

PROGRAMA DE INICIAÇÃO CIENTÍFICA JÚNIOR

INSTITUTO FEDERAL DO ESPÍRITO SANTO

EEEM PROFESSOR FERNANDO DUARTE RABELO

FUNDAÇÃO DE AMPARO À PESQUISA DO ESPÍRITO SANTO

**O uso de redes de sensores sem fio para monitoramento da
bacia hidrográfica do rio Jacaraípe para predição de
enchentes**

VITÓRIA

2017

Resumo

O Programa de Iniciação Científica Júnior é um programa regular do governo federal, representado pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), que requer a parceria com as Fundações de Amparo a Pesquisa de cada estado e prevê a participação de estudantes dos ensinos fundamental e médio da rede pública de ensino, coordenado por pesquisador com qualificação e experiência. É um programa que pretende atrair jovens estudantes para o mundo das Ciências, com a possibilidade de vivenciar a realidade do ambiente universitário, e despertar o interesse em dar continuidade aos estudos e futuramente o ingresso no ensino superior.

O Instituto Federal do Espírito Santo (Ifes) em parceria com a escola EEEM Professor Fernando Duarte Rabelo tiveram o projeto de Iniciação Científica Júnior intitulado “*O uso de redes de sensores para monitoramento da bacia hidrográfica do rio Jacaraípe para previsão de enchentes*” aprovado no edital Fapes nº 014/2014. Este projeto tem como objetivo o desenvolvimento de um dispositivo com capacidade de sensoriamento e comunicação sem fio, a fim de monitorar o nível da água nos componentes da bacia hidrográfica do rio Jacaraípe, presente no município de Serra-ES. Por meio deste é oportunizado a dez estudantes de ensino médio da escola EEEM Professor Fernando Duarte Rabelo, investigar as causas desse problema e projetar um protótipo capaz de minimizar os riscos, alertando as autoridades competentes.

Pesquisas estão sendo realizadas a fim de investigar o impacto ambiental, social e urbanístico da ocupação na região de Jacaraípe. Os alunos estão em contato com áreas de pesquisa da computação e engenharia para o desenvolvimento do protótipo e estão sendo motivados a construir uma visão social crítica a respeito de um problema real. As atividades de pesquisa ocorrem em um laboratório da EEEM Professor Fernando Duarte Rabelo, em um turno diferente das aulas regulares de cada aluno, sob a supervisão e orientação de dois monitores e um tutor.

Lista de figuras

Figura 1 - Bacia hidrográfica do rio Jacaraípe.	12
Figura 2 - Padrão construtivo da sub bacia do Rio Jacaraípe	14
Figura 3 - Renda de até um salário mínimo sub bacia do Rio Jacaraípe (IBGE 2000).	15
Figura 4 - Associação de pescadores de Jacaraípe. (Foto: Letícia Vieira Folha Vitória).	16
Figura 5 - Trecho bucólico e de restinga.	17
Figura 6 - Prática de surf nas praias de Jacaraípe.	17
Figura 7 - Desabamento de uma parte da BR ES-010	18
Figura 8 - Morte de peixes na lagoa Juara devido às grandes chuvas.	20
Figura 9 - Destino do esgoto na bacia do Rio Jacaraípe.	21
Figura 10 - Moradores de Jacaraípe são obrigados a sair de suas casas devido às enchentes.	22
Figura 11: Infraestrutura proposta	29
Figura 12: Vista aérea do rio Jacaraípe (Google Maps).	30
Figura 13: Sink node e Sensor node.	31
Figura 14 - Sensores condutivos	31
Figura 15 - Protótipo com sensor ultrassônico	32
Figura 16 - Sensor Ultrassônico	33
Figura 17 - Protótipo com bóia e sensores magnéticos	33
Figura 18 - Sensor Magnético	34
Figura 19 - Protótipo com sensores ICOS	34
Figura 20 - Sensores ICOS	35
Figura 21 - BeagleBone Black e seu cape de áudio e vídeo	41
Figura 22 - Sensor Ultrassônico	50
Figura 23 - Diagrama de funcionamento	51
Figura 24 - Ilustração do LCD	53
Figura 25 - LED's	53
Figura 26 - Buzzer	54
Figura 27 - Montagem do protótipo no simulador	54

Figura 28 - Esquema elétrico do protótipo	55
Figura 29 - Sensores ICOS e sua montagem elétrica	61
Figura 30 - Maquete construída para a simulação do funcionamento do dispositivo	65
Figura 31 - Maquete finalizada com dispositivo instalado	69
Figura 32 - Circuito soldado e instalado.	69
Figura 33 - Módulo NRF24L01+ e sua pinagem.	71
Figura 34 - Módulo GPS GY-NEO6MV2.	72
Figura 35 - Módulo para cartão SD.	73
Figura 36 - Módulo WiFi ESP8266 ESP-01.	74
Figura 37 - Circuito sensor node.	75
Figura 38 - Circuito sink node.	84
Figura 39 - Infraestrutura do web service.	91
Figura 40 - Banco de dados.	93
Figura 41 - Histórico de dados enviados pelo dispositivo e armazenados no banco de dados.	94
Figura 42 - Ícone do aplicativo de celular.	96
Figura 43 - Telas: Tela inicial; Tela de dados do dispositivo 1; Tela de estado geral da região.	97
Figura 44 - Tela de notificações e tela de configurações de notificação	97

Lista de tabelas

Tabela 1 - Bairros	19
Tabela 2 - Análise entre requisitos e trabalhos relacionados.	28
Tabela 3 - Especificações do arduino Uno	36
Tabela 4 - Especificações do arduino Mega	37
Tabela 5 - Especificações Raspberry PI B+	39
Tabela 6 - Especificações Beaglebone Black	41
Tabela 7 - Estruturas das tecnologias de prototipação	43
Tabela 8 - Sensores e características	44
Tabela 9 - Processamento dos equipamentos	44
Tabela 10 - Módulos de comunicação wireless	45
Tabela 11 - Requisito e tecnologias de prototipação	45
Tabela 12 - Pinos do Sensor Ultrassônico	51
Tabela 13 - Especificações LCD	52
Tabela 14 - Materiais para construção do protótipo com sensor ICOS	60
Tabela 15 - Materiais para construção do protótipo.	64
Tabela 16 - Materiais para construção da maquete e do dispositivo.	68

Sumário

1. Introdução	8
1.1. Motivação	8
1.2. Objetivos	9
1.3. Organização do trabalho	10
2. Referencial teórico	11
2.1. Bacia hidrográfica do rio Jacaraípe	11
2.2. Histórico	13
2.3. Importância socioeconômica	15
2.4. O problema das enchentes	18
2.5. Recorrência das enchentes	18
2.6. Impactos	19
2.6.1. Ambientais	19
2.6.2. Sociais	21
2.7. Mídia	22
2.7.1. Obras	22
2.7.2. Enchentes	24
3. Trabalhos relacionados	25
3.1. Monitoramento de rios	25
3.1.1. E-noé: Rede de sensores sem fio para monitorar rios urbanos	26
3.1.2. MONIT-RIO– Tecnologia da informação de comunicação para monitoramento de rios em casos de cheias	26
3.1.3. SIGMAOn–Sistema de Informação Geográfica para Monitoramento de Alagamentos On-line	27
3.1.4. Proposta metodológica para previsões de enchentes com uso de sistemas colaborativos	27
3.1.5. Discussão	27
4. Arquitetura proposta	29
4.1. Infraestrutura proposta	29
4.2. Dispositivos de monitoramento idealizados	30
4.2.1. Sensor Condutivo	31
4.2.2. Sensor Ultrassônico	32
4.2.3. Boia e Sensores magnéticos	33

4.2.4. Sensores de nível ICOS	34
4.3. Tecnologia de prototipação	35
4.3.1. Arduinos	35
4.3.1.1. Uno	36
4.3.1.2. Mega	37
4.3.1.3. Arduino Pro Mini	38
4.3.2. Raspberry Pi	38
4.3.3. Beaglebone Black	40
4.4. Discussão	43
5. Implementação	48
5.1. Protótipo de medição de nível d'água utilizando o sensor ultrassônico	48
5.1.1. Introdução	49
5.1.2. Desenvolvimento	49
5.1.2.1. Objetivo geral	49
5.1.2.2. Objetivos Específicos	49
5.1.3. Referencial teórico	50
5.1.3.1. Sensor Ultrassônico	50
5.1.3.2. Display LCD	52
5.1.3.3. LED's	53
5.1.3.4. Buzzer	53
5.1.4. Procedimentos experimentais	54
5.1.5. Resultados	58
5.1.6. Conclusões e recomendações	58
5.2. Protótipo sensor de nível d'água utilizando o sensor de nível ICOS	59
5.2.1. Introdução	59
5.2.2. Desenvolvimento	59
5.2.2.1. Objetivo geral	59
5.2.2.2. Objetivos Específicos	59
5.2.2.3. Materiais utilizados no protótipo com o sensor de nível ICOS:	60
5.2.2.4. Sensor de Nível	60
5.2.2.5. Sensor de Nível ICOS LA16M-40	60
5.2.2.6. Sensor de Nível ICOS LC26M-40	60
5.3. Protótipo sensor de nível d'água utilizando sensores ICOS e ultrassônico e primeira versão da maquete	63
5.3.1. Introdução	63

5.3.2. Desenvolvimento	64
5.3.2.1. Objetivo geral	64
5.3.2.2. Objetivos específicos	64
5.3.2.3. Maquete e materiais	64
5.3.2.4. Procedimentos experimentais	65
5.3.2.5. Resultados	66
5.4. Desenvolvimento e validação do dispositivo de monitoramento	66
5.4.1. Introdução	66
5.4.2. Desenvolvimento	67
5.4.2.1. Objetivo geral	67
5.4.2.2. Objetivos específicos	67
5.4.2.3. Maquete e materiais	67
5.4.2.4. Referencial teórico	70
5.4.2.4.1. Sensores ICOS (LA16M-40)	70
5.4.2.4.2. Sensor ultrassônico	70
5.4.2.4.3. Módulo NRF24L01+ - Comunicação sem fio por rádio frequência	71
5.4.2.4.4. Módulo GPS GY-NEO6MV2 - Geolocalização	72
5.4.2.4.5. Módulo Cartão SD - Armazenamento local de dados	73
5.4.2.4.6. Módulo Wifi ESP8266 ESP-01 - Comunicação sem fio com a internet	73
5.4.3. Procedimentos experimentais	74
5.4.3.1. Sensor node	74
5.4.3.2. Sink node	84
5.4.4. Web Service para persistência de dados de dispositivos de sensoriamento permitindo acesso por aplicativos híbridos	91
5.4.4.1. Introdução	91
5.4.4.2. Web Service, Aplicação Web e Banco de Dados	92
5.4.4.3. Resultados	93
5.4.4.4. Aplicações nativas, híbridas, Ionic e App	95
6. Referências	98

Capítulo 1

1. Introdução

Este capítulo apresenta a motivação, os objetivos e a organização deste trabalho.

São introduzidas as áreas de pesquisa nas quais o trabalho se apoia e descrito o problema central a ser investigado. As tecnologias usadas na solução são brevemente apresentadas, as quais são detalhadas nos capítulos que seguem.

1.1. Motivação

A região de Jacaraípe, composta por 16 bairros e cerca de 40 mil moradores, em épocas de chuvas, sofrem com problemas relacionados às enchentes. Até o início da década de 70 a bacia hidrográfica do rio Jacaraípe era tipicamente rural. O processo de ocupação rápida da região, sem nenhuma preocupação ambiental, impôs grandes modificações na superfície do solo, alterando o escoamento de águas de chuva. Ao mesmo tempo, houve assoreamento do leito do Rio Jacaraípe e estreitamento, devido, entre outros fatores, as construções executadas irregularmente nas suas margens e dentro do próprio rio.

A Lagoa Juara funciona como um reservatório amortecedor de enchentes com grande capacidade de acumulação de água. O Rio Jacaraípe é o escoadouro natural da lagoa, ele é responsável pelo escoamento das chuvas da bacia e pela entrada de água salgada na lagoa, ou seja, quando chove o nível de água da lagoa sobe e o excesso é escoado pelo rio até restabelecer o equilíbrio com o nível do mar e recomeçar o processo de fluxo e refluxo das marés para a lagoa através do rio. Isto faz com que a lagoa Juara seja um enorme criadouro de peixes e de outras espécies marinhas. O assoreamento do rio faz com que a água do mar não chegue à lagoa, pois o nível desta tem se mantido muito acima de seu ponto de equilíbrio natural. Sem a renovação da água, os níveis de oxigênio se mantém muito baixos para sustentar grandes populações de vida aquática (Prefeitura, 2014).

A solução definitiva para os problemas supracitados depende de estudos ambientais e obras com eventuais desapropriações. São processos de médio e longo prazo. Embora não seja a solução para o problema ambiental, realizar o monitoramento constante do nível de água dos elementos que compõe a bacia hidrográfica do rio Jacaraípe, pode ser uma forma imediata, de

baixo custo, para atenuar os problemas decorrentes das enchentes. Neste cenário, as Redes de Sensores sem Fio (Wireless Sensor Networks) (AKYILDIZ et al., 2002; YICK; MUKHERJEE; GHOSAL, 2008) têm se tornado ferramentas essenciais, pois são capazes de realizar monitoramento de grandezas do mundo físico.

Redes de Sensores sem Fio (RSSF) (LOUREIRO et al., 2003) são estruturas formadas por diminutos nós com capacidade de processamento, armazenamento, comunicação e sensoriamento, que permitem aos seus usuários interagirem e observarem, com grande nível de detalhe e precisão, os mais variados ambientes, domínios e objetos de interesse. As suas características de baixo custo, pequena dimensão, flexibilidade e facilidade de implantação dão a essas redes um potencial enorme de aplicação em diversas áreas, permitindo que elas se apresentem como componentes importantes de soluções de monitoramento e controle em vários cenários de aplicação. Por exemplo, em um cenário de Cidades Inteligentes (SU; LI; FU, 2011; RNP 2012), as RSSF podem ser usadas na implantação de serviços de observação e controle de aspectos do cotidiano das cidades, como vigilância de espaços públicos, prédios e outras estruturas civis, monitoramento de rios, córregos (FAVA, Maria Clara et al, 2013; MARTINS JÚNIOR, 2013; ASWALE, Pramod et al. 2015) e matas urbanas (SZEWCZYK, Robert et al., 2004), monitoramento de áreas de desastre e risco, apoio à mobilidade urbana, rastreamento e localização de cargas e objetos, entre outras.

No âmbito desta proposta de projeto de iniciação científica júnior, a problemática referente à bacia hidrográfica do rio Jacaraípe será amplamente discutida; e haverá contato com a área de pesquisa de redes de sensores sem fio e as subáreas envolvidas, com o foco na construção de um dispositivo capaz de monitorar o nível da água com intuito de alertar a população e órgãos competentes sobre possíveis enchentes.

1.2. Objetivos

Este projeto de iniciação científica júnior tem como objetivo geral construir um dispositivo com capacidade de sensoriamento e comunicação sem fio para monitoramento da bacia hidrográfica do rio Jacaraípe para predição de enchentes.

Este objetivo geral pode ser refinado nos seguintes objetivos específicos:

- Realizar uma investigação sobre a bacia hidrográfica do rio Jacaraípe e os problemas

decorrentes da ocupação na região.

- Compor um referencial teórico sobre redes de sensores sem fio e suas áreas de aplicação.
- Projetar um dispositivo de hardware com capacidade de sensoriamento e comunicação sem fio, utilizando componentes existentes no mercado.
- Implementar algoritmos para o dispositivo projetado, com intuito de captar dados dos sensores e realizar transmissão sem fio.
- Investigar possibilidades para predição de enchentes a partir dos dados de sensoriamento.
- Propor mecanismos de alerta para população afetada e aos órgãos competentes.

1.3. Organização do trabalho

Além desta Introdução, o trabalho está organizado em mais seis capítulos, a saber:

- Capítulo 2: Apresenta um breve referencial teórico abordando em diferentes dimensões o problema das enchentes na bacia hidrográfica do rio Jacaraípe.
- Capítulo 3: Discute os trabalhos relacionados
- Capítulo 4: Descreve o projeto da arquitetura conceitual e detalhes da escolha tecnológica para implementação
- Capítulo 5: Apresenta as fases da implementação do protótipo proposto.

Capítulo 2

2. Referencial teórico

Este trabalho explora questões relacionadas às enchentes em rios urbanos. Para um melhor entendimento deste projeto, este capítulo realiza uma revisão dos conceitos básicos dessas áreas de pesquisa. A seção 2.1 introduz a bacia hidrográfica do rio Jacaraípe; A seção 2.2 apresenta o conceito sobre enchentes, assoreamento do rio e as enchentes registradas; A seção 2.3 discute os impactos ambientais e sociais; A seção 2.4 apresenta matérias veiculadas na mídia; e, finalmente, a seção 2.5 traz as ações do poder público sobre as enchentes do rio Jacaraípe.

2.1. Bacia hidrográfica do rio Jacaraípe

O ribeirão Juara, como também é chamado o rio Jacaraípe, corre no sentido oeste leste, sendo um agente de transformação exógeno, desaguando no Oceano Atlântico na praia do Barrote. Possui parte de suas nascentes abrigadas em duas unidades de conservação: a Área de Preservação Ambiental Estadual Mestre Álvaro, que abriga as nascentes da sub bacia do Córrego Dr. Robson, e a Área de Preservação Ambiental Morro do Vilante, envolvendo contribuintes dos córregos Independência, Quibebe e Cavada.

O rio Jacaraípe possui 29 km de extensão, sua largura natural de calha é de cerca de 30 metros e sua profundidade varia entre setenta centímetros e um metro e meio. O rio Jacaraípe é responsável pelo escoamento das chuvas da bacia, como também da entrada de água salgada nas lagoas. A vazão estimada obtida para o rio Jacaraípe na estação chuvosa é de $7,26 \pm 1,55$ m³/s, enquanto na estação seca, a vazão apresentasse uma ordem de magnitude inferior ($0,61 \pm 0,185$ m³/s).

Fazem parte desta sub bacia as lagoas Jacuném e Largo do Juara, sendo essa última situada próxima ao complexo industrial CIVIT. Cada uma delas corresponde à respectivamente 1,4km² e 2,9km² de área, sendo estas formadas pelos córregos de Barro Branco, Jaconé, Vener, córrego Dr. Robson e o córrego São Domingos. O clima

predominante naquela região pode ser considerado tropical úmido, com estação chuvosa no verão e seca no inverno. A quantidade de chuva média anual na bacia do rio Jacaraípe é de 1400mm. Com uma temperatura média mensal, mínima mensal e máxima mensal de, respectivamente, 23,3 °C, 19,4 °C, e 26,9 °C com a umidade variando de 75% a 87%.

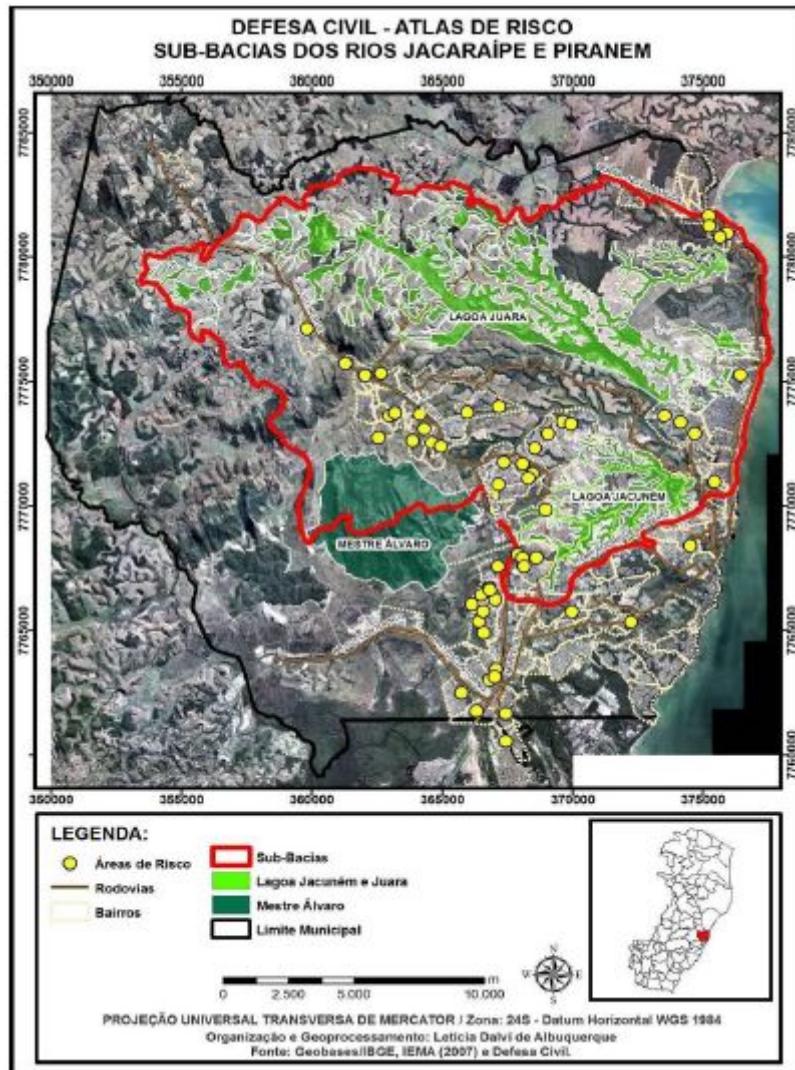


Figura 1 - Bacia hidrográfica do rio Jacaraípe.

Quanto a fauna encontrada na região, contabilizou-se espécies, incluindo os registros históricos e os recentes, pertencentes a 36 famílias em 14 ordens. Dentre estas, 40 espécies são de água doce e 27 são marinhas ou estuarinas. A maioria dos peixes de água doce capturados na bacia pertence às ordens Characiformes e Siluriformes, com 13 espécies cada um. A vegetação nas áreas da bacia é definida principalmente por manguezais.

2.2. Histórico

O processo de ocupação da área próximo a bacia hidrográfica do rio Jacaraípe teve início desde o histórico de ocupação do município da Serra em 1556. Com o desenvolvimento do município baseado na agricultura era mais favorável estabelecer moradia próximo aos rios, com isso, grande parte dos loteamentos foram aprovados antes da Lei 6.766, em função desta falta de normas mais claras, os loteadores implantaram projetos que avançavam sobre as restingas, matas ciliares de rios, lagoas e também sobre fundos de vale, não respeitando a topografia local.

A partir da década de 60, o município da Serra deixou de ser somente agrícola e passou a ter atividade industrial. Com a implantação do Centro Industrial de Vitória CIVIT, a população aumentou significativamente por conta das atividades industriais, o que aumentou ainda mais o problema da ocupação desordenada da região ao redor do rio.

Os processos devido ao loteamento residencial observados na bacia hidrográfica do rio Jacaraípe abrangem também o mal uso do solo para plantio de eucaliptos e pastagens. Essas atividades causam a degradação ambiental, a inutilização do solo, o assoreamento do rio e o risco de degradação da qualidade da água devido o escoamento inapropriado de esgoto doméstico e industrial que comprometem o uso da mesma.

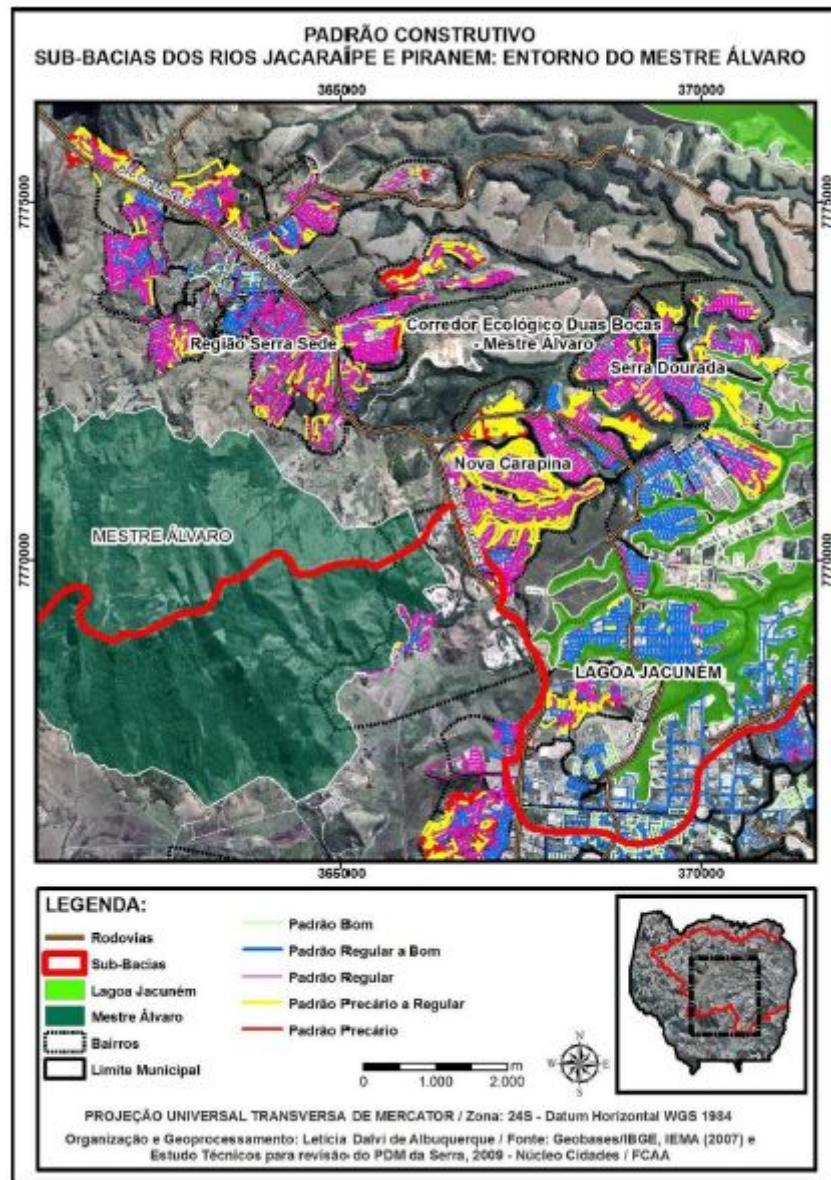


Figura 2 - Padrão construtivo da sub bacia do Rio Jacaraípe.

É ainda válido destacar que, segundo o Censo do IBGE de 2000, a maioria dos bairros localizados na região de estudo é habitado por um grande percentual de famílias com renda de até um salário mínimo.

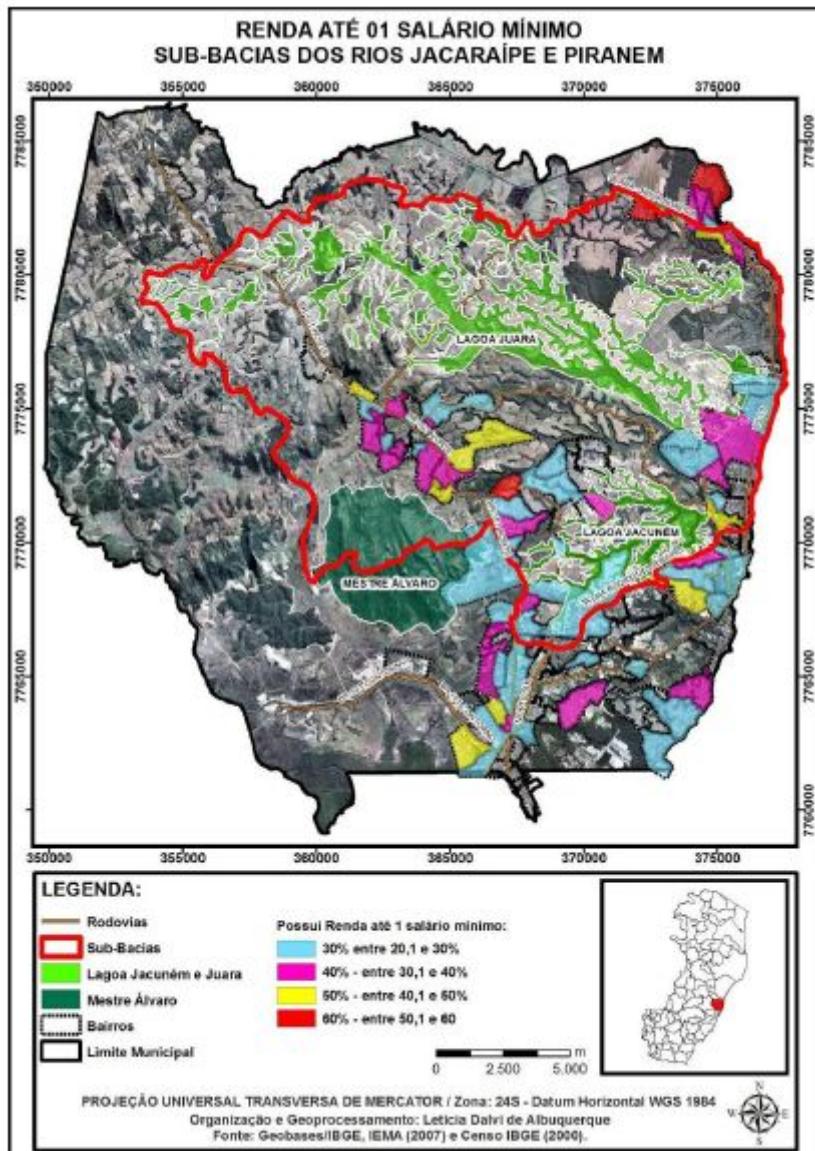


Figura 3 - Renda de até um salário mínimo sub bacia do Rio Jacaraípe (IBGE 2000).

2.3. Importância socioeconômica

A bacia do rio Jacaraípe por localizar-se próxima ao litoral adquire características próprias de ocupação voltadas para as atividades do turismo e do lazer, o que a torna uma área de interesses de agentes que procuram articular o espaço sob a ótica do capital imobiliário. Segundo Moraes, “em termos globais é um dos setores produtivos que mais cresce na zona costeira na atualidade, revelando uma velocidade de instalação exponencial”.

A bacia do rio Jacaraípe é drenada por inúmeros córregos e incorpora à sua extensão as lagoas Largo do Juara e Jacuném, que, em função das suas belezas cênicas, facilmente atrai

investimentos para o setor imobiliário e turístico. A pesca nas duas lagoas é considerada uma atividade de subsistência e de geração de emprego para moradores e pescadores da região.

Um fator relevante sobre a Lagoa Juara, que forma grande parte da área, é sua utilização como local de lazer, recreação, pesca esportiva e artesanal. No ano de 2000, foi criada uma associação de pescadores com apoio da SEBRAE e do Banco do Brasil, assim, eles iniciaram o processo de criação de peixes (tilápias). Os peixes são comercializados na sede da Associação de Pescadores, localizada na margem da lagoa Juara e são a fonte de subsistência de cerca de 50 famílias. O sistema tanques rede da lagoa Juara é o mais antigo da região e serve como modelo para associações de pescadores em outros ecossistemas aquáticos lacustres no estado. Além disso, as lagoas costeiras são consideradas importantes depositários de biodiversidade aquática, sendo ecossistemas altamente produtivos.



Figura 4 - Associação de pescadores de Jacaraípe. (Foto: Letícia Vieira Folha Vitória).

Além destes elementos, também se destaca o conjunto de praias que formam um litoral com 23 km de extensão de orla (onde deságua a sub bacia). Estas praias se apresentam das mais variadas formas, umas mais urbanizadas, como Jacaraípe, dotada de infraestrutura e de equipamentos que dinamizam e permitem a presença de atividades culturais, como os campeonatos de surf, e turísticas na região. Outras praias são mais bucólicas, como Carapebus

e trechos de Manguinhos (Fig. 05), onde ainda é possível encontrar pescadores e muita tranquilidade. Em alguns pontos do litoral, ainda é possível encontrar restingas e falésias (Fig. 05), que embelezam ainda mais a paisagem.



Figura 5 - Trecho bucólico e de restinga.

Também possui competições de *surf*, *windsurf* e *bodyboard* durante o verão, e é muito procurada por surfistas. Tem ondas fortes em certos trechos e pode ser perigosa próximo à desembocadura do rio Jacaraípe.



Figura 6 - Prática de surf nas praias de Jacaraípe.

2.4. O problema das enchentes

O assoreamento é o acúmulo de sedimentos no curso da água podendo ser um fenômeno natural ou humano. Lixos, resíduos e sedimentos (partículas de solo e rochas, que chegam no rio devido a extração da vegetação ciliar) podem estagnar o fluxo da água formando bancos de areia. Quando a quantidade de sedimentos é muito grande e pesada, eles transportam-se por rolamento (no fundo dos rios) ou se acumulam no leito normal, trazendo prejuízos ao escoamento fluvial. A água dos rios, ao encontrar tantos obstáculos, desvia-se, podendo atingir espaços onde antes não existiam cursos d'água.

2.5. Recorrência das enchentes

Novembro de 2008: Lagoa Juara transborda e moradores de Jacaraípe ficam desabrigados;

Julho de 2009: Enchente em São patrício;

Dezembro de 2013: Em 15 dias 900 mm de chuva (parte da BR ES-010 desabou);

Novembro de 2014: Em três horas, 300 mm de chuva;

Outubro de 2014: Enchentes dos bairros próximos ao rio;



Figura 7 - Desabamento de uma parte da BR ES-010.

Bairros afetados e população impactada pelas enchentes ao redor do Ribeirão Juara/Rio Jacaraípe:

Bairro	População
Norte do rio:	
São Patrício	1.232
Parque Jacaraípe	3.333
Residencial Jacaraípe	4.419
Costa Dourada	1.325
Sul do Rio:	
São Pedro	740
Conjunto Jacaraípe	3.101
Castelândia	1.248

Tabela 1 - Bairros.

2.6. Impactos

2.6.1. Ambientais

Observa-se que a ocupação irregular trouxe diversos prejuízos ambientais à região de estudo. O Rio Jacaraípe sempre foi responsável por escoar as águas das chuvas para o mar e também pela entrada de água salgada na lagoa Juara, garantindo a biodiversidade do rio e da lagoa. Com o assoreamento do rio surgiu o problema da falta de oxigenação da água da Lagoa Juara, que não recebe mais a água do mar devido ao alto nível no qual o rio se encontra. O fenômeno da piracema (período de reprodução dos peixes, onde eles se deslocam até as nascentes dos rios ou até regiões rasas dos mesmos com ervas, para desovar) deixou de ser

observado nos últimos anos devido a esse problema.

Na grande enchente que houve em dezembro de 2013, esse problema da falta de oxigenação pode ser observado de forma clara. O grande volume de chuvas, aliado com o assoreamento do rio, fez com que o nível da água permanecesse alto por muitos dias. Isso levou a morte da vegetação ao redor do rio. A decomposição dessa vegetação produziu grande quantidade de gás metano, causando a morte de milhares de peixes, inclusive peixes criados pela Associação de Pescadores da Lagoa Juara.



Figura 8 - Morte de peixes na lagoa Juara devido às grandes chuvas.

Outro problema está ligado ao lançamento de esgoto residencial e resíduos químicos de processos industriais diretamente no leito do Rio Jacaraípe e na Lagoa Juara. Os dejetos sem tratamento e os efluentes se somam, gerando uma carga de poluição elevada demais para a capacidade de depuração de córregos com pequeno fluxo e um ambiente lento (com baixa circulação e renovação, conforme costumam ser as lagoas). O aspecto visual e o odor dos córregos próximos às estações de tratamento de esgoto evidenciam as péssimas condições da água.

Estes impactos ambientais e paisagísticos foram, em geral, causados pela deposição de esgoto residencial; Pela retirada da mata ciliar; Pelo avanço sobre restingas e mangues; Pelas ocupações de fundo de vale e urbanização do mesmo; Pela abertura de vias; Pela deposição de

lixo em encostas e fundos de vale; Pela deposição de esgotos industriais em rios e córregos; entre outros.

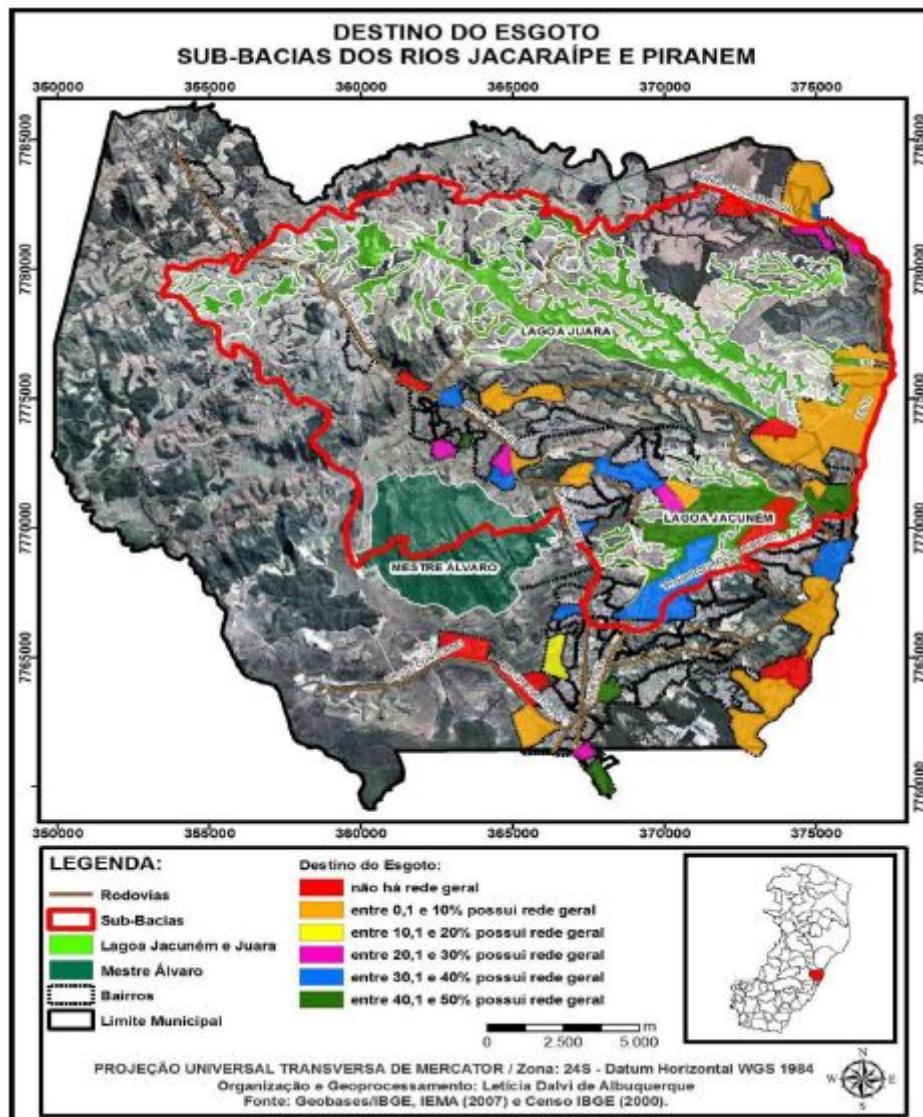


Figura 9 - Destino do esgoto na bacia do Rio Jacaraípe.

2.6.2. Sociais

A ocupação desordenada da região, além de modificar a paisagem ambiental, trouxe também vários problemas para a população, que, ao ocupar áreas com grande declividade, ficou sujeita a desmoronamento, e, ao ocupar áreas alagáveis ou de baixada, ficou sujeita a alagamentos, além de viver em áreas urbanas cada vez menos verdes.

A Associação de Pescadores da Lagoa Juara também sofre muito com as enchentes,

que causam a morte das tilápias criadas na lagoa. Grande parte (e às vezes até total) da renda de algumas famílias vem da pesca.



Figura 10 - Moradores de Jacaraípe são obrigados a sair de suas casas devido às enchentes.

2.7. Mídia

2.7.1. Obras

- TV Gazeta - 10/04/2014

<http://g1.globo.com/espirito-santo/noticia/2014/04/obra-do-rio-jacaraipe-fica-pronta-ate-2015-promete-prefeito-da-serra-es.html- 10/04/2014>

Obra do Rio Jacaraípe fica pronta até 2015, promete prefeito da Serra, ES. 'Vamos resolver o problema de enchente na região', diz Audifax Barcelos. São R\$ 15 milhões de investimentos para beneficiar 40 mil moradores.

- Jornal Tempo Novo - [www.portaltemponovo.com.br-](http://www.portaltemponovo.com.br/) 29/08/2014

Rio Jacaraípe: Impasse sobre a dragagem está perto do fim. A prefeitura da Serra, responsável pela dragagem, afirma ter começado a atender as exigências do Instituto do Patrimônio Histórico Nacional, que embargou a atividade há dois meses. O objetivo da obra

era a recuperação das matas ciliares e dos manguezais.

- Século Diário - seculodiarionline.com.br/- 19/12/2013

A prefeitura da Serra assinou dia 20 de dezembro de 2013, uma ordem de serviço para a macrodrenagem do Rio Jacaraípe. Foram investidos 10 milhões de reais. A obra não foi concluída.

- Século Diário - seculodiarionline.com.br/- 16/03/2014

O Estudo e Relatório de Impacto Ambiental (Eia/Rima) da obra de Recuperação Hidráulica do Rio Jacaraípe serão apresentados em audiência pública. A medida é passo obrigatório para o licenciamento ambiental das obras de macrodrenagem do manancial.

- TV Gazeta - <http://g1.globo.com/espírito-santo/noticia/>- 10/04/2014

Obra do Rio Jacaraípe fica pronta até 2015. Com a promessa de reduzir os alagamentos, a Prefeitura da Serra, no Espírito Santo, está fazendo uma obra de macrodrenagem no Rio Jacaraípe.

- Diário Oficial da União - <http://www.jusbrasil.com.br/diarios> - 07/11/2013

Art. 2º O Projeto de Alargamento e Dragagem do Rio Jacaraípe apresentado visa a recuperação hidráulica da bacia do rio Jacaraípe e propõe obras de regularização das margens do Rio Jacaraípe e urbanização da área, com previsão das seguintes intervenções: Dragagem, Abertura de calha, Remoção de edificações em área de Risco e seu reassentamento, Recuperação de áreas degradadas, Urbanização e Paisagismo simplificados para disciplinar e limitar a ocupação do entorno a qual inclui: Taludes, Revegetação, Praças e Micro Praças com equipamentos de lazer e contemplação, conforme plantas apresentadas no referido processo, e faz parte do Programa Gestão de Riscos e Respostas a Desastres dentro da Ação: Drenagem Urbana Sustentável e Manejo de Águas Fluviais e contará com recursos vindos do Governo Federal - Ministério das Cidades, através da Caixa Econômica Federal, no âmbito do Programa de Aceleração do Crescimento-PAC II .

2.7.2. Enchentes

- Folha Vitória - 27/10/2014

<http://www.folhavitoria.com.br/geral/noticia/2014/10/quase-um-ano-apos-tragedia-municios-capixabas-se-preparam-para-evitar-novas-mortes-com-enchentes.html>

Quase um ano após tragédia, municípios capixabas se preparam para evitar novas mortes com enchentes. Temendo o pior, municípios que registraram mortes estocam produtos de higiene e limpeza e fazem planos de contingência, treinamentos de pessoas, e até simulações de enchentes. Os bairros mais atingidos na Serra têm a limpeza das caixas e bocas de lobo mais constantes. A principal obra na Serra para o período das chuvas é a dragagem e urbanização do Rio Jacaraípe.

- Gazeta Online - 26/11/2008

http://gazetaonline.globo.com/_conteudo/2008/11/35715-lagoa+transborda+e+moradores+de+jacaraipe+ficam+desabrigados.html

Lagoa Juara transborda e moradores de Jacaraípe ficam desabrigados. A população que mora no entorno do Rio Jacaraípe e Lagoa Juara em Jacaraípe, no município da Serra, ainda sofre com as consequências da chuva que atingiu o Estado. Várias casas ficaram inundadas, nesta terça-feira (25), depois que a lagoa Juara transbordou. Muitos moradores já tiveram que deixar suas residências.

Capítulo 3

3. Trabalhos relacionados

Este capítulo apresenta exemplos de trabalhos correlatos, que tratam de questões de pesquisa relacionadas aos temas centrais deste projeto. São descritas abordagens que propõem algum tipo de infraestrutura para monitoramento de rios para predição de enchentes. O capítulo está estruturado como segue. A Seção 3.1 apresenta exemplos de infraestruturas de monitoramento de rios e a Seção 3.2 traz as considerações finais do capítulo.

3.1. Monitoramento de rios

Nos últimos anos, temos presenciado um aumento no número de pesquisas e um grande interesse na implantação de infraestruturas de redes de sensores para monitoramento de rios. O interesse é justificado pela promessa dessas infraestruturas oferecerem à população mecanismos de alerta a fim de atenuar os impactos decorrentes das enchentes.

Na literatura da área, existem vários trabalhos que descrevem infraestruturas para monitoramento de rios. Exemplos representativos são os trabalhos descritos em (PECHOTO et al, 2013), (FLORIANO et al, 2014), (LOFFI et al, 2016) e (FAVA et al, 2013), os quais são brevemente discutidos a seguir. Para efeitos de avaliação, formam o escopo da discussão os seguintes requisitos:

- Verificar e calcular a variação do nível do rio em um determinado instante de tempo;
- Encaminhar informações para uma central/banco de dados que ativará métodos de alerta para avisar a população sobre a enchente eminent;
- Integração com redes sociais

3.1.1. E-noé: Rede de sensores sem fio para monitorar rios urbanos

Em (PECHOTO at al, 2013) é apresentado um projeto de monitoramento em tempo real dos rios de São Carlos-SP. O nó da rede (composto pelo mote XBee, sensor de pressão, bateria e painel solar) envia os dados lidos pelo sensor de pressão através do protocolo de comunicação sem fio ZigBee IEEE 802.15.4 para o gateway. O gateway adotado foi um roteador sem fio IEEE 802.3, utilizando OpenWRT. Para possibilitar a comunicação com os motes do projeto, um módulo XBee foi conectado ao roteador. Os dados recebidos pelo gateway, oriundos dos motes, são encaminhados para o servidor, onde são processados e feita a análise de criticidade do nível do rio. O software presente no servidor gera gráficos em tempo real, conforme os dados são coletados pelos sensores de pressão no rio. A infraestrutura proposta foi integrada ao SISMADEN - desenvolvido pelo Instituto Nacional de Pesquisas Espaciais (INPE), em que as leituras captadas pelos sensores são transformadas em arquivos padrão PCD (Ponto de Coleta de Dados) enviados via protocolo FTP.

3.1.2. MONIT-RIO– Tecnologia da informação de comunicação para monitoramento de rios em casos de cheias

Em (LOFFI et al, 2016) é proposto um protótipo para monitoramento do nível do Rio Itajaí Açu e de chuva. Foi utilizado um Sensor Infravermelho Sharp, uma placa microcontroladora (Arduino Uno), Shield de Ethernet, para o Arduino, e antenas e cabos para enviar os dados e recepção dos mesmos na sede. No projeto foram utilizados uma boia (Isopor), um pluviômetro eletrônico, um Arduino, um sensor Sharp e uma régua de acrílico para ser demarcado os pontos de metragem. Um computador no local foi utilizado para recolher os resultados via transferência USB. Os dados foram recolhidos, no período de uma hora, em Ituporanga e os resultados obtidos foram satisfatórios ao existente no sistema CEOPS. Porém não o projeto de pesquisa não foi concluído por falta de verba.

3.1.3. SIGMAOn–Sistema de Informação Geográfica para Monitoramento de Alagamentos On-line

Em (FLORIANO et al, 2014) é proposto um Sistema de Informação Geográfica para Monitoramento de Alagamentos On-line – SIGMAOn. A ferramenta foi desenvolvida para a utilização de órgãos governamentais, da população bem como empresas privadas para permitir que sejam detectadas as regiões passíveis de alagamentos das áreas de risco das cidades cadastradas. Foi utilizado serviços de mapas do Google para apresentar visualmente as informações aos usuários. Dentre as funcionalidades do sistema estão a simulação de pontos e áreas de alagamento e a consulta de rotas não alagadas entre localidades da cidade. Para o desenvolvimento do projeto foi utilizado a APIs e serviços de manipulação de mapas, a Google Maps Javascript API v32 foi escolhida por ser gratuita e dispor de ferramentas que atendem as necessidades do sistema.

3.1.4. Proposta metodológica para previsões de enchentes com uso de sistemas colaborativos

Em (FAVA at al, 2013) foi apresentado uma metodologia para integrar informações voluntárias a modelos de previsão em curto prazo, apresentando uma solução para o preenchimento de dados espacialmente em locais onde não se têm sensores de monitoramento. O uso dos dados de VGI tem como principais vantagens o aprimoramento e atualização em tempo real das previsões nos pontos monitorados por sensores WSN e estimar os valores dos níveis dos pontos não monitorados.

3.1.5. Discussão

A Tabela 2 apresenta uma comparação dos trabalhos analisados. Na tabela, o símbolo “+” significa que o trabalho atende completamente ao requisito avaliado, o “–“ indica que o trabalho atende apenas parcialmente, enquanto que o símbolo “•” representa o não atendimento ao requisito. Os requisitos estão assim identificados: Verificar e calcular a

variação do nível do rio em um determinado instante de tempo (R1); Encaminhar informações para uma central/banco de dados que ativará métodos de alerta para avisar a população sobre a enchente eminente (R2); Integração com redes sociais (R3).

Trabalhos	Requisitos		
	R1	R2	R3
(PECHOTO et al, 2013)	+	-	•
(LOFFI et al, 2016)	+	•	•
(FLORIANO et al, 2014)	•	-	•
(FAVA et al, 2013)	•	-	•

Tabela 2 - Análise entre requisitos e trabalhos relacionados.

Conforme mostrado na Tabela 2, nenhum dos trabalhos analisados atende ao requisito R3, ratificando a existência de uma carência de uma infraestrutura que suporte integração com redes sociais a fim de potencializar a divulgação de informações críticas sobre o rio monitorado. Apenas (PECHOTO et al, 2013) e (LOFFI et al, 2016) desenvolveram protótipos para monitoramento efetivo de rios. Por fim, os trabalhos analisados não contemplam a geração de alertas para a população impactada em situações de enchentes, referentes ao rio monitorado.

Desta maneira, o desenvolvimento de uma infraestrutura que atenda aos requisitos elencados deve representar uma contribuição à literatura da área e possibilitar uma atuação mais efetiva para o seu objetivo – monitorar rios e divulgar informações críticas, com intuito de minimizar os impactos decorrentes das enchentes.

Capítulo 4

4. Arquitetura proposta

Este capítulo apresenta o projeto conceitual da infraestrutura e as tecnologias para implementação proposta neste trabalho. A seção 4.1 apresenta os componentes da infraestrutura conceitual proposta. A seção 4.2 descreve elementos que compõe os dispositivos propostos e as tecnologias relacionadas. A seção 4.3 discute as tecnologias de prototipação. Por fim, a seção 4.4 apresenta uma discussão sobre os requisitos e tecnologias para definir.

4.1. Infraestrutura proposta

A Figura 11 apresenta a infraestrutura proposta, composta de dois tipos de dispositivos de monitoramento – *Sink node* e *Sensor node*; a *cloud services* e o aplicativo *mobile*. Os dispositivos de monitoramento têm como objetivo realizar o processo de sensoriamento do nível do Rio Jacaraípe, eles devem estar dispostos no curso do rio, em pontos críticos, a uma distância limitada a capacidade do módulo de comunicação sem fio (Figura 12). O *Sink node* tem uma função adicional, provendo um mecanismo de conexão com à Internet e com o *cloud services*. Este último, trata-se de um arcabouço de serviços providos em uma plataforma de computação em nuvem, que viabiliza a disponibilização dos dados coletados, integração com redes sociais e envio de alertas aos aplicativos para dispositivos móveis.

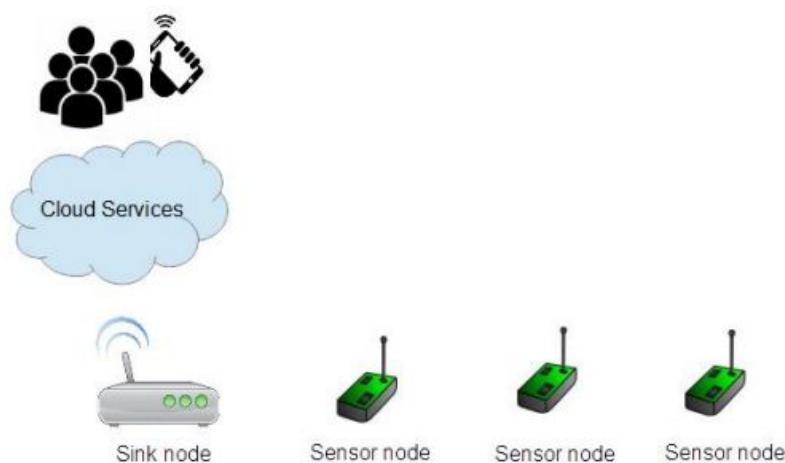


Figura 11: Infraestrutura proposta.

A figura abaixo ilustra a bacia hidrográfica do rio Jacaraípe, no qual é possível observar a lagoa do Juara, o estreitamento do rio, os bairros afetados pelas enchentes e o encontro das águas do mar e do rio Jacaraípe.



Figura 12: Vista aérea do rio Jacaraípe (Google Maps).

A seção seguinte apresenta a estrutura conceitual dos dispositivos de monitoramento.

4.2. Dispositivos de monitoramento idealizados

A Figura 13 apresenta os elementos que compõe os dispositivos de monitoramento. O *Sensor node* é composto por uma plataforma de prototipação, no qual é possível incrementar outros componentes de hardware e software a fim de especializar sua função; um componente de armazenamento de informações coletadas no processo de sensoriamento; um elemento capaz de prover a geolocalização do dispositivo, o sensor de nível capaz de monitorar o rio Jacaraípe e uma interface de comunicação sem fio de longo alcance para viabilizar a comunicação entre os dispositivos ao longo do curso do rio.

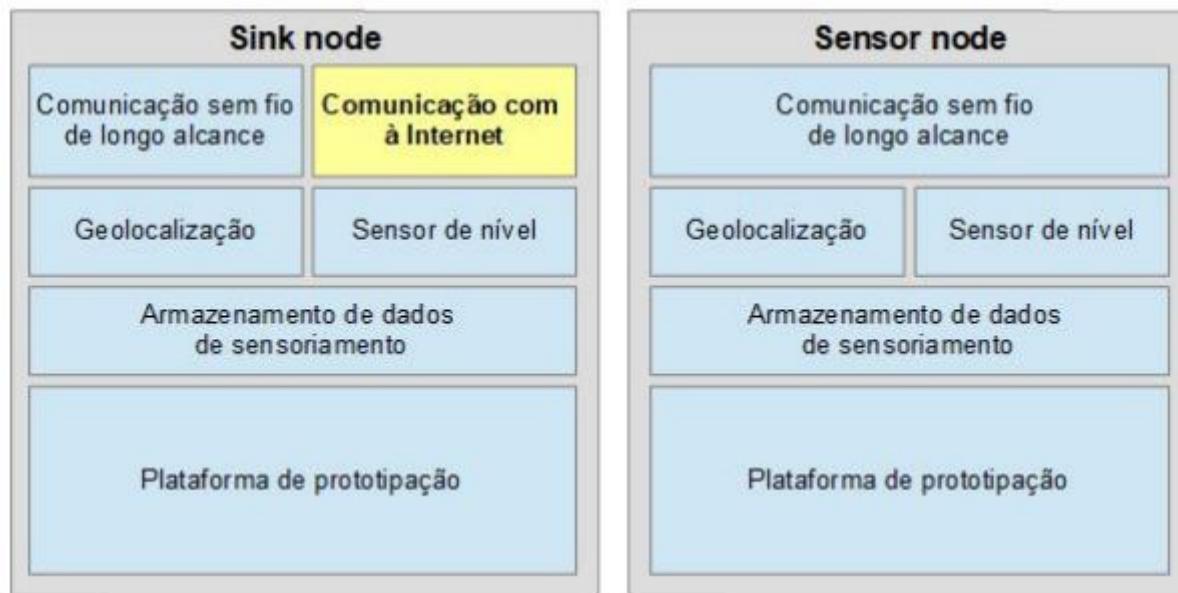


Figura 13: Sink node e Sensor node.

O *Sink node* difere por possuir uma função adicional – um componente que disponibiliza conexão com à Internet sem fio. A próxima seção discute a análise de tecnologias para construção dos dispositivos de monitoramento.

4.2.1. Sensor Condutivo

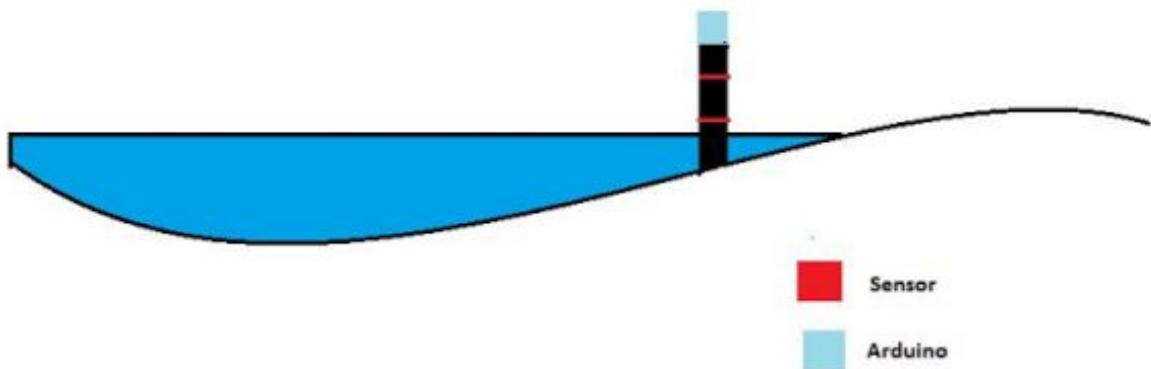


Figura 14 - Sensores condutivos.

Sustentado por uma haste que atinge o fundo do rio, possui dois sensores em determinada altura da haste, que determinam o nível da água. Quando a água tocar o primeiro sensor, o dispositivo entrará em estado de alerta. Este primeiro nível de alerta, é o alerta de

possível alagamento. O segundo nível de alerta, é o alerta de enchente eminentes. O segundo alerta, é o que ativará todos os meios de comunicação, sejam eles via postagem em redes sociais, efeitos sonoros e/ou visuais, mensagens SMS etc.

Uma das formas de se implementar o dispositivo, é através do uso do Shield Sensor de Nível, pois ele possibilita a detecção de níveis de água facilmente sem a necessidade de boias mecânicas ou sensores caros. O Shield Sensor de Nível poderá ser utilizado em conjunto com três hastes de aço inox que representarão, respectivamente, o nível de referência, o primeiro nível de alerta e o nível de alerta para enchente eminentes, ou através da instalação de parafusos de aço inox em determinadas alturas em tubos de PVC. A altura da instalação do parafuso determinará o nível de alerta. Assim que o nível da água for alto o suficiente para submergir o parafuso, um sinal será enviado ao Arduino informando qual nível de alerta deve ser ativado.

Prós: Baixo custo dos equipamentos, e uma alta velocidade de alerta de enchentes.

Contras: Não identificado

4.2.2. Sensor Ultrassônico



Figura 15 - Protótipo com sensor ultrassônico.

Sustentado por uma haste, dispõe de um sensor que emite ondas sonoras em alta frequência. A onda toca a água e volta ao sensor, que mede o tempo de retorno e a velocidade da onda, convertendo em distância ($d = [v * t]/2$). Este sensor tem um problema pois objetos podem atravessar o caminho das ondas antes destas colidirem com a água, porém,

é facilmente corrigido por um tubo que protegeria a passagem das ondas sonoras.



Figura 16 - Sensor Ultrassônico.

Prós: Baixo custo dos equipamentos e maior precisão na medição do nível do rio.

Contras: A temperatura do ambiente, turbulências no ar, pressão e umidade podem influenciar no desempenho do sensor.

4.2.3. Boia e Sensores magnéticos

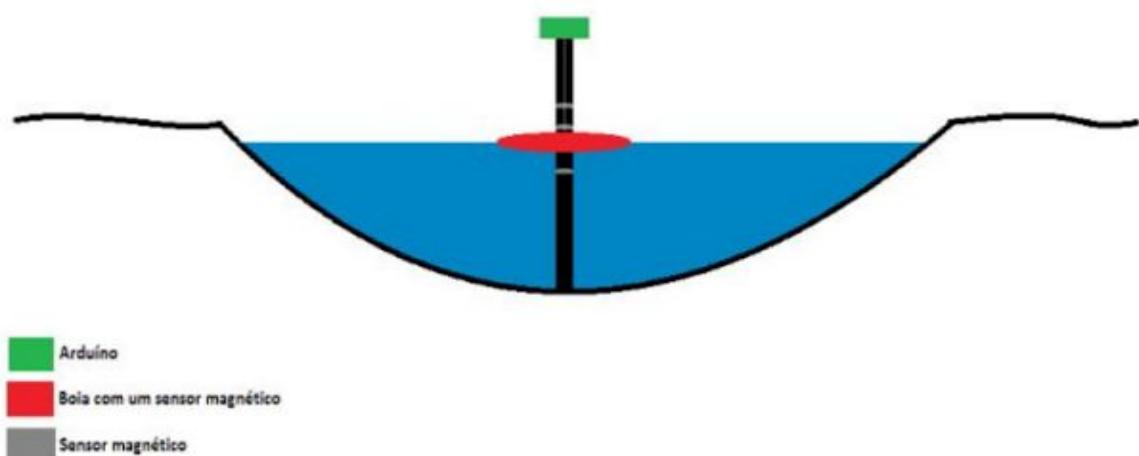


Figura 17 - Protótipo com bóia e sensores magnéticos.

A ideia se baseia em uma haste, com uma boia em volta, localizada no centro do Rio Jacaraipe. Ao longo da haste serão encontrados sensores magnéticos, assim como na boia. Conforme o nível da água estiver subindo, os sensores da boia reconhecerão os da haste, enviando um sinal para o Arduino, que irá calcular o tempo, e avisará a população de alguma forma.



Figura 18 - Sensor Magnético.

Prós: A ideia se torna vantajosa quando verificamos os custos dos produtos, que são baixos, além de não afetar a fauna e a flora do rio.

Contras: A proposta poderá atrapalhar as navegações.

4.2.4. Sensores de nível ICOS

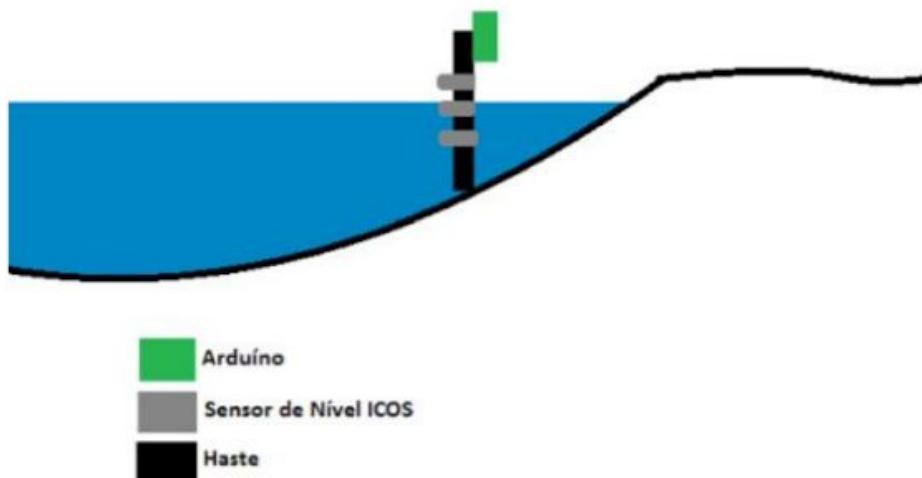


Figura 19 - Protótipo com sensores ICOS.

A proposta é colocar sensores de nível para líquidos (Sensor de Nível ICOS) na margem do rio. Esses sensores, que estarão presos em um cano, por onde passarão os fios, funcionarão como interruptores, ou seja, assim que a água do rio subir, eles serão ativados, e

irão enviar uma mensagem ao Arduino, que avisará à população de alguma forma (ainda será definida). A haste será envolvida em uma tela para evitar que lixos possam atrapalhar o funcionamento destes dispositivos.



Figura 20 - Sensores ICOS.

Prós: A ideia traz um baixo custo, fácil implementação e manutenção, além de não apresentar problemas com lixo (devido à tela de proteção).

Contras: Não identificado.

4.3. Tecnologia de prototipação

Prototipação é uma abordagem baseada em uma visão evolutiva do desenvolvimento de software, afetando o processo como um todo. Envolve a produção de versões iniciais – protótipos (análogo a maquetes para a arquitetura) – de um sistema futuro com o qual é possível realizar verificações e experimentos, com o intuito de avaliar algumas de suas características antes que o sistema venha realmente a ser construído, de forma definitiva.

4.3.1. Arduinos

Projeto italiano de prototipagem eletrônica, que torna a robótica mais acessível.

Suas unidades são constituídas por um controlador AVR de 8 bits, pinos analógicos e digitais de entrada e saída, portas USB (para conexão direta com computadores).

Não possui recursos de rede, mas pode ser combinada com outras placas criando extensões chamadas *shields*, como um *shield* de rede (*Ethernet Shield*), por exemplo.

Possui código aberto, que quando modificado dá origem a seus derivados que possuem o “ino” em seu nome, como: Netduino, Produino, Garagino.

Software pode ser programado em C e C++.

4.3.1.1. Uno

Microcontrolador	ATmega328
Tensão de operação	5V
Tensão de entrada (recomendada)	7-12V
Tensão de entrada (limites)	6-20V
Pinos de I/O Digitais	14 (6 deles com saída PWM)
Pinos Analógicos	6
Corrente CC por I/O Pino	40 mA
Corrente do Pino 3.3V	50 mA
Memória Flash	32 KB (ATmega328) 0.5 KB usado pelo bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Velocidade do Clock	16 MHz
Comprimento	68.6mm
Largura	53.4mm
Peso	25g

Tabela 3 - Especificações do arduino Uno.

Dentre os Arduinos, é a primeira opção de compra, pois possui um bom número de

portas e grande compatibilidade com os *shields*.

Esta placa, na versão com soquete, permite a troca do chip microcontrolador ATMEGA328 facilmente em caso de dano ao microcontrolador ou se o mesmo for utilizado em projetos dedicados. Existe também a placa Arduino Uno versão SMD, com o microcontrolador soldado na placa.

4.3.1.2. Mega

Microcontrolador	ATmega2560
Tensão de operação	5V
Tensão de entrada (recomendada)	7-12V
Tensão de entrada (limites)	6-20V
Pinos de I/O Digitais	54 (14 deles com saída PWM)
Pinos Analógicos	16
Corrente CC por I/O Pino	40 mA
Corrente do Pino 3.3V	50 mA
Memória Flash	256 KB (ATmega2560) 8 KB usado pelo bootloader
SRAM	8 KB (ATmega2560)
EEPROM	4 KB (ATmega2560)
Velocidade do Clock	16 MHz
Comprimento	101.52mm
Largura	53.3mm
Peso	37g

Tabela 4 - Especificações do arduino Mega.

Esse Arduino possui uma quantidade maior de portas, com isso é ótimo para projetos

mais elaborados que utilizam muitas portas. Também permite o uso de *shields*.

Exemplo de projeto com ela é automatizar uma casa, controlar um relé (interruptor eletromecânico) para acionar as tomadas, lâmpadas, etc.

4.3.1.3. Arduino Pro Mini

O Arduino Pro Mini é uma placa microcontroladora baseada na ATmega328. Ele possui 14 portas digitais de entrada/saída (das quais 6 podem ser usadas como saídas PWM), 6 portas de entrada analógica e espaços para montagem de pinos. O Arduíno Pro Mini foi desenvolvido para utilização em projetos semipermanentes. A placa vem sem headers pré-montados, dando a opção da utilização de diversos tipos de conectores ou solda direta de cabos. Existem 2 versões da placa Pro Mini. Uma que funciona com 3.3V e 8 MHz e outra que funciona com 5V e 16 MHz.

4.3.2. Raspberry Pi

O Raspberry Pi é um microcomputador que parece um chip e possui um tamanho semelhante a um cartão, e oferece as mesmas funções de um desktop. Ele foi desenvolvido com um propósito educacional e possui: processador, processador gráfico, slot para cartões de memória, interface USB, HDMI, etc.

Hardware de baixo consumo, o Raspberry Pi, é um computador mais indicado para o uso em máquinas genéricas, sistemas de controle e unidades que geram menos calor e gastam menos energia. Quando adquirido, para operá-lo é necessária uma fonte de alimentação de pelo menos 1000mAh, um cabo HDMI, uma memória microSD / SD com o adaptador OS e Wi-Fi ou um cabo RJ45 para se conectar à internet.

Existem atualmente dois modelos: Modelo A e Modelo B. A grande diferença entre os dois modelos é que o Modelo B possui um controlador Ethernet e duas portas USB, enquanto o Modelo A possui apenas uma entrada USB e nenhuma porta de Ethernet.

Esta plataforma não possui uma disponibilidade tão grande no mercado nacional se comparado com o arduino. No entanto, ele pode ser encontrado em diferentes lojas virtuais com filiais no mundo todo, tais como: RS Components, Amazon, etc. O preço desse

equipamento pode variar de 80 a 130 reais.

Especificações do Raspberry Pi B+	
SoC	Broadcom BCM2835 (CPU, GPU, DSP, SDRAM, e uma porta USB);
CPU	700 MHz ARM1176JZF-S core
GPU	Broadcom VideoCore IV 250 MHz
Memória (SDRAM)	512MB (compartilhada com a GPU)
Portas USB 2.0	4 via hub USB de 5 portas on-board
Saídas de Vídeo	Vídeo Composto (PAL e NTSC) através de conector 3.5 mm TRRS (P2) com saída de áudio integrada / HDMI (ver 1.3 e 1.4) / Interface MIPI para ligar diretamente a painéis LCD;
Saídas de Áudio	Saída de áudio analógica através de conector 3.5 mm (P2) compartilhada com o vídeo composto / HDMI;
Armazenamento Integrado	Entrada para cartão MicroSD
Rede Integrada	Adaptador Ethernet 10/100 Mbps USB (8P8C) integrado na quinta porta do hub USB interno
Portas GPIO	17 com suporte ao barramento HAT ID
Consumo energético	600 mA (3.0 W) apenas a placa
Fonte de energia	5v
Tamanho	85.6 mm x 56 mm
Peso	45g

Tabela 5 - Especificações Raspberry PI B+.

4.3.3. Beaglebone Black

BeagleBone Black é um projeto de hardware aberto e de baixo custo, desenvolvido pela Texas Instruments junto ao grupo BeagleBoard. Com objetivo de ser uma plataforma de prototipagem para computação móvel. Uma das principais propostas da Beaglebone é torná-la uma referência similar às placas Arduino, já que também usa o conceito de shields (placas de expansão, que para a Beaglebone Black recebem o nome de capes), porém com um processador muito mais potente e com muito mais possibilidades em relação a software.

A Beaglebone Black não é interessante somente para projetos pessoais ou pesquisa, mas também para empresas, que podem se aproveitar da fácil reprodução deste equipamento. O dispositivo possui um processador AM3359, ele é um ARM Cortex-A8 com acelerador gráfico 3D produzido pela Texas Instruments, roda em até 720 MHz, possui 32K de cache L1, 256K de cache L2, 176K de ROM e 64K de RAM interna. Também possui controlador LCD de 24 bits e controlador para a interface touchscreen.

A BeagleBone Black possui 256MB de SDRAM, e não tem memória flash, então é necessário o uso de um cartão SD como unidade de armazenamento. Possui dois conectores de expansão de 46 pinos que podem fornecer diferentes conexões e barramentos como SPI, I2C, GPIO, LCD, HDMI, VGA, MMC, etc. Você pode alimentá-la através de uma fonte externa ou pela porta USB (que fica ao lado da porta Ethernet). Através da porta externa o processador dela trabalha até 720 MHz e pela a porta USB até 550 MHz. A porta USB também fornece conexão serial com a placa.

Nos conectores de expansão é onde podemos ligar as *capes*, abaixo temos uma imagem de uma cape com saída de áudio e vídeo.

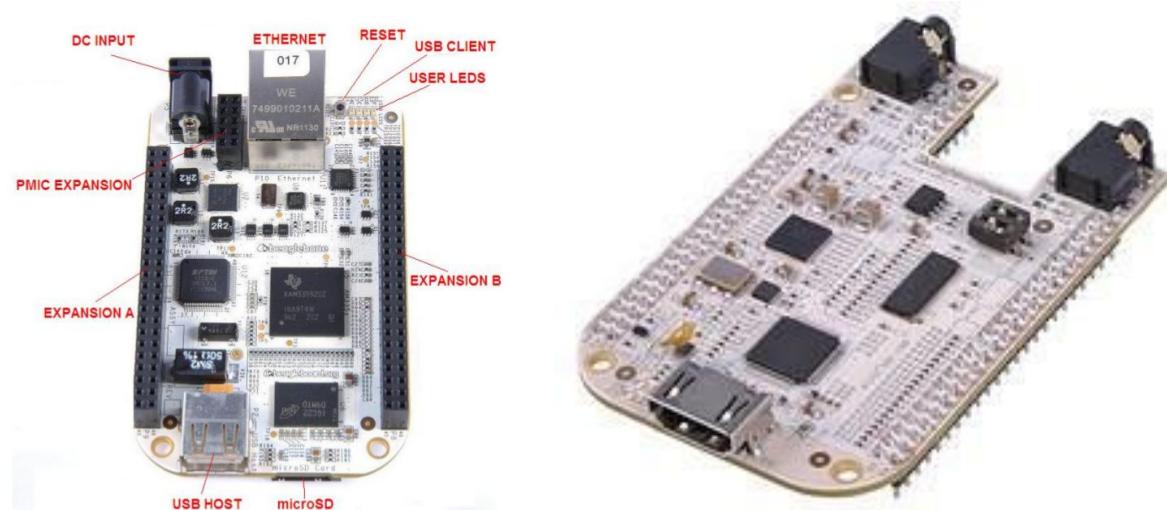


Figura 21 - BeagleBone Black e seu cape de áudio e vídeo.

Especificações do Processador:	
Chip	AM335x 1GHz ARM® Cortex-A8
Memória	512MB DDR3 RAM
Armazenamento	4GB 8-bit eMMC on-board flash storage
3D graphics accelerator	
NEON floating-point accelerator	

Interface:
USB 2.0 Client, para alimentação e comunicação
USB 2.0 Host
10/100M Ethernet (Conector RJ45)
Interface LCD
HDMI

Slot cartão TF
2x 46 Pinos

Compatibilidade de Softwares:
Debian
Android
Ubuntu
Cloud9 IDE em Node.js com biblioteca BoneScript

Especificações Gerais:	
Tensão de operação	5V/0,35 ^a
Temperatura de operação	0-70°C
Dimensões	86,36 x 54,61mm

Tabela 6 - Especificações Beaglebone Black.

4.4. Discussão

Tabela de estruturas:

Material	Resistência	Manutenção	Vida útil	Nível de oxidação	Unidade	Preço
Cano PVC	Baixa	Fácil	Baixa	Baixo	Peça	R\$ 40,00
Cano metálico	Alta	Difícil	Alta	Médio	Unidade	R\$ 14,90
Adaptador de cano PVC em T	Baixa	Fácil	Baixa	Baixo	Peça	R\$ 35,00
Adaptador de cano metálico em T	Alta	Difícil	Alta	Médio	Peça	R\$ 8,56
Caixa plástica elétrica	Baixa	Fácil	Baixa	Baixo	Peça	R\$ 94,00
Tela Galvanizada	Alta	Difícil	Alta	Médio	Rolo	R\$ 40,00
Tela plástica	Baixa	Fácil	Baixa	Baixo	Rolo	R\$ 139,90
Hastes de aço inox	Alta	Fácil	Alta	Baixo	Metro	R\$ 20,00
Parafusos inox	Alta	Fácil	Alta	Baixo	Peça	R\$ 2,80
Boia salva vidas	Alta	Fácil	Alta	Baixo	Peça	R\$ 99,00
Painel solar	Média	Fácil	Alta	Baixo	Peça	R\$ 219,00
Fio	Média	Fácil	Alta	Médio	Rolo	R\$ 49,90
Bateria	Média	Fácil	Alta	Médio	Rolo	R\$ 105,00

Tabela 7 - Estruturas das tecnologias de prototipação.

Tabela de Sensores:

Sensores	Resistência	Precisão	Implementação	Manutenção	Sem fio	Preço
Nível ICOS	Média	Ótima	Fácil	Fácil	Sim	R\$ 30,00/pç
Magnético	Média	Boa	Regular	Regular	Não	R\$ 14,90/pç
Shield sensor de nível	Alta	Boa	Fácil	Fácil	Sim	R\$ 14,90/pç
Ultrassônico	Média	Ótima	Regular	Fácil	Sim	R\$ 17,84/pç

Tabela 8 - Sensores e características.

Tabela de processamento:

Equipamento	Clock	Memória	Conexão
Raspberry PI 1 B+	700 MHz	512 MB	4 USB
Raspberry PI 1 A+	700 MHz	256 MB	1 USB
Raspberry PI 2 B	900 MHz	1 GB	4 USB
Arduino Uno	16 MHz	32 K (0,5 K usado pelo <i>bootloader</i>)	USB
Arduino Mega	16 MHz	256 K (8 KB usado pelo <i>bootloader</i>)	USB
Arduino Pro Mini	8 ou 16 MHz	16 K (2 K usado pelo <i>bootloader</i>)	Serial / módulo USB externo

Tabela 9 - Processamento dos equipamentos.

Tabela de comunicação wireless:

Módulos	Vantagens	Desvantagens	Preço
NRF24L01 Wireless Transceiver 2,4GHz	Barato e de fácil instalação e manutenção.	Alcance limitado: 10 metros em ambientes internos e 50 metros em campo aberto.	R\$ 120,00
Módulo Digi XBeePRO ZB (S2C)	Alcance estimado em 3200m.	Não consta.	R\$ 282,00

Tabela 10 - Módulos de comunicação wireless.

Equipamento	Arduino Uno	Arduino Mega	Arduino Pro Mini	Raspberry PI	Beaglebone Black
Shield sensor de nível	X	X	X	-	O
Sensor ultrassônico	X	X	X	O	O
Sensor de nível ICOS	X	X	X	X	X
Sensor Bóia Magnética	X	X	X	X	X
Alimentação por bateria/painel solar	X	X	X	X	X
Comunicação Wireless	X	X	X	X	O
Simuladores	X	X	X	O	O
Disponibilidade/Mercado	X	X	X	X	O

Tabela 11 - Requisito e tecnologias de prototipação.

Legenda:

X = suporta

O = suporta parcialmente

- = não suporta

Os Arduinos Uno, Mega e Pro Mini foram escolhidos através dos estudos realizados e das possíveis soluções encontradas. Com isso conclui-se que a região necessita de uma atenção específica, já que não seriam viabilizadas ideias que atrapalhassem o desenvolvimento ecológico do rio e até mesmo socioeconômico da região. Assim foi possível idealizar soluções que amenizassem o problema das enchentes, sem que interferisse no equilíbrio ecológico.

Com essas observações, foi identificado que, para este projeto, as placas Arduino levam vantagem e se mostram mais viáveis. Estas placas se mostram superiores por sua simplicidade, por ser um equipamento barato, por possuir maior resistência de hardware e por funcionar utilizando apenas uma bateria, que pode ser carregada por um painel solar. Além disso, ela possui uma capacidade de leitura de dados analógicos dos sensores em tempo real, o que só poderia ser conseguido com o Raspberry Pi ou o BeagleBone Black com a assistência de outros periféricos.

Sem a realização de uma fase de testes, foi identificado que dentre as placas Arduino estudadas, o Arduino Mega seria a melhor opção. Os principais motivos que levaram a essa conclusão foi o grande número de portas que a placa possui e sua capacidade de processamento. Embora a placa do Arduíno Mega possua dimensões maiores do que as placas Uno e Pro Mini, foi identificado que isso não se tornaria um problema e não irá afetar o projeto de forma significante. A placa Uno do Arduino também parece atender aos requisitos do projeto. Sendo sua principal diferença em relação à placa Mega, a quantidade de pinos oferecidos, podemos considerar, após a fase de testes e a consequente verificação de que a placa Uno suporta a quantidade de sensores a ser utilizada, que esta placa possa ser adotada para o projeto.

Ambas as placas Mega e Uno são placas que possibilitam a montagem e desmontagem de circuitos de forma fácil e simples, pois ambas as placas vêm com headers já embutidos nos pinos. Além disso, essas placas possuem entradas USB *builtin*, o que facilita o upload dos sketches desenvolvidos. A placa Pro Mini do Arduino, embora sem uma fase de testes atenda aos requisitos do projeto, não possui as características mencionadas acima, já que é uma placa desenvolvida para utilização em projetos semipermanentes. É necessário um serial 20 conversor *USBtoTTL* para a realização de upload de sketches e é necessário a compra de outros tipos de conectores ou a solda direta dos fios nos pinos a serem utilizados.

Conclui-se então que, após a fase de testes e à consequente verificação de que a placa Pro Mini atenda à capacidade de processamento e à quantidade de sensores a serem utilizados, essa placa possa ser utilizada para a definitiva instalação do modelo desenvolvido em pontos definidos no Rio Jacaraípe. A placa foi desenvolvida para esse fim, logo, possui tamanho muito reduzido em relação às placas Mega e Uno, os fios podem ser soldados, o que torna as conexões mais fortes, e a placa possui maior resistência.

OBS.: Caso necessário, poderemos trabalhar duas tecnologias ao mesmo tempo: Compreende-se que a melhor opção seria utilizar ambos os sistemas embarcados, pois o Arduino e o Raspberry Pi “se completam”. O Arduíno possui uma capacidade em tempo real de leitura dos dados dos sensores, enquanto o Raspberry Pi é um minicomputador com a capacidade de receber e manipular esses dados, podendo enviar e-mail, SMS e fazer postagens em redes sociais, ou seja, comunicar-se com a população e os órgãos de defesa de forma mais eficiente e rápida.

Capítulo 5

5. Implementação

Este capítulo apresenta a implementação da infraestrutura para monitoramento proposta neste trabalho. A seção 5.1 apresenta o uso do sensor ultrassônico, composição do circuito e código fonte. A seção 5.2 demonstra a utilização do sensor ICOS, explicitando a lógica de programação aplicada e o projeto de circuito elétrico. A seção 5.3 demonstra o primeiro protótipo com o combinado entre sensores ICOS e ultrassônico para a obtenção do nível d'água. Esta seção apresenta também a primeira versão da maquete representando o leito do Rio Jacaraípe a ser produzida. A seção 5.4 demonstra o protótipo final construído, utilizando um combinado de sensores ICOS com o sensor ultrassônico para monitoramento do nível, além de elementos de geolocalização, armazenamento de dados local, comunicação por rádio frequência e comunicação com a internet. Os métodos e tecnologias utilizados na criação do aplicativo de celular para emissão de alertas também são apresentados.

5.1. Protótipo de medição de nível d'água utilizando o sensor ultrassônico

O modelo de protótipo aqui apresentado faz uso do sensor ultrassônico para a medição de nível d'água. Esse modelo foi primeiramente proposto na fase de levantamento de requisitos (fase 2) como uma das opções mais viáveis para implementação de um dispositivo de monitoramento nos componentes da Bacia Hidrográfica do Rio Jacaraípe.

Foi realizado um estudo da pinagem do sensor ultrassônico e de outros componentes eletrônicos que foram utilizados no desenvolvimento do protótipo, sendo estes: 3 LEDs de cores distintas; 1 BUZZER; e 1 display LCD.

Na parte lógica do protótipo, o código foi escrito de duas maneiras: uma com o uso da biblioteca “Ultrasonic” no auxílio do funcionamento do sensor, e uma segunda maneira sem o uso da biblioteca.

5.1.1. Introdução

Este relatório tem como finalidade expor a pesquisa que está sendo realizada pelos alunos do Programa de Iniciação Científica Júnior.

O projeto, que tem a intenção de construir um dispositivo para medição do Rio Jacaraípe, está sendo realizado a partir de estudos com base na geografia do local, nas formas de implementação, e principalmente no funcionamento dos componentes do dispositivo.

O sensor ultrassônico é o mais viável a conquistar nosso objetivo por ter maior facilidade de ser implantado e por ter resistência superior aos outros, pois não há contato. Ele também possui uma grande diferença: ele permite que você veja a variação de um nível de alerta para outro, podendo medir a velocidade e ajudando a calcular o tempo que levará para atingir o próximo nível, já sensores de contato nos mostram apenas se a água chegou ao nível ou não.

5.1.2. Desenvolvimento

5.1.2.1. Objetivo geral

Nosso projeto tem como base o monitoramento da capacidade de um copo de água que tem a profundidade de 16 cm (dezesseis centímetros), tendo em vista o monitoramento do rio Jacaraípe futuramente.

5.1.2.2. Objetivos Específicos

- Realizar medição de nível d'água;
- Definir níveis de alerta;
- Definir para cada nível de alerta uma determinada função;
- Emitir sinais visuais e sonoros;
- Exibir dados da leitura no display LCD.

5.1.3. Referencial teórico

A ligação do sensor ao Arduino no nosso programa vai utilizar, além dos pinos de alimentação, os pinos digitais, 4 para o Trigger e 5 para o Echo. A alimentação será feita pelo pino 5V do arduino.

5.1.3.1. Sensor Ultrassônico

O sensor ultrassônico calcula a distância entre o sensor e um objeto. Seu funcionamento consiste basicamente em enviar um sinal que ao atingir um objeto é espelhado de volta para o sensor e com base nesse tempo entre o envio e recebimento, é calculada a distância. É enviado um sinal com duração de 10 ms (microsegundos) ao sensor, indicando que a medição terá início, automaticamente, o *trigger* envia 8 pulsos de 40 KHz e aguarda o retorno do sinal pelo receptor (*echo*), e caso haja um retorno de sinal (em nível HIGH), determinamos a distância entre o sensor e o obstáculo utilizando a seguinte equação: Distância = (pulso em nível alto x velocidade do som (340m/s) /2 (D=VxT). A divisão por 2 é necessária já que estamos contando o tempo de ida e volta do sinal.



Figura 22 - Sensor Ultrassônico.

Diagrama de tempo HC-SR04

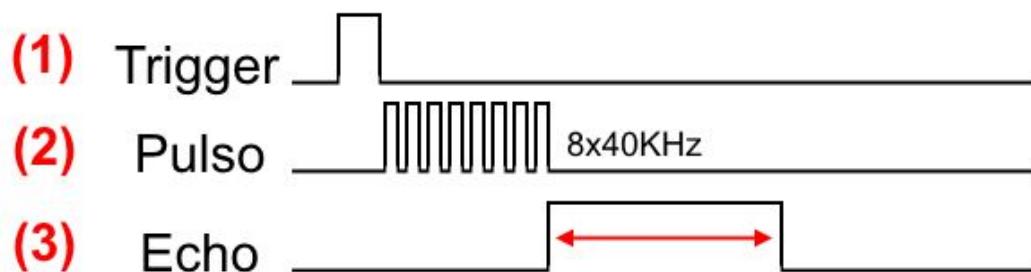


Figura 23 - Diagrama de funcionamento.

Trigger	faz o disparo do sinal do sensor ultrassônico
Echo	recebe o sinal rebatido por um obstáculo
GND	conexão com terra
VCC	alimenta o sensor com 5V

Tabela 12 - Pinos do Sensor Ultrassônico.

5.1.3.2. Display LCD

O display LCD exibe dados de sensores ou simplesmente palavras que o programador queira exibir em sua tela, que é de 16x2 em forma de matriz.

Pin No	Função	Nome
1	Ground (0V)	Ground
2	voltagem de suporte 5V (4,7V - 5,3V)	Vcc
3	Ajuste do contraste; Através de um resistor variável	Vee
4	Seleciona o registro de comando quando está LOW; E registro de dados quando HIGH	Register Select
5	LOW para gravar no registro; HIGH para ler do registro	Read/Write
6	Envia dados para os pinos de dados quando um pulso de alto a baixo é dado	Enable
7	8-bit data pins	DB0
8		DB1
9		DB2
10		DB3
11		DB4
12		DB5
13		DB6
14		DB8
15	Luz de fundo VCC (5V)	LED+
16	Luz de fundo Ground (0V)	LED-

Tabela 13 - Especificações LCD.

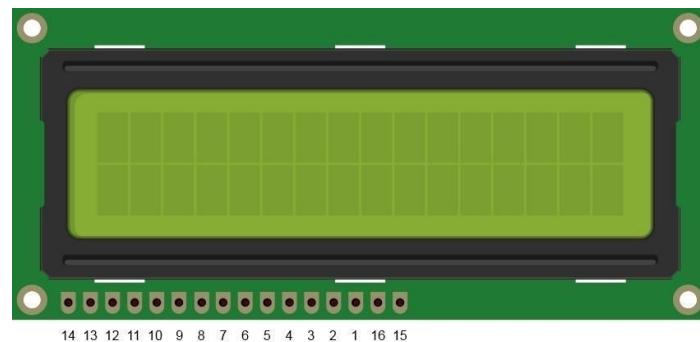


Figura 24 - Ilustração do LCD.

5.1.3.3. LED's

O LED é uma pequena luz que recebe sinal em corrente elétrica do Arduino para acender ou brilhar. Ele suporta de 1,8V a 2,5V (varia com o LED usado), e o Arduino envia tensão equivalente a 5V. Para que o LED não seja danificado usa-se um resistor limitador de corrente, e neste caso o resistor seria de aproximadamente 150 ohms.

O cálculo para saber qual resistor usar é o seguinte: $R = V/I$, onde V é volts, I é a intensidade da corrente medida em Ampère e R é a resistência elétrica medida em ohm.



Figura 25 - LED's.

5.1.3.4. Buzzer

O *buzzer* (buzina, campainha) emite sinal sonoro quando ativado.



Figura 26 - Buzzer.

5.1.4. Procedimentos experimentais

Como pode ser visto na imagem abaixo, o monitoramento é realizado com o sensor ultrassônico (modelo HC-SR04), que mede através de ondas sonoras emitidas pelo sensor, sendo assim calculadas. A equipe também utiliza um LCD modelo 1602A, que mostra a distância (em centímetros) e o nível da água (em porcentagem), além dos LEDs que informam o estado do copo, sendo eles normal, em alerta e crítico; além de um buzzer que emite um sinal quando a água atinge o nível “crítico”.

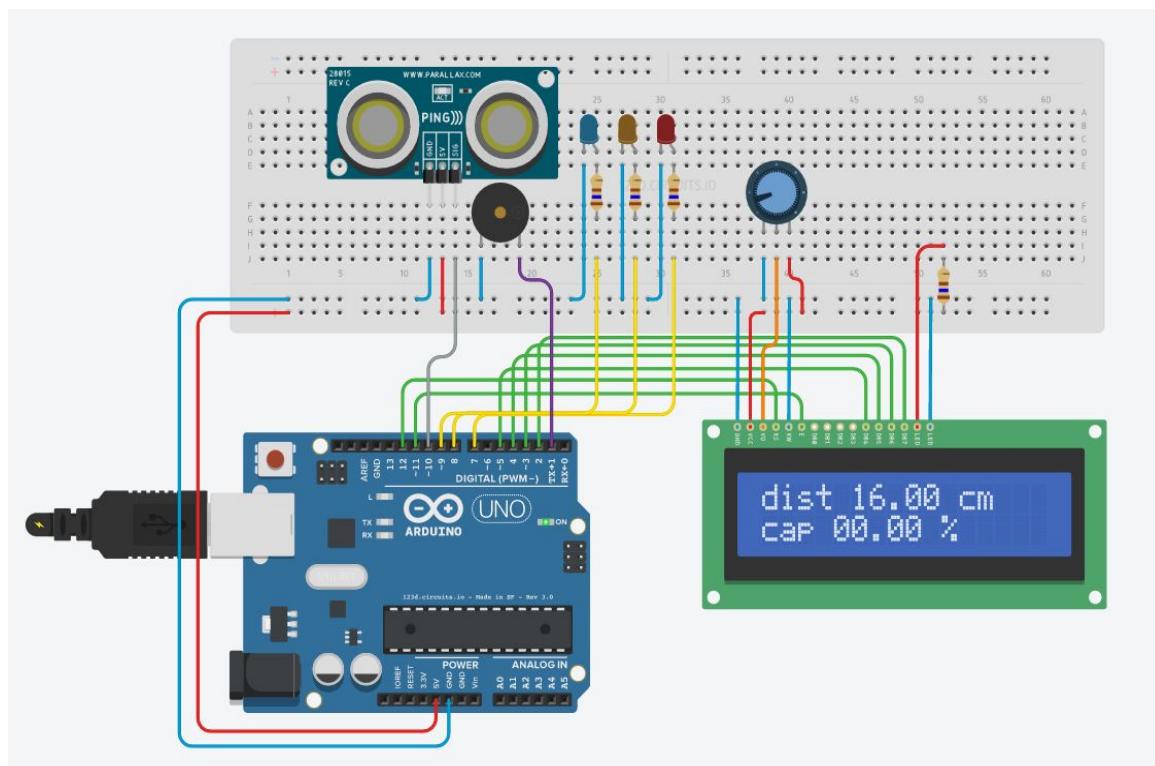


Figura 27 - Montagem do protótipo no simulador.

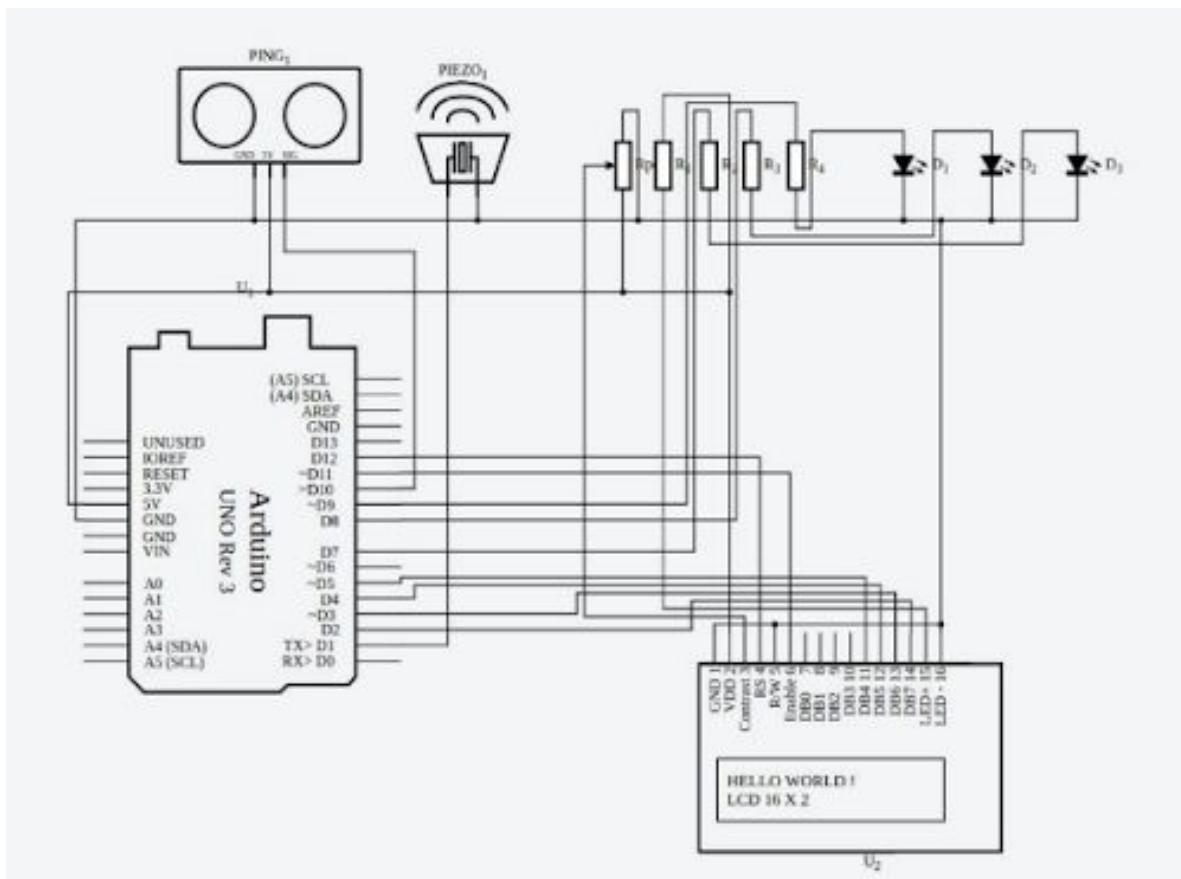


Figura 28 - Esquema elétrico do protótipo.

Durante a fase de testes, aprendemos sobre o sensor ultrassônico, assim como sobre a utilização do LCD. Depois disso, programamos o circuito sem utilizar a biblioteca do sensor ultrassônico e sem utilizar os LEDs, após, com a utilização dos LEDs e, enfim, com a biblioteca própria do sensor (juntamente com os LEDs).

Em um recipiente foi fixado o sensor ultrassônico, e assim o código abaixo foi produzido:

```
#include <LiquidCrystal.h> //Biblioteca utilizada para fazer o controle do LCD

#include <Ultrasonic.h> //Biblioteca utilizada para realizar as funções do
sensor Ultrassônico

#define PINO_TRG 30 // Define os pinos do Arduino ligados ao Trigger
#define PINO_ECHO 32 // Define os pinos do Arduino ligados ao Echo

Ultrasonic ultrasonic(PINO_TRG, PINO_ECHO); // Inicializa o sensor Ultrasonico
nos pinos especificados
```

```
LiquidCrystal lcd(7, 6, 5, 4, 3, 2); // Configura os pinos do Arduino para se  
comunicar com o LCD  
  
int azul = 24; // Pino 24 será saída do LED Azul  
int amarelo = 23; // Pino 23 será saída do LED Amarelo  
int vermelho = 22; // Pino 22 será saída do LED Vermelho  
  
int buzz = 12; // Pino 12 será saída da buzina  
  
void setup() {  
    pinMode(azul, OUTPUT); // Pino representando LED Azul  
    pinMode(amarelo, OUTPUT); // Pino representando LED Amarelo  
    pinMode(vermelho, OUTPUT); // Pino representando LED Vermelho  
  
    pinMode(buzz, OUTPUT); // Pino representando a buzina  
  
    Serial.begin(9600); // Iniciamos a porta serial com Baud Rate de 9600  
    lcd.begin(16, 2); // Inicia o LCD com dimensões 16X2(Colunas X Linhas)  
}  
  
void loop() {  
    float distancia; // Declarando a variável distância  
    long microsec = ultrasonic.timing(); // Lê os valores do sensor Ultrasonico  
  
    // Atribui os valores em cm  
    distancia = ultrasonic.convert(microsec, Ultrasonic::CM);  
  
    int nivel = 100 - distancia * 6.2; // Calcula a capacidade em porcentagem  
  
    // Usamos o IF, um estrutura de condição, para verificar as distâncias  
  
    // Se for maior que 11...  
    if (distancia > 11) {  
        digitalWrite(azul, HIGH); // O LED azul acende  
        digitalWrite(amarelo, LOW); // O LED amarelo permanece apagado  
        digitalWrite(vermelho, LOW); // O LED vermelho permanece apagado  
    }  
  
    // Se for menor igual a 11 e maior que 6...  
    if ((distancia <= 11) && (distancia > 6)) {  
        digitalWrite(azul, HIGH); // O LED azul permanece aceso  
        digitalWrite(amarelo, HIGH); // O LED amarelo acende  
        digitalWrite(vermelho, LOW); // O LED vermelho permanece apagado  
    }  
  
    // Se for menor igual a 6  
    if (distancia <= 6) {  
        digitalWrite(azul, HIGH); // O LED azul permanece aceso  
        digitalWrite(amarelo, HIGH); // O LED amarelo permanece aceso  
        digitalWrite(vermelho, HIGH); // O LED vermelho acende
```

```
digitalWrite(buzz, HIGH); // A buzina apita
delay(500); // Aguarda 0.5 segundos
digitalWrite(buzz, LOW); // A buzina desliga
delay(500); // Aguarda 0.5 segundos
}

// exibe dados no display LCD
lcd.clear();
lcd.setCursor(0, 0); // Posiciona o cursor na coluna zero(0) e na linha
zero(0)

lcd.print("Dist:"); // Printa a palavra Dist
lcd.setCursor(6, 0); // Posiciona o cursor na coluna seis(6) e na linha
zero(0)

lcd.print(distancia); // Printa o valor da variável distância
lcd.setCursor(11, 0); // Posiciona o cursor na coluna dez(10) e na linha
zero(0)

lcd.print("cm"); // Printa a palavra cm

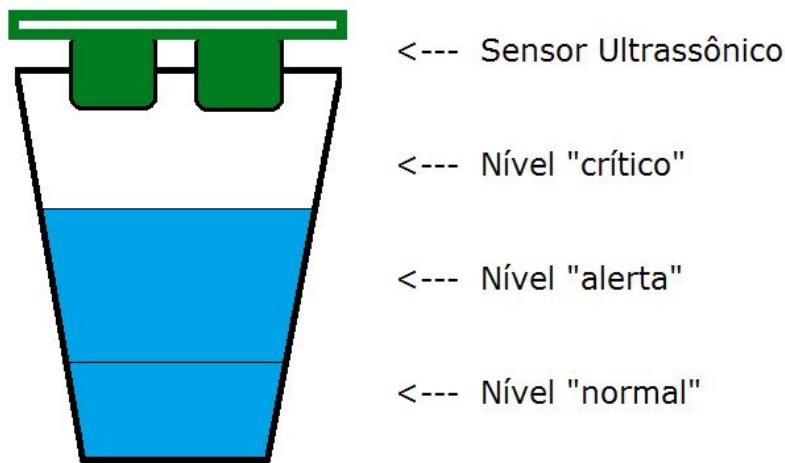
lcd.setCursor(0, 1); // Posiciona o cursor na coluna zero(0) e na linha um(1)

lcd.print("Nível:"); // Printa a palavra Quant
lcd.setCursor(7, 1); // Posiciona o cursor na coluna sete(7) e na linha um(1)

lcd.print(nível); // Print o valor da variável capacidade
lcd.setCursor(10, 1); // Posiciona o cursor na coluna onze(11) e na linha
um(1)

lcd.print("%"); // Print sinal numérico
delay(100); // Aguarda
}
```

Neste código temos três níveis de alerta, alerta 1 - led verde, alerta 2 - led amarelo, alerta 3 - led vermelho, a partir do alerta 2 o sensor envia o sinal avisando que os níveis da água estão subindo e poderá causar enchentes, sendo assim a população é avisada através de alguma sirene etc...



5.1.5. Resultados

- A medição do nível d'água foi realizada com sucesso;
- Foi exibido os dados de níveis no display LCD;
- Os LEDs 1 (verde), 2 (amarelo), 3 (vermelho) acenderam nos devidos níveis de alerta;
- Foi emitido o sinal sonoro de alerta para nível crítico.

5.1.6. Conclusões e recomendações

O objetivo de simular um pequeno teste relacionado à ultrapassagem de um limite, nesse caso a água – recurso ao qual é dedicado o projeto – foi realizado de maneira a nos ensinar sobre as funções presentes dentro de uma biblioteca ultrassônica e o cálculo realizado sem a biblioteca, possibilitando-nos concluir o aprendizado geral do sensor ultrassônico e do LCD, que foi incluído para mostrar a capacidade e o nível da água.

5.2. Protótipo sensor de nível d'água utilizando o sensor de nível ICOS

O nosso protótipo a ser desenvolvido tem como o objetivo principal o monitoramento de um recipiente de modo digital utilizando dois sensores ICOS: horizontal e vertical. Finalizaremos com o uso integrado do sensor, serão feitos dois experimentos, nos quais apresentaram separadamente o uso de cada sensor.

5.2.1. Introdução

O presente experimento tem a finalidade de montar um protótipo utilizando os sensores de nível ICOS, visando um problema real.

Estes sensores detectam o nível do recipiente a partir da altura que foram instalados, utilizando um contato on/off como saída, permitindo assim, sinalizar os níveis máximo e mínimo e também os intermediários do recipiente.

5.2.2. Desenvolvimento

5.2.2.1. Objetivo geral

Montar um protótipo utilizando os sensores de nível ICOS verticais e horizontais.

5.2.2.2. Objetivos Específicos

- Detectar o nível d'água;
- Definir níveis de alerta;
- Emitir sinais visuais e sonoros.

5.2.2.3. Materiais utilizados no protótipo com o sensor de nível ICOS:

Materiais	Preços
Galão de 10 litros	R\$ 10,00
Tubo Marrom 20mmx3m	R\$ 8,90
Joelho 90° 20 mm	R\$ 0,65
Tê 90° 25 mm	R\$ 1,30

Tabela 14 - Materiais para construção do protótipo com sensor ICOS.

5.2.2.4. Sensor de Nível

O sensor indica através do sinal ON/OFF quando o nível de líquido foi atingido, sendo o LA16M-40 instalado na parte lateral, através do furo Ø16mm. Para cada nível de líquido a detectar necessário um sensor de nível no ponto desejado. O LC26M-40 instalado pelo interior, através de furo de Ø16mm.

5.2.2.5. Sensor de Nível ICOS LA16M-40

Monitora nível no ponto (altura) em que o Sensor for instalado em seu reservatório, sendo **ideal para água, óleo, combustíveis e lubrificantes**.

5.2.2.6. Sensor de Nível ICOS LC26M-40

Detecta 1 nível de água ou líquidos em geral no topo (máximo) ou fundo (mínimo) de seu reservatório.

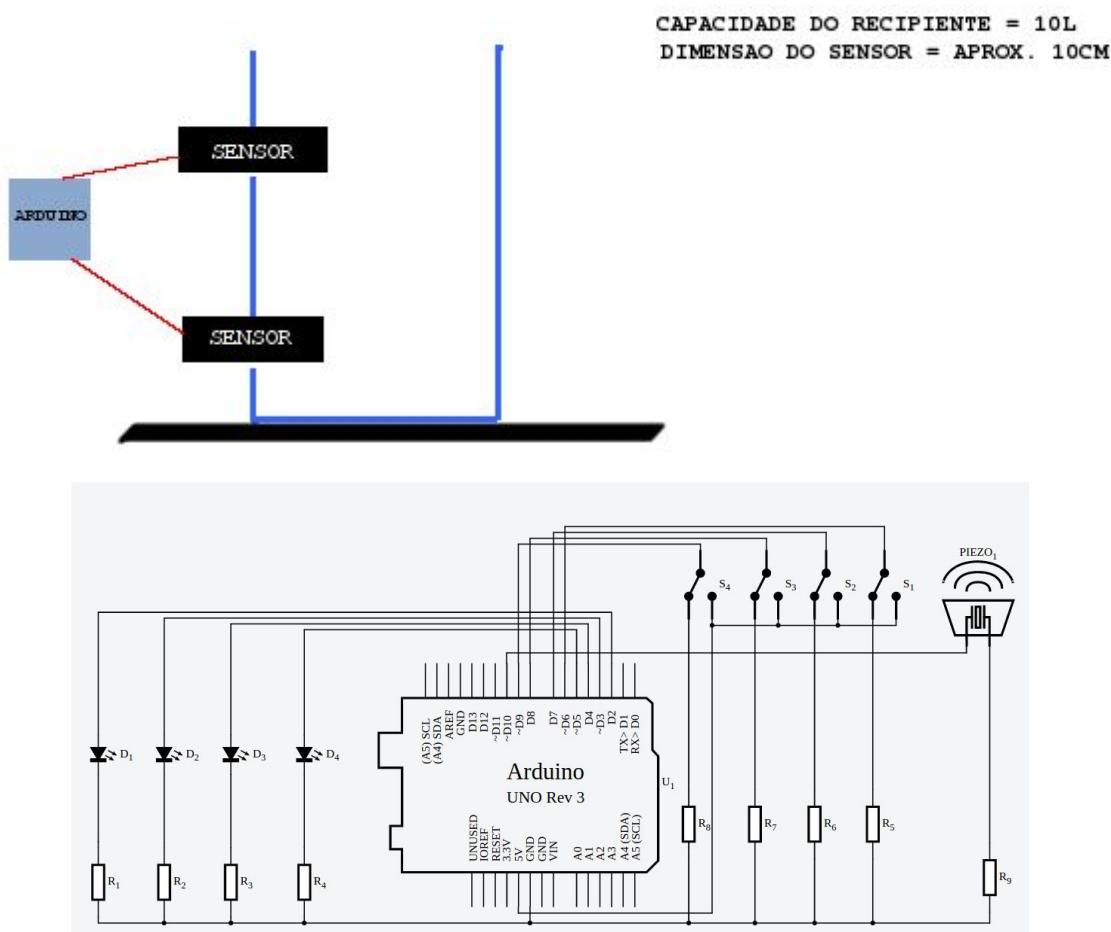


Figura 29 - Sensores ICOS e sua montagem elétrica.

Código com sensores ICOS:

```
//define pino de entrada dos sensores icos de 1 a 4.  
int s1 = 13;  
int s2 = 12;  
int s3 = 11;  
int s4 = 10;  
  
//define pino de entrada dos LEDs de 1 a 4.  
int l1 = 9;  
int l2 = 8;  
int l3 = 7;  
int l4 = 6;  
  
//define pino de entrada do buzzer.  
int buzzer = 5;  
  
void setup(){  
  
    //define os pinos como entrada ou saída.
```

```
pinMode(s1, INPUT);
pinMode(s2, INPUT);
pinMode(s3, INPUT);
pinMode(s4, INPUT);
pinMode(l1, OUTPUT);
pinMode(l2, OUTPUT);
pinMode(l3, OUTPUT);
pinMode(l4, OUTPUT);
pinMode(buzzer, OUTPUT);

// inicia o serial monitor.
Serial.begin(9600);
}

void loop(){

//armazena os dados lidos pelos sensores.
int estado1 = digitalRead(s1);
Serial.print("sensor 1: ");
Serial.print(estado1);

int estado2 = digitalRead(s2);
Serial.print("sensor 2: ");
Serial.print(estado2);

int estado3 = digitalRead(s3);
Serial.print("sensor 3: ");
Serial.print(estado3);

int estado4 = digitalRead(s4);
Serial.print("sensor 4: ");
Serial.println(estado4);

//define uma ação para o estado dos sensores.
if(estado1 == HIGH){
    digitalWrite(14, HIGH);
}
else{
    digitalWrite(14, LOW);
}

if(estado2 == HIGH){
    digitalWrite(13, HIGH);
}
else{
    digitalWrite(13, LOW);
}

if(estado3 == HIGH){
    digitalWrite(12, HIGH);
```

```

}

else{
    digitalWrite(l2, LOW);
}

if(estado4 == HIGH){
    digitalWrite(l1, HIGH);
}
else{
    digitalWrite(l1, LOW);
}

if(estado1 && estado2 && estado3 && estado4 == HIGH){
    digitalWrite(buzzer, HIGH);
}
else{
    digitalWrite(buzzer, LOW);}
```

5.3. Protótipo sensor de nível d'água utilizando sensores ICOS e ultrassônico e primeira versão da maquete

5.3.1. Introdução

O presente experimento tem a finalidade de montar um protótipo utilizando um combinado com os sensores de nível ICOS e o sensor ultrassônico, visando dados concretos para o nível do rio, criando um dispositivo que apresente pequenos riscos à falhas.

Os sensores de nível ICOS detectam o nível do recipiente a partir da altura que foram instalados, utilizando um contato ON/OFF como saída, permitindo assim, sinalizar os níveis máximo e mínimo e também os intermediários do recipiente.

O sensor ultrassônico, instalado no topo do recipiente que simula o rio, faz a leitura em tempo real do nível do mesmo. Essa leitura é importante pois permite, através da criação de um histórico de dados, a análise do comportamento do nível do rio durante diferentes épocas do ano ou momentos do dia.

5.3.2. Desenvolvimento

5.3.2.1. Objetivo geral

Montar um protótipo com o combinado dos sensores ICOS e sensor ultrassônico e realizar testes em uma maquete que simule um rio.

5.3.2.2. Objetivos específicos

- Detectar o nível d'água;
- Definir níveis de alerta;
- Emitir sinais visuais e sonoros.

5.3.2.3. Maquete e materiais

Com o intuito de testar o dispositivo em laboratório, foi necessário a construção de uma maquete que simulasse o leito do Rio Jacaraípe. A primeira versão da maquete contou com dois recipientes d'água, um funcionando como reservatório e outro como o próprio leito do rio; uma bomba d'água que simulasse a subida do nível d'água; e uma estrutura que tornasse a simulação possível.

Materiais	Preço	Quantidade
Recipiente 10 litros	R\$ 9,99	2x
Tubulações	R\$ 20,00	--
Caixa de madeira	R\$ 60,00	1x
Bomba Submersa	R\$ 80,00	1x
Placa Arduino Mega 2560	R\$ 74,90	1x
Sensores ICOS	R\$ 29,90	2x
Sensor Ultrassônico	R\$ 9,90	1x

Tabela 15 - Materiais para construção do protótipo.



Figura 30 - Maquete construída para a simulação do funcionamento do dispositivo.

5.3.2.4. Procedimentos experimentais

Conforme a estrutura mostrada na Figura 30, é possível realizar a simulação da elevação do nível d'água através da bomba presente no recipiente inferior, que armazena a água. Os sensores ICOS estão instalados no recipiente superior em posições definidas de acordo com a altura do recipiente. O sensor ultrassônico está instalado no topo do recipiente superior, onde pode fazer a leitura em tempo real da subida do nível d'água.

Através da instalação de dois sensores magnéticos ICOS, foi possível definir três diferentes faixas (ou estados) para o nível d'água. Estes estados foram definidos como: “1 - Nível Normal”; “2 - Nível de Alerta”; e “3 - Nível Crítico”. LEDs com diferentes cores foram utilizados para identificar em qual estado o nível se encontrava, baseado na leitura dos sensores ICOS.

Durante o desenvolvimento desse protótipo, novos componentes foram inseridos,

como componentes de comunicação sem fio e armazenamento de dados. Embora a implementação desses componentes ocorra apenas na versão final do protótipo, esse processo foi de extrema importância para determinar as melhores opções de integração.

5.3.2.5. Resultados

Os objetivos específicos definidos foram atingidos com sucesso e alertas foram emitidos de acordo com os estados definidos através da instalação dos sensores ICOS. Além disso, obteve-se a leitura do nível do recipiente em tempo real, através do sensor ultrassônico.

O desenvolvimento deste protótipo mostrou-se de extrema importância para o andamento do projeto. Através deste protótipo foi possível chegar à versão final da maquete e do dispositivo de monitoramento.

5.4. Desenvolvimento e validação do dispositivo de monitoramento

5.4.1. Introdução

Este experimento tem a finalidade de montar um protótipo utilizando um combinado com os sensores de nível ICOS e o sensor ultrassônico, visando dados concretos para o nível do rio, criando um dispositivo que apresente pequenos riscos à falhas. Além dos sensores mencionados, serão também incorporados ao protótipo, um componente de geolocalização, um componente de armazenamento de dados, um componente de comunicação sem fio por rádio frequência e um componente que realiza comunicação com a internet, através de redes Wi-Fi.

Este dispositivo, aliado com o *web service* desenvolvido também pelo projeto, atende todos os requisitos da arquitetura proposta e é a versão final, que colhe os dados no leito do rio e, baseado nesses dados, emite notificações de nível d'água em um aplicativo de celular, que tem por objetivo alertar a população sobre a iminência de enchentes.

5.4.2. Desenvolvimento

5.4.2.1. Objetivo geral

Projetar um dispositivo de hardware com capacidade de sensoriamento e comunicação sem fio que possa emitir alertas à população em situações de enchentes.

5.4.2.2. Objetivos específicos

- Detectar o nível d’água;
- Definir níveis de alerta;
- Detectar dados de geolocalização;
- Armazenamento de dados local;
- Comunicação sem fio por rádio frequência;
- Comunicação sem fio com a internet;
- Autossuficiente em energia;
- Emitir alerta de enchentes através de notificações em aplicativo de celular.

5.4.2.3. Maquete e materiais

Com o intuito de testar o dispositivo em laboratório, foi necessário a construção de uma maquete que simulasse o leito do Rio Jacaraípe. A maquete contou com um recipiente d’água, funcionando como reservatório; um recipiente em acrílico construído especificamente para ser encaixado no suporte da maquete; uma bomba d’água que simulasse a subida do nível d’água; e uma estrutura que tornasse a simulação possível, contendo canos PVC para a instalação dos sensores ICOS, duas caixas plásticas onde o circuito do dispositivo ficasse instalado e uma mesa servindo como suporte para todo o protótipo.

Além dos materiais para a construção da maquete e dos componentes que realizam a obtenção dos dados e comunicação sem fio, outros componentes eletrônicos foram adicionados ao protótipo, a fim de atingir todos os objetivos específicos. Entre esses componentes, estão placas solares, uma bateria, um módulo regulador de tensão e um módulo para carregamento da bateria através da energia solar captada pelos painéis solares.

O circuito construído foi soldado em uma placa de fenolite, a fim de reduzir o

tamanho do circuito e garantir o funcionamento do mesmo de forma consistente.

Tabela de materias e componentes:

Materiais	Preço	Quantidade	Total
Recipiente 10 litros	R\$ 9,99	1 unidade	R\$ 9,99
Recipiente de acrílico	R\$ 163,00	1 unidade	R\$ 163,00
Tubulações	R\$ 41,65	--	R\$ 41,65
Mesa (Madeiras)	R\$ 237,00	--	R\$ 237,00
Mesa (Gramá Artificial)	R\$ 76,00	--	R\$ 76,00
Caixa Plástica ABS	R\$ 16,90	2 unidades	R\$ 33,80
Bomba Submersa	R\$ 80,00	1 unidade	R\$ 80,00
Placa Arduino Mega 2560	R\$ 74,90	2 unidades	R\$ 149,80
Sensores ICOS	R\$ 29,90	2 unidades	R\$ 59,80
Sensor Ultrassônico	R\$ 9,90	1 unidade	R\$ 19,80
Módulo GPS	R\$ 119,00	1 unidade	R\$ 119,00
Módulo NRF24L01	R\$ 19,90	2 unidades	R\$ 39,80
Módulo Micro SD	R\$ 12,80	1 unidade	R\$ 12,80
Placa de Circuito Impresso	R\$ 12,90	1 unidade	R\$ 12,90
Módulo WiFi ESP8266 ESP-01	R\$ 26,90	1 unidade	R\$ 26,90
Display LCD 16×2	R\$ 16,90	1 unidade	R\$ 16,90
Bateria Li-Po Recarregável - 3,7V 1800mAh	R\$ 49,90	1 unidade	R\$ 49,90
Painel Solar Fotovoltaico 6V	R\$ 33,90	2 unidades	R\$ 67,80
Regulador de Tensão LM2596 DC Step Down	R\$ 11,90	1 unidade	R\$ 11,90
Lipo Rider / Carregador de Bateria Li-Ion	R\$ 99,80	1 unidade	R\$ 99,80
Total			R\$ 1.328,54

Tabela 16 - Materiais para construção da maquete e do dispositivo.



Figura 31 - Maquete finalizada com dispositivo instalado.

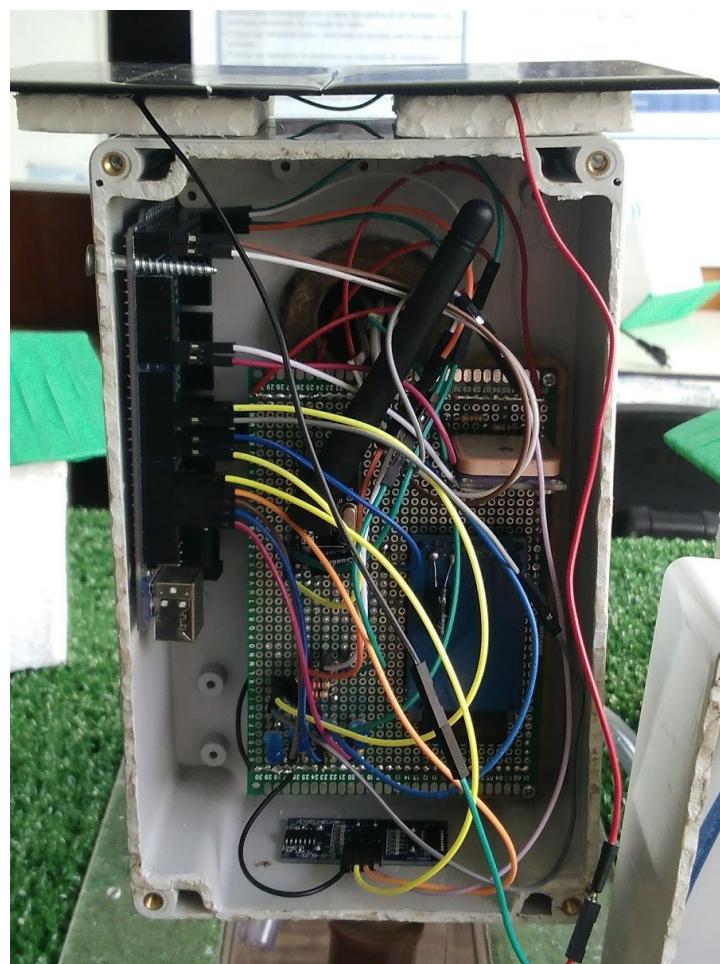


Figura 32 - Circuito soldado e instalado.

5.4.2.4. Referencial teórico

5.4.2.4.1. Sensores ICOS (LA16M-40)

Este sensor magnético indica através do sinal ON/OFF quando o nível de líquido for atingido. O LA16M-40 é instalado na linha horizontal e pode ser instalado de duas maneiras: normalmente aberto (NA), quando o sinal fornecido pelo sensor é OFF; e normalmente fechado (NF), quando o sinal fornecido pelo sensor é ON. A água alterna esses estados assim que atinge os sensores, possibilitando assim a leitura do nível d'água naquele ponto. É necessário instalar um sensor em cada ponto onde a leitura do nível é necessária. Esse protótipo faz uso de dois sensores, sendo o inferior instalado NA e o superior NF.

5.4.2.4.2. Sensor ultrassônico

O sensor ultrassônico calcula a distância entre o sensor e um objeto. Seu funcionamento consiste basicamente em enviar um sinal que ao atingir um objeto é espelhado de volta para o sensor e com base nesse tempo entre o envio e recebimento, é calculada a distância. É enviado um sinal com duração de 10 ms (microsegundos) ao sensor, indicando que a medição terá início, automaticamente, o *trigger* envia 8 pulsos de 40 KHz e aguarda o retorno do sinal pelo receptor (*echo*), e caso haja um retorno de sinal (em nível HIGH), determinamos a distância entre o sensor e o obstáculo utilizando a seguinte equação: distância = (pulso em nível alto x velocidade do som (340m/s) / 2 ($D = V * T$). A divisão por 2 é necessária já que estamos contando o tempo de ida e volta do sinal.

A implementação do código para este componente foi realizada utilizando a biblioteca *NewPing*. Esta biblioteca realiza de maneira eficiente o disparo do sinal *trigger* e o cálculo da distância medida. Além disso, fornece funções de conversão entre diferentes unidades de distância, facilitando o desenvolvimento do código Arduino.

5.4.2.4.3. Módulo NRF24L01+ - Comunicação sem fio por rádio frequência

É um componente eletrônico, utilizado principalmente para enviar e receber informações *wireless* (sem fio) entre dispositivos, através de radio frequência. Ele funciona com alimentação de 1,9V a 3,6V, possui uma velocidade de comunicação de 2Mbps e tem alcance de até 1100 metros, fazendo o uso de uma antena embutida.

Esse módulo pode funcionar como emissor ou como receptor. Quando operando como receptor, este módulo pode receber dados de até outros seis dispositivos diferentes, fazendo uso da biblioteca RF24.

Foi necessária a soldagem de um capacitor de $100\text{ }\mu\text{F}$ em os pinos VCC e GND do módulo, a fim de garantir o funcionamento de forma consistente do mesmo. Tal capacitor é chamado de capacitor de desacoplamento e tem a funcionalidade de filtrar as irregularidades elétricas que estavam causando o mal funcionamento do módulo.



Figura 33 - Módulo NRF24L01+ e sua pinagem.

5.4.2.4.4. Módulo GPS GY-NEO6MV2 - Geolocalização

Componente capaz de fornecer dados de geolocalização, utilizando as coordenadas geográficas latitude e longitude. Além disso, este módulo pode fornecer diversos outros dados, como data, hora, elevação, velocidade, curso, entre outros.

O GY-NEO6MV2 é um módulo de fácil utilização, realizando a comunicação através de comunicação serial, com taxa padrão de 9600. Esse tipo de comunicação faz com que o módulo seja facilmente integrado à projetos que utilizam microcontroladores e também placas como Raspberry Pi, Linkit, Beaglebone, NodeMCU, ESP8266 e outras. O módulo trabalha com alimentação de 2.7V a 5V e corrente de 45mA. Possui antena embutida e uma bateria para backup de dados. Possui precisão de 5m.

A biblioteca TinyGPS++ foi utilizada juntamente com este módulo para extração dos dados de sentenças NMEA. Esta biblioteca possui métodos simples de extração e validação de dados, algo que mostrou-se extremamente útil para facilitar e agilizar o processo de implementação do módulo.



Figura 34 - Módulo GPS GY-NEO6MV2.

5.4.2.4.5. Módulo Cartão SD - Armazenamento local de dados

Esse módulo permite a leitura e escrita em cartão SD, com fácil ligação ao Arduino e outros microcontroladores. Ele funciona com alimentação de 3.3V ou 5V, e se comunica através do sistema de arquivos e do driver de interface SPI. O módulo suporta formatos de arquivo FAT16 e FAT32. Fez-se uso da biblioteca SdFat para auxiliar na implementação do módulo.

Foi necessária a soldagem de um capacitor de $100\ \mu F$ em os pinos VCC e GND do módulo, a fim de garantir o funcionamento de forma consistente do mesmo. Tal capacitor é chamado de capacitor de desacoplamento e tem a funcionalidade de filtrar as irregularidades elétricas que estavam causando o mal funcionamento do módulo.

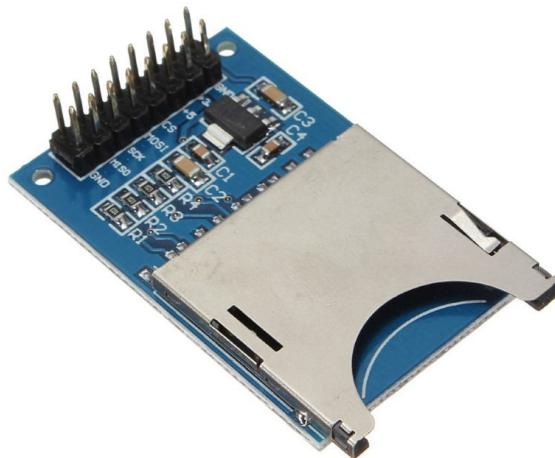


Figura 35 - Módulo para cartão SD.

5.4.2.4.6. Módulo Wifi ESP8266 ESP-01 - Comunicação sem fio com a internet

É um *System-On-Chip* com Wifi embutido. Seu CPU que opera em 80 MHz, com possibilidade de operar em 160 MHz. Ele tem como principal função conectar o microcontrolador a uma conexão Wi-fi de forma fácil e eficaz.

O módulo suporta as redes 802.11 b/g/n, muito usadas atualmente, podendo trabalhar como um Ponto de Acesso (Access Point) ou como uma Estação (Station), enviando e recebendo dados. A comunicação do módulo com o Arduino pode ser feita via serial

utilizando os pinos RX e TX, configurada através de comandos AT.

O módulo possui alcance de até 90 metros e opera em tensão de 3,3V.

O módulo foi implementado com o suporte da biblioteca weeESP8266, que possui métodos de configuração do módulo e conexão com redes Wifi, com o objetivo de facilitar e agilizar o processo de implementação do mesmo.

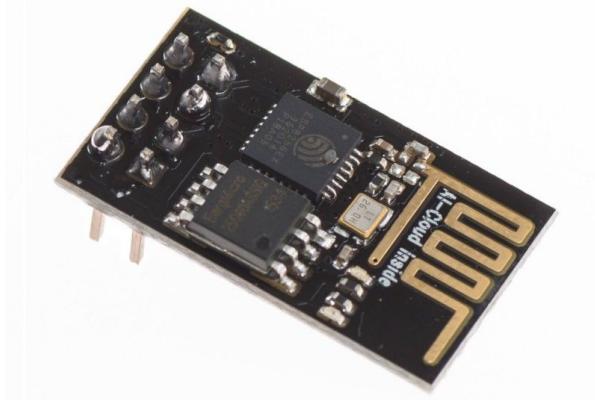


Figura 36 - Módulo WiFi ESP8266 ESP-01.

5.4.3. Procedimentos experimentais

5.4.3.1. Sensor node

A Figura 37 mostra todos os componentes utilizados no dispositivo de monitoramento desenvolvido. O dispositivo mostrado é identificado na arquitetura proposta como *sensor node*. Este dispositivo realiza a leitura do nível d'água no rio, através de ambos os sensores ICOS e do sensor ultrassônico e obtém os dados de geolocalização (latitude, longitude, data, hora e elevação), através do módulo GPS. Estes dados são armazenados em uma *struct* de dados.

A fim de manter registros de todos os dados coletados, faz-se o uso do cartão SD para gravar os dados obtidos. Um arquivo .csv, em formato de planilha, é criado no cartão SD presente no dispositivo. A cada novo dado válido obtido, este arquivo é atualizado e o novo dado é gravado. Tem-se assim um *log* dos dados.

Todos os dados supracitados são também enviados a outro dispositivo através do

módulo NRF24L01. Assim tem-se a aplicação de uma rede de sensores sem fio.

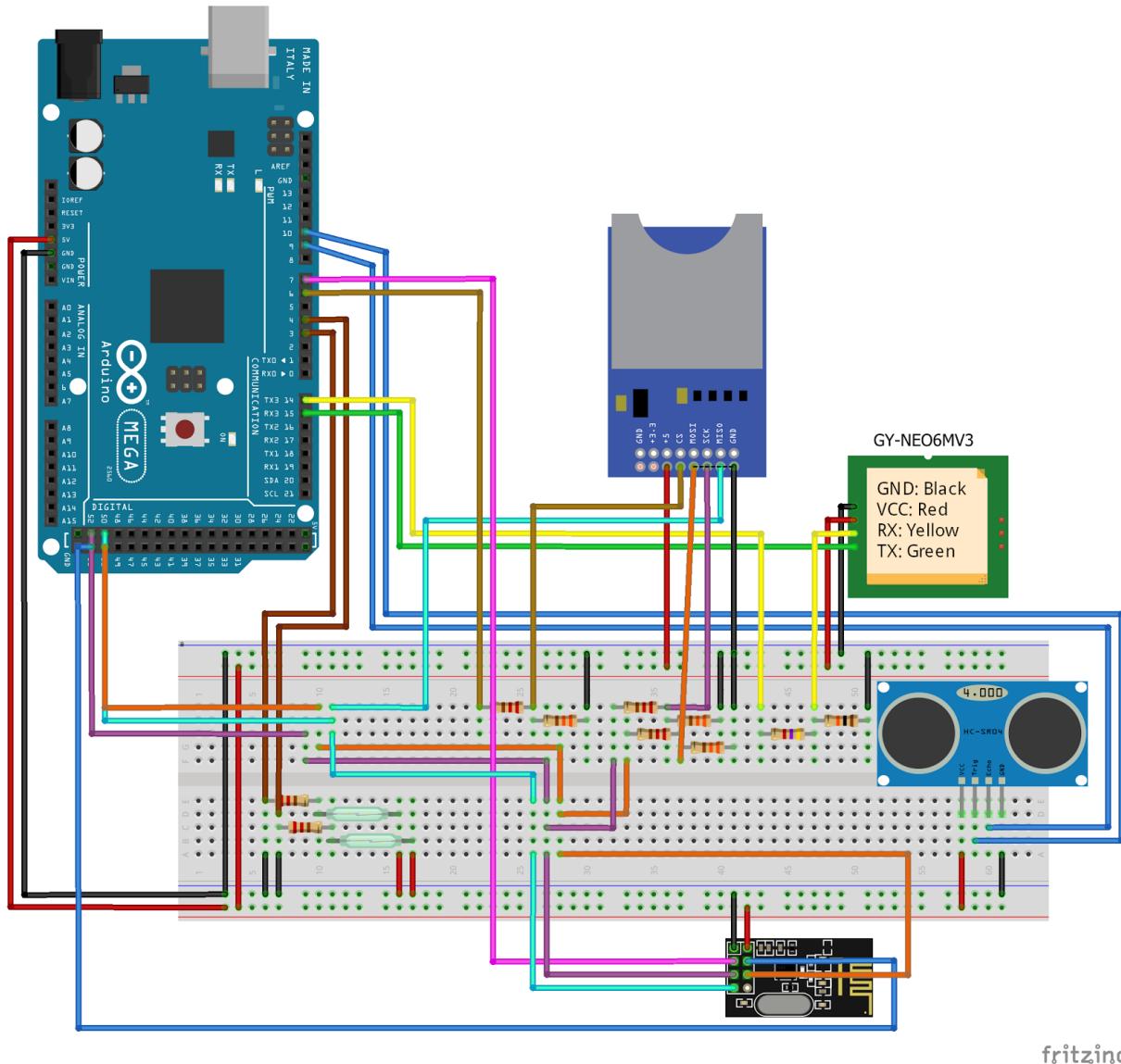


Figura 37 - Circuito sensor node.

Durante todo o processo de desenvolvimento do dispositivo, realizou-se teste de unidade, testes de integração e testes de sistema. O código produzido para o dispositivo pode ser observado logo abaixo.

```
#define DEBUG

#include <SPI.h> // SPI
#include <NewPing.h> // Sensor ultrassônico
#include <TinyGPS++.h> // Módulo GPS
#include <SD.h> // Módulo Cartão SD
#include <nRF24L01.h> // Módulo Com RF - RF24
#include <RF24.h> // Módulo Com RF - RF24

// definindo as constantes do programa
// ID DO DISPOSITIVO
const int ID_dispositivo = 1; // ID do pacote enviado

// LEDs de status
// LED_1 - Informa se dispositivo está alimentado
// LED_2 - Indica falha no módulo SD (Ligado - Erro Inicialização / 300ms ON e
// 2000ms OFF - erro no log de dados)
// LED_3 - Indica falha no módulo GPS
// LED_4 - Indica falha no módulo RF24
const int LED_2 = 11;
const int LED_3 = 12;
const int LED_4 = 13;

// SENSORES ICOS
// Identificação Sensor Superior - Azul e Branco
// Identificação Sensor Inferior - Verde e Marrom
const int S1 = 4; // pinos sensores ICOS - sensor inferior
const int S2 = 3; // sensor superior

// SENSOR ULTRASSÔNICO
const int ECHO = 9; // pinos sensor ultrassônico
const int TRIG = 10;
const float REF = 35.0; // referência de altura de instalação do sensor
ultrassônico

// MÓDULO GPS
static const uint32_t GPSB = 9600; // definindo a velocidade de comunicação do
módulo GPS

// MÓDULO RF24
const int CEpin = 7;
const int CSpin = 53;

// MÓDULO SD CARD
const int pinSD = 6;
```

```

// nome dos arquivos
#define LOG_FILE_PREFIX "log"
#define MAX_LOG_FILES 100
#define LOG_FILE_SUFFIX "csv"
char logNomeArquivo[10];

// colunas do arquivo
#define LOG_COLUMN_COUNT 11
char* logColunas[LOG_COLUMN_COUNT] = {
    "ID", "ICOS Inferior", "ICOS Superior", "Nível Ultrassônico", "Latitude",
    "Longitude", "Elevação", "Data", "Hora", "Minuto", "Segundo"
};

// controle de rate de log
const unsigned long LOG_RATE = 3000;
unsigned long ultimoLog = 0;

// inicialização dos objetos
SdFat sdCard;
TinyGPSPPlus gps;
NewPing us(TRIG, ECHO);
RF24 radio(CEPin, CSpin);
const uint64_t pipe = 0xF0F0F0F0E1LL;

// DEFININDO STRUCT DE DADOS
// struct com dados dos sensores
typedef struct{
    int ID;          // ID do dispositivo - inteiro tamanho variável (3 máxim
    int ICOS_INF;    // sensor ICOS inferior - inteiro tamanho 1
    int ICOS_SUP;    // sensor ICOS superior - inteiro tamanho 1
    float NIVEL;     // ultrassônico - float com 2 casas decimais (cm)
    bool LAT_NEG;    // booleano para latitude negativa
    int LAT_DEG;     // graus da latitude - inteiro tamanho 2
    uint16_t LAT_BILLIONTHS; // decimal da latitude - inteiro tamanho 32
    bool LNG_NEG;    // booleano para longitude negativa
    int LNG_DEG;     // graus da longitude - inteiro tamanho 2
    uint16_t LNG_BILLIONTHS; // decimal da longitude - inteiro tamanho 32
    double ELEVACAO; // elevacao - double
    uint16_t ANO;     // ano - inteiro tamanho 4
    uint8_t MES;      // mês - inteiro tamanho 2
    uint8_t DIA;      // dia - inteiro tamanho 2
    uint8_t HORA;     // hora - inteiro tamanho 2
    uint8_t MINUTO;   // minuto - inteiro tamanho 2
    uint8_t SEGUNDO;  // segundo - inteiro tamanho 2
}
S_t;

// declarando as structs que armazenarão os dados
S_t dadosSensores;

```

```

void setup()
{
#define DEBUG
    Serial.begin(57600); // comunicação Serial com o computador

    Serial.println(F("\r\n----- PICJr - MOPPE -----"));
    Serial.println(F("----- Programa inicializado ----- \r\n"));
    Serial.println(F("Iniciando Setup..."));

#endif

// Define PIN Mode LEDs
pinMode(LED_2, OUTPUT);
pinMode(LED_3, OUTPUT);
pinMode(LED_4, OUTPUT);

digitalWrite(LED_2, LOW);
digitalWrite(LED_3, LOW);
digitalWrite(LED_4, LOW);

// módulo GPS
Serial3.begin(GPSB);

// define os sensores ICOS como input
pinMode(S1, INPUT);
pinMode(S2, INPUT);

// Inicializa o módulo SD
if(!sdCard.begin(pinSD, SD_SCK_MHZ(50))){
    digitalWrite(LED_2, HIGH);

#define DEBUG
    sdCard.initErrorHalt();
#endif
}

// cria novo arquivo a cada inicialização
updateFileName(); // cria um novo arquivo a cada inicialização
printHeader(); // imprime o cabeçalho do arquivo

#define DEBUG
    Serial.println(F("Setup Finalizado!"));
#endif

// RF24
radio.begin();
radio.openWritingPipe(pipe);
} // fecha void setup()

```

```

void loop()
{
    // obtenção dos dados dos sensores
    dadosSensores.ID      = ID_dispositivo;
    dadosSensores.ICOS_INF = digitalRead(S1); // leitura do sensor ICOS inferior
    dadosSensores.ICOS_SUP = digitalRead(S2); // leitura do sensor ICOS superior
    dadosSensores.NIVEL    = dados_su();        // sensor ultrassônico

    dados_gps(); // alimenta o objeto gps

    // alimenta struct
    dadosSensores.LAT_NEG = gps.location.rawLat().negative;
    dadosSensores.LAT_DEG = gps.location.rawLat().deg;
    dadosSensores.LAT_BILLIONTHS = gps.location.rawLat().billions;
    dadosSensores.LNG_NEG = gps.location.rawLng().negative;
    dadosSensores.LNG_DEG = gps.location.rawLng().deg;
    dadosSensores.LNG_BILLIONTHS = gps.location.rawLng().billions;
    dadosSensores.ELEVACAO = gps.altitude.meters();
    dadosSensores.ANO = gps.date.year();
    dadosSensores.MES = gps.date.month();
    dadosSensores.DIA = gps.date.day();
    dadosSensores.HORA = gps.time.hour();
    dadosSensores.MINUTO = gps.time.minute();
    dadosSensores.SEGUNDO = gps.time.second();

    // DEBUG
    #ifdef DEBUG
        Serial.println(dadosSensores.ID);
        Serial.println(dadosSensores.ICOS_INF);
        Serial.println(dadosSensores.ICOS_SUP);
        Serial.println(dadosSensores.NIVEL);
        Serial.println(dadosSensores.LAT_NEG);
        Serial.println(dadosSensores.LAT_DEG);
        Serial.println(dadosSensores.LAT_BILLIONTHS);
        Serial.println(dadosSensores.LNG_NEG);
        Serial.println(dadosSensores.LNG_DEG);
        Serial.println(dadosSensores.LNG_BILLIONTHS);
        Serial.println(dadosSensores.ELEVACAO);
        Serial.println(dadosSensores.ANO);
        Serial.println(dadosSensores.MES);
        Serial.println(dadosSensores.DIA);
        Serial.println(dadosSensores.HORA);
        Serial.println(dadosSensores.MINUTO);
        Serial.println(dadosSensores.SEGUNDO);
        Serial.println();
    #endif
}

```

```

if((ultimoLog + LOG_RATE) <= millis()) // Se LOG_RATE em milissegundos desde
o último registro
{
    if(gps.location.isValid())
    {
        if(logData(dadosSensores)) // Registrar dados no cartão SD
        {
            #ifdef DEBUG
                Serial.println("DADOS LOGADOS"); //mostrar essa mensagem
            #endif
            ultimoLog = millis(); // atualizar a variável
        }
        else // se não atualizou
        {
            digitalWrite(LED_2, HIGH);
            delay(300);
            digitalWrite(LED_2, LOW);
            delay(1000);

            #ifdef DEBUG
                Serial.println("Falha no log de dados."); // será mostrado uma
mensagem de erro no GPS
            #endif
        }
    }

    // ENVIAR DADOS AO SINK NODE
    bool ok = radio.write(&dadosSensores, sizeof(dadosSensores));
    if(ok) {
        #ifdef DEBUG
            Serial.println(F("DADOS ENVIADOS"));
        #endif
    } else {
        digitalWrite(LED_4, HIGH);
        delay(300);
        digitalWrite(LED_4, LOW);
        delay(1000);

        #ifdef DEBUG
            Serial.println(F("FALHA NO ENVIO DOS DADOS"));
        #endif
    }
}
else // dados do GPS não válidos
{
    digitalWrite(LED_3, HIGH);
    delay(300);
    digitalWrite(LED_3, LOW);
    delay(1000);

    #ifdef DEBUG
        Serial.println("Dados de GPS invalidos. Buscando...");
```

```

        #endif
    }
}
} // fecha void loop()

// obtendo dados do sensor ultrassônico
float dados_su(){
    float dist = us.ping_cm();
    float nivel = REF - dist;
    return nivel;
}

// alimentando o objeto gps com dados do módulo GPS
void dados_gps(){
    while(Serial3.available())
        gps.encode(Serial3.read());
}

void updateFileName(){

for(int i=0; i < MAX_LOG_FILES; i++){
    memset(logNomeArquivo, 0, strlen(logNomeArquivo));
    sprintf(logNomeArquivo, "%d.%s", LOG_FILE_PREFIX, i, LOG_FILE_SUFFIX);

    if (!sdCard.exists(logNomeArquivo))
        break;
    else{
        #ifdef DEBUG
            Serial.print(logNomeArquivo);
            Serial.println(" existe!");
        #endif
    }
}

#ifndef DEBUG
    Serial.print("Nome do Arquivo: ");
    Serial.println(logNomeArquivo);
#endif
}

void printHeader(){

SdFile logfile;
bool sd_open = logfile.open(logNomeArquivo, O_RDWR | O_CREAT | O_AT_END);

if(sd_open){
    for(int i=0; i < LOG_COLUMN_COUNT; i++){
        logfile.print(logColunas[i]);
        if(i < (LOG_COLUMN_COUNT -1))
            logfile.print(',');
    }
}
}

```

```

        logfile.println();
    }
    logfile.close();
}
else{
    digitalWrite(LED_2, HIGH);
    delay(300);
    digitalWrite(LED_2, LOW);
    delay(2000);

#ifndef DEBUG
    Serial.println("\r\nErro na abertura do arquivo (HEADER)!");
#endif
}

byte logData(S_t dadosSensores)
{
    SdFile logfile;
    bool sd_open = logfile.open(logNomeArquivo, O_RDWR | O_CREAT | O_AT_END);

    // grava os dados no arquivo do cartão SD
    if(sd_open){
        logfile.print(dadosSensores.ID);
        logfile.print(',');
        logfile.print(dadosSensores.ICOS_INF);
        logfile.print(',');
        logfile.print(dadosSensores.ICOS_SUP);
        logfile.print(',');
        logfile.print(dadosSensores.NIVEL);
        logfile.print(',');
        logfile.print(dadosSensores.LAT_NEG ? "-" : "+");
        logfile.print(dadosSensores.LAT_DEG);
        logfile.print(".");
        logfile.print(dadosSensores.LAT_BILLIONTHS);
        logfile.print(',');
        logfile.print(dadosSensores.LNG_NEG ? "-" : "+");
        logfile.print(dadosSensores.LNG_DEG);
        logfile.print(".");
        logfile.print(dadosSensores.LNG_BILLIONTHS);
        logfile.print(',');
        logfile.print(dadosSensores.ELEVACAO);
        logfile.print(',');

        if(dadosSensores.DIA < 10)
            logfile.print("0");

        logfile.print(dadosSensores.DIA);
        logfile.print("/");

        if(dadosSensores.MES < 10)

```

```
    logfile.print("0");

    logfile.print(dadosSensores.MES);
    logfile.print("/");
    logfile.print(dadosSensores.ANO);
    logfile.print(',');

    if(dadosSensores.HORA < 10)
        logfile.print("0");

    logfile.print(dadosSensores.HORA);
    logfile.print(',');

    if(dadosSensores.MINUTO < 10)
        logfile.print("0");

    logfile.print(dadosSensores.MINUTO);
    logfile.print(',');

    if(dadosSensores.SEGUNDO < 10)
        logfile.print("0");

    logfile.print(dadosSensores.SEGUNDO);
    logfile.println();
    logfile.close();
    return 1;
}

else{
    digitalWrite(LED_2, HIGH);
    delay(300);
    digitalWrite(LED_2, LOW);
    delay(3000);

#define DEBUG
    Serial.println("\r\nErro na abertura do arquivo! (DADOS)");
#endif

    return 0;
}
}
```

5.4.3.2. Sink node

A Figura 38 mostra o dispositivo identificado na arquitetura proposta como *sink node*.

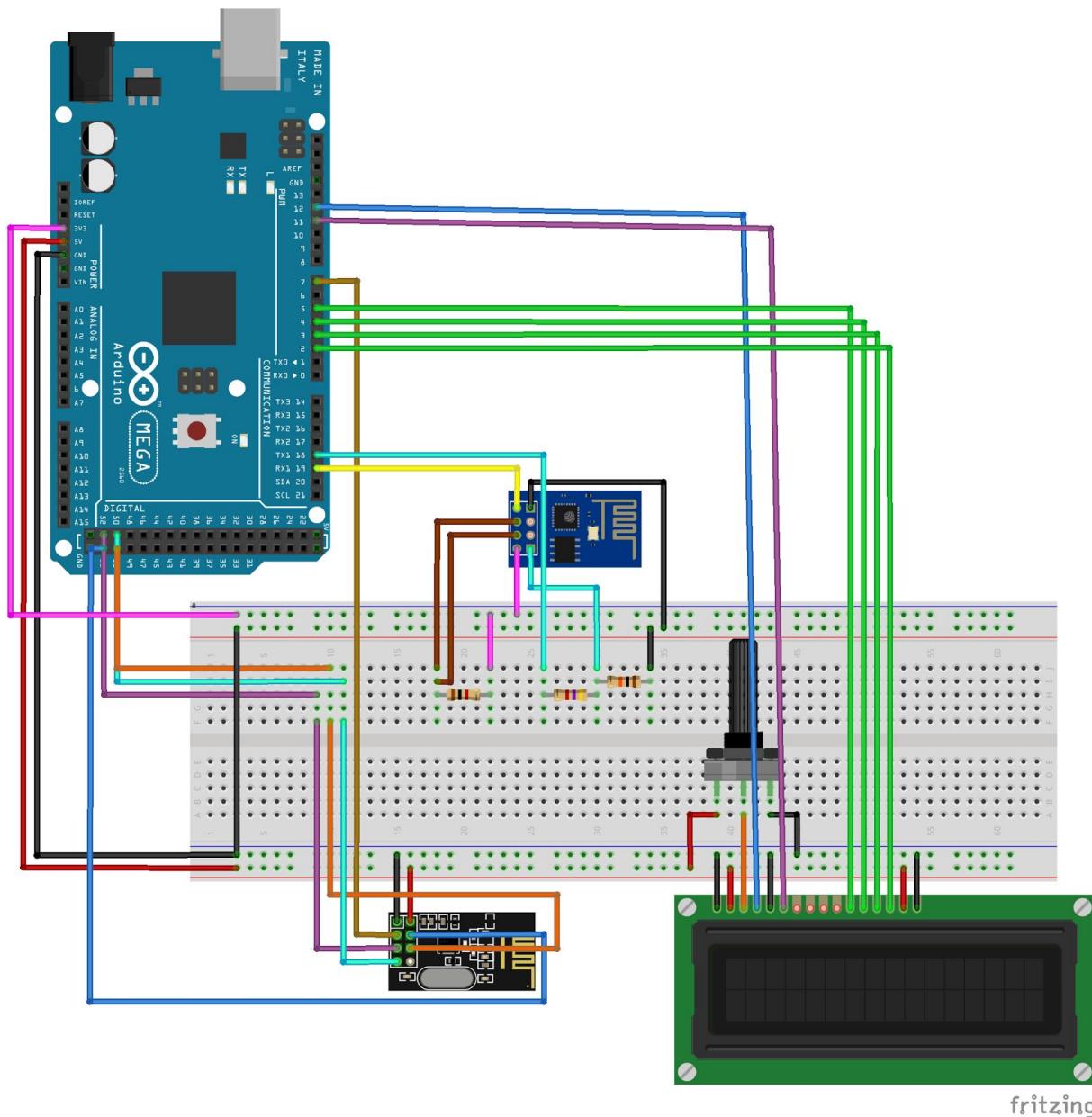


Figura 38 - Circuito sink node.

Fez-se a escolha no desenvolvimento deste dispositivo, apenas para demonstração, de criar um circuito, com o módulo NRF24L01 atuando como receptor, o display LCD exibindo a leitura do sensor ultrassônico e o estado do rio e do componente ESP8266, responsável pela comunicação com a internet. Porém, na realidade de implementação do dispositivo, o *sink node* também possui todos os outros componentes de leitura de nível, dados de geolocalização e armazenamento local de dados. O que difere o *sink node* é a adição de um módulo que possa se comunicar com a internet.

O dispositivo aqui apresentado recebe a *struct* de dados enviada pelo *sensor node* através do módulo NRF24L01 e exibe no display LCD qual a leitura de nível obtida pelo sensor ultrassônico e o estado do rio, baseado na leitura dos sensores ICOS. Os estados podem ser “ESTADO NORMAL”, “ESTADO DE ALERTA” e “ESTADO DE EMERGÊNCIA”.

O dispositivo também realiza o envio desses dados para um servidor na internet, através do método GET. Todos os dados enviados ao servidor são armazenados em um banco de dados relacional MySQL. Esses dados serão então usados pelo *web service* desenvolvido, que irá gerar notificações no aplicativo de celular desenvolvido, alertando a população da situação do nível do rio, em caso de enchentes iminentes.

O código produzido para o dispositivo pode ser observado logo abaixo.

```
#define DEBUG

#include <SPI.h>                      // SPI
#include <nRF24L01.h>                    // RF24
#include <RF24.h>
#include <ESP8266.h> // biblioteca do módulo Wi-Fi
#include <LiquidCrystal.h> // biblioteca para o display LCD

// inicialização do módulo RF24
const int CEPin = 7;
const int CSpin = 53;
RF24 radio(CEPin, CSpin);
const uint64_t pipe = 0xF0F0F0F0E1LL; // tunel de conexão

// constantes de conexão
#define SSID      "rede-wifi"
#define PASSWORD  "senha-rede-wifi"
#define HOST_NAME "servidor.com"
#define HOST_PORT (80)

// inicialização do módulo Wi-Fi
ESP8266 wifi(Serial1, 115200);
```

```

// inicialização do módulo LCD
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

// DEFININDO STRUCT DE DADOS
// struct dados sensores
typedef struct{
    int ID;
    int ICOS_INF;
    int ICOS_SUP;
    float NIVEL;
    bool LAT_NEG;
    int LAT_DEG;
    uint16_t LAT_BILLIONTHS;
    bool LNG_NEG;
    int LNG_DEG;
    uint16_t LNG_BILLIONTHS;
    double ELEVACAO;
    uint16_t ANO;
    uint8_t MES;
    uint8_t DIA;
    uint8_t HORA;
    uint8_t MINUTO;
    uint8_t SEGUNDO;
}
S_t;

// declarando as structs que armazenarão os dados
S_t dadosSensores;

void setup()
{
#define DEBUG
    Serial.begin(57600); // comunicação Serial com o computador
    Serial.println(F("\r\n----- PICJr - MOPPE -----"));
    Serial.println(F("\r\nIniciando Setup..."));
#define ENDIF

    //Define o número de colunas e linhas do LCD
    lcd.begin(16, 2);

    // Imprime mensagem de inicialização
    lcd.clear();
    lcd.setCursor(2, 0);
    lcd.print("PICJr MOPPE");
    lcd.setCursor(1, 1);
    lcd.print("Aguardando...");

    // inicializa módulo RF24
    radio.begin();
    radio.openReadingPipe(1,pipe);
}

```

```

radio.startListening();

// seta modo de operação como estação
#ifndef DEBUG
    if (wifi.setOprToStationSoftAP()) {
        Serial.println(F("para estacao + softap OK"));
    } else {
        Serial.println(F("para estacao + softap ERROR"));
    }
#else
    wifi.setOprToStationSoftAP();
#endif

// conecta a rede Wi-Fi especificada
#ifndef DEBUG
    if(wifi.joinAP(SSID, PASSWORD)) {
        Serial.println(F("conexao OK"));
        Serial.print(F("IP:"));
        Serial.println(wifi.getLocalIP().c_str());
    } else {
        Serial.println(F("conexao ERROR"));
    }
#else
    wifi.joinAP(SSID, PASSWORD);
#endif

// modo de conexao única
#ifndef DEBUG
    if (wifi.disableMUX()) {
        Serial.println(F("conexao unica OK"));
    } else {
        Serial.println(F("conexao unica ERROR"));
    }
#else
    wifi.disableMUX();
#endif

#ifndef DEBUG
    Serial.println(F("Setup Finalizado!"));
#endif
} // fecha void setup()

// envia dados GET
void loop()
{
    if(radio.available()){
        bool ok = false;

        while(!ok){
            ok = radio.read(&dadosSensores, sizeof(dadosSensores));
        }
    }
}

```

```

if(ok) {
    exibeNoLCD();
    enviaDados();

#define DEBUG
    Serial.print("dadosSensores.ID = ");
    Serial.println(dadosSensores.ID);
    Serial.print("dadosSensores.ICOS_INF = ");
    Serial.println(dadosSensores.ICOS_INF);
    Serial.print("dadosSensores.ICOS_SUP = ");
    Serial.println(dadosSensores.ICOS_SUP);
    Serial.print("dadosSensores.NIVEL = ");
    Serial.println(dadosSensores.NIVEL);
    Serial.print("dadosSensores.LAT_NEG = ");
    Serial.println(dadosSensores.LAT_NEG);
    Serial.print("dadosSensores.LAT_DEG = ");
    Serial.println(dadosSensores.LAT_DEG);
    Serial.print("dadosSensores.LAT_BILLIONTHS = ");
    Serial.println(dadosSensores.LAT_BILLIONTHS);
    Serial.print("dadosSensores.LNG_NEG = ");
    Serial.println(dadosSensores.LNG_NEG);
    Serial.print("dadosSensores.LNG_DEG = ");
    Serial.println(dadosSensores.LNG_DEG);
    Serial.print("dadosSensores.LNG_BILLIONTHS = ");
    Serial.println(dadosSensores.LNG_BILLIONTHS);
    Serial.print("dadosSensores.ELEVACAO = ");
    Serial.println(dadosSensores.ELEVACAO);
    Serial.print("dadosSensores.ANO = ");
    Serial.println(dadosSensores.ANO);
    Serial.print("dadosSensores.MES = ");
    Serial.println(dadosSensores.MES);
    Serial.print("dadosSensores.DIA = ");
    Serial.println(dadosSensores.DIA);
    Serial.print("dadosSensores.HORA = ");
    Serial.println(dadosSensores.HORA);
    Serial.print("dadosSensores.MINUTO = ");
    Serial.println(dadosSensores.MINUTO);
    Serial.print("dadosSensores.SEGUNDO = ");
    Serial.println(dadosSensores.SEGUNDO);
    Serial.println();
#endif
}
}

else {
    // caso nao existam mensagens
#ifndef DEBUG
    //Serial.println(F("NAO HA MENSAGENS DISPONIVEIS"));
#endif
}
}

```

```

String floatToString(float x, byte precision = 2) {
    char tmp[50];
    dtostrf(x, 0, precision, tmp);
    return String(tmp);
}

void exibeNoLCD()
{
    //Limpa a tela
    lcd.clear();
    //Posiciona o cursor na coluna 3, linha 0;
    lcd.setCursor(0, 0);
    //Envia o texto entre aspas para o LCD
    if(dadosSensores.ICOS_INF == 1 && dadosSensores.ICOS_SUP == 0) {
        lcd.print("NIVEL NORMAL");
    } else if(dadosSensores.ICOS_INF == 0 && dadosSensores.ICOS_SUP == 0) {
        lcd.print("NIVEL ALERTA");
    } else if(dadosSensores.ICOS_INF == 0 && dadosSensores.ICOS_SUP == 1) {
        lcd.print("NIVEL CRITICO");
    }
    else {
        lcd.print("INDETERMINADO");
    }
    lcd.setCursor(0, 1);
    lcd.print("NIVEL: ");
    lcd.setCursor(8, 1);
    lcd.print(dadosSensores.NIVEL);
}

void enviaDados()
{
    // cria conexao TCP com servidor
    #ifdef DEBUG
        if (wifi.createTCP(HOST_NAME, HOST_PORT)) {
            Serial.println(F("servidor tcp OK"));
        } else {
            Serial.println(F("servidor tcp ERROR"));
        }
    #else
        wifi.createTCP(HOST_NAME, HOST_PORT);
    #endif

    // monta o path do GET
    String elevacao_str = floatToString(dadosSensores.ELEVACAO);
    String nivel_str    = floatToString(dadosSensores.NIVEL);

    // monsta requisição Http GET
    String temp_request = "GET /moppe-ws/public/index.php/get_insere/";
    temp_request.concat(dadosSensores.ID);
    temp_request.concat("/");
}

```

```

temp_request.concat(dadosSensores.ICOS_INF);
temp_request.concat("/");
temp_request.concat(dadosSensores.ICOS_SUP);
temp_request.concat("/");
temp_request.concat(nivel_str);
temp_request.concat("/");
temp_request.concat(dadosSensores.LAT_NEG);
temp_request.concat("/");
temp_request.concat(dadosSensores.LAT_DEG);
temp_request.concat("/");
temp_request.concat(dadosSensores.LAT_BILLIONTHS);
temp_request.concat("/");
temp_request.concat(dadosSensores.LNG_NEG);
temp_request.concat("/");
temp_request.concat(dadosSensores.LNG_DEG);
temp_request.concat("/");
temp_request.concat(dadosSensores.LNG_BILLIONTHS);
temp_request.concat("/");
temp_request.concat(elevacao_str);
temp_request.concat("/");
temp_request.concat(dadosSensores.ANO);
temp_request.concat("/");
temp_request.concat(dadosSensores.MES);
temp_request.concat("/");
temp_request.concat(dadosSensores.DIA);
temp_request.concat("/");
temp_request.concat(dadosSensores.HORA);
temp_request.concat("/");
temp_request.concat(dadosSensores.MINUTO);
temp_request.concat("/");
temp_request.concat(dadosSensores.SEGUNDO);
temp_request.concat(" HTTP/1.1\r\nHost: ");
temp_request.concat(HOST_NAME);
temp_request.concat("\r\nUser-Agent: Arduino\r\nConnection: close\r\n\r\n");

// converte a String para char array
char httpGET[300];
temp_request.toCharArray(httpGET, 300);

// realiza a conexao com o servidor e envia os dados
#ifndef DEBUG
if (wifi.send((const uint8_t*)httpGET, strlen(httpGET))) {
    Serial.println(F("Dados enviados ao servidor!"));
} else {
    Serial.println(F("FALHA AO ENVIAR AO SERVIDOR"));
}
#else
    wifi.send((const uint8_t*)httpGET, strlen(httpGET));
#endif
}

```

5.4.4. Web Service para persistência de dados de dispositivos de sensoriamento permitindo acesso por aplicativos híbridos

5.4.4.1. Introdução

Para que os dispositivos funcionem de forma correta e os dados obtidos pelos sensores possam ser levados de forma mais rápida e precisa para a população e órgãos, se tornou necessário a criação de um sistema de informação por trás desses dispositivos. Esse sistema deveria ser capaz de unir, analisar e disponibilizar os dados obtidos em tempo real, permitir a inserção de novos dados e por fim mostrar um histórico com todos os dados já reunidos em todos os dispositivos, fazendo assim, o monitoramento da bacia. A infraestrutura do sistema pode ser observada na Figura 39.

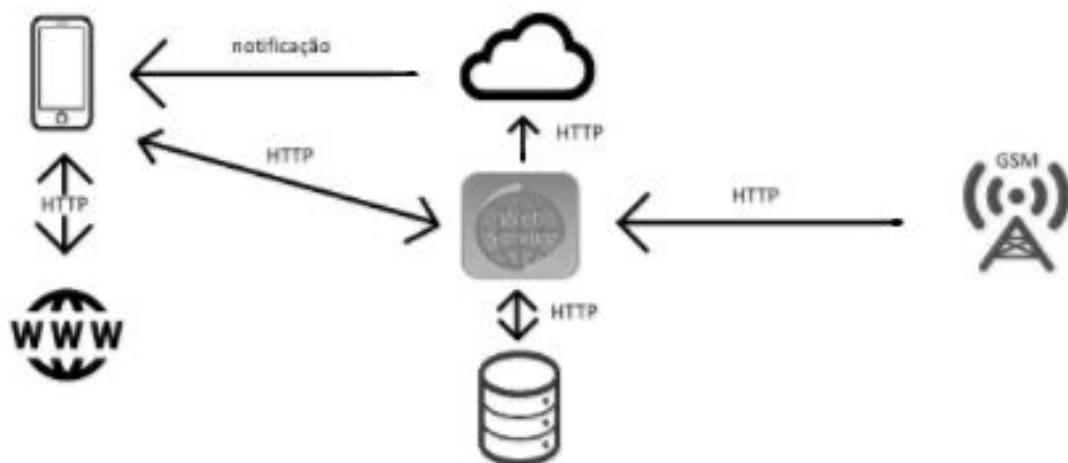


Figura 39 - Infraestrutura do web service.

- A antena representa o dispositivo que envia os dados através de um módulo GSM/Wifi conectado à internet;
- O cilindro representa o banco de dados;
- O símbolo no centro representa o *web service*;
- A nuvem representa a plataforma que envia notificações, nesse caso o *OneSignal*.
- O celular representa os dispositivos *mobile*;
- O *WWW* representa às páginas *web* que o aplicativo mobile acessa.

5.4.4.2. Web Service, Aplicação Web e Banco de Dados

Parte responsável por receber os dados gerados pelos dispositivos, gerar solicitações de notificações e fazer o histórico dos dados. O *web service* recebe os dados, exibe os dados (uma pequena aplicação web), verifica a necessidade de notificação aos aplicativos e persiste os dados no banco de dados.

Antes de começar o desenvolvimento deste serviço, primeiramente foi feita uma análise de quais ferramentas poderiam ser usadas e quais atenderiam melhor os dispositivos desenvolvidos. As ferramentas foram avaliadas nos seguintes critérios: dificuldade de implementação, custo, desempenho e dificuldade didática (por se tratar de um Projeto de Iniciação Científica Júnior - PIC JR, isso se torna importante).

As ferramentas definidas foram:

- Servidor Web Apache: Simples, sem custo, rápido e com recursos úteis;
- O banco de dados MySQL: fácil integração com as outras ferramentas, sem custo e ótimo desempenho;
- A linguagem de programação PHP (utilizando o framework Slim): de fácil implementação, sem custo e alto desempenho;

Após a análise, as ferramentas foram estudadas e em seguida passaram a ser implementadas. Primeira ferramenta foi o servidor Apache (auto implementável) seguido do banco MySQL e por último a linguagem PHP.

Com o servidor Apache as páginas puderam ser hospedadas na web de forma local (rede local), apenas para testes. Futuramente um servidor pode ser contratado para que o *web service* possa ser acessado por qualquer dispositivo ligado à internet, e não ser acessado somente pela rede local.

Com o banco de dados MySQL, foi possível fazer a persistência de dados (gravar os dados continuamente para produzir um histórico); quando um dispositivo ou a aplicação web solicita algum dado, o sistema busca os dados no banco e mostra.

Com a linguagem PHP e o framework Slim, foi possibilitada a criação das nossas

páginas web que mostram e inserem os dados; a parte visual do nosso sistema foi feita dentro dos arquivos em PHP utilizando pequenas rotinas em HTML (uma pequena aplicação web).

O *web service*, mostra os dados guardados no sistema à medida que os dispositivos (*smartphones*, computadores, etc.) solicitam esses dados, seja o histórico ou a medida em tempo real.

5.4.4.3. Resultados

A Figura 40 mostra o banco de dados criado para armazenar os dados obtidos pelos dispositivos.

		id_leitura	id_dispositivo	valor_icos_fundo	valor_icos_superficie	valor_ultrassonico	latitude	longitude	elevacao	data_hora
<input type="checkbox"/>	<input type="button" value="Editar"/> <input type="button" value="Copiar"/> <input type="button" value="Remover"/>	1	1	1	0	25.4	-20.1205663	-40.2360598	37	2017-08-16 11:50:00
<input type="checkbox"/>	<input type="button" value="Editar"/> <input type="button" value="Copiar"/> <input type="button" value="Remover"/>	43	2	0	1	30.5	-20.1515615	-40.4898489	25	2017-09-22 09:14:12
<input type="checkbox"/>	<input type="button" value="Editar"/> <input type="button" value="Copiar"/> <input type="button" value="Remover"/>	2	2	0	0	30.5	-20.1515615	-40.4898489	25	2017-09-22 09:14:12
<input type="checkbox"/>	<input type="button" value="Editar"/> <input type="button" value="Copiar"/> <input type="button" value="Remover"/>	63	1	0	1	30.5	-20.1515615	-40.4898489	25	2017-09-22 09:14:12
<input type="checkbox"/>	<input type="button" value="Editar"/> <input type="button" value="Copiar"/> <input type="button" value="Remover"/>	62	1	0	1	30.5	-20.1515615	-40.4898489	25	2017-09-22 09:14:12
<input type="checkbox"/>	<input type="button" value="Editar"/> <input type="button" value="Copiar"/> <input type="button" value="Remover"/>	61	1	0	1	30.5	-20.1515615	-40.4898489	25	2017-09-22 09:14:12
<input type="checkbox"/>	<input type="button" value="Editar"/> <input type="button" value="Copiar"/> <input type="button" value="Remover"/>	60	1	0	1	30.5	-20.1515615	-40.4898489	25	2017-09-22 09:14:12
<input type="checkbox"/>	<input type="button" value="Editar"/> <input type="button" value="Copiar"/> <input type="button" value="Remover"/>	59	1	0	1	30.5	-20.1515615	-40.4898489	25	2017-09-22 09:14:12
<input type="checkbox"/>	<input type="button" value="Editar"/> <input type="button" value="Copiar"/> <input type="button" value="Remover"/>	58	1	0	1	30.5	-20.1515615	-40.4898489	25	2017-09-22 09:14:12

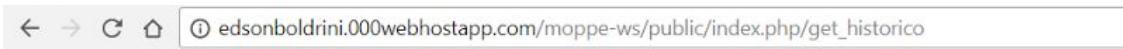
Figura 40 - Banco de dados.

Colunas da tabela de leituras:

- ***id_leitura***: Representa a identificação da leitura, quanto maior o número mais nova ela é.
- ***id_dispositivo***: Representa o dispositivo do qual aquela leitura veio.
- ***valor_icos_fundo***: Representa o valor do sensor de fundo (verdadeiro ou falso).
- ***valor_icos_superficie***: Representa o valor do sensor da superfície (verdadeiro ou falso).
- ***valor_ultrassonico***: Representa o valor do sensor ultrassônico (em centímetros).
- ***latitude***: Representa a latitude do dispositivo no globo terrestre.
- ***longitude***: Representa a longitude do dispositivo no globo terrestre.
- ***elevacao***: Representa a elevação do dispositivo em relação ao mar.
- ***data_hora***: Representa a data e hora que a leitura foi medida no rio.

Os resultados obtidos no *web service* são as próprias requisições GET dos dispositivos

ao mesmo. Nos exemplos mostrados abaixo foram simulados dados que poderiam estar guardados dentro do banco de dados já obtidos (histórico) e uma nova medida sendo inserida no banco. Estas solicitações usam o método GET do protocolo HTTP que utiliza da URL escrita no navegador para obter o histórico e inserir o dado no banco.



Moppe - Leituras

Dispositivo 1:

Leitura	Dispositivo	I. Baixo	I. Alto	Ultras.	Lat.	Long.	Elevação	Data/Hora	Nível
4978	1	1	0	30.5	-20.31	-40.29	7	22/09/2017 07:14:12	Normal
4975	1	1	0	30.5	-20.31	-40.29	7	22/09/2017 07:14:12	Normal
4974	1	1	0	30.5	-20.31	-40.29	7	22/09/2017 07:14:12	Normal
4973	1	1	0	30.5	-20.31	-40.29	7	22/09/2017 07:14:12	Normal
4972	1	1	0	30.5	-20.31	-40.29	7	22/09/2017 07:14:12	Normal
4971	1	1	0	30.5	-20.31	-40.29	7	22/09/2017 07:14:12	Normal
4970	1	1	0	30.5	-20.31	-40.29	7	22/09/2017 07:14:12	Normal
4969	1	1	0	30.5	-20.31	-40.29	7	22/09/2017 07:14:12	Normal
4968	1	1	0	30.5	-20.31	-40.29	7	22/09/2017 07:14:12	Normal
4967	1	1	0	30.5	-20.31	-40.29	7	22/09/2017 07:14:12	Normal

D1:

Contador normal = 10
 Contador intermediário = 0
 Contador crítico = 0

Figura 41 - Histórico de dados enviados pelo dispositivo e armazenados no banco de dados.

A inserção de dados no banco se dá através do método `/get_insere`. Através do padrão de URL criado, os parâmetros da requisição GET são passados ao servidor, onde são recuperados através do método `$_GET` do PHP e inseridos no banco de dados. Os parâmetros da URL foram definidos como:

```

/get_insere/
{id_dispositivo}/
{valor_icos_fundo}/
{valor_icos_superficie}/
{valor_ultrassonico}/
{latitude_sinal}/
{latitude_inteiro}/
{latitude_decimal}/
{longitude_sinal}/
{longitude_inteiro}/
{longitude_decimal}/
{elevacao}/
{dia}/
{mes}/
{ano}/
{hora}/
{minuto}/
{segundo}

```

Com este *web service* foi possível a comunicação entre dispositivos de sensoriamento e seus veículos de informação, seja para análise ou notificação de pessoas. Com ele não é possível apenas trabalhar informações relacionadas ao nível de um rio ou informar se haverá uma enchente, com pequenas mudanças e ajustes esse *web service* pode ser usado em qualquer dispositivo de sensoriamento que precise de persistência de dados e disponibilidade para aplicativos, tornando assim seu uso mais abrangente.

5.4.4.4. Aplicações nativas, híbridas, Ionic e App

Aplicações nativas são aplicações programadas em uma linguagem de programação de nível superior exclusiva para cada sistema operacional e os aplicativos se encontram na loja de cada plataforma.

Os dados dos aplicativos nativos são armazenados dentro do próprio dispositivo tornando o acesso ao aplicativo rápido. Além de programados com as linguagens específicas de cada sistema operacional, eles são feitos utilizando os SDKs (Kit de desenvolvimento de software, ou Software development kit), facilitando o acesso de configurações e funcionalidades do sistema operacional.

Uma aplicação híbrida, é a combinação de linguagens web como HTML, CSS (SCSS) e JavaScript (TypeScript) e aplicações nativas.

Essa aplicação usa a praticidade de programação, configuração e manutenção web que é de código único e mais simples, escrevendo apenas um código para todas as plataformas, sem necessitar de linguagens específicas para cada sistema operacional. Utiliza das aplicações nativas para melhorar a acessibilidade das configurações e funcionalidades do dispositivo mobile. As aplicações híbridas são executadas em um ambiente de processo nativo no dispositivo, igual a uma aplicação normal.

No aplicativo foi utilizado o *framework* de desenvolvimento de aplicativos híbridos Ionic. O Ionic utiliza as mesmas linguagens de programação que uma página Web (comuns em qualquer framework de aplicativos híbridos). Partes visuais são programadas em HTML, com pequenas mudanças para se adequar ao app, os estilos são programados em SCSS (similar ao CSS), e o back-end é programado em TypeScript (o popular sucessor do JavaScript).

O App não foi produzido nativamente por conta da implementação ser mais complicada e dificuldade didática, e por conta disso, a aplicação híbrida foi escolhida para projeto.

A Figura 42 mostra o ícone que identifica o App desenvolvido.



Figura 42 - Ícone do aplicativo de celular.

As próximas figuras mostram as telas do aplicativo de celular.

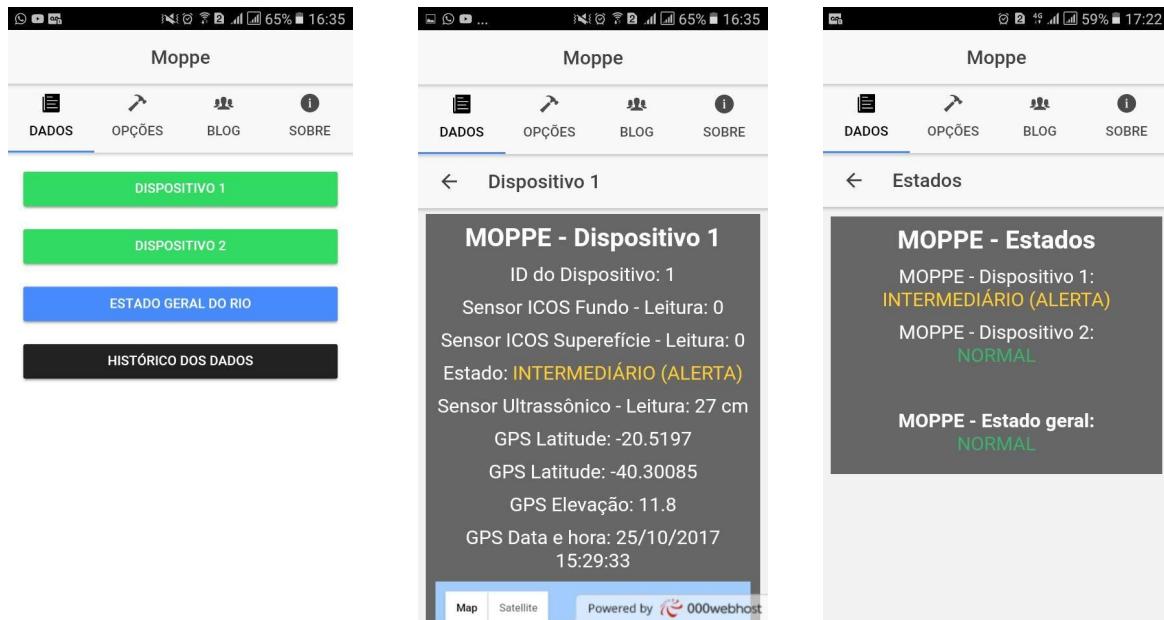


Figura 43 - Telas: Tela inicial; Tela de dados do dispositivo 1; Tela de estado geral da região.

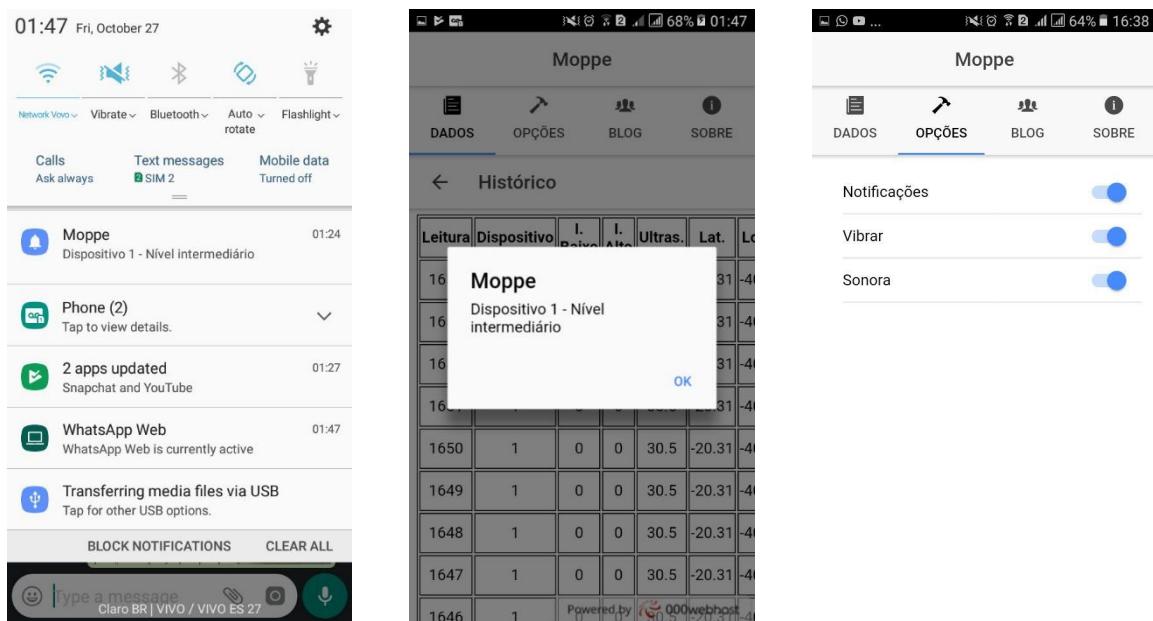


Figura 44 - Tela de notificações e tela de configurações de notificação.

Com este App é possível analisar todos os dados que os dispositivos estão fornecendo, bem como seu principal objetivo, avisar a população e órgãos responsáveis alertando possíveis comportamentos incomuns.

6. Referências

- Celante, Jovania. “Mapeamento da dinâmica do uso e ocupação do solo na Bacia Hidrográfica do Rio Jacaraípe- potencialização dos impactos ambientais sobre a qualidade dos recursos hídricos”, Vitória, 2004.
- Albuquerque, Letícia. “Ocupação de Áreas de Interesse Ambiental: um desafio da gestão da cidade no século XXI”, Vitória, 2010.
- Em: <<https://www.google.com.br/maps/@-20.1548276,-40.1882731,17z>> Acesso em: 18 março 2016.)
- COMETTI, Any. Audiência pública apresentará EIA/Rima de obras no Rio Jacaraípe. Em:
<http://seculodiarario.com.br/15840/10/audiencia-publica-apresentara-eiarima-de-obras-no-rio-jacaraipe-1> Acesso em: 18 março 2016.
- Obras rio Jacaraípe. Em:
<http://www.portaltemponovo.com.br/rio-jacaraipe-impasse-sobre-a-dragagem-esta-pronto-do-fim> Acesso em: 23 março 2016.
- Obras rio Jacaraípe. Em:
http://www.eshoje.jor.br/_conteudo/2014/04/noticias/meio_ambiente/16826-obra-de-limpeza-e-dragagem-continua-no-rio-jacaraipe.html Acesso em: 23 março 2016.
- (Em: <<https://www.achetudoeregiao.com.br/es/serra/localizacao.html>> Acesso em: 23 março 2016.)
- (Em:
<http://g1.globo.com/espirito-santo/noticia/2014/04/obra-do-rio-jacaraipe-fica-pronta-ate-2015-promete-prefeito-da-serra-es.html>) Acesso em: 13 abril 2016.)
- (ALVES, Rodolfo. Assoreamento. Em:
<http://escolakids.uol.com.br/assoreamento.htm>) Acesso em: 20 abril 2016.)
- (Em: <<http://www.embarcados.com.br/medindo-o-nivel-de-agua-com-Arduino/>>

Acesso em: 24 junho 2016.)

- (Em: <<http://pt.slideshare.net/IgorFastroniCorra/Arduino-36963863>> Acesso em: 24 junho 2016.)
- (Em: <http://www.corradi.junior.nom.br/sensores_Ind.pdf> Acesso em: 24 junho 2016.)
- (Em:
<<http://www.techtudo.com.br/listas/noticia/2016/03/raspberry-pi-conheca-os-modelos-e-saiba-qual-o-mais-indicado-para-voce.html>> Acesso em: 29 junho 2016.)
- (Em: <<http://www.oArduino.com/conheca-o-Arduino-mega-2560/>> Acesso em: 1 julho 2016.)
- (Em: <<http://www.embarcados.com.br/Arduino-mega-2560/>> Acesso em: 1 julho 2016.)
- (BOURQUE, Brad. Arduino vs. Raspberry Pi: Mortal enemies, or best friends? (inglês). Em: <<http://www.digitrends.com/computing/Arduino-vs-raspberry-pi/>> Acesso em: 1 julho 2016.)
- (ORSINI, Lauren. Arduino Vs. Raspberry Pi: Which Is The Right DIY Platform For You? (inglês) Em:
<<http://readwrite.com/2014/05/07/Arduino-vs-raspberry-pi-projects-diy-platform/>> Acesso em: 1 julho 2016.)
- AKYILDIZ, I. F. et al. Wireless sensor networks: a survey. Computer networks, v. 38, p. 393-422, 2002.
- ALKANDARI, Abdulrahman et al. Water monitoring system using Wireless Sensor Network (WSN): Case study of Kuwait beaches. In: Digital Information Processing and Communications (ICDIPC), 2012 Second International Conference on. IEEE, 2012. p. 10-15.
- AMBARKÜTÜK, Murat; GÜNER, Hüseyin Emre. Building wireless mesh sensor networks using xbee and arduino. 2013.

- ASWALE, Pramod et al. Water environment monitoring system based on wsn. International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE). 2015.
- DE CARVALHO, Fabricio Braga S. et al. Aplicações ambientais de redes de sensores sem fio. Revista de Tecnologia da Informação e Comunicação. Iecom-Instituto de Estudos Avançados em Comunicações, v. 2, n. 1, p. 14-19, 2012.
- FAVA, Maria Clara et al. Proposta metodológica para previsões de enchentes com uso de sistemas colaborativos. XX Simpósio Brasileiro de Recursos Hídricos, p. 1-8, 2013.
- FLORIANO, DIOGO et al. SIGMAOn–Sistema de Informação Geográfica para Monitoramento de Alagamentos On-line. XXXIV Congresso da Sociedade Brasileira de Computação (CSBC), 2014.
- LOFFI, LEANDRO ET AL. MONIT-RIO– Tecnologia da informação de comunicação para monitoramento de rios em casos de cheias, XIII Simpósio de Excelência em Gestão e Tecnologia (SEGeT). 2016.
- LOUREIRO, A. A. et al. Redes de sensores sem fio. Simpósio Brasileiro de Redes de Computadores (SBRC). Anais...2003
- MAINWARING, Alan et al. Wireless sensor networks for habitat monitoring. In:Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications. ACM, 2002. p. 88-97.
- MARTINS JÚNIOR, Cláudio José et al. Riversense: um sistema para monitoramento de rios através de redes de sensores sem fio. 2013.
- PECHOTO, MURILO M.; UEYAMA, JÓ; PEREIRA, J. P. A. E-noé: Rede de sensores sem fio para monitorar rios urbanos. In: Congresso Brasileiro Sobre Desastres Naturais. 2012.
- POLASTRE, J. et al. Wireless sensor networks. In: RAGHAVENDRA, C. S.; SIVALINGAM, K. M.; ZNATI, T. (Eds.). Norwell, MA, USA: Kluwer Academic Publishers, 2004. p. 399-423

- RAGHAVAN, Vinay; SHAHNASSER, Hamid. Embedded Wireless Sensor Network for Environment Monitoring. *Journal of Advances in Computer Networks*, v. 3, n. 1, 2015.
- RNP. CIA2 - Construindo Cidades Inteligentes da Instrumentação dos Ambientes ao Desenvolvimento de Aplicações. Disponível em:
<http://www.nr2.ufpr.br/~cia2/?page_id=185>. Acesso em: 16 jul. 2012
- SILVA, J. F. M. C. et al. Building a node for wireless sensor network based on open source platform arduino. In: Computing System Engineering (SBESC), 2012 Brazilian Symposium on. IEEE, 2012. p. 224-224.
- SU, K.; LI, J.; FU, H. Smart city and the applications. *Electronics, Communications and Control (ICECC)*, 2011 International Conference on. Anais...IEEE, 2011.
- SZEWCZYK, Robert et al. Habitat monitoring with sensor networks. *Communications of the ACM*, v. 47, n. 6, p. 34-40, 2004.
- YICK, J.; MUKHERJEE, B.; GHOSAL, D. Wireless sensor network survey. *Computer networks*, v. 52, p. 2292-2330, 2008.
- Driver for nRF24L01(+) 2.4GHz Wireless Transceiver. Disponível em:
<<http://maniacbug.github.io/RF24/>>. Acesso em 30 de outubro de 2017.
- Documentação da biblioteca WeeESP8266. Disponível em:
<http://docs.iteadstudio.com/ITEADLIB_Arduino_WeeESP8266/index.html>. Acesso em 30 de outubro de 2017.
- Documentação da biblioteca NewPing. Disponível em:
<<https://bitbucket.org/teckel12/arduino-new-ping/wiki/Home>>. Acesso em 30 de outubro de 2017.
- Documentação da biblioteca TinyGPS++. Disponível em:
<<http://arduiniana.org/libraries/tinygpsplus/>>. Acesso em 30 de outubro de 2017.
- Arduino SdFat Library. Disponível em:
<<http://arduinominas.com.br/SDFATLib/html/index.html>>. Acesso em 30 de outubro

de 2017.

- Blog do FilipeFlop. Disponível em:
<<https://www.filipeflop.com/blog/category/arduino/>>. Acesso em 30 de outubro de 2017.
- Blog Arduino e Companhia. Disponível em: <<http://www.arduinoecia.com.br/>>. Acesso em 30 de outubro de 2017.
- SLIM FRAMEWORK, First Application Walkthrough, Disponível em:
<<https://www.slimframework.com/docs/tutorial/first-app.html>>. Acesso em: 18 ago.2017.
- JAYR ALENCAR, API RESTful com PHP e Slim Framework, Disponível em:
<<https://clubedosgeeks.com.br/programacao/php/api-restful-com-php-e-slim-framework/>>. Acesso em: 23 ago.2017.
- DANIEL SCHMITZ, Aprenda a usar o RESTful com PHP e Slim Framework, Disponível em:
<<https://imasters.com.br/linguagens/php/aprenda-a-usar-o-restful-com-php-e-slim-framework/?trace=1519021197&source=single>>. Acesso em: 16 ago.2017.
- SILVA, LLB; PIRES, Daniel Facciolo; NETO, Silvio Carvalho. Desenvolvimento de aplicações para dispositivos móveis: tipos e exemplo de aplicação na plataforma IOS. Franca/SP, 2015.
- Documentação Ionic. Disponível em: <<https://ionicframework.com/docs/>>. Acesso em 08 de outubro de 2017.
- GRILLO, RAFAEL. Introdução ao Ionic Framework. Disponível em:
<<https://tableless.com.br/introducao-ao-ionic-framework/>>. Acesso em 08 de outubro de 2017.
- Udemy. Ionic 3 Essencial. Disponível em:
<<https://www.udemy.com/ionic-3-essencial/>>. Acesso em 08 de outubro de 2017.
- Udemy. Ionic 3 para iniciantes. Disponível em:

<<https://www.udemy.com/ionic-3-para-iniciantes/>>. Acesso em 08 de outubro de 2017.