UNIVERSIDADE DE PASSO FUNDO

Vitor Tramontina Casassola

ALIMENTADOR AUTOMÁTICO DE BEZERROS

Passo Fundo

2018

Vitor Tramontina Casassola

ALIMENTADOR AUTOMÁTICO DE BEZERROS

Trabalho apresentado ao curso de Engenharia Elétrica, da Faculdade de Engenharia e Arquitetura, da Universidade de Passo Fundo, como requisito parcial para obtenção do grau de Engenheiro Eletricista, sob orientação do professor Dr. Adriano Luís Toazza.

Passo Fundo

2018

Vitor Tramontina Casassola

Alimentador automático de bezerros

Trabalho apresentado ao curso de Engenharia Elétrica, da Faculdade de Engenharia e Arquitetura, da Universidade de Passo Fundo, como requisito parcial para obtenção do grau de Engenheiro Eletricista, sob orientação do professor Dr. Adriano Luís Toazza.

Aprovado em ____ de _____ de_____.

BANCA EXAMINADORA

_____

Prof. Dr. Orientador Adriano Luís Toazza - UPF

_____

Prof. Dr. Eduardo Appel - UPF

_____

Prof. Dr. Carlos Alberto Ramirez Behaine - UPF

AGRADECIMENTOS

Nessa longa caminhada do curso de Engenharia Elétrica aprendi que não devemos desistir diante dos desafios por mais difíceis que sejam, que quando conseguimos vencer um desafio é que ocorre o crescimento, tanto pessoal quanto profissional.

Agradeço aos meus pais pela oportunidade de estudar, de conseguir me dedicar integralmente ao processo de profissionalização, por me ajudarem nas horas que mais preciso de apoio.

“A grandeza vem não quando as coisas sempre vão bem para você, mas a grandeza vem quando você é realmente testado, quando você sofre alguns golpes, algumas decepções, quando a tristeza chega. Porque apenas se você esteve nos mais profundos vales você poderá um dia saber o quão magnífico é estar no topo da mais alta montanha.”

Richard Milhous Nixon

RESUMO

A produção de leite é uma atividade fundamental para economia atual, além disso o leite é um alimento rico em vitaminas e muito nutritivo. Para melhorar a produção, é necessário entender que o começo da economia é a reposição das fêmeas leiteiras, por esse motivo a criação dos bezerros deve ser feita cuidadosamente. Diante disso, o objetivo do trabalho é fazer a alimentação dos animais de maneira automática. Para isso, foi desenvolvido um protótipo capaz de interpretar *tags* de *Radio Frequency Identification* (RFID) para fazer a identificação dos animais e preparar o alimento, baseado em leite em pó, conforme a necessidade. O protótipo conta com sensores para fazer a dosagem correta de água, com uma interface interativa com usuário utilizando um *display touch screen* e uma *Launchpad* TM4C123GXL para fazer o controle de todas as ações do projeto.

Palavras-chaves: Alimentador automático de bezerras, leite em pó, criação de bezerras, produção leiteira, desmama gradativa, RFID, *Launchpad* e TM4C123GXL.

ABSTRACT

The milk production is an important activity for the actual economy, besides that the milk is an aliment with a lot of vitamins and very nutritious. To improve the production, it's necessary to understand the beginning of the economy is the replacement of dairy females, that's why the rearing of calves must be done carefully. Because of this, the objective of this work is to do all the functions about the feeding of the calves automatically. That's why it was developed a prototype that is capable of reading RFID tags to identify the animals and make the needed meals, based in powdered milk. The prototype has sensors to verify the amount of water used, an user interface with a display touch screen and a Launchpad TM4C123GXL to control all actions of the project.

Keywords: Automatic calf feeder, powdered milk, calf rearing, milk production, correct weaning, RFID, Launchpad and TM4C123GXL.

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS

º C = Graus Celsius

CONAB – Companhia Nacional de Abastecimento

J – Joule

K - Kelvin

m – Massa de um material

ΔT – Diferença de temperatura, variação

Ω - Ohm

RFID – *Radio frequency identification*

VDC – *Voltage Direct Current*

VAC – *Voltage Alternate Current*

W – Watts

A – Ampere

V – Volts

kg – quilograma

g – grama

CI – Circuito Integrado

J - Joule

ms – milissegundos

us - microsegundos

USB – *Universal Serial Bus*

UART – *Universal Asynchronous Receiver / Transmitter*

SPI – *Serial Peripheral Interface*

I²C – *Inter-Integrated Circuit*

IFF – *Identify Friend or Foe*

SUMÁRIO

1 INTRODUÇÃO

O leite está entre os principais produtos da agropecuária brasileira, desempenhando um papel relevante no suprimento de alimentos e na geração de emprego e renda para população. Em 2016, o Brasil produziu 34.650.000 toneladas de leite de vaca (foi considerado 1 litro de leite igual a 1,032kg), assim garantiu o seu lugar na quinta posição dos maiores produtores mundiais de leite. (CONAB, 2017a).

Além da importância econômica, o leite é um alimento natural com um grande valor nutritivo, pois possui uma grande concentração de cálcio, que é fundamental para boa formação e condicionamento dos ossos. Também, contém muitas vitaminas (como por exemplo as vitaminas A, B1 e B2) e sais minerais que favorecem uma vida saudável. (CARVALHO et al., 2002).

Como em qualquer atividade, o seu sucesso depende de como o primeiro passo é dado. Para exploração leiteira, pode-se considerar a criação de bezerras como o primeiro passo para o sucesso dessa grande atividade econômica, pois a criação de fêmeas de reposição é extremamente importante ao se considerar que a rápida substituição de fêmeas mais velhas por animais jovens e mais produtivos é essencial para o crescimento da produção. (PEIXOTO, MOURA e DE FARIA, 2000).

Os cuidados com os bezerros começam na gestação da vaca. Aproximadamente 60 dias antes do parto provável, é fundamental fazer a secagem da vaca, ou seja, interromper a sua lactação afim de regenerar os seus tecidos secretores de leite. Desse modo o colostro produzido tem uma qualidade superior, o que é essencial para sobrevivência da cria recém-nascida.

Segundo KRUG et al. (1993) a secagem da vaca deve ser feita da seguinte maneira: transferi-la do local onde está acostumada a ser ordenhada; acomodá-la em um local com pouco pasto, porém com água a vontade; não a ordenhar mesmo que o úbere (teta) estiver cheio de leite, pois seu próprio organismo absorverá o leite produzido em excesso. Após o período de duas semanas, a vaca não produzirá mais leite e a secagem estará completa.

Os bezerros nascem praticamente desprovidos de anticorpos contra as doenças a que estarão sujeitos no período neonatal. Dessa maneira, os anticorpos serão fornecidos para o recém-nascido via colostro, ou seja, o primeiro leite ordenhado da mãe contém o maior número de anticorpos e fará a proteção do animal. (BATISTTON, 1983).

Há diferentes maneiras de fornecer leite aos bezerros, as principais são: aleitamento natural e artificial. No aleitamento natural, o bezerro permanece ao lado da mãe depois do

parto e se alimenta diretamente do úbere. Já no aleitamento artificial os bezerros devem ser apartados da vaca, logo após o nascimento, e recebem a dieta líquida (leite, colostro excedente ou sucedâneo de leite) em balde, mamadeira ou biberão. Para o método artificial ter sucesso, a higiene é fundamental, pois alguma contaminação durante a fase inicial da vida do animal gera um risco muito elevado para sua sobrevivência. (KRUG et al., 1993).

As vantagens do método artificial são: racionalizar o manejo dos animais, separando os bezerros das vacas; controle da quantidade de alimento por animal; ordenha higiênica; menor incidência de doenças; controle de custos na fase de cria; planejamento do desaleitamento.

O aleitamento artificial permanecerá até os 60 dias de vida do animal, quando será feito o desaleitamento (desmama), ou seja, termina a alimentação a partir do leite e passa a consumir apenas concentrado. Dessa maneira, reduz-se o gasto com sucedâneos de leite e passa utilizar o pasto como fonte principal de alimentação. Esse passo traz algumas vantagens ao produtor, como redução no custo da alimentação, redução da mão-de-obra e menores ocorrências de distúrbios gastrintestinais. (BATISTTON, 1983).

Nota-se que a alimentação e nutrição correta das bezerras são fundamentais para o bom desenvolvimento, ou seja, para que, quando completamente maduras, tenham uma produtividade excelente. Porém, a criação desses animais em locais de pequeno e médio porte ainda utiliza de mão-de-obra humana para trabalhos repetitivos, como dosar a quantidade ideal de alimento e verificar se o animal realmente completou sua refeição corretamente. Isso acontece devido ao elevado preço das máquinas automáticas presentes no mercado.

1.1 OBJETIVO GERAL

O objetivo do projeto é construir um alimentador automático para bezerros, o qual utilizará de leite em pó para fazer o alimento. Para dar flexibilidade ao produtor, o sistema identificará qual a idade do animal a ser alimentado utilizando de um sensor RFID (*radio frequency identification*) e, com isso, calcular a quantidade de alimento correta.

1.2 OBJETIVOS ESPECÍFICOS

- Desenvolver um reservatório para leite em pó com acoplamento em uma rosca helicoidal;
- Desenvolver um sistema para descompactar o leite em pó;

- Projetar um reservatório de água com eletroválvulas para saída e entrada;
- Projetar um sistema para aquecimento e controle da temperatura da água.
- Projetar um sistema para fazer a mistura entre a água, já aquecida, e o leite em pó;
- Desenvolver sistema de identificação utilizando sensores RFID;
- Desenvolver o hardware para realizar todos os acionamentos necessários do projeto;
- Desenvolver o *firmware* para controlar todo sistema do projeto;
- Desenvolver método de interface com o usuário para parametrização de dados utilizados de cada produtor;

1.3 JUSTIFICATIVA

Há diversas vantagens obtidas com a automação da alimentação das bezerras, sendo que as principais delas são: redução da mão-de-obra utilizada para o processo de preparação do alimento e dar o alimento; diminuição do contato humano com os animais; aumento da higienização da alimentação, diminuindo a chance de adquirir doenças; alimento disponível 24 horas por dia; nutrição correta dos animais, sem excessos ou subnutrição.

Além disso, as máquinas existentes no mercado são extremamente caras e robustas, impossibilitando produtores de médio e pequeno porte de adquiri-las. Desse modo, o trabalho visa maior simplicidade em relação aos produtos citados, diminuindo o custo e garantindo maior acesso aos produtores.

2 REVISÃO DA LITERATURA

Esse capítulo apresentará os conhecimentos necessários para a execução do projeto, como por exemplo, a quantidade de alimento, sua temperatura, como fazer a desmama e assuntos sobre os componentes elétricos a serem utilizados no projeto.

2.1 ALEITAMENTO ARTIFICIAL

Quando o animal recém-nascido não é alimentado diretamente pela figura materna, mas sim por utensílios como mamadeira e baldes, isso é o que se chama de aleitamento artificial. Segundo KRUG et al. (1993), os filhotes devem ser separados da mãe logo após a ingestão da dose mínima necessária de colostro, a fim de evitar que os terneiros tornem-se um estímulo à descida do leite da vaca, sendo extremamente desfavorável para produção.

"O período de ingestão do colostro deve durar de 5 a 7 dias, mesmo se a absorção das imunoglobulinas se encerra em torno de 24 horas após o parto." (KRUG et al., 1993, p.324). Além disso, até os 15 dias de vida o animal deverá receber somente leite de vaca e depois desse período pode-se considerar outros tipos de alimentos. (BATISTTON, 1983).

2.1.1 Substituto Lácteo em pó

Como já foi mencionado, há diversas vantagens na utilização do substituto lácteo sendo a principal delas a facilidade de armazenamento, pois diferentemente do leite natural não há necessidade de resfriamento.

Pode-se afirmar que o leite em pó também traz economia ao produtor, pois o litro de alimento produzido é mais barato que o preço de venda do litro de vaca. Segundo a CONAB (2017b) o preço médio bruto pago ao produtor brasileiro em agosto de 2017 foi de R$ 1,26 por litro de leite entregue no mês de julho. Agora o leite em pó *Real Milk* ofertado no site "Loja Agropecuária" tem o preço de R$69,00 por 10 kg, e sua receita diz, como mostra a Figura 1, que 1 quilograma do produto é usado em 9 litros de água.

Figura 1 - Bula do substituto lácteo *Real Milk.*

LEITE EM PÓ REAL MILK

Descrição:
O **Leite em pó Real Milk** é um alimento substitutivo do leite para alimentação animal. Com ele seu bezerro fica forte e saudável!

Benefícios:
Leite em pó Real Milk para maior crescimento para o rebanho.
Para mais produção de leite.
Para maior rentabilidade para você.

Níveis de garantia do produto Leite em pó Real Milk:
Cálcio (máx).....7500mg/kg
Cálcio (mín).....3000mg/kg
Fósforo (mín).....2000mg/kg
Selênio (mín).....8mg/kg
Carboidratos (mín).....295g/kg
Extrato Etéreo (mín).....95g/kg
Matéria Fibrosa (máx).....40g/kg
FDA (máx).....33g/kg
Matéria Mineral (máx).....40g/kg
Proteína Bruta (mín).....210g/kg
Umidade (máx).....90g/kg
Energia Metabolizável.....3.800kcal/kg

Indicação de uso do produto Leite em pó Real Milk:
Diluído em água morna (40-45ºC): na proporção de 1kg para 9 litros de água. Ou seja 1 medida cheia da caixa contida "dentro da embalagem" para cada litro de água utilizado.

Espécies e categorias de animais e modo de usar o produto Leite em pó Real Milk:
Esse produto é recomendado para utilização em bovinos (bezerras e bezerros), conforme a seguir:
A partir do 7º dia de vida até o 9º dia, fazer uma mistura de 50% de leite de vaca e 50% de **Leite em pó Real Milk** já diluído em água morna e fornecer a mistura: 2 litros, 2 vezes ao dia (totalizando 4 litros ao dia).
A partir do 10º dia de vida até o 15º dia, fornecer: 2 litros de **Leite em pó Real Milk** já preparado, 2 vezes ao dia (totalizando 4 litros ao dia).
Após o 16º dia de vida até o desmame, fornecer: 2,5 litros de **Leite em pó Real Milk** já preparado, 2 vezes ao dia (totalizando 5 litros ao dia).

Fonte: Adaptado de LOJA AGROPECUÁRIA

Considerando o preço da tarifa de água informado pela Prefeitura Municipal de Porto Alegre (2017) sendo de R$ 3,25 por metro cúbico, o produtor gastaria para produzir 100 kg do substituto lácteo cerca de R$ 69,29 (desconsiderando as taxas de esgoto).

Em contrapartida, 100 litros de leite de vaca rendem ao produtor em média R$ 126 (preço bruto, desconsiderando taxas de transporte, armazenamento e higienização dos materiais), desse modo a economia na alimentação é muito significativa, guardando R$ 56,71.

Há diversos fabricantes de leite em pó, cada um tem sua própria receita e modo de preparo, por esse motivo o protótipo a ser construído necessita ter uma interface com o usuário, pois o produtor deverá escolher quantas gramas por litro será utilizado na mistura.

Há fabricantes que contém programas de alimentação já preparados para criação de gado leiteiro, como por exemplo a empresa *Trouw Nutrition*. Essa companhia tem diferentes produtos para substitutos lácteos, cada um com sua composição própria, o que altera o modo de preparo, pois a quantidade de pó por litro de água não é constante. Segundo o fabricante Sprayfo (2017), a concentração do produto deve manter a relação de 140 gramas por litro de água. O quadro 1 mostra um exemplo de rotina de alimentação dos animais utilizando o produto da *Sprayfo*.

Quadro 1 - Programa de Alimentação

Idade	Colostro por dia	Fase de colostro
dia 1	6,0 L	Primeiro colostro, durante a primeira hora: 4 L Total: 6 L
dia 2	3 x 2,0 L	Colostro (Primeira ou segunda ordenha)
dia 3	3 x 2,0 L	Colostro ou Sprayfo

Idade	Sprayfo por dia	Relação de mistura Sprayfo + água	Volumosos, concentrados e água à vontade
dia 4 - 7	3 x 2,0 L	1 kg + 7 L	
dia 8 - 14	2 x 3,0 L	1 kg + 7 L	Pelete inicial granulado

Transferir as bezerras sadias para o alimentador automático. Desmame entre 12 e 14 semanas de vida.

Fase	dias	porções	de	até	Concentração		
Início	7	4	6	7 L	140 g / L	Pelete inicial granulado	Feno picado
Principal ou Pré-desmam	14	4	7	7 L	140 g / L	Pelete inicial granulado	Feno picado
Desmame	28	3	7	1 L	140 g / L	Pelete inicial granulado	Feno picado

Fonte: Adaptado de SPRAYFO

> Devido aos terneiros apresentarem dificuldades em digerir proteínas, amidos e açúcares de origem vegetal, por não disporem, nesta fase, de enzimas para decompor esses nutrientes em formas mais simples, é recomendável verificar a composição do substitutivo do leite, pois um bom produto deve conter, no mínimo, 20% de proteínas e lactose, e de 15 a 20% de gordura de origem animal. Com a utilização do leite substitutivo, o produtor estará associando todas as vantagens do aleitamento artificial, com a liberação do leite animal para a comercialização, o que é realmente compensador, quando se compara o preço do substitutivo com aquele pago ao produtor pelo leite in natura. (KRUG et al., 1993, p. 327).

Analisando a Figura 1 pode-se concluir que o produto alcança aos requisitos propostos pelos autores Krug et al. (1993), o que justifica sua utilização.

2.1.2 Modo de Preparo do Alimento

Pode-se notar a concordância entre os fabricantes do substituto lácteo e os autores Krug et al. (1993) no modo de preparação do alimento e na temperatura ideal para alimentar os bezerros. O método que será utilizado terá os seguintes procedimentos:

1. Adicionar o substituto lácteo na proporção correta para alimentar o animal;
2. Adicionar água na temperatura de 45ºC e misturar o alimento para eliminar as partes não dissolvidas, pois essas partes causam problemas digestivos.
3. Tratar os animais com a mistura na temperatura de 37º C a 40º C.

Segundo o manual da Sprayfo (2017), é muito importante o alimento estar na temperatura correta, pois desse modo induz o reflexo necessário no estômago.

2.1.3 Desaleitamento Precoce

Os custos com alimentação e mão-de-obra associados à criação de bezerros são muito maiores no período inicial do que após essa etapa (DE SOUZA, 2011). Desse modo, a aceleração do período de desmama é essencial no ponto de vista econômico, visto que não são necessárias grandes quantidades de leite para o pleno desenvolvimento dos animais. (BATISTTON, 1983).

> Sabe-se hoje, que os bezerros podem realizar a ruminação antes de completarem o 2.º mês de vida, desde que a sua alimentação seja controlada. Pesquisas demonstraram que a partir da 3.ª semana de vida, o animal já pode digerir pequena quantidade de gramíneas, tanto de piquetes como de fenos e também alguns concentrados. (BATISTTON, 1983, p. 346).

Os seguintes critérios devem ser analisados para saber o momento ideal para desmamar os animais: idade, peso vivo, ganho de peso, total da dieta líquida consumida ou consumo diário de concentrado. Segundo os autores Peixoto, De Moura e De Faria (2000) deve-se levar em conta dois ou mais desses pré-requisitos para desleitar. Levando isso em consideração, os critérios mais utilizados são a idade e o consumo de concentrado diário, pois não necessitam de medições, como no caso do ganho de peso, para serem implementados. O critério de idade é analisado pelo Quadro 2.

Quadro 2 - Relação entre a taxa de mortalidade e a idade ao desaleitamento

Idade ao desaleitamento (semanas)	Mortalidade (%)	
	0-30 dias	0-182 dias
3 a 4	8,8	14,2
5 a 6	5,1	9
7 a 8	4,6	7,1
Acima de 8	3,4	6,1

Fonte: Adaptado de PEIXOTO, DE MOURA e DE FARIA (2000).

Para fazer o desaleitamento de maneira gradativa, são considerados necessários dois fatos fundamentais: diminuição gradativa do fornecimento de leite e disponibilização de concentrados a partir da segunda ou terceira semana de vida dos animais para incentivar o desenvolvimento das papilas ruminais. Segundo De Souza (2011) devem ser fornecidos alimentos sólidos à vontade, pois a medida que se diminui a quantidade de leite, maior será o consumo de concentrado, ou seja, a ingestão de alimentos secos estimula o crescimento do rúmen. A quantidade detalhada de alimento que deve ser fornecido aos animais é mostrada no Quadro 3.

Quadro 3 - Programa de Alimentação Detalhado

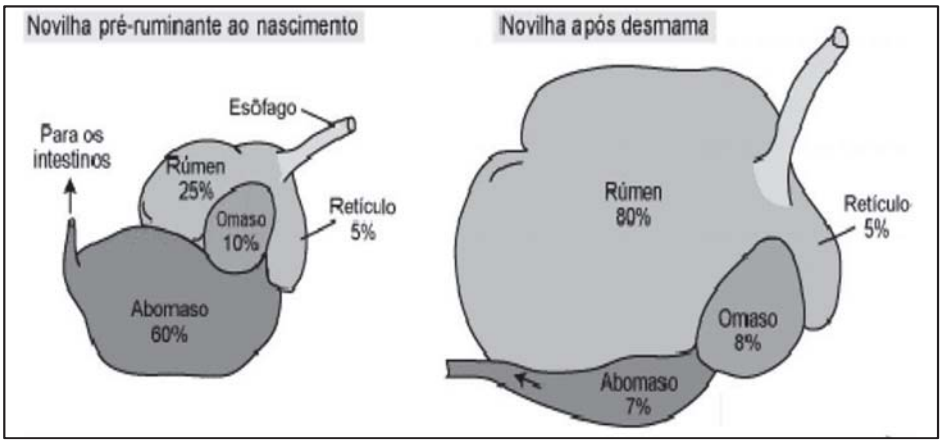
Idade	Sprayfo por dia	Relação de mistura Sprayfo + água	Forragem - Concentrados + água à vontade	
dia 4 - 7	3 x 2,0 L	1 kg + 7 L		
semana 2	2 x 3,0 L	1 kg + 7 L	Pelete de início granulado	
semana 3	2 x 3,5 L	1 kg + 7 L	Pelete de início granulado	Palha picada
semana 4	2 x 3,5 L	1 kg + 7 L	Pelete de início granulado	Palha picada
semana 5	2 x 3,5 L	1 kg + 7 L	Pelete de início granulado	Palha picada
semana 6	2 x 3,0 L	1 kg + 7 L	Pelete de início granulado	Palha picada
semana 7	2 x 2,0 L	1 kg + 7 L	Pelete de início granulado	Palha picada
semana 8	2 x 2,0 L	1 kg + 7 L	Pelete de início granulado	Palha picada
semana 9	1 x 2,0 L	1 kg + 7 L	Pelete de início granulado	Palha picada

Fonte: Adaptado de SPRAYFO

Peixoto, De Moura e De Faria (2000) defendem que a melhor maneira de se efetuar o desaleitamento é através do corte abrupto no fornecimento de leite quando alguns critérios já mencionados forem atendidos. Desse modo, os bezerros aumentam prontamente o consumo de concentrados, atingindo níveis diários de 1,5 kg ou mais poucos dias após a desmama.

O tamanho do rúmen de um bezerro recém-nascido representa 25% do tamanho total de seu estômago. Quando o animal atinge 8 semanas de idade, se o tratamento e os estímulos foram feitos de maneira correta, seu rúmen representará aproximadamente 80% do seu estômago, igualando a proporção de um animal adulto. Nesse momento, o animal está pronto para ser desmamado. A figura 2 mostra a diferença do rúmen de um bezerro recém-nascido em relação a um animal desmamado. (DE SOUZA, 2011)

Figura 2 - Comparação do estômago do bezerro com um adulto



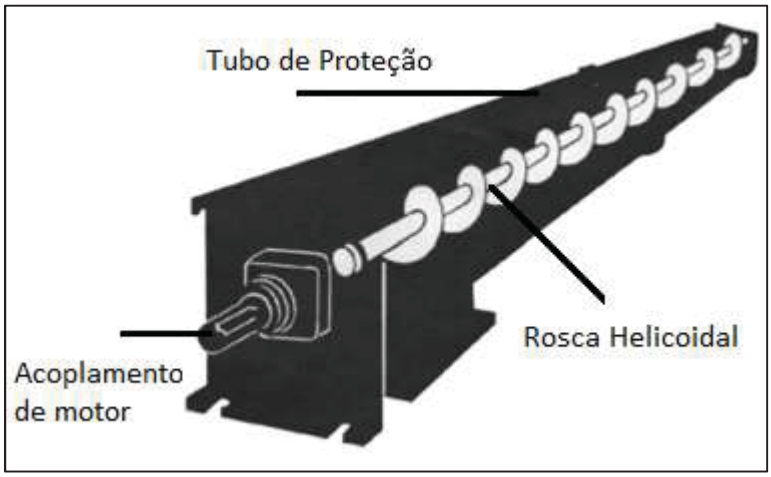
Fonte: Adaptado de DE SOUZA (2011)

2.2 TRANSPORTADOR DE LEITE EM PÓ

O transportador, também chamado de rosca transportadora que está representada na figura 3, é destinado ao transporte de produtos sólidos na horizontal ou de forma inclinada, sendo que o sentido de transporte dependerá do sentido de rotação do dispositivo e do tipo de helicoide, se é direita ou esquerda. O transporte de grãos ou de pó acontece por arrasto, devido ao giro do helicoide. (MILMAN, 2002)

O dispositivo é basicamente composto por um tubo, dentro do qual se localizará a helicoide. Desse modo, o tubo tem a finalidade de encobrir e proteger a rosca, além de fixar suas extremidades aos mancais e rolamentos.

Figura 3 - Rosca transportadora e tubo protetor



Fonte: Adaptado de MILMAN (2002)

"Os helicoides, também chamados de sem-fim, são fitas de chapas de aço que sofrem um processo de perfilação, que deve deixar uma espessura, na borda externa, de no mínimo 4 milímetros." (MILMAN, 2002, p. 107). Como o projeto utilizará o equipamento para transportar leite em pó, quanto menor a folga externa, mais precisamente será feita a dosagem.

2.3 AQUECIMENTO DE ÁGUA

Nessa divisão serão abordados temas relevantes ao aquecimento de água, desde o método de aquecimento, transferência de calor e a forma de medição da temperatura.

2.3.1 Calor

A quantidade de energia transferida para um sistema afim de aquecê-lo é denominado calor. A quantidade de calor para aquecer um sistema é dependente de algumas variáveis, como massa do produto a ser aquecido, natureza do material, diferença de temperatura (o ponto inicial e o final). Para elevar em 1ºC a temperatura de um quilograma de água, é necessário transferir uma quantidade de calor igual a 4190 J (Joules). (YOUNG e FREEDMAN, 2015)

$$Q = m \times c \times \Delta T \tag{1}$$

Q = Calor em Joules

m = Massa do material em quilogramas.

c = Calor Específico em $\frac{\text{J}}{Kg\ .\ K}$

ΔT = Variação de Temperatura em Kelvin (K)

Para calcular o tempo necessário de aquecimento, basta dividir a quantidade de calor pela energia fornecida ao sistema, onde 1 Joule por segundo equivale a 1 Watt. O calor específico da água é aproximadamente igual a:

$$4190 \frac{J}{kg\ K} \tag{2}$$

2.3.2 Resistência de Aquecimento

Condutores são materiais que apresentam um grande número de portadores de carga elétrica, facilitando o movimento das cargas. Quando há uma diferença de potencial em um material condutor, ocorre o movimento de partículas, criando a corrente elétrica. Essas partículas movimentam-se rapidamente, e acabam se chocando com outras partes inertes do condutor, causando assim um efeito de aquecimento, conhecido como Efeito Joule. (BARROS, 2017)

Resistores nada mais são do que condutores que produzem uma certa quantia de energia. Para calcular essa energia deve-se considerar uma série de fatores, como resistência

elétrica do condutor, diferença de potencial aplicada e corrente gerada. A resistividade elétrica de um condutor é dada por:

$$R = p \times \frac{L}{A} \tag{4}$$

Onde:
R = Resistividade Elétrica em Ohm (Ω)
p = Condutividade de um material específico (Ω.m)
L = Comprimento do material em metros (m)
A = Área do material em metros² (m²)

A partir da resistência do material utilizado e da diferença de potencial aplicado, podemos calcular a corrente elétrica resultando utilizando a Lei de Ohm.

$$I = \frac{V}{R} \tag{5}$$

Onde:
I = Corrente Elétrica
V = Diferença de Potencial

A potência de aquecimento, causada pelo efeito Joule, pode ser calculada da seguinte maneira:

$$P = V \times I \tag{5}$$

Onde:
P = Potência Elétrica Fornecida

O reservatório utilizado no projeto tem capacidade de aproximadamente 15 litros de água e a resistência utilizada para aquecer têm aproximadamente 1000W. Desse modo, o tempo de aquecimento pode ser calculado utilizando a equação 1 e dividindo pela potência do equipamento. Foi considerado que a temperatura inicial da água é de 20ºC.

$$Tempo = (m \times c \times \Delta T) \times \frac{1}{P_{watt}} \quad (6)$$

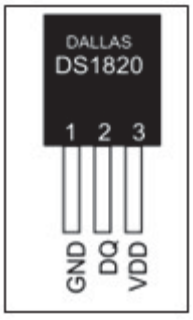
$$Tempo = (15 \times 4190 \times (45 - 20) \times \frac{1}{1000} \cong 1570\ segundos \cong 26\ minutos$$

2.3.3 Sensores de Temperatura

Os primeiros termômetros foram criados no século XVII por Galileu, e desde então foram estudadas técnicas para aprimorar o sistema de medições, seja para aplicação em laboratório, seja para aplicações industriais em linhas de produção. (BALBINOT e BRUSAMARELLO, 2012).

O DS18B20 é um sensor de temperatura digital, o qual retorna o valor da temperatura com o tamanho configurável de 9 a 12 bits, desse modo, não é necessária entrada analógica para fazer a leitura da temperatura. A comunicação é feita com apenas um fio, configuração *1-wire*, e o tempo máximo de conversão, quando utilizado 12 bits é de 750 ms e quando utilizando em 9 bits o tempo fica próximo aos 100 ms.

Figura 4 - Sensor DS18B20



Fonte: Adaptado de DALLAS SEMICONDUCTOR.

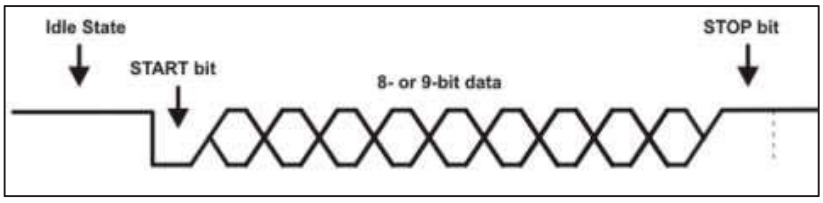
2.4 MICROCONTROLADOR

Os microcontroladores são circuitos integrados que contém um núcleo de processador, memória, periféricos programáveis de entrada/saída, interrupções, temporizadores, contadores, *Pulse Width Modulation* (PWM), conversores analógico/digital e alguns sistemas de comunicações como por exemplo: *Serial Peripheral Interface* (SPI), *Universal Asynchronous Receiver Transmitter* (UART) e *Inter-Integrated Circuit* (I2C). Com isso, forma-se um *hardware* extremamente complexo. (MARTINS, 2005).

2.4.1 Comunicação UART

Como o próprio nome já diz, é uma comunicação que trabalha de forma assíncrona, transmitindo dados utilizando apenas dois fios sendo um deles *transmission* (TX) e o outro *reception* (RX). A figura 6 mostra o funcionamento da comunicação UART.

Figura 5 – Funcionamento da comunicação UART



Fonte: https://paginas.fe.up.pt/~hsm/docencia/comp/uart/ (27/11/18)

A partir da figura 6 nota-se que quando a rede está inativa o pino de saída encontra-se em nível lógico alto e antes do início da transmissão recebe o *Start bit*, em nível lógico baixo. Depois do start recebe os dados, normalmente 8 bits, podendo ser 9 bits de tamanho também, começando pelo LSB. Ao fim da transmissão recebe o sinal de *Stop bit*, em nível lógico alto para sinalizar que o pacote de dado enviado chegou ao fim. (MARTINS, 2005).

Há algumas configurações a serem determinadas antes da comunicação ser utilizada, que são:

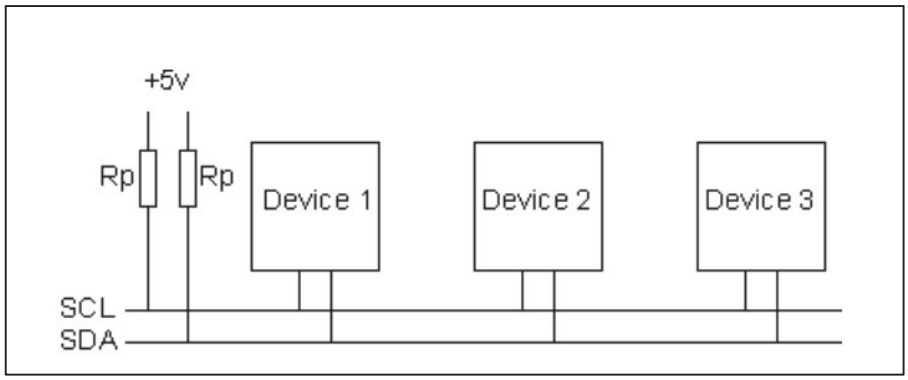
- Velocidade da transmissão (1200, 2400, 4800, 9600, 19200, 38400, 57600 e 115200);
- Número de bits: 8 ou 9;
- Paridade: Par ou Ímpar;
- Stop bits: 1 ou 2;

2.4.2 Comunicação I2C

Comunicação utilizada entre periféricos e um mestre, protocolo desenvolvido pela Philips, possibilita que até 127 dispositivos diferentes estejam ligados simultaneamente. A transferência de dados, diferentemente da UART, é feita de maneira síncrona e bidirecional, a única semelhança é o meio físico, o qual utiliza também 2 fios que são *Serial Data Line* (SDA) e *Serial Clock Line* (SCL).

Como todos os escravos partilham a mesma linha de transmissão eles devem ter endereços únicos. Para o início da comunicação, o mestre envia o endereço do dispositivo e entra em modo de recepção, aguardando o *Acknowledge* (ACK), quando o escravo com endereço solicitado retorna, começa a transferência de dados até o mestre enviar o *Stop bit*. A figura 7 mostra como é feito um barramento I2C, onde todos os dispositivos são ligados em paralelo e também deve ser utilizado resistor de *pull up* de 10kΩ em cada um dos barramentos.

Figura 6 – Diagrama de dispositivos I2C



Fonte: https://robot-electronics.co.uk/i2c-tutorial (27/11/18)

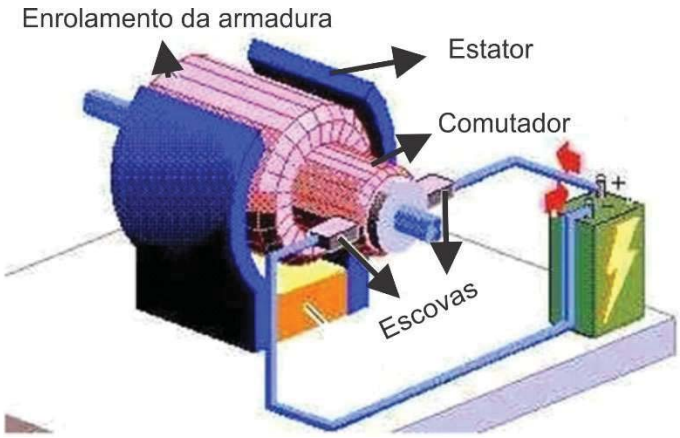
2.5 MOTORES

Motores elétricos são máquinas girantes destinadas a converter a energia elétrica em energia cinética. São muito utilizados na indústria por serem mais econômicos, menores e melhores de serem controlados. Existem muitos tipos de motores, como por exemplo: motores síncronos, motores assíncronos, motores de passo, motores a corrente contínua etc.

2.5.1 Motor a CC de imãs permanentes com escovas

Os elementos que caracterizam esse modelo de motor são o estador, o rotor, o comutador, o eixo e as escovas. O estator é o responsável por gerar o campo magnético, o qual consiste em utilizar imãs para manter o campo constante. O rotor é responsável por aplicar o torque para a carga. O comutador, onde são alojadas as pontas das bobinas, é onde passa a corrente elétrica, que será comutada pelas escovas. A figura 7 representa o motor CC. (TORO, 1994).

Figura 7 – Motor a CC de imãs permanentes com escovas



Fonte: Adaptado de http://www.fisicanet.com.ar/fisica/electrotecnia/ap2/motores04.jpg (22/10/2017)

2.5.2 Motor de indução monofásico com capacitor permanente

O que caracteriza o motor de indução é que só o estator é ligado à rede de alimentação, já o rotor não tem alimentação externa e suas correntes são induzidas pelo estator. Esse tipo de motor por possuir apenas uma fase de alimentação não tem campo magnético girante e sim um campo magnético pulsante. Isso faz com que o conjugado de partida seja nulo e por isso é necessário enrolamentos auxiliares para criar uma segunda fase fictícia.

Nesse tipo de motor o enrolamento auxiliar e o capacitor ficam permanentemente energizados, e o efeito do capacitor é criar condições de fluxo semelhantes aos motores trifásicos. Normalmente esse tipo de motor é utilizado em ventiladores, exaustores, bombas centrífugas e são de potência baixa. (TORO, 1994).

2.6 RFID

A identificação por radiofrequência (*radio frequêncy identification*) é uma técnica que usa ondas de rádio para captura ou gravação de dados em etiquetas (*tags)*, assim não sendo necessário qualquer tipo de contato físico para transmissão de dados. A ideia inicial sobre esse sistema foi criada pelo físico escocês Sir Robert Alexander Watson-Watt, em 1935, para identificar aeronaves inimigas ou amigas, o projeto foi nomeado IFF (*Identify Friend or Foe*). (GONSALES, 2017).

Um sistema RFID é composto, basicamente, por três componentes: antena, transceptor e transponder (etiqueta). A antena manda os sinais de rádio para ativar o *tag*, ler e escrever dados, ela é o canal entre o aparelho a ser lido e o transceptor, o qual controla a aquisição de dados e comunicação do sistema. O transponder é o objeto que irá armazenar os dados, há vários modelos deles, porém, o modelo a ser utilizado será passivo, ou seja, não terá bateria e só será ativado pelo campo magnético emitido pelo leitor. (RFID SYSTEMS, 2017)

Há diferentes frequências de operações para sistemas assim, quanto maior for a frequência maior é a quantidade de dados que pode ser transmitida e maior a distância de transmissão, porém, o custo também é maior. Para sistemas de identificação animal é comum utilizar sistemas de baixa frequência, normalmente 125 kHz, pois não há necessidade de grandes alcances e o custo é baixo. Atualmente, os formatos mais comuns de etiquetas para bovinos são em forma de brinco ou colar, conforme mostra a figura 8.

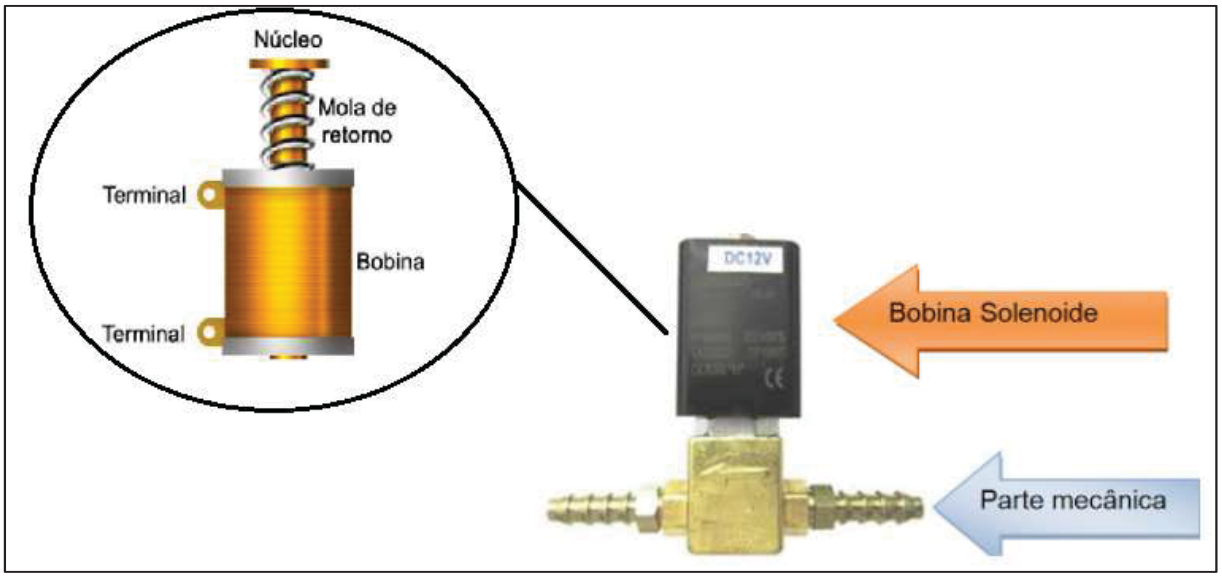
Figura 8 - Brinco RFID para bovinos

Fonte: http://www.agriexpo.online/pt/prod/destron-fearing/product-173364-28358.html

2.7 ELETROVÁLVULAS

Eletroválvulas são dispositivos que do ponto de vista mecânica é destinado a abrir e fechar um fluxo, seja ele pneumático ou hidráulico, e seu funcionamento é idêntico aos dispositivos elétricos conhecidos como relés. A parte mecânica desse dispositivo atua da mesma maneira que um registro, abre e fechada um canal, e a parte elétrica é uma bobina solenoide. (BRAZ, 2013).

Figura 9 – Eletroválvula e bobina solenoide.



Fonte: Adaptado a partir dos autores BRAZ (2013) e BRAGA (2014)

O funcionamento da bobina solenoide é simples, o eletroímã é criado pelo campo magnético gerado pela circulação de corrente nos fios da bobina, como mostra a Figura 9. Esse campo atrai o núcleo para o centro, o qual fica retido enquanto há circulação de corrente. Assim que a corrente cessa, a mola é responsável por retornar o núcleo ao seu local de repouso, fechando o circuito mecânico caso seja NF (Normalmente Fechado). (BRAGA, 2014)

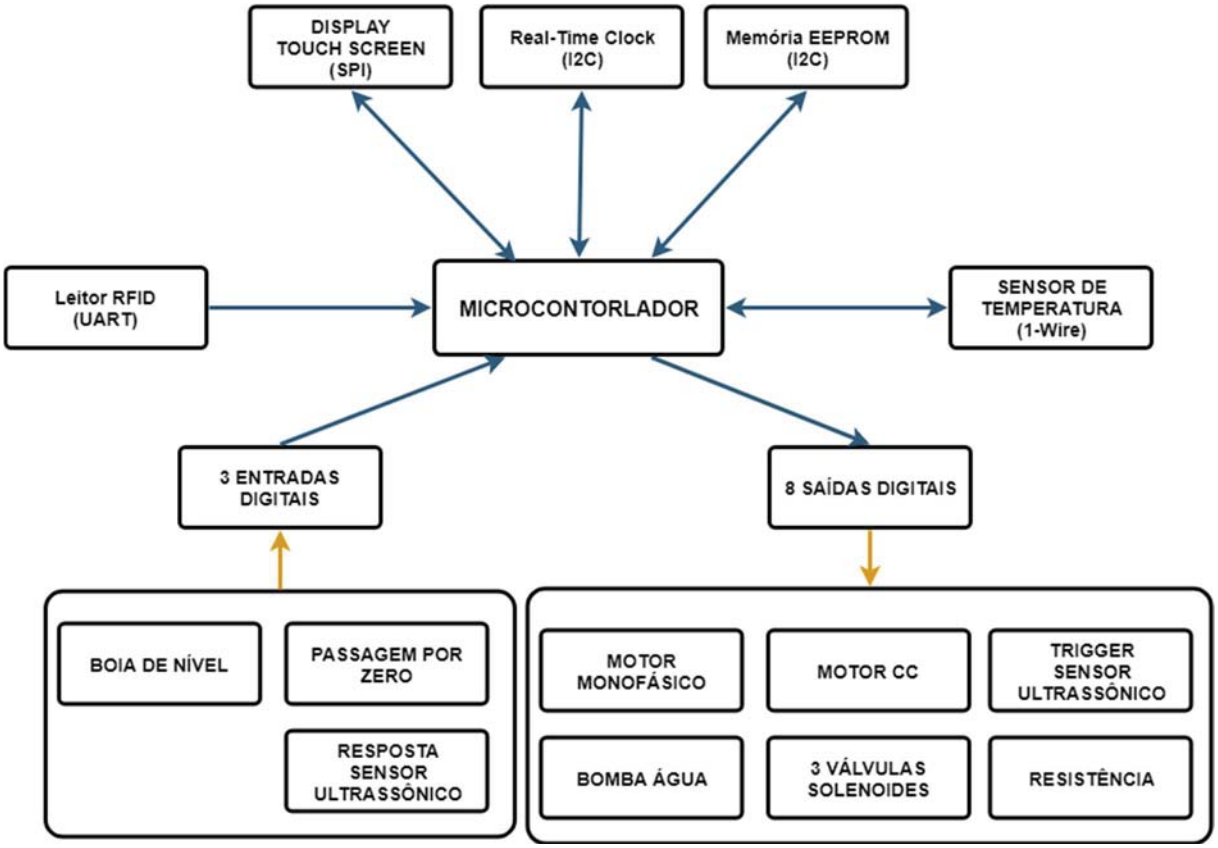
3 DESENVOLVIMENTO DO PROJETO

Neste capítulo são apresentadas as especificações, características relacionadas a estrutura mecânica do projeto, do *hardware* e *firmware*. Todas essas características levam em consideração os objetivos e necessidades do protótipo.

3.1 ESPECIFICAÇÕES DO PROJETO

O diagrama de blocos da figura 10 representa todos os periféricos utilizados no projeto, servindo de referencial para as especificações do *hardware.* O núcleo do projeto é o microcontrolador, que deve ser capaz de ler todos os dispositivos usados, envolvendo entradas digitais, saídas digitais, comunicação 1-wire, I2C, UART e SPI. A figura 10 representa o diagrama geral do projeto.

Figura 10 – Diagrama geral do projeto



Fonte: Próprio autor

As funções dos periféricos representados na figura 10 são:

- *Display touch screen:* Fazer a interface com o usuário, desde cadastro e exclusão de animais do sistema, verificar a idade e a quantidade de alimentos já preparados individualmente;
- RTC: Para fazer verificação de idade do animal e intervalo de horário entre as refeições;
- Memória EEPROM: Gravar os dados de alimentações;
- Leitor RFID: Identificar *tags* na frequência de 125kHz;
- Sensor de temperatura: informar para o microcontrolador a temperatura atual da água no reservatório;
- Saídas digitais:
    - Acionamento do misturador;
    - Acionamento da rosca transportadora;
    - Acionamento da bomba d'água;
    - Acionamento de válvulas solenoide (3x);
    - Disparo do sensor ultrassônico;
- Entradas digitais:
    - Boia de nível;
    - Passagem por zero;
    - Resposta sensor ultrassônico;

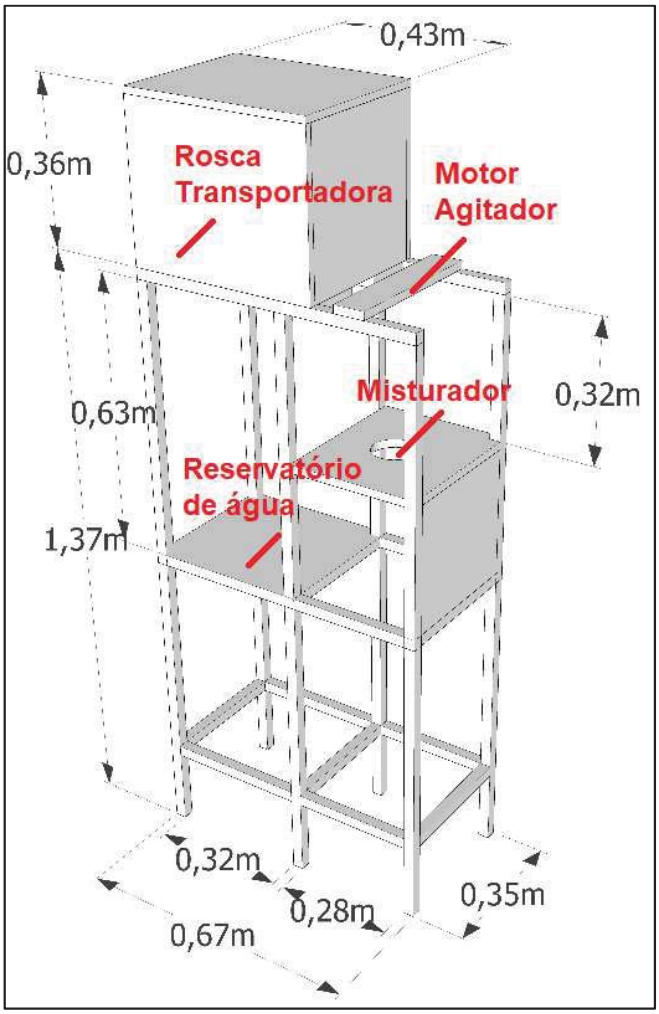
3.2 ESTRUTURA MECÂNICA DO PROTÓTIPO

A estrutura mecânica não foi o enfoque do projeto, por esse motivo foram utilizados os materiais mais baratos encontrados no mercado que atendiam a funcionalidade do processo.

3.2.1 Estrutura de sustentação

A estrutura desenvolvida foi um protótipo para testes das funcionalidades do equipamento, ou seja, não é preparada para testes em campo devido a fragilidade da mesma, a qual foi projetada com o objetivo de reduzir os gastos com materiais. Desse modo, foi utilizado madeira para a confecção.

Figura 11 – Projeto estrutural do protótipo

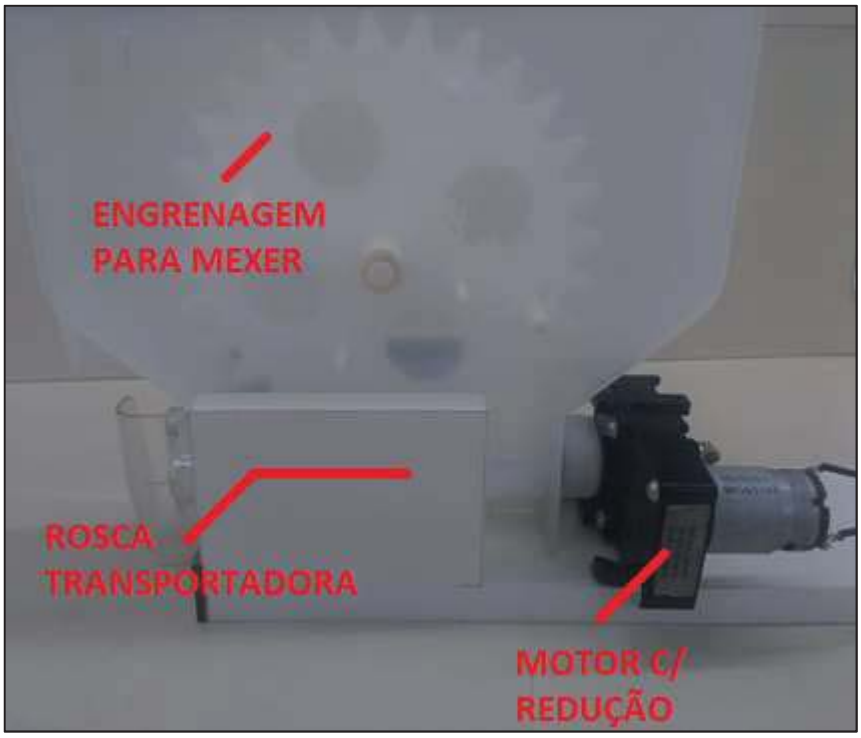


Fonte: Próprio autor

3.2.2 Reservatório de leite em pó e seu transporte

O reservatório de leite em pó foi reaproveitado de um recipiente utilizado nas máquinas de café expresso. Em consequência disto já veio com a rosca, engrenagem para evitar que o produto se acumule nas bordas e o motor já com eixo de redução e acoplamento. Foram feitos vários testes com a rosca helicoidal, e a dosagem dela é muito próxima a 2 gramas de produto por segundo, portanto foi utilizado temporizadores para fazer a dosagem do leite em pó. O modelo utilizado está representado na figura 12.

Figura 12 – Transportador horizontal



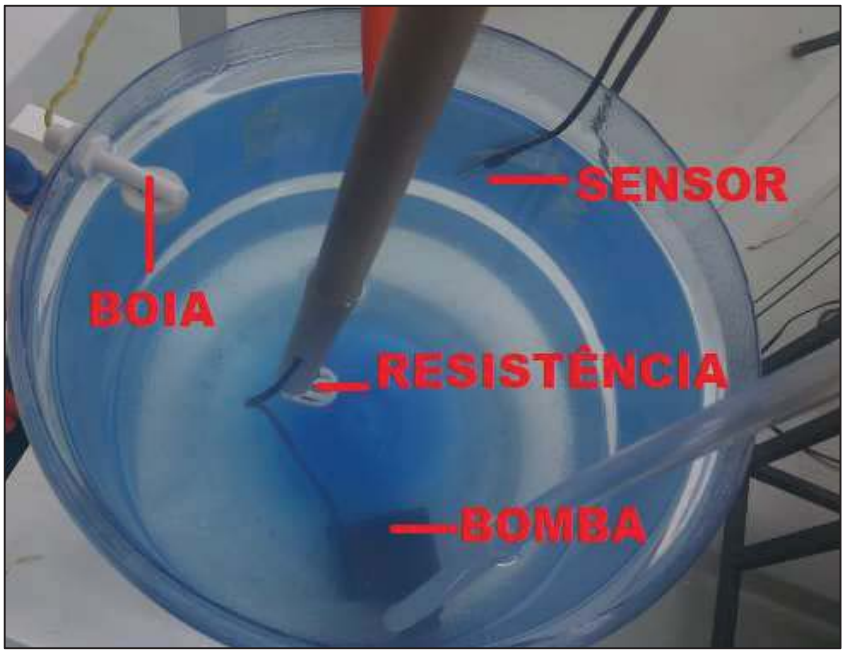
Fonte: Próprio Autor

3.2.3 Reservatório de água

Para fazer o reservatório de água foi utilizado um galão de 20 litros, pois fazer um especial de acordo com a necessidade influenciaria muito no custo do protótipo, desse modo foi utilizado uma estrutura barata e de fácil acesso.

Nesse reservatório foi fixada uma boia de nível (sensor com resposta digital *on/off*), uma bomba de água, uma resistência e um sensor de. O reservatório está representado na figura 13, bem como os equipamentos utilizados nele.

Figura 13 – Reservatório de água

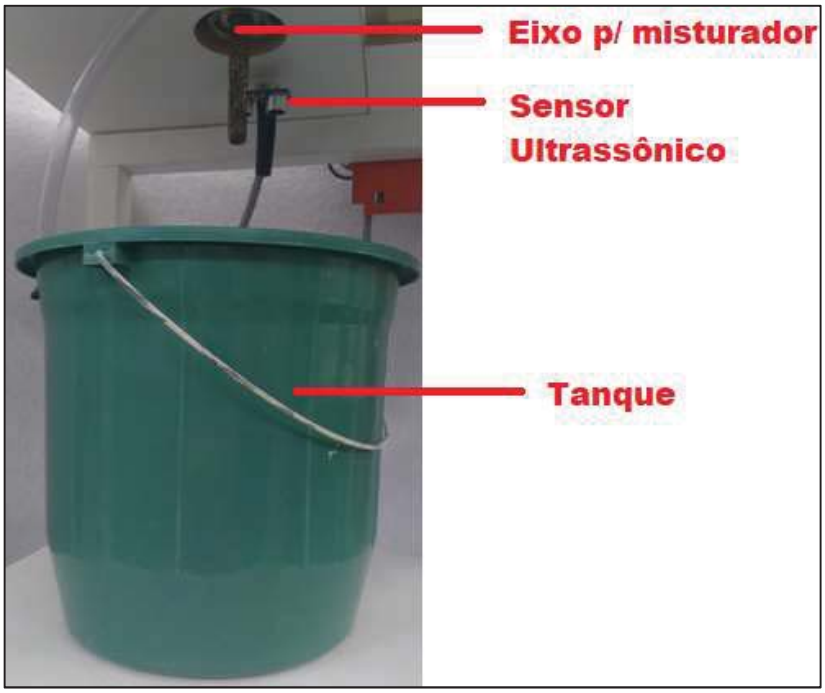


Fonte: Próprio Autor

3.2.4 Misturador

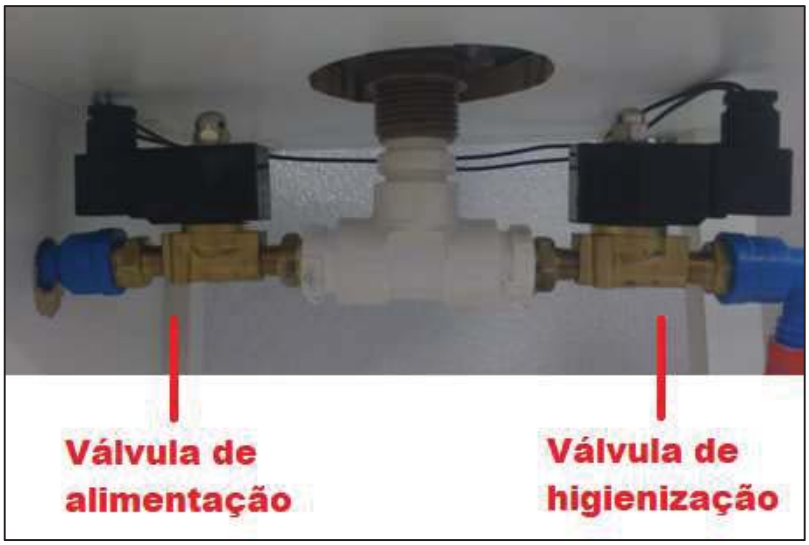
O ambiente do misturador foi utilizado um balde, também devido ao custo de confeccionar um de acordo com a necessidade. Para fazer a furação e vedação foi utilizado um adaptador e flange com rosca, além de silicone para resolver o problema de vedação e alteração no volume do balde. O motor utilizado para agitador foi um motor monofásico com capacitor permanente utilizado normalmente em ventiladores, pois não há necessidade de potência para fazer a mistura. As figuras 14 e 15 representam o misturador e os equipamentos integrados a ele.

Figura 14 – Tanque de mistura



Fonte: Próprio Autor

Figura 15 – Válvulas de saída de produto



Fonte: Próprio Autor

3.2.5 Válvulas

Foram utilizados dois modelos de válvulas solenoides no projeto com o objetivo de tornar o protótipo mais econômico. Para a entrada de água no reservatório de água foi utilizado uma válvula solenoide simples, normalmente utilizada em máquinas de lavar roupa,

porém para saída de produto do misturador essa válvula não é eficaz. Isso acontece pois a bobina solenoide não atua diretamente no êmbolo de vedação de água, apenas atua como se fosse uma tranca. Quando energizamos a bobina, o pino magnético é atraído, porém para abrir o êmbolo é necessário pressão de água de no mínimo 0,2 bar.

Desse modo, o projeto necessitava de válvulas mais refinadas para a saída de produto do misturador, válvulas que não necessitassem de pressão, por isso foi utilizado uma válvula de latão normalmente usada em tubulações de gás. A figura 16 representa a válvula solenoide simples e a figura 17 a válvula de gás.

Figura 16 – Válvula solenoide simples

Fonte: Adaptado de https://www.msseletronica.com/detalhes/valvula-solenoide-12v-1-2-x-1-2-180%C2%B0-para-agua-ideal-para-automacao-va-03-/994.html (30/11/2018)

Figura 17 – Válvula solenoide para gás

Fonte: Adaptado de https://www.direcionalbauru.com.br/valvula-solenoide-gas (30/11/2018)

3.3 HARDWARE

A partir das especificações e dos objetivos que o projeto demanda, foram projetados e desenvolvidos componentes os quais deveriam suprir as necessidades. Porém, como o projeto trata-se de um protótipo, as escolhas foram retidas ao custo dos materiais.

3.3.1 Medição de temperatura do reservatório

Para fazer esta medição, foi utilizado o sensor de temperatura submerso DS18B20, como mostra a figura 18, esse sensor tem os seguintes dados:

- Tensão de alimentação: 3,0 a 5,5VDC.
- Temperaturas: -55ºC a 125ºC.
- Precisão: ±0.5ºC de -10ºC a 85ºC.
- Resposta programável: 9 a 12 bits.
- Tempo máximo de reposta: 750ms (12 bits) a 93,75ms (9bits)
- Sensor digital: comunicação 1-wire.
- Resistor de *pull up*: 4,7kΩ.
- Grau de proteção do encapsulamento: IP68.

Figura 18 - Sensor DS18B20 encapsulado

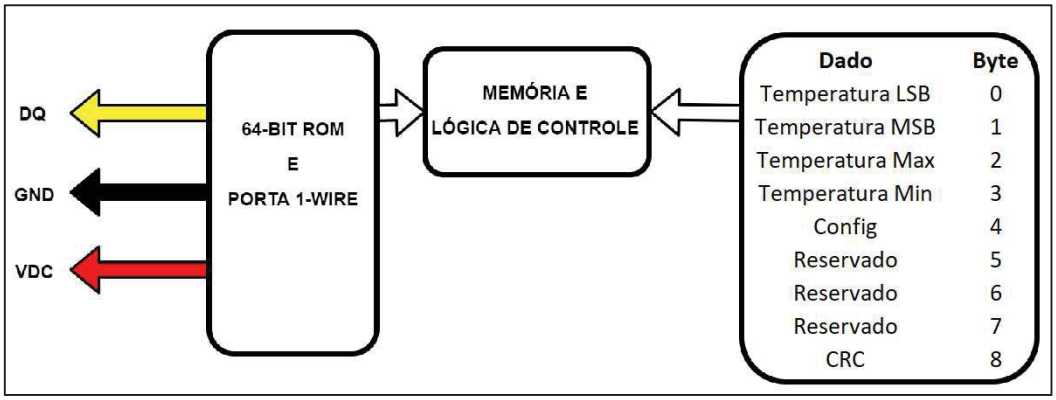
Fonte: FILIPEFLOP.

O diagrama do sensor é organizado como mostra a figura 19. O sensor possui uma memória *Read Only Memory* (ROM) onde é armazenado os seguintes dados: Código CRC (8bits), Número de série (48bits) e Código da Família (8bits), o código CRC é o byte mais

significativo da ROM e é formado pelos outros 56 bits, sua função é apenas para validação da ROM.

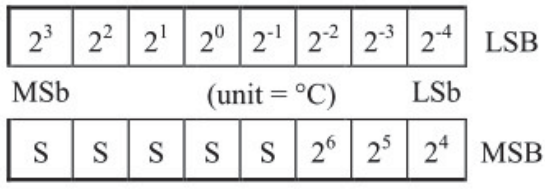
Os dados de transferência entre o sensor e o mestre do barramento são acessados através de endereços e funções pré-definidas no próprio equipamento. Os dados de temperatura são organizados em 2 *bytes,* já a temperatura máxima e mínima apenas 8, desse modo é perdido precisão nesses dados. A figura 20 possui exemplos de leituras de temperaturas diferentes e organização dos bits.

Figura 19 – Diagrama de funcionamento do sensor DS18B20



Fonte: Adaptado de http://pdf1.alldatasheet.com/datasheet-pdf/view/58557/DALLAS/DS18B20.html.

Figura 20 – Diagrama de organização dos bits de temperatura



TEMPERATURE	DIGITAL OUTPUT (Binary)	DIGITAL OUTPUT (Hex)
+125°C	0000 0111 1101 0000	07D0h
+85°C	0000 0101 0101 0000	0550h*
+25.0625°C	0000 0001 1001 0001	0191h
+10.125°C	0000 0000 1010 0010	00A2h
+0.5°C	0000 0000 0000 1000	0008h
0°C	0000 0000 0000 0000	0000h
-0.5°C	1111 1111 1111 1000	FFF8h
-10.125°C	1111 1111 0101 1110	FF5Eh
-25.0625°C	1111 1110 0110 1111	FF6Fh
-55°C	1111 1100 1001 0000	FC90h

Fonte: Adaptado de http://pdf1.alldatasheet.com/datasheet-pdf/view/58557/DALLAS/DS18B20.html

3.3.2 Medição de nível do misturador

Para aquisição do nível de água dentro do misturador foi utilizado o sensor ultrassônico HC-SR04, representado na figura 21. O princípio de funcionamento do sensor é a medição do tempo entre o envio e o retorno de uma onda sonora, ou seja, como a velocidade do som é algo conhecido, basta multiplicar o tempo de resposta pela velocidade do som e encontrar a distância do objeto que refletiu tal onda.

$$Distância = \frac{Tempo \times Velocidade_{som}}{2} \quad (7)$$

Figura 21 – Sensor ultrassônico HC-SR04

Fonte: https://www.filipeflop.com/produto/sensor-de-distancia-ultrassonico-hc-sr04/(19/06/2018)

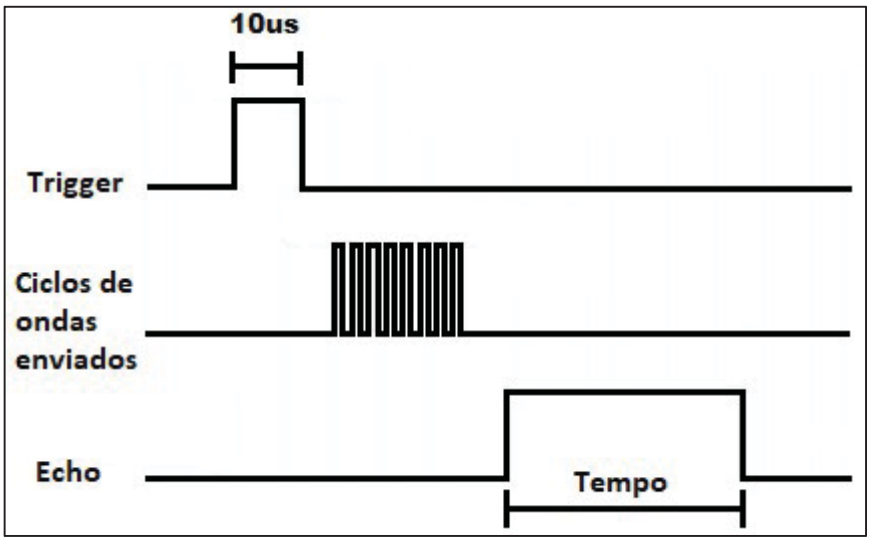
As características elétricas desse sensor são:

- Alimentação: 5 VDC;
- Corrente quiescente: 2mA;
- Corrente de trabalho: 15mA;
- Faixa de medição: 2cm a 400cm;
- Precisão: 0,3cm;
- Ângulo de medição: 15 graus;
- Largura de pulso para trigger: 10us;

O seu funcionamento acontece da seguinte maneira: para iniciar a medição, é necessário um pulso de disparo no pino identificado como "*Trig*", no formato *Transistor-Transistor Logic* (TTL) com largura de pulso de no mínimo 10us, informado pelo datasheet do componente. Logo após o sinal de *trigger* cair para nível lógico baixo, o componente

dispara 8 ciclos de ondas ultrassônicas na frequência de 40kHz e ao terminar essa sequência de disparo o pino identificado como "*Echo*" é colocado em nível lógico alto e quando o componente recebe as ondas refletidas o pino novamente é colocado em nível lógico baixo. O tempo em que o "*Echo*" fica em nível lógico alto, é o tempo que as ondas levaram para encontrar um objeto e retornar, desse modo é necessário apenas calcular a distância em função do tempo que o pino digital ficou em nível lógico alto. Os sinais do sensor estão representados na figura 22.

Figura 22 – Sinais do sensor HC-SR04



Fonte: Adaptado de https://www.datasheetspdf.com/pdf/909919/ELECFreaks/HC-SR04/1.

Com a medição de distância do sensor é possível identificar o nível do misturador de forma dinâmica, ou seja, verificar a quantidade de líquido de maneira a determinar a porcentagem de utilização, assim é viável monitorar qual o volume de água interno.

Este sensor não possui proteção contra umidade e nem contra respingos de água, o que o torna incapaz de ser utilizado em um produto real, porém, o objetivo é o teste de funcionalidade do protótipo.

3.3.3 Medição de nível do reservatório de água

Para fazer o controle do nível da água do reservatório e garantir que não há possiblidade de ligar o sistema de aquecimento com ele vazio, foi utilizado um sensor boia de nível lateral. O sensor funciona como uma chave magnética em formato de boia, pois entra

em curto quando o objeto emerge, desse modo o controlador identificará quando o limite de água for atingido. As características elétricas do sensor, representado na figura 23, são:

- Tensão de chaveamento máxima: 100VDC;
- Corrente de chaveamento máxima: 0,5A;
- Temperatura de operação: -10ºC a 85ºC;

Figura 23 – Sensor de nível, boia lateral.

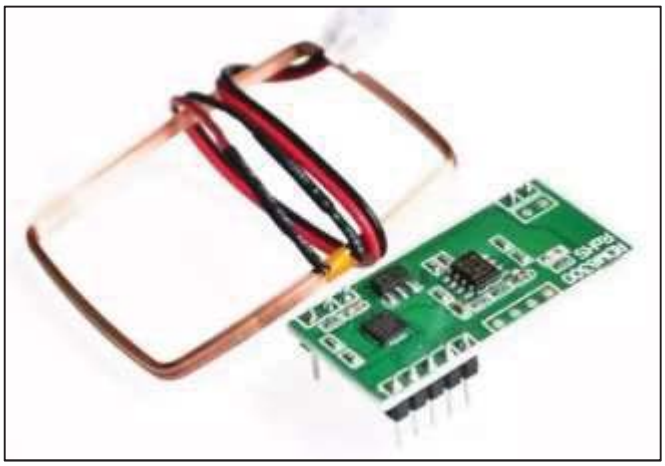
Fonte: Adaptado de https://www.soldafria.com.br/sensor-de-nivel-de-agua-lateral-para-arduino

3.3.4 Leitor de cartão RFID

Com o objetivo de identificar cada animal, foi utilizado um leitor de RFID para 125kHz, modelo RDM630, o qual está representado pela figura 24. Esse leitor tem comunicação TTL RS232 - UART. As características elétricas são:

- Alimentação: 5VDC;
- Protocolo de comunicação: TTL RS232 UART;
- Corrente máxima: 50mA;
- *Baud Rate:* 9600bps;
- Dado: 8 bits (byte a byte, caractere);
- Paridade: sem paridade;
- Stop bit: 1 bit;
- Frequência de leitura: 125kHz;
- Distância máxima de leitura: 5cm;

Figura 24 – Leitor RFID RDM630



Fonte: Adaptado de http://www.rfidhy.com/hy-rdm630-125khz-134-2khz-reader-module-access-control-rfid-reader-writer-module/.

3.3.5 Fonte de alimentação 24VDC

Para fazer a conversão CA/CC do projeto foi utilizado uma fonte chaveada pronta, mostrada na figura 25. Características elétricas são:

- Tensão de alimentação: 110 a 240VCA;
- Tensão de saída: 24VDC;
- Corrente máxima de saída: 5A (120W);
- Proteções: Aquecimento, curto-circuito e sobrecarga;

Figura 25 – Fonte de alimentação chaveada 24VDC

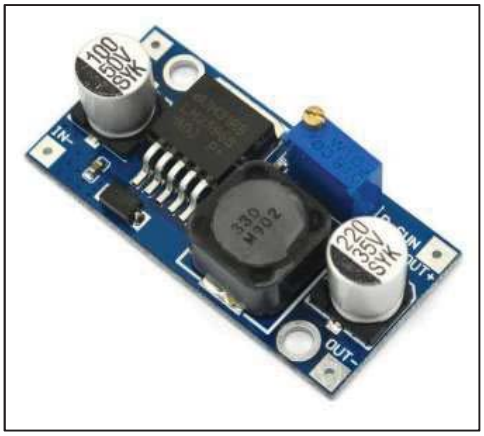
Fonte: Próprio autor

3.3.6 Conversor *buck* CC/CC

Para fazer a alimentação dos sensores, microcontrolador e do *display* foi utilizando um conversor CC/CC *step down* para reduzir a tensão de 24VDC para 5VDC. O componente foi o LM2596, um regulador de tensão com saída ajustável, representado na figura 26. Suas características elétricas são:

- Tensão de entrada máxima: 45VDC;
- Tensões de saída fixa: 3V3, 5V, 12V e 15V;
- Tensão de saída ajustável: 1,23 a 37VDC (inferior a tensão de entrada);
- Corrente máxima de saída: 3A;
- Temperatura de operação: -40ºC a 85ºC;

Figura 26 – Placa de alimentação LM2596



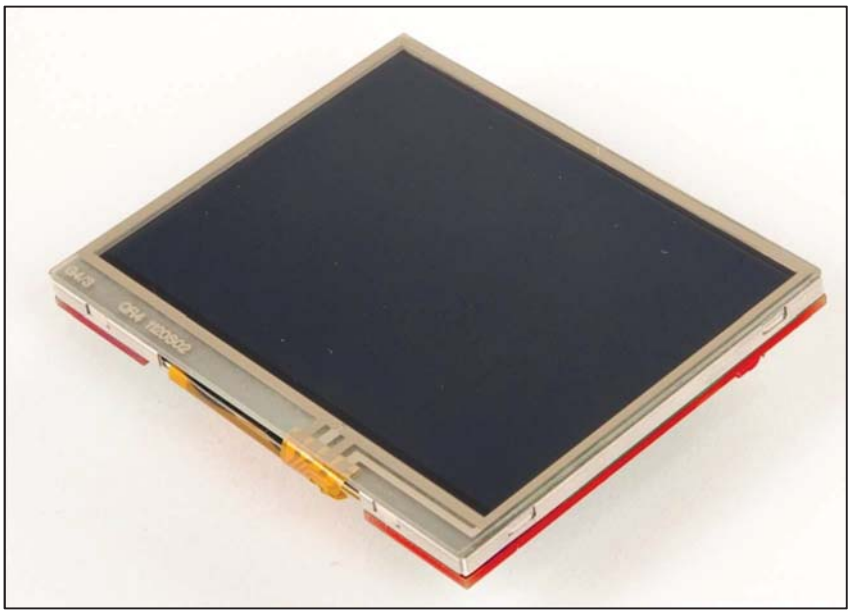
Fonte: http://www.baudaeletronica.com.br/modulo-regulador-de-tensao-lm2596.html (27/11/2018)

3.3.7 Display Touch Screen

Para a interface do usuário foi utilizado o *display* BOOSTXL-K350QVG-S1, produzido pela *Texas Instruments* (TI), pois além de ser um equipamento de baixo custo é de fácil acoplamento com a *launchpad* utilizada. As principais características do display, representado pela figura 27, são:

- 3,5 polegadas com resolução 320x240 pixels;
- Circuito de driver de luz de fundo LED;
- 262000 cores;
- *Touch* resistivo de 4 fios;
- Comunicação padrão SPI;

Figura 27 – Display touch screen



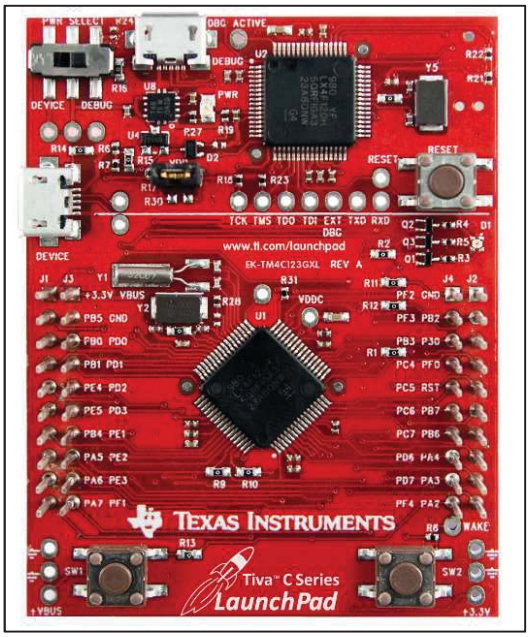
Fonte: http://www.ti.com/tool/boostxl-k350qvg-s1 (27/11/2018)

3.3.8 Microcontrolador

Depois de definir todos os equipamentos necessários para o protótipo, conforme explicado no capítulo 3.1 e representado na figura 10, chegou-se a conclusão que o melhor microcontrolador a ser utilizado é a placa de desenvolvimento MCU TM4C123G *LaunchPad* ™, produzido pela TI. Isso facilitou muito, pois como o display e a *launchpad* têm o mesmo fabricante, eles foram integrados facilmente um ao outro. Também possui todos os módulos de comunicação necessários para o projeto, I2C, UART e SPI. A placa é mostrada na figura 28, e possui as seguintes características:

- Processador ARM Cortex-M4F;
- Frequência máxima de *clock 80MHz;*
- Memórias: *Flash* 256 kB, SRAM: 32 kB, EEPROM: 2 kB;
- Comunicações: UART, I2C, USB e SPI;
- Tensão de alimentação: 5V;
- Corrente máxima por pino de saída: 25mA;
- Máxima tensão nos pinos de entradas: 5,5V;
- Temperatura de operação: -65ºC a 150ºC;

Figura 28 – LaunchPad TM4C123GXL da *Texas Instruments*

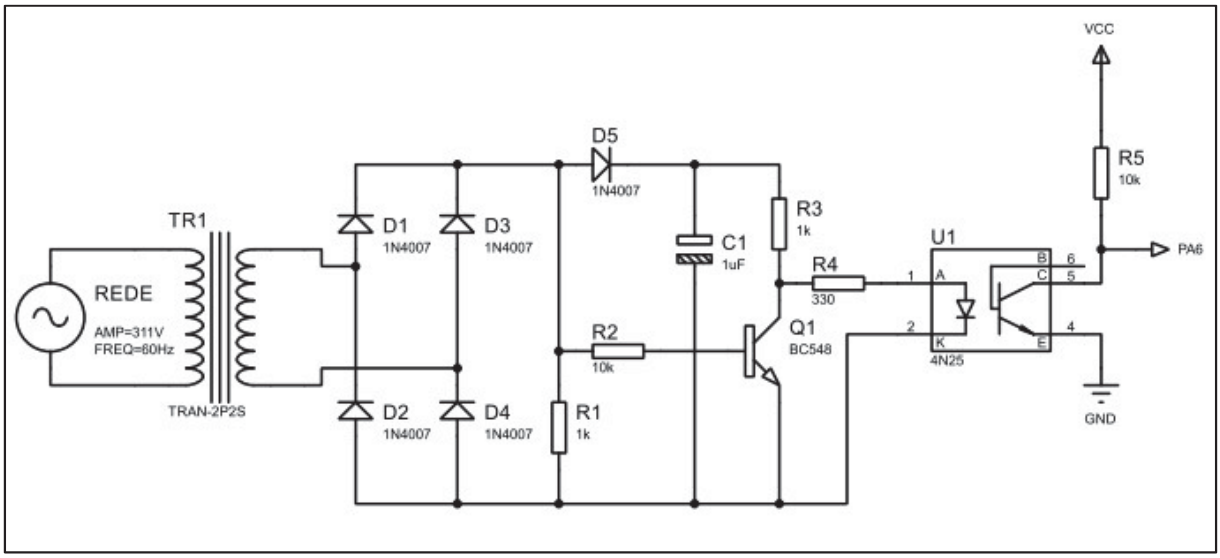


Fonte: http://www.ti.com/tool/ek-tm4c123gxl (27/11/2018)

3.3.9 Placa 1 – Resistência de aquecimento

Para fazer o aquecimento da água, pensou-se em fazer uma placa separada para suprir as necessidades do circuito, onde a ideia inicial era fazer um controle de fase para modular a tensão RMS em cima da resistência, desse modo alterando a potência. Porém, a ideia foi substituída por um controle *ON/OFF*, pois além de ser mais eficiente não causa distorção harmônica na rede. Para isso foi feito um circuito detector de passagem por zero para fazer o sincronismo do acionamento com a rede elétrica, o qual está representado na figura 29.

Figura 29 – Circuito de detecção de passagem por zero



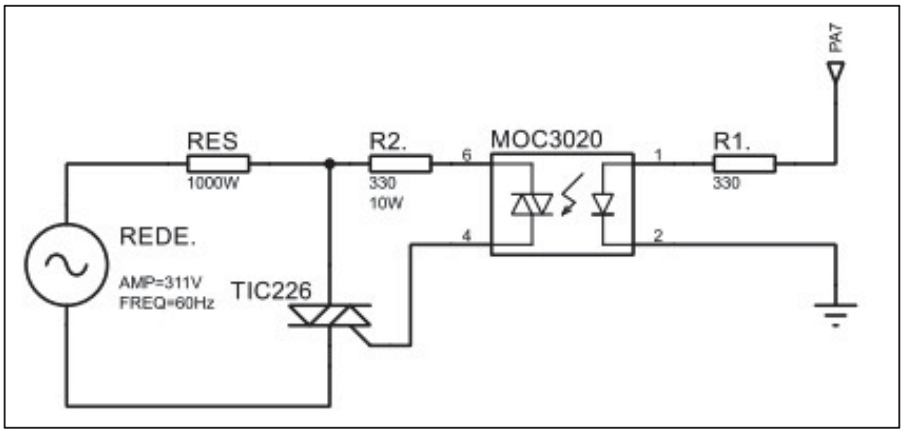
Fonte: Próprio autor

O circuito tem a função de enviar um sinal para o microcontrolador quando a tensão passa pelo 0. O funcionamento do circuito é simples, porém eficaz.

Quando a tensão na base do transistor for superior a 0,7V (tensão VBE de polarização do transistor) a tensão em cima do resistor R4 tende a ser igual ao GND. Quando a tensão for inferior ao VBE de 0,7V, o transistor não está polarizado e nesse momento a tensão em cima do R4 é suficiente para fazer com que o acoplador 4N25 fique ativo e faça com que a entrada PA6 fique em nível lógico baixo. O circuito funciona de maneira inversora, quando a rede está na passagem por zero a entrada do microcontrolador fica em nível lógico 0.

A figura 30 mostra a placa do acionamento da resistência, a qual está utilizando um MOC3020 para fazer o acoplamento e proteção da saída do microcontrolador. Esse circuito ele é capaz de fazer o controle de fase e o controle do tipo *ON/OFF*. O TIC226 foi escolhido devido a sua capacidade de corrente, 7A RMS e 400V de isolação, a sua corrente máxima de Igt é 50mA.

Figura 30 – Acionamento do TIC226 da resistência



Fonte: Próprio autor

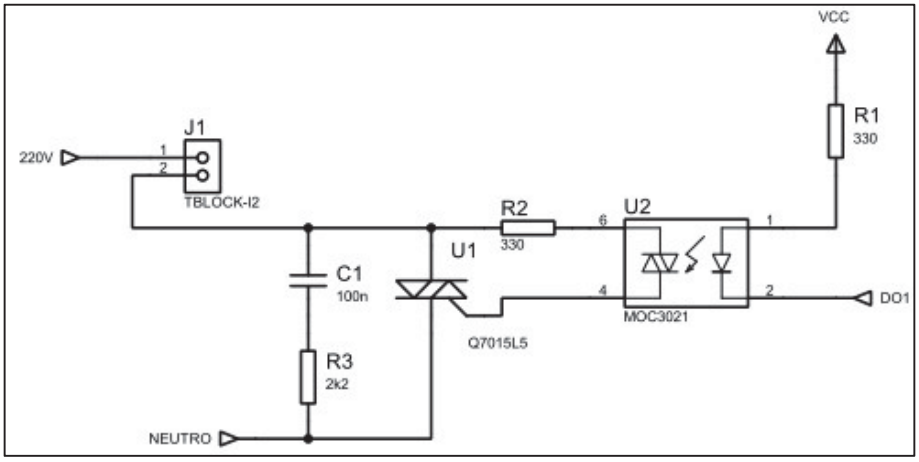
3.3.10 Placa de acionamentos

Há necessidade de 6 acionamentos de potência, são eles:

1. Motor de indução monofásico com capacitor permanente 220VAC do agitador;
2. Motor da rosca helicoidal 24VDC;
3. Motor 220VAC para bomba submersa;
4. Válvula solenoide simples (3x);

Para realizar esses acionamentos foram projetados circuitos opto acoplados, buscando evitar qualquer problema eletromagnético ou indução. Há 5 circuitos para acionamento de cargas em 220VAC e estão representados pela figura 31.

Figura 31 – Acionamento de cargas 220VAC



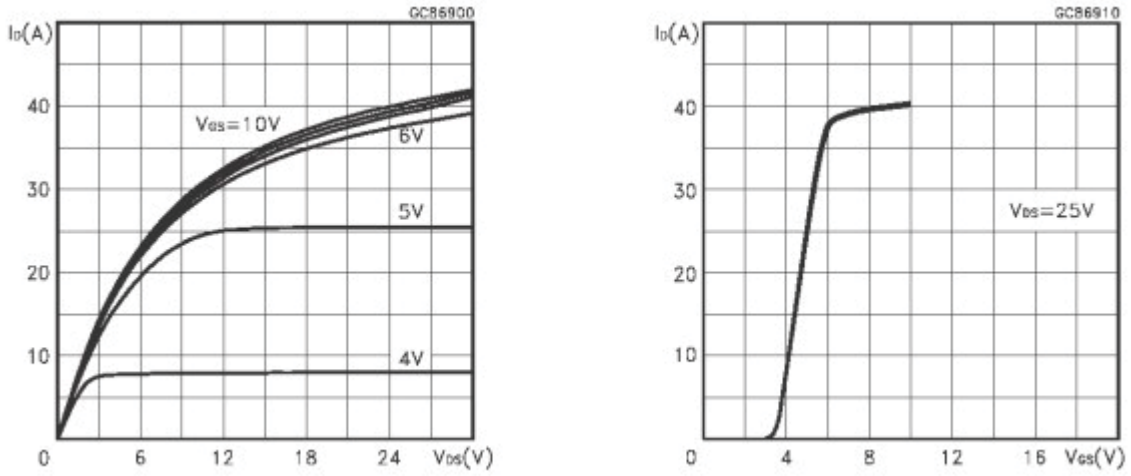
Fonte: Próprio autor

Para a potência foi utilizado o TIC226 descrito no capítulo 3.3.9. O opto acoplador está com o chaveamento no *ground* para garantir o VBE do transistor na placa principal, que também está opto acoplado.

A fim de evitar problemas nos acionamentos das válvulas solenoides e dos motores monofásicos foram utilizados *snubbers*. Esses circuitos têm a função de suprimir os picos de tensões gerados pela comutação dos triacs.

Para o motor do transportador horizontal, que trabalha na tensão de 24VDC, o acionamento utilizado foi projetado com o MOSFET IRF640 e tensão de VGS igual a 5V. A corrente elétrica do motor em questão é de 1,5A com carga e 1,2A a vazio. A figura 32 representa as curvas do dispositivo analisando a sua capacidade de corrente e tensão de *gate*.

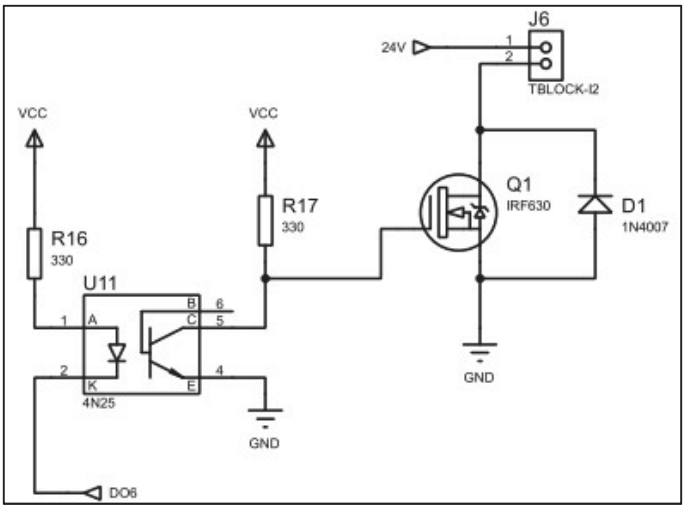
Figura 32 – Curvas do IRF640



Fonte: http://www.alldatasheet.com/datasheet-pdf/pdf/22397/STMICROELECTRONICS/IRF640.html

A partir da análise dos gráficos, pode-se concluir que 5V de VGS é suficiente para acionar o motor. O circuito completo de acionamento está na figura 33.

Figura 33 – Circuito de acionamento do MOSFET



Fonte: Próprio autor

3.3.11 Placa Principal

Com todo o hardware já definido, faltou apenas a placa principal onde o microcontrolador será instalado. Ela possui todas as entradas para os sensores e os demais dispositivos do projeto.

O quadro 4 mostra todos os pinos utilizados do microcontrolador e quais as funções empregadas a eles.

Quadro 4 – Configuração do microcontrolador

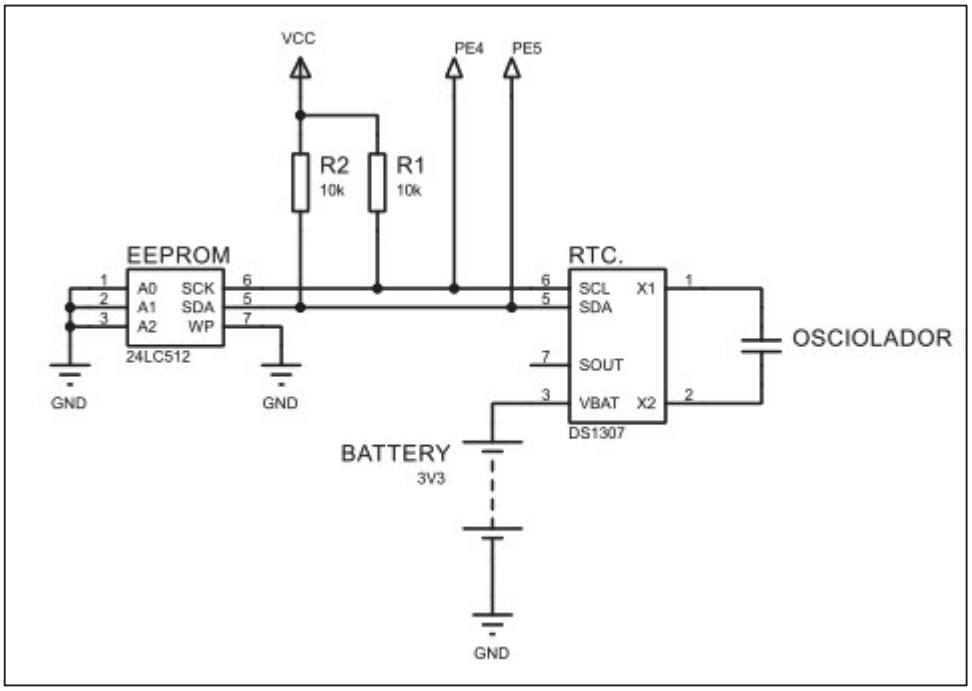
SENSORES	3,3V	Vbus	ALIMENTAÇÃO	DISPLAY_LED_PWM	PF2	GND	GND
	PB5	GND	GND		PF3	PB2	GPIO_DOSADOR
UART_RFID_RX	PB0	PD0	DISPLAY_TOUCH_YP		PB3	PE0	GPIO_BOMBA
UART_RFID_TX	PB1	PD1	DISPLAY_TOUCH_XP		PC4	PF0	SENSOR_NIVEL
I2C2_SCL	PE4	PD2	SU_TRIGGER_BALDE		PC5	RST	
I2C2_SDA	PE5	PD3	SU_FEEDBACK_BALDE		PC6	PB7	DISPLAY_LCD_SDI
DISPLAY_LCD_SDL	PB4	PE1	GPIO_VALVULA1		PC7	PB6	GPIO_DS18B20
DISYPLAY_LCD_SDC	PA5	PE2	GPIO_VALVULA2		PD6	PA4	DISPLAY_LCD_SCS
GPIO_CROSS0	PA6	PE3	GPIO_VALVULA3	DISPLAY_LCD_RST	PD7	PA3	
GPIO_RESISTENCIA	PA7	PF1	GPIO_AGITADOR	DISPLAY_TOUCH_XN	PF4	PA2	

Fonte: Próprio autor

3.3.11.1 Periféricos I2C

O circuito da figura 34 representa o esquemático necessário para as ligações dos periféricos I2C, como o DS1307 (RTC) e o 24LC512 (Memória EEPROM).

Figura 34 – Circuito para periféricos I2C



Fonte: Próprio autor

O RTC DS1307 é um dispositivo usado para medir o tempo de forma muito aproximada ao real, ele é um contador de segundos, minutos, horas, dias, meses e anos válido até 2100. Possui uma memória RAM de 56 bytes para armazenar os dados citados. Suas características técnicas são:

- Alimentação do componente: 5V;
- Bateria necessária para memória: 3,3V;
- Oscilador externo: 32,768 kHz;
- Endereço I2C do componente: 68H;

Para fazer a leitura dos dados gravados em sua memória, necessitamos dos seguintes endereços pré-definidos pelo fabricante:

- 00H – Segundos
- 01H – Minutos

- 02H – Horas
- 03H – Dias da semana (1-7)
- 04H – Dias do mês (1-31)
- 05H – Meses
- 06H – Anos

A memória EEPROM 24LC512 teve de ser utilizada para gravar os dados de alimentações dos animais, pois a memória nativa do microcontrolador é muito pequena. O componente possui as seguintes características técnicas:

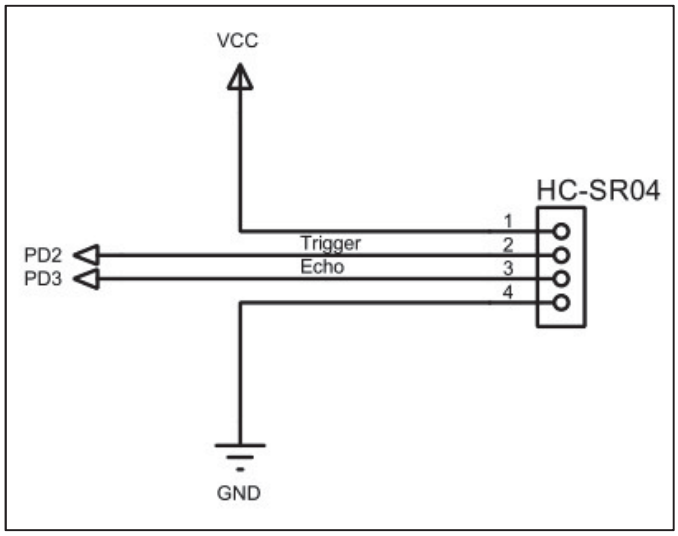
- Tensão de alimentação: 1,8 a 5,5V;
- Corrente máxima de trabalho: 5mA;
- Corrente quiescente: 400uA;
- Retenção de dados: 200 anos;
- Número máximo de gravações: 1 milhão;
- Temperatura de trabalho: -40ºC até 85ºC;
- Endereço do dispositivo: 50H + A2, A1, A0
- Tamanho da memória: 64kbytes;

O endereço I2C do dispositivo é configurado de acordo com as entradas A2, A1 e A0 do componente, desse modo é viável utilizar até 8 componentes iguais no mesmo barramento de comunicação.

3.3.11.2 Leitura sensor HC-SR04

Para leitura deste sensor, não foi necessário nenhum circuito de acoplamento ou resistores de *pull up*, pois a resposta do sensor é diretamente ligada às entradas e saídas digitais do microcontrolador. Como dito anteriormente, a medição é feita através do tempo que o pino de *Echo* fica em nível lógico alto. Circuito representado na figura 35.

Figura 35 – Circuito de comunicação para sensor ultrassônico

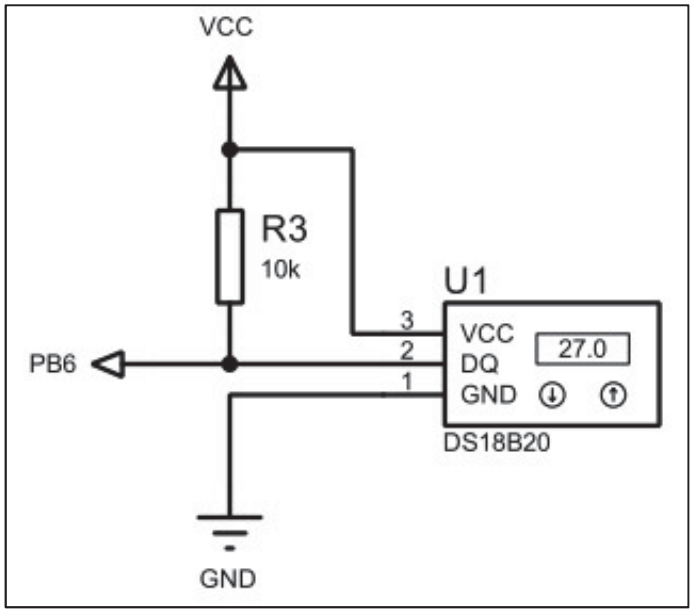


Fonte: Próprio autor

3.3.11.3 Leitura sensor DS18B20

Para leitura desse sensor foi necessário utilizar um resistor de *pull up* de 4,7kΩ conforme seu manual solicita. Ele foi conectado diretamente ao microcontrolador sem nenhum tipo de acoplamento, conforme mostra o circuito da figura 36.

Figura 36 – Circuito de comunicação com o DS18B20

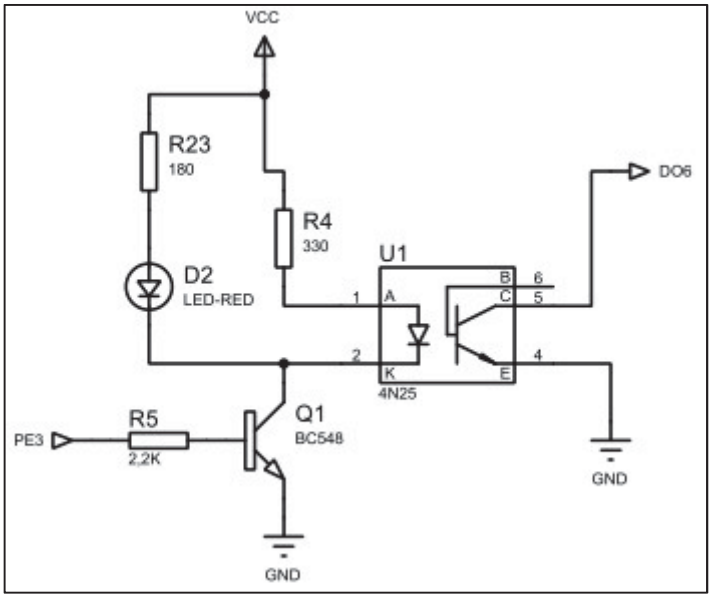


Fonte: Próprio autor

3.3.11.4 Acionamentos

Para os circuitos de acionamentos das cargas foram utilizados opto acopladores 4N25 para evitar qualquer tipo de indução ou condução reversa. Os circuitos foram dimensionados para que a corrente elétrica dos pinos de saídas digitais não ultrapasse o limite de 25mA. A figura 37 mostra como o acoplamento foi feito.

Figura 37 – Circuito de acoplamento para saídas digitais



Fonte: Próprio autor

Para os cálculos de corrente foi considerado a tensão VBE igual a zero, ou seja, o transistor está em saturação.

$$I_{Coletor} = \frac{VCC - V_{4N25}}{330} + \frac{VCC - V_{Led}}{180} \quad (8)$$

$V_{4N25} = 1{,}5V;$

$V_{Led} = 1{,}8V;$

$VCC = 5V;$

$$I_{Coletor} = \frac{5 - 1{,}5}{330} + \frac{5V - 1{,}8}{180} = \ 28{,}4mA$$

Para a utilização dos transistores, foi calculada a corrente mínima de base para que o modo de operação entre na região de saturação, o que significa que VCE tende a ser igual a zero.

$$I_{Coletor} = hFE \times I_{BaseMin} \quad (9)$$

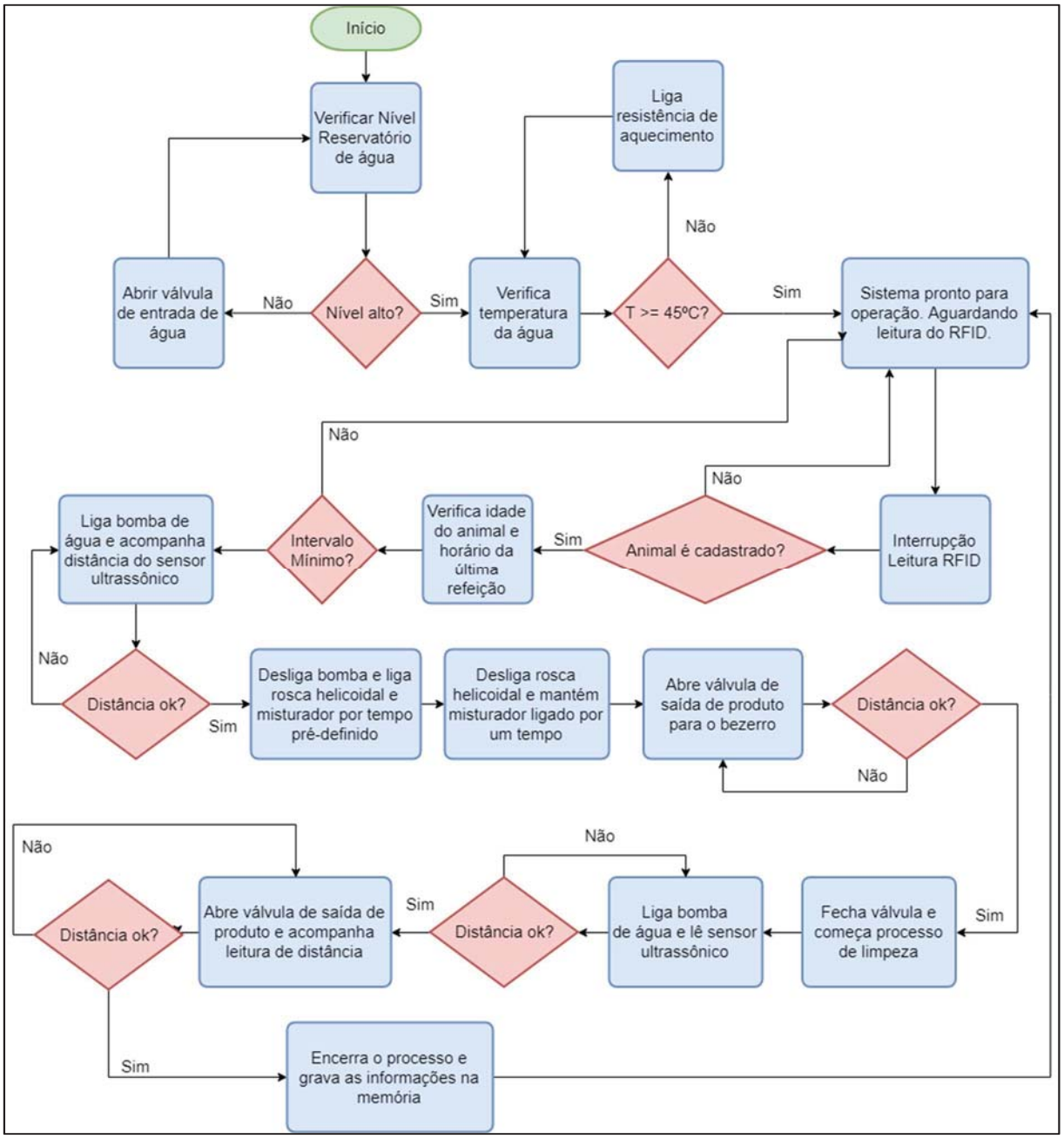
$$I_{BaseMin} = \frac{28,4}{150} = 0,19mA$$

$$I_{BaseReal} = \frac{3,3 - 0,7}{2200} = 1,18mA$$

3.4 FIRMWARE

O *firmware* foi escrito em linguagem C, a qual é utilizada no compilador escolhido. Dentre as opções de compiladores foi escolhido o *IAR Embedded Workbend* para ARM, pois apresenta uma interface mais interativa e familiar. Para o desenvolvimento do firmware foram analisadas as necessidades de hardware, porém, o que mais influenciou em como ele foi desenvolvido foi a escolha da interface com o sistema, o *display*. O funcionamento do protótipo está simplificado no diagrama de blocos da figura 38.

Figura 38 - Diagrama de Blocos do firmware



Fonte: Próprio autor

3.4.1 Leitura dos sensores

Para interpretação dos sensores foram utilizados diferentes métodos, pois cada um tem suas particularidades, como por exemplo a medição de distância do sensor ultrassônico, onde é levado em conta o tempo em que sua saída fica em nível lógico alto.

3.4.1.1 Leitura da distância

Para ler a distância do sensor HC-SR04 é necessário utilizar as configurações e as interrupções do apêndice A. Necessita-se configurar um pino como saída para disparo do trigger, um temporizador com função *time out* para desliga-lo e outro temporizador com função de capturar o tempo em qualquer mudança do estado de um pino pré-definido. O detalhe do funcionamento do sensor está explicado no capítulo 3.3.2, onde mostra os tempos de trigger, o formato da reposta e a equação 7 é utilizada para o cálculo.

Os dados considerados nessa equação foram:

- Velocidade do som: 340,0;
- Conversão do *clock*: 80MHz
- Multiplicação do resultado por 100 para conversão de metro para centímetro;
- Números no formato de 0,0 pois foi utilizado variáveis do tipo *float;*
- Temporizador no formato *Wide-Timer* (32 bits para A e para B);

$$Distância_{cm} = \frac{(Tempo_2 - Tempo_1) \times 340{,}0 \times 100{,}0}{2 \times 80.000.000{,}0} \quad (10)$$

Como não é possível zerar um temporizador, foi utilizado duas equações, uma quando o Tempo2 for maior que o Tempo1 e outra na situação contrária;

$$Distância_{cm} = (2^{32} - [Tempo_1 - Tempo_2]) \times \frac{340{,}0 \times 100{,}0}{2 \times 80.000.000{,}0} \quad (11)$$

3.4.1.2 Leitura da temperatura

O microcontrolador escolhido não possui pinos com comunicação *1-wire* nativa, por isso foi utilizada uma biblioteca a qual tem a função de realizar a comunicação, incluindo os *delays* necessários e a configuração da precisão do sensor. A biblioteca citada está no apêndice E.

Como esse sensor possui uma memória ROM, como explicado no capítulo 3.3.1, há endereços de comandos pré-definidos, ou seja, há necessidade de configurar o pino de dados onde o sensor está conectado como saída para enviar os comandos necessários, posteriormente configuramos o pino como entrada para receber os dados.

Os principais comandos e seus endereços são:

- *convert_T = 0x44;*
- *read_scratchpad = 0xBE;*
- *write_scratchpad = 0x4E;*
- *copy_scratchpad = 0x48;*
- *recall_E2 = 0xB8;*
- *read_power_supply = 0xB4;*
- *skip_ROM = 0xCC;*

Para inicializar o sensor, deve-se executar primeiro a função de reset, a qual é utilizada para detectar o dispositivo. Depois de detectar o dispositivo, envia o comando *skip_ROM*, isso permite que a memória do sensor seja acessada sem necessitar enviar o código ROM de 64 bits, agora já é possível enviar o comando de conversão de temperatura *convert_T*.

Quando o sensor recebe esse comando, o sensor envia o sinal 0 enquanto está realizando a medição e convertendo ela em digital, quando essa conversão está pronta o dispositivo envia sinal 1 para sinalizar o termino. O tempo de conversão varia de acordo com a resolução digital solicitada, variante entre 94ms e 750ms.

Depois de receber o sinal de finalização, envia-se o comando para leitura dos dados, onde os 2 primeiros *bytes* são a temperatura instantânea do sensor. Portanto é feita a conversão respeitando o valor de cada bit, como foi explicado no capítulo 3.3.1.

No *firmware* foi utilizado temporizadores para leitura dos sensores a cada segundo, inicialmente foi utilizado a resolução máxima de 12 *bits* para uma maior precisão no aquecimento da água, porém notou-se que os 750ms de *delay* atrapalhavam o restante do funcionamento, então a resolução final utilizada foi de 11 bits, reduzindo pela metade o tempo. Abaixo segue informações de resolução e seus respectivos *delas:*

- 12 *bits* – LSB: 0,0625 – *delay*: 750ms;
- 11 *bits* – LSB: 0,125 – *delay*: 375ms;
- 10 *bits* – LSB: 0,25 – *delay:* 188ms;
- 9 *bits* – LSB: 0,5 – *delay*: 94ms;

3.4.2 Leitura de RFID

O RMD630, explicado no capítulo de *hardware* foi lido através da comunicação UART. As configurações e as funções de leitura estão no apêndice B. O quadro 5 mostra como são recebidos os dados do leitor pelo microcontrolador.

Quadro 5 - Organização dos dados recebidos pelo leitor RFID

byte 0	byte 1 - 10	byte 11 - 12	byte 13
.	TAG - Caracteres ASCII	checksum	.

Fonte: Próprio autor

Nota-se que são recebidos 14 caracteres, onde o primeiro e o último byte recebido são indicadores de início e fim (“.”). Na sequência, os bytes de número 1-10 são a informação contida no cartão, e as informações nos bytes 11 e 12 são o *checksum*.

O *checksum* são caracteres que representam a função OU exclusivo da informação recebida pelo microcontrolador, utilizado para fazer a verificação e validação da informação. No código é feita a checagem para garantir a integridade dos dados enviados, e somente se ela for verdadeira o vetor com as informações do *TAG* é interpretado pelo programa.

3.4.3 Aquecimento automático

Como mostrado no capítulo 3.3.9, o hardware projetado para fazer o aquecimento da água é capaz de fazer tanto o controle de fase, como o controle *ON/OFF*. Foi escolhido o método *ON/OFF* porque não há chaveamento constante em ângulos diferentes de zero, ou seja, não há distorção harmônica pois não há mudança na forma de onda na carga. Para isso foi utilizado interrupção externa em uma entrada digital com o objetivo de identificar o ângulo zero na tensão de alimentação. Para variar a potência em cima da resistência, foram utilizados variáveis para contar pulsos de passagem por zero e dependendo da temperatura em que a água se encontra, a resistência fica ligada ou desligada por determinados períodos de ciclo.

No apêndice F é mostrado o trecho do *firmware* utilizado para configuração desse sistema, desde a configuração das portas até a quantidade de ciclos ligados e desligados.

3.4.4 Interface com usuário

Para facilitar o método de operação e verificação de dados durante o projeto, foi desenvolvida uma interface do modo mais simples possível. Foram criadas várias telas no projeto, porém, todas elas possuem os mesmos 3 botões de configuração com suas funções de voltar a tela principal, configuração do sistema e verificação de informações do sistema.

O *display* utilizado não tem processamento próprio, ou seja, todas as ações realizadas por ele são feitas pelo próprio microcontrolador, desde a animação dos pixels até a informação recebida do *touch*. Para fazer a programação do *display* deve-se inserir uma série de drivers e bibliotecas feitas pelas *Texas Instruments* (TI), onde possuem funções para cada desenho e escrita utilizada na tela. Tudo é feito com linguagem de programação, pois não há uma interface visual para programação desse display quando utilizado com IAR. No apêndice C estão as configurações e o nome das bibliotecas utilizadas.

O software contém aproximadamente 13 modelos de telas, as quais são utilizadas de forma dinâmica, por exemplo: cada animal possui uma tela específica, porém, a estrutura e os botões são os mesmos, é apenas mudada a variável de estado da tela e o *firmware* sabe quais ações tomar. As figuras 39, 40 e 41 representam as telas principais, as demais estão no apêndice D.

Figura 39 - Tela inicial do *display*

Fonte: Próprio autor

Figura 40 – Tela de configuração

Fonte: Próprio autor

Figura 41 - Tela de informações do sistema

Fonte: Próprio autor

Cada tela tem sua função no protótipo, a tela inicial é o local onde se encontram as informações e controle dos animais. Como o protótipo tem limitação de 8 animais, portanto há 8 botões especificados por números, como por exemplo “B1” que contém informações do bezerro número 1. Quando se utiliza o botão B1, é redirecionado a tela de informações, nela é possível fazer a exclusão do animal, inclusão de outro, verificar a idade do animal cadastrado e a quantidade de refeições já executadas para ele.

A tela de configuração é o local onde é configurado a data e hora do sistema. Também se encontra nessa tela o botão “Motores”, que quando utilizado é levado a uma tela de acionamentos, onde todos os motores e válvulas podem ser ligados de maneira manual. Ela foi desenvolvida com o objetivo de facilitar o teste de cada equipamento utilizado no protótipo.

A tela de informações do sistema é o local onde pode-se verificar a data, hora, temperatura e leitura do sensor de distância. Ela é utilizada principalmente para aferir as informações dos sensores do protótipo e verificar se não há nenhum erro de funcionamento desses dispositivos.

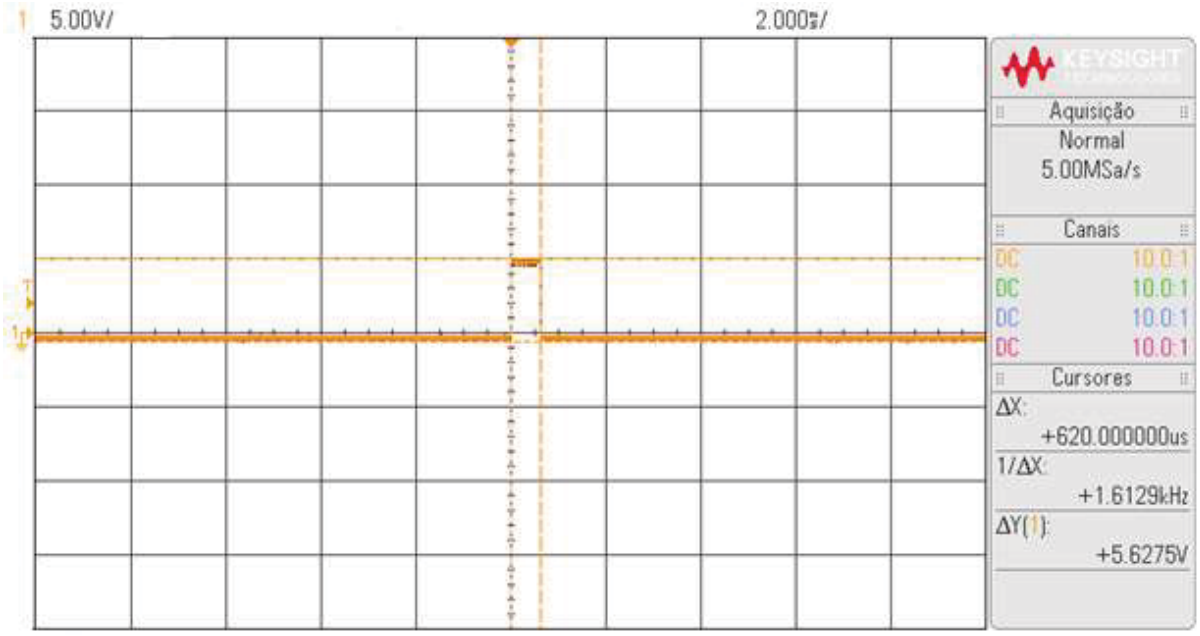
4 RESULTADOS E DISCUSSÃO

Nesse capítulo serão abordados todos os resultados obtidos com a fabricação e montagem do protótipo, os quais apresentaram resultados compatíveis com o objetivo do projeto.

4.1 MEDIÇÃO DE NÍVEL

A figura 42 mostra um sinal enviado pelo sensor e para capturá-lo foi utilizado o osciloscópio.

Figura 42 – Resposta do sensor ultrassônico



Fonte: Próprio autor

A partir do cálculo da distância com as equações do capítulo 3.3.2 o resultado do sinal representado pela figura equivale a distância de 10,54cm, enquanto a distância real medida com régua era de 10,7cm no teste em questão, o local do teste já era no recipiente do misturador. O erro obtido de 0,16cm é aceitável nessa medição.

Testando o controle de nível, foi solicitado ao sistema que ligasse a bomba de água até chegar em uma distância pré-estabelecida e se estabilizasse com uma resposta menor ou igual

a essa. Com o movimento da água, notou-se que havia variação na resposta do sensor em alguns momentos, por isso foi utilizado um temporizador para estabilizar a resposta.

Para a conversão do volume necessário em distância, foi utilizado o método prático, devido aos volumes não lineares do misturador. Desse modo, foi usado um recipiente com volume conhecido para encher o tanque de água com os volumes especificados pelo quadro 1 do capítulo 2.2.1 e retirar os resultados, os quais estão no quadro 6.

Quadro 6 – Conversão de volume em distância

Volume	Distância
0L	34cm
2L	26,4cm
3L	23,9cm
3,5L	22,2cm

Fonte: Próprio autor

Notou-se que a velocidade do som é variável em relação a temperatura do meio ambiente por ser uma onda mecânica. O quadro 7 mostra a variação aproximada:

Quadro 7 – Velocidade do som variável com a temperatura ambiente

Temperatura (ºC)	Velocidade (m/s)
- 10	330
0	332
10	337
20	343
30	350
100	390
500	550
1000	700

Fonte: https://www.todamateria.com.br/velocidade-do-som/ (14/12/2018)

4.2 AQUECIMENTO DE ÁGUA

Para fazer o levantamento das informações necessárias ao aquecimento de água, utilizando o controle *ON/OFF,* foi empregado método de tentativa e erro para encontrar os parâmetros de tempo e temperatura, pois isso varia muito de acordo com a potência do aquecedor e com o volume de água a ser aquecido. Diferentes testes de funcionamento foram feitos, o último realizado e que melhor se adaptou às condições do reservatório de água teve os resultados representados pelo quadro 8.

Quadro 8 – Aquecimento de água

Temperatura	Ciclos ON	Ciclos OFF
≤ 42°C	500	0
42° - 45°C	500	500
45° - 46°C	125	500
≥ 46°C	20	1000

Fonte: Próprio autor

Os valores de ciclos estão em relação ao número de passagem por zeros da rede elétrica, que cada unidade de ciclo equivale a 8,33ms. A rotina de aquecimento acontece da seguinte maneira:

- Temperatura menor 42ºC: resistência 100% do tempo ligada;
- Temperatura entre 42ºC e 45ºC: resistência 50% ligada (período: 8,33s);
- Temperatura entre 45ºC e 46ºC: resistência 25% ligada (período: 5,2s);
- Temperatura acima de 46ºC: resistência 2% ligada (período: 8,5s);

Acima dos 46ºC o tempo ligado é apenas para suprir as perdas térmicas em relação ao ambiente, pois o reservatório utilizado não tem isolação térmica. A temperatura final da água estabilizou-se em 46°C, porém chegou ao objetivo de 45ºC ao longo de 25 minutos. Esses tempos variam de acordo com a temperatura ambiente, pois altera a temperatura inicial da água e variam os valores de perdas térmicas.

4.3 VÁLVULAS DE SAÍDA DE PRODUTO DO MISTURADOR

Com a utilização de válvulas de latão normalmente utilizadas em tubulações de gás, não houve problema com a falta de pressão do misturador, pois estas válvulas não necessitam de pressão para abrir completamente o êmbolo de vedação. Devido a falta de conhecimento na área de mecânica, mais especificamente vazão de fluídos, houve problemas com o tempo de esvaziamento do misturador. Para esvaziar o volume máximo utilizado de 3,5 litros apenas pela gravidade, o tempo levado foi cerca de 5 minutos e esse problema seria facilmente solucionado utilizando uma válvula maior, porém, quanto maior a bitola maior é o preço do equipamento.

4.4 TRANSPORTADOR DE LEITE EM PÓ

Após vários testes de dosagem, notou-se que a rosca transportadora tinha uma dosagem quase que constante de 2 gramas por segundo. Por esse motivo, foram utilizados temporizadores para fazer a dosagem conforme a necessidade de alimentação dos animais, conforme relata o capítulo 2.2.1.

O motor utilizado, já acoplado na rosca, possui uma caixa de redução com saída de 80 rotações por minuto (RPM). A corrente elétrica solicitada por este motor, quando utilizado com carga, é de 1,2A.

4.5 LISTA DE MATERIAIS

O quadro 9 mostra os materiais mais significativos que tiveram que ser comprados para fabricação do projeto. Os produtos que não aparecem no quadro foram retirados do almoxarifado da universidade ou emprestados por empresas.

Quadro 9 – Lista de materiais comprados

Descrição	Qtd.	Preço Unitário	Frete	Valor total
Válvula solenoide latão 1/4	2	R$ 120,00	R$ 40,00	R$ 280,00
Válvula solenoide simples 1/2	1	R$ 16,00	R$ -	R$ 16,00
Estrutura de madeira	1	R$ 280,00	R$ 50,00	R$ 330,00
Motor monofásico com capacitor permanente	1	R$ 50,00	R$ 50,00	R$ 100,00
Bobma de água submersa	1	R$ 90,00	R$ -	R$ 90,00
Resistência	1	R$ 20,00	R$ 23,00	R$ 43,00
Sensor de Temperatura	1	R$ 20,00	R$ 30,00	R$ 50,00
Sensor Ultrassônico	1	R$ 10,00	R$ 25,00	R$ 35,00
Leitor RFID	1	R$ 50,00	R$ 30,00	R$ 80,00
Reservatório misturador	1	R$ 3,00	R$ -	R$ 3,00
Reservatório de água	1	R$ 20,00	R$ -	R$ 20,00
Placa de montagem	1	R$ 50,00	R$ -	R$ 50,00
Canaletas - 2x	2	R$ 17,00	R$ -	R$ 34,00
Cabo 1mm² (100m)	1	R$ 43,00	R$ -	R$ 43,00
LaunchPad TI TM4C123GXL	1	R$ 50,00	R$ 50,00	R$ 100,00
Display Kentec TI	1	R$ 80,00	R$ 30,00	R$ 110,00
Total:	R$			1.384,00

Fonte: Próprio autor (2018)

5 CONSIDERAÇÕES FINAIS

O trabalho desenvolvido teve como desafio inicial o projeto da estrutura mecânica da máquina, a qual foi estudada para ser executada da maneira mais rápida e econômica possível. Não há embasamento teórico para cálculos estruturais, por isso, foi fundamental o conselho de pessoas experientes nesta área. Também, houve um grande desafio no desenvolvimento de uma interface interativa para fazer a operação da máquina, como o acionamento manual, cadastros e exclusão de animais do sistema.

Através do desenvolvimento conjunto do *hardware* e *firmware* foi possível explorar diversas áreas da engenharia elétrica, que como consequência gerou uma ligação entre toda teoria vista no curso e a prática desenvolvida.

O protótipo não foi preparado para se tornar um produto, pois nesse projeto foi priorizado a economia e não a qualidade necessária para sua comercialização. Um exemplo disso foi o leitor RFID empregado, que tem distância máxima de leitura de 50mm e deveria ser de no mínimo 500mm, porém, o custo de um componente capaz de alcançar tais distâncias era totalmente inviável. Outro exemplo foi a estrutura de madeira utilizada que no campo não é indicado, pois além de frágil esse material não é resistente a umidade.

Nota-se que o transportador horizontal, por ser uma peça que foi retirada de uma máquina de café, não é ideal ao protótipo, pois a capacidade de armazenamento de leite em pó é inferior a necessidade real. O preço de uma peça semelhante que atenda essa necessidade é superior e para testes funcionais não era necessário.

Com o término do projeto, é possível verificar que o mesmo cumpriu o objetivo proposto, embora não alcançou os patamares para tornar-se um produto. Através dos testes realizados comprovou-se que todos os periféricos trabalharam em harmonia, desde os sensores, leitor RFID, motores e interface com o usuário. Portanto, pode-se concluir que o sistema apresenta as funcionalidades necessárias, porém há muito ainda a ser desenvolvido caso o objetivo seja evoluir o protótipo a um produto, pois todos os componentes utilizados não são preparados para ambientes industriais.

3REFERÊNCIAS

AGUIRRE, Luis Antonio. **Fundamentos de Instrumentação**. São Paulo: Perason Education do Brasil, 2013.

BALBINOT, A. e BRUSAMARELLO, V. J. **Instrumentação e Fundamentos de Medidas**. 2. ed. Rio de Janeira: LTC, 2012.

BARROS, Vicente Pereira de. **Física Geral**: Eletricidade – para além do dia a dia. Curitiba: InterSaberes, 2017.

BATISTTON, Walter Cazellato. **Gado Leiteiro**. Campinas: Instituto Campineiro de Ensino Agrícola, 1983.

BRAGA, Newton C. 2014. Disponível em: <http://www.newtoncbraga.com.br/index.php/como-funciona/3890-mec095>. Acesso em: 15 out. 2017.

BRAZ, Joesley. **Eletroválvula**. 2017. Disponível em: <http://eletrofisica1.blogspot.com.br/2013/04/eletrovalvula.html>. Acesso em: 15 out. 2017.

CARVALHO, Limírio de Almeida, et al. **Importância Econômica**. Embrapa. 2002. Disponível em: <https://sistemasdeproducao.cnptia.embrapa.br/FontesHTML/Leite/LeiteCerrado/importancia.html>. Acesso em: 9 out. 2017.

CONAB – Companhia Nacional de Abastecimento. **Conjuntura Mensal Especial - Leite e Derivados**. Abril de 2017. Disponível em: <http://www.conab.gov.br/OlalaCMS/uploads/arquivos/17_05_15_14_13_38_leite_abril_2017.pdf>. Acesso em: 9 out. 2017a.

CONAB – Companhia Nacional de Abastecimento. **Conjuntura Mensal - Leite e Derivados**. Abril de 2017. Disponível em: <http://www.conab.gov.br/OlalaCMS/uploads/arquivos/17_09_13_17_27_54_leite_agosto_2017.pdf>. Acesso em: 12 out. 2017b.

DALLAS SEMICONDUCTOR. **1-Wire Digital Thermometer**. Disponível em: <http://pdf1.alldatasheet.com/datasheet-pdf/view/58557/DALLAS/DS18B20.html>. Acesso em: 19 out. 2017.

DE SOUZA, Flávia Martins. **Docente da UFG**: manejo alimentar do nascimento ao desaleitamento de fêmeas bovinas leiteiras. 2011. Dissertação (Pós-graduação em ciência animal) – Escola de Veterinária e Zootecnia, Universidade Federal de Goiás, Goiânia, 2011.

GONSALES, Samuel. **Etiquetas RFID revolucionando a gestão estoques**. Março de 2017. Disponível em: <https://www.ecommercebrasil.com.br/artigos/etiquetas-rfid-revolucionando-gestao-estoques/>. Acesso em: 21 out. 2017.

KRUG, Ernesto Enio Budke, et al. **Manual da Produção Leiteira**. 2. ed. Porto Alegre: CCGL, 1993.

MARTINS, Nardênio Almeida. **Sistemas Microcontrolados:** Uma abordagem com o Microcontrolador PIC 16F84. São Paulo: Novatec, 2005.

MILMAN, Mário José. **Equipamentos para pré-processamento de grãos**. Pelotas: Editora e Gráfica Universitária – Universidade Federal de Pelotas, 2002.

PEIXOTO, A. M.; DE MOURA, J. C.; DE FARIA, V. P. **Bovinocultura Leiteira**: Fundamentos da Exploração Racional. 3. ed. Piracicaba: FEALQ, 2000.

PREFEITURA MUNICIPAL DE PORTO ALEGRE. **Tarifas 2017**. 2017. Disponível em: <http://www2.portoalegre.rs.gov.br/dmae/default.php?p_secao=370>. Acesso em: 12 out. 2017.

SPRAYFO. **Programa de Alimentação**. 2017. Disponível em: <https://www.sprayfo.com/siteassets/br-produtos/sprayfo-1-7-basic-feeding-plan-automatic-web_bra.pdf>. Acesso em: 13 out. 2017.

TORO, **Vicente Del. Fundamentos de máquinas elétricas.** Tradução Onofre de Andrade Martins. Rio de Janeiro: Prentice-Hall do Brasil, 1994.

YOUNG, H. D. e FREEDMAN, R. A. **Física II**: Termodinâmica e Ondas. 14. ed. São Paulo: Pearson Education do Brasil Ltda., 2015.

WEIGHTECH. **Célula de carga**. Disponínel em: <http://www.weightech.com.br/detalhes.asp?id=100793&n=BSPL-040>. Acesso em: 21 out. 2017.

APÊNDICE A – TRECHO DO FIRMWARE QUE REALIZA MEDIÇÃO DE DISTÂNCIA DO SENSOR ULTRASSÔNICO

```
#define PD2_TriggerSU (*((volatile long *) 0x40007010)) // saída (0x04)
#define GPIO_PD3_WT3CCP1        0x00030C07 // border count PD3 – entrada

 SysCtlPeripheralEnable(SYSCTL_PERIPH_WTIMER3);
 TimerConfigure(WTIMER3_BASE, TIMER_CFG_SPLIT_PAIR |
TIMER_CFG_A_ONE_SHOT_UP | TIMER_CFG_B_CAP_TIME_UP);
 //A
 TimerLoadSet(WTIMER3_BASE, TIMER_A, 200000); // 12,5ns * valor, tempo final do timer
TRIGGER
 TimerIntEnable(WTIMER3_BASE, TIMER_TIMA_TIMEOUT);    // habilita o tipo da funcao
 IntEnable(INT_WTIMER3A_TM4C123);
 IntRegister(INT_WTIMER3A_TM4C123, WTimer3A_IntHandler);
 TimerIntClear(WTIMER3_BASE, TIMER_TIMA_TIMEOUT);
 //B
 TimerControlEvent(WTIMER3_BASE, TIMER_B, TIMER_EVENT_BOTH_EDGES);
 SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD);
 GPIOPinTypeTimer(GPIO_PORTD_BASE,GPIO_PIN_3);
 GPIOPinConfigure(GPIO_PD3_WT3CCP1);
 IntRegister(INT_WTIMER3B_TM4C123,WTimer3B_IntHandler);
 TimerIntClear(WTIMER3_BASE, TIMER_CAPB_EVENT);
 TimerIntEnable(WTIMER3_BASE, TIMER_CAPB_EVENT);
 IntEnable(INT_WTIMER3B_TM4C123);
 IntPrioritySet(INT_WTIMER3B_TM4C123, 0);
 //GPIO define o pino PD2_TriggerSU como saida
 GPIOPinTypeGPIOOutput(GPIO_PORTD_BASE,GPIO_PIN_2);
```

```
void WTimer3B_IntHandler(void){
  if (Nivel_aux == 1 )
  {
    t2 = TimerValueGet(WTIMER3_BASE,TIMER_B);
    TimerDisable(WTIMER3_BASE,TIMER_B);
    if (t2 > t1)    {
      distancia = (t2 - t1)*340.0/2.0/800000.0;
    }
    else    {
      distancia = (pow(2,32) - (t1 - t2))*340.0/2.0/800000.0;
    }
    Nivel_aux = 2;
  }
  if (Nivel_aux == 0 )  {
    t1 = TimerValueGet(WTIMER3_BASE,TIMER_B);
    Nivel_aux = 1;
  }
  TimerIntClear(WTIMER3_BASE, TIMER_CAPB_EVENT);
}

void WTimer3A_IntHandler(void){
  TimerDisable(WTIMER3_BASE,TIMER_A);
  PD2_TriggerSU=0;
  TimerIntClear(WTIMER3_BASE, TIMER_TIMA_TIMEOUT);
}
```

APÊNDICE B - TRECHO DO SOFTWARE DE LEITURA DO CARTÃO RDM6300

```
#define GPIO_PB0_U1RX   0x00010001
#define GPIO_PB1_U1TX   0x00010401

  SysCtlPeripheralReset(SYSCTL_PERIPH_UART1);
  SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB);
  SysCtlPeripheralEnable(SYSCTL_PERIPH_UART1);
  GPIOPinConfigure(GPIO_PB0_U1RX);
  GPIOPinConfigure(GPIO_PB1_U1TX);
  GPIOPinTypeUART(GPIO_PORTB_BASE, GPIO_PIN_0|GPIO_PIN_1);
  UARTClockSourceSet(UART1_BASE, UART_CLOCK_SYSTEM);
  UARTFIFOLevelSet(UART1_BASE, UART_FIFO_TX6_8, UART_FIFO_RX6_8);
  UARTFIFOEnable(UART1_BASE);
  UARTConfigSetExpClk(UART1_BASE, SysCtlClockGet(), 9600, (UART_CONFIG_WLEN_8
| UART_CONFIG_STOP_ONE | UART_CONFIG_PAR_NONE));
  UARTIntEnable(UART1_BASE,UART_INT_RT);
  IntEnable(INT_UART1_TM4C123);
  IntRegister(INT_UART1_TM4C123,UART1_IntHandler);

void UART1_IntHandler (void){
  x=UARTCharGet(UART1_BASE);
  if ((aux2 > 0) && (aux2 < 13) ) TagRFID [aux2] = x;
  aux2++;
  if (aux2 == 14)   {
    aux2 =0;
    Att1 = true;
  }
  UARTIntClear(UART1_BASE,UART_INT_RT);
}
```

APÊNDICE C – TRECHO DE CONFIGURAÇÕES E FUNÇÕES DO DISPLAY

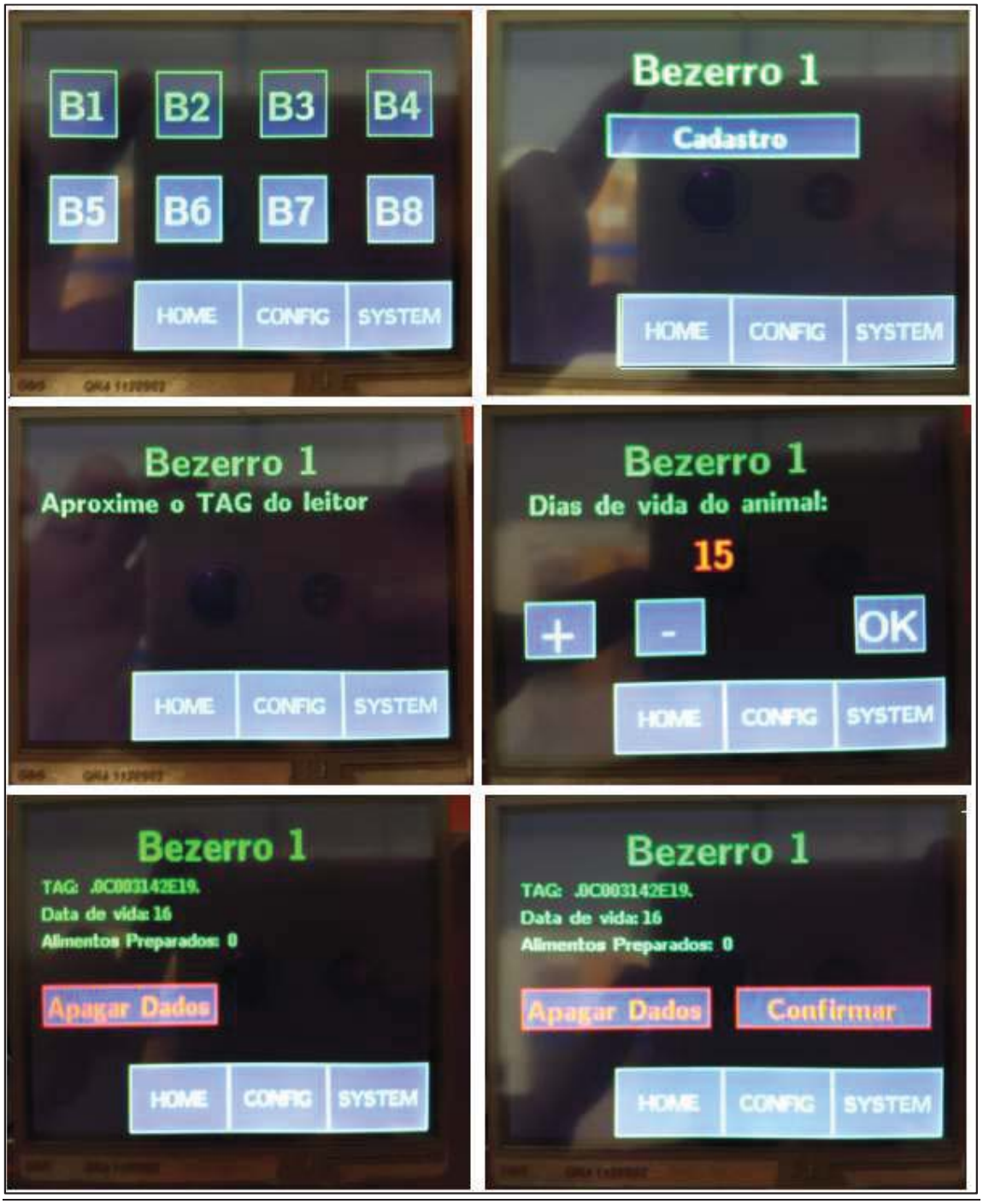
```
#include "drivers/touch.h"
#include "grlib/grlib.h"
#include "drivers/Kentec320x240x16_ssd2119_8bit.h"
#include "grlib/widget.h"
#include "grlib/canvas.h"
#include "grlib/checkbox.h"
#include "grlib/container.h"
#include "grlib/pushbutton.h"
#include "grlib/radiobutton.h"
#include "grlib/slider.h"
tContext sContext;
tRectangle sRect;
uint32_t ui32SysClock_touch;
//Ponteiros
extern tPushButtonWidget g_B1;
//Interrupções TOUCH
void OnButtonPressB1(tWidget *pWidget){
Int_B1 = true;
}
//Botão TOUCH
RectangularButton( // BOTAO B1
        g_B1, // nome do botão em questão
        WIDGET_ROOT,
        0, //nome do próximo botão a ser utilizado
        0,
        &g_sKentec320x240x16_SSD2119,
        15, //Posicao X Inicial
        30, //Posicao Y Inicial
        50, //Largura do Retangulo
        50, //Altura do Retangulo
        (PB_STYLE_OUTLINE | PB_STYLE_TEXT_OPAQUE | PB_STYLE_TEXT |
PB_STYLE_FILL),
        ClrGray, //Cor do fundo do botão
```

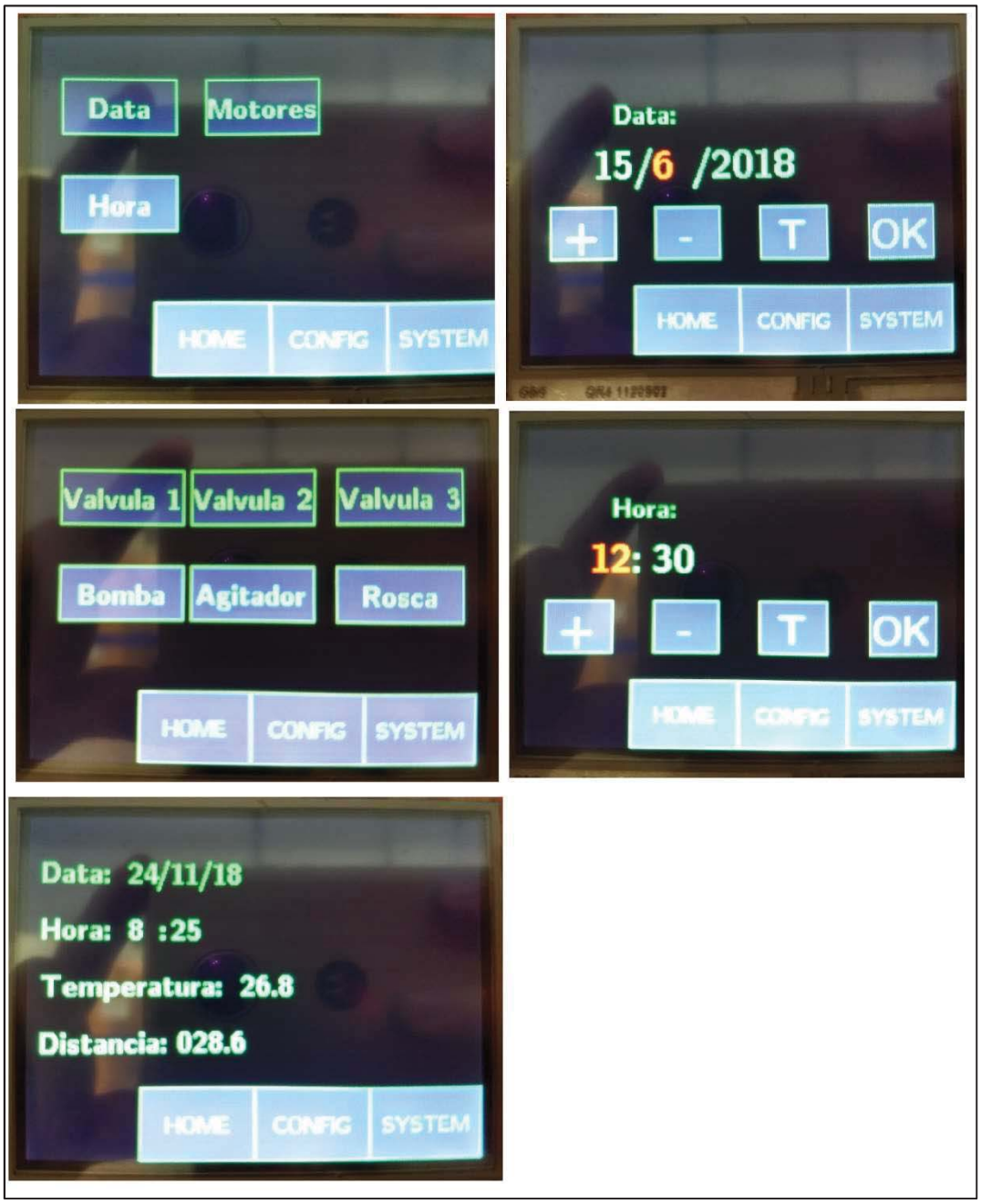
```
            ClrWhite, //Cor do fundo quanto pressionado
            ClrLightGreen, //Cor do retangulo de fora
            ClrLightGreen, // Cor da letra
            g_psFontCmss30b, //Fonte e tamanho da letra
            "B1", //Texto
            0, //Imagem do botão
            0, // Imagem do botão pressionado
            0, //Auto repeat delay
            0, //Auto repeat rate
            OnButtonPressB1 //Nome da função de interrupção
             );

void ClrScreen(void){ // Desenha um retangulo preto na tela
  sRect.i16XMin = 1;
  sRect.i16YMin = 1;
  sRect.i16XMax = 319;
  sRect.i16YMax = 239;
  GrContextForegroundSet(&sContext, ClrBlack);
  GrRectFill(&sContext, &sRect);
  GrFlush(&sContext);
}

Main{
        ClrScreen();
         WidgetAdd(WIDGET_ROOT, (tWidget *)&g_B1);
        WidgetPaint(WIDGET_ROOT);
        While{
                WidgetMessageQueueProcess(); //LEITURA DO TOUCH
        }
}
```

APÊNDICE D – TELAS FEITAS PARA INTERAÇÃO COM O USUÁRIO




Data
Motores
Hora
HOME
CONFIG
SYSTEM
Data:
15/6 /2018
+
-
T
OK
HOME
CONFIG
SYSTEM
Valvula 1
Valvula 2
Valvula 3
Bomba
Agitador
Rosca
HOME
CONFIG
SYSTEM
Hora:
12: 30
+
-
T
OK
HOME
CONFIG
SYSTEM
Data: 24/11/18
Hora: 8 :25
Temperatura: 26.8
Distancia: 028.6
HOME
CONFIG
SYSTEM

APÊNDICE E – BIBLIOTECA PARA UTILIZAÇÃO DO SENSOR DS18B20

```
#define convert_T             0x44
#define read_scratchpad       0xBE
#define write_scratchpad      0x4E
#define copy_scratchpad       0x48
#define recall_E2             0xB8
#define read_power_supply     0xB4
#define skip_ROM              0xCC

unsigned char resolution = 9;

void init_DS18B20();
float get_temperature();

//void delayMs(uint32_t ui32Ms) {
//
//  // 1 clock cycle = 1 / SysCtlClockGet() second
//  // 1 SysCtlDelay = 3 clock cycle = 3 / SysCtlClockGet() second
//  // 1 second = SysCtlClockGet() / 3
//  // 0.001 second = 1 ms = SysCtlClockGet() / 3 / 1000
//
//  SysCtlDelay(ui32Ms * (SysCtlClockGet() / 3 / 1000));
//}

short onewire_reset();
void onewire_write_bit(short bit_value);
short onewire_read_bit();
void onewire_write(unsigned char value);
unsigned char onewire_read();
void delayUs(uint32_t ui32Us) {
  SysCtlDelay(ui32Us * (SysCtlClockGet() / 3 / 1000000));
}
short onewire_reset()  {
   short res = 0;          // detecta o pulso de presenÃ§a do dispositivo.
```

```
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB);
    GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE,GPIO_PIN_6);
    GPIOPinWrite(GPIO_PORTB_BASE,GPIO_PIN_6,0);
    delayUs(480);
    GPIOPinTypeGPIOInput(GPIO_PORTB_BASE,GPIO_PIN_6); //libera o barramento
atraves do resistor de pull-up.
    delayUs(60);              //aguarda de 60 a 100us.
    res = GPIOPinRead(GPIO_PORTB_BASE,GPIO_PIN_6);                    //antes de
puxar o Ã´nibus para baixo.
     delayUs(480);          //escravo segura o barramento de 60 a 240 us antes de liberar o
Ã´nibus novamente.
    return res;              //retorna o pulso de prensenÃ§a do dispojsotivo.
}
void onewire_write_bit(short bit_value){
   GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE,GPIO_PIN_6);
   GPIOPinWrite(GPIO_PORTB_BASE,GPIO_PIN_6,0);
   if(bit_value){
      delayUs(104);
   GPIOPinWrite(GPIO_PORTB_BASE,GPIO_PIN_6,GPIO_PIN_6);
   }
}
short onewire_read_bit() {
   GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE,GPIO_PIN_6);
   GPIOPinWrite(GPIO_PORTB_BASE,GPIO_PIN_6,0);
   GPIOPinWrite(GPIO_PORTB_BASE,GPIO_PIN_6,GPIO_PIN_6);
   delayUs(15);
   GPIOPinTypeGPIOInput(GPIO_PORTB_BASE,GPIO_PIN_6);
   return(GPIOPinRead(GPIO_PORTB_BASE,GPIO_PIN_6));
}
void onewire_write(unsigned char value) {
    unsigned char s = 0;
    while(s < 8)   {
       if((value & (1 << s)))   {
         GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE,GPIO_PIN_6);
```

```
        GPIOPinWrite(GPIO_PORTB_BASE,GPIO_PIN_6,0);
        delayUs(1);
        GPIOPinTypeGPIOInput(GPIO_PORTB_BASE,GPIO_PIN_6);
        delayUs(60);
      }
      else   {
        GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE,GPIO_PIN_6);
        GPIOPinWrite(GPIO_PORTB_BASE,GPIO_PIN_6,0);
        delayUs(60);
        GPIOPinTypeGPIOInput(GPIO_PORTB_BASE,GPIO_PIN_6);
        delayUs(1);
      }
      s++;
   }
}
unsigned char onewire_read()   {
   unsigned char s = 0x00;
   unsigned char value = 0x00;
   while(s < 8)       {
     GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE,GPIO_PIN_6);
     GPIOPinWrite(GPIO_PORTB_BASE,GPIO_PIN_6,0);
     delayUs(2);
     GPIOPinTypeGPIOInput(GPIO_PORTB_BASE,GPIO_PIN_6);
      if(GPIOPinRead(GPIO_PORTB_BASE,GPIO_PIN_6)) {
         value |=  (1 << s);
      }
      delayUs(60);
      s++;
   }
   return value;
}
void init_DS18B20()    {
  onewire_reset(); //inicia processo de reset
}
```

```
float get_temperature() { //executa o recebimento do valor de temperatura
  unsigned char msb = 0;
  unsigned char lsb = 0;
  register float temp = 0.0;
  onewire_reset();       //executa o reset
  onewire_write(skip_ROM);//permitindo que o mestre de rede para acessar as funÃ§Ãµes de memÃ³ria sem fornecer o cÃ³digo ROM de 64 bits.
  onewire_write(convert_T); // meste solicita que o escravo faÃ§a a conversao
  switch(resolution)  {       // resoluÃ§ao selevionÃ¡vel que no programa foi selecionada a 12
    case 12: {
      delayMs(750);
      break;
    }
    case 11:  {
      delayMs(375);
      break;
    }
    case 10:  {
      delayMs(188);
      break;
    }
    case 9:   {
      delayMs(94);
      break;
    }
  }

  onewire_reset();              //executa o reset
  onewire_write(skip_ROM);        //permitindo que o mestre de rede para acessar as funÃ§Ãµes de memÃ³ria
  onewire_write(read_scratchpad);  //Este comando permite ao mestre ler o conteÃºdo do scratchpad do escravo
  lsb = onewire_read();           //Os dados sÃ£o lidos primeiro em LSB
  msb = onewire_read();           //Le os dados do lado msb
```

```
  temp = msb;
  temp *= 256; //a = a * b  ou seja temp= temp*256 //le a parte mais significativa
  temp += lsb; //= temp = temp+lsb  //le a parte menos significtiva
  switch(resolution)  {
    case 12:  {
      temp *= 0.0625;
      break;
    }
    case 11:  {
      temp *= 0.125;
      break;
    }
    case 10:  {
      temp *= 0.25;
      break;
    }
    case 9:  {
      temp *= 0.0625;
      break;
    }
  }
  delayMs(40);
  return (temp);
}
```

APÊNDICE F – TRECHO DO FIRMWARE PARA AQUECIMENTO DE AGUA

```
#define PA6_InputCrossZero (*((volatile long *) 0x40004100)) // entrada digital
#define PA7_OutputRES (*((volatile long *) 0x40004200)) // saida digital disparo MOC3020

  //PA6_InputCrossZero - CrossZero Input + Interrupção
  GPIOPinTypeGPIOInput(GPIO_PORTA_BASE,GPIO_PIN_6);
  GPIOPadConfigSet(GPIO_PORTA_BASE, GPIO_PIN_6, GPIO_STRENGTH_2MA,
GPIO_PIN_TYPE_STD_WPD);//pulldown interno
 GPIOIntTypeSet(GPIO_PORTA_BASE,GPIO_PIN_6,GPIO_FALLING_EDGE);//configura
duas bordas
  GPIOIntEnable(GPIO_PORTA_BASE,GPIO_PIN_6);// habilita o pino
  IntEnable(INT_GPIOA_TM4C123);
  IntRegister(INT_GPIOA_TM4C123,GPIO_A_IntHandler); // nome da funcao de
interrupcao
  //PA7_SAIDA
  GPIOPinTypeGPIOOutput(GPIO_PORTA_BASE,GPIO_PIN_7);
void GPIO_A_IntHandler (void){ //Interrupção GPIO PA6_InputCrossZero
//  if(CountSemiCiclos%100 == 0)
//  {
//    UARTprintf("\nPasso: %d", PassoAutomatico);
//    UARTprintf("\nPF0: %d", PF0_NivelBoia);
//  }
  if (CountSemiCiclos <= Res_TempoOn)   {
    if ((GPIOPinRead(GPIO_PORTF_BASE, PF0_NivelBoia)) == 0)     PA7_OutputRES
=0x80;
    else PA7_OutputRES = 0;
  }
  if (CountSemiCiclos > Res_TempoOn)    PA7_OutputRES=0;
  if (CountSemiCiclos >= Res_TempoOn + Res_TempoOff)   CountSemiCiclos=0;
  CountSemiCiclos ++;
  GPIOIntClear(GPIO_PORTA_BASE, GPIO_PIN_6);
}
//MAIN
```

```
if (temperatura_agua <= 42.0){
  Res_TempoOff = 0;
  Res_TempoOn = 500;
}
if ((temperatura_agua > 42.0)&&(temperatura_agua<=45.0)){
  Res_TempoOff = 500;
  Res_TempoOn = 500;
}
if ((temperatura_agua > 45.0)&&(temperatura_agua<=46.0)){
  Res_TempoOff = 1000;
  Res_TempoOn = 100;
}
if (temperatura_agua > 46.0){
  Res_TempoOff = 1600;
  Res_TempoOn = 0;
}
```

APÊNDICE G – TRECHO DO FIRMWARE PARA CONFIGURAÇÃO I2C

```
#define GPIO_PE4_I2C2SCL        0x00041003
#define GPIO_PE5_I2C2SDA        0x00041403
//Endereço memória eeprom I2C
#define SLAVE_EE0 0x50
//Endereços do RTC
#define SLAVE_ADDR 0x68
#define SEC   0x00
#define MIN   0x01
#define HRS   0x02
#define DAY   0x03
#define DATE  0x04
#define MONTH 0x05
#define YEAR  0x06
#define CNTRL 0x07
//CONFIG
  SysCtlPeripheralEnable(SYSCTL_PERIPH_I2C2);
  SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOE);
  GPIOPinConfigure(GPIO_PE4_I2C2SCL);
  GPIOPinConfigure(GPIO_PE5_I2C2SDA);
  GPIOPinTypeI2CSCL(GPIO_PORTE_BASE, GPIO_PIN_4);
  GPIOPinTypeI2C(GPIO_PORTE_BASE, GPIO_PIN_5);
  I2CMasterInitExpClk(I2C2_BASE, SysCtlClockGet(), false);

void ByteWrite(char slave, unsigned int end, char dado)  {
  int add_h, add_l;
  add_h=end;
  add_l=end;
  add_h=add_h/256;
  add_l=add_l&0x00ff;
  for(uint32_t j=0; j<50000; j++); //tempo antes de nova gravação
  I2CMasterSlaveAddrSet(I2C2_BASE, slave, false); //end. do dispositivo
  I2CMasterDataPut(I2C2_BASE, add_h); //parte alta do endereço
  I2CMasterControl(I2C2_BASE, I2C_MASTER_CMD_BURST_SEND_START); //start
```

```
  while(I2CMasterBusy(I2C2_BASE));
  I2CMasterDataPut(I2C2_BASE, add_l); //parte baixa do endereço
  I2CMasterControl(I2C2_BASE, I2C_MASTER_CMD_BURST_SEND_CONT);
  while(I2CMasterBusy(I2C2_BASE));
  I2CMasterDataPut(I2C2_BASE, dado); //dado a ser gravado
  I2CMasterControl(I2C2_BASE, I2C_MASTER_CMD_BURST_SEND_FINISH);
  while(I2CMasterBusy(I2C2_BASE));
}

unsigned char GetData24LC512(char slave, unsigned int end)  {
  unsigned char dado;
  int add_h, add_l;
  add_h=end;
  add_l=end;
  add_h=add_h/256;
  add_l=add_l&0x00ff;
  I2CMasterSlaveAddrSet(I2C2_BASE, slave, false);
  I2CMasterDataPut(I2C2_BASE, add_h); //end. inici
  I2CMasterControl(I2C2_BASE, I2C_MASTER_CMD_BURST_SEND_START); //start
  while(I2CMasterBusy(I2C2_BASE));
  I2CMasterDataPut(I2C2_BASE, add_l); //end. inicial
  I2CMasterControl(I2C2_BASE, I2C_MASTER_CMD_BURST_SEND_FINISH); //start
  while(I2CMasterBusy(I2C2_BASE));

  I2CMasterSlaveAddrSet(I2C2_BASE, slave, true);
  I2CMasterControl(I2C2_BASE, I2C_MASTER_CMD_SINGLE_RECEIVE); // sel.recepção
  while(I2CMasterBusy(I2C2_BASE));

  dado=I2CMasterDataGet(I2C2_BASE); //leitura

  return dado;
}
```

```
unsigned char dec2bcd(unsigned char valor)  {  //transforma um número do formato decimal
para BCD
  return (((valor/10)<<4) | (valor%10));
}
unsigned char bcd2dec(unsigned char valor)  {  //transforma um número do formato BCD
para decimal
  return (((valor&0xF0)>>4)*10) + (valor&0x0F);
}
void SetTimeRTC(char hora, char minuto, char segundo)  {  //Seta o tempo
  uint8_t vet[3];
  vet[0]=dec2bcd(segundo);
  vet[1]=dec2bcd(minuto);
  vet[2]=dec2bcd(hora);

  I2CMasterSlaveAddrSet(I2C2_BASE, SLAVE_ADDR, false); //end. do dispositivo, R/S=0
  I2CMasterDataPut(I2C2_BASE, SEC); //end. inicial de gravação SEC=0x00
  I2CMasterControl(I2C2_BASE, I2C_MASTER_CMD_BURST_SEND_START); //start
  while(I2CMasterBusy(I2C2_BASE));

  for(char i=0; i<3; i++)    {
    I2CMasterDataPut(I2C2_BASE, vet[i]);
    I2CMasterControl(I2C2_BASE, I2C_MASTER_CMD_BURST_SEND_CONT); //envia
    while(I2CMasterBusy(I2C2_BASE));
  }

  I2CMasterControl(I2C2_BASE, I2C_MASTER_CMD_BURST_SEND_FINISH); //stop
  while(I2CMasterBusy(I2C2_BASE));
}
void SetDataRTC(char dia, char mes, char ano)  {  //Seta a data
  uint8_t vet[3];
  vet[0]=dec2bcd(dia);
  vet[1]=dec2bcd(mes);
  vet[2]=dec2bcd(ano);
```

```
  I2CMasterSlaveAddrSet(I2C2_BASE, SLAVE_ADDR, false); //end. do dispositivo, R/S=0
  I2CMasterDataPut(I2C2_BASE, DATE); //end. inicial de gravação DATE=0x04
  I2CMasterControl(I2C2_BASE, I2C_MASTER_CMD_BURST_SEND_START); //start
  while(I2CMasterBusy(I2C2_BASE));

  for(char i=0; i<3; i++)    {
    I2CMasterDataPut(I2C2_BASE, vet[i]);
    I2CMasterControl(I2C2_BASE, I2C_MASTER_CMD_BURST_SEND_CONT); //envia
    while(I2CMasterBusy(I2C2_BASE));
  }

  I2CMasterControl(I2C2_BASE, I2C_MASTER_CMD_BURST_SEND_FINISH); //stop
  while(I2CMasterBusy(I2C2_BASE));
}
unsigned char GetDataRTC(unsigned char end)   {   //Pega os dados
  unsigned char dado;
  I2CMasterSlaveAddrSet(I2C2_BASE, SLAVE_ADDR, false);  //R/S=0 (Send)
  I2CMasterDataPut(I2C2_BASE, end);  //end. Memória RAM do RTC
  I2CMasterControl(I2C2_BASE, I2C_MASTER_CMD_SINGLE_SEND);
  while(I2CMasterBusy(I2C2_BASE));
  I2CMasterSlaveAddrSet(I2C2_BASE, SLAVE_ADDR, true);  //R/S=1 (Read)
  I2CMasterControl(I2C2_BASE, I2C_MASTER_CMD_SINGLE_RECEIVE); //sel. recepção
  while(I2CMasterBusy(I2C2_BASE));
  dado=I2CMasterDataGet(I2C2_BASE);  //leitura
  return bcd2dec(dado);
}
```