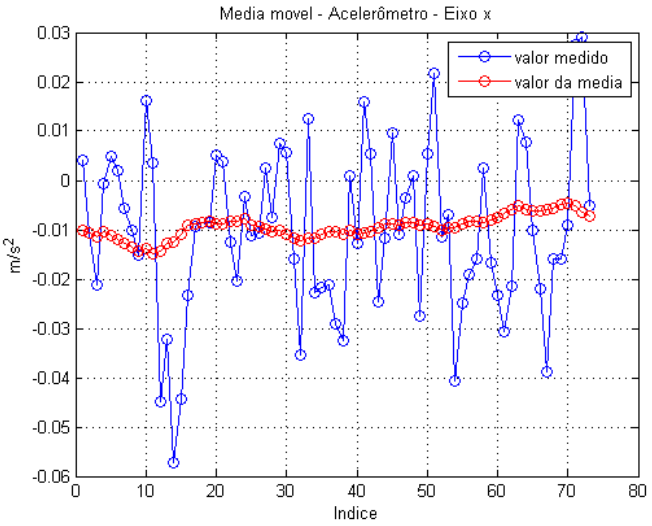

Figura 40: Amostra e filtragem do eixo x do acelerômetro.

Ao aplicar um filtro média móvel de 30 amostras, o desvio padrão resultante caiu para 0,0023.

No eixo y as amostras variaram em torno do 0 com picos entre -0,06 m/s^2 a 0,06 m/s^2 com desvio padrão de 0,0207, conforme figura (41).

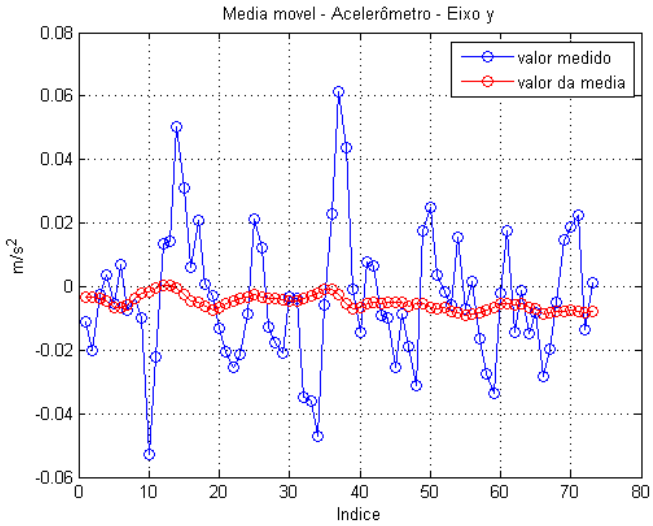

Figura 41: Amostra e filtragem do eixo y do acelerômetro.

Ao aplicar um filtro média móvel de 30 amostras, o desvio padrão resultante caiu para 0,0023.

No eixo z as amostras variaram em torno do valor 0,01 m/s^2 com picos entre -0,02 m/s^2 a 0,09 m/s^2 com desvio padrão 0,0171, conforme figura (42).

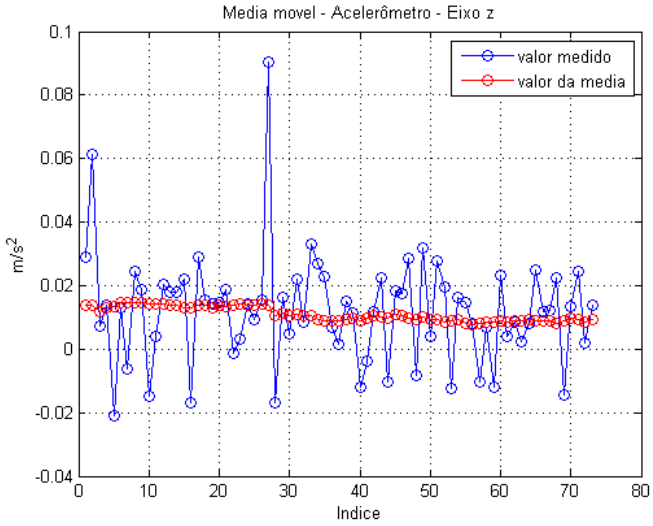

Figura 42: Amostra e filtragem do eixo z do acelerômetro.

Ao aplicar um filtro média móvel de 30 amostras, o desvio padrão resultante caiu para 0,0023.

A segunda análise ocorreu com o giroscópio. Foram adquiridas 103 amostras de cada eixo do giroscópio. No eixo x as amostras variaram em torno do valor $-1E-3rad/s$

com picos entre $-8E-3rad/s$ e $7E-3rad/s$ com desvio padrão de 0,0028 conforme figura (43).

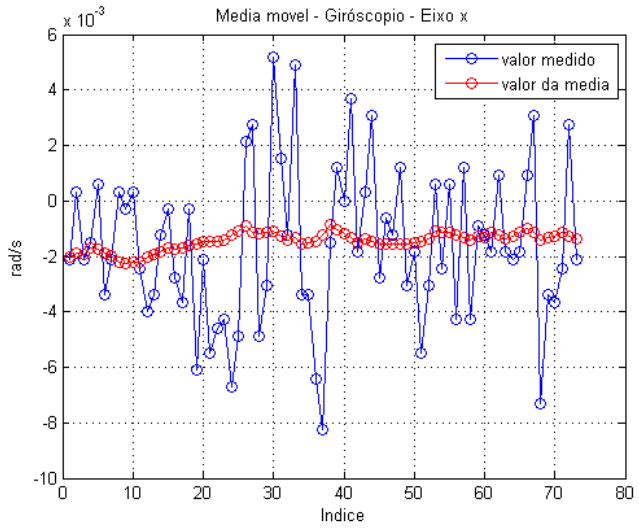

Figura 43: Amostra e filtragem do eixo x do giroscópio.

Ao aplicar um filtro média móvel de 30 amostras, o desvio padrão resultante caiu para $3,25E-4$.

No eixo y as medições variaram em torno do valor 0,029 rad/s com picos entre 0,021 rad/s e 0,035 rad/s com desvio padrão de 0,0026, conforme figura (44).

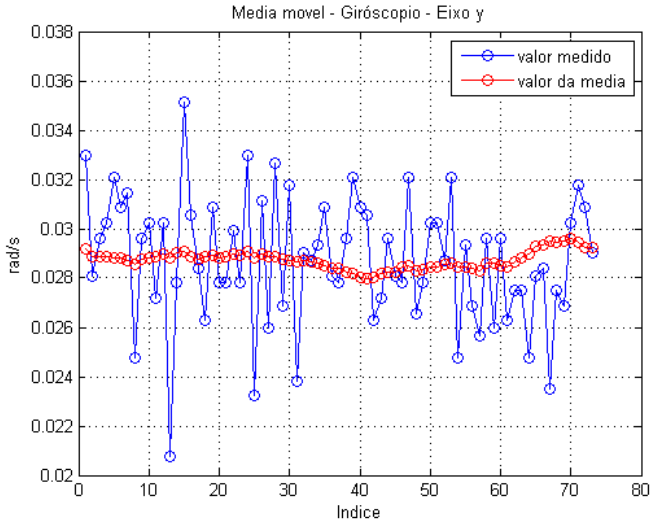

Figura 44: Amostra e filtragem do eixo y do giroscópio.

Ao aplicar um filtro média móvel de 30 amostras, o desvio padrão resultante caiu para $3,68E-4$.

No eixo z as amostras variaram em torno do valor $-2xE-3rad/s$, com picos entre $-7E-3rad/s$ a $3E-3rad/s$ com desvio padrão de 0,0017, conforme figura (45).

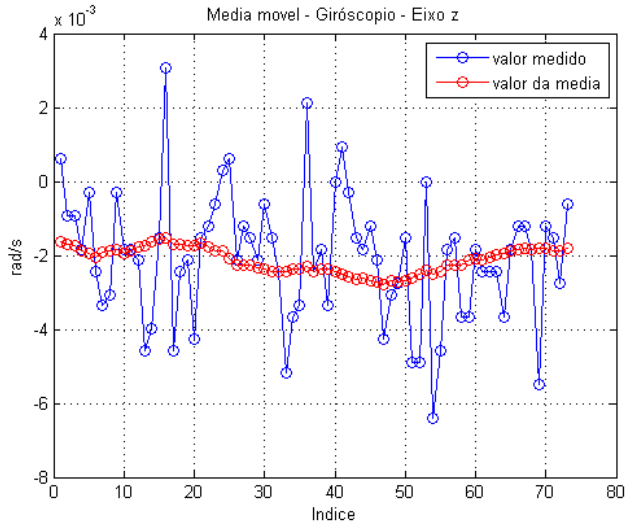

Figura 45: Amostra e filtragem do eixo z do giroscópio.

Ao aplicar um filtro média móvel de 30 amostras, o desvio padrão resultante caiu para $3,50E-4$.

A terceira análise ocorreu com o magnetômetro. Foram adquiridas 103 amostras da cada eixo do magnetômetro. No eixo x as amostras variaram em torno do valor -2,8 mT com picos entre -3,3 mT e -2,2 mT com desvio padrão de 0,33 conforme figura (46).

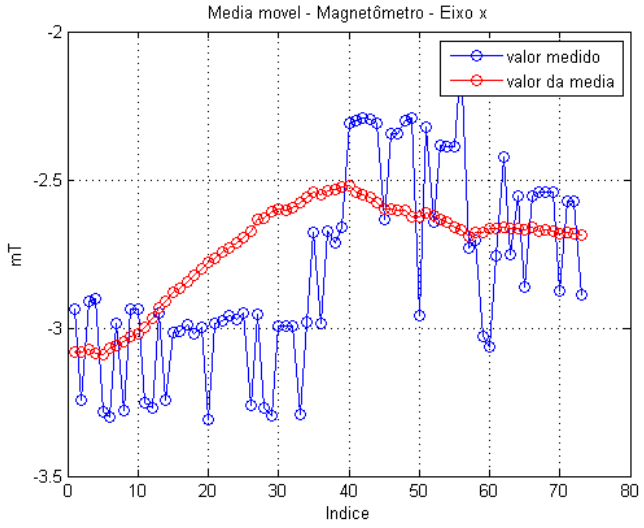

Figura 46: Amostra e filtragem do eixo x do magnetômetro.

Ao aplicar um filtro média móvel de 30 amostras, o desvio padrão resultante caiu para 0,17.

No eixo y as amostras variaram em torno do valor -25,5 mT com picos de -26,1 mT a -24,8 mT com desvio padrão de 0,23 conforme figura (47).

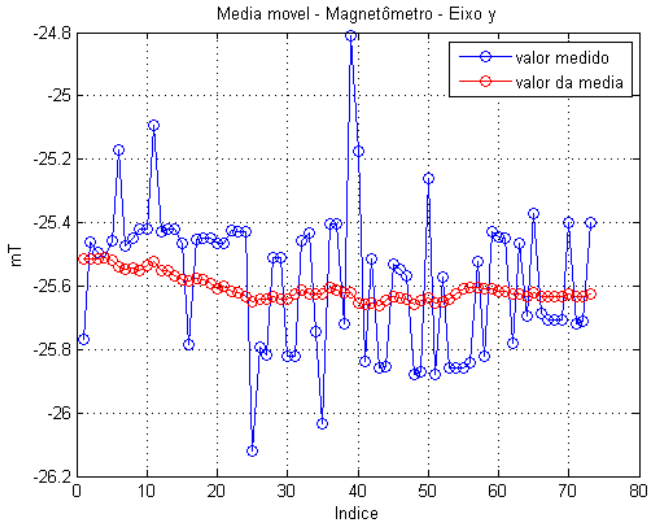

Figura 47: Amostra e filtragem do eixo y do magnetômetro.

Ao aplicar um filtro média móvel de 30 amostras, o desvio padrão resultante caiu para 0,04.

No eixo z as amostras variaram em torno do valor -2,35 mT com picos de -3,1 mT a -1,6 mT com desvio padrão de 0,39 conforme figura (48).

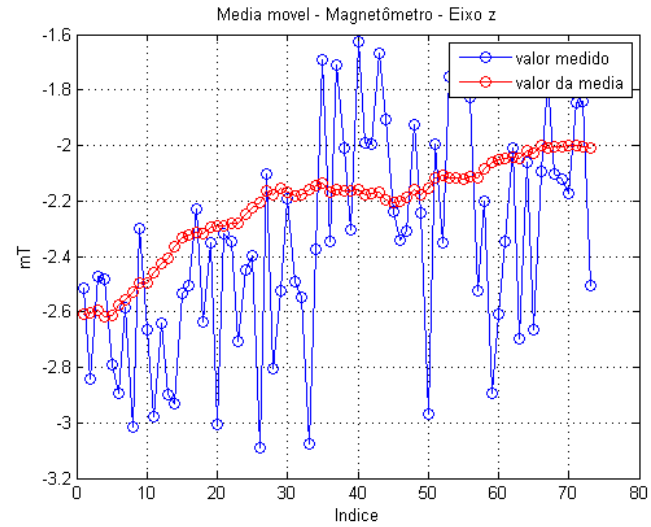

Figura 48: Amostra e filtragem do eixo z do magnetômetro.

Ao aplicar um filtro de média móvel de 30 amostras, o desvio padrão resultante caiu para 0,18.

Em todos os casos, observou-se que ao aplicar o filtro média móvel nas amostras, houve uma queda significativa no desvio padrão, além da suavização da curva ao longo do tempo. Desse modo, verificou-se a utilidade do filtro na atenuação dos ruídos e efeitos espúrios durante a medição.

4.1.3 Aplicativo final para prova de conceito

A partir dos testes realizados com a plataforma Android, foi implementado o aplicativo que fica em constante execução no dispositivo Android, dentro da platatorma. A interface do aplicativo encontra-se na figura (49).

Figura 49: Aplicativo final proposto.

Quando o aplicativo é iniciado, ele inicia automaticamente a aquisição dos sensores embarcados no dispositivo, em paralelo inicia a comunicação com o Arduino ADK. Em seguida ele mostra para o usuário através da interface os dados dos sensores em Labels. Tais valores antes de serem apresentados ao usuário são filtrados pelo aplicativo utilizando o filtro media móvel. Em seguida o aplicativo precisa ser inicializado, pressionando o Button "Iniciar", ao acionar esse Button e iniciada a comunicação com o webservice, inicializando assim o fluxo de dados para controle da plataforma e trafego de dados uteis ao usuário.

4.1.4 Interface Gráfica do Usuário

No projeto desenvolvido a comunicação com o usuário é realizada através de um programa computacional que oferece uma interface gráfica ao usuário. Basicamente esse

programa deve receber um comando do usuário e enviar um pacote de dados ao webservice. O layout pode ser visualizado na figura (50).

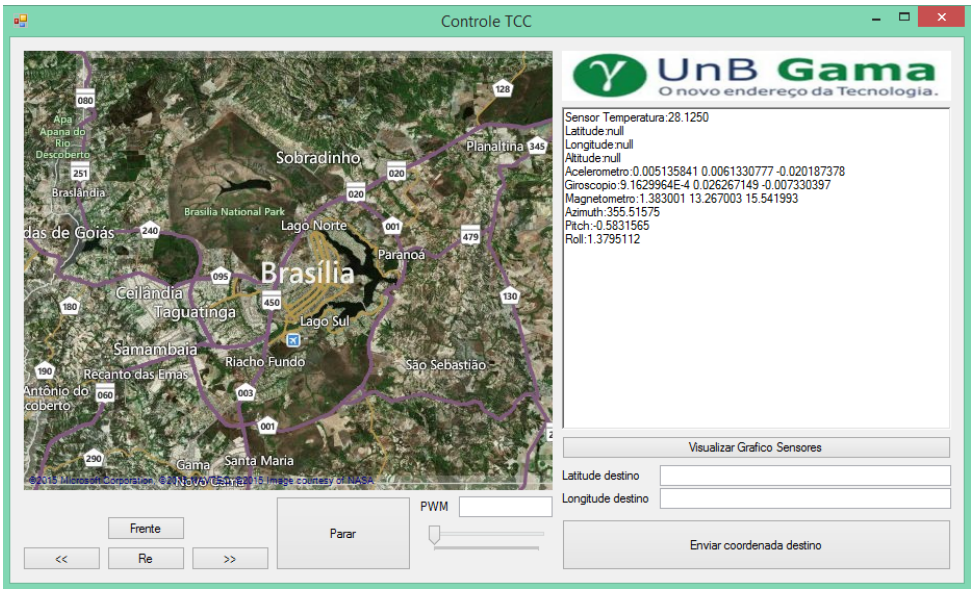

Figura 50: Layout do programa controlador.

Ao iniciar o programa o usuário tem somente a visualização do mapa. Automaticamente o programa vai buscar no servidor a localização mais recente disponível enviada pela plataforma e vai plotar essa posição adicionando um marcador verde ao mapa, conforme figura (51).

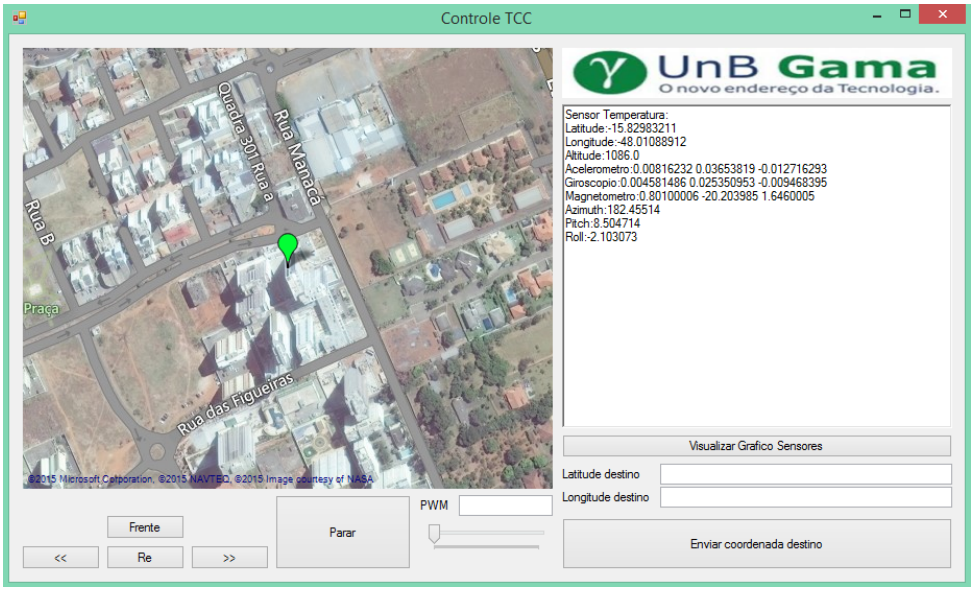

Figura 51: Layout do programa controlador com posição atual da plataforma.

Para movimentar a plataforma móvel de forma direta da forma que o usuário desejar, existem cinco Buttons e uma TrackBar para controle de velocidade através do

PWM, cada Button envia um determinado comando para o webservice, e a plataforma móvel verifica qual o ultimo comando enviado para tomar uma ação. Os cinco Buttons e a TrackBar são mostrados na figura (52).

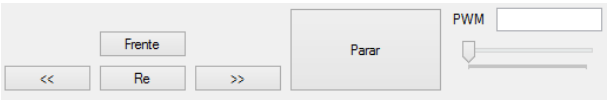

Figura 52: Buttons de controle direto da plataforma.

O programa atualiza automaticamente, a cada 1 segundo, os valores dos sensores tanto embarcados no dispositivo Android como o sensor de temperatura. Essa atualização é feita via webservice em uma Thread implementada apenas para atualização de valores. A figura (53) mostra o TextBox com os valores mais recentes no webservice.

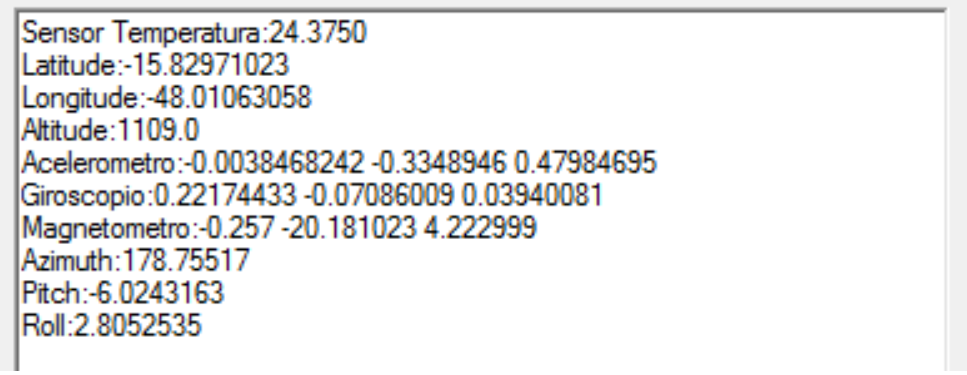

Figura 53: Layout do programa retornando na TextBox os dados dos sensores.

Para melhor visualização dos dados dos sensores foi criado um layout com gráficos dos sensores: acelerômetro, giroscópio, magnetômetro e temperatura. Para acessar esse layout com os gráficos basta o usuário pressionar o Button "Visualizar Grafico Sensores". A figura (54) mostra esse layout.

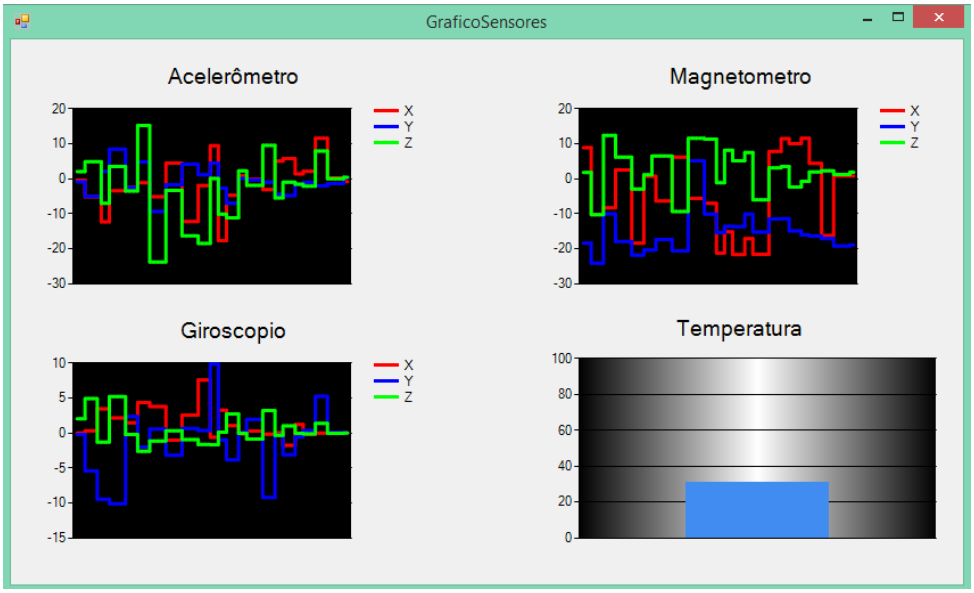

Figura 54: Layout implementado para mostrar graficamente os dados dos sensores.

Para realizar a navegação o usuário deve clicar no mapa para selecionar a posição de destino desejada. Automaticamente o programa irá plotar um marcador vermelho, colocar nas TextBox Latitude final e Longitude final as coordenadas onde o usuário clicou e em seguida traça a rota ideal (uma reta) entre o ponto atual da plataforma e o ponto de destino desejado. Caso o usuário queira alterar o destino antes de pressionar o Button "Enviar coordenada destino"é necessário apenas clicar em outro local e o programa automáticamente irá apagar a rota antiga e substituí-la por uma nova rota. Ao pressionar o Button "Enviar coordenada destino"não poderá mais alterar a posição de destino e o programa enviará para o servidor o comando para que a plataforma inicie a navegação.

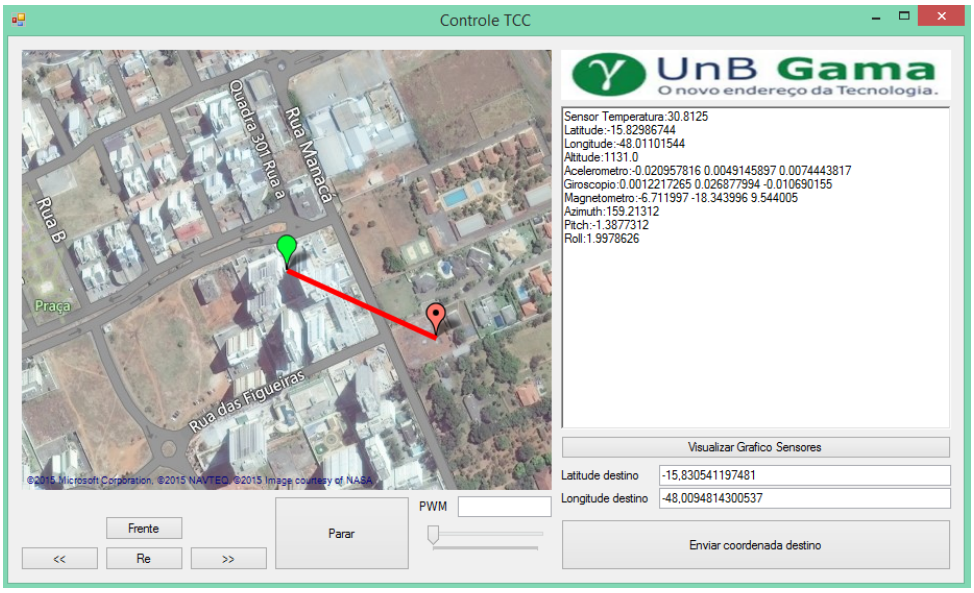

Figura 55: Layout do programa com posição de destino selecionada.

4.1.5 Protótipo da plataforma móvel

Nesta é apresentada o protótipo da plataforma móvel construído. O protótipo foi construído com base em uma estrutura de um barco elétrico. O Arduino ADK, o Driver de motor, os motores, bateria e o aparelho *smartphone* ficaram no interior da plataforma. O sensor de temperatura fica do lado de fora, conectado por um fio. Nas figuras (56) é apresentada a plataforma.

Figura 56: Protótipo da plataforma móvel.

Durante teste realizado em campo, verificou-se que plataforma estava pesada. Com o objetivo de aumentar a estabilidade da mesma foram instaladas duas garrafas na laterais

da plataforma. Nas figuras (57) é apresentada a atuação da plataforma durante testes em campo.

Figura 57: Protótipo em ação no lago Paranoá.

Verificou-se que a plataforma atendeu à proposta, razoavelmente bem, possibilitando a medição do parâmetro desejado nos pontos requeridos do Lago Paranoá.

4.1.6 Testes de campo

Os testes descritos a seguir tem o objetivo de testar a plataforma móvel, no que diz respeito ao fluxo de comunicação e navegação, em um ambiente real. Todos os testes foram realizados na região do Lago Paranoá que fica às margens da via L4, próximo a ponte do Bragueto.

O procedimento em cada teste foi realizado da seguinte forma: utilizando um caiaque a plataforma era colocada em um determinado local do lago; um operador fica na beira do Lago com um notebook com conexão 3G e envia uma coordenada de destino para a plataforma; ao receber a coordenada de destino a plataforma iniciava a navegação e se dirige ao destino e chegando lá realiza a medição da temperatura; ao final do teste a plataforma é recolhida, utilizando o caiaque.

No primeiro teste a plataforma foi colocada próxima à margem do Lago e seu destino também próximo à margem. Na figura (58) temos a trajetória da plataforma. As bandeiras em verde são coordenadas de GPS enviadas pela plataforma. A linha em vermelho seria a menor trajetória entre a coordenada inicial e o destino.

Figura 58: Teste de campo realizado no lago Paranoá.

Ao chegar no destino (considerando a margem de erro de 20 metros) a plataforma realiza a medição da temperatura, obtendo um valor de 23,75 graus Celsius. A plataforma percorreu o trajeto em, aproximadamente, 5 minutos.

No segundo teste, a coordenada de início é um pouco mais afastada da margem (em relação ao primeiro teste). Na figura (59) temos a trajetória da plataforma.

Figura 59: Teste de campo realizado no lago Paranoá.

Ao chegar no destino a plataforma realiza a medição da temperatura, obtendo um valor de 23,97 graus Celsius. A plataforma percorreu o trajeto em, aproximadamente, 20 minutos.

No terceiro teste, a coordenada de início foi colocada na segunda margem do lago. Na figura (60) é mostrada a trajetória da plataforma.

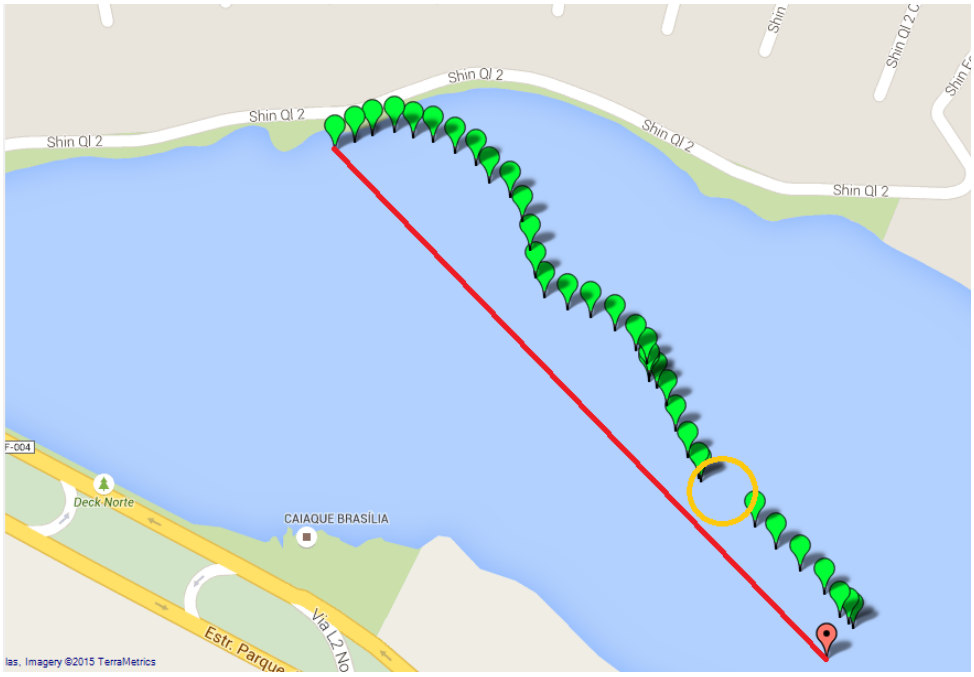

Figura 60: Teste de campo realizado no lago Paranoá.

Nesse teste observa-se que durante uma parte do trajeto (representado por um circulo amarelo) não há bandeira verde, o que significa que não houve registro de coordenadas de GPS no servidor nesse pedaço do trajeto. Há duas possibilidades plausíveis para isto: existe a possibilidade do hardware de GPS ter parado de funcionar durante esse trajeto e há a alternativa do aparelho *smartphone* ter perdido a conexão 3G durante essa parte do trajeto. Considerando que a navegação da plataforma depende do posicionamento do GPS e não foi prejudicada, e considerando ainda que durante a navegação a conexão 3G não é pré-requisito para que o percurso seja bem sucedido, a conclusão é que durante o trajeto o aparelho perde a conexão 3G durante algum momento. Ao chegar no destino, a plataforma realiza a medição da temperatura, obtendo um valor de 24,83 graus Celsius. A plataforma percorreu o trajeto em, aproximadamente, 30 minutos.

No quarto teste, a coordenada de início está próxima à margem do Lago novamente. Na figura (61) apresenta-se a trajetória da plataforma.

Figura 61: Teste de campo realizado no lago Paranoa.

Ao chegar no destino, a plataforma realiza a medição da temperatura, obtendo um valor de 23,93 graus Celsius. A plataforma percorreu o trajeto em, aproximadamente, 20 minutos.

Em todos os casos, observou-se que a plataforma dirigiu-se à coordenada de destino, considerando a margem de erro de vinte metros. Além disso, verificou-se em todos os testes a medição da temperatura e recebimento dessa informação na interface gráfica do usuário.

5 Conclusões

Este trabalho consiste no desenvolvimento de uma plataforma móvel controlada remotamente que faça a medição da temperatura da água de determinada zona do Lago Paranoá.

Para a realização desse desenvolvimento foi proposto um veículo autônomo de superfície controlado por um Arduino ADK que comunica-se com o servidor através de uma conexão de rede 3G, viabilizado pela utilização de um aparelho smartphone Android. Como estudo de caso escolheu-se fazer a medição da temperatura da água, e limitou-se a zona de atuação da plataforma a uma área definida do Lago Paranoá.

A implementação do aparelho smartphone Android na plataforma, possibilitou a utilização da rede 3G como meio de comunicação. Além disso, o aparelho Android já possui uma série de sensores (giroscópio, acelerômetro, bússola e magnetômetro) e um serviço de localização por GPS que foi utilizado na navegação da plataforma e dispensou a aquisição de um hardware de GPS.

Como resultados obteve-se aplicativos Android que fazem a leitura dos sensores embarcados no smartphone, comunicam-se com o Arduino ADK e com o servidor, e implementa um filtro média móvel que atenua ruídos durante a leitura dos sensores. Obteve-se também, um aplicativo para desktop com inteface para usuário, onde o usuário pode enviar a coordenada de destino à plataforma e visualizar os valores das leituras de sensores. Além disso, obteve-se o protótipo da plataforma móvel com navegação orientada por posição que foi utilizada em testes de campo no Lago Paranoá.

A tarefa mais desafiadora deste trabalho foi a construção do protótipo, visto a falta de experiência e conhecimento técnico dos autores nessa área. Por isso houve a decisão de adquirir um modelo comercial de barco elétrico e desenvolver o protótipo sobre a estrutura desse barco. Mesmo assim a equipe enfrentou problemas que não estavam previstos, como: falhas na vedação da estrutura, de modo que nos primeiros testes houve entrada de água no interior da plataforma; excesso de peso na plataforma, do modo que ela se apresentava instável nos testes. A solução para o primeiro problema foi o reforço da vedação da estrutura com a utilização de massas e resinas. A solução para o segundo problema foi a instalação de garrafas nas laterais da plataforma, com o objetivo ajustar a estabilidade.

A implementação completa da comunicação pode ser considerada o principal objetivo desse trabalho e possibilitou o fluxo de informações entre a plataforma e o usuário. A informação da temperatura da água é medida pelo sensor, que é lido pelo Arduino, transmitido ao aparelho Android, enviado ao servidor, e por fim é requisitada pelo programa

controlador e mostrada ao usuário, após percorrer todo o caminho de dados. Do modo que a solução foi implementada, a plataforma móvel pode ser controlada por qualquer computador do mundo que possua acesso a internet, de maneira que a solução alinha-se perfeitamente aos novos conceitos de tecnologia da informação, tais como objetos interativos e IoT (Internet of Things).

Por fim, verifica-se o potencial para trabalhos futuros, como a construção de uma estrutura de plataforma mais robusta e a adição de sondas e sensores que permitam a aquisição de mais dados e parâmetros da água que está sendo analisada, além disso melhoramento do algoritmo de navegação acrescentando os sensores inerciais ao algoritmo, adicionando esses sensores inerciais pretende-se explorar filtros para estimação estocástica. A medição da temperatura nesse trabalho foi feita apenas com o objetivo de prova de conceito, mas existem sondas comerciais que fazem a medição de diversos parâmetros da água, permitindo uma análise mais rica e embasada tecnicamente. Espera-se que este trabalho possa servir de embasamento e inspiração para soluções ainda maiores que busquem atender problemas reais da humanidade, alinhando tecnologia de ponta e inovação.

Referências

ALMEIDA, D. R. de. *Prancha Robótica Autônoma*. 2012. Dissertação realizada no âmbito do Mestrado Integrado em Engenharia Eletrotécnica e de Computadores Major Automação. Faculdade de Engenharia da Universidade do Porto. Citado 6 vezes nas páginas 6, 8, 24, 25, 26 e 27.

ANA. *Agência Nacional de Águas. Panorama da Qualidade das Águas Superficiais do Brasil: 2012*. <http://ana.gov.br>: [s.n.], 2012. Citado na página 14.

ANA. *Agencia Nacional das Águas - Portal da qualidade das águas*. 2014. Acessado em: 18/10/2014. Citado 9 vezes nas páginas 6, 8, 13, 18, 19, 20, 21, 23 e 24.

ANDROID. *Android Open Source Project Accessories for Android*. <http://source.android.com/accessories/index.html>: [s.n.], 2014. Acessado em: 09/10/2014. Citado na página 37.

ANDROID. *Android Develop Reference*. <http://developer.android.com/develop/index.html>: [s.n.], 2015. Acessado em: 13/10/2014. Citado 9 vezes nas páginas 6, 8, 28, 29, 30, 31, 32, 33 e 34.

ARDUINO. <http://arduino.cc>: [s.n.], 2014. Acessado em: 05/10/2014. Citado 2 vezes nas páginas 6 e 38.

BARONI, L. Análise de algoritmos de navegação para um sistema gps diferencial em tempo real. 2004. Dissertação de Mestrado do Curso de Pós-Graduação em Engenharia e Tecnologia Espaciais/Mecânica Espacial e Controla. Instituto Nacional de Pesquisas Espaciais. Citado na página 26.

BILDR. *One Wire Digital Temperature. DS18B20 + Arduino*. <http://bildr.org/2011/07/ds18b20-arduino>: [s.n.], 2011. Acessado em: 15/10/2014. Citado 3 vezes nas páginas 6, 65 e 66.

BOEHMER, M. *Beginning Android ADK with Arduino: learn how to use the android open accessory development kit to create amazing gadgets with arduino*. [S.l.]: Apress, 2012. Citado 4 vezes nas páginas 6, 31, 38 e 39.

DALLAS. *DS18B20 Datasheet*. <http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Sensors/Temp/DS18B20.pdf>: [s.n.], 2015. Acessado em: 25/10/2014. Citado 2 vezes nas páginas 6 e 44.

HADDAD, R. *Web Services*. <https://msdn.microsoft.com/pt-br/library/cc564893.aspx>: [s.n.], 2014. Acessado em: 05/11/2014. Citado na página 37.

HUINFINITO. *Módulo Driver Motor com Dupla Ponte H - L298N*. <http://www.huinfinito.com.br/controladores/583-modulo-driver-motor-com-dupla-ponteh-st-l298n.html?search_query=modulo+driver+motor&results=4>: [s.n.], 2015. Acessado em: 08/09/2014. Citado 2 vezes nas páginas 6 e 41.

HUINFINITO. *Sensor de Temperatura DS18B20 (prova d'água)*. <http://www.huinfinito.com.br/sensores/931-sensor-de-temperatura-ds18b20-prova-d-agua.html?search_query=sensor+temperatura&results=23>: [s.n.], 2015. Acessado em: 08/09/2014. Citado 2 vezes nas páginas 6 e 44.

LECHETA, R. R. *Google Android: Aprenda a criar aplicações para dispositivos móveis com o Android SDK*. [S.l.]: Novatec, 2013. Citado na página 28.

MACORATTI, J. C. *ASP.NET 2008 - Criando Web Services II*. <http://www.macoratti.net/09/08/aspn_wbs.htm>: [s.n.], 2014. Acessado em: 06/11/2014. Citado na página 37.

MICROSOFT. *Introdução a plataforma .NET da Microsoft*. <https://msdn.microsoft.com/pt-br/aa702903.aspx>: [s.n.], 2015. Acessado em: 01/02/2015. Citado na página 35.

MICROSOFT. *Windows Forms*. <https://msdn.microsoft.com/pt-br/library/dd30h2yb(v=vs.110).aspx>: [s.n.], 2015. Acessado em: 20/02/2015. Citado 2 vezes nas páginas 35 e 36.

NOGUEIRA, F. *Modelagem e simulação - Modelos de Previsão*. <http://www.ufjf.br/epd042/>: [s.n.], 2014. Apontamentos - Pesquisa Operacional II. Universidade Federal de Juiz de Fora. Citado 2 vezes nas páginas 26 e 27.

SPARKFUN. *Pulse-width Modulation*. <https://learn.sparkfun.com/tutorials/pulse-width-modulation>: [s.n.], 2015. Acessado em: 09/02/2015. Citado 2 vezes nas páginas 6 e 40.

ST. *L298N Datasheet*. <http://pdf.datasheetcatalog.com/datasheet/stmicroelectronics/1773.pdf>: [s.n.], 2015. Acessado em: 02/11/2014. Citado 4 vezes nas páginas 6, 8, 42 e 43.

VALVANO, R. Y. J. *Embedded Systems - Shape the World*. <https://courses.edx.org/courses/UTAustinX/UT.6.02x/1T2015/info>: [s.n.], 2015. Acessado em: 02/04/2015. Citado 3 vezes nas páginas 6, 40 e 41.

VENESS, C. *Calculate distance, bearing and more between Latitude/Longitude points*. <http://www.movable-type.co.uk/scripts/latlong.html>: [s.n.], 2015. Acessado em: 11/03/2015. Citado na página 54.