UNIEVANGÉLICA

CURSO DE ENGENHARIA CIVIL

MARCELO WEGENER POSSAMAI

ESTUDO COM PROTÓTIPO DO EFEITO DO VÁCUO NO PROCESSO DE DECANTAÇÃO: ETA SANEAGO – ANÁPOLIS

ANÁPOLIS / GO
2018

MARCELO WEGENER POSSAMAI

ESTUDO COM PROTÓTIPO DO EFEITO DO VÁCUO NO PROCESSO DE DECANTAÇÃO: ETA SANEAGO –ANÁPOLIS

TRABALHO DE CONCLUSÃO DE CURSO SUBMETIDO AO CURSO DE ENGENHARIA CIVIL DA UNIEVANGÉLICA

ORIENTADOR: EDUARDO DOURADO ARGOLO
COORIENTADOR: WILLIAN PEREIRA DOS SANTOS JÚNIOR

ANÁPOLIS / GO
2018

FICHA CATALOGRÁFICA

POSSAMAI, MARCELO WEGENER

Estudo com protótipo do efeito do vácuo no processo de decantação: ETA Saneago – Anápolis

107P, 297 mm (ENC/UNI, Bacharel, Engenharia Civil, 2017).

TCC - UniEvangélica.

Curso de Engenharia Civil.

1. Decantação
2. Níveis de qualidade da água
3. Arduino
4. Automação com Arduino

I. ENC/UNI

II. Estudo com protótipo do efeito do vácuo no processo de decantação: ETA Saneago – Região de Anápolis

REFERÊNCIA BIBLIOGRÁFICA

POSSAMAI, Marcelo Wegener. Estudo com protótipo do efeito da temperatura e pressão no processo de sedimentação: ETA Saneago – Anápolis. TCC, Curso de Engenharia Civil, UniEvangélica, Anápolis, GO, 107 p. 2018.

CESSÃO DE DIREITOS

Marcelo Wegener Possamai

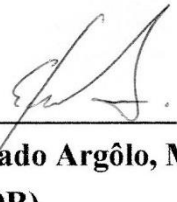
E-mail: marcelo.wegener@gmail.com

MARCELO WEGENER POSSAMAI

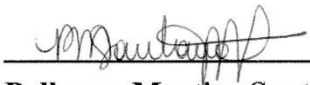
ESTUDO COM PROTÓTIPO DO EFEITO DO VÁCUO NO PROCESSO DE DECANTAÇÃO: ETA SANEAGO – ANÁPOLIS

TRABALHO DE CONCLUSÃO DE CURSO SUBMETIDO AO CURSO DE ENGENHARIA CIVIL DA UNIEVANGÉLICA COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL

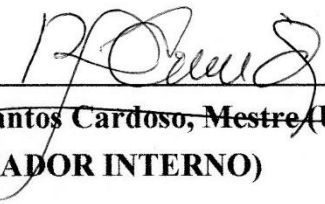
APROVADO POR:



Eduardo Dourado Argôlo, Mestre (UniEvangélica)
(ORIENTADOR)



Pollyana Martins Santana Guimarães, Mestra (UniEvangélica)
(EXAMINADOR INTERNO)



Rogério Santos Cardoso, Mestre (UniEvangélica)
(EXAMINADOR INTERNO)

DATA: ANÁPOLIS/GO, 06 de Junho de 2018.

AGRADECIMENTOS

Agradecimentos fortes a Deus, por sempre ser guiado a obter mais sabedoria e usufruir dela para o bem. Agradecimentos a meu orientador, mestre Eduardo Dourado Argolo, que me proporcionou grande parte de seu tempo e me privilegiou com a oportunidade de abordar um grande tema como este. Agradecimentos a minha noiva, Letícia De Césaro, pelos incentivos no processo. Agradecimentos a meus pais, Ireneu Possamai e Rosani Beatriz Wegener, por terem me criado e educado com muito carinho e respeito, e terem me proporcionado uma excelente oportunidade de cursar uma das melhores faculdades de Engenharia do Brasil.

Agradecimentos em especial também à equipe do curso de Mecânica, a qual prestou grande apoio, material e tempo. Agradecimentos a meu coorientador Wiliam, que me sanou dúvidas e dispôs de seu tempo. Agradecimentos ao professor mestre Rogério, que me auxiliou em partes complexas do código final.

Marcelo Wegener Possamai

RESUMO

Neste trabalho foram coletadas amostras da estação de tratamento de água (ETA) da empresa SANEAGO, onde o foco se destinava na parte de decantação, com um protótipo feito com a plataforma de programação Arduino. Foram mensurados os parâmetros de qualidade das amostras na entrada do protótipo e na saída, tais como: Ph, turbidez, cor e temperatura. No trabalho foram demonstrados todos os desafios na construção do protótipo, assim como a variação do objetivo inicial, uma vez que foi constatado que, para verificar todas as propostas iniciais, isso demandaria, além de um longo tempo de análise, um alto custo financeiro. Com toda a parte prática concluída foi verificada a eficiência do vácuo no processo de decantação.

PALAVRAS-CHAVE:

Estação de tratamento de água ETA. Processo de tratamento de efluentes. Processo de tratamento primário. Processo de sedimentação. Princípios físicos.

ABSTRACT

In this work, samples were collected from the water treatment plant (ETA) of the company SANEAGO, where the focus was on the decantation part, as well as a prototype made with the Arduino programming platform. The quality parameters of the samples were measured at the entrance prototype and output, such as: Ph, turbidity, color, temperature. In the work it was demonstrated all the challenges in the construction of the prototype, as well as the variation of the initial objective, since it was diverted the focus of the work when being verified that to verify all the initial proposals would require besides a long time of analysis, a high expense of money. With all the practical part concluded the vacuum efficiency in the decanting process was verified.

KEYWORDS:

ETA water treatment plant. Process of treatment of effluents. Primary treatment process, Sedimentation process. Physical principles.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ETA - Estação de tratamento de efluentes.

SST - Sólidos suspensos totais.

SSV - Sólidos voláteis.

ST - Sólidos totais.

SUMÁRIO

1 INTRODUÇÃO

A preocupação com o uso das águas não se data somente atualmente; foi esta que determinou a importância dos locais de desenvolvimento de grandes civilizações. Antigamente, o homem vivia como nômade, mudando de região em região, até que os recursos se esgotassem. Com o domínio da agricultura, começou a haver abundância de alimentos e, assim, o homem começou a se fixar em áreas produtivas com recursos de água e alimentos, aumentando seus descendentes e criando os primeiros povoados. As primeiras civilizações originaram-se na Mesopotâmia e no Egito, à margem de rios como o Eufrates e o Nilo (FABER, 2011).

Em Goiás também se teve a necessidade de água potável e tratada para os consumidores: em 1969 foi fundada a SANEAGO, empresa responsável pelo devido tratamento de água e esgoto das cidades goianas, a qual nasceu pelo "Sistema Financeiro do Saneamento do Banco Nacional da Habitação (BNH/SFS)", na então gestão de Pedro Ludovico Teixeira, que concedeu à empresa o direito de 30 anos de exploração.

Com o crescimento da população em Anápolis (GO) também se teve a necessidade de aumento do sistema de coleta de água e abastecimento das residências, sendo notório que, se for possível ter um sistema mais compacto é algo extremamente vantajoso, uma vez que, com um terreno reduzido, pode ser ocupado com maior eficiência pelo sistema de fornecimento e tratamento.

Em meio à necessidade é evidente que, com ferramentas certas, podemos analisar a possibilidade de princípios físicos poderem influenciar em grande parte certas partes do tratamento da água, como os tanques de sedimentação. Se estes princípios forem variáveis cruciais, podem ser destinados à reformulação do tratamento atual, que utiliza gravidade na decantação, ou até mesmo se dados evidenciarem o contrário, pode se ter o descarte na tentativa de disseminar tais métodos. Utilizar de artifícios para mudar um pouco o uso de químicos como metais pesados no tratamento reduziria drasticamente riscos de desenvolvimento de doenças destes oriundas.

O uso do Arduino como ferramenta de automação e monitoramento de dados é uma elegante ferramenta para verificação de processos que demoram quando analisados em laboratório. Com o Arduino pode-se ter o Ph e a turbidez das amostras de forma automática e atualizada, na entrada e na saída de sensores

usados para verificar a sedimentação com o aumento e a diminuição de forma isolada ou conjunta da temperatura e da pressão.

1.1 JUSTIFICATIVA

O tratamento do lodo produzido nas ETA's é um desafio constante. A busca de uma melhor performance e o reaproveitamento da água que é descartada é de grande valia para o meio ambiente e a própria companhia de abastecimento.

O processo de floculação é muito importante no tratamento de uma ETA, pois nele são coletados cerca de 60 a 90 % do lodo produzido. Neste lodo é encontrado 0,1 a 4% de ST, sendo deste 75% a 90% de SST e 20% a 35% de sólidos voláteis. (CUNHA, 2004). Se comprovada uma redução significativa do tempo de decantação com variantes físicas, podemos ter uma nova linha de entendimento a respeito da sedimentação, ou até mesmo a mudança do meio de decantação da linha atual de tratamento, para a conjuntura de novos métodos.

Quando se desenvolvem métodos mais eficazes de tratamento sem a adição de químicos, tais como os sulfatos de alumínios, que são amplamente utilizados, retiramos agentes causadores de doenças. O sulfato de alumínio, assim como outros metais pesados, gera, a longo prazo, doenças graves para a saúde, como câncer de estômago, doenças cerebrais ou até mesmo o avanço do quadro clínico de alguma delas, como o Alzheimer (FERREIRA P.C.; PIAI, K.A; TAKAYANAGUI, A.M.M.; SEGURA-MUÑOZ, S.I,, 2008).

1.2 OBJETIVOS

1.2.1 Objetivo geral

O objetivo deste trabalho é utilizar a plataforma de circuito integrado Arduino para controlar e monitorar a temperatura e o uso da pressão negativa, sendo ela retirada e gerando vácuo, na aplicação de um sistema de água bruta ou com adição de coagulantes no processo de tratamento de água.

Tendo o propósito de relacionar a velocidade de sedimentação das partículas suspensas nas amostras de efluentes da ETA da SANEAGO com as variantes físicas, esta proposta visa verificar a possibilidade de tornar o processo de coagulação e decantação mais intenso, sem a necessidade de aumento significante em suas dimensões (decantador), assim podendo atender com mais eficiência e reduzir a utilização de químicos no tratamento.

1.2.2 Objetivos específicos

- Demonstração dos vários usos do Arduino na área de saneamento.

- Construção de um protótipo automatizado, capaz de simular diversas condições específicas do tratamento atual e determinar dados desejados a respeito das amostras durante os testes.

- Análise do comportamento de um tanque modelo batelada de sedimentação submetido à variação de tempo na submissão de pressão negativa.

2 METODOLOGIA

Para a verificação dos objetivos específicos tem-se a necessidade do conhecimento da amostragem na entrada do experimento, e também do conhecimento da plataforma Arduino, que será utilizada para a feitura de toda a automação do processo, além desta ser utilizado para, em tempo real, monitorar todo o aspecto da amostra, desde a entrada até a saída do experimento.

Serão recolhidas também informações do tipo de amostra que será tratada, onde, com base em estudos científicos já realizados, poderemos ter uma noção da composição da amostragem para entender, por exemplo, que tipo de coagulante é utilizado e se há um outro material que compõe o objeto de estudo com alguma função, como a correção de Ph.

2.1 ESTAÇÃO ATUAL SANEAGO

As amostras serão retiradas na ETA SANEAGO em Anápolis em dois momentos do processo:

1) Entrada no sistema (água bruta).
2) Entrada dos misturadores de coagulantes (a montante dos decantadores).

Na figura 1 se observa a ETA SANEAGO de Anápolis.

Figura 1 - Estação SANEAGO

Fonte: autor.

A estação de tratamento de água SANEAGO do município de Anápolis se localiza na Rua Venezuela, número 155, no Bairro Jardim das Américas. Na estação de tratamento é seguido um modelo convencional, no qual é usada a decantação para a separação de sólidos e correções de ph, a fim de seguir os padrões técnicos.

2.1.1 Etapas do tratamento

1. Etapa de adição de sulfato de alumínio como coagulante e adição de cal para correção de Ph. Sequência de acontecimentos, conforme figuras 2, 3 e 4.

Figura 2 - Adição de sulfato de alumínio

Fonte: autor.

Figura 3 - Adição de cal

Fonte: autor.

Figura 4 - Entrada nos tanques de mistura e decantação

Fonte: autor.

2. Misturador para homogeneizar o produto de entrada com o sulfato e cal. Ilustração: figura 5.

Figura 5 – Misturador

Fonte: autor.

3. Decantação com passagens pelas colmeias para aderir os flóculos e facilitar a limpeza da água. Ilustração: figura 6.

Figura 6 - Decantação com colmeias

Fonte: autor.

4. Decantação por gravidade. Ilustração: figura 7.

Figura 7 - Decantação por gravidade

Fonte: autor.

5. Adição de cloro para controle biológico. Ilustração: figura 8.

Figura 8 - Adição de cloro para controle biológico

Fonte: autor.

6. Controle dos parâmetros de qualidade e envio aos tanques de abastecimento. Ilustrações: figuras 9, 10 e 11.

Figura 9 - Monitoramento de qualidade da água

Fonte: autor.

Figura 10 - Parâmetros de qualidade

SANEAGO — SANEAMENTO DE GOIÁS S.A

PLANO DE CONTROLE DA QUALIDADE DA ÁGUA

SIGLA DA UO: DA01P — NOME DA UO: ETA – ANÁPOLIS — Nº DA PÁGINA: 1 de 1

Cidade: ANÁPOLIS

PRODUTO INSPESIONADO	LOCAL DE AMOSTRAGEM	AMOSTRAGEM Nº	AMOSTRAGEM FREQ.	ANÁLISES	UNID.	ESPECIFICAÇÃO MIN.	ESPECIFICAÇÃO MAX.	QUEM FAZ AS ANÁLISES	AÇÕES EM CASO DE NÃO CONFORMIDADE DO PRODUTO	QUEM FAZ
Água Bruta	Câmara de Chegada	01	2h/2h	Temperatura	°C	Sem valor definido		Laboratório de Controle de Processos (ETA)	- Informar à Supervisão de Produção	Operador de Sistema
		01	1h/1h	Turbidez	uT	-	-		- Dependendo da gravidade paralisa o sistema	Supervisor de Produção
		01	2h/2h	pH	--	-	-		-Avaliando a gravidade, suspender bombeamento e comunicar a Gerência imediata, que tomará em conjunto com a P-GAQ/P-SPM e demais gerências envolvidas as providências cabíveis	Supervisão de Produção/GRS/SUINT/ P-GAQ/ P-SPM/DIPRO
		01	2h/2h	Cor Aparente	uH	-	-			
		01	-	*Ensaio de Floculação	mg/L	-	-			
		01	-	1 Alcalinidade	mg/L $CaCO_3$	-	-			
		01	-	1 Odor	N.L.O	Não objetável		Laboratório da GRS		
		01	-	1 Oxigênio Consumido	mg/L O_2	Sem valor definido -				
Água em Processo de Tratamento	Floculador	01	2h/2h	pH	--	5,4	7,0	Laboratório de Controle de Processos (ETA)	- Atuar no Processo - Avisar supervisor de Sistema - Adequar dosagem de produtos Químicos	Operador de Sistema
	Decantador	01	2h/2h	Turbidez	uT	10,0 uT ou 10% do valor de Água Bruta				
		01	2h/2h	Cor Aparente	uH	-	15,0			
	2 Filtro	01	2h/2h	Turbidez	uT	-	0,5			
Água Tratada	Saída da ETA	01	2h/2h	Temperatura	°C	Sem valor definido		Laboratório de Controle de Processos (ETA)	- Rever o Processo de Tratamento - Adequar dosagem de produtos Químicos - Comunicar ao Supervisor– 25% de resultados não-conformes em 24h - Emitir RACP – Investigação da Causa PR08.0019 - Emitir Autorização de Produção sob Desvio – PR08.0017/ - Negociar com a GRS/SUINT/ P-GAQ/DIPRO	Operador de Sistema
		01	1h/1h	Turbidez	uT	-	1,0			
		01	2h/2h	Cor Aparente	uH	-	15,0			
		01	2h/2h	pH	--	6,0	9,5			
		01	1h/1h	Flúor	mg/L F	0,7	1,0			Supervisor de Produção
		01	1h/1h	3 Cloro	mg/L Cl_2	0,5	2,0			
	Saída da ETA	01	Mensal	4 Ferro total	mg/L Fe	-	0,3	Laboratório da GRS	- Comunicar imediatamente ao Supervisor de Produção	Laboratorista da GRS
		01	Mensal	Alumínio	mg/L	-	0,2	01 análise Semanal de (Turbidez, Cloro, Cor, pH e Flúor)		
		01	4XSemana	Col. Totais	NMP	Ausência				
		01	4XSemana	E. Coli	NMP	Ausência				

Observações:
1 – O Parâmetro poderá ser realizado conforme a necessidade do sistema e a critério do responsável pela produção ou Técnico da GRS,
2 – Atender o que determina os artigos 30º e 31º, § 3º.
3 – Atender o que determina os artigos 32º, § 1º, art. 34º e anexo V – desinfetante e produtos secundarios da desinfecção.
4 – As unidades de tratamento que utilizam o Ortopolifosfato, atender o que determina o art. 39, § 4º.

VERSÃO: 21

Este Plano se baseia na Portaria nº 2914, de 12/12/2011 – do Ministério da Saúde, e o que determina o PR08.0015.

Responsável pela Aprovação: Gerência de Proteção Ambiental e Qualidade do Produto - P-GAQ

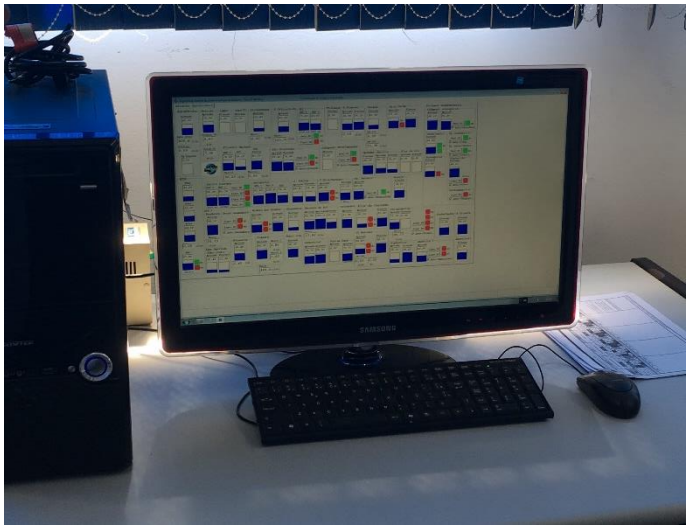
DATA: 23/05/2017

RUBRICA:

FR04.0081 — 04 — 30/06/2015 — PR08.0015

Fonte: autor.

Figura 11 - Envio para os tanques de abastecimento, tudo de forma automática



Fonte: autor.

2.2 FUNCIONAMENTO DO PROTÓTIPO

O processo de coagulação será implementado no protótipo simulando a mistura que ocorre no processo real, a mistura lenta e tempo de retenção dentro dos tanques variável. Com a implementação do vácuo e variação do tempo podemos estipular o seu efeito na formação do floco, na dosagem de água bruta. Verificando melhoria e proposta de decantação após injeção de coagulante na ETA.

2.3 ÁGUA E PADRÕES TÉCNICOS PARA CONSUMO

2.3.1 Turbidez

Por definição de KOWATA, RIBEIRO, & TELLES (2000), a turbidez da água é devido a matérias em suspensão tais como argila, silte e substâncias orgânicas, entre outras, que alteram a penetração da luz através da difusão e absorção, asim dando um aspecto turvo à água, contribuindo de forma análoga tanto como agente agressivo ao aspecto estético, como um composto protetor de agentes patogênicos contra desinfectantes.

Por definição de Metcalf (2003), turbidez é relacionada à reflexão de que um feixe de luz sofre atravessando uma solução contendo partículas suspensas e coloidais. Medidas de turbidez requerem uma fonte de luz (incandescente ou diodo emissor de luz) e um sensor para medir a luz refletida. Conforme a figura 1, item “a”, o sensor de luz fica localizado a uma

posição de 90 graus em relação à fonte luminosa. A turbidez medida é acrescida à medida que a luz emitida sofre maior reflexão. A turbidez é expressa em unidades nefelométricas de turbidez (UNT). A distribuição espacial e a intensidade da luz refletida, ilustradas na figura 12, item b (i = pequenas, ii = intermediárias e iii = grandes), dependerão do tamanho das partículas em relação ao comprimento de onda da fonte de luz. Para partículas menores que um décimo de comprimento de onda da luz incidente, a reflexão da luz é bem simétrica.

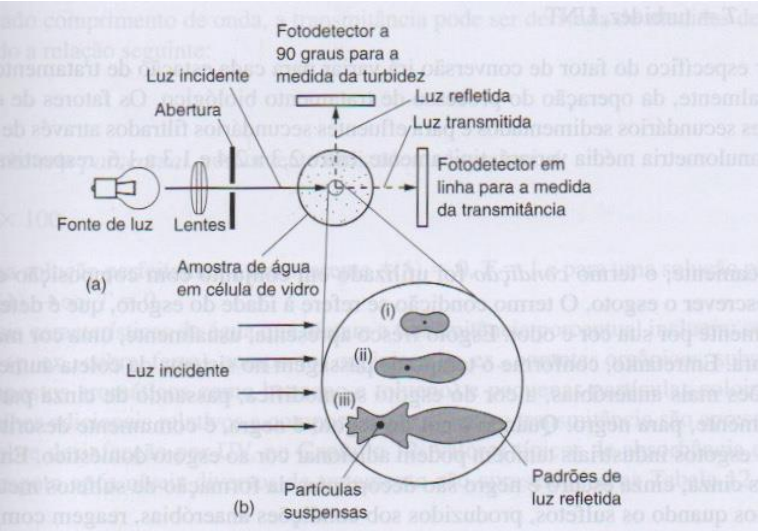
Algumas das limitações da medição de turbidez estão ligadas diretamente a aspectos físicos das partículas componentes presentes na solução ou meio aquoso; uma delas é que a luz emitida nesta não pode não apenas ser refletida como absorvida por eles, como no caso de uma solução de negro fumo, que retornará um valor de zero.

A turbidez não apresenta uma relação precisa com SSTs, mas existe uma relação aproximada com estes, onde, dependendo da origem do líquido, haverá valores variáveis para turbidez com SST.

De acordo com Franco (2009), turbidez é um parâmetro importante na aceitação ou rejeição da água tratada, uma vez que a turbidez integra um dos requisitos básicos de controle de potabilidade, na medida em que índices elevados podem determinar uma ineficácia do processo de tratamento. Com uma alta turbidez pode se ter diminuição do processo de desinfecção, uma vez que as partículas em suspensão dão certa proteção aos micro-organismos, assim prejudicando o tratamento com cloro.

Publicada pelo CONAMA, a portaria nº 2914, de 2011, estabelece padrões de qualidade da água para fornecimento. No anexo II tem-se a tabela que relata a turbidez e, entre seus padrões, estão a desinfecção de águas subterrâneas 1,0 uT para 95% das amostras, filtração rápida (tratamento completo ou filtração direta) de 0,5 uT para cada 95% das amostras e filtração lenta de 1,0 uT para cada 95% das amostras. Sendo a estação um tratamento completo e lento, temos que a entrega dessa água deve estar com no máximo 1,0 uT de turbidez.

Figura 12 - Representação da medição de turbidez



Fonte: METCALF, 2003.

2.3.2 Cor

Segundo Franco (2009), a cor da água teve por muito tempo uma indevida importância, já que era levada apenas em conta como padrão estético da água. Entretanto, a cor pode indicar e representar a presença de matéria orgânica: ácidos fúlvicos, húmicos e himatomelânicos.

A cor pode ter definições como cor aparente e cor verdadeira, sendo que, para se obter a cor verdadeira, deve-se levar a amostragem para centrifugação ou filtração para eliminar previamente a turbidez. Um exemplo de diferença de uma água pura com outra de coloração influenciada por sua composição é demonstrada na figura 13.

Conforme Kowata et al. (2000), a cor da água é devida em grande parte a ácidos húmicos, que se originam da decomposição de matérias orgânicas como plantas e animais. Isso também pode se dar por íons metálicos como plâncton, macrófitas e despejos industriais. A unidade de medida de cor é Hazen (uH) e é obtida quando se dissolve 1 mg de cloroplatinato de potássio e 0,5 mg de cloreto de cobalto em 1 litro de água destilada com ph 7.

Ainda segundo Kowata et al. (2000), é justificada uma diferença entre a cor verdadeira e a cor aparente, sendo a cor verdadeira a captada pelos olhos humanos e a cor aparente, devido à remoção da turbidez existente na água, de forma análoga, se justifica uma

generalização de que partículas com diâmetro superior a 1,2mm causam turbidez e, abaixo desse tamanho, classificados como cólides, causam cor.

Em se tratando de cor aparente, o CONAMA, em sua Portaria n° 2914, de 2011, anexo X, página 32D, estabelece padrões organolépticos de qualidade da água, onde, em uma das divisões tabeladas, determina a cor aparente apropriada para padrões de potabilidade de 15uH.

Figura 13: Água não pura e água pura

Fonte: Site Micro Ambiental.
Disponível em: < http://microambiental.com.br/servicos/monitoramento/controle-de-cor-agua/ >
Acesso em nov. 2017.

2.3.3 Temperatura

A temperatura, segundo Franco (2009), causa agitação das partículas, o que de tal forma aumenta a cinética das moléculas de água, determinando que temperaturas inferiores a 10ºC causam um retardamento do efeito dos coagulantes. Utilizando temperaturas superiores a esta pode-se ter mais choque entre as moléculas, aumentando a taxa de reação e, assim, tem-se uma melhor taxa de coagulação.

Em Kowata et al. (2000) é determinado papel importante da temperatura como em etapas como a hidrólise do coagulante, a eficiência do desinfectante na solubilidade dos gases, além de esta influenciar diretamente no desempenho de etapas do tratamento como mistura, floculação, decantação e filtração.

Metcalf (2003) exemplifica alguns laços entre a temperatura e vida biológica, sendo alguns desses a solubilidade do oxigênio diminuir conforme o aumento de temperatura, tendo uma água mais fria maior solubilidade de oxigênio em relação a uma água morna. Este autor também define que as atividades biológicas surtem mais efeitos a temperaturas entre 25°C e 35°C, e ainda cita que, quando a temperatura atinge um patamar de 50°C, a digestão anaeróbica e a nitrificação são interrompidas, e com temperaturas muito frias entre 5°C e 2°C há a dormência das bactérias.

2.3.4 PH

Em conformidade com Kowata et al. (2000), a água é definida com aspectos de seu Ph, onde o valor de 7 indica um meio neutro; acima deste, considera-se a água alcalina e abaixo, um meio ácido. Definindo tais padrões de Ph, ainda se leva em conta a definição de que alcalina é a capacidade de neutralizar ácidos e ácido é o inverso. Algumas afirmativas também são importantes quanto ao Ph: em meio ácido as ETA´s sofrem danos ao concreto e seus componentes, já que o ácido causa corrosão. Com quantidade alcalina, por sua vez, há um efetivo acúmulo de matéria nas tubulações.

Metcalf (2003) afirma que a maior parte dos constituintes, tanto de águas naturais como de esgoto, dependem da quantidade de concentração de íons de hidrogênio. O Ph se torna um dos padrões de qualidade, e a formulação padrão de Ph tem uma equação negativa logarítmica de base 10 da concentração de íons de hidrogênio.

$$\text{pH} = -log_{10}[H^+]$$

PAVANELLI & DISSERTAÇÃO (2001) demonstra que uma das principais funções de conhecer o Ph é a interferência deste no processo de coagulação, devido aos coagulantes se comportarem como ácidos. Assim sendo, é necessária sempre a adição de certa parte de alcalinizante para manter o sistema em equilíbrio.

No Site da COHESP pode-se ter uma noção abrangente do efeito da água com Ph´s diferentes. Muitas são as crenças acerca do efeito na área da saúde, como a cura de doenças gastrointestinais, mas como é relevado, isto são mitos, devido ao fato de que os estômagos dos seres humanos serem compostos de ácido cloridríco com Ph em torno de 2.5 a 3.0, o que é

um ácido extremamente forte. Com base nas informações, é ressaltado que a água mais adequada para o consumo humano tenha que ser a definida pelo Ministério da Saúde.

Segundo os padrões de qualidades estabelecidos pelo CONAMA n° 2914, de 2011, capitulo III, página 16D, tem-se que, para a rede de distribuição, o valor recomendado de Ph é estabelecido dentro da faixa de 6,00 a 9,50.

Com o Arduino podemos ter dados mais precisos sobre a medição de Ph, tornando seu uso relacionado com a de um peagômetro, cujos resultados dos Ph vão ser obtidos com o auxílio de uma sonda. É ilustrado o uso do Arduino na figura 14.

Figura 14: Sonda de Ph Arduino

Fonte: site karma-laboratory.
Disponível em: < http://karma-laboratory.com/workshops/arduinoforprogrammers/arduino_types.html> Acesso em nov. 2017.

2.4 ARDUINO

No contexto histórico, o Arduino, cujo nome se deve a um bar frequentado por membros docentes e alunos de Interection Design Institute na cidade de Ivre na Itália, foi desenvolvido em 2005 e, assim como muitas outras idéias novas, partiu de um problema preexistente, que era a necessidade do professor Massimo Banzi ensinar os estudantes de Design a trabalharem com tecnologia. De forma a sanar ao problema proposto, desenvolveram junto a demais colaboradores uma ferramenta que atendesse não só o que era necessário, mas também fosse de tal forma barata, como o que um estudante gastaria para comer uma pizza. (EVANS, NOBLE & HOCKENBAUN, 2013).

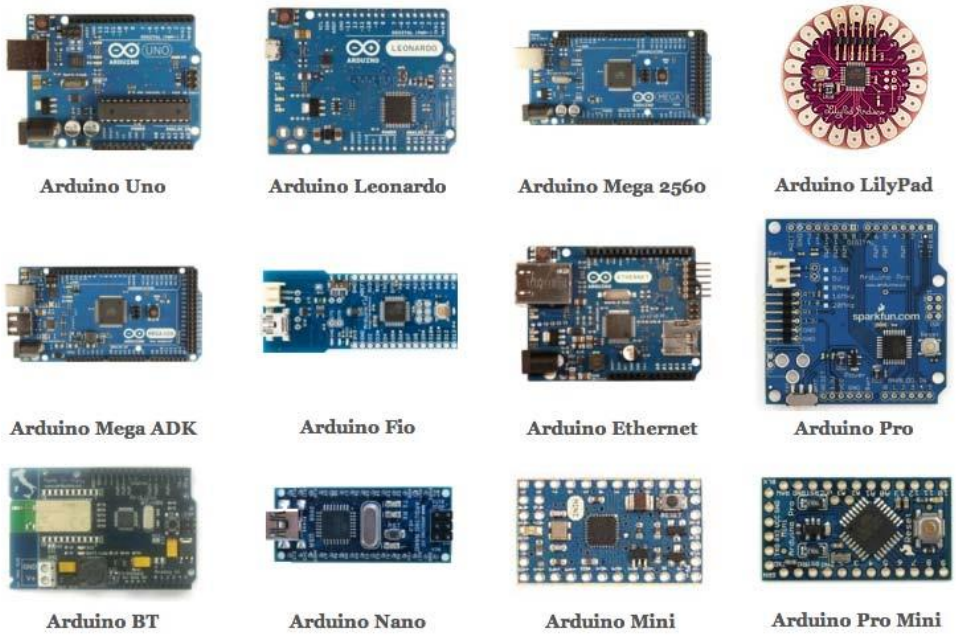
Arduino é uma plataforma de *hardware open source*, projetada sobre o microcontrolador Atmel AVR, podendo ser programada através de linguagem C/C++, permitindo leigos em eletrônica a elaborar projetos. É uma plataforma de fácil prototipação de projetos interativos embarcados com *software* e *hardware*. (OLIVEIRA & ZANETTI, 2015).

Com um *software* próprio de desenvolvimento do código chamado de IDE, são construidos de forma interativa todos os códigos, os quais, em várias situações, são encontrados prontos ou na maioria das vezes aplicados a projetos já realizados. Na IDE também se encontram exemplos prontos para demonstrar algumas funções, também ao incluir bibliotecas, às vezes com exemplos inclusos.

Os Arduinos originais são italianos; porém, atualmente, ja é notável a existência a longa lista de genéricos, os quais, na maioria das vezes, são mais baratos. Com a ampliação do uso do Arduino e com diversas intenções de usos, foram desenvolvidos modelos diferentes com processadores mais potentes, memória ampliada, tamanho reduzido, internet integrada e demais.

No protótipo aqui disposto vai ser utilizado o Arduino Mega+*Wifi*. Mesmo que existam vários outros Arduinos, este foi escolhido por possuir a possibilidade de se conectar ao computador por meio de um módulo integrado de wifi, onde é possível o envio de informações sem fio, além do mesmo possuir, segundo o *site* Embarcados, “54 pinos de entradas e saídas digitais, onde 15 destes podem ser utilizados como saídas PWM. Possui também 16 entradas analógicas e 4 portas de comunicação serial”. Na Figura 15 vêem-se alguns modelos de Arduino.

Figura 15 – Arduínos – Modelos de Arduino



Fonte: site karma-laboratory.
Disponível em: < http://karma-laboratory.com/workshops/arduinoforprogrammers/arduino_types.html> Acesso em set. 2017.

3 RESULTADOS

A construção do projeto do protótipo foi realizada com materiais adquiridos pelo próprio autor e fornecidos pela entidade Unievangélica.

3.1 EXECUÇÃO DO SISTEMA DE AUTOMAÇÃO E MATERIAIS

3.1.1 SENSORES

Trevisan & Poppi (2006) determinam que sensores são dispositivos capazes de relatar informações fisicas ou químicas de um determinado sistema, convertendo essa informação em um sinal elétrico. No principio, sensores são compostos de três elementos: receptor sensível (capta as informações do sistema), transdutor (transforma o sinal em sinal elétrico) e amplificador (amplifica intensidade do sinal).

Segundo Silva (2015), sensores são todos os dispositivos capazes de sentir determinada variação de grandeza física e a caracterizar com outra grandeza física, que, no caso do protótipo, é uma grandeza de sinal elétrico capaz de ser identificado pelo Arduino.

3.1.1.1 SENSOR DE FLUXO

O sensor de fluxo ou vazão, conforme as definições de Silva (2016), é a mistura de um rotor e sensor magnético, onde, ao rotacionar este envia pulsos de corrente. Assim, com a contagem de pulsos, pode-se aferir a quantidade de vezes em que o rotor sofreu rotação e, deste modo determinar a quantidade de líquido que está fluindo, uma vez que, para a água passar nas tubulações, esta deve rotacinar o rotor.

O sensor de fluxo adotado no experimento foi o SEA modelo F-S201, o qual opera com vazões mínimas de 1 litro e máxima de 30 litros por minuto. No protótipo, este será importante para determinr a quantidade de amostras a serem colocadas no tanque de experimento. O sensor utilizado pode ser observado na figura 16.

Figura 16: Sensor de fluxo

Fonte: autor, 2017

3.1.1.1.1 CALIBRAÇÃO

Para a correta medição, foi feito um código levando em conta o funcionamento do sensor pela contagem de pulsos; sendo assim, foi colocado um funil na entrada principal do sensor e foram descarregadas quantidades de 500 ml de água a cada teste, mudando sempre o valor da quantidade de pulsos.

3.1.1.1.2 CÓDIGO ISOLADO

```
byte sensorInterrupt = 0;  // 0 = digital pin 2
byte sensorPin       = 2;

// The hall-effect flow sensor outputs approximately 5.64 pulses per second per
// litre/minute of flow.
float calibrationFactor = 5.64;

volatile byte pulseCount;

float flowRate;
unsigned int flowMilliLitres;
unsigned long totalMilliLitres;

unsigned long oldTime;

unsigned int frac;

void setup()
{

  // Initialize a serial connection for reporting values to the host
  Serial.begin(9600);

  pinMode(sensorPin, INPUT);
  digitalWrite(sensorPin, HIGH);

  pulseCount        = 0;
  flowRate          = 0.0;
  flowMilliLitres   = 0;
```

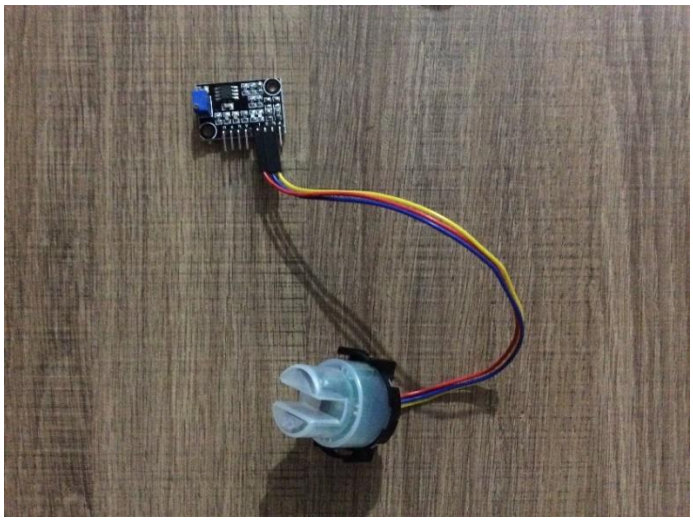
```
  totalMilliLitres  = 0;
  oldTime           = 0;

  // The Hall-effect sensor is connected to pin 2 which uses interrupt 0.
  // Configured to trigger on a FALLING state change (transition from HIGH
  // state to LOW state)
  attachInterrupt(sensorInterrupt, pulseCounter, FALLING);
}

/**
 * Main program loop
 */
void loop()
{
   // Print the cumulative total of litres flowed since starting
   Serial.print("  Output Liquid Quantity: ");             // Output separator
   Serial.print(getFLUXO());
   Serial.println("mL");
}

float getFLUXO(){

   if((millis() - oldTime) > 1000)    // Only process counters once per second
  {
   // Disable the interrupt while calculating flow rate and sending the value to
   // the host
   detachInterrupt(sensorInterrupt);

   // Because this loop may not complete in exactly 1 second intervals we calculate
   // the number of milliseconds that have passed since the last execution and use
   // that to scale the output. We also apply the calibrationFactor to scale the output
   // based on the number of pulses per second per units of measure (litres/minute in
   // this case) coming from the sensor.
   flowRate = ((1000.0 / (millis() - oldTime)) * pulseCount) / calibrationFactor;

   // Note the time this processing pass was executed. Note that because we've
   // disabled interrupts the millis() function won't actually be incrementing right
   // at this point, but it will still return the value it was set to just before
   // interrupts went away.
   oldTime = millis();
```

```
    // Divide the flow rate in litres/minute by 60 to determine how many litres have
    // passed through the sensor in this 1 second interval, then multiply by 1000 to
    // convert to millilitres.
    flowMilliLitres = (flowRate / 60) * 1000;

    // Add the millilitres passed in this second to the cumulative total
    totalMilliLitres += flowMilliLitres;

    // Determine the fractional part. The 10 multiplier gives us 1 decimal place.
    frac = (flowRate - int(flowRate)) * 10;

    // Reset the pulse counter so we can start incrementing again
    pulseCount = 0;

    // Enable the interrupt again now that we've finished sending output
    attachInterrupt(sensorInterrupt, pulseCounter, FALLING);
  }
   return totalMilliLitres;
  }
/*
Insterrupt Service Routine
 */
void pulseCounter()
{
  // Increment the pulse counter
  pulseCount++;
}
```

3.1.1.2 SENSOR DE TURBIDEZ

Sensor utilizado para medição da turbidez (figura 17).

Figura 17: Sensor de turbidez



Fonte: autor, 2017.

3.1.1.2.1 CALIBRAÇÃO

Para a calibração foi realizada a medição conjunta entre o aparelho pré-calibrado com a sonda de turbidez do Arduino, chegando a um código final.

3.1.1.2.2 CÓDIGO ISOLADO

```
#define SENSOR A0
#define N 10
double turbidty;

void setup() {
  // put your setup code here, to run once:
Serial.begin(9600);
pinMode( SENSOR,INPUT);
}
 int filtered;
 int vals[N];// VETOR COM N POSIÇÕES
```

```
void loop() {
  int value = analogRead(SENSOR);// FAZENDO A LEITURA
  // trocando os valores//
  for(int i = N -1; i > 0; i--){
    vals[i] = vals [i-1];
    }
  vals[0] = value;
 //long
 long sum = 0;
 for(int i = 0; i < N; i++){
  sum = sum+ vals[i];
  }
  //sum vai ser a soma de todos os valores
  filtered = sum/N;

  //calculando a turbidez

  turbidty = - 0.0176921755*pow(filtered,3) +45.5858789633*pow(filtered,2)-
39153.2817095755*filtered+11209754.0500094;
  Serial.println(turbidty);
  delay(100);// A cada 100ms
}
```

3.1.1.3 SENSOR DE PH

Sensor que utiliza sondas para a medição do Ph dos líquidos. Nos experimentos tinhamos dois módulos diferentes, V1.1 e V2.0, conforme ilustrados na figura 18.

Figura 18: Módulo sensor de Ph V1.1 e V2.0 com sensor de temperatura integrado

Fonte: autor, 2017.

3.1.1.3.1 CALIBRAÇÃO

Para a calibração teve-se que aferir os valores da sonda de um aparelho ja pré-calibrado com a sonda de Arduino. Com as pesquisas feitas, soube-se que a sonda variava linearmente, e que a temperatura funcionava com um *offset*. Assim, foi realizado o código.

3.1.1.3.2 CÓDIGO ISOLADO

```
#define N 10// numero de amostras
int Phsensor;
int sensorPh = A12;

// arrawy de filtação
int Ph1[N];
// valor filtardo do Ph
float Phsensorfiltered;
float PH;
```

```
void setup() {
 // inicializando a conexão PC
Serial.begin(9600);
}

void loop() {
    PH = -0.0290856031128*(getPh() - 7) + 22.1012451361866; // alterar -6 para efeito da teperatura no caso -2*temperatura
    Serial.println("Ph 1 : ");
    Serial.println(PH);
    delay(1000);
}

float getPh()
{
  Phsensor = (analogRead(sensorPh));

  //FILTRO VERMELHO
   for(int i = N -1; i > 0; i--){
   Ph1[i] = Ph1 [i-1];
   }
   Ph1[0] = Phsensor;
   //long
   long sumPh = 0;
   for(int i = 0; i < N; i++){
   sumPh = sumPh + Ph1[i];
   }
  //sum vai ser a soma de todos os valores
   Phsensorfiltered = sumPh/N;
   return Phsensorfiltered;
}
```

3.1.1.4 SENSOR DE TEMPERATURA DS18B20

Versão impermeabilizada do sensor de temperatura DS18B20 Arduino (figura 19). Este sensor funciona com boa medição de temperaturas de até 125°C. Contudo, seu cabeamento é revestido de PVC e é de suma importância manter temperaturas inferiores a 100°C. O sinal emitido pelo sensor é digital, não havendo degradamento resultante da

distância de cabeamento, fornecendo leituras de temperatura de 9 a 12 bits (configuráveis) em uma interface de 1 fio, de modo que apenas um fio (e terra) precisa ser conectado em um microprocessador central.

Como cada DS18B20 contém um número de série de silício exclusivo, vários DS18B20s podem existir no mesmo barramento de 1 fio. Isso permite colocar sensores de temperatura em vários lugares diferentes. As aplicações nasquais esta característica é útil incluem controles ambientais HVAC, temperaturas de detecção dentro de edifícios, equipamentos ou máquinas e monitoramento e controle de processos (Dfrobot).

Figura 19: Sensor de temperatura DS18B20

Fonte: autor, 2017.

3.1.1.4.1 CALIBRAÇÃO

É realizada a verificação do sensor quanto a determinada temperatura de um líquido. Se se constatar uma diferença do mesmo quanto a um medidor já calibrado, é realizada no código uma adição de um “offset”, aumentando ou diminuindo o nível de temperatura retornado na medição, uma vez que a temperatura, analisando o *datashet* do sensor, não sofre influência significativa de demais fatores do meio ao qual o sensor está inserido, se comportando de forma linear sua medição.

3.1.1.4.2 CÓDIGO ISOLADO

```
// Programa : Sensor de temperatura DS18B20
// Autor : FILIPEFLOP

float tempC;

#include <OneWire.h> // biblioteca nescessária do sensor
#include <DallasTemperature.h> // biblioteca nescessária do sensor

// Porta do pino de sinal do DS18B20
#define ONE_WIRE_BUS 18

// Define uma instancia do oneWire para comunicacao com o sensor
OneWire oneWire(ONE_WIRE_BUS);
// Armazena temperaturas minima e maxima
DallasTemperature sensors(&oneWire);
DeviceAddress sensor1;

void setup(void)
{
  Serial.begin(9600);// incicia comunicação serial com computador (velocidade de troca de informações)
  sensors.begin();// inicia o sensor
  sensors.getAddress(sensor1, 0); // localiza o sensor
}

void loop()
{

getTEMPERATURA();
  {
  // Mostra dados no serial monitor
  Serial.print("Temp C: ");
  Serial.println(tempC);
  }
}
void getTEMPERATURA(){
    // Le a informacao do sensor
   sensors.requestTemperatures();
```

```
 tempC = sensors.getTempC(sensor1);
delay(3000);
}
```

3.1.1.5 SENSOR DE TEMPERATURA MTE THOMSON 4053

Sensor não utilizado no experimento devido à grande variação da temperatura e o mesmo demorar para se equilibrar com a temperatura. Ilustração: figura 20.

Figura 20: Sensor de temperatura Mte Thomson 4053

Fonte: autor, 2017.

3.1.1.5.1 CÓDIGO ISOLADO

```
// Programa : Sensor de temperatura 4053
// Autor : Marcelo Wegener Possamai
//sensor código
  #define N 10// número de amostras qqqa
int TEMPERATURA3;
int sensorTemp1 = A0;
// arrawy de filtação
int TEMP3[N];
// valor filtardo do Ph
float TEMP3filtered;

void setup(void)
{
  Serial.begin(9600);// incicia comunicação serial com computador (velocidade de troca de informações)
}
```

```
void loop()
{

//CÓDIGO DO SENSOR
   TEMPERATURA3 = (analogRead(sensorTemp1));

  //FILTRO VERMELHO
   for(int i = N -1; i > 0; i--){
   TEMP3[i] = TEMP3 [i-1];
   }
   TEMP3[0] = TEMPERATURA3;
   //long
   long sumTEMPERA = 0;
   for(int i = 0; i < N; i++){
   sumTEMPERA = sumTEMPERA + TEMP3[i];
   }
  //sum vai ser a soma de todos os valores
   TEMP3filtered =  0.0018*pow((sumTEMPERA/N),2) - 1.764*(sumTEMPERA/N) + 489.41;
//mostrando a temperatura
 Serial.println(TEMP3filtered);
  delay(3000);
}
```

3.1.1.6 MÓDULO SENSOR DE RECONHECIMENTO DE COR TCS230

O módulo de reconhecimento de cor é utilizado para reconhecimento das cores, operando com o reconhecimento RGB (*Red*, *Green*, *Blue*); logo, se pode reconhecer a intensidade das cores vermelha, verde e azul, ou até mesmo a combinação em si das cores que compõem o elemento analisado. Sendo assim, ele será utilizado para avaliar a cor aparente de entrada e saída da amostra com os testes. Módulo ilustrado na figura 21.

Figura 21: Sensor de reconhecimento de cor TCS230

Fonte: autor, 2017.

3.1.1.6.1 CÓDIGO ISOLADO

O código do programa foi encontrado já feito pelo site Arduino & Cia, mas, para funcionar com medição na escala de 0 a 255 no RGB, este foi aprimorado com funções map e função de filtro para não apresentar grandes variações de indicação.

```
//VCC——5V
//GND——GND
//S0——D3
//S1——D4
//S2——D5
//S3——D6
//OUT——D2

 //filtro
 #define N 10
 int vals1[N];
 int vals2[N];
```

```
 int vals3[N];

//código normal- portas
const int ss0 = 24;
const int ss1 = 25;
const int ss2 = 26;
const int ss3 = 23;
const int outt = 22;

//variáveis para realizar a filtração do sinal e mapeamento
int R, G, B;
int Ro,Go,Bo;

byte countRed = 0;
byte countGreen = 0;
byte countBlue = 0;

void setup() {
  //código normal
  Serial.begin(9600);
  pinMode(ss0, OUTPUT);
  pinMode(ss1, OUTPUT);
  pinMode(ss2, OUTPUT);
  pinMode(ss3, OUTPUT);
  pinMode(outt, INPUT);
  digitalWrite(ss0, HIGH);
  digitalWrite(ss1, HIGH);
}

void loop() {

//imprimindo valores
  Serial.println("Red: ");
  Serial.println((getColorRed1()), DEC);
  Serial.println("Green: ");
  Serial.println((getColorGreen1()), DEC);
  Serial.println("Blue: ");
  Serial.println((getColorBlue1()), DEC);
  delay(1000);
}
```

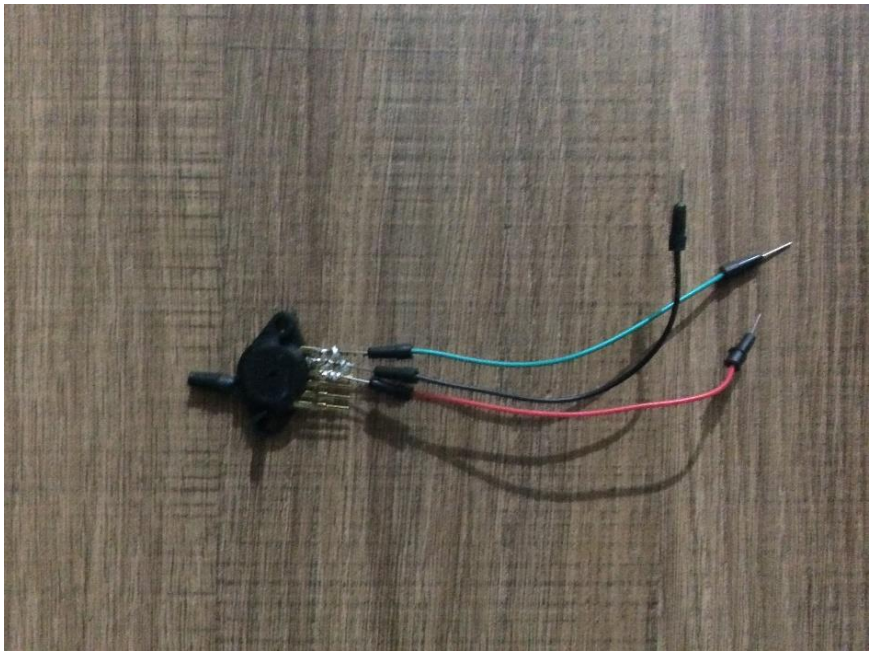
```
int getColorRed1()
{

  digitalWrite(ss2, LOW);
  digitalWrite(ss3, LOW);
  countRed = pulseIn(outt, digitalRead(outt) == HIGH ? LOW : HIGH);
//mapeando valores de 0 a 255
  R = map(countRed,6,24,255,0);
  digitalWrite(ss3, HIGH);
  digitalWrite(ss2, HIGH);


  //FILTRO VERMELHO
   for(int i = N -1; i > 0; i--){
   vals1[i] = vals1 [i-1];
   }
   vals1[0] = R;
   //long
   long sum1 = 0;
   for(int i = 0; i < N; i++){
   sum1 = sum1+ vals1[i];
   }
 //sum vai ser a soma de todos os valores
   Ro = sum1/N;
   return Ro;
}

int getColorGreen1()
{

  digitalWrite(ss2, LOW);
  digitalWrite(ss3, LOW);
  digitalWrite(ss3, HIGH);
  countBlue = pulseIn(outt, digitalRead(outt) == HIGH ? LOW : HIGH);
//mapeando valores de 0 a 255
  G = map(countGreen,6,26,255,0);
  digitalWrite(ss2, HIGH);

// FILTRO GREEN
```

```
    for(int i = N -1; i > 0; i--){
    vals2[i] = vals2 [i-1];
    }
    vals2[0] = G;
    //long
    long sum2 = 0;
    for(int i = 0; i < N; i++){
    sum2 = sum2+ vals2[i];
    }
  //sum vai ser a soma de todos os valores
    Go = sum2/N;
    return Go;
}

int getColorBlue1()
{

   digitalWrite(ss2, LOW);
   digitalWrite(ss3, LOW);
   digitalWrite(ss3, HIGH);
   digitalWrite(ss2, HIGH);
   countGreen = pulseIn(outt, digitalRead(outt) == HIGH ? LOW : HIGH);
//mapeando valores de 0 a 255
   B = map(countBlue,5,21,255,0);

// FILTRO BLUE
    for(int i = N -1; i > 0; i--){
    vals3[i] = vals3 [i-1];
    }
    vals3[0] = B;
    //long
    long sum3 = 0;
    for(int i = 0; i < N; i++){
    sum3 = sum3+ vals3[i];
    }
  //sum vai ser a soma de todos os valores
    Bo = sum3/N;
    return Bo;
}
```

3.1.1.7 SENSOR DE PRESSÃO MPX5700AP

O sensor em destaque é utilizado para medição de pressões absolutas, sendo a pressão absoluta encontrada no portal Smar Automação Industrial como sendo a pressão em relação ao vácuo perfeito, ou seja a pressão 0 absoluta. Assim, ao nível do mar, vai indicar 760 mmHg. O sensor, segundo Silva (2016), pertence à classe dos MPX5700, os quais resistem a 70m de coluna de água, 101,5 PSI e 7 bar, e foi utilizado por apresentar elevado grau de resistência a pressões elevadas, podendo medir a pressão interna absoluta do cilindro. Ilustração: figura 22.

Figura 22: Sensor de pressão absoluta MPX5700AP



Fonte: autor, 2017.

3.1.1.7.1 CALIBRAÇÃO

Para a calibração do sensor foi utilizada a técnica desenvolvida para aferição dos altimetros aeronáuticos, onde é feita a calibração da câmara bárica deste. No site de metereologia Redmet pode-se ter a pressão atual do município de Anápolis, através da estação alocada na base aérea. Verifica-se a pressão medida com a voltagem do sensor e anota-se. Após algumas horas, anota-se a pressão e a voltagem do sensor. Desta forma pode-se ter noção da pressão absoluta relacionada com a voltagem.

3.1.1.7.2 CÓDIGO ISOLADO

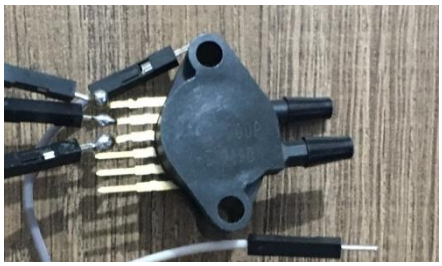
```
#define N 10// numero de amostras
 //constante de calibração
const float SensorOffset = 108;
float sensorValue;
// arrawy de filtação
int PRESSABS[N];
// valor filtardo do Ph
float PRESSABSfiltered;

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

void loop() {
     // imprimindo os valores
  Serial.print("Pressão absoluta: ");
  Serial.print(getPRESSABS());
  Serial.println(" kPa");
    delay(1000);
}

float getPRESSABS()
{
 // Fazendo a leitura do sensor
  sensorValue = (analogRead(A0)-40.96)/1,3166592; //Do maths for calibration // A9
  //FILTRO PH
   for(int i = N -1; i > 0; i--){
   PRESSABS[i] = PRESSABS [i-1];
   }
   PRESSABS[0] = sensorValue;
   //long
   long sumPRESSABS = 0;
   for(int i = 0; i < N; i++){
   sumPRESSABS = sumPRESSABS + PRESSABS[i];
   }
 //sum vai ser a soma de todos os valores
```

```
  PRESSABSfiltered = (sumPRESSABS/N) + SensorOffset;

 return(PRESSABSfiltered);
}
```

3.1.1.8 SENSOR DE PRESSÃO MPX5700DP

O sensor em destaque é utilizado para medição de pressão diferencial, sendo a pressão diferencial encontrada no portal Smar Automação Industrial como sendo a pressão relativa a outra pressão qualquer, ou seja, pode-se comparar a pressão interna do cilindro com a pressão externa, ou até mesmo pode-se medir o cilindro submetido a vácuo com a pressão externa do cilindro. Este sensor pertence à classe dos MPX5700 de resistência. Ilustração: figura 23.

Figura 23: Sensor de pressão diferencial MPX5700DP



Fonte: autor, 2017.

3.1.1.8.1 CALIBRAÇÃO

Para a calibração do sensor foi mensurado por *datasheet* o valor angular de indicação do sensor. Sabendo-se este, usa-se um *offset* para que a pressão se iguale a “0”. Aparecendo uma reta, o sensor estará calibrado.

3.1.1.8.2 CÓDIGO ISOLADO

```
#define N 10// numero de amostras

 // constante para calibração
const float SensorOffset1 = 6;
float sensorValue1;
```

```
      // arrawy de filtação
      int PRESSRELA[N];
      // valor filtardo do Ph
      float PRESSRELAfiltered;

      void setup() {

        Serial.begin(9600);
      }
      // the loop routine runs over and over again forever:
      void loop() {
        // imprimindo valores do sensor:
        Serial.print("Pressão relativa: ");
        Serial.print(getPRESSRELA());
        Serial.println(" kPa");
        delay(1000);
      }

      float getPRESSRELA()
      {
        // Leitura dos pinos e filtragem:
         sensorValue1 = (analogRead(A0)-40.96)/1,3166592; //Calculo para calibração // A8

         //FILTRO PH
          for(int i = N -1; i > 0; i--){
          PRESSRELA[i] = PRESSRELA [i-1];
          }
          PRESSRELA[0] = sensorValue1;
          //long
     long sumPRESSRELA = 0;
     for(int i = 0; i < N; i++){
     sumPRESSRELA = sumPRESSRELA + PRESSRELA[i];
     }
    //sum vai ser a soma de todos os valores
     PRESSRELAfiltered = (sumPRESSRELA/N) + SensorOffset1;
   return(PRESSRELAfiltered);
}
```

3.1.2 COMPONENTES DE ATUAÇÃO

Nos componentes de atuação, estará incluso todo componente que de forma análoga trabalhe com a atuação de alguma atividade que não se enquadre no quesito de sensoriamento. Estão nesta classe o ignitor, solenóides e demais componentes.

3.1.2.1 IGNITOR

Para acender as chamas do aquecedor, foi levada em conta a teoria do poder das pontas, com demonstrado por Gomes (2016), que relata que, ao se submeter um material condutor a um acúmulo de cargas, toda superfície do material tem que apresentar o mesmo potencial. Deste modo, se houver um afunilamento do material, este apresentará menor raio e acumulará maior carga para manter seu potencial equiparado com as demais áreas do material.

Baseado nesse princípio, quando houver acúmulo suficiente de cargas na ponta positiva, e na outra uma carga oposta, os elétrons vão saltar para manterem equilibrio, devido a cargas de sinais opostos se atraírem, fazendo do ar o meio condutor e gerando, assim, uma faísca, que servirá de fonte de calor inicial para o sistema de chamas.

3.1.2.2 VÁLVULA SOLENÓIDE ELÉTRICA

A válvula solenóide eletrica (figura 24) se comporta com um registro onde, com baixa tensão ou nula, a mesma encontra-se em posição fechada. Ao ser energizada, esta se abre totalmente, permitindo a passagem do fluido.

No protótipo foram adquiridas 4 válvulas solenóides; duas sendo utilizadas para controle do fluxo de amostra na entrada e na saída, uma no controle do ar comprimido ou vácuo e, por fim, uma no controle de gás no sistema de aquecimento.

As válvulas de uso no protótipo são da marca Saier, modelo SEN-SZ 43W, que operam com voltagem de 12V. O Arduino opera com tensão máxima de 5V. Então, com um módulo relé de 16 canais, podemos ter um circuito isolado. Também com o uso dos demais sensores pode-se ter o exato momento de trabalho das válvulas.

Figura 24: Válvula solenóide

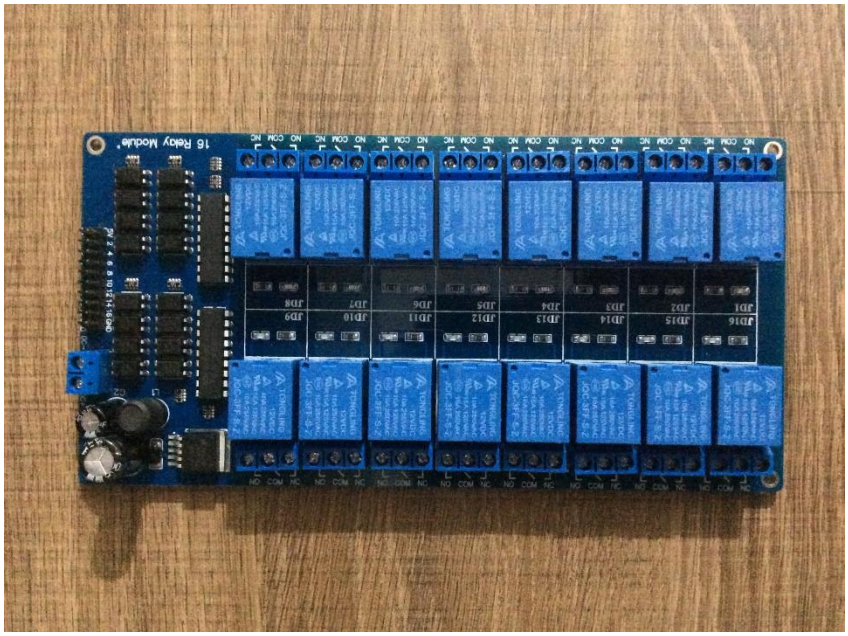
Fonte: autor, 2017.

3.1.2.3 MÓDULO RELÉ 16 CANAIS

O módulo relé é um módulo atuador acoplado no Arduino, onde é possível ter o controle de 16 chaves de energia, como ilustrado na figura 25, sempre obedecendo as limitações de cada módulo. No caso do módulo relé adotado no sistema elétrico do protótipo, foi utilizado um módulo que opera com 12 V de corrente contínua, com 15 A a 125 V de corrente alternada e 10 A a 250V de corrente alternada.

O módulo de 16 canais vai ter utilidade em várias partes do protótipo, que vão desde ignitores a outras funções como controle de válvulas. Para ser realizado o acionamento de cada chave é realizado todo um código no Arduino, sem nescessidade de bibliotecas específicas para o módulo, apenas com a definição de pinagem e o uso de sinal “verdadeiro” ou “TRUE” para envio de corrente para abertura da chave ou sinal “falso” para fechamento da “chave” pelas portas digitais. No caso, se for uma porta analógica ou pmw, há que se definir um valor de envio de 256.

Figura 25: Módulo relé 16 canais



Fonte: autor, 2017.

3.1.2.4 *DISPLAY* NEXTION

O *display* Nextion (figura 26) é um lcd, que foi feito para garantir uma melhor integração humano/máquina, tendo o *display* um circuito integrado, sem nescessidade de ocupar a memória do Arduino ou de outro microcontrolador com extensas linhas de código para identificação do toque ou *touchscreen* , ou até mesmo com comandos para desenho ou importação de imagens para exibição no lcd.

No site da Nextion podem-se encontrar *displays* de tamanhos bem variáveis, podendo atender diversas finalidades, com *displays* que variam 2.4” a 7.0”, tendo resoluções de 320 x 240 a 800 x 480 *pixels* e também diversas memórias internas, que vão de 4 MB a 16 Mb.

No site de seu desenvolvedor pode se ter uma plataforma própria para desenvolver toda parte gráfica do *display*, cujo codinome se determina como “Nextion Editor”. Este aplicativo, que funciona tanto em sistema operacional Windows ou Mac, pode-se ter excelentes ferramentas gráficas como barras de avanço, listas, inserção de imagens para determinação de botões e outros comandos, além de diversos outros que podem ser criados e editados.

Figura 26: Display Nextion 4.3”

Fonte: site Itead studio.

Disponível em: < https://www.itead.cc/nextion-nx4827t043.html > Acesso em set. 2017.

3.1.3 Diversos materiais

Na lista dos diversos materiais temos outros componentes que não atuam em si conectados ao Arduino, mas são de extrema necessidade para o funcionamento global do projeto, cada um com uma função específica.

3.1.3.1 CÂMARA DE TESTE

Para realização de testes pensou-se em uma câmara que fosse suficientemente resistente para pressões máximas de 5 Bar de ar comprimido e que, no máximo, fosse submetida sem ruptura ou esmagamento a pressões de 1 Bar a vácuo. Sendo assim, através de conhecimentos e auxílio do engenheiro mecânico prof. me. Sérgio Mateus Brandão, CREA-1016988052 DR-GO, chegou-se à escolha do cilindro gás ou balão de ar do caminhão Mercedes, com 10 Bar de pressão máxima e volume total interno de 40 Litros, modelo MB 1620. Ilustração: figura 27.

Figura 27: Câmara de teste (Balão de ar Mercedes MB1620)

Fonte: autor, 2017.

3.1.3.2 ENCANAÇÃO

Para o encanamento do protótipo foi de suma importância o uso de peças de PVC, aço galvanizado e também peças de gás.

3.1.3.3 COMPONENTES ELÉTRICOS

Nos componentes elétricos encontramos toda a parte elétrica do protótipo ainda não destacada, como fontes de alimentação para o sistema e outros compnentes que serviram para receber ou transmitir energia. São estes:

- 10 Metros de cabo de internet.
- 2 metros de cabos paralelos de energia.

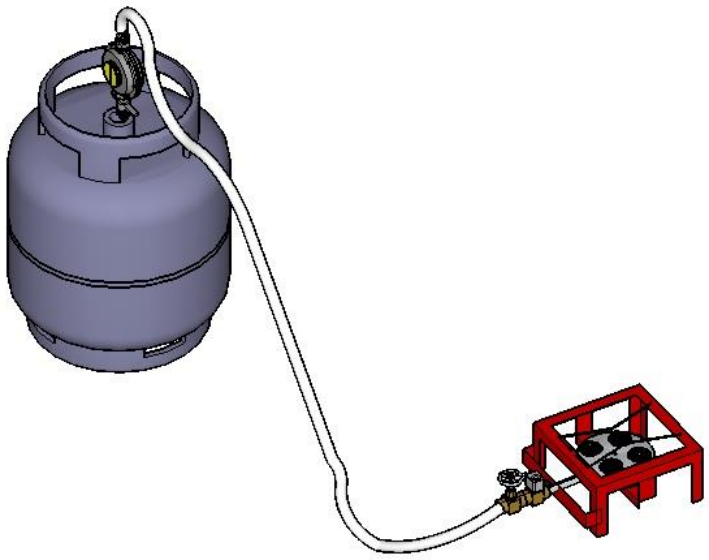
- 1-Unidade fonte de alimentação 12V 1.0 A (Flex industries / Modelo:WA-12M12FB-EAAA).
- 1- Unidade fonte de alimentação 9V 1.0 A (Leader eletronics inc/ Modelo: MU12-210901QN6S).
- 1-Unidade fonte de alimentação Cftv 12v 10A (Efm1210).

3.1.3.4 COMPONENTES REDE DE GÁS

Na composição da rede de gás (esquema na figura 28), foram adquiridos alguns itens essenciais para a montagem do protótipo, sendo estes:

- 2,02 m de mangueira de gás (Arqua / multi-uso 3/8” 9,52mm).
- 2- Unidades Abraçadeira rosca sem fim 9mm (Matrix/ Modelo S091422 mínimo 9/16” máximo 7/8”).
- 1-Unidade Botijão de gás ().
- 1-Unidade Fogão pasteleiro 1 boca (Omega e luxo queimador de ferro fundido esmaltado).
- 1-Unidade Regulador para gás (Aliança 2,8 Kpa 1kg/h GLP/ Entrada 5/8” Saída 3/8”).

Figura 28: Esquema de ligações rede de gás e componentes

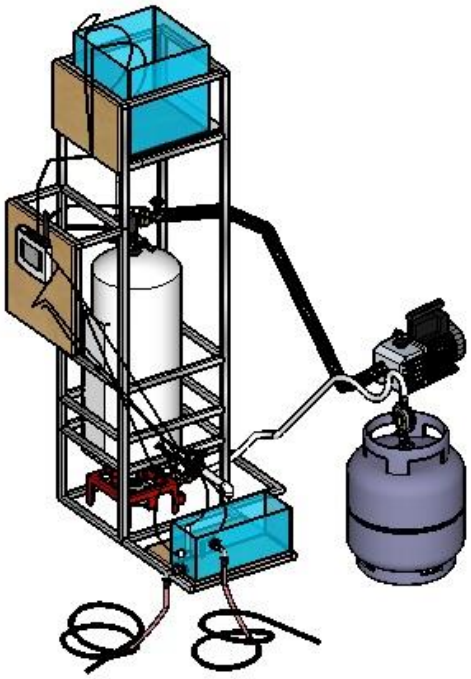


Fonte: autor, 2017.

3.2 EXECUÇÃO DA PARTE FISÍCA DO PROTÓTIPO

Para se ter previsão e evitar possíveis riscos futuros, o projeto foi realizado no software de modelagem *Sketchup* 2017 (protótipo idealizado na figura 29), onde foram previstos todos os componentes necessários para a execução do experimento, desde lugares específicos do cilindro para conexões até válvulas de segurança.

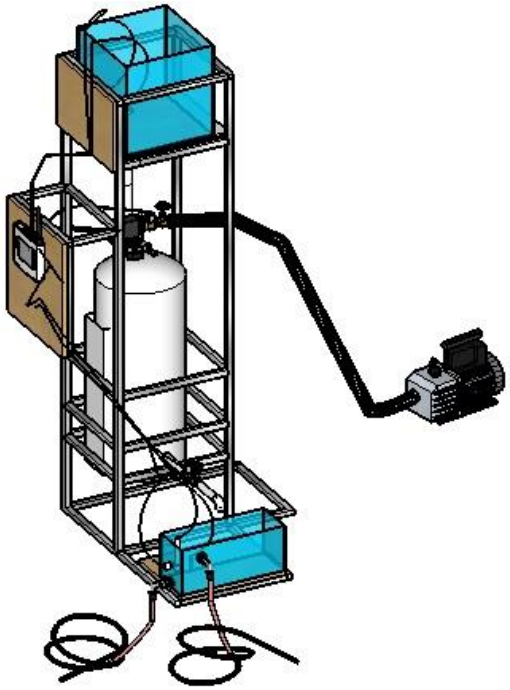
Figura 29: Protótipo *Sketchup*



Fonte: autor, 2017.

Durante a ida ao laboratório do Centro Universitário Unievangelica de Anápolis, foi realizado um questionário com o engenheiro mecânico me. Sérgio Mateus Brandão, CREA-1016988052 DR-GO, o qual afirmou que, com os materiais do protótipo inicial, seria arriscado trabalhar com uso de compressão, vácuo e temperatura. Para o uso simultâneo também de duas variáveis do experimento deveria ser fabricado um reservatório com chapas de 8 mm, o qual demandaria um orçamento muito alto. Assim optou-se apenas para verificação da influência do vácuo no processo atual de decantação, chegando a outro protótipo, comforme a figura 30.

Figura 30: Protótipo atualizado do *Sketchup*



Fonte: autor, 2017.

3.2.1 Execução do experimento

3.2.1.1 1º Etapa: construção da armação em metalon

Nos primeiros dias de confecção do protótipo foi primeiramente realizada a montagem da estrutura de sustentação, com o uso do material de metalon devido à sua facilidade de modelagem e também à sua resistência, que atendia às necessidades do protótipo. Ilustração: figura 31.

Figura 31: Estrutura de metalon

Fonte: autor, 2018.

3.2.1.2 2º Etapa: Modificação do tanque

Para o devido uso do tanque, foi necessário moldar algumas conexões para que se conseguisse ter 4 orifícios no tanque, que são o dreno, a saída de água, a entrada de água e a porta de ar. Ilustração; figura 32.

Figura 32: Tanque de teste modificado

Fonte: autor, 2018.

3.2.1.3 3º Etapa: Colocação da parte elétrica e hidráulica

Nessa etapa foram colocados os aquários devidamente perfurados. Junto a estes foi instalada toda a parte hidráulica e a rede de gás, a qual foi testada com o uso completo do volume do tanque com água, assim verificando vazamentos e feito correções. Também foram instalados os atuadores e sensores, e toda a fiação. Ilustração: figura 33.

Figura 33: Montagem da parte hidráulica e elétrica

Fonte: autor, 2018.

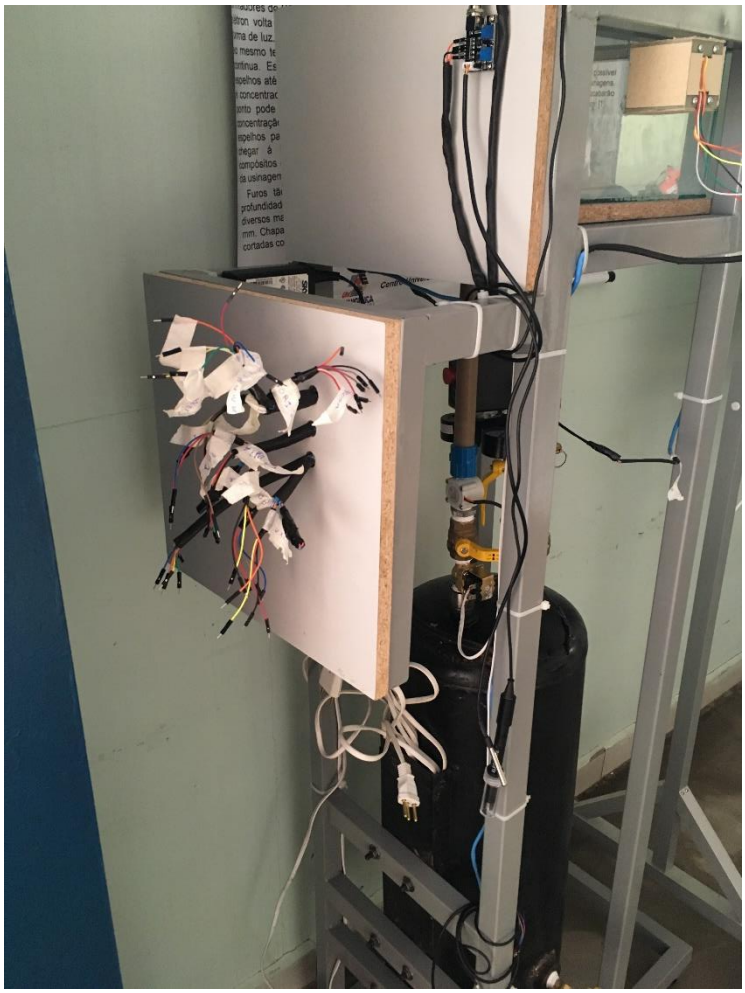
3.2.1.4 4º Etapa: Enumeração dos fios e regulagem dos sensores

Para o resultado ser obitido com exatidão, foram realizados vários testes com misturas padrões de Ph para calibração das sondas, além de calibrações do sensor de turbidez, que, posteriormente, foi descartado por apresentar grandes variações com a luminosidade ambiente. (Apêndice A).

Para usar corretamente a fiação dos sensores e atuadores, foi realizada enumeração dos fios, os tabelando e denominando suas respectivas funções, com o uso de fita crepe. (Apêndice B).

Ilustração da figura 34 demonstra todo o processo de denominação de cada fiação do sistema e para o que foi empregada.

Figura 34: Denominação dos fios e suas funções



Fonte: autor, 2018.

3.2.1.5 5º Etapa: Conexão do Arduino com o protótipo e confecção do código final

Com todos os sensores calibrados, foi feita a ligação do Arduino com os fios, conforme ilustrado na figura 35. Nessa etapa foi adicionado, além do previsto pelo projeto inicial, um módulo Sd e um Buzzer, para se determinar um fluxograma final (Apêndice C). Com este fluxograma foi realizado o código final (Apêndice E).

Figura 35: Conexão Arduino- protótipo

Fonte: autor, 2018.

3.2.1.5.1 Código Sd

```
#include <SD.h>
#include <SPI.h>
File myFile;
int pinCS = 53; // Pin 10 on Arduino Uno
void setup() {

  Serial.begin(9600);
  pinMode(pinCS, OUTPUT);

 // INICIALIZAÇÃO SD Card
    SD.begin();
```

```
  // CRIANDO ARQUIVO DE TEXTO
  myFile = SD.open("tcc.txt", FILE_WRITE);


    myFile.println("marcelo gato demais ");
    myFile.close(); // close the file
    delay(1000);
    Serial.println("Done.");
  }

  void loop() {

  }
```

3.2.1.5.2 Código Buzzer

```
//Programa : Som no Arduino - Sirene
//Autor : Arduino e Cia

#define tempo 10
int Pinofalante = 10;

void setup()
{
  pinMode(Pinofalante,OUTPUT); //Pino do buzzer
}

void loop()
{

   tone(Pinofalante, 1800, tempo);
   delay(1);

}
```

3.2.1.6 6º Etapa: Conexão da tela Nextion e verificações iniciais

Assim que todos os sensores e todas as partes foram devidamente conectados, foi realizada a confecção da tela Nextion, já pré-configurada para o uso no TCC. Esta só foi conectada nesta etapa por utilizar os pinos Tx0 e Rx0 do Arduino, os mesmos que são utilizados para passar o código do computador para o microcontrolador.

Nessa etapa também verificou-se que estava funcionando de acordo com o planejado (figura 36). Com sua eficácia verificada, posteriormente foi realizada a confecção de um *checklist* para o uso correto, evitando assim erros indesejáveis (Apêndice D).

Figura 36: Verificações iniciais da máquina

Fonte: autor, 2018.

3.3 RESULTADO DE EXPERIMENTOS REALIZADOS

3.3.1 Teste realizado com água no misturador de sulfato de alumínio

Com o protótipo pronto foi iniciada a fase de verificação do protótipo, normatizado com 14 litros de entrada e 82 kpa de vácuo. Assim, os primeiros testes foram realizados com a água que entra incialmente no misturador, a qual estava misturada com sulfato de alumínio (coagulante) e cal (corretor de Ph). Nesses primeiros testes foi verificado a olho nu que a água se encontrava bem escura, com presença de alguns flóculos já formados, e, ao testar sua turbidez, foi verificado que, nas primeiras três amostras, encontravam-se de 25,9 a 26,27 NTU, e na última amostra, 22,06 NTU. O Ph variáva de 6,07 a 6,48. A cor variáva no sensor conforme a hora do experimento e, por isso, era medida sempre antes dos testes para ver se se mantinha a mesma. Para informações do experimento, ver a tabela 1.

Tabela 1: Teste com sulfato de alumínio

	Turbidez	PH	R-Entra	G-Entra	B-Entra	R-Saída	G-Saída	B- Saída	Tempo
Natural	25,9	6,07	0	14	11	11	22	16	Agitada
Natural	22,36	6							5min
1º teste	20,66	5,82	0	14	11	0	3	5	5min
2º teste	21,56	6,23	9	16	12	13	18	16	5min
Média	21,11	6,025	4,5	15	11,5	6,5	10,5	10,5	5min
Diferença	-1,25	0,025	4,5	1	0,5	-4,5	-11,5	-5,5	5min
Natural	14%	1%							
Estação	18%	1%				41%	52%	34%	

	Turbidez	PH	R-Entra	G-Entra	B-Entra	R-Saída	G-Saída	B- Saída	Tempo
Natural	26,27	6,48	10	14	12	13	19	25	Agitada
Natural	24,77	6							10min
1º teste	22,07	6,43	10	14	12	7	13	20	10min
2º teste	23,77	6,66	10	14	12	15	14	24	10min
Média	22,92	6,545	10	14	12	11	13,5	22	10min
Diferença	-1,85	0,545	0	0	0	-2	-5,5	-3	10min
Natural	6%	7%							
Estação	13%	-1%				15%	29%	12%	

	Turbidez	PH	R-Entra	G-Entra	B-Entra	R-Saída	G-Saída	B- Saída	Tempo
Natural	26,08	6,36	10	14	12	13	19	25	Agitada
Natural	22,08	6,92							15min
1º teste	24,88	6,9	6	11	15	14	14	18	15min
2º teste	24,02	6,2	6	18	9	16	15	24	15min
Média	24,45	6,55	6	14,5	12	15	14,5	21	15min
Diferença	2,37	-0,37	-4	0,5	0	2	-4,5	-4	15min
Natural	15%	-9%							
Estação	6%	-3%				-15%	24%	16%	

	Turbidez	PH	R-Entra	G-Entra	B-Entra	R-Saída	G-Saída	B- Saída	Tempo
Natural	22,06	6,3	9	15	11	15	31	21	Agitada
Natural	19,95	6,6							20min
1º teste	25	6,4	10	15	11	12	18	31	20min
Média	25	6,4	10	15	11	12	18	31	20min
Diferença	5,05	-0,2	1	0	0	-3	-13	10	20min
Natural	10%	-5%							
Estação	-13%	-2%				20%	42%	-48%	

Fonte: autor, 2018.

3.3.2 Teste realizado com água bruta com dosagem no protótipo com policloreto de alumínio

Na segunda fase do experimento foi realizado o teste com o coagulante policloreto de alumínio, que geralmente é utilizado em piscinas e contém o mesmo principio ativo de quebra coloidal que se faz com o cation Alumínio (Al^{++}) para remoção de cor. Na busca das amostras foi constatado, em relação à primeira amostra coletada, que a água, no dia da coleta, se encontrava com turbidez bem inferior à primeira amostragem, sendo a turbidez de 9,70 a 9,84 NTU. O Ph variava de 6,01 a 6,21. Resultados do experimento são vistos na tabela 2.

Tabela 2: Teste com policloreto de alumínio

	Turbidez	PH	R-Entra	G-Entra	B-Entra	R-Saída	G-Saída	B- Saída	Tempo
Natural	9,79	6,12	4	3	11	10	14	26	Agitada
Natural	7,16	6,4							5min
1º teste	10,5	6,91	4	3	10	8	12	23	5min
2º teste									5min
Média	10,5	6,91	4	3	10	8	12	23	5min
Diferença	3,34	0,51	0	0	-1	-2	-2	-3	5min
Natural	27%	-5%							
Estação	-7%	-13%				20%	14%	12%	
	Turbidez	PH	R-Entra	G-Entra	B-Entra	R-Saída	G-Saída	B- Saída	Tempo
Natural	9,7	6,01	4	3	11	10	14	26	Agitada
Natural	4,63	6,53							10min
1º teste	10,76	6,61	4	5	4	7	11	20	10min
2º teste									10min
Média	10,76	6,61	4	5	4	7	11	20	10min
Diferença	6,13	0,08	0	2	-7	-3	-3	-6	10min
Natural	52%	-9%							
Estação	-11%	-10%				30%	21%	23%	
	Turbidez	PH	R-Entra	G-Entra	B-Entra	R-Saída	G-Saída	B- Saída	Tempo
Natural	9,84	6,21	4	3	11	10	14	26	Agitada
Natural	3,51	6,47							15min
1º teste	10,04	6,5	4	8	5	7	15	17	15min
2º teste									15min
Média	10,04	6,5	4	8	5	7	15	17	15min
Diferença	6,53	0,03	0	5	-6	-3	1	-9	15min
Natural	64%	-4%							
Estação	-2%	-5%				30%	-7%	35%	

Fonte: autor, 2018.

4 CONCLUSÃO

Para o manufaturamento de qualquer projeto, devem-se ter todas as etapas de planejamento muito bem definidas, sejam elas o escopo do projeto, o planejamento, a execução e a finalização. Com todas essas atividades sendo seguidas nesse roteiro, dificilmente ocorrem grandes alterações. No protótipo aqui descrito foi relizada toda uma idealização com o uso principal do *Sketchup* para modelagem. Após isso, seguiu-se para a fase de planejamento, com a escolha de todos os sensores e materiais constituintes, e, por fim, seguiram-se as últimas duas fases finais, onde se montou todo o protótipo e foi feita a finalização, na qual se averiguaram os resultados obtidos e o efeito oriundo do processo.

Conclusivamente, foi entendido o quanto é necessário conhecer todos os sensores à risca e os agentes externos que influenciam os resultados. Quando foi realizada a fase de calibração notou-se que sensores de cor e turbidez se alteravam com a luminosidade ambiente. Desta forma, optou-se por pintar o aquário de preto, que, na escala RGB, é a ausência de cor, assim eliminando boa parte dos inconvenientes dos sensores. Contudo, percebeu-se que alguns sensores, como o de turbidez, seriam incapazes de fornecer resultados precisos, uma vez que tinham escalas de medições altas e sofriam alterações com a luminosidade.

Com os testes de implantação do vácuo no sistema foi identificado que, embora sensores MPX5700 suportem altas pressões, estes não têm grandes escalas para o vácuo, uma vez que no máximo, leem pressões negativas de 24 kPa. Como a bomba de vácuo causou, no processo, pressões de 82 kPa, optou-se por manter os sensores do Arduino apenas para monitoramento e não para serem automatizados em tais etapas que estes envolviam. As pressões maiores, então, foram lidas pelos manômetros.

Na modelagem do código final também se chegou à conclusão dos limites da plataforma Arduino, com a qual não seria possivel implementar códigos para funcionarem paralelamente. Então, se o processo exigisse além do planejado seria necessário algo mais robusto para o processamento dessas informações, como sugestão um “CLP” ou um “Raspiberry Pi”.

Na montagem do sistema, também verificou-se a importância de se ter um dimensionamento correto de cabos e fontes, uma vez que inicialmente foram colocados cabos de diâmetros menores ao motor utilizado no misturador e isso gerou aquecimento, optando-se pela trocas pelos de diâmetros maiores. Também ao realizar a parte de teste dos atuadores

verificou-se que a fonte utilizada em primeiro plano para o sistema de misturador foi de pequena corrente para alcançar a rotação desejada; assim, foi colocada uma fonte chaveada capaz de atingir voltagem de 12 V e corrente de 10 A, contornando o problema da falta de corrente.

Quando se iniciava a fase de verificação do funcionamento do protótipo também foi notado que o aquário superior trincou o fundo por causa do furo que tinha sido feito para passagem de água, e isso reduziu sua resistência. Para sanar tal deficiência, foi despejada cola de silicone industrial neutral para que esta, ao mesmo tempo, vedasse o líquido das amostras e não influenciasse na sua qualidade química.

Com os testes realizados, foi constatado que, se a amostra contém flóculos maiores, o vácuo a um curto espaço de tempo - cerca de 10 minutos - ajuda de certa forma a diminuir a turbidez, mas após a passagem desse tempo acontece o efeito reverso, desprendendo os agregados aglutinados e tornando a água mais turva com o passar do tempo. A coloração da água até os 10 minutos se torna mais clara; porém, com o tempo aumenta um pouco e se estabiliza.

Nas amostras com flóculos menores e turbidez menor, foi verificado que o vácuo aumenta um pouco a turbidez e paralisa o efeito do coagulante. Desta forma, mantém-se a turbidez ao longo do tempo, pouco se altera da forma natural da água agitada, podendo ser compreendida como variação de medição dos sensores.

Considerando sobre a execução do experimento, nota-se que o sensor de turbidez tem de ter uma escala pequena de medição para uma precisão mais confiável. Para o uso do vácuo seria melhor um sensor de vácuo com uma escala menor, uma vez que o MPX5700 lê apenas 24 kpa de vácuo e a bomba chega a 82 kpa. Para medir a cor é necessaria uma pequena variação da luz ambiente. Para sanar deficiências conforme a luminosidade seria interessante uma pesquisa do efeito que esta causa na medição dos sensores; assim, com um fotoresistor poderíamos compensar esse efeito no código.

Sabendo-se também que o vácuo age de forma retardante ao efeito de decantação, podemos concluir que talvez o inverso do vácuo, ou seja, a compressão, pode afetar de forma posistiva a decantação. Então, como sugestão de pesquisa, seria interessante fazer o experimento em um tanque submetido a compressão.

Ao final dos testes e da construção do protótipo constatou-se que o projeto tem futuro no que tange ao seu aperfeiçoamento. O esforço inicial de se criar um equipamento automatizado e que gerasse resposta na constituição de flocos maiores foi alcançado em parte

dos testes, não negligenciando a idéia inicial de se gerar um campo de pressão controlada para melhoria da decantação em estações de tratamento de água ou esgoto. A intenção é não parar a pesquisa em função do fim do trabalho de conclusão de curso, mas sim incluir na pesquisa a obtenção de pressão por compressão no cilindro, trabalhando com outros polímeros existentes no mercado e gerando possibilidade de se obter tratamento de água com menos custo e menos resíduos químicos para o consumidor final.

REFERÊNCIAS BIBLIOGRÁFICAS

Arduino e Cia. **Portal Arduino e Cia**. Código e informação do sensor de reconhecimento de cor. 2017. Disponível em: <http://www.arduinoecia.com.br/2014/02/sensor-de-reconhecimento-de-cor-tcs230.html > Acesso em: 07/11/2017.

Automação e Robótica, **Portal Automação e Robótica**. Definição de sensores e atuadores. 2017. Disponível em: <http://automacaoerobotica.blogspot.com.br/2012/07/sensores-e-atuadores-aplicados-robotica.html>. Acesso em: 07/11/2017.

Baú da eletrônica, **Portal Baú da eletrônica**. Informações extras DS18B20. 2017. Disponível em: <http://www.baudaeletronica.com.br/sensor-de-temperatura-a-prova-de-agua-ds18b20-2m.html?gclid=CjwKCAjw0qLOBRBUEiwAMG5xME9wzueOO-9FyIIlm0tC6GKHCl7MocEF0MfvsRiLwANTKCBsRLXnLBoC4QoQAvD_BwE >. Acesso em: 07/11/2017.

Biriba Acessórios, **Portal Biriba Acessórios**. Informações do Balão de ar comprimido. 2017. Disponível em: <http://www.biribaacessorios.com.br/ViewItem.php?ItemID=476>. Acesso em: 07/11/2017.

BOXELETRONICA, **Blog boxeletronica**. Imagem de Sonda de pH. 2017. Disponível em: <http://blog.boxelectronica.com/2017/04/18/medidor-ph-analogico/-imagem>. Acesso em: 22/10/2017.

COHESP, **Portal da Cohesp**. Efeitos do ph na saúde humana 2017. Disponível em: <http://cohesp.com.br/qual-ph-ideal-da-agua-para-consumo-humano/>.Acesso em: 02/11/2017.

Dfrobot, **Portal Dfrobot Robótica**. Informações sensor DS18B20. 2017. Disponível em: <https://www.dfrobot.com/wiki/index.php/Waterproof_DS18B20_Digital_Temperature_Sensor_(SKU:DFR0198)>. Acesso em: 07/11/2017.

EMBARCADOS, **Site Embarcados**. Especificações Arduino Mega. 2017. Disponível em: < https://www.embarcados.com.br/arduino-mega-2560/ >. Acesso em: 18/11/2017.

FABER, Marcos. A Importância dos Rios para as Primeiras Civilizações. História Ilustrada, Vol. 2. História Livre. com.1ª edição, 2011.

FERREIRA. P.C.; PIAI, K.A; TAKAYANAGUI, A.M.M.; SEGURA-MUÑOZ. S.I. Alumínio como fator de risco para a doença de Alzheimer. 2008.

FRANCO, Elton Santos. Avaliação da Influência dos Coagulantes Sulfato de Alumínio e Cloreto Férrico na Remoção de Turbidez e Cor da Água Bruta e sua Relação com Sólidos na Geração de Lodo em Estações de Tratamento de Água. p. 207, 2009.

GOMES, Rogério Viana. Atividade experimental no ensino de Física: A montagem da máquina Wimshurst como proposta de recurso didático no ensino de eletrostática. Brasília, 2016.

Itead Studio, **Portal Itead Studio**. Imagem Display Nextion 4.3". 2017. Disponível em: <https://www.itead.cc/nextion-nx4827t043.html>. Acesso em: 22/11/2017.

KOWATA, Emília Akemi; RIBEIRO, José Tarcísio; TELLES, Dirceu D Alkimin. Coagulação de água de abastecimento no mecanismo de adsorção-neutralização de cargas. 2000.

METCALF, L.; EDDY, H. Wastewater engineering: treatment and reuse. 5. ed. Revisado por George Tchobanoglous, Franklin L. Burton, H. David Stensel. McGrawHill, New York, 2003, 1819 p.

MICRO AMBIENTAL, **Portal Micro ambiental análise de águas**. Imagem da cor da água pura e outra não pura. 2017. Disponível em: <http://microambiental.com.br/servicos/monitoramento/controle-de-cor-agua/>. Acesso em: 22/11/2017.

OLIVEIRA, Cláudio Luis Vieira; ZANETTI; Humberto Augusto Piovesana. Arduino descomplicado. 1ª edição, São Paulo, 2015.

PAVANELLI, Gerson; DISSERTAÇÃO. Eficiência de Diferentes Tipos de Coagulantes na Coagulação, Floculação e Sedimentação de Água com Cor ou Turbidez Elevada. P. 233, 2001.

SANEAGO, **Portal da Saneago**. História da empresa em Anápolis e Goiás. 2017. Disponível em: <http://www.saneago.com.br/2016/#institicional>. Acesso em: 12/10/2017.

Sergio Luiz Stevan Jr / Rodrigo Adamshuk Silva. Automação e Instrumentação Industrial Com Arduino - Teoria e Projetos, Editora Érica.

WILLIAN FREIRE, **Advogados e associados**. Padrões de potabilidade e qualidade da água. 2017. Disponível em: <http://williamfreire.com.br/publicacoes/artigos/saneamento-ambiental/>. Acesso em: 12/10/2017.

APÊNDICE A

Regulagem Sensor de cor:

Sem Cor	Sensor Cor	Cor Branca
23	Red	6
26	Green	6
21	Blue	5

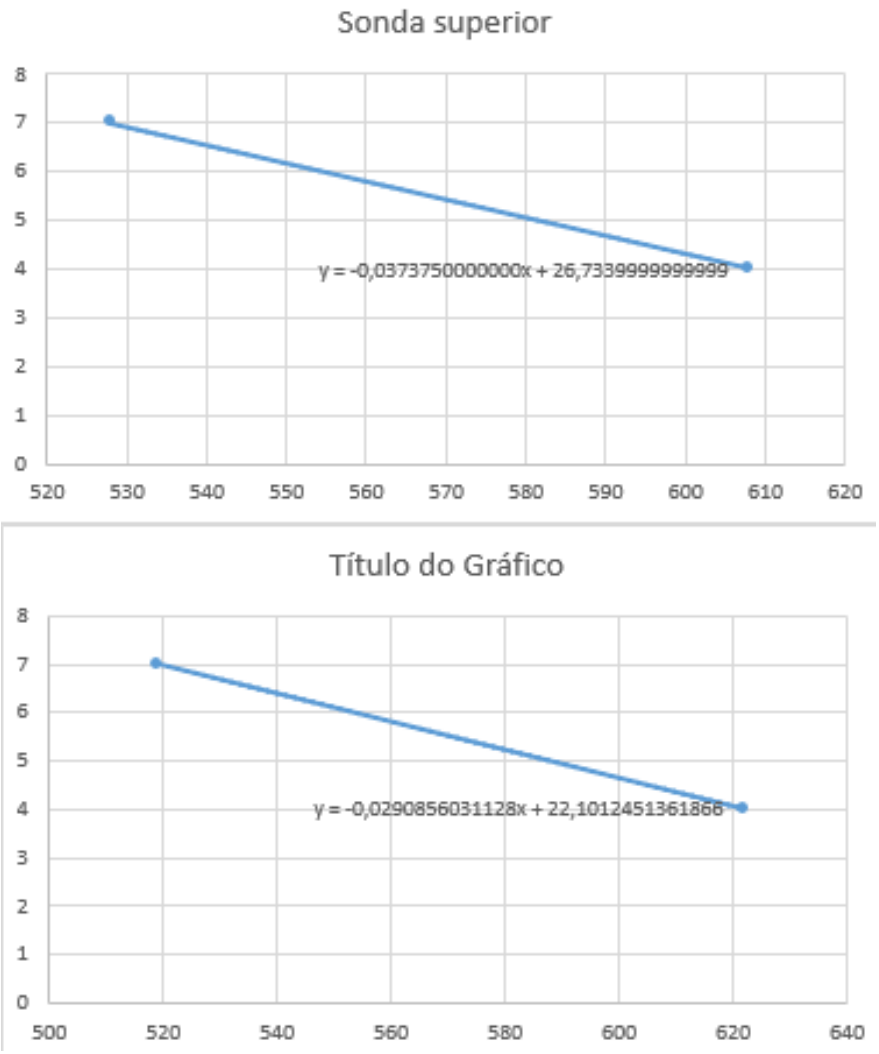
Fonte: autor, 2018.

Regulagem Sensor PH

PH Sonda Superior		
teste1		
Ph	sensor	temperatura
7	520	19
4,01	627	17
teste 2		
Ph	sensor	temperatura
7	548	
4,01	625	22
função:	y = -0,0384318766067x + 28,3681233933157	

PH Sonda inferior		
teste1		
Ph	sensor	temperatura
7	526	17
4,01	630	20
teste 2		
Ph	sensor	temperatura
7		
4,01		
função:	y = -0,0290856031128x + 22,1012451361866	

Fonte autor, 2018.



Fonte: autor, 2018.

Regulagem sensor turbidez

FTU	sensor	VOLTAGEM	EQUAÇÃO 3	ERRO	**equação mod1**	ERRO
0,46	870	4,24804688	0,106723987	-0,353276013	0,361413956	-0,098586044
6,38	869	4,24316406	7,00893835	0,62893835	6,714705467	0,334705467
12,96	868	4,23828125	12,83590759	-0,124092414	12,62869358	-0,331306419
22,02	866	4,22851563	21,68872288	-0,331277121	22,13506985	0,115069847
29,41	863	4,21386719	29,55743366	0,147433665	29,38898754	-0,021012459
35,35	858	4,18945313	35,32078944	-0,029210559	35,35127831	0,001278305
				Erro		ERRO
				maior		maior
				0,62893835		0,334705467
				menor		menor
				-0,353276013		-0,331306419

Fonte: autor, 2018.

Regulagem sensor de temperatura Thomson 4053

temperatura	resistência
50	500
60	578
70	602
100	667
	$y = 0,0018x^2 - 1,764x + 489,41$

Fonte: autor, 2018.

APÊNDICE B

Fiação dos sensores

SENSORES DA BASE				
SENSOR	senor Cor	COR FIO	FUNÇÃO	PINAGEM ARDUINO
COLORÍMETRO 1 BASE	VERMELHO	AMARELO	VCC	VCC
	PRETO	LARANJA	GND	GND
	AMARELO	AZUL	OUT	22
	MARROM	PRETO	S3	23
	AZUL	BRANCO	S0	24
	CINZA	VERDE	S1	25
	ROSA	ROSA	S2	26
SENSOR DE TEMPERATURA 1	ROSA	VERMELHO	VCC	VCC
	CINZA	PRETO	GND	GND
	AZUL	VERDE	SINAL	18
SENSOR DE PH 1	ROSA	VERMELHO	VCC	VCC
	CINZA	MARROM	GND	GND
	AMARELO	AZUL	SINAL	A12
SENSORES INTERMEDIÁRIOS				
SENSOR	senor Cor	COR FIO	FUNÇÃO	PINAGEM ARDUINO
SENSOR DE FLUXO	VERMELHO	ROSA	VCC	VCC
	PRETO	PRETO	GND	GND
	BRANCO	AMARELO	SINAL	2
SENSOR DE PRESSÃO RELATIVA	VERMELHO	LARANJA	VCC	VCC
	PRETO	PRETO	GND	GND
	VERDE	AMARELO	SINAL	A8
SENSOR DE PRESSÃO ABSOLUTA	VERMELHO	VERMELHO	VCC	VCC
	PRETO	CINZA	GND	GND
	BRANCO	AZUL	SINAL	A9

SENSORES DO TOPO				
SENSOR	senor Cor	COR FIO	FUNÇÃO	PINAGEM ARDUINO
COLORÍMETRO 2 TOPO	VERMELHO	VERMELHO	VCC	VCC
	MARROM	PRETO	GND	GND
	CINZA	CINZA	OUT	30
	AZUL	MARROM	S3	31
	AZUL	LARANJA	S0	32
	AMARELO	ROSA	S1	33
	ROSA	BRANCO	S2	34
SENSOR DE PH E TEMPERATURA 2	VERMELHO	VERMELHO	VCC	VCC
	PRETO	PRETO	GND	GND
	LARANJA	VERDE	SINAL PH	A13
	LARANJA	AMARELO	S TEMPERATURA	19

ATUADORES				
SENSOR	senor Cor	COR FIO	FUNÇÃO	PINAGEM ARDUINO
CONTROLE DE PORTAS RELÉS	ROSA	AMARELO	VCC	VCC
	CINZA	LARANJA	GND	GND
	ROSA	LARANJA	RELÉ(-) MISTURAD	14
	VERDE	AZUL	RELÉ(-) 2 ENTRA ÁG	15
	AZUL	AMARELO	RELÉ(+) 1 SAÍDA ÁG	16
	VERDE	CINZA	RELÉ(+) VÁCUO	17
BUZZER	BRANCO	BRANCO	SINAL	13
	CINZA	CINZA	GND	GND
LEITOR SD	MARROM	MARROM	VCC	VCC
	BRANCO	BRANCO	GND	GND
	LARANJA	LARANJA	MOSO	50
	LARANJA	LARANJA	MOSI	51
	MARROM	MARROM	SCK	52
	VERDE	VERDE	CS	53
TELA NEXTION	VERMELHO	VERMELHO	VCC	VCC
	PRETO	PRETO	GND	GND
	AZUL	AZUL	RX	TX1
	AMARELO	AMARELO	TX	RX1

		TX0	RX0
		PINO 1 ou bluethooth 3º pino apartir +	PINO 0 ou bluethooth 4º pino apartir +
		TX1	RX1
APC220		4º Pino apartir -	5º pino apartir -

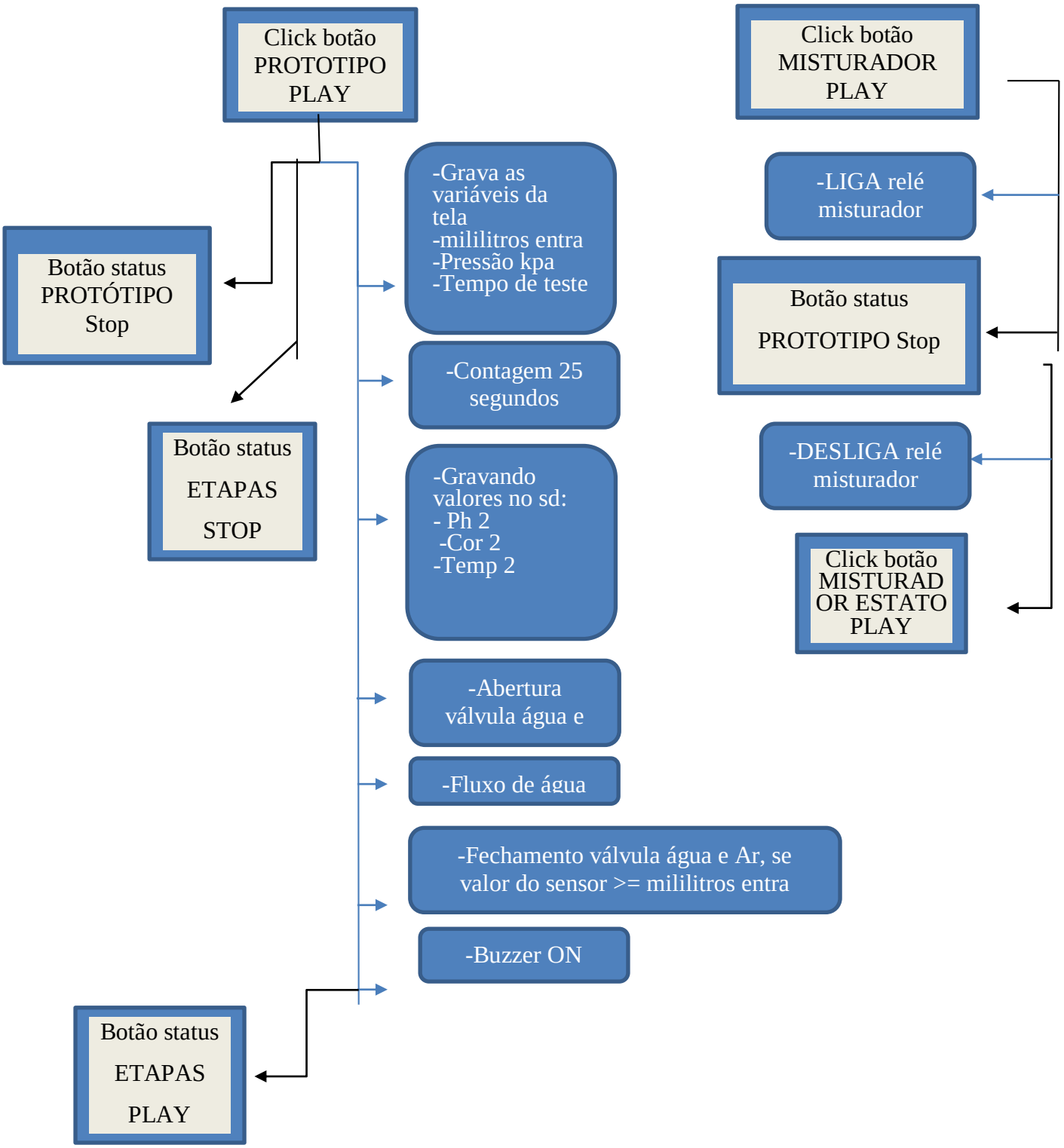
PINAGEM ARDUINO				
DIGITAL	PMW	ENTRADA ANALOG	TX	RX
14	2	A0	TX0	RX0
15	3	A1	TX1	RX1
16	4	A2		
17	5	A3		
18	6	A4		
19	7	A5		
20	8	A6		
21	9	A7		
22	10	A8		
23	11	A9		
24	12	A10		
25	13	A11		
26		A12		
27		A13		
28		A14		
29		A15		
30				

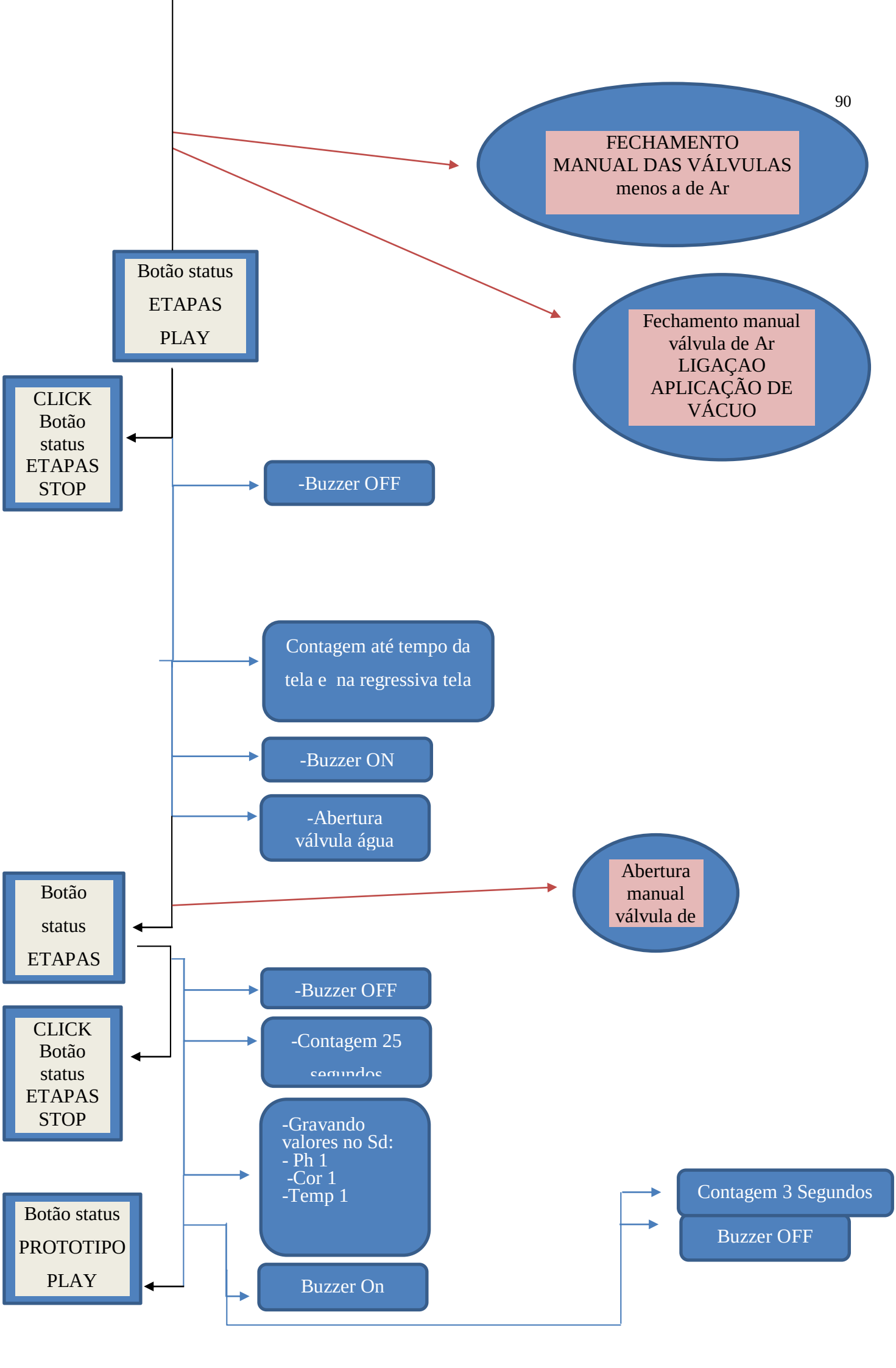
31				
32				
33				
34				
35				
36				
37				
38				
39				
40				
41				
42				
43				
44				
45				
46				
47				
48				
49				
50				
51				
52				
53				

Fonte: autor, 2018.

APÊNDICE C

Fluxograma




FECHAMENTO MANUAL DAS VÁLVULAS menos a de Ar
Botão status ETAPAS PLAY
Fechamento manual válvula de Ar LIGAÇAO APLICAÇÃO DE VÁCUO
CLICK Botão status ETAPAS STOP
-Buzzer OFF
Contagem até tempo da tela e na regressiva tela
-Buzzer ON
-Abertura válvula água
Botão status ETAPAS
Abertura manual válvula de
-Buzzer OFF
-Contagem 25 segundos
CLICK Botão status ETAPAS STOP
-Gravando valores no Sd:
- Ph 1
-Cor 1
-Temp 1
Botão status PROTOTIPO PLAY
Contagem 3 Segundos
Buzzer OFF
Buzzer On

APÊNDICE D

Manual de uso

CHECK LIST - TCC MARCELO
☐ 1º - Verificar se o cartão SD esta no lugar.
☐ 2º - ligar todos os plugs na tomada, inclusive o principa da máquina.
☐ 3º Após a ligar a máquina esperar uns 25 segundos para estabilizar os sensores.
☐ 4º - Desconectar a Bomba de vácuo.
☐ 5º - Ajustar tempo de teste e quantidade de água na entrada.
☐ 6º - fechar válvulas de saída de água e dreno, abrir válvulas de entrada de água e Ar
☐ 7º - Ligar misturador durante 10 segundos ou mais.
☐ 8º - Play em protótipo, esperar Buzzer.
☐ 9º - Enquanto Buzzer estiver tocando Fechar válvula de entrada de água, conectar Bomba de vácuo e ligar ela até a pressão desejada.
☐ 10º - Fechar todas as vávulas .
☐ 11º - Play em etapas.
☐ 12º - Quando o Buzzer tocar, abrir válvuas de saída água e AR, até a quantidade suficiente para realizar os testes.Lembrando de desconectar a Bomba de vácuo.
☐ 13º Play em etapas
☐ 14º Buzzer final, após a bipe pode retirar o SD para verificação.

Fonte: autor, 2018.

APÊNDICE E

Código final

```
#define N 10// numero de amostras

 //========== Sensores de temperatura ============
#include <OneWire.h> // biblioteca nescessária do sensor
#include <DallasTemperature.h> // biblioteca nescessária do sensor

float tempC;// TEMEPRATURA 1
float tempC2;// TEMEPRATURA 2


// Porta do pino de sinal do DS18B20
#define ONE_WIRE_BUS 18// porta de ligação TEMEPERAUTA 1
#define ONE_WIRE_BUSS 19 // porta de ligação TEMEPERAUTA 2


// Define uma instancia do oneWire para comunicacao com o sensor
OneWire oneWire(ONE_WIRE_BUS); // TEMPERATURA 1
OneWire oneWiree(ONE_WIRE_BUSS); // TEMPERATURA 2

// Armazena temperaturas minima e maxima- TEMPERATURA 1
DallasTemperature sensors(&oneWire);
DeviceAddress sensor1;

// Armazena temperaturas minima e maxima- TEMPERATURA 2
DallasTemperature sensorss(&oneWiree);
DeviceAddress sensor2;

 //========== Sensor de Fluxo ============
 byte sensorInterrupt = 0;  // 0 = digital pin 2
byte sensorPin       = 2;

// The hall-effect flow sensor outputs approximately 5.64 pulses per second per
// litre/minute of flow.
float calibrationFactor = 5.64;

volatile byte pulseCount;

float flowRate;
unsigned int flowMilliLitres;
unsigned long totalMilliLitres;

unsigned long oldTime;

unsigned int frac;

 //========== Sensores de Pressão Relativa ============
```

```
 // constante para calibração
const float SensorOffset1 = -36;
float sensorValue1;

// arrawy de filtação
int PRESSRELA[N];
// valor filtardo do Ph
float PRESSRELAfiltered;

 //=========== Sensores de Pressão Absoluta =============
 //constante de calibração
const float SensorOffset = -10;
float sensorValue;

// arrawy de filtação
int PRESSABS[N];
// valor filtardo do Ph
float PRESSABSfiltered;

 //=========== Sensores de PH 1 =============

int Phsensor;
int sensorPh = A12;

// arrawy de filtação
int Ph1[N];
// valor filtardo do Ph
float Phsensorfiltered;
float PH;

 //=========== Sensores de PH 2 =============
int Phsensorr;
int sensorPhh = A13;

// arrawy de filtação
int Phh1[N];
// valor filtardo do Ph
float Phsensorfilteredd;
float PHH;

//=========== Sensores de Cor 1 =============

 int vals1[N];
 int vals2[N];
 int vals3[N];


//código normal- portas
const int ss0 = 24;
const int ss1 = 25;
```

```
const int ss2 = 26;
const int ss3 = 23;
const int outt = 22;

//variáveis para realizar a filtração do sinal e mapeamento
int R, G, B;
int Ro,Go,Bo;

byte countRed = 0;
byte countGreen = 0;
byte countBlue = 0;

//========== Sensores de Cor 2 ============

int valss1[N];
 int valss2[N];
 int valss3[N];


//código normal PORTAS
const int s0 = 32;
const int s1 = 33;
const int s2 = 34;
const int s3 = 31;
const int out = 30;

//variáveis para realizar a filtração do sinal e mapeamento
int RR, GG, BB;
int RRo,GGo,BBo;
int RRoo,GGoo,BBoo;

byte countRedd = 0;
byte countGreenn = 0;
byte countBluee = 0;

// ================== TELA NEXTION ====================
// ============== Declaração de Objetos ===================
//page id:0, id componente:1(vê isso na tela), nome do componente: "bt0"
#include  "Nextion.h"   //biblioteca Nextion

//---------------declaração dos números da tela----------------------
NexNumber valortelamilientra   = NexNumber(0, 25, "n0");
NexNumber valortelapressentra   = NexNumber(0, 16, "n2");
NexNumber valortelatempentra   = NexNumber(0, 19, "n3");
NexNumber valorphentra   = NexNumber(0, 34, "n4");
NexNumber valorcorRentra   = NexNumber(0, 35, "n6");
NexNumber valorcorGentra   = NexNumber(0, 41, "n5");
NexNumber valorcorBentra   = NexNumber(0, 42, "n8");
NexNumber valortempentra   = NexNumber(0, 36, "n11");
NexNumber valorpressabs   = NexNumber(0, 40, "n1");
```

```
NexNumber valorpressrela   = NexNumber(0, 38, "n13");
NexNumber valorphsaida   = NexNumber(0, 20, "n7");
NexNumber valorcorRsaida   = NexNumber(0, 43, "n9");
NexNumber valorcorGsaida   = NexNumber(0, 44, "n12");
NexNumber valorcorBsaida   = NexNumber(0, 45, "n14");
NexNumber valortempsaida   = NexNumber(0, 21, "n10");
NexNumber valoretapas   = NexNumber(0, 51, "n15");

//----------------------declaração dos botões da tela-------------------
NexDSButton botaoprototipo   = NexDSButton(0, 10, "bt1");
NexDSButton botaoetapas   = NexDSButton(0, 50, "bt3");
NexDSButton botaomisturador   = NexDSButton(0, 46, "bt2");

//armazena o estado do botão
uint32_t ButtonPro_var;// comando da biblioteca nextion que transforma byte em texto ou
etc..
uint32_t ButtonEta_var;
uint32_t ButtonMis_var;
uint32_t valorcontregressiva;
uint32_t valormilientra;

// ==================== BUZZER =====================

#define tempo 10
int Pinofalante = 10;

// ==================== RELÉS ====================
int rele_MISTURADOR = 14;
int rele_aguaentrada = 15;
int rele_aguasaida = 16;
int rele_ar = 17;


// ==================== CARTAO  ====================
#include <SD.h>
#include <SPI.h>
File myFile;
int pinCS = 53; // Pin 10 on Arduino Uno

//
================================================================================
================================================================================
==========================================
// ========================================== INICIO DO VOID SETUP
================================================================================
==============================================

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

```
//=========== BUZZER ============

pinMode(Pinofalante,OUTPUT); //Pino do buzzer

 //=========== TELA NEXTION ============

  nexInit();  //inicializa o tft

 //=========== Sensores de temperatura ============
  // put your setup code here, to run once:

  sensors.begin();// inicia o sensor TEMEPRATURA 1
  sensors.getAddress(sensor1, 0); // localiza o sensor TEMPERATURA 1
  sensorss.begin();// inicia o sensor TEMPERATURA 2
  sensorss.getAddress(sensor2, 0); // localiza o sensor TEMPERATURA 2

   //=========== Sensor de Fluxo ============
     pinMode(sensorPin, INPUT);
  digitalWrite(sensorPin, HIGH);

  pulseCount        = 0;
  flowRate          = 0.0;
  flowMilliLitres   = 0;
  totalMilliLitres  = 0;
  oldTime           = 0;

  // The Hall-effect sensor is connected to pin 2 which uses interrupt 0.
  // Configured to trigger on a FALLING state change (transition from HIGH
  // state to LOW state)
  attachInterrupt(sensorInterrupt, pulseCounter, FALLING);

 //=========== Sensores de COR 1 ============


   pinMode(ss0, OUTPUT);
   pinMode(ss1, OUTPUT);
   pinMode(ss2, OUTPUT);
   pinMode(ss3, OUTPUT);
   pinMode(outt, INPUT);
   digitalWrite(ss0, HIGH);
   digitalWrite(ss1, HIGH);

  //=========== Sensores de COR 2 ============

   //definindo portas

   pinMode(s0, OUTPUT);
   pinMode(s1, OUTPUT);
   pinMode(s2, OUTPUT);
   pinMode(s3, OUTPUT);
```

```
  pinMode(out, INPUT);
  digitalWrite(s0, HIGH);
  digitalWrite(s1, HIGH);

 //=========== RELÉS ============

 pinMode(rele_MISTURADOR, OUTPUT);
 pinMode(rele_aguaentrada, OUTPUT);
 pinMode(rele_aguasaida, OUTPUT);
 pinMode(rele_ar, OUTPUT);

 //=========== CARTÃO SD ============

  pinMode(pinCS, OUTPUT);
    // INICIALIZAÇÃO SD Card
    SD.begin();


 //============================================================
SETANDO AS VALVULAS TODAS FECHADAS(INICIO)
============================================================

     digitalWrite(rele_MISTURADOR,HIGH);
     digitalWrite(rele_aguaentrada,HIGH);
     digitalWrite(rele_aguasaida,LOW);
     digitalWrite(rele_ar,LOW);
}
//
============================================================
============================================================
==========================================
// ========================================== ININIO DO
LOOP============================================================
==========================================================
void loop() {

//=========== NEXTION ATUALIZAÇÃO DE VALORES ============

// OBJETO.COMANDO(VARIÁVEL)
valorphentra.setValue(getPh2());
valorcorRentra.setValue(getColorRed2());
valorcorGentra.setValue(getColorGreen2());
valorcorBentra.setValue(getColorBlue2());
valortempentra.setValue(getTEMPERATURA2());
valorpressabs.setValue(getPRESSABS());
valorpressrela.setValue(getPRESSRELA());
valorphsaida.setValue(getPh1());
valorcorRsaida.setValue(getColorRed1());
valorcorGsaida.setValue(getColorGreen1());
valorcorBsaida.setValue(getColorBlue1());
```

```
valortempsaida.setValue(getTEMPERATURA());



//
==========================================================================
==========================================================================
=========================================
//
==========================================================================
==INICIO DO
CÓDIGO==========================================================================
==================================

   //====================================== BOTÃO MISTURADOR
==================================
    botaomisturador.getValue(&ButtonMis_var);
    if(ButtonMis_var>0){
    digitalWrite(rele_MISTURADOR,LOW);
    }
    else {

   digitalWrite(rele_MISTURADOR,HIGH);

    }
    //lendo valor do botão protótipo
      botaoprototipo.getValue(&ButtonPro_var);
    while(ButtonPro_var >0)
    {

      //============================================================
SETANDO AS VALVULAS TODAS FECHADAS(INICIO)
============================================================================


      digitalWrite(rele_MISTURADOR,HIGH);
      digitalWrite(rele_aguaentrada,HIGH);
      digitalWrite(rele_aguasaida,LOW);
      digitalWrite(rele_ar,LOW);

      //============================================================
SETANDO AS VALVULAS TODAS FECHADAS(FIM)
============================================================================
      botaomisturador.setValue(0);// Desligando misturador
      digitalWrite(rele_MISTURADOR,HIGH);// Desligando o misturador
      botaoetapas.setValue(1);// Atuaziando Botão Etapas para STOP
      //
============================================================
GRAVANDO VALORES DE ENTRADA DA TELA
============================================================
      valortelamilientra.getValue(&valormilientra);// atribuindo valor para valor valormilientra;
```

```
     valortelatempentra.getValue(&valorcontregressiva); // atribuindo valor para valor
valorcontregressiva;


     //
==========================================================ETA
PA 1
INICIO=====================================================
===
    int  ETAPAS =1;// falando que é a etapa 1
    valoretapas.setValue(ETAPAS);// MOSTRANDO O NUMERO DE ETAPA
      // CRIANDO ARQUIVO DE TEXTO
        myFile = SD.open("tcc.txt", FILE_WRITE);
        myFile.println("====VALORES DE ENTRADA DA TELA=====");
        myFile.println("VOLUME DE TESTE EM MILILITROS:");
        myFile.println(valormilientra);
        myFile.println("TEMPO DE TESTE EM SEGUNDOS:");
        myFile.println(valorcontregressiva);
        myFile.println("");
        myFile.println("");

     botaoetapas.getValue(&ButtonEta_var);//Lendo valor do Botão

     if(ETAPAS == 1 && ButtonEta_var >0)// INICIO DA ETAPA 1
     {
      // while para atualizar valores durante 25 segundos
      int Contadoretapa1 = 0;
      while(Contadoretapa1 <= 25){// ATUALIZANDO OS VALORES DOS SENSORES
DURANTE 25 SEGUNDOS
      valorphentra.setValue(getPh2());
      valorcorRentra.setValue(getColorRed2());
      valorcorGentra.setValue(getColorGreen2());
      valorcorBentra.setValue(getColorBlue2());
      valortempentra.setValue(getTEMPERATURA2());
      Contadoretapa1++;
      delay(100);
       }

      // código para gravar no SD
        myFile.println("VALOR DE PH ENTRADA: ");
        myFile.println(getPh2());
        myFile.println("VALOR DE COR VERMELHA DE ENTRADA: ");
        myFile.println(getColorRed2());
        myFile.println("VALOR DE COR VERDE DE ENTRADA: ");
        myFile.println(getColorGreen2());
        myFile.println("VALOR DE COR AZUL DE ENTRADA: ");
        myFile.println(getColorBlue2());
        myFile.println("VALOR DE TEMPERATURA DE ENTRADA: ");
        myFile.println(getTEMPERATURA2());
```

```
      int contmili = valormilientra;// recebendo o valor da tela em inteiro
      int recontmili = 0;


//=================================================================
=== CÓDIGO WHILE PARA FLUXO  (INICIO) ======================
       while(contmili >= recontmili){// CONTAGEM REGRESSIVA
       // pegando valor do sensor de fluxo
        recontmili = getFLUXO();
       //ligando os relé de água de entrada e Ar
       digitalWrite(rele_aguaentrada,LOW);
       digitalWrite(rele_ar,HIGH);
       digitalWrite(rele_MISTURADOR,LOW);
       }
       digitalWrite(rele_MISTURADOR,HIGH);

//=================================================================
=== CÓDIGO WHILE PARA FLUXO  (FIM) ======================
       //FECHANDO VALVULA DE ENTRADA DE ÁGUA
       digitalWrite(rele_aguaentrada,HIGH);
       // MANTENDO VALVULA DE AR ABERTA
       digitalWrite(rele_ar,HIGH);

      botaoetapas.setValue(0);
      botaoetapas.getValue(&ButtonEta_var);
      while(ButtonEta_var == 0){// ENQUANTO BOTÃO ETAPA NÃO FOR
PRESSIONADO O BUZZER TOCARÁ
       tone(Pinofalante, 500, 1000);
       delay(3000);
       valorpressabs.setValue(getPRESSABS());
       valorpressrela.setValue(getPRESSRELA());
       botaoetapas.getValue(&ButtonEta_var);//LENDO BOTÃO ETAPA
       }
       ETAPAS = 2;// ATRIBUI PARA PRÓXIMA ETAPA
       valoretapas.setValue(ETAPAS);
      }
      //
==========================================================ETA
PA 2
INICIO=========================================================
===

     botaoetapas.getValue(&ButtonEta_var);//Lendo valor do Botão

    if(ETAPAS == 2 && ButtonEta_var >0)// INICIO DA ETAPA 2
     {

      int contregressiva = valorcontregressiva;
      int recontregressiva = 0;
```

```
//=================================================================
=== CÓDIGO WHILE PARA CONTAGEM REGRESSIVA (INICIO)
======================
      while(contregressiva >= recontregressiva){// CONTAGEM REGRESSIVA

      valortelatempentra.setValue(contregressiva);
      contregressiva--;
      delay(1000);

      }

//=================================================================
=== CÓDIGO WHILE PARA CONTAGEM REGRESSIVA (FIM)
======================

     botaoetapas.setValue(0);              // SETANDO VALOR DO BOTÃO ETAPA PARA 0
(PLAY)
     botaoetapas.getValue(&ButtonEta_var);   // LENDO VALOR DO BOTAO
     while(ButtonEta_var == 0){// ENQUANTO BOTÃO ETAPA NÃO FOR
PRESSIONADO O BUZZER TOCARÁ
      tone(Pinofalante, 100, 1000);
      delay(5000);
      digitalWrite(rele_aguasaida,HIGH);//ABRINDO RELÉ DE SAIDA DE ÁGUA
      digitalWrite(rele_ar,HIGH);// ABRINDO RELÉ DE ENTRADA DE AR
      botaoetapas.getValue(&ButtonEta_var);//LENDO BOTÃO ETAPA
      }

      ETAPAS = 3;
      valoretapas.setValue(ETAPAS); // MOSTRANDO VALORES DAS ETAPAS
     }
      //
==========================================================ETA
PA 3
INICIO=========================================================
===

     if(ETAPAS == 3 && ButtonEta_var >0)// INICIO DA ETAPA 3
      {

     // contador para grava no sd
     int Contadoretapa3 = 0;
     while(Contadoretapa3 <= 25){// ATUALIZANDO OS VALORES DOS SENSORES
DURANTE 25 SEGUNDOS
       valorphsaida.setValue(getPh1());
       valorcorRsaida.setValue(getColorRed1());
       valorcorGsaida.setValue(getColorGreen1());
       valorcorBsaida.setValue(getColorBlue1());
       valortempsaida.setValue(getTEMPERATURA());
```

```
        Contadoretapa3++;
        delay(100);
       }
        // código para gravar no SD
        myFile.println("VALOR DE PH SAÍDA: ");
        myFile.println(getPh1());
        myFile.println("VALOR DE COR VERMELHA DE SAÍDA: ");
        myFile.println(getColorRed1());
        myFile.println("VALOR DE COR VERDE DE SAÍDA: ");
        myFile.println(getColorGreen1());
        myFile.println("VALOR DE COR AZUL DE SAÍDA: ");
        myFile.println(getColorBlue1());
        myFile.println("VALOR DE TEMPERATURA DE SAÍDA: ");
        myFile.println(getTEMPERATURA());

myFile.println("==========================================================
==================");
        myFile.println("");
        myFile.println("");

       }

     botaoetapas.setValue(0);
     botaoetapas.getValue(&ButtonEta_var);
     tone(Pinofalante, 2000, 3000);
     delay(3000);

     botaoprototipo.setValue(0);//Colocando botão protótipo em stop
     ButtonPro_var = (0);// Finalizando o while
     myFile.close(); // close the file
   }
   botaoetapas.setValue(0);

    int  ETAPAS =0;// falando que é a etapa 1
    valoretapas.setValue(ETAPAS);// MOSTRANDO O NUMERO DE ETAPA


//
============================================================================
============================================================================
=============================================
//
============================================================================
==FIM DO
CÓDIGO=====================================================================
================================
}
//
============================================================================
```

```
//=============================================================================
//
//=============================================================================
//===      FUNÇÕES
//=============================================================================
//=========== FUNÇÃO TEMPERATURA 1============

float getTEMPERATURA(){
   // Le a informacao do sensor
  sensors.requestTemperatures();
  tempC = sensors.getTempC(sensor1);
  return tempC;
 }
//=========== FUNÇÃO TEMPERATURA 2============
float getTEMPERATURA2(){
   // Le a informacao do sensor
 sensorss.requestTemperatures();
 tempC2 = sensorss.getTempC(sensor2);
 return tempC2;
 }


//=========== FUNÇÃO FLUXO =============
float getFLUXO(){

   if((millis() - oldTime) > 1000)    // Only process counters once per second
 {
   // Disable the interrupt while calculating flow rate and sending the value to
   // the host
   detachInterrupt(sensorInterrupt);

   // Because this loop may not complete in exactly 1 second intervals we calculate
   // the number of milliseconds that have passed since the last execution and use
   // that to scale the output. We also apply the calibrationFactor to scale the output
   // based on the number of pulses per second per units of measure (litres/minute in
   // this case) coming from the sensor.
   flowRate = ((1000.0 / (millis() - oldTime)) * pulseCount) / calibrationFactor;

   // Note the time this processing pass was executed. Note that because we've
   // disabled interrupts the millis() function won't actually be incrementing right
   // at this point, but it will still return the value it was set to just before
   // interrupts went away.
   oldTime = millis();

   // Divide the flow rate in litres/minute by 60 to determine how many litres have
   // passed through the sensor in this 1 second interval, then multiply by 1000 to
   // convert to millilitres.
   flowMilliLitres = (flowRate / 60) * 1000;
```

```
    // Add the millilitres passed in this second to the cumulative total
    totalMilliLitres += flowMilliLitres;



    // Determine the fractional part. The 10 multiplier gives us 1 decimal place.
    frac = (flowRate - int(flowRate)) * 10;


    // Reset the pulse counter so we can start incrementing again
    pulseCount = 0;

    // Enable the interrupt again now that we've finished sending output
    attachInterrupt(sensorInterrupt, pulseCounter, FALLING);
  }
   return totalMilliLitres;
  }
/*
Insterrupt Service Routine
 */
void pulseCounter()
{
  // Increment the pulse counter
  pulseCount++;
}

 //=========== FUNÇÃO PRESSÃO RELATIVA =============
 float getPRESSRELA()
{
  // Leitura dos pinos e filtragem:
   sensorValue1 = (analogRead(A9)-40.96)/1,3166592; //Calculo para calibração // A8

   //FILTRO PRESSÃO RELATIVA
    for(int i = N -1; i > 0; i--){
    PRESSRELA[i] = PRESSRELA [i-1];
    }
    PRESSRELA[0] = sensorValue1;
    //long
    long sumPRESSRELA = 0;                          // testar como long float ou double
    for(int i = 0; i < N; i++){
    sumPRESSRELA = sumPRESSRELA + PRESSRELA[i];
    }
  //sum vai ser a soma de todos os valores
    PRESSRELAfiltered = (sumPRESSRELA/N) + SensorOffset1;

   return(PRESSRELAfiltered);
}

 //=========== FUNÇÃO PRESSÃO ABSOLUTA =============
```

```
 float getPRESSABS()
{
  // Fazendo a leitura do sensor
   sensorValue = (analogRead(A8)-40.96)/1,3166592; //Do maths for calibration // A9
   //FILTRO PRESSÃO ABSOLUTA
    for(int i = N -1; i > 0; i--){
    PRESSABS[i] = PRESSABS [i-1];
    }
    PRESSABS[0] = sensorValue;
    //long
    long sumPRESSABS = 0;
    for(int i = 0; i < N; i++){
    sumPRESSABS = sumPRESSABS + PRESSABS[i];
    }
  //sum vai ser a soma de todos os valores
    PRESSABSfiltered = (sumPRESSABS/N) + SensorOffset;

   return(PRESSABSfiltered);
}

 //=========== FUNÇÃO PH 1 ============
float getPh1()
{
  Phsensor = (analogRead(sensorPh));

   //FILTRO VERMELHO
    for(int i = N -1; i > 0; i--){
    Ph1[i] = Ph1 [i-1];
    }
    Ph1[0] = Phsensor;
    //long
    long sumPh = 0;
    for(int i = 0; i < N; i++){
    sumPh = sumPh + Ph1[i];
    }
  //sum vai ser a soma de todos os valores
    Phsensorfiltered = sumPh/N;
    PH = -0.0290856031128*(Phsensorfiltered - (2*getTEMPERATURA())) +
22.1012451361866; // temperatura no caso -2*temperatura
    return PH;
}

 //=========== FUNÇÃO PH 2 ============
float getPh2()
{
  Phsensorr = (analogRead(sensorPhh));

   //FILTRO VERMELHO
    for(int i = N -1; i > 0; i--){
    Phh1[i] = Phh1 [i-1];
```

```
    }
    Phh1[0] = Phsensorr;
    //long
    long sumPhh = 0;
    for(int i = 0; i < N; i++){
    sumPhh = sumPhh + Phh1[i];
    }
  //sum vai ser a soma de todos os valores
    Phsensorfilteredd = sumPhh/N;
    PHH = -0.0373750000000*(Phsensorfilteredd - (2*getTEMPERATURA2())) +
25.7339999999999; // caso -2*temperatura
    return PHH;
}

//=========== FUNÇÃO COR 1 =============

int getColorRed1()
{

   digitalWrite(ss2, LOW);
   digitalWrite(ss3, LOW);
   countRed = pulseIn(outt, digitalRead(outt) == HIGH ? LOW : HIGH);
//mapeando valores de 0 a 255
   R = map(countRed,7,42,255,0);// original 36
   digitalWrite(ss3, HIGH);
   digitalWrite(ss2, HIGH);


   //FILTRO VERMELHO
    for(int i = N -1; i > 0; i--){
    vals1[i] = vals1 [i-1];
    }
    vals1[0] = R;
    //long
    long sum1 = 0;
    for(int i = 0; i < N; i++){
    sum1 = sum1+ vals1[i];
    }
  //sum vai ser a soma de todos os valores
    Ro = sum1/N;
    return Ro;
}

int getColorGreen1()
{

   digitalWrite(ss2, LOW);
   digitalWrite(ss3, LOW);
   digitalWrite(ss3, HIGH);
   countBlue = pulseIn(outt, digitalRead(outt) == HIGH ? LOW : HIGH);
```

```
//mapeando valores de 0 a 255
   G = map(countGreen,8,46,255,0);// original 39
   digitalWrite(ss2, HIGH);

// FILTRO GREEN
    for(int i = N -1; i > 0; i--){
    vals2[i] = vals2 [i-1];
    }
    vals2[0] = G;
    //long
    long sum2 = 0;
    for(int i = 0; i < N; i++){
    sum2 = sum2+ vals2[i];
    }
  //sum vai ser a soma de todos os valores
    Go = sum2/N;
    return Go;
}

int getColorBlue1()
{

   digitalWrite(ss2, LOW);
   digitalWrite(ss3, LOW);
   digitalWrite(ss3, HIGH);
   digitalWrite(ss2, HIGH);
   countGreen = pulseIn(outt, digitalRead(outt) == HIGH ? LOW : HIGH);
//mapeando valores de 0 a 255
   B = map(countBlue,6,37,255,0);// original 32

// FILTRO BLUE
    for(int i = N -1; i > 0; i--){
    vals3[i] = vals3 [i-1];
    }
    vals3[0] = B;
    //long
    long sum3 = 0;
    for(int i = 0; i < N; i++){
    sum3 = sum3+ vals3[i];
    }
  //sum vai ser a soma de todos os valores
    Bo = sum3/N;
    return Bo;
}

//=========== FUNÇÃO COR 2 =============

int getColorRed2()
{
```

```
  digitalWrite(s2, LOW);
  digitalWrite(s3, LOW);
  countRedd = pulseIn(out, digitalRead(out) == HIGH ? LOW : HIGH);
  //mapeando valores de 0 a 255
  RR = map(countRedd,6,49,255,0);// original 44
  digitalWrite(s3, HIGH);
  digitalWrite(s2, HIGH);



  //FILTRO VERMELHO
   for(int i = N -1; i > 0; i--){
   valss1[i] = valss1 [i-1];
   }
   valss1[0] = RR;
   //long
   long summ1 = 0;
   for(int i = 0; i < N; i++){
   summ1 = summ1+ valss1[i];
   }
 //sum vai ser a soma de todos os valores
   RRo = summ1/N;
   return RRo;
}

int getColorGreen2()
{

  digitalWrite(s2, LOW);
  digitalWrite(s3, LOW);
  digitalWrite(s3, HIGH);
  countBluee = pulseIn(out, digitalRead(out) == HIGH ? LOW : HIGH);
  //mapeando valores de 0 a 255
  GG = map(countGreenn,6,51,255,0);// original 46
  digitalWrite(s2, HIGH);


// FILTRO GREEN
   for(int i = N -1; i > 0; i--){
   valss2[i] = valss2 [i-1];
   }
   valss2[0] = GG;
   //long
   long summ2 = 0;
   for(int i = 0; i < N; i++){
   summ2 = summ2+ valss2[i];
   }
 //sum vai ser a soma de todos os valores
   GGo = summ2/N;
   return GGo;
```

```
}

int getColorBlue2()
{
  digitalWrite(s2, LOW);
  digitalWrite(s3, LOW);
  digitalWrite(s3, HIGH);
  digitalWrite(s2, HIGH);
  countGreenn = pulseIn(out, digitalRead(out) == HIGH ? LOW : HIGH);
//mapeando valores de 0 a 255
  BB = map(countBluee,5,41,255,0);//original 38

// FILTRO BLUE
   for(int i = N -1; i > 0; i--){
   valss3[i] = valss3 [i-1];
   }
   valss3[0] = BB;
   //long
   long summ3 = 0;
   for(int i = 0; i < N; i++){
   summ3 = summ3+ valss3[i];
   }
 //sum vai ser a soma de todos os valores
   BBo = summ3/N;
   return BBo;
}
```