2021-1

Trabalho de Graduação do Curso de Análise e Desenvolvimento de Sistemas

SISTEMA AUTOMÁTICO DE PREVENÇÃO DE ENCHENTES COM A UTILIZAÇÃO DO MICROCONTROLADOR ESP32

MENDANHA, Guilherme Bueno; MORBECK, Leonardo; (PARDO, Mario Henrique de Souza)

e-mail:

guimendanha@hotmail.com; morbeck2015@gmail.com

Resumo: Esse estudo tem como intuito de desenvolver uma automação capaz de prevenir a ocorrência de enchentes através do controle da vazão de agua em três áreas. Para sua criação será utilizado o microcontrolador ESP32, sensores ultrassónicos, servo motor e o software RemoteXY. Para constatar a sua viabilidade será realizado testes em um ambiente controlado e com o auxílio de um protótipo físico em miniatura. Neles será comparado o funcionamento dos componentes , levando em conta o valor esperado pela lógica criada com o valor real encontrado. O protótipo se provou ineficiente ao apresentar uma diferença média relativamente grande entre esses valores no funcionamento do servo motor.

Palavras-chave: Alagamentos, ESP32, Comportas, Automação

Abstract: *This study aims to develop an automation capable of preventing the occurrence of floods by controlling the flow of water in three areas. For its creation will be used the ESP32 microcontroller, ultrasonic sensors, servo motor and RemoteXY software. To verify its feasibility, tests will be conducted in a controlled environment and with the aid of a miniature physical prototype. In them, the functioning of the components will be compared, considering the value expected by the logic created with the real value found. The prototype proved inefficient by presenting a relatively large average difference between these values in the functioning of the servo motor.*

Keywords: *Flood. ESP32. Floodgates, Automation.*

1 Introdução

Enchentes e inundações são problemas sérios que afetam os humanos a diversos séculos atras nas mais diversas regiões. Existem relatos que se estendem de 600 a.C. até 1946 sobre as enchentes do rio Amarelo, no norte da China, que já causou a morte de milhões de pessoas em suas 1593 ocorrências (UNZER MACEDO, 2018). No entanto, com os passar dos

anos os seus episódios pelo mundo aumentaram de vido a alguns fatores, como a drenagem natural deficiente e baixa capacidade de suporte dos solos locais, geralmente causados pela constante urbanização e destruição da vegetação nativa, que acarreta a impermeabilização do solo. (PONTES, 2017)

O problema da inundação é mais acentuado em áreas que são próximas a rios e córregos, principalmente em locais alteram o curso natural do rio através das instalações de represas e diques. A fim de resolver esses eventos, foram desenvolvidas diversas técnicas, entre elas a criação de comportas hidráulicas, que apresentam diversos tipos e tamanhos, podendo ser encontrados em diversas regiões afetadas. (HAJJAJ, 2020)

A evolução contínua da tecnologia nos permite tornar essa técnica ainda mais efetiva e precisa. Esse artigo científico tem como objetivo construir um sistema capaz de prevenir e atrasar efeitos causados por enchentes e inundações utilizando de comportas hidráulicas automáticas.

Esse projeto teve origem em um trabalho de faculdade e será ampliado, com o desenvolvimento de uma plataforma de controle online que permitirá compartilhar informações em tempo real sobre o estado da comporta e o nível da área rio ou córrego em que foi instalado, além de conceder ao usuário a opção de alterar manualmente a altura de cada comporta, aumentando assim a segurança em caso de falha do algoritmo.

2 Justificativa

As inundações têm sido um grande problema para a história da civilização humana. Alguns estudos alegam que está acontecendo um aumento em sua ocorrência, devido a urbanização desenfreada em algumas áreas (HODGKINS, 2019) e as mudanças climáticas que acontecem pelo mundo (DIDOVETS et al, 2019). Tal desastre é acompanhado de inúmeros danos a propriedades, alteração da vegetação local e cidadãos feridos e mortos. Somente no século 20, já foram registrados oito milhões de morte causadas por inundações pelo mundo (SHANH, 2017).

Segundo FitzGerald (2016), entre o período de 1990-2015 foram relatos cerca de 3415 ocorrências de inundações, o que contabiliza 36,03% de todos os desastres naturais que aconteceram durante esse período (Figura 1).

Figura 1- Relação dos desastres naturais no período de 1990-2015

Type of disaster (% of all disasters)	*Occurrence*	*Deaths*	*Injured*	*Total affected*
Flood (36.03)	3415	182,702	1,086,122	2,855,214,859
Storm (26.14)	2477	401,701	607,843	766,030,113
Epidemic (11.89)	1127	180,452	499,176	21,194,391
Earthquake/tsunami (7.43)	704	816,294	1,720,306	134,884,514
Extreme temperature (4.67)	443	166,415	1,964,945	99,098,929
Landslide (4.62)	438	22,009	4,424	6,103,932
Drought (4.15)	393	24,272	–	1,245,945,379
Wildfire (2.99)	283	1,770	5,928	5,598,133
Volcanic activity (1.48)	140	1,639	1,517	4,034,275
Insect infestation (0.32)	30	–	–	2,802,200
Mass movement: dry (0.27)	26	1,210	312	15,130
Impact (0.01)	1	–	1,491	301,491

Fonte: FitzGerald (2016)

No Brasil, do período de 1998 até 2000, o Instituto Brasileiro de Geografia e Estatística (IBGE) registrou cerca de 1.235 municípios atingidos por enchentes ou inundações, que corresponde a aproximadamente 22,43% do total de municípios do país, 5.507 países. Apenas esse total já foi capaz de afetar mais de 48 mil hectares do território brasileiro (Tabela 1). Já entre o período de 2003 até 2008, houve um aumento nessas ocorrências atingindo um número de 2.274 de municípios afetados, aproximadamente 40,87% do total de municípios (Tabela 2).

Tabela 1 - Número total de municípios do Brasil, municípios que sofreram com inundações ou enchentes e extensão da área em hectares nos anos de 1998 a 2000

Número total de municípios registrados	5.507
Número de municípios atingidos por inundações ou enchentes	1.235
Extensão das áreas onde ocorreram inundações ou enchentes (Hectares)	48.809

Fonte: IBGE – Pesquisa Nacional de Saneamento Básico

Tabela 2 - Número total de municípios do Brasil e municípios que sofreram com inundações nos anos de 2003 a 2008

Número total de municípios registrados	5.564
Número de municípios atingidos por inundações ou enchentes	2.274

Fonte: IBGE – Pesquisa Nacional de Saneamento Básico

Segundo Paterson (2018), além do impacto imediato do desastre, é comprovado que existem efeitos a curto e longo prazo na saúde da população afetada. Eles são causados podem ser causados pela evacuação da área, pioras na condição de vida e transmissão de doenças, como por exemplo a leptospirose, infecções de pele e tecido e gastroenterites.

Todos esses fatos comprovam a importância do estudo da prevenção de enchentes. Um avanço nessa área possibilitaria salvar um número grande de pessoas e minimizar os dados causados pelo desastre.

3 Objetivos

3.1 Objetivo Geral

- Criar um sistema capaz de prevenir ou atrasar a ocorrência de enchentes;

3.2 Objetivos específicos

- Utilizar o microcontrolador ESP32
- Implementar um aplicativo móvel capaz de receber informações do sistema e habilitar controle manual
- Montagem de um protótipo físico

4 Fundamentação Teórica

Para entendermos melhor sobre a construção do dispositivo, daremos uma breve conceituação dos componentes utilizados no projeto. Os hardwares usados foram o

microcontrolador ESP32, sensor ultrassônico de distância e o servo motor, e os softwares utilizados são o Arduino IDE e Remote XY

ESP32

Devido a extensão do projeto, foi feito a troca da placa Arduino Uno pelo microcontrolador ESP32. O motivo dessa mudança foi pelo fato de que a placa Arduino necessita de uns módulos externos para executar a função de wi-fi e apresentar uma menor performance.

O ESP32 (Figura 2) é um microcontrolador que tem a função wi-fi e Bluetooth integrado na placa, por causa disso, ele é muito utilizado em aplicações que aplicam o conceito de internet das coisas (IOT), acesso remoto, webservers e dataloggers. Outro ponto positivo em sua utilização seria o seu tamanho e performance, mesmo sendo mais compacto que Arduino Uno, ele apresenta uma performance maior por causa de uma memória FLASH de até no máximo 64MB, uma memória de acesso aleatório estática (SRAM) de 520KB e ser um sistema dual-core. (MAIER, 2017)

O modelo do ESP32 escolhido para esse estudo foi o ESP32-WROOM-32. Ele apresenta um valor de memória integrada de 4MB e opera na voltagem de 3.0V a 3.6V. Ele apresenta 39 com as mais diversas funções, entre eles 30 pinos de *General Purpose Input/Output*, que são as estradas programáveis de entrada e saída que serão mais utilizadas.

Sensor Ultrassônico

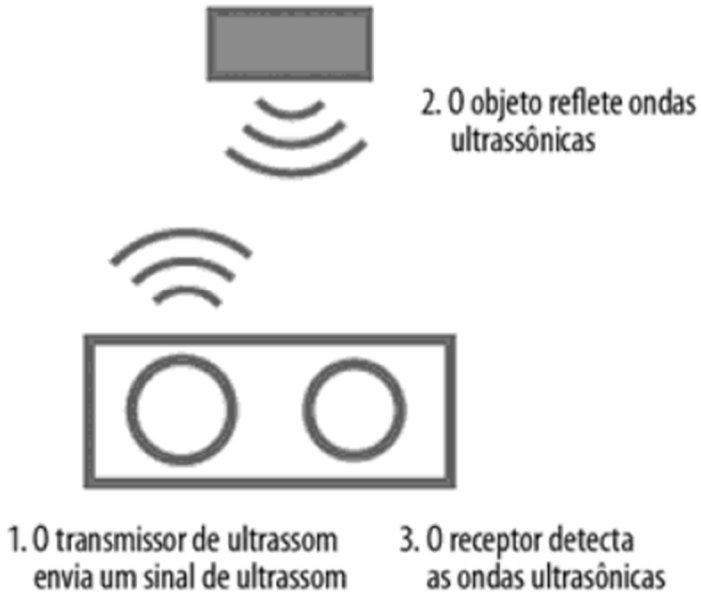
O livro de Evans (2013, p. 158) faz a seguinte caracterização desse tipo de sensor:

> "Um sensor ultrassônico consiste de dois componentes separados: um que envia o sinal e o outro que recebe de volta. O sensor também conterá componentes adicionais, incluindo um microcontrolador pequeno, que é responsável por determinar o tempo entre o envio e o recebimento do som"

O tempo entre o envio e recebimento do sinal, é codificado em forma de uma tensão. Com esse resultado é possível fazer um cálculo matemático que retorna um tempo aproximado em unidades de marcação de tempo, geralmente é transformado para segundos ou milissegundos, devido a potência do equipamento. (EVANS, 2013) O cálculo da distância leva

em conta o meio de propagação das ondas, que no projeto atual seria o ar, e o tempo encontrado no cálculo anterior. Um fato importante seria que, essa tensão deverá ser dividida por 2, pois ela representa a ida e volta da onda de som. Podemos ver um exemplo prático de como funciona o envio e o retorno dos sinais do sensor na Figura 2 e a equação que será utilizada para encontrar a distância na Equação 1.

Figura 2 - Sistema do sensor ultrassônico



Fonte: Evans (2013)

$$d = \frac{t.v}{2} \quad (1)$$

No qual:

d: Distância em centímetros

t: Tempo da resposta

v: Velocidade do som

Servo Motor

Os servo motores são atuadores rotativos que permitem ao usuário um controle preciso de movimentos. Em sua programação é possível controlar sua posição angular, velocidade,

aceleração na sua utilização (LATHA, 2016), através de um *feedback* sobre a posição final e movimento, que é enviado ao microcontrolador (KAUR, 2016).

O motor é conectado a um sensor, que permite encontrar a faixa da distância de até 180º em torno do mesmo(LATHA, 2016). Durante seu movimento é consumido energia até que o motor atinja a posição requisita, após ele entra em estado de espera, que não apresenta esse consumo (KAUR, 2016).

Esse tipo de motor foi escolhido como componente ao invés do motor CC, pois pela pequena proporção do projeto seria necessário que o motor fosse capaz de realizar movimentos precisos para controlar a porcentagem de abertura e fechamento das comportas.

IDE e Linguagem de Programação

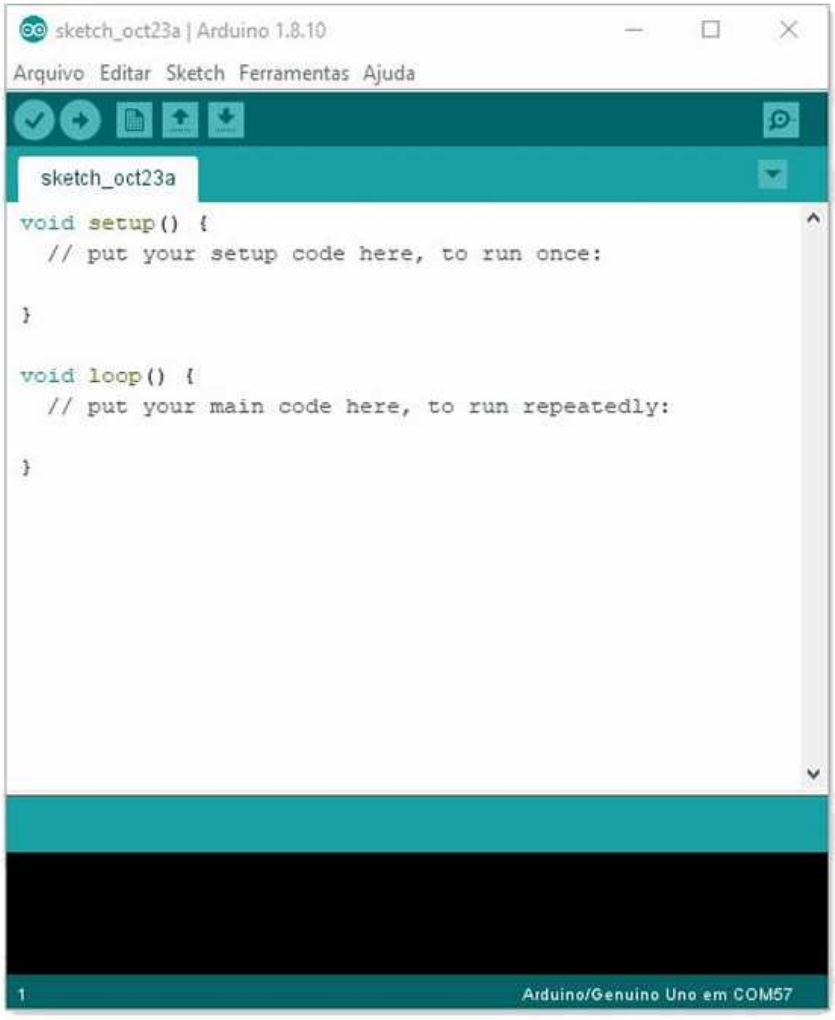
A codificação do projeto será feita através do uso do Arduino IDE. O que possibilita programar o código em linguagem C/C++, que será processada no módulo ESP32. (ARDUINO, 2018)

A IDE (Integrated Development Environment) é um ambiente de desenvolvimento integrado, que facilita a programação de diversos softwares. O ambiente que será utilizado foi desenvolvido pela Arduino.cc, com capacidade de editar, compilar e carregar código no dispositivo Arduino. (ARDUINO, 2018)

A estrutura para codificar programas para o Arduino é separado em duas sessões, sendo a função setup ( ) e a função loop ( ), é possível visualizá-las na Figura 3. A função setup é usada para configurar os componentes utilizados na prototipação. Já a função loop é utilizada para o desenvolvimento do algoritmo, que permitirá a execução de modo contínuo dos componentes. (ARDUINO, 2018)

Após terminar toda codificação, será necessário conectar a placa no computador via cabo USB para fazer o upload do código, com isso a placa já pode ser desconectada e ela conterá todos os dados necessários para a utilização do protótipo. (ARDUINO, 2018)

Figura 3 - Arduino IDE versão 1.8.10



Fonte: Straub (2019)

Remotexy

O Remotexy é uma plataforma online que nos permite a criação e edição de interface gráfica para controlar Arduino via smartphone ou tablet. (REMOTEXY, 2016)

Depois que desenvolver a interface, você obterá o código fonte para o microcontrolador que está utilizando. Esse código dá suporte para interação entre o programa, o controle e sua exibição. (REMOTEXY, 2016)

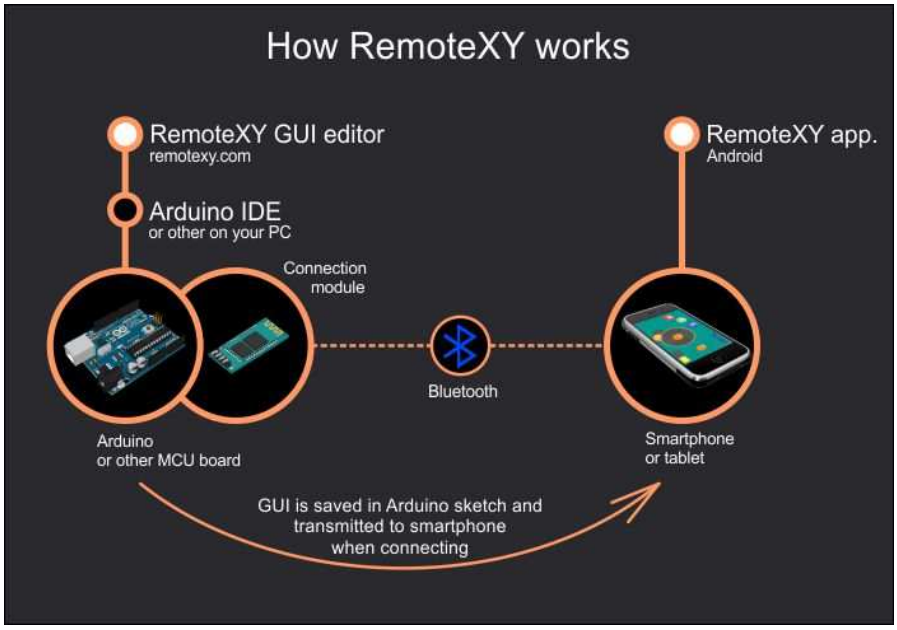
Uma característica distintiva do Remotexy, é que a estrutura da interface é armazenada no controlador e não no dispositivo móvel. Outra característica é a capacidade de controlar vários microcontroladores a partir de um único smartphone ou tablet. (REMOTEXY, 2016)

O dispositivo móvel e o controlador podem se conectar a partir de Bluetooth, WIFI, Ethernet, USB e qualquer outra internet através de servidor nuvem. (REMOTEXY, 2016)

Através das explicações sobre a funcionalidade do Remotexy citadas acima, podemos ver o mesmo na Figura 4 (REMOTEXY, 2016).

O gerador de código-fonte da plataforma tem suporte aos seguintes controladores: Arduino UNO, Arduino MEGA, Arduino Leonardo, Arduino Pro Mini, Arduino Nano, Arduino MICRO, WeMos D1, WeMos D1 R2, WeMos D1 mini, NodeMCU V2, NodeMCU V3 e outros controladores baseados no núcleo ESP8266. Controladores baseados no núcleo ESP32, O AirBoard, ChipKIT UNO32, ChipKIT uC32, ChipKIT Max32. (REMOTEXY, 2016)

Figura 4 – Funcionamento do RemoteXY



Fonte: RemoteXY (2016)

5 Trabalhos Similares

Existem artigos científicos que propõem a utilização de microprocessadores para controlar o nível de água, como o trabalho de Wahyuni (2021) que propôs esse sistema utilizando do Arduino Uno R3 para controlar automaticamente a quantidade de água armazenada em reservatórios de sua faculdade ou invés do sistema puramente mecânico de boias que apresenta algumas falhas, como vazamento nas torneiras por causa da pressão da água nos canos.

Alguns estudos abrangem esse sistema em largar escala, como no caso do uso da tecnologia como meio de se prevenir a ocorrência de enchentes ou controlar níveis de água em reservatórios, como por exemplo o estudo de Kavitha (2021), que utilizou o microprocessador Arduino Uno R3, e o estudo de Hajjaj (2020), que utilizou o microcontrolador Rasberry Pi 3 (RPi).

A proposta montada por Kavitha (2021) tem como objetivo impedir que a água presente em represas transborde e cause uma enchente nas áreas próximas. Seu sistema propõe fazer uma divisão em três represas e controlar o nível de água delas através do escoamento entre elas.

O trabalho de Kavitha(2021) se diferencia do nosso nos pontos que propõe a instalação de um microcontrolador em cada umas das represas, no qual a comunicação entre eles se dá pelo sistema mestre e escravo, no tipo de barragem de água entre represas, em que consiste em somente em aberto e fechado, e por último no fato de que é utilizado apenas um display de LCD é usado localmente para mostrar informações dos passos de sua lógica.

O artigo escrito por Hajjaj (2020) é o que apresenta o maior grau de semelhança ao trabalho atual. O seu projeto tem como objetivo controlar automaticamente, e quando necessário manualmente, as comportas do tipo gaveta em uma grande área à um baixo custo de produção. Em seu projeto foram utilizados um sensor ultrassónico HC-SR04 acoplado em cada comporta e o recursos que implementam o conceito de internet das coisas (IoT).

A utilização da função de Wi-fi do RPi, dispositivos IoT e o desenvolvimento de um algoritmo possibilitou que as informações do funcionamento do sistema fossem divulgadas em tempo real para os administradores da região e que fossem armazenadas na nuvem para análises. A utilização desse sistema mostrou-se efetivo nessa pesquisa, o que demostra a possiblidade do nosso sistema também apresentar sucesso.

6 Metodologia

6.1 Tipo de pesquisa

Pesquisa descritiva, experimental, de laboratório e quantitativa

6.2 População e amostra de dados

Leitura dos sensores e movimento dos servo motores esperados em comparação com as encontradas nos testes reais.

6.3 Coleta de dados

Será medido com o auxílio de uma régua a altura encontrada pelos sensores e o movimento das comportas durante a fase de testes. O movimento da comporta será acionada através de pedaços de isopor de tamanhos distintos.

6.4 Tratamento e análise de dados

Será feito através da montagem de gráficos e tabelas.

6.5 Instrumento de medida (métricas)

A métrica utilizada tanto para a leitura dos sensores e quanto para o tamanho encontrado das comportas será centímetros.

6.6 Ferramentas e tecnologias utilizadas

Será utilizado as seguintes tecnologias:

- Microcontrolador ESP-WROOM-32, responsável pelo processamento do código e controle dos componentes
- Servo motor TowerPro MG90S, responsável por movimentar as comportas
- Sensor ultrassónico HC- SR04, responsável pela leitura do nível de água
- Controladores de tensão de 5V, responsáveis por controlar tensões de alimentação do sensor ultrassónico
- Protoboard utilizado para facilitar a conexão entre o componentes
- Jumpers foram utilizados para passar sinais entre os componentes
- Arduino IDE foi utilizado como ambiente de desenvolvimento do código e lógica. O código foi feito na linguagem C.
- RemoteXY software em que será desenvolvidos todas as telas IOT

6.7 Recursos materiais

Durante o projeto foram utilizados os seguintes recursos de maneiram, para maior detalhes sobre quantidade e valores conferir a Tabela 3:

- Computador e celular
- Materiais de maquete: isopor, fita adesiva, palitos de churrasco;
- Componentes de hardware:
  - Servo motor TowerPro MG90S
  - Sensor ultrassónico HC- SR04
  - Microcontrolador ESP-WROOM-32
  - Protoboard de 830 pontos
  - Jumpers
  - Fonte Ajustável Para Protoboard
  - Fonte 9V 1A Bivolt para Arduino

Tabela 3 - Tabela de componentes

Componentes	Qtd.	Valor Unitário	Total
Componentes de Hardware			
ESP-WROOM-32	1	R$ 67,50	R$ 67,50
Jumpers Fêmea - Fêmea - 20cm - Pacote com 40	1	R$ 8,90	R$ 8,90
Jumpers Macho - Fêmea - 20cm - Pacote com 40	1	R$ 8,90	R$ 8,90
Jumpers Macho – Macho – 20cm – Pacote com 40	1	R$ 18,90	R$ 18,90
Protoboard 830 pontos	2	R$ 15,90	R$ 31,80
Sensor ultrassónico HC- SR04	3	R$ 10,90	R$ 32,70
Blackskull - Suporte para Sensor Ultrassônico	3	R$ 9,00	R$ 27,00
Micro Servo 9g SG90	2	R$ 15,90	R$ 31,80

Fonte Ajustável Para Protoboard	1	-	-
Placas de alumínio 5cm x 10cm	2	R$ 5,00	R$ 10,00
Fonte 9V 1A Bivolt para Arduino	1	-	-
Materiais para Maquete			
Isopor 1 m X 50cm X 50 mm	3	R$ 16,90	R$ 50,70
Bastão de Cola Quente	2	-	-
Pistola de Cola Quente	1	-	-
Barbante	1	R$ 6,00	R$ 6,00
Carretel de linha	4	R$ 2,50	R$ 10,00
Estilete	1	-	-
Régua 30 cm	1	-	-
Fita adesiva	1	-	-
Palito de churrasco	1	R$ 3,60	R$ 3,60
Total			**R$ 310,80**

6.8 Plano de trabalho

Etapa 1: Levantamento Bibliográfico: Procurar por artigos científicos parecidos que comprovam o sucesso do sistema

Etapa 2: Estudo do problema: Ambiente de aplicação, variáveis existentes, elementos sensoriais, elementos de atuação. Desenvolvimento de um projeto de solução.

Etapa 3: Levantamento dos componentes a serem utilizadas: Escolher todos os componentes principais de hardware do projeto

Etapa 4: Adaptação do código e algoritmo: Adaptar a código feito em Arduino para ESP 32 e alterar as lógicas de acordo com a escolha de peças

Etapa 5: Adição do sistema IoT: Criar a lógica para a utilização do Wi-fi e desenvolvimento de uma aplicação que visualiza os dados e permite o controle manual das comportas

Etapa 6: Compra dos materiais: Comprar todas as peças de hardware que serão utilizadas

Etapa 7: Desenvolvimento do protótipo físico: Desenvolver de maneira física o sistema

Etapa 8: Confecção de uma maquete: Melhoramento visual do protótipo físico

6.9 Cronograma

Tabela 4 - Cronograma

Atividades	Março	Abril	Maio	Junho	Julho	Agosto	Setembro	Outubro
Etapa 1	X	X	X					
Etapa 2		X	X	X	X	X	X	X
Etapa 3					X	X		
Etapa 4						X	X	
Etapa 5						X		
Etapa 6						X		
Etapa 7						X	X	X
Etapa 8								X

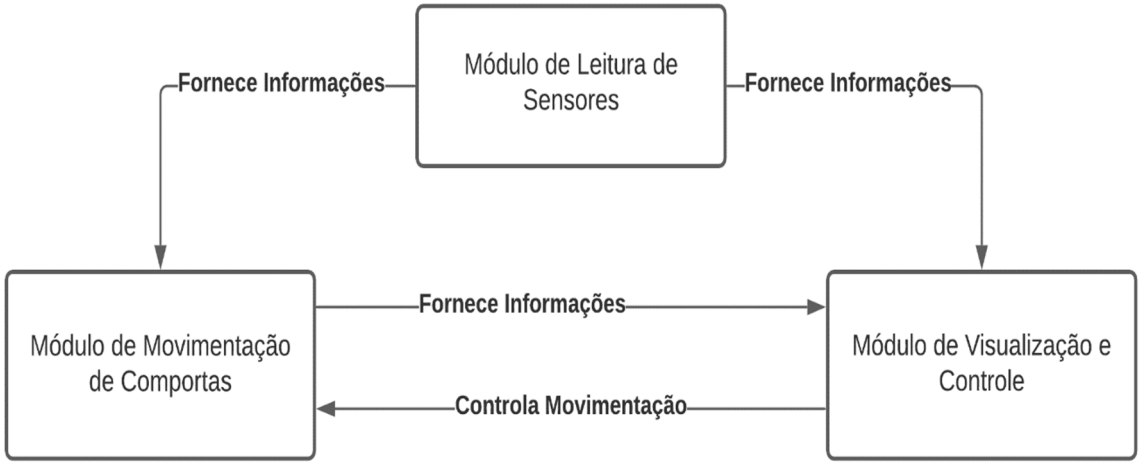
7 Desenvolvimento

O lógica do sistema tem como intuito testar se o controle da vazão de água em várias áreas pode ser capaz de prevenir enchentes. Como um córrego sempre corre em uma direção, ao diminuir a vazão em um trecho com o auxílio de uma comporta, acarretara o aumento do volume no local, ao mesmo tempo que diminuirá o volume da água na próxima área, sendo assim, seria possível controlar o volume da água afim de evitar enchentes.

O sistema a ser desenvolvido conterá duas comportas acompanhadas de dois servo motores que realiza o seu movimento e três sensores ultrassónicos que se serão utilizados para controlar esse movimento.

O desenvolvimento do projeto foi dividido em três módulos: Leitura dos sensores, Movimentação das comportas e Modulo de controle e visualização. (Figura 5)

Figura 5 – Relações entre os módulos dos sistemas



Fonte: Autoria Própria

O módulo de Leitura de sensores, como o próprio nome sugere será responsável por fazer a leituras dos sensores ultrassónicos e armazená-las esses dados em variáveis do sistema.

O módulo de Movimentação de comportas será responsável por controlar a altura da comporta, tendo como base as variáveis que foram salvadas pelo modulo de leitura de sensores. Para a sua lógica foram preparadas duas tabelas que são divididas em etapas com intervalos de 0,47 centímetros.

As tabelas representam dois tipos de situações. Para definir em qual será utilizada é realizado o cálculo da diferença entre as leituras de dois sensores seguidos, caso o resultado seja menor que 0,93 centímetros a tabela usada seria a primeira tabela (Tabela 2) e caso seja maior será a segunda tabela (Tabela 3).

Tabela 2 – Primeira situação de cálculo do movimento da comporta

Etapa	Média dos Sensores	Comporta
1	0 --| 0,47	0%
2	0,47 --| 0,94	12,5%
3	0,94 --| 1,41	25%
4	1,41 --| 1,88	37,5%
5	1,88 --| 2,35	50%
6	2,35 --| 2,82	62,5%
7	2,82 --| 3,29	75%
8	3,29 --| 3,76	87,5%
9	3,76 --| ∞	100%

Tabela 3 – Segunda situação de cálculo do movimento da comporta

Etapa	Comporta Anterior ao Sensor	Leitura	Comporta Posterior ao Sensor
1	100%	0 --| 0,47	0%
2	87,5%	0,47 --| 0,94	12,5%
3	75%	0,94 --| 1,41	25%
4	62,5%	1,41 --| 1,88	37,5%
5	50%	1,88 --| 2,35	50%
6	37,5%	2,35 --| 2,82	62,5%
7	25%	2,82 --| 3,29	75%
8	12,5%	3,29 --| 3,76	87,5%
9	0%	3,76 --| ∞	100%

Na situação em que se usa a primeira tabela (Tabela 2), o primeiro passo é fazer uma média entre os valores das duas leituras, verificar em qual etapa ela se encaixa e mover a comporta entre os sensores de acordo com a porcentagem especificada.

A segunda tabela (Tabela 3) é utilizada através de um sistema de prioridade, a área que apresente uma menor leitura sempre terá maior prioridade, ou seja, ela sempre será utilizada para controlar a altura das comportas.

O módulo de controle e visualização será responsável por exibir em um aplicativo de celular os dados de altura da comporta, leitura do sensores ultrassónicos e a possibilidade do acontecimento de enchente (Figura 6). Já sua parte de controle será responsável por permitir que o usuário controle manualmente a altura da comporta de acordo com sua vontade, essa função é de extrema importância, pois uma falha na logica poderá acarretar uma enchente que não viria a acontecer sem a presença da comporta (Figura 7).

Figura 6 – Design da tela de visualização de sensores

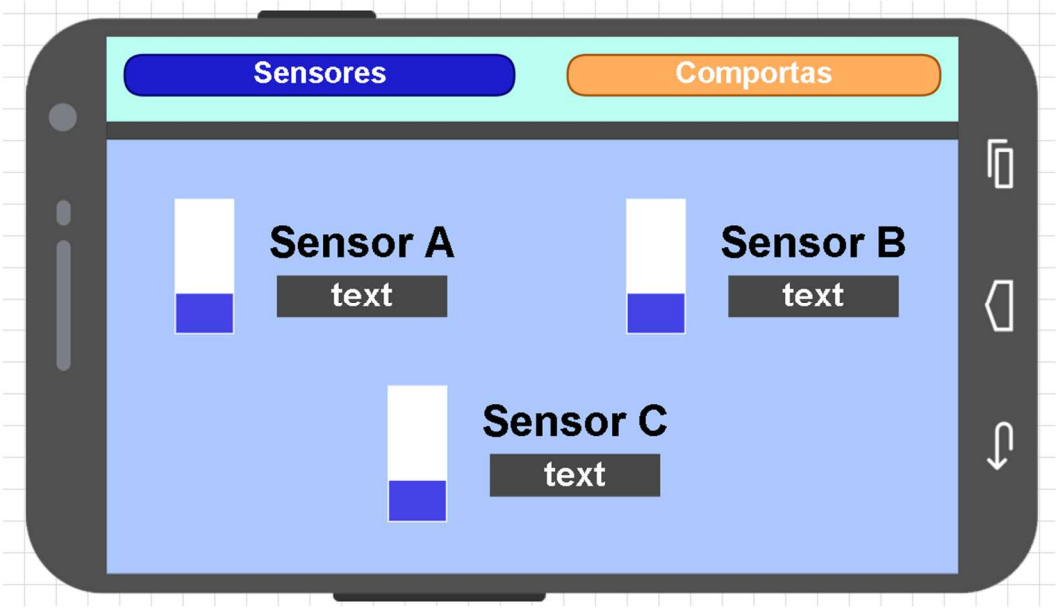
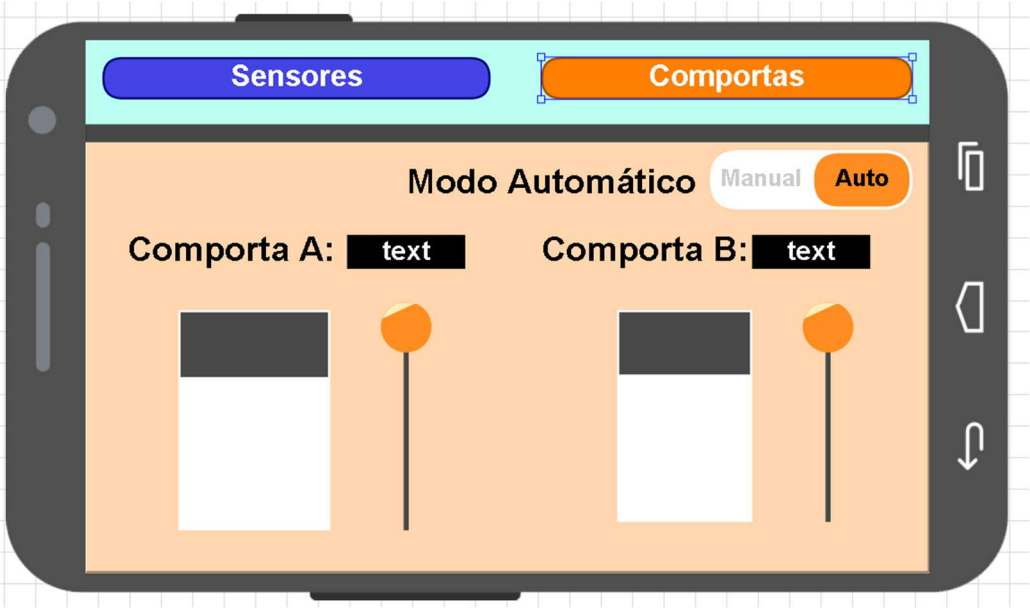
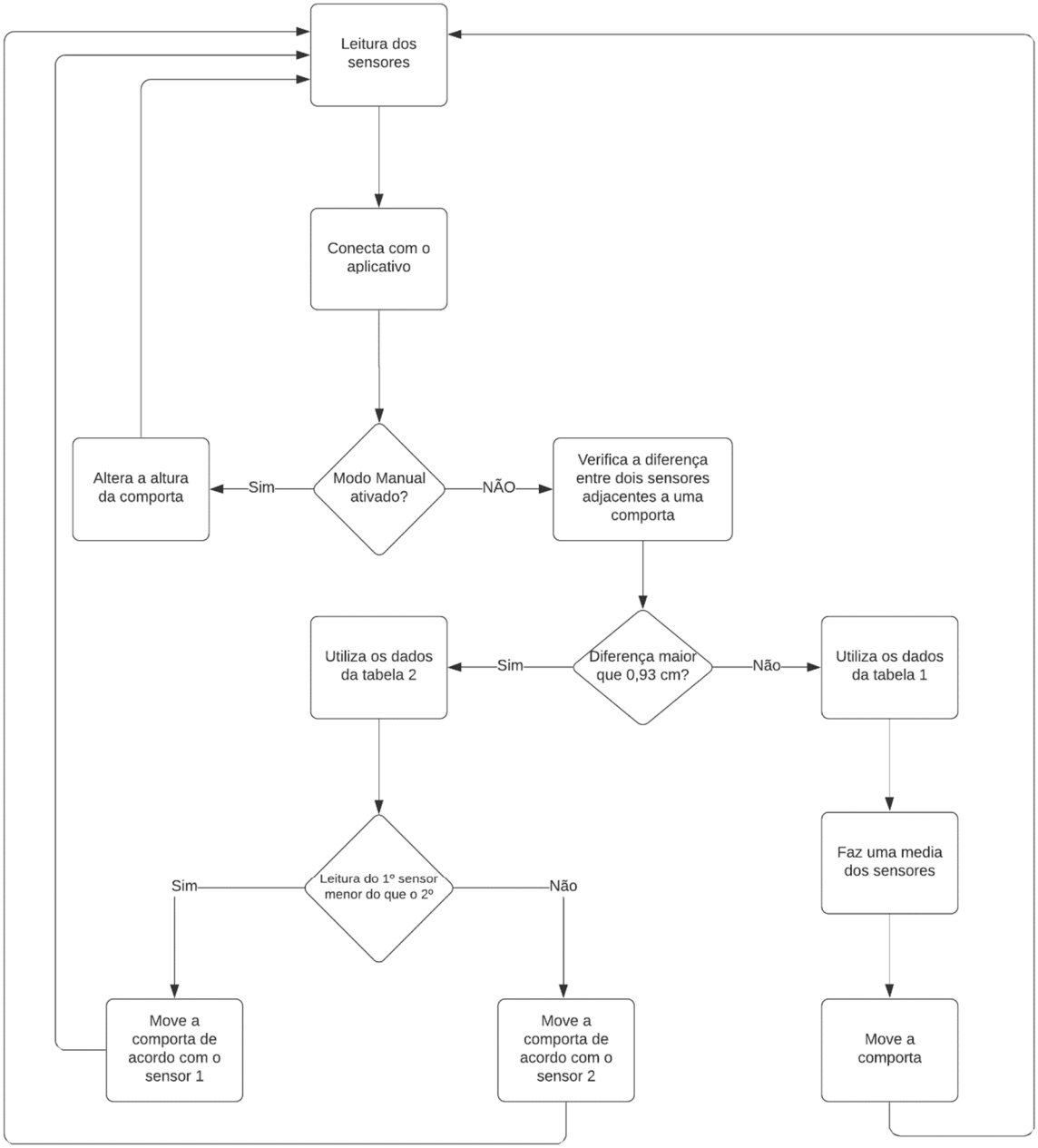


Figura 7 – Design da tela de nível de comportas e controle



A utilização do nosso sistema pode ser complementa ilustrado através do algoritmo apresentado na figura 8.

Figura 8 – Algoritmo de funcionamento do sistema



O quadro 1 demonstra o código da classe sensor. Essa classe é responsável por armazenar a pinagem necessária para o funcionamento do sensor ultrassónico HC- SR04, além realizar o cálculo de transforma a leitura obtida para distância em centímetros.

Quadro 1 – Codificação da classe sensor

```
// NewPing - Version: Latest
```

```
#include <NewPing.h>

unsigned long tempo;
unsigned long tempoDesejado;

class Sensor
{
  public:
    int trigger;
    int echo;
    float distanciaCm;
    float espacoVazio = 9.0;
    float alturaComporta = 4.5;

    Sensor(int triggerIn, int echoIn)
    {
      this -> trigger = triggerIn;
      this -> echo = echoIn;

    }

    void leitura()
    {
      NewPing sonar( this -> trigger, this -> echo, 100 );

      this -> distanciaCm = (float( sonar.ping_median(5)) / 57.0 ) -
espacoVazio;

      if (this -> distanciaCm > this -> alturaComporta) {
        this -> distanciaCm = this -> alturaComporta;
      }

      if (this -> distanciaCm < 0) {
        this -> distanciaCm = 0;
      }

      tempo = millis();
      tempoDesejado = tempo + 500;
```

```
      while ( tempo <= tempoDesejado )
      {
        tempo = millis();
      }
    }
};
```

O quadro 2 demonstra o código da classe comporta. Essa classe é responsável por armazenar dados da altura da comporta e controlar seu movimento de acordo com a lógica aplicada.

Quadro 2 – Codificação da classe comporta

```
// ESP32Servo - Version: Latest
#include <ESP32Servo.h>

unsigned long tempo;
unsigned long tempoDesejado;

class Comporta
{
  public:
    float alturaAtual;
    float alturaDesejada;
    int grauAtual;
    int grauDesejado;
    float alturaTotal = 4.5;
    unsigned long tempoComporta;
    int anguloMap;

    Servo servoMotor;
    int motorPin;

    Comporta( int pin )
    {
      Servo;

      this -> servoMotor = servo;
      this -> motorPin = pin;
```

```
      ESP32PWM::allocateTimer(0);
      ESP32PWM::allocateTimer(1);
      ESP32PWM::allocateTimer(2);
      ESP32PWM::allocateTimer(3);
      servo.setPeriodHertz(50);

      this -> servoMotor.attach( this -> motorPin, 500, 2400 );
      this -> grauAtual = this -> servoMotor.read();
    }

    void moveComporta() {

      if(this -> grauAtual > this -> grauDesejado)
      {
        for (this -> grauAtual; this -> grauAtual > this -> grauDesejado;
this -> grauAtual--)
        {
          this -> servoMotor.write(this -> grauAtual);

          tempo = millis();
          tempoDesejado = tempo + 40;

          while ( tempo <= tempoDesejado )
          {
            tempo = millis();
          }

        }
      } else {
        for (this -> grauAtual; this -> grauAtual < this -> grauDesejado;
this -> grauAtual++)
        {
          this -> servoMotor.write(this -> grauAtual);

          tempo = millis();
          tempoDesejado = tempo + 40;

          while ( tempo <= tempoDesejado )
```

```
            {
              tempo = millis();
            }

          }

        }

        this -> alturaAtual = this -> alturaDesejada;
    }

    void  porcentagemAltura( float porcentagem )
    {
      this -> alturaDesejada = this -> alturaTotal * porcentagem;

    }

    // Logica presente na segunda tabela( com alturas diferentes e ira
mover a comparta anterior ao sensor)
    void padraoMovimentoAsc( float leitura )
    {
      if ( leitura > this -> alturaTotal * 0.752 )
      {
        this -> grauDesejado = 0;
        this -> alturaDesejada = alturaTotal;

      } else if ( leitura > this -> alturaTotal * 0.658 )
      {
        this -> grauDesejado = 22;
        this -> alturaDesejada = this -> alturaTotal * 0.875;

      } else if ( leitura > this -> alturaTotal * 0.564 )
      {
        this -> grauDesejado = 45;
        this -> alturaDesejada = this -> alturaTotal * 0.750;

      } else if ( leitura > this -> alturaTotal * 0.470 )
      {
        this -> grauDesejado = 67;
```

```
        this -> alturaDesejada = this -> alturaTotal * 0.625;

      } else if ( leitura > this -> alturaTotal * 0.376 )
      {
        this -> grauDesejado = 90;
        this -> alturaDesejada = this -> alturaTotal * 0.500;

      } else if ( leitura > this -> alturaTotal * 0.282 )
      {
        this -> grauDesejado = 112;
        this -> alturaDesejada = this -> alturaTotal * 0.375;

      } else if ( leitura > this -> alturaTotal * 0.188 )
      {
        this -> grauDesejado = 135;
        this -> alturaDesejada = this -> alturaTotal * 0.250;

      } else if ( leitura > this -> alturaTotal * 0.094 )
      {
        this -> grauDesejado = 157;
        this -> alturaDesejada = this -> alturaTotal * 0.125;

      } else {
        this -> grauDesejado = 180;
        this -> alturaDesejada = 0;

      }
    }

    // Logica presente na primeira e segunda tabela(com alturas
diferentes e ira mover a comparta posterior ao sensor)
    void padraoMovimentoDesc( long leitura ) {
      if ( leitura > this -> alturaTotal * 0.752 )
      {
        this -> grauDesejado = 180;
        this -> alturaDesejada = 0.0;

      } else if ( leitura > this -> alturaTotal * 0.658 )
      {
```

```
        this -> grauDesejado = 157;
        this -> alturaDesejada = this -> alturaTotal * 0.125;

      } else if ( leitura > this -> alturaTotal * 0.564 )
      {
        this -> grauDesejado = 135;
        this -> alturaDesejada = this -> alturaTotal * 0.250;

      } else if ( leitura > this -> alturaTotal * 0.470 )
      {
        this -> grauDesejado = 112;
        this -> alturaDesejada = this -> alturaTotal * 0.375;

      } else if ( leitura > this -> alturaTotal * 0.376 )
      {
        this -> grauDesejado = 90;
        this -> alturaDesejada = this -> alturaTotal * 0.500;

      } else if ( leitura > this -> alturaTotal * 0.282 )
      {
        this -> grauDesejado = 67;
        this -> alturaDesejada = this -> alturaTotal * 0.625;

      } else if ( leitura > this -> alturaTotal * 0.188 )
      {
        this -> grauDesejado = 45;
        this -> alturaDesejada = this -> alturaTotal * 0.750;

      } else if ( leitura > this -> alturaTotal * 0.094 )
      {
        this -> grauDesejado = 22;
        this -> alturaDesejada = this -> alturaTotal * 0.875;

      } else {
        this -> grauDesejado = 0;
        this -> alturaDesejada = this -> alturaTotal;
      }
    }
};
```

O quadro 3 demonstra o código do corpo principal da automação. Esse código é o mais importante, pois ele é responsável por instanciar todos os objetos utilizados, controlar a utilização as utilizações deles, configurar a interface criada no ArduinoXY e fazer a conexão dela com um telefone móvel via Wifi.

Quadro 3 – Codificação do corpo principal da automação

```
Sensor *sensorA;
Sensor *sensorB;
Sensor *sensorC;

Comporta *comportaA;
Comporta *comportaB;

int modoManual;

float alturaComporta = 4.5;
float diferencaLeitura;
float mediaLeitura;
float calculoManual;

// Remote XY ////////////////////////////////////////////////////////
#define REMOTEXY_MODE__ESP32CORE_WIFI_POINT
#include <WiFi.h>

#include <RemoteXY.h>

#define REMOTEXY_WIFI_SSID "esp32"
#define REMOTEXY_WIFI_PASSWORD "senha12345"
#define REMOTEXY_SERVER_PORT 6377


#pragma pack(push, 1)
uint8_t RemoteXY_CONF[] =
{ 255, 3, 0, 60, 0, 60, 1, 13, 21, 4,
  130, 1, 0, 12, 100, 51, 0, 68, 130, 0,
  0, 12, 100, 51, 1, 194, 66, 1, 8, 19,
  7, 16, 1, 6, 31, 66, 1, 61, 19, 7,
  16, 1, 6, 31, 66, 1, 33, 41, 7, 16,
```

```
  1, 6, 31, 67, 5, 45, 49, 20, 5, 1,
  31, 26, 11, 67, 5, 20, 28, 20, 5, 1,
  31, 26, 11, 67, 5, 73, 28, 20, 5, 1,
  31, 26, 11, 129, 0, 19, 22, 22, 5, 1,
  24, 83, 101, 110, 115, 111, 114, 32, 65, 0,
  129, 0, 72, 22, 22, 5, 1, 24, 83, 101,
  110, 115, 111, 114, 32, 66, 0, 129, 0, 44,
  43, 22, 5, 1, 24, 83, 101, 110, 115, 111,
  114, 32, 67, 0, 130, 0, 0, 10, 100, 2,
  0, 26, 131, 1, 2, 2, 46, 5, 1, 6,
  31, 83, 101, 110, 115, 111, 114, 101, 115, 0,
  131, 0, 54, 2, 44, 5, 2, 64, 31, 67,
  111, 109, 112, 111, 114, 116, 97, 115, 0, 2,
  1, 74, 13, 24, 7, 2, 2, 31, 24, 24,
  65, 117, 116, 111, 0, 77, 97, 110, 117, 97,
  108, 0, 129, 0, 38, 15, 34, 4, 2, 24,
  77, 111, 100, 111, 32, 65, 117, 116, 111, 109,
  195, 161, 116, 105, 99, 111, 0, 66, 65, 11,
  32, 18, 26, 2, 26, 31, 66, 65, 63, 32,
  16, 25, 2, 26, 31, 67, 5, 31, 23, 14,
  4, 2, 31, 24, 11, 67, 5, 79, 23, 14,
  4, 2, 31, 24, 11, 4, 64, 35, 31, 6,
  30, 2, 2, 26, 129, 0, 5, 23, 24, 4,
  2, 24, 67, 111, 109, 112, 111, 114, 116, 97,
  32, 65, 58, 0, 4, 64, 85, 31, 6, 29,
  2, 2, 26, 129, 0, 54, 23, 24, 4, 2,
  24, 67, 111, 109, 112, 111, 114, 116, 97, 32,
  66, 58, 0
};

// this structure defines all the variables and events of your control
interface
struct {

  // input variables
  uint8_t modoManual; // =1 if switch ON and =0 if OFF
  int8_t controleComportaA; // =0..100 slider position
  int8_t controleComportaB; // =0..100 slider position
```

```
  // output variables
  int8_t indicadorSensorA; // =0..100 level position
  int8_t indicadorSensorB; // =0..100 level position
  int8_t indicadorSensorC; // =0..100 level position
  char textoSensorC[11];  // string UTF8 end zero
  char textoSensorA[11];  // string UTF8 end zero
  char textoSensorB[11];  // string UTF8 end zero
  int8_t indicadorComportaA; // =0..100 level position
  int8_t indicadorComportaB; // =0..100 level position
  char textoComportaA[11];  // string UTF8 end zero
  char textoComportaB[11];  // string UTF8 end zero

  // other variable
  uint8_t connect_flag;  // =1 if wire connected, else =0

} RemoteXY;
#pragma pack(pop)

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

void setup() {
  RemoteXY_Init ();
  Serial.begin(9600);

  RemoteXY.modoManual = 1;

  sensorA = new Sensor(33, 32);
  sensorB = new Sensor(23, 22);
  sensorC = new Sensor(19, 18);

  comportaA = new Comporta(27);
  comportaB = new Comporta(4);
}

void loop() {
  RemoteXY_Handler();

  // Sensor A - Leitura e visualizacao dos dados na GUI -----------------
  sensorA -> leitura();
```

```
  dtostrf( sensorA -> distanciaCm, 0, 2, RemoteXY.textoSensorA );

  calculoManual = round( ( sensorA -> distanciaCm / alturaComporta ) *
100.0 );

  calculoManual = 100.0 - calculoManual;

  RemoteXY.indicadorSensorA = int( calculoManual );

  // ------------------------------------------------------------------


  // Sensor B - Leitura e visualizacao dos dados na GUI ----------------
  sensorB -> leitura();

  dtostrf( sensorB -> distanciaCm, 0, 2, RemoteXY.textoSensorB );

  calculoManual = round( ( sensorB -> distanciaCm / alturaComporta) *
100.0 );

  calculoManual = 100.0 - calculoManual;

  RemoteXY.indicadorSensorB = int( calculoManual );

  // ------------------------------------------------------------------


  // Sensor C - Leitura e visualizacao dos dados na GUI ----------------
  sensorC -> leitura();

  dtostrf( sensorC -> distanciaCm, 0, 2, RemoteXY.textoSensorC );

  calculoManual = round( ( sensorC -> distanciaCm / alturaComporta) *
100.0 );

  calculoManual = 100.0 - calculoManual;

  RemoteXY.indicadorSensorC = int( calculoManual );
```

```
  // ------------------------------------------------------------------


  if ( RemoteXY.modoManual == 1 )
  {
    // Comparacao sensor A e B ____________________________________________
    diferencaLeitura = sensorA -> distanciaCm - sensorB -> distanciaCm;

    diferencaLeitura = abs( diferencaLeitura );

    // Identifica qual sera a logica utilizada ---------------------------

    if ( diferencaLeitura > 0.93 )
    {
      if ( sensorA -> distanciaCm < sensorB -> distanciaCm )
      {
        comportaA -> padraoMovimentoAsc( sensorA -> distanciaCm );

      } else {
        comportaA -> padraoMovimentoDesc( sensorB -> distanciaCm );

      }
    } else {
      mediaLeitura = ( sensorA -> distanciaCm + sensorB -> distanciaCm )
/ 2;

      comportaA -> padraoMovimentoAsc( mediaLeitura );
    }

    // ------------------------------------------------------------------

    comportaA -> moveComporta();

    // Exibe dados sobre a comporta na GUI -------------------------------

    dtostrf( comportaA -> alturaAtual, 0, 2, RemoteXY.textoComportaA );
```

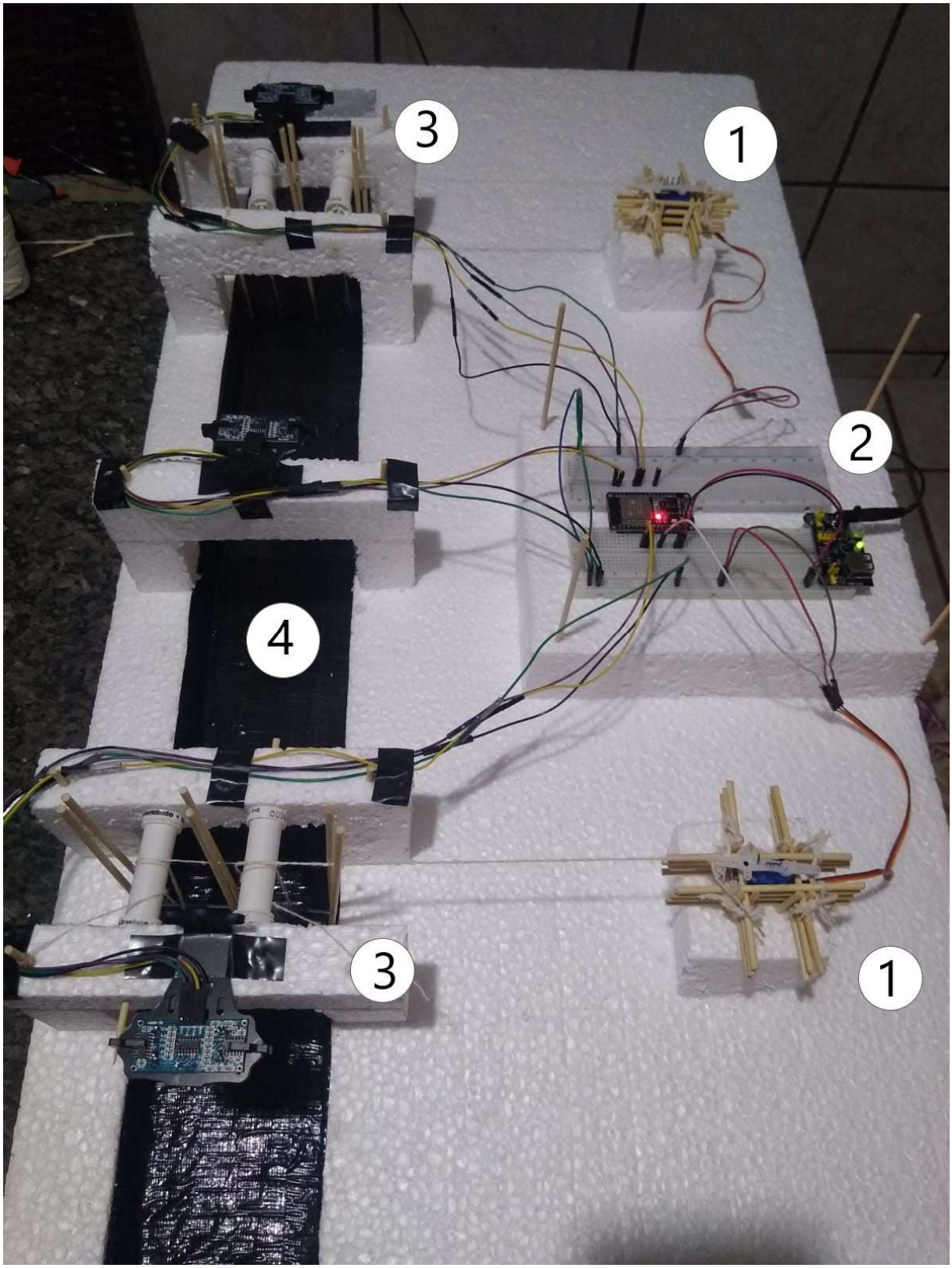
```
    calculoManual = round( ( comportaA -> alturaAtual / alturaComporta )
* 100.0 );

    RemoteXY.indicadorComportaA = int( calculoManual );

    // ---------------------------------------------------------------

    // _____________________________________________________________

    // Comparacao sensor B e C _____________________________________
    diferencaLeitura = sensorB -> distanciaCm - sensorC -> distanciaCm;

    diferencaLeitura = abs( diferencaLeitura );

    // Identifica qual sera a logica utilizada ----------------------

    if ( diferencaLeitura > 0.93 )
    {
      if ( sensorB -> distanciaCm < sensorC -> distanciaCm )
      {
        comportaB -> padraoMovimentoAsc( sensorB -> distanciaCm );

      } else {
        comportaB -> padraoMovimentoDesc( sensorC -> distanciaCm );

      }
    } else {
      mediaLeitura = ( sensorB -> distanciaCm + sensorC -> distanciaCm )
/ 2;

      comportaB -> padraoMovimentoAsc( mediaLeitura );
    }

    // ---------------------------------------------------------------

    comportaB -> moveComporta();

    // Exibe dados sobre a comporta na GUI ----------------------------
```

```
    dtostrf( comportaB -> alturaAtual, 0, 2, RemoteXY.textoComportaB );

    calculoManual = round( ( comportaB -> alturaAtual / alturaComporta )
* 100.0);
    RemoteXY.indicadorComportaB = int( calculoManual );

    // ---------------------------------------------------------------

    // _______________________________________________________________

  } else {
    // Comporta A _____________________________________________________

    // Recebe e trata o dado de altura da comporta definida pelo usuario

    calculoManual = float( RemoteXY.controleComportaA ) / 100.0;
    comportaA -> porcentagemAltura( calculoManual );

    calculoManual = round( calculoManual * 180.0 );
    calculoManual = 180 - calculoManual;
    comportaA -> grauDesejado = int( calculoManual );

    // ---------------------------------------------------------------

    comportaA -> moveComporta();

    // Exibe os dados sobre a comporta para o usuario ----------------

    dtostrf( comportaA -> alturaAtual, 0, 2, RemoteXY.textoComportaA );

    calculoManual = round( ( comportaA -> alturaAtual / alturaComporta )
* 100.0 );
    RemoteXY.indicadorComportaA = int( calculoManual );

    //----------------------------------------------------------------

    // _______________________________________________________________

    // Comporta B _____________________________________________________
```

```
    // Recebe e trata o dado de altura da comporta definida pelo usuario

    calculoManual = float( RemoteXY.controleComportaB ) / 100.0;
    comportaB -> porcentagemAltura( calculoManual );

    calculoManual = round( calculoManual * 180.0 );
    calculoManual = 180 - calculoManual;
    comportaB -> grauDesejado = int( calculoManual );

    // ------------------------------------------------------------------

    comportaB -> moveComporta();

    // Exibe os dados sobre a comporta para o usuario --------------------

    dtostrf( comportaB -> alturaAtual, 0, 2, RemoteXY.textoComportaB );

    calculoManual = round( ( comportaB -> alturaAtual / alturaComporta )
* 100.0 );
    RemoteXY.indicadorComportaB = int( calculoManual );

    //-------------------------------------------------------------------

    // ___________________________________________________________________
  }
}
```

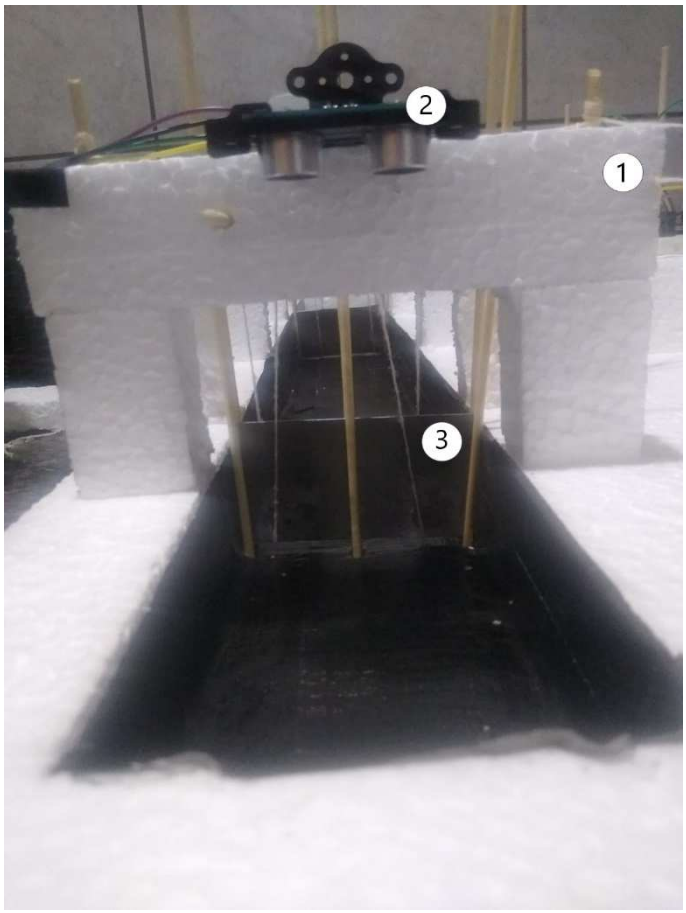
A conexão via *wi-fi* foi realizada através biblioteca disponibilizada pelo RemoteXY, que inicia o modo *wi-fi* como *Soft Access Point*. Esse modo permite que o ESP32 funcione como uma espécie de servidor/roteador ou *hotspot*. Ele disponibiliza uma rede de *wi-fi* na qual os dispositivos conseguem se conectar e através dela transferir informações.

Figura 8 – Protótipo desenvolvido em uma maquete



A figura 8 ilustra o protótipo desenvolvido em uma maquete de aproximadamente 1 m de comprimento, 50 cm de largura e 20 cm de largura. O número 1 representa o servo motor fixado com o auxílio de palitos de churrasco e barbante. Esse segue em direção ao suporte, número 3, onde passa pelos carreteis e embaixo da comporta. Quando o motoro rotaciona, acontece uma tração do barbante consegue criar uma movimentação das placas de alumínio. O número 2 representa as protoboards utilizadas na protótipo, através delas foi realizado as conexões entre os atuadores e microcontrolador, além de fornecer energia a todos os componentes com o auxílio da fonte. O número 4 representa o córrego em que a lógica será aplicada, ele mede aproximadamente 90 cm de largura 10 cm de cumprimento e 4,5 cm de profundidade.

Figura 9 – Componentes presentes no suporte



A figura 9 apresenta uma visão mais aproximada do suporte (Número 1). Em média os suportes apresentam uma altura igual a 10cm. Nele também é possível verificar a existência do sensor ultrassónico (Número 2), que estão a uma distância média de 9 cm do topo do córrego,

e uma placa de alumino (Número 3), que está envolta em barbante e cercada por palitos a fim de manter o movimento dela o mais retilíneo possível.

Tabela 4 – Testes de funcionamento dos sensores

Nº	**Distância do sensor (cm)**								
	Sensor A			Sensor B			Sensor C		
	Ideal	Real	Diferença	Ideal	Real	Diferença	Ideal	Real	Diferença
1	4,39	4,3	0,09	4,42	4,4	0,02	4,5	4,5	0
2	3,12	3,0	0,12	4,42	4,4	0,02	4,5	4,5	0
3	3,11	3,0	0,11	3,26	3,4	0,14	4,5	4,5	0
4	3,12	3,0	0,12	3,21	3,4	0,19	3,23	3,1	0,13
5	1,53	1,2	0,33	4,42	4,4	0,02	4,5	4,5	0
6	1,53	1,2	0,33	2,89	2,7	0,19	4,5	4,5	0
7	3,58	2,6	0,98	0,81	1,1	0,29	4,5	4,5	0
8	4,39	4,4	0,01	0,81	1,2	0,39	3,46	2,7	0,76
9	4,37	4,3	0,07	2,46	2,8	0,34	2,26	1,6	0,66
10	0	0	0	4,30	4,4	0,10	4,50	4,5	0
Total			2,16			1,7			1,55
Média			**0,216**			**0,170**			**0,155**

Tabela 5 – Testes de funcionamento dos servo motores no modo automático

Nº	**Altura submersa (cm)**					
	Comporta A			Comporta B		
	Ideal	Real	Diferença	Ideal	Real	Diferença
1	4,5	4,5	0	4,5	4,5	0

2	3,94	4,2	0,26	4,5	4,5	0
3	3,94	4,2	0,26	3,94	4,5	0,56
4	3,94	4,2	0,26	3,94	3,1	0,84
5	1,69	2,3	0,61	4,5	4,5	0
6	1,69	2,3	0,61	3,37	4,5	1,13
7	4,5	4,5	0	0,56	4,5	3,94
8	4,5	4,4	0,1	0,56	2,7	2,14
9	2,25	2,7	0,45	3,37	1,6	1,77
10	0	1,3	1,3	4,5	4,5	0
Total			3,85			10,38
Média			**0,385**			**1,038**

Tabela 6 - Testes de funcionamento dos servo motores no modo manual

Nº	**Altura submergida (cm)**					
	Comporta A			Comporta B		
	Ideal	Real	Diferença	Ideal	Real	Diferença
1	0	3,4	3,4	0	2,2	2,2
2	1,03	2,0	0,97	1,66	2,1	0,44
3	2,12	3,0	0,88	1,03	1,5	0,47
4	3,51	4,0	0,49	2,52	3,2	0,68
5	1,85	2,8	0,95	3,51	4,3	0,79
6	3,15	3,8	0,65	1,85	2,3	0,45
7	1,39	2,3	0,91	2,75	3,3	0,55
8	4,10	4,3	0,20	0,94	1,5	0,56

9	3,24	4,2	0,96	2,30	2,7	0,4
10	4,50	4,4	0,10	4,50	4,3	0,2
Total			9,51			6,74
Média			**0,951**			**0,674**

A tabela 4 e 5 são tabelas complementares que representam os testes realizados no protótipo no modo automático. Já a tabela 6, são testes separados, pois o modo manual não leva em conta a leitura dos sensores. Nessas tabelas, o tamanho ideal seria baseado na lógica que foi apresentada, e foi coletado com o auxílio da interface gráfica. Já o valor real foi coletado através de medições físicas na maquete.

8 Conclusão

Através do cálculo da média da diferença encontrada entre o valor ideal e o real, foi possível constatar que o funcionamento do sensores ocorreu próximo ao esperado, afetando apenas levemente a lógica do sistema. O que não aconteceu em relação ao servo motor, a variação na execução foi muito alta, o que poderia prejudicar completamente o algoritmo e o intuito do projeto em controlar a vazão da água.

Uma das possíveis causas dessas diferença no funcionamento nos componentes seria o fato da maquete não apresentar medidas exatas por toda a sua extensão. Embora durante a sua construção tenha sido feito diversas medias, é impossível para o ser humano fazer com que todas as proporções sejam perfeitas.

Uma outra causa poderia ser o modo com que as comportas são movimentadas, os servos motores fazem um movimento circular ao serem acionados, isso pode causar um diferença entre força de tração nos fios, fazendo com que parte do movimento seja perdido.

Para que o trabalho atual realmente consiga comprovar a lógica desenvolvida seria necessário refazer todo o protótipo utilizando apenas peças feitas sob medida e alterar a forma de movimento do motor para que seja apenas retilíneo. Dessa forma ambos os possíveis problemas relatados seriam resolvidos.

Além de realizar as mudanças discutidas, uma melhoria para o protótipo seria adicionar indicadores visuais de funcionamento, como por exemplo, LEDs que indicam a

direção do movimento e quando o sistema se encontra ativo. Dessa forma mesmo que ele atinja uma maior proporção, seria fácil de visualizar seu funcionamento sem o auxílio da aplicação desenvolvida no celular.

Uma outro melhoramento seria criar uma função de teste no aplicativo de controle, dessa forma seria possível testar o funcionamento do protótipo sem a necessidade de alterar manualmente a leitura dos sensores. Isso teria um maior efeito quando o sistema apresentar dimensões maiores.

Referências

ARDUINO IDE – O SOFTWARE PARA GRAVAÇÃO DE CÓDIGOS NO ARDUINO. **USINAINFO**, 2019. Disponível em: <https://www.usinainfo.com.br/blog/arduino-ide-o-software-para-gravacao-de-codigos-no-arduino/>. Acesso em: 17 mar. 2021.

DIDOVETS, Julii, et al. Climate change impact on regional floods in the Carpathian region. **Journal of Hydrology: Regional Studies**, v. 22, p 100590, 16 feb. 2019

EVANS, Martins; NOBLE, Joshua; HOCHENBAUM, Jordan. **Arduino em Ação**. 1 ed. São Paulo: Novatec Editora Ltda., 2013.

FITZGERALD, Gerry; TARRANT, Mike; AITKEN, Peter; FREDRIKSEN, Marie. **Disaster Health Management: A Primer for Students and Practitioners**. 1a ed. Oxford: Routledge, 2016

HAJJAJ, S.S.H.; SULTAN, M.T.H.; MOKTAR, M.H.; LEE, S.H.. Utilizing the Internet of Things (IoT) to Develop a Remotely Monitored Autonomous Floodgate for Water Management and Control. **Water**, v.12, n. 2, p. 502, 17 fev. 2020

HODGKINS, G.A.; DUDLEY, R.W.; ARCHFIELD,S.A.; RENARD, B.. Effects of climate, regulation, and urbanization on historical flood trends in the United States. **Journal of Hydrology**, v.573, p. 697-709, 01 abr 2019

KAVITHA, M.; NAYAGAM, V.; JAYAPRAKASH, S.; NIRMALRAJ, S.; SHARMILA, M.; ALIMALAR, N.. Dam Flood Prevention through Data Analysis using Automatic Control System. **EAI**, 27 feb. 2021

LATHA, N. Anju; MURTHY, B. Rama; KUMAR, K. Bharat. Distance Sensing with Ultrasonic Sensor and Arduino. **International Journal of Advance Research, Ideas and Innovations in Technology**, v.2, n. 5, set. - out. 2016

MAIER, A.; SHARP, A.; VAGAPOV, Y.. Comparative analysis and practical implementation of the ESP32 microcontroller module for the internet of things, **2017 Internet Technologies and Applications (ITA)**, p. 143-148, 9 nov. 2017

PATERSON, David L.; WRIGHT, Hugh; HARRIS, Patrick N. A.. Health Risks of Flood Disasters. **Clinical Infectious Diseases**, v. 67, n. 9, p. 1450-1454, 1 nov. 2018

Pesquisa Nacional de Saneamento Básico: Tabela 2245 - Número de municípios, total e os que sofreram inundações ou enchentes nos últimos cinco anos, por fatores agravantes, **SIDRA: Banco de Tabelas Estatísticas**, 2010. Disponível em: < https://sidra.ibge.gov.br/Tabela/2245>. Acesso em: 06 jun. 2021.

Pesquisa Nacional de Saneamento Básico: Tabela 2246 - Número de municípios, total e os que sofreram inundações ou enchentes nos últimos dois anos e Extensão das áreas onde ocorreram inundações ou enchentes, **SIDRA: Banco de Tabelas Estatísticas**, 2005. Disponível em: < https://sidra.ibge.gov.br/Tabela/2246>. Acesso em: 06 jun. 2021.

PONTES, M. L.C.; LIMA, A. M. M.; JÚNIOR, J. de A. S.;SADECK, C. C. de A.. Dinâmica das áreas de várzea do município de Belém/PA e a influência da precipitação pluviométrica na formação de pontos alagamentos. **Caderno de Geografia**, v. 27, n. 49, p. 285-303. abr. – maio 2017

SHAH, Syed; MUSTAFFA, Zahiraniza; WAN YUSOF, Khamaruzaman. Disasters Worldwide and Floods in the Malaysian Region: A Brief Review. **Indian Journal of Science and Technology**, v. 10, n. 2, 20 jan. 2017

Software Arduino (IDE). **ARDUINO**, 2018. Disponível em: <https://www.arduino.cc/en/Guide/Environment/>. Acesso em: 14 mar. 2021.

STRAUB, Matheus Gebert. ARDUINO IDE – O SOFTWARE PARA GRAVAÇÃO DE CÓDIGOS NO ARDUINO. **USINAINFO**, 2019. Disponível em: <https://www.usinainfo.com.br/blog/arduino-ide-o-software-para-gravacao-de-codigos-no-arduino/>. Acesso em: 17 mar. 2021.

UNZER MACEDO, E. Os rios e a história. **Revista Científica Foz**, v. 1, n. 1, p. 10, 9 out. 2018.

WAHYUNI, R.; SENTANA, J. T.; MUHARDI, M.; IRAWAN, Y.. Water Level Control Monitoring Based On Arduino Uno R3 Atmega 238p Using Lm016l LCD at STMIK Hang Tuah Pekanbaru. **Journal of Robotics and Control (JRC)**, v. 2, n. 4, p. 265-269, jul. 2021