UNIVERSIDADE DE PASSO FUNDO

Vitor Augusto Spagnol

SISTEMA DE MONITORAMENTO DE PRODUÇÃO DE LEITE

Passo Fundo

2019

Vitor Augusto Spagnol

SISTEMA DE MONITORAMENTO DE PRODUÇÃO DE LEITE

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

2019

Vitor Augusto Spagnol

Sistema de Monitoramento de Produção de Leite

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. Orientador Dr. Adriano Luís Toazza - UPF

_____

Prof. Dr. Eduardo Appel - UPF

_____

Prof. Me. Amauri Fagundes Balotin – UPF

Dedicado a Leocadio e Rosicler Spagnol.

RESUMO

O Brasil é um dos maiores produtores de leite do mundo, mas, apesar disso, a cadeia produtiva brasileira não pode ser considerada especializada, com a maioria dos produtores sendo considerados pequenos, não fazendo uso de tecnologias e apresentando uma produtividade muito baixa quando comparada aos demais países líderes em produção. Atualmente, a grande parte da tecnologia oferecida para a área é voltada para o grande produtor, tendo muitas funcionalidades e um custo elevado. Com este trabalho busca-se alcançar um sistema para monitoramento da produção de leite simples, barato e eficaz, voltado ao pequeno e médio produtor. O sistema adquire o tempo de ordenha, a temperatura e o volume de leite ordenhados, a partir de um real time clock, um sensor de temperatura e um sensor de fluxo posicionados na ordenhadeira. Estes dados são então enviados através de uma rede Wi-Fi para um banco de dados, onde, posteriormente, podem ser acessados através de uma aplicação web. A aplicação permite que o produtor faça o acompanhamento e monitoramento de sua produção, apresentando o desempenho de cada animal ordenhado.

Palavras-Chave: produção de leite; sistemas de produção; sistemas de monitoramento; monitoramento de produção; sensor de temperatura; sensor de fluxo; aplicação web.

ABSTRACT

Brazil is one of the biggest milk producers in the world, but, even thou, Brazilian productive chain cannot be considered specialized, as it is composed mostly of small producers, without the use of technology, and presenting a very low productivity when compared to the other leaders of production. Currently, most of the technology that is offered to this area is aimed at the big producers, with a lot of functionalities and a very high implementation cost. This article is aimed to accomplish a system to monitor the milk production that is simple, effective and cheap, aimed at the small and medium producers. The system acquires the milking time, the temperature and the milk volume, using a real time clock, a temperature sensor and a flow sensor positioned at the milking machine. The acquired data is then sent to a database using a Wi-Fi network, where, afterwards, they can be accessed through a web application. The application allows the producer to monitor and follow his production, showing the performance of every animal that was milked.

Keywords: milk production; systems of production; monitoring systems; production monitoring; temperature sensor; flow sensor; web application.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS

API – Application Programming Interface

CSS – Cascading Style Sheets

HTML – Hypertext Markup Language

HTTP – Hyper Text Transfer Protocol

IBGE – Instituto Brasileiro de Geografia e Estatística

IDF – International Dairy Federation

IHM – Interface Homem Máquina

IN – Instrução Normativa

FAO – Food and Agriculture Organization of the United Nations

JS – JavaScript

Mapa – Ministério da Agricultura, Pecuária e Abastecimento

NPM – Node Package Manager

PCI – Placa de Circuito Impresso

PPM – Pesquisa Pecuária Municipal

REST – Representational State Transfer

RTC – Real Time Clock

SUMÁRIO

1 INTRODUÇÃO

Com mais de 790 milhões de toneladas consumidas anualmente, o leite (e seus derivados) é o alimento mais consumido no mundo. Movimentando mais de 320 bilhões de dólares em 2013, foi o terceiro produto agrícola mais produzido em termos de peso e alcançou o primeiro lugar entre os produtos agrícolas em termos de valor. (INTERNATIONAL DAIRY FEDERATION, 2016; FOOD AND AGRICULTURE ORGANIZATION OF THE UNITED NATIONS, 2016; SUPER INTERESSANTE, 2011).

Em 2016, o Brasil foi responsável por 7% da produção mundial de leite, posicionando o país em quinto lugar no ranking de produção, em termos de volume. Estes números fazem com que o leite seja considerado um dos seis produtos mais importantes da agropecuária brasileira, estando a frente de produtos tradicionais, como café e arroz. (BARBOSA et al., 2002; OLIVEIRA NETO, 2018).

De acordo com os dados oficiais de produção brasileira de leite de 2017, divulgados pelo Instituto Brasileiro de Geografia e Estatística (IBGE), através da Pesquisa Pecuária Municipal (PPM), foram produzidos 33,5 bilhões de litros de leite no ano. Aqui, dá-se destaque a região sul, que foi responsável por 35,7% de toda a produção nacional. (MILKPOINT, 2018).

Segundo dados da Estatística da Produção Pecuária de 2018, disponibilizado pelo IBGE, cerca de 65% das propriedades produtoras comercializavam entre 10 e 200 litros de leite diariamente. Este volume de produção caracteriza estes estabelecimentos como pequenos e médios. Esta análise do extrato de produção revela a grande importância dos estabelecimentos de pequena e média escala na produção nacional de leite. (MAIA et al., 2017).

Apesar de ser um dos grandes produtores mundiais de leite, quando observada a partir do início da cadeia produtiva, a produção brasileira não pode ser considerada como especializada, devido a uma pecuária considerada como extrativista, com baixo emprego de tecnologia e baixa produtividade. (DESENVOLVIMENTO REGIONAL SUSTENTÁVEL, 2010).

Para obter o volume produzido no ano de 2017, 17 milhões de vacas foram ordenhadas, resultando em um volume médio de 1.963 litros/vaca, valor muito abaixo dos alcançados por países onde a produção é considerada especializada, como Coreia do Sul e Estados Unidos da América (EUA), onde a produção anual supera os 10.000 litros/vaca. (MILKPOINT, 2018; OLIVEIRA NETO, 2018).

A comparação de produtividade entre o Brasil e os principais países produtores mostra que há espaço e necessidade da aprimoração dos processos produtivos, e, conforme indicado por Barbosa et al. (2002, p.1) “É seguro afirmar que os ganhos de produtividade advêm, basicamente, da adoção de tecnologias que melhoram a eficiência do uso dos fatores de produção”.

Muitas vezes, devido à falta de conhecimento, o produtor supõe que uma melhor qualidade do leite possa ser atingida apenas com altos investimentos em tecnologia, no entanto, é possível alcançar uma produção de qualidade superior com tecnologias simples, disponíveis a um baixo custo. Esta melhoria na qualidade pode ser observada em uma minoria das propriedades rurais, que fazem uso de tecnologia, apresentando índices de produtividade muito superiores à média nacional. (MAIA et al., 2017).

Segundo Mcgee (2004), sendo o leite um produto altamente perecível, há a necessidade de que o mesmo seja mantido sob monitoramento durante todo o processo produtivo, desde a ordenha, até a sua chegada ao consumidor. Através deste acompanhamento, é possível assegurar um leite de melhor qualidade, resultando em um pagamento diferenciado ao produtor. (MAIA et al., 2017).

1.1 OBJETIVO GERAL

Este trabalho tem como objetivo o desenvolvimento de um sistema para monitoramento e acompanhamento da produção de leite.

O sistema deverá contar com sensores, para fazer a aquisição de dados sobre a produção, como volume e temperatura do leite. Estes dados devem ser adquiridos para cada vaca que passa pelo processo de ordenha.

Com os dados coletados, pretende-se então fazer o uso de um software que permita ao produtor a visualização e acompanhamento de sua produção.

1.2 OBJETIVOS ESPECÍFICOS

a) Realização de pesquisa sobre o cenário nacional e tecnologias disponíveis;

b) Desenvolver um sistema para aquisição de dados de volume e temperatura do leite;

c) Desenvolver um sistema de servidor para armazenar os dados coletados;

d) Desenvolver um software para acesso e acompanhamento dos dados coletados;

e) Criação de protótipo para simulação de um ambiente real;

f) Aquisição de dados e demonstração de resultados;

1.3 JUSTIFICATIVA

O presente trabalho busca trazer ao pequeno e médio produtor uma tecnologia acessível, necessária para a melhoria da produção, tecnologia esta que está em falta na maior parte do sistema produtivo de leite brasileiro. Através de um sistema de monitoramento simples, mas eficiente, das principais variáveis, que permitirá ao produtor identificar falhas, e, a partir disso, buscar corrigi-las, visando o aumento da produtividade, e uma melhor qualidade do leite.

2 REVISÃO DA LITERATURA

Neste capítulo são apresentados os temas principais sobre o assunto estudado, os quais servem de base para a compreensão e desenvolvimento do projeto.

2.1 PRODUÇÃO DE LEITE

O leite é um dos produtos agrícolas mais produzidos e mais valiosos em todo o mundo. Em 2013, com uma produção total de 770 bilhões de litros, e movimentando 328 bilhões de dólares, o leite alcançou o primeiro lugar no ranking mundial de produtos agrícolas, em termos de valor.

Em praticamente todos os países do mundo há o consumo e a produção de leite, e, na maioria deles, este alimento se encontra entre os cinco produtos agrícolas mais importantes, tanto em termos de quantidade, quanto em termos de valor.

A produção mundial de leite tem projeção de aumentar em 177 milhões de toneladas até 2025, com uma taxa de crescimento de 1,8% ao ano. (FOOD AND AGRICULTURE ORGANIZATION OF THE UNITED NATIONS, 2016).

2.1.1 Cenário Mundial

Os principais produtores de leite bovino do mundo são EUA, Índia, China, Rússia, Alemanha, Brasil e Nova Zelândia, que, juntos, produzem 48% do leite mundial. (DESENVOLVIMENTO REGIONAL SUSTENTÁVEL, 2010). A Tabela 1 apresenta a evolução da produção mundial de leite, entre os anos de 2008 a 2016. Os países estão ordenados segundo a média de sua produção no período.

A produtividade de leite é estimada em litros por vaca por ano. Essa informação é apresentada na Tabela 2.

Tabela 1 – Evolução da produção mundial de leite.

1.000 toneladas

País/ Bloco	2008	2009	2010	2011	2012	2013	2014	2015	2016	Média
Mundo	**436.031**	**433.025**	**440.252**	**452.003**	**462.848**	**466.543**	**484.289**	**492.989**	**493.910**	**462.432**
União Europeia	133.848	133.700	135.472	138.220	139.000	140.100	146.500	150.200	151.000	140.893
EUA	86.173	85.821	87.488	89.020	91.010	91.277	93.485	94.619	96.343	90.582
Índia	46.870	48.160	50.300	53.500	55.500	57.500	60.500	64.000	68.000	56.037
China	34.300	28.445	29.300	30.700	32.600	34.300	37.250	37.550	36.020	33.385
Brasil	27.585	29.085	30.715	32.096	32.304	34.255	35.124	34.610	33.625	32.156
Rússia	32.500	32.600	31.847	31.646	31.831	30.529	30.499	30.548	30.510	31.390
Nova Zelândia	15.580	16.983	17.173	18.965	20.567	20.200	21.893	21.587	21.224	19.352
México	10.907	10.866	11.033	11.046	11.274	11.294	11.464	11.736	11.956	11.286
Argentina	10.010	10.350	10.600	11.470	11.679	11.519	11.326	11.552	10.191	10.966
Ucrânia	11.524	11.370	10.977	10.804	11.080	11.189	11.152	10.584	10.375	11.006

Fonte: Adaptado de Oliveira Neto, 2018.

Tabela 2 - Produtividade mundial de leite.

Cabeça/ano

País/Bloco	2008	2009	2010	2011	2012	2013	2014	2015	2016	Média
Mundo	**3.361**	**3.340**	**3.399**	**3.456**	**3.478**	**3.455**	**3.520**	**3.528**	**3.512**	**3.450**
Coreia do Sul	10.234	10.144	10.162	9.885	10.100	10.160	10.644	11.010	10.670	10.334
EUA	9.252	9.326	9.590	9.677	9.853	9.896	10.099	10.159	10.328	9.798
Japão	9.260	9.328	9.302	9.284	9.386	9.409	9.488	9.839	9.832	9.459
Canadá	8.404	8.458	8.512	8.545	8.973	8.786	8.835	9.196	9.610	8.813
União Europeia	5.536	5.527	5.749	5.978	6.030	6.041	6.243	6.375	6.400	5.986
Argentina	4.656	4.929	5.048	5.335	5.326	5.485	6.203	6.468	5.925	5.486
Austrália	5.793	5.564	5.844	5.906	5.946	5.645	5.681	5.918	5.613	5.768
Bielorússia	4.267	4.530	4.584	4.398	4.581	4.371	4.397	4.595	4.722	4.494
Ucrânia	3.722	3.981	4.012	4.106	4.291	4.381	4.445	4.558	4.661	4.240
China	4.000	3.998	4.003	4.029	4.075	4.108	4.435	4.470	4.503	4.180
Nova Zelândia	3.710	3.694	3.669	3.938	4.105	4.036	4.230	4.270	4.249	3.989
Rússia	3.316	3.421	3.595	3.658	3.701	3.700	3.789	3.942	4.217	3.704
México	1.758	1.698	1.703	1.726	1.775	1.793	1.805	1.834	1.854	1.772
Brasil	1.278	1.296	1.340	1.382	1.417	1.492	1.525	1.639	1.709	1.453

Fonte: Adaptado de Oliveira Neto, 2018.

2.1.2 Cenário Nacional

Entre 2008 e 2016 o Brasil alcançou a quinta colocação em termos de volume, sendo responsável, em média, por 7% da produção mundial de leite. (OLIVEIRA NETO, 2018).

Em 2017, segundo dados divulgados pelo IBGE, através da Pesquisa Pecuária Municipal, foram produzidos 33,5 bilhões de litros de leite. Com uma melhoria na

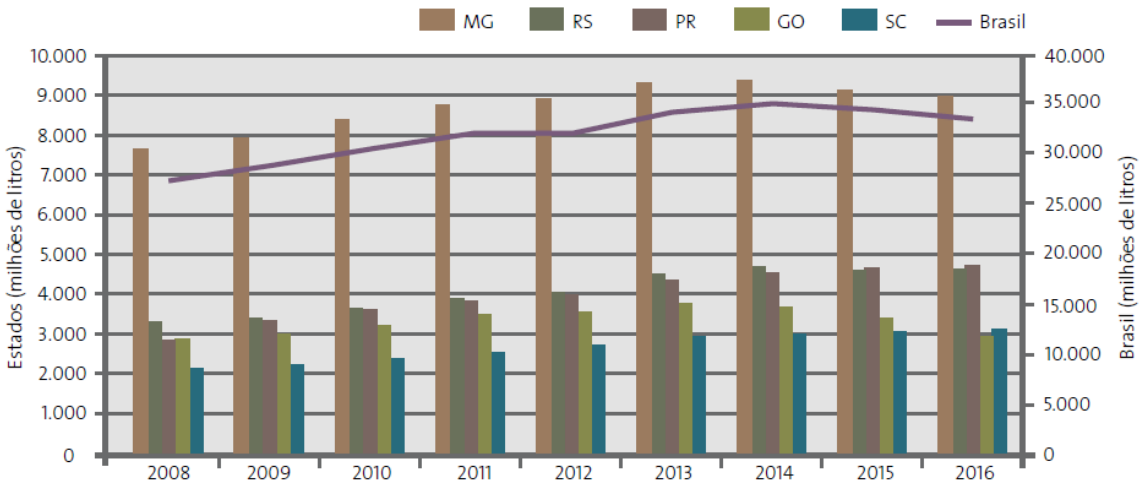
produtividade de quase 15% em relação a 2016, o volume médio de leite produzido por vaca foi de 1.963 litros em um ano, resultando em um total de 17 milhões de vacas ordenhadas. (MILKPOINT, 2018).

Mesmo sendo considerado um dos grandes produtores mundiais de leite, o sistema de produção no país é considerado de baixa rentabilidade para o produtor rural, não podendo ser considerado como especializado, devido a um baixo nível tecnológico e baixa produtividade. Estima-se que 90% dos produtores sejam caracterizados como pequenos, permanecendo com baixos índices de eficiência e baixos índices econômicos. (DESENVOLVIMENTO REGIONAL SUSTENTÁVEL, 2010).

A evolução da produtividade nos países produtores mostra que o Brasil possui condições de aprimorar o processo produtivo e alcançar melhores patamares de produtividade.

No Gráfico 1, tem-se a evolução da produção de leite no Brasil e nos principais estados produtores de 2008 até 2016.

Gráfico 1 – Evolução da produção leiteira no Brasil e nos principais estados produtores.



Fonte: Adaptado de Oliveira Neto, 2018.

Na Tabela 3 detalha-se a evolução da produção de leite no Brasil e nas principais Unidades da Federação.

A produtividade do rebanho, medida em litros por cabeça por ano, é apresentada na Tabela 4.

Tabela 3 – Evolução da produção de leite no Brasil e nos principais estados.

milhões de litros

UF	2008	2009	2010	2011	2012	2013	2014	2015	2016	Média
MG	7.657	7.931	8.388	8.756	8.906	9.309	9.370	9.145	8.971	8.715
RS	3.315	3.400	3.634	3.879	4.049	4.509	4.687	4.600	4.614	4.076
PR	2.828	3.339	3.596	3.816	3.969	4.347	4.541	4.660	4.730	3.981
GO	2.874	3.003	3.194	3.482	3.546	3.777	3.659	3.406	2.933	3.319
SC	2.126	2.218	2.381	2.531	2.718	2.918	2.983	3.060	3.114	2.672
SP	1.589	1.584	1.606	1.601	1.690	1.676	1.736	1.768	1.692	1.660
BA	952	1.182	1.239	1.181	1.079	1.163	1.212	984	858	1.095
RO	723	747	803	707	717	920	941	818	791	796
PE	726	788	877	953	609	562	657	856	839	763
MT	657	681	708	743	722	682	721	734	663	701
Brasil	**27.585**	**29.085**	**30.715**	**32.096**	**32.304**	**34.255**	**35.124**	**34.610**	**33.625**	**32.156**

Fonte: Adaptado de Oliveira Neto, 2018.

Tabela 4 – Evolução da produtividade do leite no Brasil.

litros/cabeça/ano

UF	2008	2009	2010	2011	2012	2013	2014	2015	2016	Média
RS	2.336	2.334	2.430	2.536	2.670	2.900	3.036	3.073	3.157	2.719
SC	2.362	2.375	2.432	2.478	2.521	2.577	2.694	2.755	2.788	2.553
PR	2.124	2.242	2.319	2.402	2.456	2.534	2.631	2.839	2.916	2.496
DF	2.231	1.722	1.769	1.538	2.117	1.415	1.485	1.577	1.591	1.716
AL	1.497	1.533	1.549	1.538	1.613	1.642	1.887	1.810	1.759	1.647
MG	1.489	1.502	1.540	1.555	1.570	1.591	1.613	1.686	1.803	1.594
PE	1.457	1.391	1.523	1.538	1.412	1.364	1.396	1.726	1.717	1.503
SE	1.307	1.320	1.343	1.392	1.320	1.414	1.466	1.648	1.636	1.427
GO	1.216	1.230	1.288	1.331	1.317	1.387	1.387	1.352	1.311	1.313
SP	1.114	1.110	1.079	1.102	1.150	1.205	1.370	1.427	1.463	1.224
Brasil	**1.278**	**1.296**	**1.340**	**1.382**	**1.417**	**1.492**	**1.525**	**1.639**	**1.709**	**1.453**

Fonte: Adaptado de Oliveira Neto, 2018.

Observando-se os custos operacionais para os produtores de leite em algumas cidades brasileiras, e relacionando estes custos com as receitas recebidas pelos mesmos, os resultados da Tabela 5 são apresentados. Nesta tabela, se o resultado da divisão for maior do que a unidade, isso significa que as despesas operacionais são maiores do que a receita bruta, ou seja, caracteriza prejuízo para o produtor.

Tabela 5 – Indicador custo operacional / receita

Local	2014	2015	2016	2017
Ouro Preto do Oeste - RO	1,19	1,32	1,51	1,50
Pau dos Ferros - RN	-	1,76	1,70	1,63
Morada Nova - CE	-	1,67	1,56	1,67
Orizona - GO	1,20	1,30	1,06	1,18
São Miguel do Oeste - SC	1,10	1,09	**0,94**	1,07
Teotonia - RS	1,24	1,40	1,23	1,50
Ijuí - RS	**0,42**	**0,38**	**0,29**	1,22
Passo Fundo - RS	1,45	1,44	1,19	1,41
Guaratinguetá - SP	1,25	1,32	1,20	1,23
Mococa - SP	1,15	1,22	1,13	1,11
Unaí - MG	2,05	2,16	1,93	2,09
Patos de Minas - MG	1,27	1,45	1,34	1,47
Ibiá - MG	1,33	1,38	1,14	1,23
Pompéu - MG	1,10	1,32	1,12	1,16

Fonte: Adaptado de Oliveira Neto, 2018.

Observando os indicadores analisados no âmbito dos custos operacionais, pode-se perceber que todos estão próximos ou superiores à unidade. Tal situação demonstra a necessidade de melhoria no processo de gestão da unidade produtiva. (OLIVEIRA NETO, 2018).

Considerando a produção primária como elo mais fragilizado da cadeia produtiva do leite, em que ações específicas podem gerar transformações positivas, com resultados expressivos, este é o elo que merece maior atenção e investimento. (DESENVOLVIMENTO REGIONAL SUSTENTÁVEL, 2010).

2.2 NORMAS PARA PRODUÇÃO DE LEITE

As normas tem como objetivo regulamentar a produção de leite, garantindo a qualidade do mesmo, com foco nas boas práticas agropecuárias.

O regramento faz com que o produtor precise intensificar o controle na obtenção do leite, aplicando ferramentas de gestão na propriedade, ferramentas de manejo sanitário, refrigeração e estocagem.

A aplicação das normas permite o avanço nos índices de qualidade, aumento da produtividade e uma oferta de alimentos mais seguros a população. (MINISTÉRIO DA AGRICULTURA, 2018).

No que diz respeito a obtenção do leite, definindo padrões de identidade e qualidade, o Ministério da Agricultura, Pecuária e Abastecimento (Mapa) fixou novas regras para a produção de leite no país, as Instruções Normativas (IN) 76, 77 e 78.

A IN 76 trata das características e da qualidade do produto na indústria. Na IN 77, são definidos critérios para obtenção de leite de qualidade e seguro ao consumidor e que englobam desde a organização da propriedade até a formação e capacitação dos responsáveis pelas tarefas cotidianas, o controle sistemático de mastites, da brucelose e da tuberculose. Na IN 78, são definidos os critérios a serem seguidos nas provas de produção. (MINISTÉRIO DA AGRICULTURA, 2018).

De acordo com o Diário Oficial da União, estas normativas passam a entrar em vigor a partir de 30 de maio de 2019, substituindo as normativas 51/2002, 22/2009, 62/2011, 07/2016 e 31/2018.

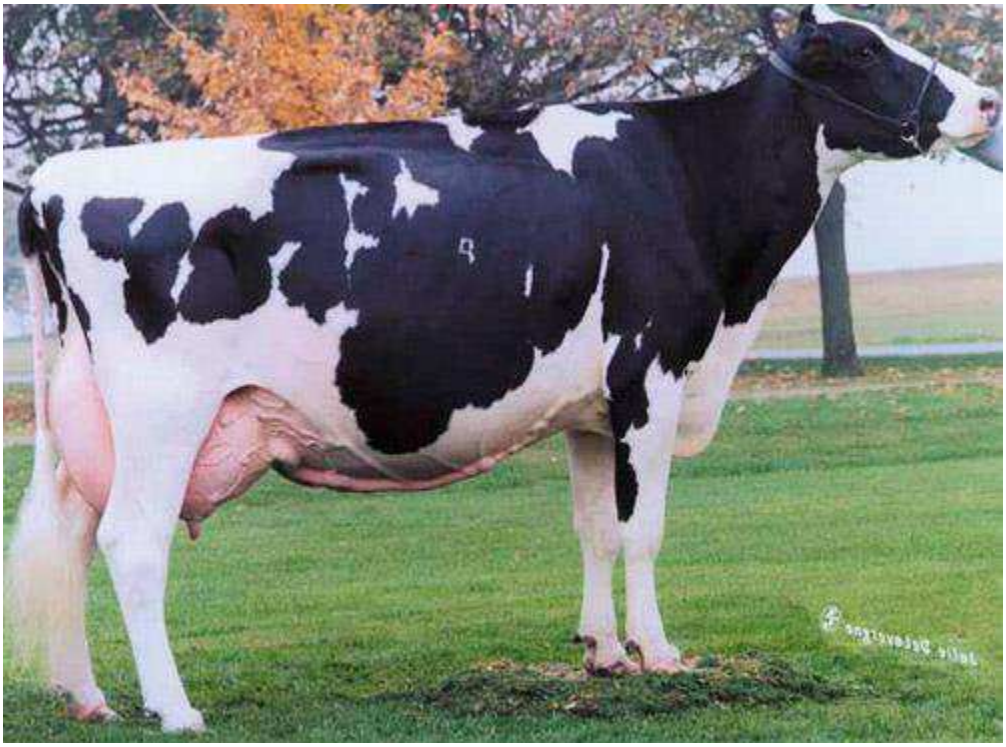
As condições impostas pelas novas instruções são mais exigentes e rigorosas, alinhando nossa legislação a critérios praticados nos países europeus. (FAGNANI, 2019).

No que diz respeito aos equipamentos e instrumentos que entrarão em contato com o alimento, é estabelecido que estes devem ser preferencialmente em aço inoxidável. Outro material utilizado não deve transmitir substâncias tóxicas, odores e sabores, não serem absorventes, serem resistentes à corrosão e capazes de resistir à repetidas operações de limpeza e desinfecção. (CODEX ALIMENTARIUS OFFICIAL STANDARDS, 1969; INSTITUTO MINEIRO DE AGROPECUÁRIA, s. d.).

2.3 CARACTERÍSTICAS DAS VACAS PRODUTORAS DE LEITE

A raça mais conhecida e difundida é a Holstein-Frísia, popularmente conhecida como Holandesa, raça europeia pura, especialmente selecionada para a produção de leite. Esta raça é demonstrada na Figura 1. No que diz respeito ao seu traço mais importante, a produção de leite, ela lidera os mais diversos rankings, podendo atingir mais de 50 litros de leite em um mesmo dia, em duas ordenhas, sendo que seu leite apresenta pouca gordura. (CANAL RURAL, 2012; PROCREARE, 2016).

Figura 1 – Vaca da raça Holstein-Frísia (Holandesa).



Fonte: Rural Pecuária, 2019.

A medida padrão de produção de leite mais usada nas avaliações genéticas de vacas é a produção total de leite da mesma após 305 dias de lactação, sendo conhecida como medida P305. (VARGAS et al., 2006).

Dados de controle registrados em algumas propriedades indicam que os rebanhos brasileiros podem apresentar um nível produtivo de 12 litros de leite por animal por dia, mas, este número é maior do que a média de produção anual brasileira, previamente apresentada na Tabela 4. (FAÇANHA et al., 2010).

O leite, quando retirado, apresenta temperatura média de 37 °C. Esta temperatura é também a temperatura corporal do animal, normalmente. (GOFF, 2019).

Quanto ao tempo de duração da ordenha, uma meta a ser obtida é ordenhar os primeiros 12,5 litros de leite em 4 minutos de ordenha. Para cada 4,5 litros de leite adicional, o tempo adicional necessário deve ser de 0,5 minutos. Esses índices não são normalmente atingidos em vacas de baixa produção. (SANTOS, 2007).

Durante o tempo de ordenha, o fluxo de leite do úbere não é constante. Depois que a unidade é colocada, o fluxo de leite aumenta e atinge o pico após 30 a 60 segundos, permanecendo nos próximos dois minutos e diminuindo gradualmente.

O tempo de ordenha geralmente aumenta para vacas de maior produção enquanto que os fluxos de leite são similares para a maioria dos tetos em qualquer raça de vaca. (DELAVAL, 2018).

O fluxos médios de leite recomendados são ordenhar cerca de 3,9 litros/minuto, para vacas ordenhadas duas vezes por dia e 3 litros/minuto para vacas em três ordenhas/dia. (SANTOS, 2007).

2.4 ORDENHADEIRAS MECÂNICAS

Considera-se ordenha, o ato de realizar a extração do leite da glândula mamária, podendo ser feita de forma manual quando realizada pelo ordenhador e mecânica quando for utilizada ordenhadeira. É uma prática que deve ser efetuada com cuidados, para garantir a qualidade do produto e o bem-estar do animal. (NETTO; BRITO; FIGUEIRÓ, 2006).

A ordenha mecânica é a principal máquina em um sistema de produção de leite. Seu funcionamento se baseia na geração de vácuo ou pressão negativa em volta do teto, o que produz um efeito de sucção que vence o seu fechamento, permitindo a saída do leite. (RIBEIRO; CARVALHO, 2019).

A ordenha mecanizada possibilita a extração do leite mais rápida do que a ordenha manual e, quando realizada da forma correta, apresenta menor risco de contaminação do produto. Geralmente, é feita em um local específico, a sala de ordenha. (ROSA et al., 2009).

As ordenhadeiras mecânicas podem ser divididas, basicamente, em dois tipos, ordenhadeira balde ao pé e ordenhadeira em linha.

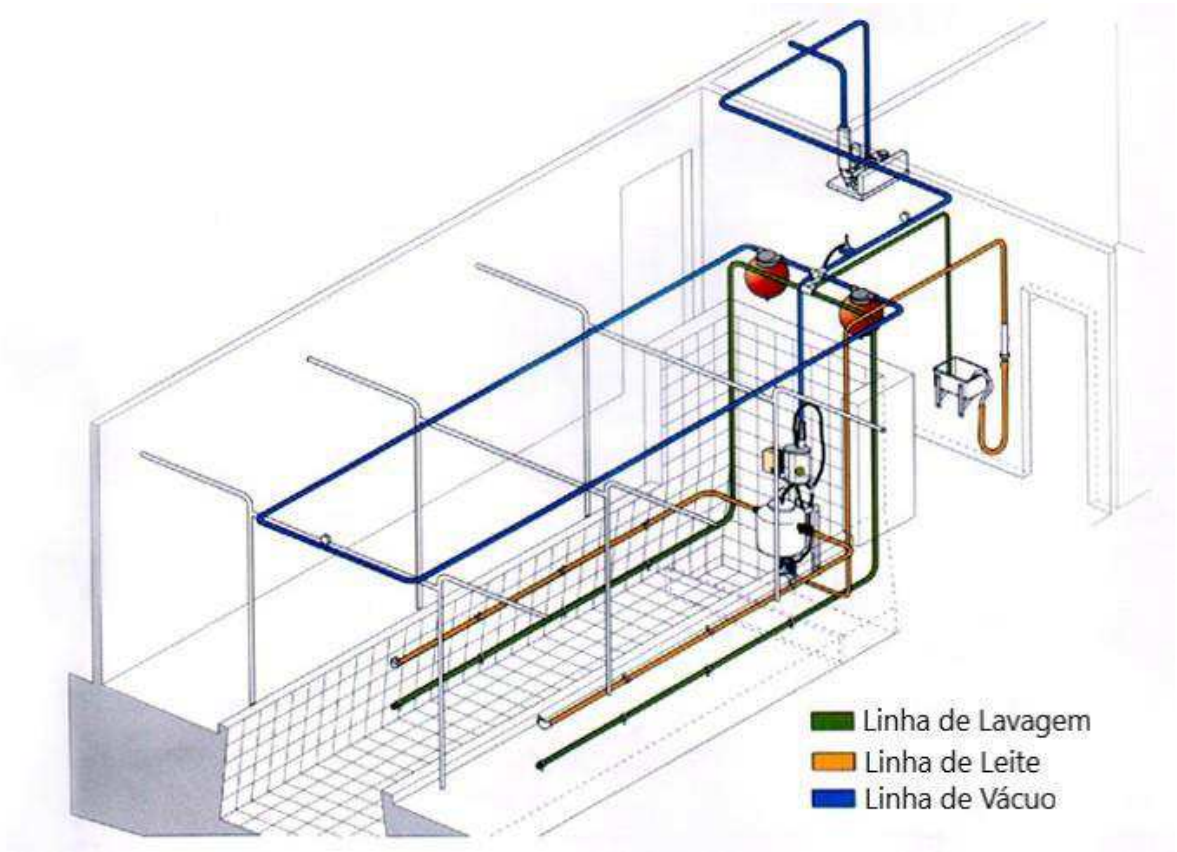
Na ordenhadeira balde ao pé os animais são ordenhados individualmente, através de um sistema de vácuo. As vacas podem ser ordenhadas no estábulo ou na sala de ordenha, possui baixa eficiência, mas o seu custo de implantação é relativamente barato. A ordenhadeira balde ao pé é representada na Figura 2. (INFOESCOLA, 2019).

A ordenhadeira em linha é um sistema mais intensivo de ordenha, onde várias vacas podem ser ordenhadas simultaneamente. Existe uma linha de vácuo que fará a ordenha das vacas e depositará o leite na linha de leite. Este sistema é mais caro, quando comparado ao sistema de balde ao pé, porém apresenta maior eficiência. Esta ordenhadeira é apresentada na Figura 3. Nesta imagem, pode-se dar destaque as linhas de vácuo e de leite. (INFOESCOLA, 2019).

Figura 2 – Representação de ordenhadeira balde ao pé.

Fonte: ORDEMAX, 2019.

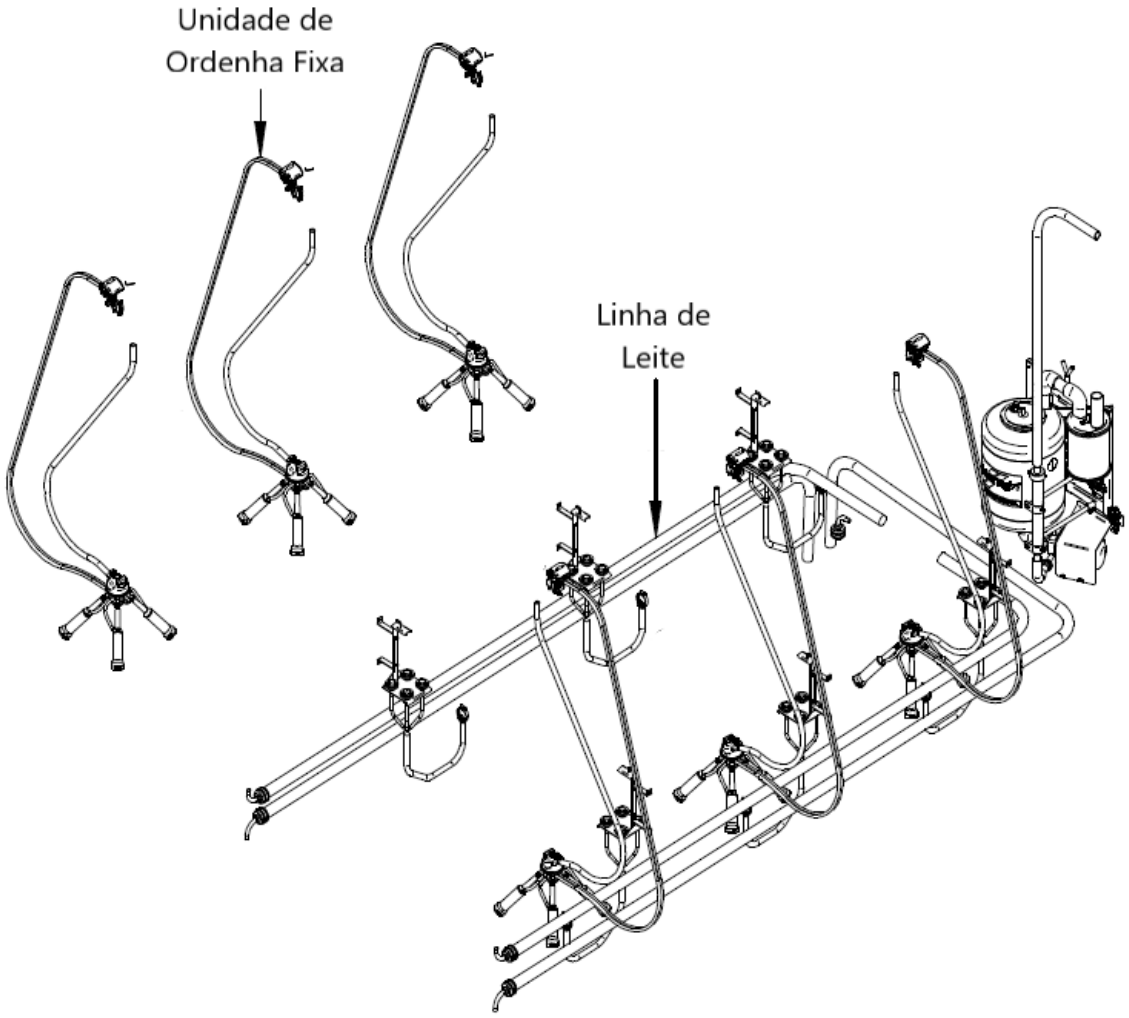
Figura 3 – Representação de ordenhadeira em linha.



Fonte: Adaptado de INCOMAGRI, 2019.

A Figura 4 apresenta uma vista parcialmente explodida de uma ordenhadeira em linha, onde pode-se dar destaque a unidade de ordenha.

Figura 4 – Vista explodida de ordenhadeira em linha.



Fonte: Adaptado de INCOMAGRI, 2019.

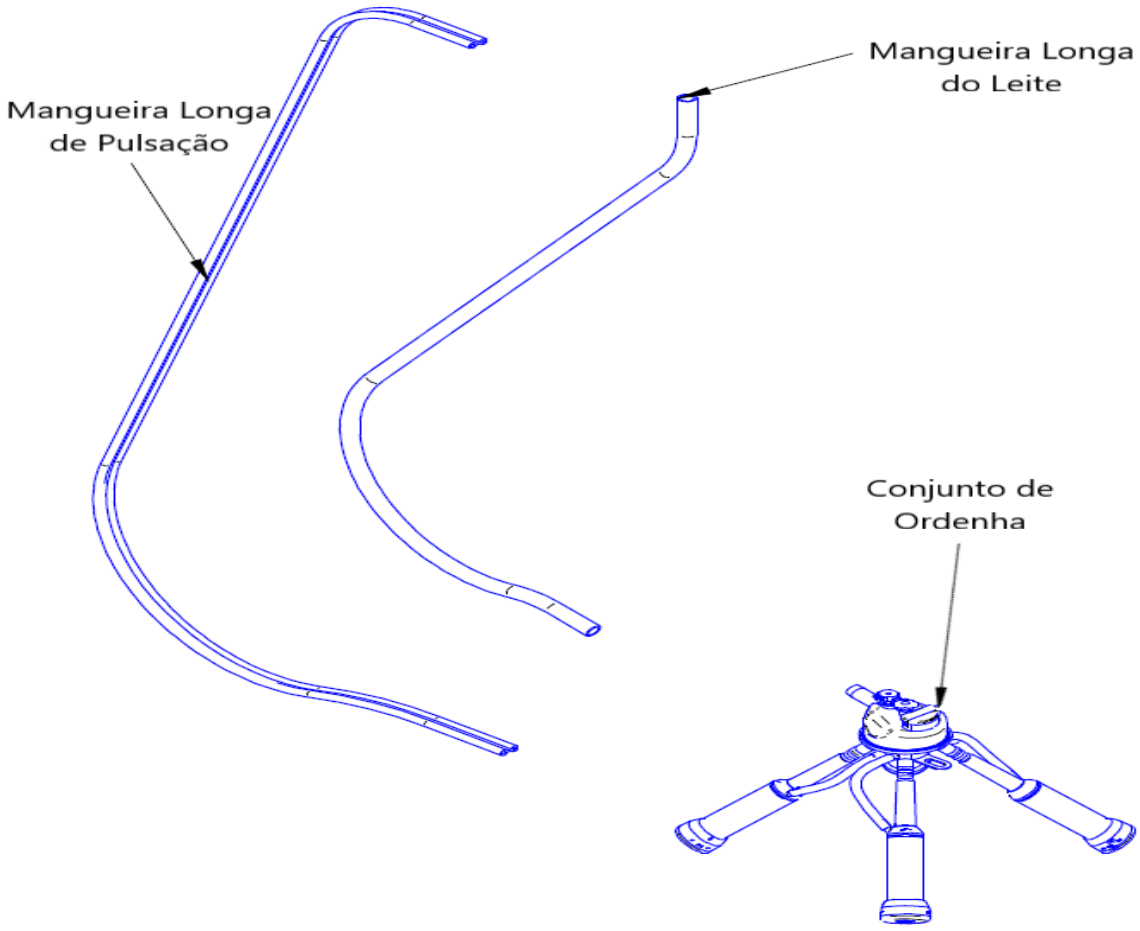
As ordenhadeiras podem ser divididas em três sistemas, de acordo com sua função, sejam eles, os equipamentos e componentes de vácuo, os equipamentos e componentes de pulsação e os equipamentos e componentes de ordenha. (RIBEIRO; CARVALHO, 2019).

Os equipamentos e componentes de vácuo são responsáveis por extrair o ar do sistema de ordenha, comprimindo e expulsando-o por um escapamento. É necessário manter o nível de vácuo constante durante toda a ordenha, assim, este sistema também é responsável por fazer a regulação do vácuo.

Os equipamentos e componentes de pulsação são essenciais para a ordenha, pois o mesmo faz a alternância durante a ordenha, entre extração de leite ou massageamento. A relação de pulsação é determinada pelo tempo de duração da fase de extração e de massageamento, normalmente tendo uma relação 60:40, onde 60% do tempo é feita a extração do leite, e 40% do tempo é o massageamento do teto. Os pulsadores são ativados para gerar uma frequência entre 58 e 62 ciclos por minuto (geralmente as taxas de pulsação são reguladas em 60 ciclos por minuto). (RIBEIRO; CARVALHO, 2019).

Os equipamentos e componentes de ordenha são compostos do conjunto de ordenha (quatro copos de teteiras unidos por uma mangueira curta de vácuo, teteira e coletor), mangueira do leite, mangueira longa de pulsação, podendo também apresentar componentes extras de acordo com o fabricante. Estes componentes são responsáveis por fazer a coleta do leite, e posteriormente o transporte do mesmo, através da mangueira de leite, até a linha de leite. O conjunto de ordenha é apresentado na Figura 5. (GRUPO AGUIAR, 2019).

Figura 5 – Componentes do conjunto de ordenha.



Fonte: Adaptado de INCOMAGRI, 2019.

2.5 SISTEMAS DE MONITORAMENTO DISPONÍVEIS NO MERCADO

Nesta seção são apresentados os principais sistemas para monitoramento de produção de leite e gestão de animais disponíveis no mercado atualmente, apresentando suas principais características e aplicabilidades.

2.5.1 Sistema de Gerenciamento de Fazendas Leiteiras SysMax – Ordemax

O sistema de gerenciamento para fazendas leiteiras SysMax, desenvolvido e comercializado pela empresa Ordemax, é um sistema com o objetivo de organizar e automatizar a propriedade produtora de leite. De acordo com o catálogo de divulgação da empresa, o sistema é de fácil operação, sendo capaz de proporcionar o acompanhamento e o gerenciamento da propriedade. (ORDEMAX, 2019).

O sistema é composto por um medidor eletrônico do volume de leite, um extrator automático do conjunto de ordenha, um software para gestão do rebanho e um aplicativo para smartphone onde é possível fazer o acompanhamento da produção. (ORDEMAX, 2019).

A medição do volume de leite é feita através de sensores posicionados dentro de um copo medidor, apresentado na Figura 6. Este copo medidor é instalado entre a mangueira de leite e a linha de leite, assim todo o volume ordenhado passa pelo copo. Os dados coletados são enviados a um painel eletrônico. (ORDEMAX, 2019).

Figura 6 – Copo medidor utilizado no sistema Ordemax.

Fonte: Adaptado de ORDEMAX, 2019.

O painel eletrônico, apresentado na Figura 7 conta com um teclado, onde o usuário identifica qual vaca será ordenhada. No painel são gravadas as últimas 4500 ordenhas. Ele também é responsável por fazer a extração automática do conjunto de ordenha, quando a mesma é finalizada, e, além disso, todo o sistema de pulsação da ordenhadeira é controlado por este painel. (ORDEMAX, 2019).

Figura 7 – Painel eletrônico do sistema Ordemax.

Fonte: Adaptado de ORDEMAX, 2019.

Devido as várias funções desempenhadas pelo mesmo sistema de aquisição do volume de leite, o mesmo não é recomendado apenas para o monitoramento da produção. O custo de implementação para um sistema como este é de cerca de R$5.000,00 por posto de ordenha[1]. Além disso, há o custo de implementação do software, que para sistemas como este fica em torno de R$35.000,00[1] (informação verbal).

Os dados coletados sobre a ordenha não são enviados automaticamente para o software de gestão, sendo necessário que o usuário faça a leitura dos dados armazenados no painel eletrônico, e o lançamento dos mesmos no software.

O software de gerenciamento, segundo o catálogo da empresa, oferece ferramentas completas para a gestão da propriedade, proporcionando o gerenciamento de custos, controle

[1] Orçamento obtido na Agropecuária Frumi, em maio de 2019.

dos animais cadastrados, controle da produção do rebanho, gerenciamento sanitário, além do gerenciamento de índices zootécnicos. (ORDEMAX, 2019).

O sistema também conta com um aplicativo para smartphone, através do qual é possível acessar os dados do software de gestão, e acompanhar relatórios sobre a propriedade. (ORDEMAX, 2019).

2.5.2 Afimilk – Weizur

O sistema Afimilk é comercializado pela empresa Weizur. Este sistema busca a total automação e gerenciamento da propriedade produtora. O Afimilk MPC é um medidor de volume de leite, que é usado integrado ao sistema de gestão Afifarm.

O módulo Afimilk MPC, Figura 8, é um painel para medição de extração do leite. Este módulo coleta as informações sobre o volume, taxa de fluxo, temperatura do leite e tempo de ordenha, enviando alertas ao operador ao detectar temperaturas anormais. O mesmo também conta com sensores para identificar doenças, mais especificamente a mastite. (AFIMILK, 2019).

Um único módulo, usado para o controle de uma unidade de ordenha, pode ser encontrado em lojas on-line por preços em torno de U$ 1.500,00. (EBAY, 2019).

Figura 8 – Módulo Afimilk MPC.

Fonte: Weizur, 2019.

Assim como no sistema disponibilizado pela empresa Ordemax, o módulo Afimilk MPC também é responsável por fazer o controle do processo de pulsação, possibilitando um controle avançado de acordo com o fluxo de leite atual.

Os dados coletados por esse módulo são automaticamente enviados para o sistema de gestão Afifarm.

O software de gestão Afifarm, de acordo com o site da fabricante, é um software flexível e amigável para automação e gerenciamento de fazendas leiteiras. Este software possui integração com vários módulos disponibilizados pela empresa, dentre eles o módulo Afimilk MPC. (AFIMILK, 2019).

Através deste software é possível fazer o gerenciamento do rebanho, da saúde dos animais, análise da produção de leite, eficiência da ordenha, além do monitoramento de aspectos zootécnicos do rebanho. De acordo com o site do produto, o software é capaz de gerenciar rebanhos com mais de 2.000 animais. (AFIMILK, 2019).

Além disso, o software conta com uma aplicação para smartphone, onde o usuário pode acessar os mesmos dados, acompanhar o desempenho e verificar relatórios sobre a propriedade. Este aplicativo também tem a função de enviar avisos ao proprietário, informando sobre eventuais doenças ou anomalias na produção. (AFIMILK, 2019).

O custo para implementação deste sistema varia de acordo com o tamanho da fazenda, podendo ultrapassar U$80.000,00 para uma propriedade com 500 animais. (CARIA; TODDE; PAZZONA, 2019).

Devido as características deste sistema, pode-se perceber que o mesmo tem como público alvo os grandes produtores, que tem condições de realizar um elevado investimento para a automação de suas propriedades.

2.5.3 DelPro – DeLaval

A empresa DeLaval oferece aos produtores o sistema DelPro, desenvolvido para o gerenciamento da propriedade, e ajuda na tomada de decisões através de dados coletados.

Para a medição do volume de leite ordenhado, a empresa dispõe de duas soluções. O indicador de fluxo DeLaval Fi7 é mostrado na Figura 9. Este indicador monitora a produção de leite de cada vaca, e faz a indicação do mesmo em seu display. (DELAVAL, 2019). Este valor medido é indicado no display, para conferência do operador, mas não é gravado nem enviado para qualquer software onde possa ser visualizado futuramente.

Além deste indicador de fluxo, a empresa também oferece o medidor de leite MM27 BC, apresentado na Figura 10.

Figura 9 – Medidor de fluxo DeLaval Fi7.

Fonte: DELAVAL, 2019.

Figura 10 – Medidor de Leite DeLaval MM27 BC.

Fonte: DELAVAL,2019.

Este medidor permite o acompanhamento da produção, apresentando o volume de leite produzido por cada vaca ordenhada. Através desses dados é possível avaliar o desempenho da

sala de ordenha e a eficiência do rebanho. Os dados deste medidor podem ser enviados para o software DelPro, permitindo o acompanhamento e análise. Além disso, este módulo também conta com um sensor de condutividade e um indicador de entrada de ar na unidade de ordenha. (DELAVAL, 2019).

O software DelPro é voltado para a gestão total da fazenda, podendo trabalhar com até 20.000 animais. O software pode ser integrado a diversos módulos oferecidos pela DeLaval, recebendo dados dos mesmos, e permitindo a análise, monitoramento e gestão através de relatórios. Também há a possibilidade de acessar o software de forma remota, permitindo assim que toda a fazenda seja gerenciada sem a necessidade de o gestor encontrar-se presente na mesma. (DELAVAL, 2019).

O software DelPro também é acompanhado de um aplicativo móvel, de onde é possível acessar os registros do rebanho e fazer o monitoramento da produção.

Novamente, nota-se o foco do sistema nos grandes produtores de leite. Devido à grande complexidade do software de gerenciamento, torna-se inviável ao pequeno produtor a implantação apenas do monitoramento do volume de leite.

2.5.4 Dairy Plan C21 – GEA

Desenvolvido pela empresa GEA, o Dairy Plan C21 é um projeto modular, onde diversos módulos e sensores podem ser combinados e integrados a um único software, buscando a gestão completa da fazenda produtora de leite.

Para a análise do fluxo de leite, pode ser integrado ao sistema o módulo Metatron Premium 21, apresentado na Figura 11. Este é responsável por registrar os dados gerados durante a produção leiteira, como o peso de leite, e condutividade do mesmo. Os dados coletados são então enviados para o software Dairy Plan C21, onde são processados. O display gráfico contido no módulo permite que o mesmo faça a exibição dos dados coletados durante a ordenha. (GEA, 2019).

O valor para a aquisição de um módulo Metatron Premium 21, encontrado em lojas de vendas on-line, é de cerca de U$2.000,00. (HAMBY DAIRY SUPPLY, 2019).

Através do software disponível no sistema Dairy Plan C21, é possível operar, planejar e monitorar a produção de leite e a saúde das vacas, contando com dados coletados através dos módulos oferecidos pela empresa. (GEA, 2019).

O software permite o gerenciamento do rebanho e análise dos dados totais e individuais de produção. Também é possível fazer o controle dos sistemas de alimentação automática,

separação automática de animais e ordenha automática, nas fazendas onde estes sistemas se fazem presente. (GEA, 2019).

Figura 11 – Módulo Metatron Premium 21 da empresa GEA.

Fonte: GEA, 2019.

Assim como observado nos demais sistemas apresentados, este também tem como foco o produtor especializado, devido à complexidade de tarefas realizadas, e, consequentemente, um custo de implementação elevado.

2.6 CARACTERIZAÇÃO DO SISTEMA

A seguir, apresentam-se conceitos interessantes, com o intuito de facilitar a compreensão do projeto desenvolvido, que é apresentado nos capítulos posteriores.

2.6.1 Medição de Fluxo

A medição de fluxo pode ser caracterizada como a quantificação do movimento de um fluido. Esta medição pode ser realizada através de uma variedade de elementos. Os elementos de medição de fluxo são dispositivos que podem ser inseridos no meio fluido, sendo capazes de produzir uma grandeza que pode ser relacionada com o fluxo.

No Quadro 1 são apresentados, resumidamente, os principais métodos utilizados em medidores de fluxo.

Quadro 1 – Principais métodos utilizados em medidores de fluxo.

Método ou dispositivo utilizado	Sinal de entrada	Sinal de saída
Tubo de Pitot	Velocidade pontual ou local do fluido ou fluxo volumétrico	Pressão diferencial
Anemômetro (método do fio quente)	Velocidade pontual ou local do fluido	Temperatura
Eletromagnético	Velocidade média do fluido	Tensão elétrica
Ultrassom	Velocidade média do fluido	Tempo ou por frequência (Doppler)
Placa de orifício	Fluxo volumétrico	Pressão diferencial
Tudo de Venturi	Fluxo volumétrico	Pressão diferencial
Bocal	Fluxo volumétrico	Pressão diferencial
Turbina	Fluxo volumétrico	Ciclos ou revoluções
Deslocamento positivo	Fluxo volumétrico	Ciclos ou revoluções
Draga ou força de arrasto	Fluxo volumétrico	Força
Área variável (rotâmetro)	Fluxo volumétrico	Deslocamento do elemento flutuante
Vórtice	Fluxo volumétrico	Frequência
Efeito Coriolis	Massa média do fluxo	Força
Transporte térmico	Massa média do fluxo	Temperatura

Fonte: Adaptado de Balbinot e Brusamarello, 2011.

O Quadro 2 apresenta as características básicas que devem ser levadas em conta na escolha dos principais medidores de fluxo encontrados no mercado.

Quadro 2 – Características básicas para seleção de medidores de fluxo.

Medidor de fluxo	Recomendado principalmente para	Perda de pressão	Precisão típica (%)	Custo relativo
Placa de orifício	Líquidos limpos	Média	de ±2 a ±4 do fundo de escala	Baixo
Tubo de Venturi	Líquidos limpos, sujos e viscosos	Baixa	±1 do fundo de escala	Médio
Bocal	Líquidos limpos e sujos	Média	de ±1 a ±2 do fundo de escala	Médio
Tubo de Pitot	Líquidos limpos	Baixa	de ±3 a±5 do fundo de escala	Baixo
Área variável	Líquidos limpos, sujos e viscosos	Média	de ±1 a ±10 do fundo de escala	Baixo
Deslocamento positivo	Líquidos limpos e viscosos	Alta	±0,5	Médio
Turbina	Líquidos limpos e viscosos	Alta	±0,25	Alto
Vórtice	Líquidos limpos e sujos	Média	±1	Alto
Eletromagnéticos	Líquidos condutivos limpos e sujos	Nenhuma	±0,5	Alto
Ultrassônico (efeito Doppler)	Líquidos sujos e viscosos	Nenhuma	±5 do fundo de escala	Alto
Ultrassônico (tempo)	Líquidos limpos e viscosos	Nenhuma	de ±1 a ±5 do fundo de escala	Alto
Efeito Coriolis (massa)	Líquidos limpos, sujos e viscosos	Baixa	±0,4	Alto
Massa térmica	Líquidos limpos, viscosos e sujos	Baixa	±1 do fundo de escala	Alto

Fonte: Adaptado de Balbinot e Brusamarello, 2011.

2.6.1.1 Medidores Mecânicos tipo Turbina

Na medição de fluxos ou vazões de líquidos podem ser utilizadas técnicas diretas, baseadas no deslocamento de algum componente mecânico. Constituindo esta família de medidores que utilizam partes móveis, estão os compostos de engrenagens, rotores ou turbinas. (BALBINOT; BRUSAMARELLO, 2011).

O medidor de fluxo do tipo turbina foi inventado por Reinhard Woltman no século XVIII. Este medidor é considerado preciso e confiável para a medição do fluxo de líquidos e gases. (BALBINOT; BRUSAMARELLO, 2011).

Consiste em um rotor balanceado com diâmetro um pouco menor do que o apresentado pela tubulação. Sua velocidade de rotação é proporcional à razão de fluxo volumétrico e pode ser detectada por sensores de estado sólido (indutância, capacitância ou por efeito Hall) bem como por sensores mecânicos. (BALBINOT; BRUSAMARELLO, 2011).

A turbina mede o volume médio de fluxo $\left(\frac{m^3}{s}\right)$ de um fluido, onde a velocidade de rotação da turbina é descrita pela equação (1),

$$\frac{Q}{n \cdot D^3} = f \cdot \left(\frac{n \cdot D^2}{v}\right) \tag{1}$$

onde Q é o volume de fluxo $\left(\frac{m^3}{s}\right)$, n a velocidade do rotor $\left(\frac{rev}{s}\right)$, D o diâmetro e v a viscosidade cinemática $\left(\frac{m^2}{s}\right)$.

Normalmente, a turbina funciona dentro de uma faixa linear, tal que a velocidade da turbina é diretamente proporcional ao volume médio de fluxo. (BALBINOT; BRUSAMARELLO, 2011).

2.6.1.2 Sensor de Fluxo YF-S401

O sensor de fluxo YF-S401, apresentado na Figura 12, é um sensor do tipo turbina, desenvolvido e fabricado em conjunto com um sensor de efeito hall. Desta maneira, este sensor gera uma onda quadrada, cuja quantidade de pulsos pode ser medida para determinar o volume de fluido através do mesmo.

De acordo com o datasheet do sensor, o mesmo tem seu corpo construído em PVC, é compacto e de fácil instalação. É acompanhado de um sensor de efeito hall de alta qualidade, e a turbina é construída em aço inoxidável, sendo resistente à abrasão. (DFROBOT, 2019).

O sensor pode operar em pressões de até 8 bar, trabalhando na faixa de 0,3 a 6 litros por minuto, e apresentando precisão de ±5%, quando trabalhando com fluxos de até 3 litros por minuto. (DFROBOT, 2019).

Ainda segundo o datasheet, 5880 pulsos gerados pelo sensor hall equivalem a 1 litro de água percorrido pelo sensor.

Figura 12 – Sensor de fluxo YF-S401.

Fonte: USINA INFO ELETRÔNICA E ROBÓTICA, 2019

2.6.1.3 Sensor de fluxo Meister DMIK-7

O sensor de fluxo Meister DMIK-7, apresentado na Figura 13 é capaz de medir o fluxo operando pelo princípio de indução magnética, desta forma, este sensor não possui nenhuma parte móvel.

Este sensor possibilita a medição de fluxos entre 0,5 e 30 litros por minuto, tendo sua saída na forma de pulsos, sendo que 1000 pulsos representam 1 litro.

O sensor DMIK-7 tem sua construção em aço inoxidável, e possui uma precisão de ±1,5% do valor medido.

Figura 13 – Sensor de fluxo Meister DMIK-7.

Fonte: GRUNN, 2019.

2.6.2 Medição de Temperatura

O avanço tecnológico proporcionado pela eletrônica, possibilita nos dias de hoje, a confecção e utilização de uma ampla gama de sensores de temperatura, tornando estes os instrumentos mais empregados no campo da engenharia. Dentre os sensores disponíveis atualmente, é possível mencionar os termômetros bimetálicos, os termômetros de variação de resistência elétrica, os termopares e os sensores semicondutores de temperatura. (BALBINOT; BRUSAMARELLO, 2011).

2.6.2.1 Sensores semicondutores de temperatura

Dispositivos semicondutores, tais como os diodos e os transistores, são sensíveis à temperatura e, partindo deste princípio, podem ser utilizados no desenvolvimento de sensores de temperatura.

As principais vantagens na utilização desses dispositivos são a linearidade, a simplicidade e a boa sensibilidade, apresentando como principal desvantagem a sua faixa de temperatura, que normalmente pode chegar a no máximo 200 °C, dependendo do modelo do sensor, pois acima desta temperatura os dispositivos semicondutores podem acabar danificados. (BALBINOT; BRUSAMARELLO, 2011).

Também existem disponíveis no mercado, sensores semicondutores de temperatura com saída digital, os quais apresentam diversas vantagens, principalmente pela facilidade de utilização e interfaceamento com sistemas microcontrolados.

2.6.2.2 Sensor de Temperatura DS18B20

O sensor DS18B20, apresentado na Figura 14, é um sensor de temperatura digital, capaz de realizar medições, e devolver um valor digital, configurável entre 9 e 12 bits. (MAXIM INTEGRATED, 2019).

Figura 14 – Sensor de temperatura DS18B20.

Fonte: FILIPEFLOP, 2019.

Este sensor é implementado junto a um microcontrolador, se comunicando através do protocolo serial 1-Wire, desta forma, é necessário apenas um fio para que o microcontrolador faça a leitura do sensor.

É capaz de realizar medidas de temperatura de -55 °C até +125 °C, com precisão de ±0,5 °C. Além disso, conforme a Figura 14, este sensor está disponível com um encapsulamento de aço inoxidável, a prova d'água, permitindo assim que medições de temperatura de líquidos possam ser realizadas, sem danificar o sensor.

2.6.3 LaunchPad TM4C123G

Desenvolvida pela Texas Instruments, a LaunchPad TM4C123G, é comercializada como um kit de desenvolvimento de baixo custo da plataforma ARM, baseado nos microcontroladores da família ARM Cortex-M4F. (TEXAS INSTRUMENTS, 2019). Esta é apresentada na Figura 15.

Figura 15 – LaunchPad TM4C123G da Texas Instruments.

Fonte: TEXAS INSTRUMENTS, 2019.

Esta placa de desenvolvimento está equipada com o microcontrolador TM4C123GH6PM, sendo este um microcontrolador de 32 bits, com clock de até 80 MHz. A memória RAM deste dispositivo é de 256 kB, e a memória EEPROM disponível é de 2 kB.

Além disso, também estão disponíveis, entre outras funcionalidades, 2 conversores A/D de 12 bits, 8 barramentos UART, 6 barramentos I²C e 4 barramentos SPI. (TEXAS INSTRUMENTS, 2019).

Dos periféricos disponíveis, foram utilizados para este projeto 2 barramentos UART, 1 barramento I²C, memória EEPROM e 8 timers.

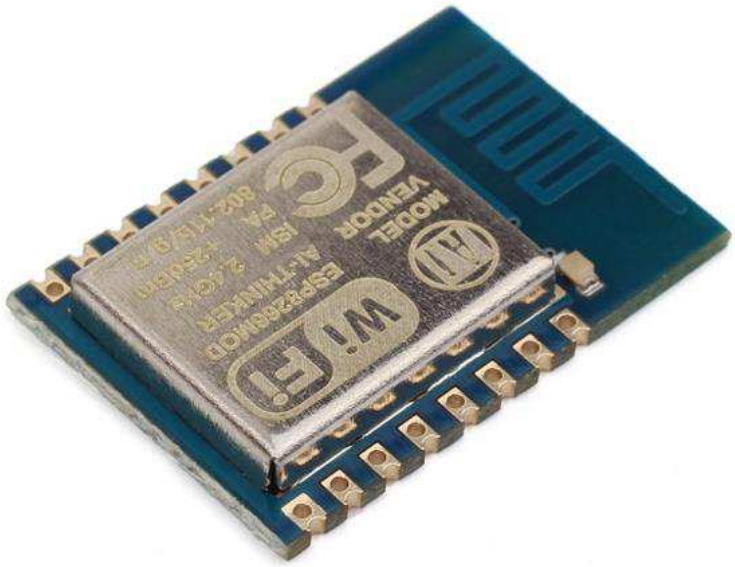
2.6.4 Módulo WiFi ESP8266 ESP-12E

O módulo WiFi ESP8266 foi desenvolvido visando possibilitar a conexão de um microcontrolador a uma rede WiFi de forma fácil, eficaz, e com baixo custo.

Este módulo, apresentado na Figura 16, suporta as redes 802.11 b/g/n. podendo se comunicar com um microcontrolador através de uma conexão UART. Esta comunicação ocorre através de comandos, denominados pela fabricante de comandos AT.

Possui um ótimo custo x benefício e possui uma grande comunidade de usuários.

Figura 16 – Módulo WiFi ESP8266.



Fonte: FILIPEFLOP, 2019.

2.6.5 Servidor

Um servidor normalmente consiste de um programa rodando em um dispositivo específico, com o intuito de prover funcionalidades a outros dispositivos, chamados clientes. Diversos serviços podem ser providos pelos servidores, como o processamento de tarefas ou o compartilhamento de dados com o cliente.

Atualmente, a maioria dos sistemas de servidor é implementada através de um modelo de requisições e respostas, onde um cliente enviará uma requisição para o servidor, que então executará a ação necessária e responderá de acordo com o solicitado. (COMER; STEVENS, 1993).

2.6.5.1 Node.js

O Node.js é um ambiente de execução para JavaScript (JS), permitindo que scripts sejam executados no lado do servidor. O modo de trabalho utilizado pelo Node.js faz com que este se torne leve e eficiente, ideal para aplicações em tempo real com troca intensa de dados através de diversos dispositivos. (DEVMEDIA, 2019a).

Apesar de recente, o Node.js já é utilizado por grandes empresas no mercado de tecnologia, como Netflix, Uber e LinkedIn. (OPUS SOFTWARE, 2019).

O Node.js também conta com um gerenciador de pacotes, o Node Package Manager (NPM), sendo este também o maior repositório de softwares do mundo. Estes pacotes podem ser entendidos como bibliotecas, oferecendo diversas funções, dando potencial para a utilização do Node.js em diversas situações. O pacote mais conhecido se chama Express.js e é um framework completo para desenvolvimento de aplicações Web. (OPUS SOFTWARE, 2019).

2.6.5.2 Application Programming Interface – API

Uma Application Programming Interface (API) é um conjunto de rotinas e padrões que são estabelecidos para que aplicações terceiras possam acessar as funcionalidades de um software. Desta maneira, esta atua como um meio de campo, possibilitando, por exemplo, a comunicação entre um serviço e um banco de dados, além de possibilitar a integração entre sistemas que possuem linguagens totalmente distintas. (CANALTECH, 2019; VERTIGO TECNOLOGIA, 2019).

2.6.5.3 Hyper Text Transfer Protocol – HTTP

O Hyper Text Transfer Protocol (HTTP) é um protocolo de comunicação e troca de dados. É a base para qualquer troca de dados na web, e é um protocolo cliente-servidor, onde as requisições são geradas pelo cliente, chegando até o servidor, onde são processadas e respondidas. (MDN WEB DOCS, 2019a).

Devido à sua extensibilidade, ele é usado não apenas para buscar documentos, mas também imagens e vídeos, ou publicar conteúdo em servidores. O HTTP foi projetado para ser simples e legível às pessoas. As mensagens HTTP podem ser lidas e entendidas

facilmente, proporcionando uma maior facilidade para desenvolvimento e testes. (MDN WEB DOCS, 2019a).

Os principais métodos HTTP, usados na geração de requisições para servidores são GET, DELETE, POST e PUT.

Ao enviar uma requisição do tipo GET, é indicado que estão sendo pegos dados do servidor. Uma requisição do tipo DELETE indica a exclusão de dados. Tanto as requisições do tipo POST quanto as do tipo PUT são utilizadas para o envio de dados, onde a primeira é indicada para o envio, e a segunda indicada para a alteração de dados já existentes. (MDN WEB DOCS, 2019b).

2.6.5.4 Representational State Transfer API

Sendo o HTTP o principal protocolo de comunicação para sistemas web, é interessante o uso do mesmo para a implementação de uma API.

O Representational State Transfer (REST), é um estilo de arquitetura de software que define um conjunto de regras a serem utilizadas para a criação de serviços da internet, podendo utilizar para isto o protocolo HTTP, permitindo, desta forma, que aplicações se comuniquem. (IRIAS, 2019).

Ao utilizar as características do protocolo HTTP, o REST permite um trânsito de informações mais eficiente e, por consequência, mais rápido. (CAMPOMORI, 2019)

Assim, pode-se construir uma REST API, executada no ambiente Node.js, onde a mesma aguarda por requisições por parte dos clientes, requisições estas que utilizam os métodos HTTP. A API é então responsável por executar uma ação, e responder aos clientes.

2.6.6 Desenvolvimento de Aplicações Web

As aplicações web são serviços executados diretos no navegador do usuário, sem a necessidade de que haja a instalação de qualquer software.

A diferença entre uma aplicação web e um website é que este último é estático, apenas disponibilizando informação, sem realizar processamento no servidor. (IMPACTA, 2019).

Uma das vantagens do desenvolvimento de aplicações web é que a maioria dos usuários já está acostumada com o funcionamento dos navegadores. Além disso, a manutenção e a atualização são centralizadas, portanto, não é necessário instalar em todos os computadores, apenas é feita a atualização dos servidores. (IMPACTA, 2019).

Para o desenvolvimento do front-end, ou seja, a aplicação com a qual o usuário tem contato, faz-se o uso, basicamente, de três diferentes linguagens: Hypertext Markup Language (HTML), Cascading Style Sheets (CSS) e JavaScript.

2.6.6.1 Hypertext Markup Language - HTML

O HTML é uma linguagem baseada em marcação, sendo o bloco de construção mais básico da web. Através da criação de tags, são marcados elementos para mostrar quais informações a página exibe, e como essas informações são exibidas. (MDN WEB DOCS, 2019c).

Os arquivos HTML são responsáveis pela estrutura da aplicação web, informando sobre como a mesma é composta.

2.6.6.2 Cascading Style Sheets - CSS

Cascading Style Sheets (CSS) é uma linguagem de estilo utilizada para descrever a apresentação de um documento escrito em HTML, assim, definindo como os elementos são mostrados na tela. (MDN WEB DOCS, 2019d).

O CSS é a segunda camada de uma aplicação web, podendo ser pensada como a decoração da página. Através do mesmo é possível alterar cores de textos, fundos, disposições de espaçamentos, parágrafos, entre outros. (HOSTINGER, 2019).

2.6.6.3 JavaScript

JavaScript é uma linguagem de programação orientada a objetos, multiplataforma. É uma linguagem pequena e leve. (MDN WEB DOCS, 2019e).

Utilizando apenas as linguagens HTML e CSS, a página ainda é considerada estática. A linguagem JavaScript entra como a terceira camada da aplicação web, permitindo a implementação de funcionalidades mais complexas a aplicação. (DEVMEDIA, 2019b).

JavaScript em si é bastante compacto, mas muito flexível. Diversos desenvolvedores escreveram uma ampla variedade de ferramentas utilizando a linguagem JavaScript principal, proporcionando assim uma grande quantidade de funcionalidades extras com o mínimo de esforço, tendo como exemplo a criação de gráficos e tabelas. (MDN WEB DOCS, 2019f).

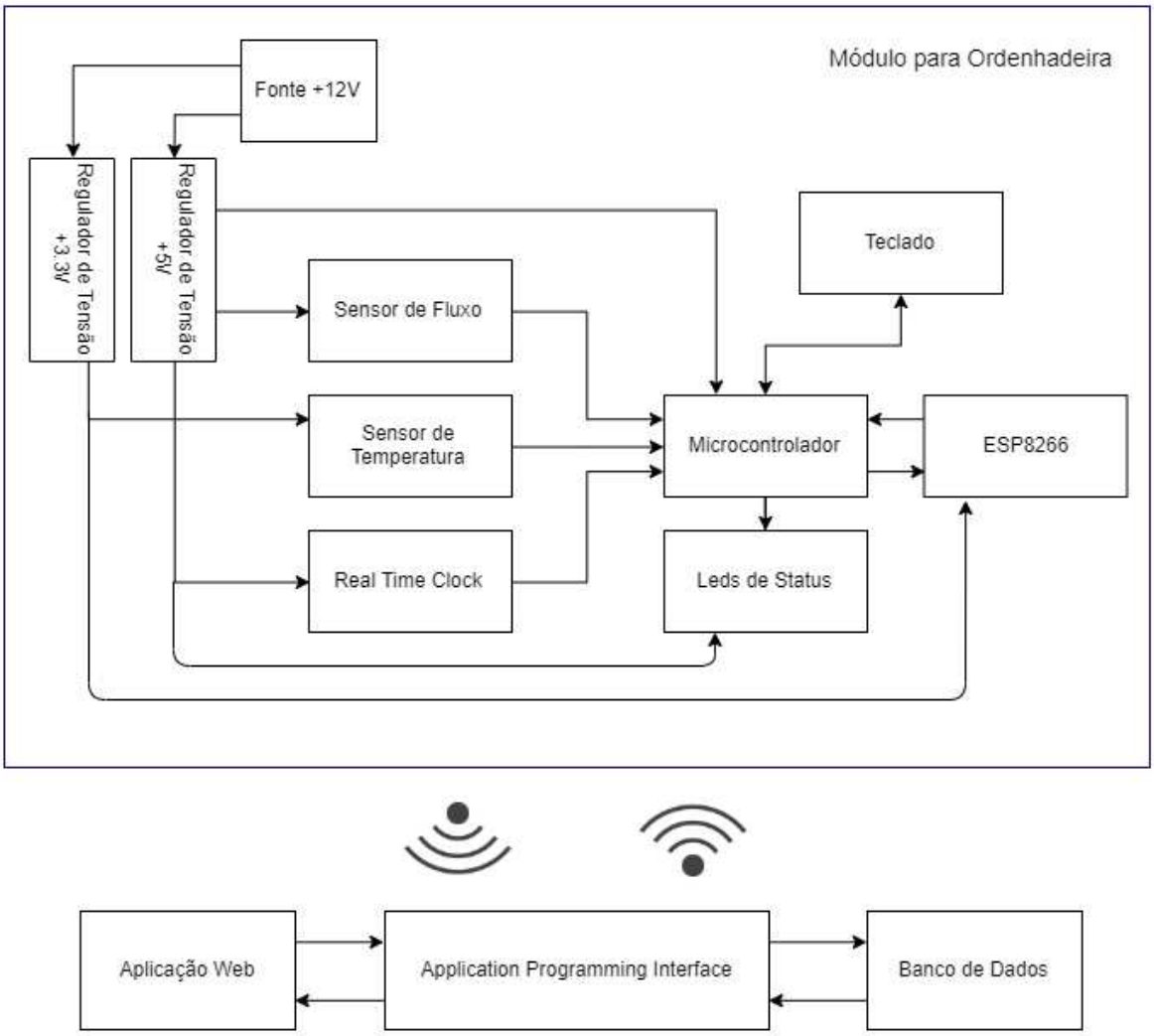
3 DESENVOLVIMENTO DO PROJETO

Neste capítulo são apresentados os métodos adotados para a realização do projeto, demostrando a integração dos componentes no sistema.

O sistema proposto é dividido em partes, onde cada parte tem sua função específica, e trabalha em conjunto com as demais. Um módulo principal de instrumentação é responsável por fazer a coleta de dados sobre a ordenha, um banco de dados armazena os mesmos, e uma aplicação web acessa-os para que a análise possa ser feita. Ainda, permitindo a integração de todas as partes, encontra-se uma Application Programming Interface (API).

A Figura 17 permite ter uma visão geral do projeto. Nela, podem ser identificadas todas as partes que compõe o projeto, partes estas que terão seu desenvolvimento e funcionamento explicadas na sequência.

Figura 17 – Diagrama de blocos do projeto.



Fonte: O autor, 2019.

3.1 BANCO DE DADOS

Um banco de dados é um conjunto de arquivos, os quais possuem uma coleção de informações com relação entre si.

Através do uso de um banco de dados, todas as informações coletadas sobre as ordenhas podem ser salvas, de maneira coerente, permitindo o acesso as mesmas posteriormente para análise. Da mesma forma, os dados referentes aos animais produtores podem ser armazenados e acessados facilmente.

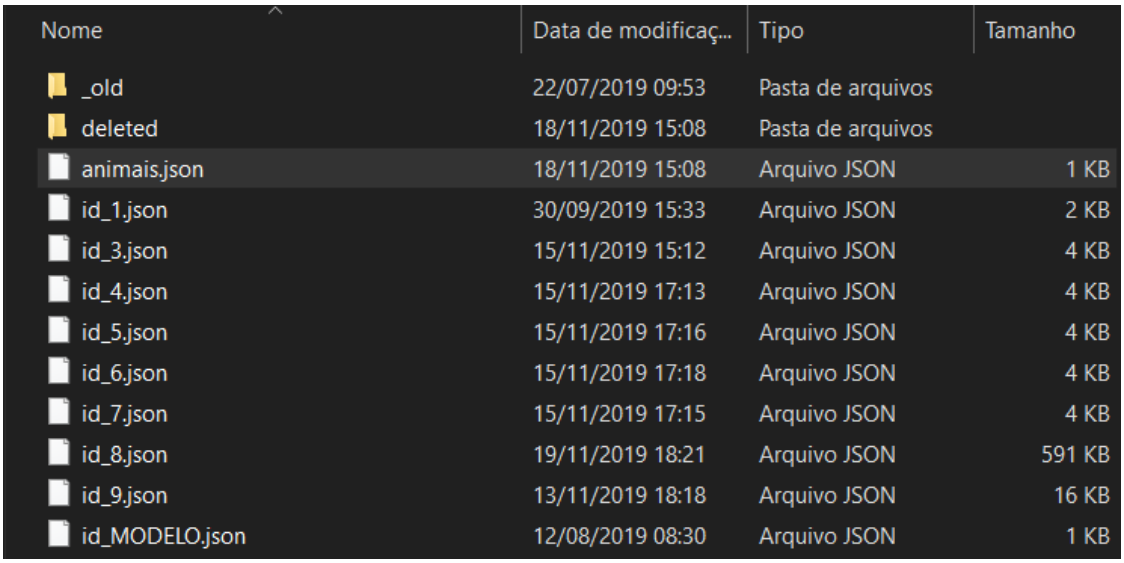
Para a implementação deste banco, foi optado pelo uso de arquivos no formato JSON, sendo este um formato compacto, com ótima aplicação em trocas de dados simples e rápidas. O formato trabalha em um padrão “chave”: valor, sendo facilmente gerado e interpretado.

Os arquivos JSON são leves e de rápida manipulação, mesmo quando estes contêm uma grande quantidade de dados.

Além disso, estes arquivos podem ser manipulados diretamente na API, podendo ser criados, modificados ou excluídos pela mesma, sem que haja a necessidade da utilização de um software específico para gerenciamento de bancos de dados.

Os dados ficam armazenados em uma pasta específica no computador. O local desta pasta não deve ser alterado, para permitir que a mesma seja acessada posteriormente sem nenhum tipo de problema. Como o desenvolvimento se deu de forma local, a pasta de dados fica localizada no mesmo computador em que a API é executada. A Figura 18 apresenta esta pasta, com os arquivos que nela estão contidos.

Figura 18 – Pasta que contém os arquivos do banco de dados.



Fonte: O autor, 2019.

O banco de dados criado, conta com um arquivo principal chamado `animais.json`, neste, ficam armazenados as informações sobre os animais que estão cadastrados no sistema. A Figura 19 apresenta o formato como os dados se encontram. Para cada animal cadastrado, a chave `name` contém o nome do animal, a chave `id` contém o número de identificação, `raca` contém a raça e `lote` o número do lote em que o animal está.

Figura 19 – Arquivo animais.json.


```
[
    {"name":"Vaca 1","id":1,"raca":"Teste Cadastro","lote":1},
    {"name":"Brian","id":6,"raca":"Teste Diario","lote":3},
    {"name":"Toretto","id":3,"raca":"Teste Diario","lote":3},
    {"name":"Dom","id":8,"raca":"Teste Stress","lote":2},
    {"name":"Princesa","id":4,"raca":"Teste Diario","lote":3},
    {"name":"Mimosa","id":5,"raca":"Teste Diario","lote":3},
    {"name":"Pipoca","id":7,"raca":"Teste Diario","lote":3}
]
```


Fonte: O autor, 2019.

Também, nesta mesma pasta de dados, para cada animal cadastrado no arquivo `animais.json`, existe um arquivo `id_XX.json`, onde `XX` é o número de identificação do animal. Dentro deste arquivo, como pode ser visto na Figura 20, ficam armazenados todos os dados que são coletados durante a ordenha.

Figura 20 – Arquivo id_XX.json.


```
[
    {
        "id" : 1,
        "data" : "01/01/2019",
        "hora" : "06:00:00",
        "ordenha_n" : 1,
        "tempo_ordenha" : "05:00",
        "temperatura" : 37,
        "volume" : 12.5
    },
    {
        "id" : 1,
        "data" : "01/01/2019",
        "hora" : "18:00:00",
        "ordenha_n" : 2,
        "tempo_ordenha" : "05:00",
        "temperatura" : 37,
        "volume" : 12.5
    }
]
```


Fonte: O autor, 2019.

Aqui, a chaves `data` e `hora` representam a data e hora da coleta dos dados, respectivamente, `ordenha_n` apresenta se esta é a primeira, segunda, terceira ou a enésima ordenha realizada no dia, `tempo_ordenha` é o tempo de duração do processo de ordenha, `temperatura` é a temperatura média em que o leite foi retirado, em graus Celsius, e `volume` é o volume produzido nesta ordenha, em litros. Também neste mesmo arquivo existe a chave `id`. Esta chave se faz desnecessária internamente, neste arquivo, já que todas as informações nele contidas se referem ao mesmo número de identificação. Porém, é interessante mantê-la, pois no caso da manipulação de vários arquivos de dados em paralelo, de diferentes animais, esta chave indica a que animal o dado que está sendo manipulado se refere.

Tendo definido o formato para armazenamento dos dados, pode-se dar sequência ao projeto, partindo para o desenvolvimento da API, que será responsável, entre outras coisas, pela manipulação do banco de dados.

3.2 DESENVOLVIMENTO DA APPLICATION PROGRAMMING INTERFACE – API

Como visto na Figura 17, a API desempenha função chave neste projeto, devido ao fato de a mesma possibilitar a comunicação e troca de dados entre as partes.

A API é responsável por receber requisições e fazer o tratamento das mesmas, realizando uma rotina, de acordo com o tipo de requisição recebida, e devolvendo uma resposta aos clientes geradores destas requisições.

As requisições recebidas seguem o protocolo HTTP, protocolo este que enviará um dos seguintes métodos: GET, POST, PUT ou DELETE. As funções que são realizadas pela API se dão de acordo com o método recebido.

Considerando que a API é executada no ambiente Node.js, o primeiro passo para o desenvolvimento da mesma é a instalação deste ambiente.

Para inicializar a construção, fez-se o uso do pacote para JavaScript express-generator. Com este pacote, ao executar o comando `express –view=hbs API` no ambiente Node.js, todos os arquivos e pacotes necessários para a criação da API são criados e instalados.

Um dos pacotes instalados de grande importância é o pacote Express. Este pacote disponibiliza uma série de funções facilitadoras na criação de servidores que utilizam o protocolo HTTP. Assim, a API utilizada neste projeto foi desenvolvida com base nas funções do pacote Express.

Outro pacote instalado foi o fs-extra. Este pacote precisa ser instalado manualmente, já que o mesmo não é configurado pelo express-generator. O pacote fs-extra adiciona funções a linguagem JavaScript para permitir a manipulação de arquivos JSON, desta forma este se faz necessário para permitir que a API manipule o banco de dados diretamente.

Dentro da API, as rotinas realizadas são também conhecidas como rotas. Com todos os pacotes necessários instalados, dá-se então início a construção das rotas.

3.2.1 Rotas da API

Sendo as rotas da API, as rotinas realizadas pela mesma, estas foram desenvolvidas para serem utilizadas pelas demais partes do projeto.

No processo de criação da API, as rotas não foram desenvolvidas todas em sequência. A API e suas rotas foram desenvolvidas em paralelo com as demais partes do projeto, sendo assim, as rotas eram criadas de acordo com a necessidade das mesmas. No momento em que alguma parte do projeto precisasse manipular os dados, uma rota era criada para suprir esta necessidade, caso uma das já existentes não atendesse as especificações necessárias.

Nos subcapítulos a seguir, estão apresentadas as principais rotas desenvolvidas. As rotas aqui apresentadas são utilizadas pelo módulo de instrumentação e/ou pela aplicação web. Nestes subcapítulos, não estão explicadas as utilizações das rotas por estas partes, as formas como estas são utilizadas estão descritas no decorrer do trabalho.

3.2.1.1 Rotas do tipo GET

As rotas do tipo GET são utilizadas pelos clientes da API quando os mesmos necessitam pegar dados, do banco de dados ou de outro local. Assim, as rotas deste tipo devem retornar dados.

3.2.1.1.1 GET /idCadastrados

Esta rota é acessada pelo cliente, quanto este gera uma requisição do tipo GET para o endereço /idCadastrados.

Ao receber uma requisição, esta rota usa o pacote fs-extra para abrir o arquivo do banco de dados `animais.json`, e retorna para o cliente o número de identificação de todos os animais cadastrados no sistema.

3.2.1.1.2 GET /animalData/:id

Nesta rota, `:id` indica um parâmetro, que deve ser passado pelo cliente no momento da requisição.

Ao receber essa requisição, a API abre o arquivo de dados coletados do animal indicado pelo parâmetro `id`, e devolve para o cliente todos os dados que já foram coletados para este animal.

3.2.1.1.3 GET /loteAvg/:lote

Esta rota é acessada quando um cliente gera uma requisição do tipo GET para o endereço /loveAvg/:lote, onde `:lote` deve ser passado como parâmetro, informando sobre qual lote pretende-se pegar as informações.

A função desta rota é devolver a média de todos os dados já coletados para este lote.

O primeiro passo realizado é abrir o arquivo `animais.json`, percorrer os animais cadastrados no mesmo, e identificar todos os que pertencem ao lote desejado.

Como esta rota devolverá três médias, são criados três vetores, um que receberá os dados para fazer a média do volume, um para a média de temperatura e um para a média de tempo.

Tendo os animais pertencentes ao lote, os arquivos que contém os dados coletados sobre os mesmos, `id_XX.json` (onde `XX` corresponde aos números de identificação), são abertos, um após o outro. Todos os dados coletados sobre estes animais são então adicionados aos vetores criados, sendo que o vetor de volume recebe os dados de volume, o de temperatura os dados de temperatura e o de tempo os dados de tempo.

Ao final da leitura dos arquivos de dados de todos os animais do lote, o vetor de volume contém todos os dados de volume já coletados, para todos os animais deste lote, assim como os vetores de temperatura e tempo.

As médias de volume, temperatura e tempo são então calculadas, fazendo a soma de todos os dados de cada vetor, e dividindo pelo número de itens deste vetor, conforme demonstrado pela Equação 2, onde *n* é igual ao número de dados contidos no vetor.

$$Média = \frac{1}{n} \cdot \sum_{i=1}^{n} dado_i \qquad (2)$$

As três médias calculadas são então retornadas para o cliente.

3.2.1.1.4 GET /lote360/:lote; /lote60/:lote; /lote14/:lote

Essas três rotas realizam o mesmo procedimento que a rota GET/loteAvg/:lote, apresentada no subcapítulo 3.2.1.1.3, porém, para realizar as médias, estas rotas utilizam apenas as últimas 360, 60 e 14 coletas de dados, respectivamente.

Desta forma, estas rotas retornam para o cliente as médias do lote, apenas para as últimas ordenhas.

Caso não haja dados suficientes para realizar as médias, ou seja, foram coletados menos de 360, 60 ou 14 dados, todos os dados disponíveis são utilizados.

3.2.1.1.5 GET /setDate

Para acessar esta rota, o cliente deve gerar uma requisição do tipo GET para o endereço /setDate. Ao receber uma requisição neste endereço, esta rota utiliza as funções Date, nativas da linguagem JavaScript, para pegar a data e hora atuais.

Esta rota então retorna uma resposta no formato JSON, com as informações sobre data e hora, da seguinte forma: [ano, mês, dia, hora, minuto, segundo].

A criação desta rota foi pensada para que o módulo de instrumentação possa gerar uma requisição, e utilizar a resposta recebida para ajustar data e hora do Real Time Clock (RTC) presente no mesmo.

3.2.1.2 Rotas do tipo POST

As rotas do tipo POST devem ser utilizadas quando há a necessidade do envio de dados por parte dos clientes para a API. Uma importante diferença entre as requisições POST e as requisições GET, é que a primeira possui um corpo, enquanto a segunda não.

O corpo da requisição pode ser entendido como a mensagem que está sendo enviada pelo cliente para a API. Assim, quando o cliente desejar enviar dados, estes dados devem ser colocados no corpo da requisição.

3.2.1.2.1 POST /newAnimal

Uma requisição do tipo POST, no endereço /newAnimal é usada para realizar o cadastro de um novo animal no banco de dados. Como esta é uma rota do tipo POST, e pretende-se enviar dados, no corpo da requisição devem estar informados o número de identificação do animal que será cadastrado, o lote, o nome e a raça.

Ao receber esta requisição, a API abre o arquivo `animais.json`, e verifica se o número de identificação que está sendo cadastrado é válido.

Esta rota foi construída de forma a não aceitar um número de identificação maior que 999 ou menor que 0. Se identificado um desses casos, a rota retorna a mensagem “invalid” para o cliente. Além disso, a rota também verifica se o número informado já se encontra cadastrado, respondendo “already” caso este seja o caso.

Tendo sido aprovado na verificação do número de identificação, os dados do corpo da requisição são adicionados ao arquivo `animais.json`, e o arquivo `id_XX.json` é criado (onde `XX` é o número de identificação informado), para o armazenamento dos dados coletados. Ao final, a rota ainda responde ao cliente com a mensagem “OK”.

3.2.1.2.2 POST /sendData

Esta rota é responsável por fazer o recebimento dos dados que são coletados sobre a ordenha.

Ao gerar esta requisição, o cliente deve informar no corpo da mesma o número de identificação do animal, a data e a hora em que a ordenha foi realizada, a duração da ordenha, a temperatura média do leite, e o volume produzido.

Verificando a existência dos dados no corpo da requisição, a rota responde ao cliente com a mensagem “Data Received”.

Após isso, a rota abre o arquivo de dados do animal informado pelo número de identificação, e adiciona os dados recebidos neste arquivo.

3.2.1.2.3 POST /customData/:id

Ao receber uma requisição POST neste endereço, esta rota é ativada, e a função da mesma é devolver os dados coletados sobre o animal `:id`, dentro de um intervalo de tempo, definido por uma data inicial e uma data final.

Mesmo as rotas POST sendo recomendadas apenas para o envio de dados, as mesmas também possibilitam retornar informações. Esta rota foi implementada como POST, e não como GET, pois as requisições GET não possuem corpo, e para esta rota há a necessidade de informar no corpo da requisição as datas inicial e final.

Ao receber esta requisição, as datas no corpo da mesma vêm no formado dd/mm/aaaa. O primeiro passo é converter estas datas para dias, conforme a Equação 3.

$$dias = aaaa \cdot 365 + mm \cdot 30 + dd \tag{3}$$

Agora, o arquivo de dados coletados do animal `:id` é aberto, e todos estes dados são avaliados, para saber se foram coletados dentro do período desejado.

Para fazer esta avaliação, as datas de coleta também são convertidas para dias, novamente de acordo com a Equação 3. Se a data de coleta em dias estiver compreendida entre a data inicial e final, este dado é selecionado para ser retornado ao cliente.

Após ter avaliado todos os dados coletados, e selecionados apenas os desejados, estes são então retornados.

3.2.1.2.4 POST /customAvg/:id

Esta rota tem seu funcionamento muito parecido com as rotas GET /loteAvg/, GET /lote360/, GET /lote60/ e GET /lote14/, porém, esta deve receber no corpo da requisição uma data inicial e uma data final, e os dados utilizados para calcular as médias são apenas os que se encontram dentro do período informado.

Aqui, novamente, é usada uma requisição do tipo POST para pegar dados.

A seleção dos animais que são utilizados para realizar a média é feita da mesma forma que na rota GET /loteAvg/, analisando todos os animais cadastrados e selecionando apenas os que fazem parte do lote desejado.

A seleção dos dados para cálculo da média é feita da mesma forma que na rota POST /customData/, convertendo as datas para dias e analisando se a data de coleta se encontra dentro do período desejado.

Novamente, conforme a rota GET /loteAvg/ após ter todos os dados selecionados, separados em três vetores, um para volume, um para temperatura e um para tempo, as médias são calculadas, de acordo com a Equação 2, previamente apresentada, e são então retornadas para o cliente.

3.2.1.3 Rotas do tipo PUT

As rotas do tipo PUT apresentam grande semelhança com as do tipo POST. Assim como as rotas POST, as rotas PUT também possuem corpo, e são utilizadas para o envio de dados. Porém, esta última tem seu uso recomendado quando há necessidade de alterar dados já existentes.

3.2.1.3.1 PUT /modifyAnimal/:id

Quando uma requisição do tipo PUT é gerada para este endereço, passando o animal `:id` como parâmetro, as informações sobre este animal podem ser modificadas.

Além do parâmetro `:id`, no corpo da requisição deve haver o nome do animal, a raça e o número do lote.

Para realizar a modificação, o arquivo `animais.json` é aberto, e o número de identificação passado como parâmetro é localizado. As informações sobre o animal contidas neste arquivo são então substituídas pelas informações passadas no corpo da requisição.

3.2.1.4 Rotas do tipo DELETE

Conforme indicado pelo nome, as rotas do tipo DELETE são utilizadas para a exclusão de dados. Estas rotas podem ou não apresentar corpo, e suas respostas não devem retornar nenhum dado.

3.2.1.4.1 DELETE /deleteAnimal/:id

Esta rota é utilizada para fazer a exclusão de um animal cadastrado no banco de dados. O animal que será excluído é definido pelo parâmetro `:id`.

Quando a API recebe esta requisição, o arquivo `animais.json` é aberto, e o número de identificação passado como parâmetro é encontrado. As informações referentes a este animal são então excluídas deste arquivo.

Além disso, o arquivo que contém os dados coletados sobre a ordenha deste animal é movido para uma segunda pasta, pasta esta que contém os arquivos de dados excluídos.

3.3 MÓDULO PARA ORDENHADEIRA

Tendo toda a construção da API explicada, parte-se para a explicação do desenvolvimento do Módulo para Ordenhadeira, o qual é responsável por fazer a identificação do animal que será ordenhado, além de contar com sensores para a análise da ordenha, e uma conexão com redes Wi-Fi para comunicação com a API.

3.3.1 Definições Iniciais

Em princípio, como microcontrolador para este projeto foi escolhida a placa de desenvolvimento LaunchPad TM4C123G da Texas Instruments, sendo que este, posteriormente, passou com testes juntamente com o módulo Wi-Fi ESP8266, para verificar se ambos trabalhando em conjunto atenderiam as necessidades do projeto. Este microcontrolador foi escolhido por atender as necessidades para comunicações serial, e também possuir um grande número de pinos de I/O, além de disponibilizar diversos Timers. Além disso, este microcontrolador facilita muito a prototipagem, por poder ser facilmente conectado as demais partes do projeto, e ser facilmente programado e gravado.

Em um primeiro momento, foi pensado em fazer a divisão do módulo para ordenhadeira em duas partes separadas, uma parte para a instrumentação e outra parte para a conexão e troca de dados com o servidor. Com esta divisão, poderiam ser instalados vários módulos de instrumentação, todos conectados a um único módulo de conexão com o servidor. Porém esta ideia logo foi abandonada, optando-se pelo projeto de um único módulo, com as funções de instrumentação e conexão. Assim, este único módulo tem todas as funções necessárias para o funcionamento, a instalação no posto de ordenha é facilitada, e todo o posto de ordenha fica isolado dos demais, sendo que um problema em um posto não impede o funcionamento dos outros.

Também foi pensado no início do projeto, na criação de uma interface homem máquina (IHM) junto ao módulo para ordenhadeira, através de um display LCD. Porém, esta ideia também foi descartada, optando-se ao invés disso, pela implementação de funções mais completas na aplicação web para a avaliação das ordenhas. Devido ao fato de que o acompanhamento da ordenha seria feito através desta aplicação, a implementação de uma IHM para acompanhamento acabaria tendo pouco uso, desta maneira não compensando ser desenvolvida.

3.3.2 Definição dos Sensores

Para a leitura dos dados de temperatura, optou-se pelo uso do sensor DS18B20, devido as facilidades oferecidas pelo mesmo. Este sensor é comercializado em um encapsulamento de aço inox, a prova d'água, o que atende perfeitamente as necessidades do projeto. Este sensor também tem sua leitura digital, sendo facilmente utilizado junto ao microcontrolador.

Para a medição de fluxo, tentou-se fazer uso do sensor Meister DMIK-7, pois este, a princípio, apresentava ser uma ótima escolha para o projeto, tendo sua construção em aço inoxidável e sem possuir partes móveis. Porém ao entrar em contato com a empresa Grunn, revendedora brasileira deste sensor, foi verificado o preço elevado do mesmo, de aproximadamente 1950,00 €, preço este que acabaria por inviabilizar totalmente o projeto [2].

O projeto foi então desenvolvido utilizando o sensor de fluxo YF-S401, mesmo esse não sendo o sensor ideal para a instalação do sistema em uma ordenhadeira real.

Com as definições iniciais do módulo para ordenhadeira, é dado início ao desenvolvimento deste.

3.3.3 Implementação do Módulo Wi-Fi ESP8266 12-E

O primeiro passo no desenvolvimento foi a implementação do módulo ESP8266, pois no início não se tinha a certeza de que seria possível realizar as operações necessárias com a API utilizando o conjunto TM4C123G e ESP8266, logo iniciou-se o desenvolvimento por esta parte, para que fossem testados o envio e recebimento de dados.

O módulo ESP8266 permite sua utilização de duas formas, de maneira independente, onde o mesmo é programado através da interface do Arduino, utilizando a linguagem Lua, ou em conjunto com um microcontrolador, se comunicando com este através de uma interface UART. Para este projeto foi utilizado o módulo ESP8266 em conjunto com um microcontrolador.

Trabalhando em conjunto com o microcontrolador, segundo o fabricante, o ESP8266 está operando no modo AT. Este nome modo AT se dá, devido ao fato de que quando o microcontrolador precisa realizar operações utilizando o ESP8266, é necessário enviar através

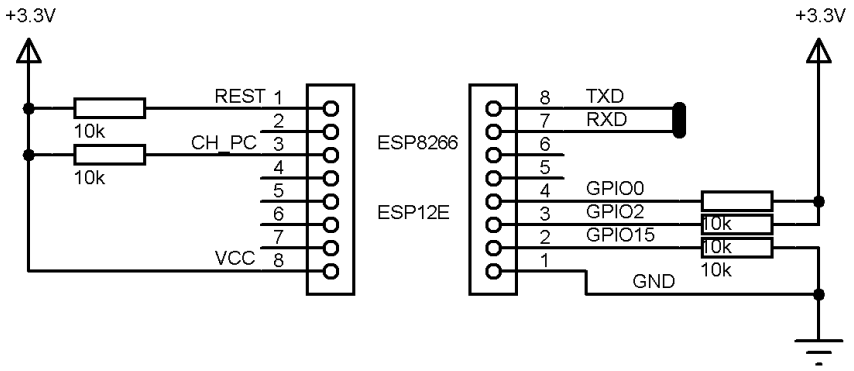
---

[2] Orçamento obtido com a empresa Grunn, em junho de 2019.

da UART a mensagem “AT” seguido do comando que representa a operação a ser feita. Uma lista com todos os possíveis comandos é disponibilizada pela fabricante Espressif Systems.

Neste projeto foi utilizado a variação 12-E do módulo ESP8266, sendo assim, de acordo com a fabricante, há a necessidade de alguns resistores de pull-up e pull-down para que o módulo seja operado no modo AT, conforme o esquemático mostrado na Figura 21.

Figura 21 – Esquemático ESP8266 no modo AT.



Fonte: O autor, 2019

Para facilitar a utilização da ESP8266, e permitir sua reutilização em outros projetos, uma biblioteca foi desenvolvida, onde todas as funções necessárias foram criadas.

Para a comunicação entre o microcontrolador e o ESP8266 foi utilizada a porta UART1 disponível no TM4C123G. Esta porta foi configurada para trabalhar com um baud rate de 115200 bps, taxa em que opera o ESP8266.

Para fazer o acompanhamento da execução do programa do microcontrolador, foi utilizado o terminal virtual TeraTerm. A conexão entre o computador e o microcontrolador é feita através de uma porta USB, e o microcontrolador envia as informações para o terminal através de sua porta UART0.

Quando pretende-se realizar uma operação com o ESP8266, um comando deve ser enviado através da UART1. Para o envio dos comandos foi criada uma função, que recebe o comando a ser enviado, e faz o envio do mesmo, caractere após caractere através da UART, até que todo o comando tenha sido enviado.

Para cada comando enviado, o ESP8266 faz o processamento do mesmo, e retorna uma resposta ao microcontrolador. Esta resposta pode ser interpretada, para saber se o comando foi bem executado ou não. Dessa maneira, a cada comando enviado, é necessário fazer o monitoramento da porta UART1, aguardando a resposta por parte do ESP8266. O Quadro 3 apresenta alguns dos comandos utilizados, e as possíveis respostas para estes.

Quadro 3 – Comandos enviados para ESP8266 e suas respostas.

Comando Enviado	**Função**	**Possíveis Respostas**
AT	Verifica o modo AT	OK
ATE	Desativa o eco	OK
AT+CWJAP_CUR=”ssid”,”senha”	Conecta-se ao ponto de acesso ssid	OK WIFI CONNECTED ERROR
AT+CIPSTATUS	Verifica o status da conexão Wi-Fi	STATUS:2 STATUS:3 STATUS:4 STATUS:5
AT+CIPSTART	Abre conexão com API	OK ERROR

Fonte: O autor, 2019.

O monitoramento da UART1 para receber a resposta do ESP8266 deve ser feito durante um período de tempo “timeout”, devido ao fato de que o processamento de alguns comandos demanda certo tempo, logo a resposta não é imediata. Estes tempos de timeout variam muito de comando para comando, e tiveram que ser ajustados na prática, já que os mesmos não são disponibilizados pela fabricante. A Tabela 6 apresenta os principais comandos utilizados, e seu tempo timeout ajustados.

Tabela 6 – Principais comandos utilizados e tempos de timeout.

Comando utilizado	Tempo de Timeout (em ms)
AT	1
ATE0	1
AT+CWMODE_DEF=1	1
AT+CIPSTATUS	300
AT+CWJAP_CUR=”ssid”,”senha”	4000
AT+CIPMUX=1	1
AT+CIPSTART	1500
AT+CIPSEND	5

Fonte: O autor, 2019.

Para analisar a reposta recebida pelo ESP8266 para cada comando enviado, foi desenvolvida uma função. Esta função, chamada `receiveATstatus`, recebe como parâmetro o tempo timeout, que define por quanto tempo deve ser feito o monitoramento da porta UART1, e a resposta que é esperada do ESP8266. A função deve então ser chamada após o envio de cada comando, para verificar se o mesmo foi aceito e executado.

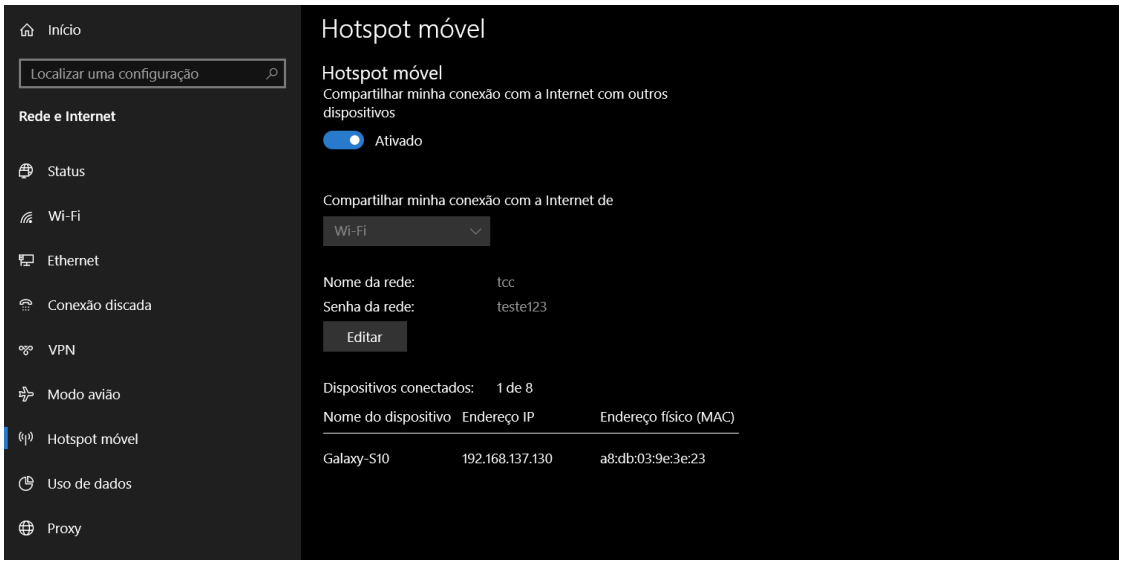
A função `receiveATstatus` faz o monitoramento da porta UART1, durante o período de tempo informado, armazenando a resposta recebida em uma string. Tendo esgotado o tempo de monitoramento, a reposta recebida é comparada com a resposta desejada, que foi passada para a função como parâmetro. Caso a resposta recebida seja igual a desejada, esta função retorna TRUE, informando que o a operação foi bem-sucedida.

Agora, na biblioteca desenvolvida para o ESP8266 diversas funções puderam ser criadas, todas com o mesmo princípio. Cada função envia um comando específico, e aguarda por uma resposta desejada, retornando TRUE ou FALSE, para indicar se a reposta recebida é igual ou não ao que se deseja.

Tendo todas as funções necessárias para o funcionamento do módulo Wi-Fi, o próximo passo é conectar o mesmo a uma rede deste tipo. Como a API não foi hospedada em um servidor on-line, mas está rodando em uma rede local, há a necessidade de que o módulo se conecte a mesma rede do computador em que a API está sendo executada.

Para a criação da rede Wi-Fi, foi utilizado uma ferramenta do sistema operacional Windows, chamada Hotspot Móvel, que permite transformar o computador em um ponto de acesso Wi-Fi. A Figura 22 apresenta esta ferramenta.

Figura 22 – Windows Hotspot Móvel.



Fonte: O autor, 2019.

Como a API está rodando neste mesmo computador, no momento em que o módulo de ordenha se conectar a esta rede, tanto a API quanto o módulo estarão conectados à mesma rede, permitindo a troca de dados. Esta ferramenta substitui o uso de um roteador, que também poderia ter sido utilizado, sem nenhum tipo de problema, desde que tanto o computador que executa a API quanto o módulo se conectem ao mesmo.

3.3.3.1 Conexão do Módulo a rede Wi-Fi

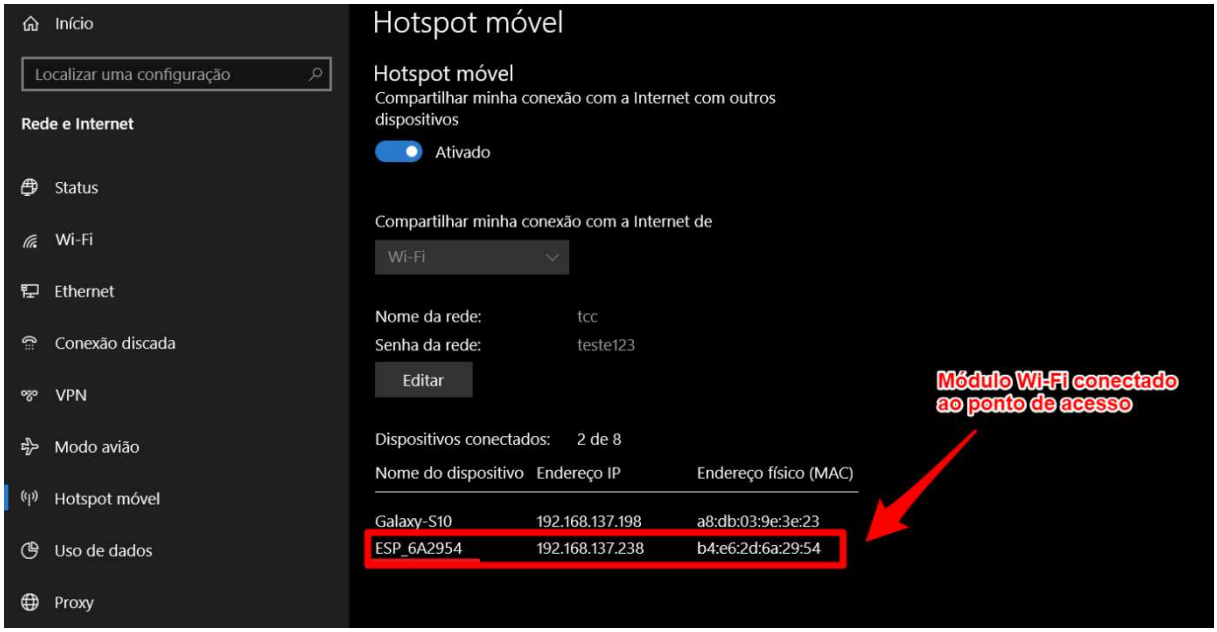
Antes de fazer a conexão ao ponto de acesso Wi-Fi, o microcontrolador deve enviar outros comandos para o ESP8266, fazendo algumas configurações iniciais. Assim, foi implementada uma rotina de inicialização do ESP8266. Esta rotina é realizada sempre que o módulo para ordenhadeira é ligado, sem a necessidade de sua repetição durante o uso do mesmo.

A rotina de inicialização consiste no envio de alguns comandos. Inicialmente é enviado para o ESP8266 o comando “AT”. Este é usado para verificar se o ESP8266 está conectado e operando no modo AT, a resposta esperada é “OK”. Após é enviado o comando “ATE0”, seguido do comando “AT+CWMODE_DEF=1”, para definir que o ESP8266 será um cliente e se conectará a um ponto de acesso Wi-Fi. Por fim, deve ser habilitado o modo de múltiplas conexões, através do comando “AT+CIPMUX=1”.

Tendo sido inicializado, o ESP8266 está pronto para se conectar a uma rede Wi-Fi. A conexão é feita quando o microcontrolador envia o comando ‘AT+CWJAP_CUR=”ssid”,”senha”’. Neste comando, `ssid` é o nome do ponto de acesso ao qual está tentando ser feita a conexão, e `senha` é a senha do mesmo. Aqui, `ssid` e `senha` são definidos na programação do microcontrolador, sendo assim, o usuário não tem a opção de selecionar em qual roteador o sistema se conectará. O módulo tentará se conectar sempre ao mesmo ponto de acesso, que não deve ter seu nome ou senha alterados.

Após enviado o comando para conexão, a porta UART1 é monitorada. Caso a conexão seja bem-sucedida, o ESP8266 responde com a mensagem “WIFI CONNECTED OK”, e a conexão pode ser vista no Hotspot criado, conforme mostra a Figura 23.

Figura 23 – Windows Hotspot com modulo ESP8266 conectado.



Fonte: O autor, 2019.

Após concluída a conexão com a rede Wi-Fi, é implementada a troca de dados com a API, confirmando a possibilidade de realizar operações com o conjunto ESP8266 e LaunchPad TM4C123G.

3.3.3.2 Recebendo dados da API

O primeiro passo para troca de dados com a API é a abertura de uma conexão com esta. Para conectar-se com a API, é enviado o comando AT+CIPSTART=4,”TCP”,”HOST”,3002. Neste comando, `TCP` é o tipo de conexão que está sendo feita, `HOST` se refere ao roteador em que o módulo está conectado e `3002` é a porta em que a API está sendo executada. Caso a conexão seja estabelecida, o ESP8266 responde com a mensagem “OK”, indicando que não há nenhum erro com a API.

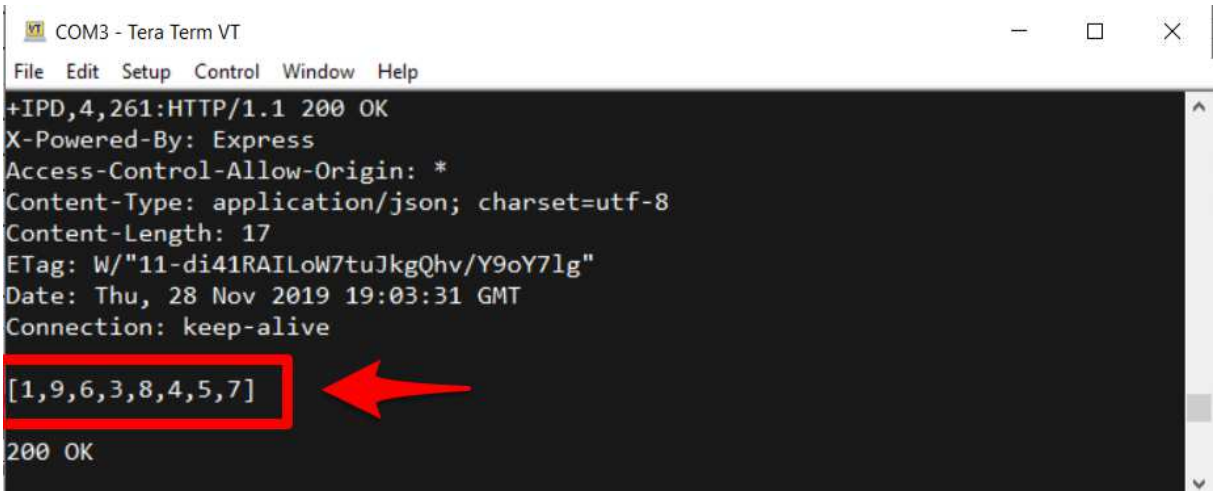
Com a conexão aberta, será gerada uma requisição do tipo GET para o endereço /idCadastrados. Uma requisição neste endereço retorna o número de identificação dos animais que estão cadastrados no banco de dados.

Para gerar esta requisição, é enviado o comando “AT+CIPSEND=4,”, informando o início do envio. Então é enviada a mensagem “GET /idCadastrados HTTP/1.1”, seguida da mensagem “HOST: host”, onde `host` novamente refere-se ao roteador em que o módulo está conectado.

Após isso, é utilizada a função `receiveATstatus` para monitorar a porta UART1, e receber a mensagem retornada pela API.

A mensagem enviada pela API e recebida pelo microcontrolador pode ser vista utilizando o terminal TeraTerm, e está apresentada na Figura 24. Nesta figura, o vetor recebido contém os números de identificação.

Figura 24 – Mensagem recebida da API, exibida no TeraTerm.



Fonte: O autor, 2019.

3.3.4 Implementação de Rotinas Automáticas

Com a conexão à rede Wi-Fi e a troca de dados funcionando normalmente, foi então implementada uma rotina automática, fazendo uso de um timer.

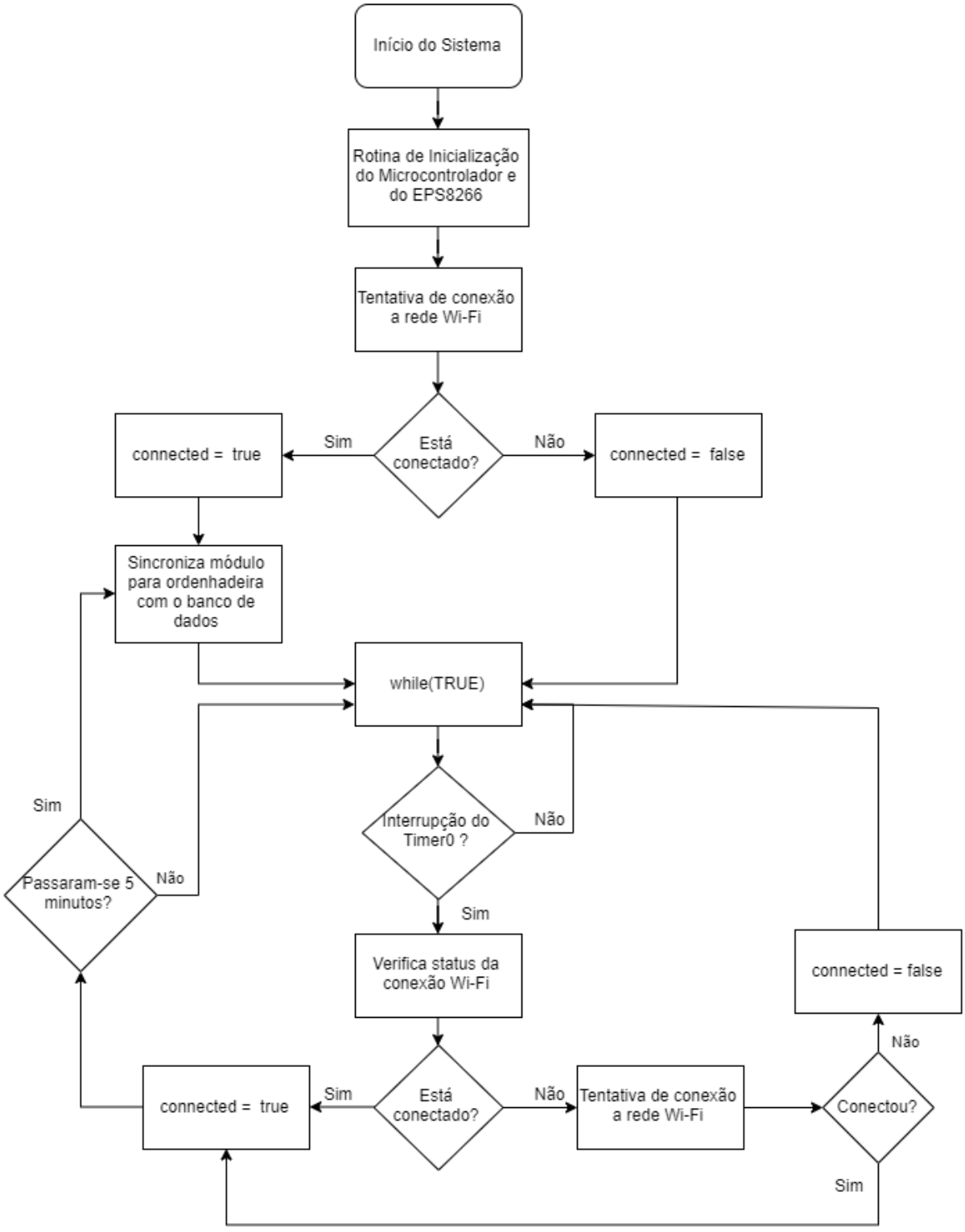
O Timer0 do microcontrolador foi programado para temporizar 60 segundos. Ao final desses 60 segundos, indicados por uma interrupção gerada pelo timer, o microcontrolador envia os comandos para o ESP8266, verificando a conexão com a rede Wi-Fi. Caso seja verificado que o módulo não está conectado, uma tentativa de conexão é feita. Uma variável global `connected` indica se o módulo está ou não conectado, recebendo os valores TRUE ou FALSE, respectivamente.

Além disso, na quinta interrupção gerada pelo Timer0, indicando a passagem de 5 minutos, é gerada uma requisição do tipo GET para o endereço /idCadastrados. Através desta requisição, o número de identificação dos animais cadastrados no sistema é pego, e estes são salvos em um vetor. Assim, o módulo para ordenhadeira é sincronizado com o banco de dados a cada 5 minutos.

Aqui, vale ressaltar que esta sincronização só é feita caso seja verificado que o módulo possui conexão com a rede Wi-Fi.

O Figura 25 apresenta o fluxograma desta parte do firmware.

Figura 25 – Fluxograma Rotinas Automáticas



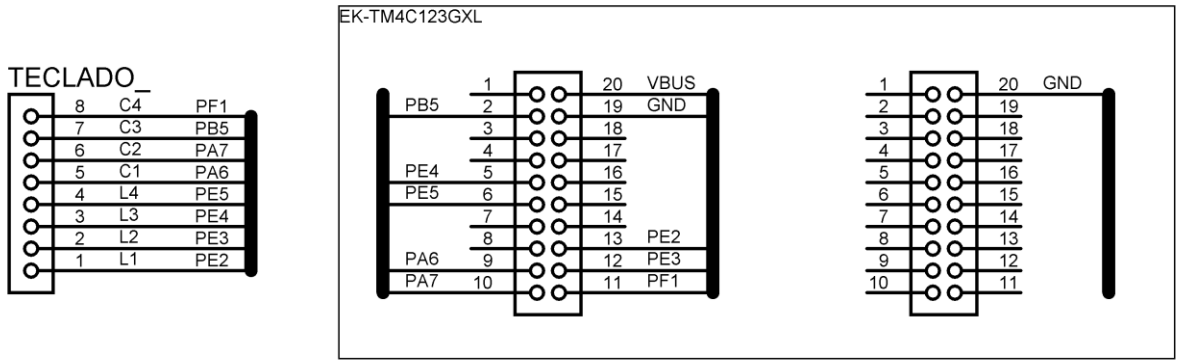
Fonte: O autor, 2019.

3.3.5 Identificação do animal para ordenha

Com o número de identificação dos animais cadastrados disponível no módulo para ordenha, foi então trabalhado na identificação do animal que será ordenhado.

Para realizar a identificação, optou-se pelo uso de um teclado matricial 4x4, conectado ao microcontrolador conforme o esquemático apresentado na Figura 26.

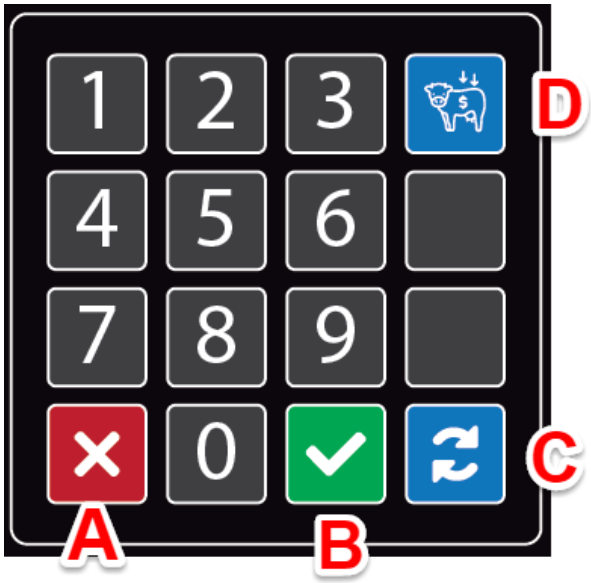
Figura 26 – Esquemático de ligação teclado x LaunchPad TM4C123G.



Fonte: O autor, 2019.

O layout do teclado foi definido conforme a Figura 27.

Figura 27 – Layout do teclado.



Fonte: O autor, 2019.

O botão A é utilizado para cancelar e o botão B é utilizado para confirmar. Ao clicar no botão C, é realizado o sincronismo com o servidor. Este sincronismo também é feito de forma automática, a cada 5 minutos, mas, caso o usuário queira realizar antecipadamente o sincronismo, isto pode ser feito através deste botão.

Ao ser pressionado o botão D, é gerada uma interrupção por mudança de estado no pino do microcontrolador, indicando uma nova ordenha.

No momento em que uma nova ordenha é indicada, o Timer0 é desabilitado, impedindo que novas verificações de conexão sejam realizadas durante a ordenha. É então esperado que o usuário digite o número de identificação do animal que será ordenhado.

Ao pressionar o botão confirmar, o número informado pelo usuário é comparado com os números de identificação salvos no módulo para ordenhadeira, recebidos da API. Caso seja verificado que o número informado corresponde a um animal já cadastrado, é dado sinal positivo para a coleta dos dados, caso contrário, o sistema retorna ao estado anterior. Querendo digitar um novo número de identificação, o usuário precisa pressionar novamente o botão D, para que todo o processo se repita.

3.3.6 Implementação dos Sensores

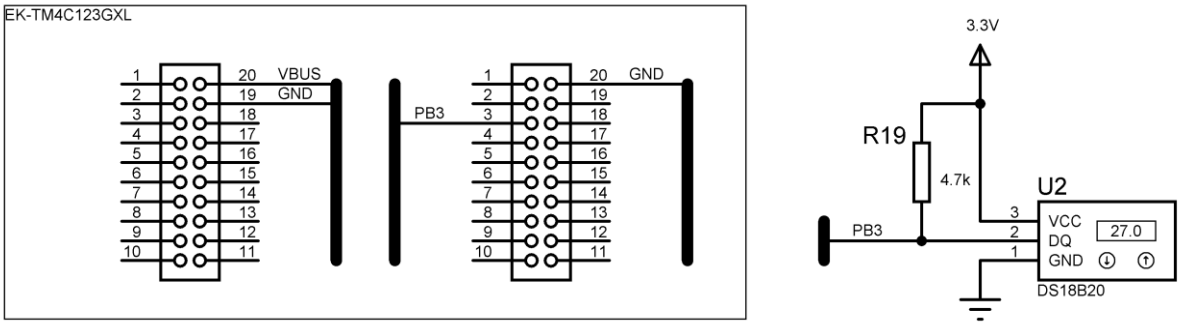
Com a parte de identificação pronta, e os animais cadastrados já armazenados no módulo para ordenhadeira, parte-se para a implementação dos sensores, para que posteriormente sejam coletados os dados.

3.3.6.1 Sensor de Temperatura DS18B20

O sensor de temperatura é utilizado para fazer a leitura da temperatura média do leite durante a ordenha. Este sensor DS18B20 é fabricado pela empresa Maxim Integrated, e trabalha com o protocolo de comunicação serial 1-Wire.

O sensor é conectado ao microcontrolador conforme o esquemático mostrado na Figura 28.

Figura 28 – Esquemático de conexão sensor DS18B20 x LaunchPad TM4C123G.



Fonte: O autor, 2019

Existem quatro operações básicas que podem ser realizadas no barramento 1-Wire: reset, escrever bit 1, escrever bit 0 e ler bit. O tempo para realizar cada operação varia de acordo com o sensor. Além disso o barramento 1-Wire exige um resistor de pull-up para correto funcionamento.

Para implementação do sensor DS18B20, a fabricante Maxim Integrated disponibiliza uma biblioteca pronta, que utiliza um timer para a geração dos tempos de cada operação. Este timer gera tempos de 0.25μS.

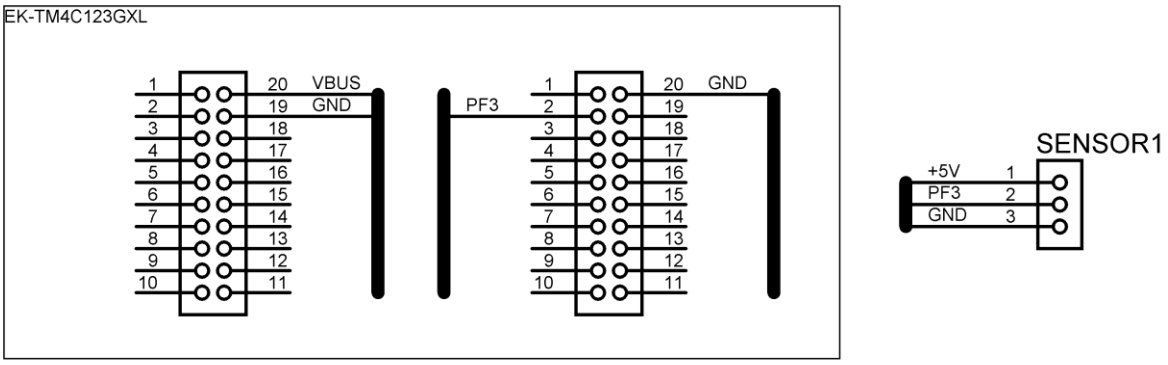
Assim foi construída uma biblioteca para implementar este sensor no projeto, com base na biblioteca disponibilizada pela fabricante, apenas adaptando para o microcontrolador aqui utilizado. Para a geração dos tempos, foi utilizado o Timer3 do microcontrolador.

A leitura da temperatura é feita através de uma função, que envia os comandos necessários para o barramento 1-Wire, e monitora o mesmo, aguardando pela indicação de que a temperatura está pronta para ser lida. Esta função faz então a leitura da temperatura e retorna o valor lido.

3.3.6.2 Sensor de Fluxo YF-S401

O esquemático da Figura 29 demonstra a conexão do sensor de fluxo com o microcontrolador.

Figura 29 – Esquemático de conexão sensor de fluxo YF-S401 x LaunchPad TM4C123G.



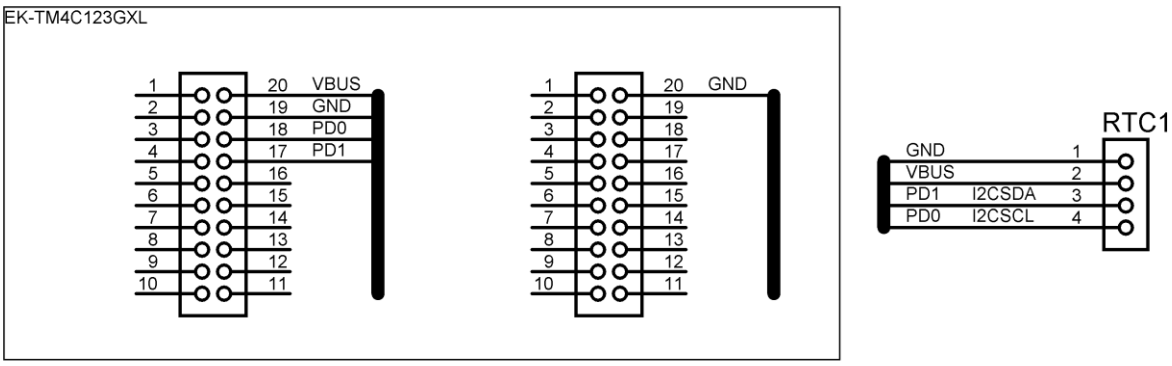
Fonte: O autor, 2019.

Para a implementação deste sensor, é necessário fazer a leitura do número de pulsos gerados pelo mesmo. Para fazer esta leitura, foi utilizado o Timer1B do microcontrolador, configurado para trabalhar no modo de contagem de eventos externos. Assim, cada pulso gerado pelo sensor é aplicado a entrada deste timer, incrementando-o.

3.3.6.3 Real Time Clock DS1307

O RTC DS1307 comunica-se com o microcontrolador através de um barramento I²C. Para este projeto, o RTC foi implementado utilizando o barramento I2C3 do microcontrolador. A conexão entre estes pode ser vista no esquemático apresentado na Figura 30.

Figura 30 – Esquemático elétrico RTC DS1307 x LaunchPad TM4C123G.



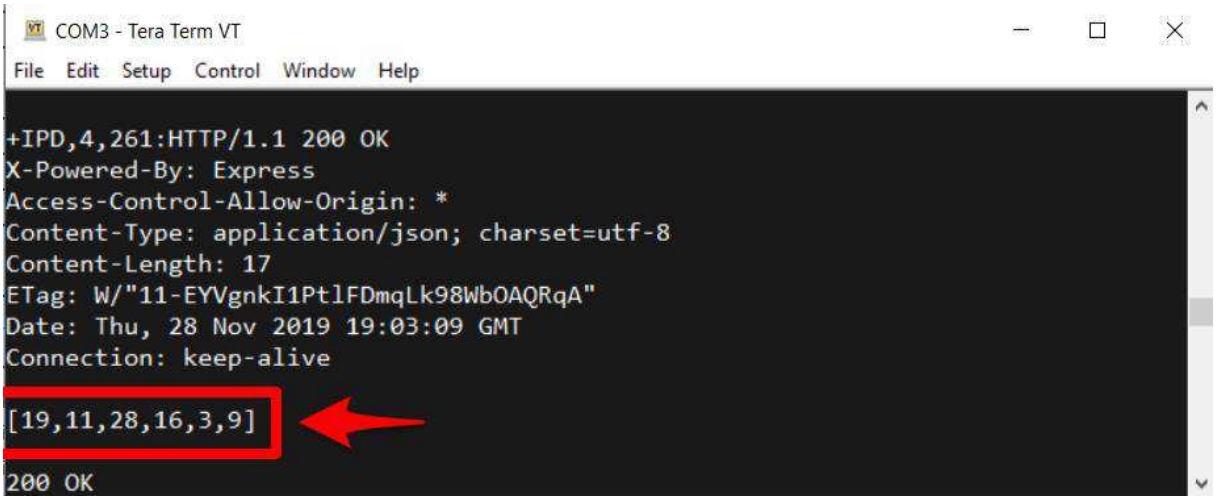
Fonte: O autor, 2019.

O ajuste de data e hora do RTC foi implementado de maneira automática, através da conexão Wi-Fi, sem que o usuário precise realizar nenhuma ação. Após o módulo ser inicializado, e verificado que há conexão com a rede Wi-Fi, é então feito o ajuste do RTC. Caso não haja conexão, este ajuste simplesmente é pulado.

Para realizar o ajuste, é feita uma requisição do tipo GET para o endereço /setDate. O procedimento para gerar esta requisição é o mesmo demonstrado anteriormente, onde são pegos os animais cadastrados no sistema. A Figura 31 apresenta a resposta recebida da API pelo microcontrolador. Na resposta, o vetor recebido está no formato [ano, mês, dia, hora, minutos, segundos].

Tendo a reposta da API, os dados recebidos são separados em dia, mês, ano, hora, minuto e segundo, e são então gravados no RTC.

Figura 31 – Resposta recebida da API para a requisição /setDate.



Fonte: O autor, 2019.

3.3.7 Coleta de Dados

Tendo instalados os sensores necessários para o módulo, parte-se para a implementação da coleta dos dados sobre a ordenha. A coleta de dados é liberada após ser indicada uma nova ordenha, e informado um número de identificação que está cadastrado no sistema.

Sendo liberada a coleta de dados, é aguardado o início da ordenha. O início da ordenha é identificado pelo sensor de fluxo. O recebimento do primeiro pulso indica que a ordenha iniciou.

No momento em que a coleta de dados é liberada, também é habilitado o Timer4 do microcontrolador. Este timer é configurado para contar um tempo de 5 minutos. Caso esses 5

minutos passem, e o início da ordenha não for identificado, a coleta de dados é automaticamente cancelada. Caso o início seja identificado, ao receber o primeiro pulso do sensor de fluxo, o Timer4 é desativado.

Quando identificado o início da ordenha, o RTC é lido, pegando Dia/Mês/Ano e Hora:Minuto:Segundo.

Enquanto a ordenha está em progresso, os pulsos recebidos são incrementados, e o valor de temperatura é constantemente lido e atualizado, de acordo com a Equação 4. Esta equação caracteriza uma média móvel, e a mesma foi escolhida por ser capaz de suavizar quaisquer grandes flutuações que possam vir a ocorrer na medição da temperatura.

$$temperatura = \frac{[temperatura + DS18B20ReadTemperature()]}{2} \qquad (4)$$

Ao decorrer 5 segundos, contados pelo Timer2, sem o recebimento de nenhum pulso do sensor de fluxo, é detectado o final do processo de ordenha.

Identificando-se o final da ordenha, o RTC é lido novamente, pegando HoraFinal:MinutoFinal:SegundoFinal. A duração da ordenha, em segundos é calculada de acordo com a Equação 5.

$$duracao = [HoraFinal \cdot 3600 + MinutoFinal \cdot 60 + SegundoFinal - 5] - [Hora \cdot 3600 + Minuto \cdot 60 + Segundo] \qquad (5)$$

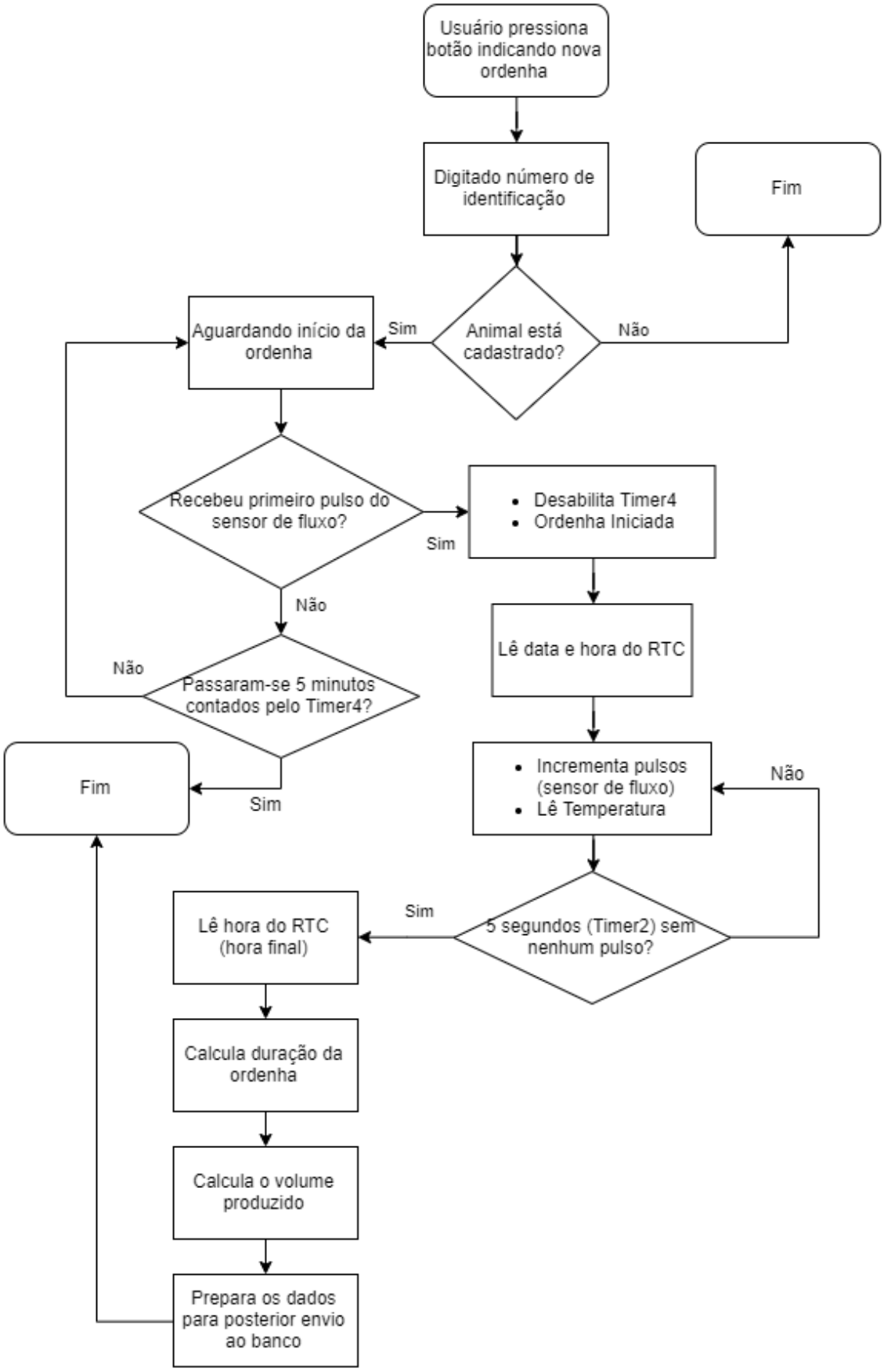
Após, a duração é convertida para o formato Minuto:Segundo.

O volume produzido é calculado, em mililitros, pela Equação 6, onde 5880 referem-se aos pulsos gerados pelo sensor de fluxo para cada litro que passa pelo mesmo.

$$volume = \frac{pulsos \cdot 1000}{5880} \qquad (6)$$

O fluxograma da Figura 32 demonstra como é feita a coleta de dados.

Figura 32 – Fluxograma processo de coleta dos dados.



Fonte: O autor, 2019.

3.3.8 Envio de dados para o banco

Tendo finalizada a coleta de dados, trata-se de enviar os mesmos para a API, que os escreverá no banco de dados.

Para fazer o envio, todos os dados coletados precisam ser escritos em uma única string. Esta string se tornará o corpo da requisição que é gerada.

O primeiro passo para o envio dos dados é a abertura de uma conexão com a API, seguido da indicação do início do envio, da mesma forma feita anteriormente para o recebimento dos animais cadastrados no sistema. Em seguida é gerada uma requisição do tipo POST no endereço /sendData.

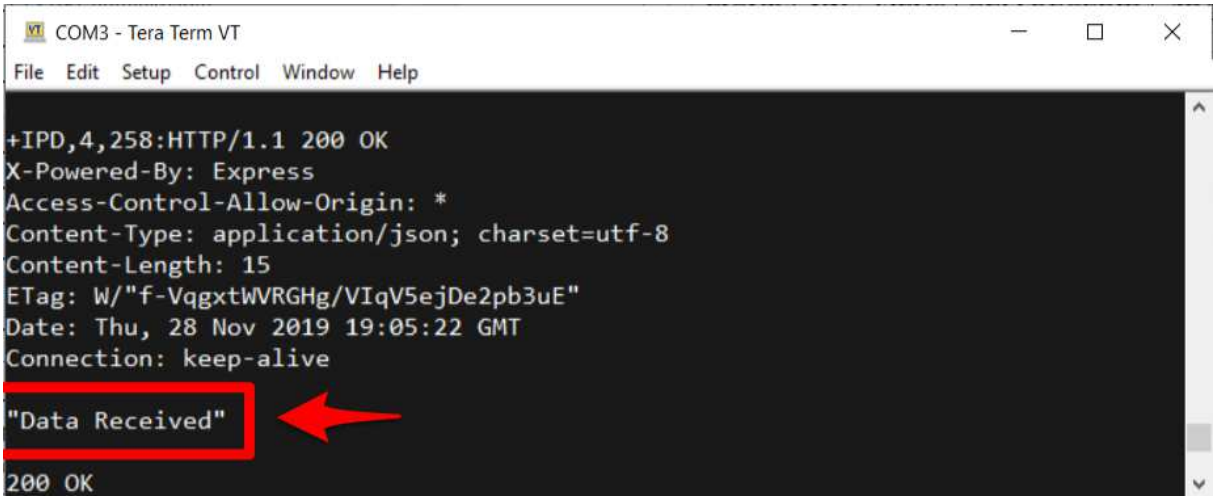
Para gerar a requisição, é enviada a mensagem “POST /sendData HTTP/1.1”, seguida da mensagem “HOST: host”, onde host refere-se ao roteador em que o módulo está conectado. Após deve ser enviada a mensagem “Content-Length: XX”, onde XX é o número de caracteres que serão enviados no corpo da requisição. Este número é calculado através da função nativa da linguagem C strlen(). Feito isso, é enviada a mensagem “Content-Type: application/json”, seguida do corpo da requisição. O corpo da requisição é apresentado a seguir:

```
"{"id":id, "data":"dd/mm/aaaa", "hora":"hh:mm:ss",
"tempo_ordenha":"mm:ss", "temperatura":temperatura, "volume":volume}".
```

No corpo da requisição, a chave id é o número de identificação do animal, data e hora referem-se a data e hora da coleta dos dados, tempo_ordenha é a duração da ordenha.

Enviando a requisição, a reposta recebida da API é apresentada na Figura 33.

Figura 33 – Resposta recebida da API ao envio de dados.



Fonte: O autor, 2019.

Na resposta da API, pode-se perceber a mensagem “Data Received”, indicando o sucesso no envio dos dados.

3.3.9 Ordenhas quando não há conexão com a rede Wi-Fi

Buscando permitir que o usuário possa coletar os dados de ordenha, mesmo quando há um problema com a rede Wi-Fi, os números de identificação dos animais cadastrados, recebidos da API, passaram a ser salvos na memória EEPROM do microcontrolador. Assim, caso o módulo seja iniciado, e a rede Wi-Fi não esteja disponível, pode-se saber sobre os animais cadastrados lendo a memória EEPROM.

Os endereços de 0 a 99 da memória EEPROM são separados para guardar o número dos animais cadastrados, permitindo que 100 animais sejam salvos.

Também, caso ocorra erro no momento de envio dos dados coletados, estes são salvos na memória EEPROM. O endereço 100 da memória é usado para indicar quantas ordenhas estão salvas, aguardando envio.

Cada ordenha salva na EEPROM utiliza 11 endereços, salvando os dados id, dia, mês, ano, hora, minuto, segundo, minuto duração, segundo duração, temperatura e volume, cada um em um endereço.

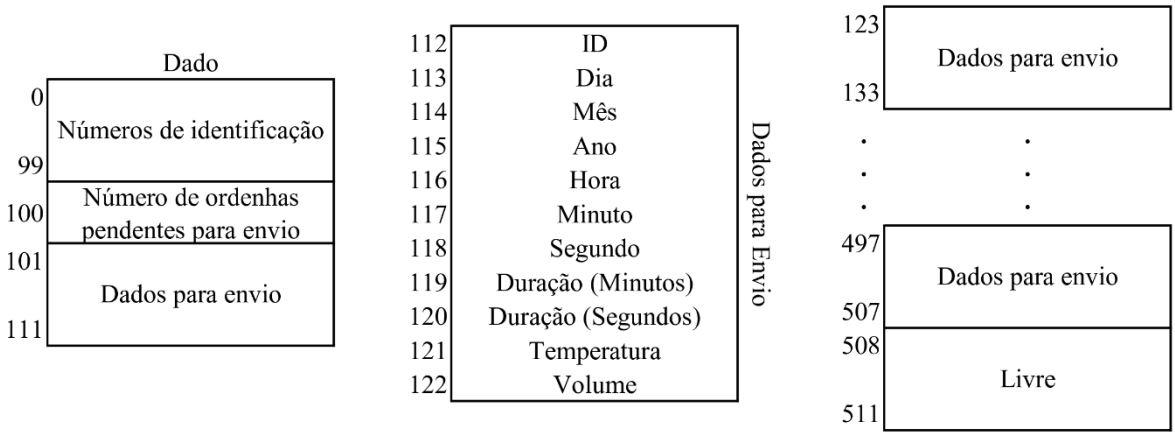
Como os endereços de 0 a 99 são ocupados para salvar os animais cadastrados, o endereço 100 é usado para salvar o número de ordenhas pendentes para envio, e a memória EEPROM possui 512 endereços, podem ser salvas 37 ordenhas na memória EEPROM, conforme Equação 7. Assim, o usuário pode realizar 37 ordenhas sem conexão com a Wi-Fi.

$$ordenha_{offline} = \frac{512 - 101}{11} = 37{,}3636 \quad (7)$$

Caso uma 38ª ordenha seja realizada, sem conexão com a rede Wi-Fi, estes dados não poderão ser salvos, e serão descartados pelo módulo.

Na Figura 34 é apresentado o mapa da memória EEPROM.

Figura 34 – Mapa da Memória EEPROM.



Fonte: O autor, 2019.

O envio dos dados salvos na memória EEPROM é feito em duas ocasiões. Ao ser inicializado o módulo para ordenhadeira, se o mesmo conseguir conectar-se à rede Wi-Fi, e verificado que existem dados pendentes para envio, o envio dos mesmos é feito.

Outro momento em que os dados são enviados, é ao ter a interrupção pelo Timer1, marcando a passagem de 5 minutos. Ao ser gerada esta interrupção, também são sincronizados os animais cadastrados no banco de dados.

Após fazer o envio dos dados da memória EEPROM com sucesso, o conteúdo dos endereços onde estes se encontravam são limpos.

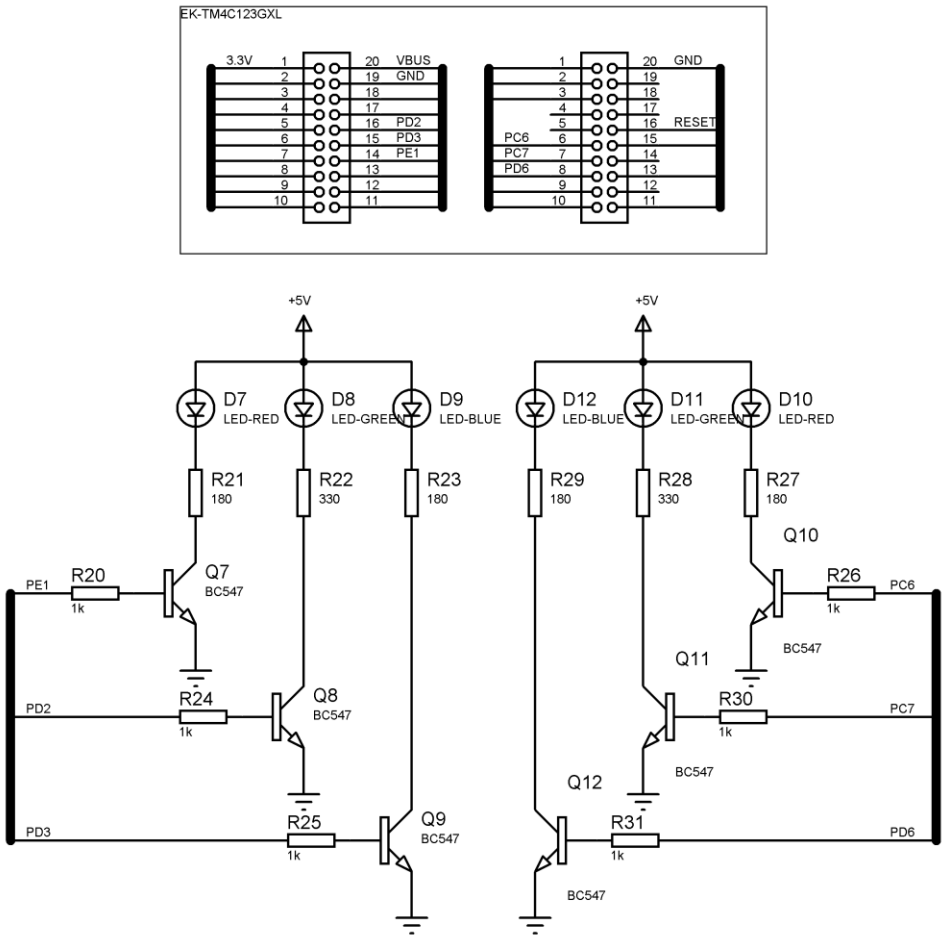
3.3.10 LEDs de Status

Devido ao fato de que, no começo do projeto, foi optado pela não utilização de um display LCD junto ao módulo para ordenhadeira, foram implementados LEDs RGB, que servem para indicar o status de operação do módulo.

Foram utilizados dois LEDs, um para o status da conexão com a Wi-Fi, e outro para o status da parte de ordenha e coleta de dados.

Considerando a corrente limitada nas portas do microcontrolador, para ser feito o acionamento dos LEDs fez-se o uso de transistores BJT, conforme o esquemático apresentado na Figura 35.

Figura 35 – Esquema elétrico dos LEDs de status.



Fonte: O autor, 2019.

Também foram utilizados 3 timers do microcontrolador, operando no modo PWM, com uma frequência de 2 Hz, para permitir que os LEDs piscassem.

O Quadro 4 apresenta as possibilidades para os LEDs, bem como os status indicados pelos mesmos.

Quadro 4 – Status indicados pelos LEDs.

Operação	**LED1 - Ordenha**	**Piscando?**	**LED2 – Wi-Fi**	**Piscando?**
Sistema desligado	-	-	-	-
Inicializando sistema	Azul	Sim	-	-
Sistema pronto para receber número de identificação	Azul	Não	Verde / Vermelho	Não
Usuário digitando número de identificação	Azul	Sim	Verde / Vermelho	Não
Número de identificação aceito	Verde	Não	Verde / Vermelho	Não
Número de identificação recusado	Vermelho	Não	Verde / Vermelho	Não
Coletando dados	Verde	Sim	Verde / Vermelho	Não
Verificando conexão Wi-Fi	-	-	Azul	Sim
Conectando a rede Wi-Fi	-	-	Azul	Sim
Wi-Fi conectado	-	-	Verde	Não
Wi-Fi desconectado	-	-	Vermelho	Não
Sincronizando dados com o servidor	-	-	Azul	Sim
Erro na API	-	-	Vermelho	Não

Fonte: O autor, 2019.

3.3.11 Circuito de Alimentação

Como os sensores, microcontroladores e módulos trabalham em funções diferentes, precisou ser projetado um circuito de alimentação. O mesmo foi criado utilizando uma fonte 12V / 1A. Esta fonte foi escolhida por atender as necessidades de corrente do projeto, conforme demonstrado pela Tabela 7. Nesta tabela foram utilizados os valores máximos de corrente para cada componente, de acordo com o datasheet dos mesmos.

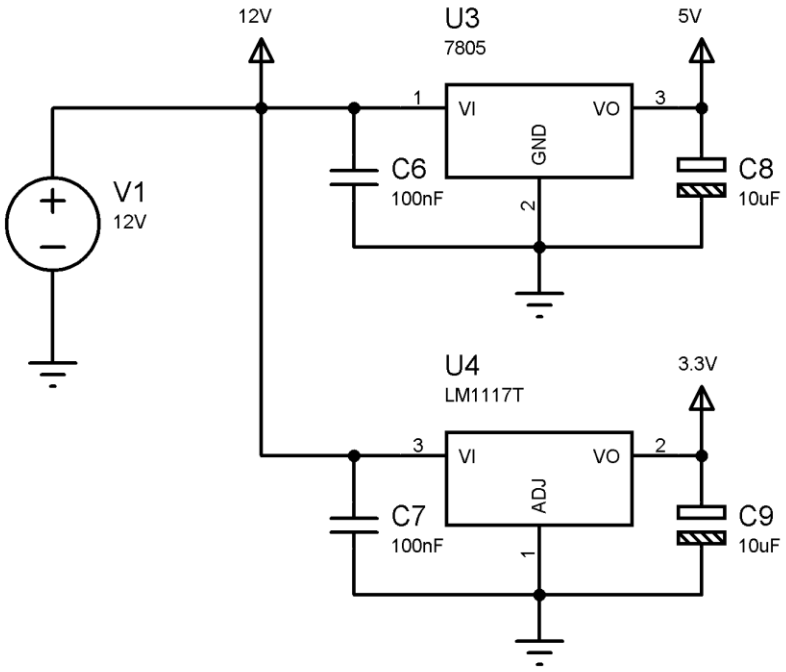
Tabela 7 – Consumo dos componentes utilizados no projeto.

Peça	Consumo Máximo (mA)
Microcontrolador TM4C123GH6PM	58,7
2x Leds RGB	30
Sensor de Fluxo YF-S401	15
Sensor de Temperatura DS18B20	4
RTC DS1307	1,5
Módulo Wi-Fi ESP8266	215
Consumo Total	324,2

Fonte: O autor, 2019.

A tensão de 12V é aplicada a dois reguladores de tensão, um LM7805, que regula para +5V, e um LM1117T, que regula para +3.3V, conforme o esquemático mostrado na Figura 36. Estas duas tensões reguladas são usadas para alimentar as partes eletrônicas do projeto.

Figura 36 – Esquema elétrico dos reguladores de tensão.



Fonte: O autor, 2019.

3.4 CONSTRUÇÃO DO PROTÓTIPO

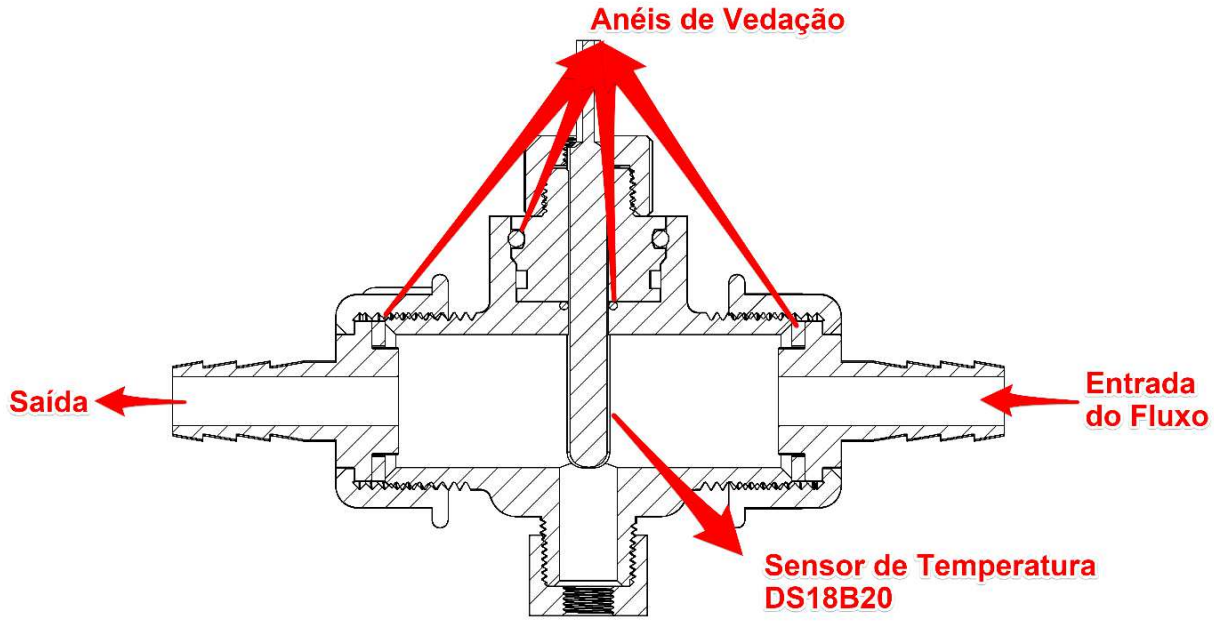
Com a finalização de todo o módulo para ordenha, e do banco de dados pronto para utilização, parte-se para a construção do protótipo, através do qual pode-se pôr à prova a capacidade do sistema em desenvolvimento.

3.4.1 Protótipo do Módulo para Ordenhadeira

Para que seja possível a coleta de dados sobre a ordenha, os sensores devem ser instalados entre a unidade de ordenha e a linha de leite, de maneira que todo o leite ordenhado passe pelos mesmos.

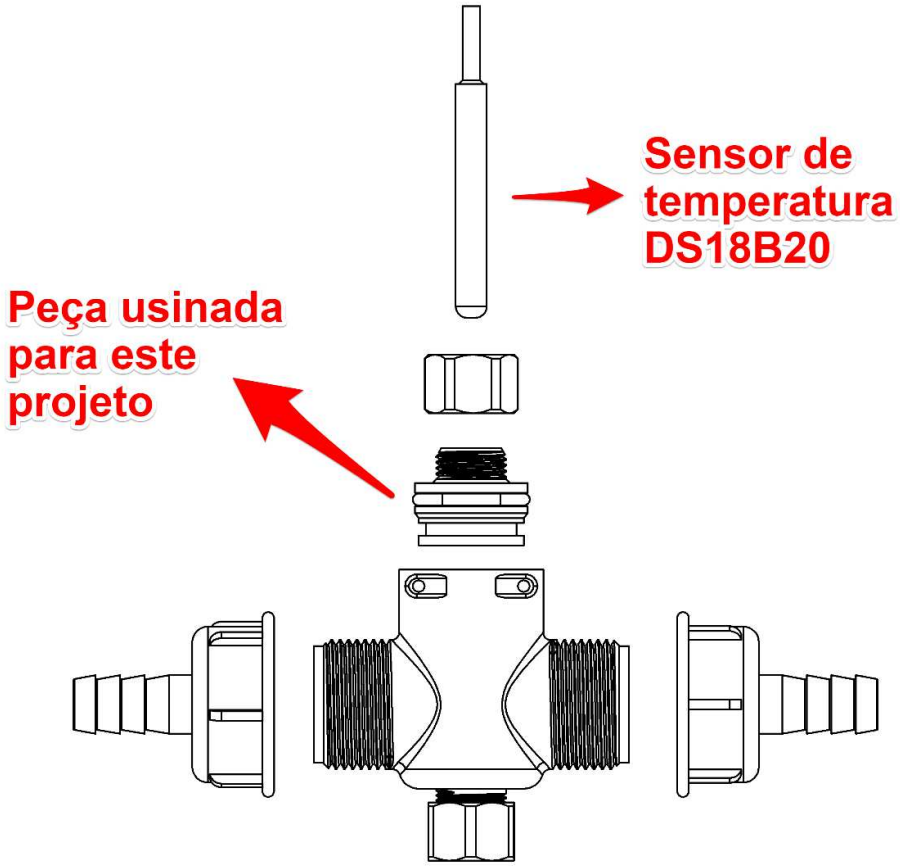
Saindo da unidade de ordenha, o leite deve chegar até a entrada do sensor de fluxo. Após passar pelo sensor de fluxo, deve entrar em contato com o sensor de temperatura. Para fazer a instalação do sensor de temperatura foi utilizada uma peça em nylon, e para acoplar o sensor de temperatura a mesma foi usinada uma peça complementar, conforme indicado pelas Figuras 37 e 38. Aqui houve a preocupação com a vedação das partes, garantindo que não houvessem vazamentos.

Figura 37 – Desenho da instalação do sensor DS18B20.



Fonte: O autor, 2019.

Figura 38 – Vista explodida da instalação do sensor DS18B20.



Fonte: O autor, 2019.

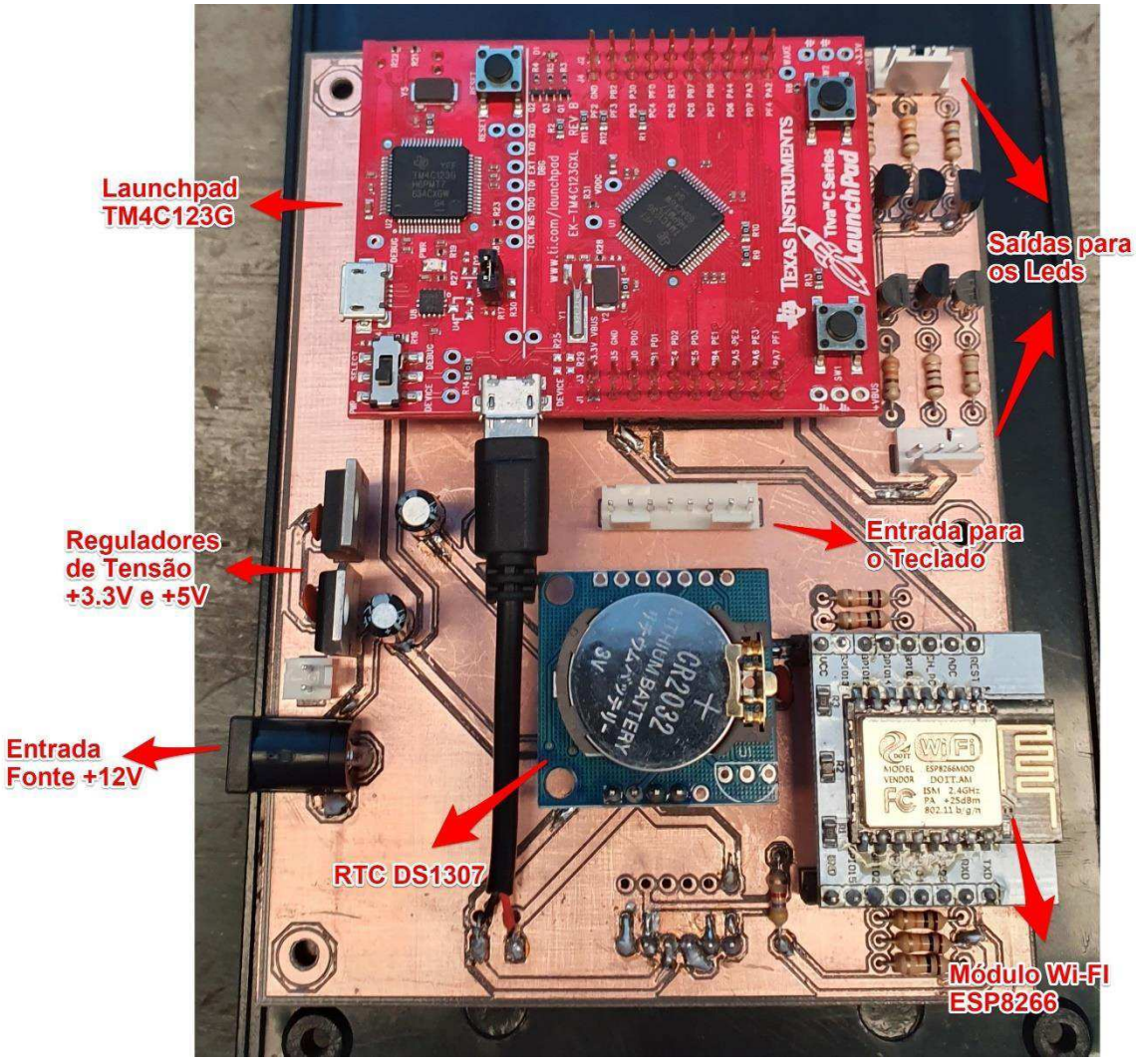
Após passar pelo sensor de temperatura, o leite ordenhado deve seguir para a linha de leite. A Figura 39 apresenta os sensores de fluxo e temperatura, prontos para utilização no sistema.

Figura 39 – Sensores de temperatura e fluxo prontos para utilização.

Fonte: O autor, 2019.

Para a implementação de todas as partes eletrônicas foi confeccionada uma placa de circuito impresso (PCI). Nesta PCI é conectada a LaunchPad TM4C123G, além dos módulos Wi-Fi ESP8266 e do RTC DS1307. Esta placa também possui um conector para a entrada da fonte de alimentação 12V, e os reguladores de tensão, que irão alimentar os componentes do circuito, além de possuir conector onde é feita a ligação dos sensores instalados na ordenhadeira. A Figura 40 apresenta a placa de circuito impresso, com suas partes identificadas.

Figura 40 – PCI montada com componentes identificados.



Fonte: O autor, 2019.

Tendo a PCI montada com todos seus componentes, esta foi instalada dentro de uma caixa plástica. O teclado matricial usado para identificação também foi instalado nesta caixa, e um adesivo com o layout do teclado e a descrição dos LEDs de status foi confeccionado.

A Figura 41 apresenta a caixa que contém as partes eletrônicas. Nesta figura podem ser vistos os LEDs de status, o teclado para identificação, a entrada da fonte de alimentação, além de uma chave liga/desliga, instalada na caixa.

Figura 41 – Módulo para ordenhadeira finalizado.

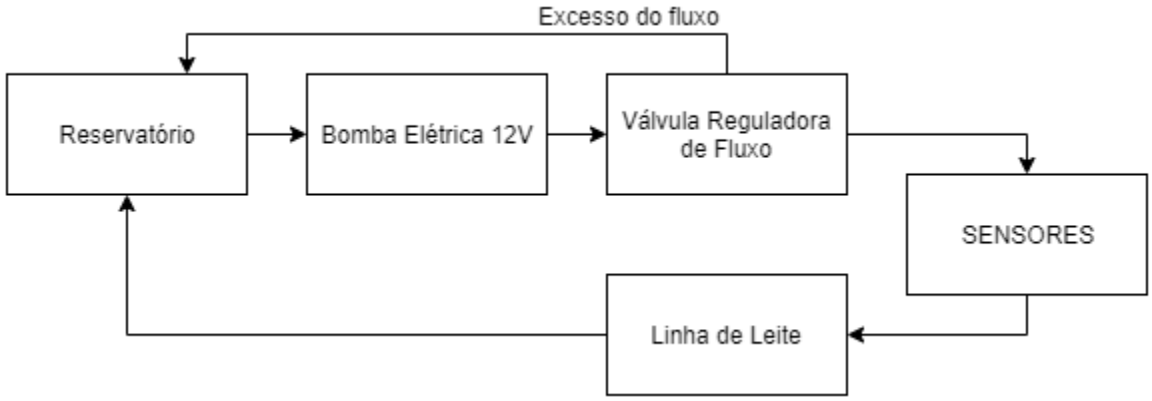
Fonte: O autor, 2019.

3.4.2 Protótipo para Simulação de Ordenha

Devido ao fato de o sensor escolhido apresentar dificuldades para instalação em um ambiente real de ordenha, foi decidido pela construção de um protótipo, através do qual pudessem ser feitas simulações do processo de ordenha.

Para simular a ordenha é utilizado um reservatório contendo água, uma bomba para fazer a circulação do líquido no sistema e uma válvula reguladora de fluxo. O fluxo passa então pelos sensores instalados no protótipo, e chega até uma linha de leite. A água sai da linha de leite e retorna para o reservatório. O diagrama de blocos do protótipo é apresentado na Figura 42.

Figura 42 – Diagrama de blocos do protótipo.



Fonte: O autor, 2019.

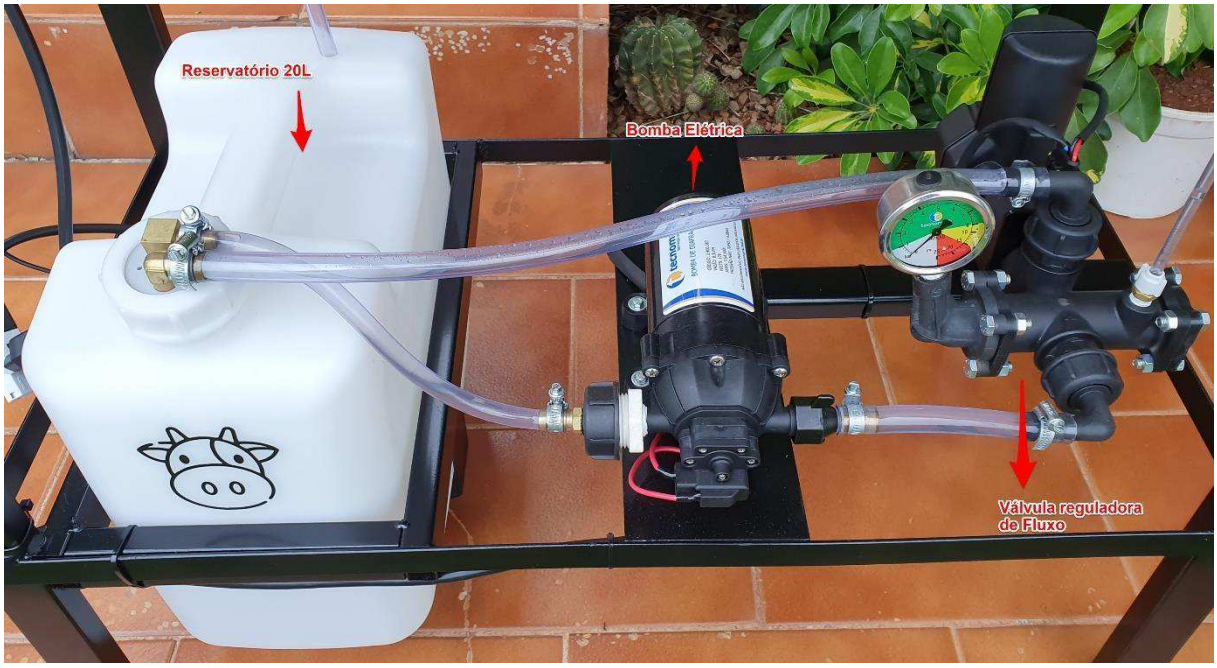
A água utilizada na simulação fica armazenada em um reservatório de 20 litros. Para fazer a circulação, é utilizada uma bomba elétrica. Esta bomba é alimentada com 12V, tendo um consumo máximo de 7,5A. A mesma consegue produzir um fluxo máximo de 8,3 litros por minuto, em condições ideais, e pode operar com uma pressão máxima de 4,8 bar.

Para que o fluxo que passa pelos sensores possa ser regulado, fez-se o uso de uma válvula reguladora de fluxo. Esta válvula também é alimentada com 12V, e pode consumir um máximo de 6A. A regulagem do fluxo é feita através de uma chave, que aplica 12V a um dos dois terminais disponíveis na válvula.

Para fazer a alimentação, tanto da bomba elétrica, quanto da válvula reguladora, foi utilizada uma fonte chaveada 12V/30A.

A Figura 43 apresenta o reservatório, a bomba e a válvula reguladora, já conectadas entre si.

Figura 43 – Reservatório, bomba elétrica e válvula reguladora de fluxo.



Fonte: O autor, 2019.

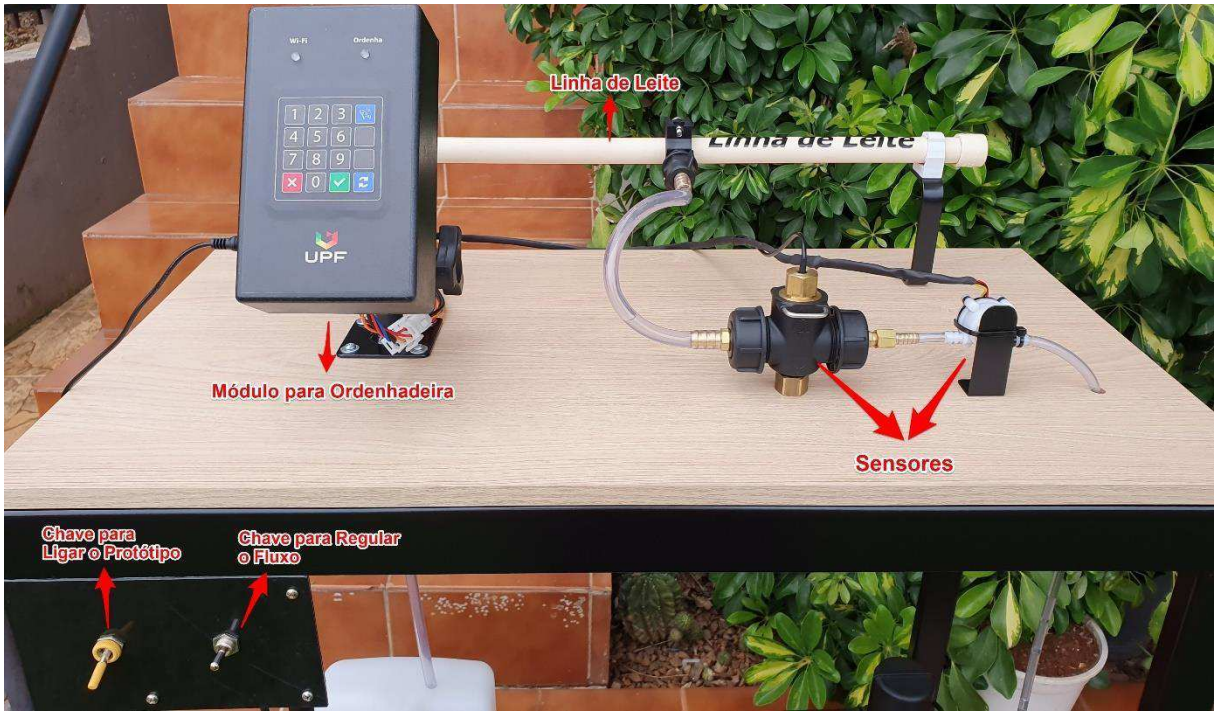
Para fazer a instalação de todas as partes, foi criada uma estrutura de ferro. Esta estrutura foi inicialmente desenhada utilizando o software SolidWorks, e após enviada para fabricação, em empresa do ramo. A Figura 44 apresenta o projeto desta estrutura.

Com a estrutura pronta, foram instaladas as peças e feitas as conexões hidráulicas. Além das partes previamente apresentadas, foi também instalado um tubo, para simulação de uma linha de leite. Além disso, também foi fabricado um suporte para fazer a fixação do módulo com as partes eletrônicas nesta bancada de simulação. A Figura 45 apresenta a parte superior desta bancada, enquanto a Figura 46 demonstra todo o protótipo para simulação pronto.

Figura 44 – Desenho da estrutura do protótipo.

Fonte: O autor, 2019.

Figura 45 – Parte superior da bancada de simulação.



Fonte: O autor, 2019.

Figura 46 – Protótipo para simulação finalizado.

Fonte: O autor, 2019.

3.5 APLICAÇÃO WEB

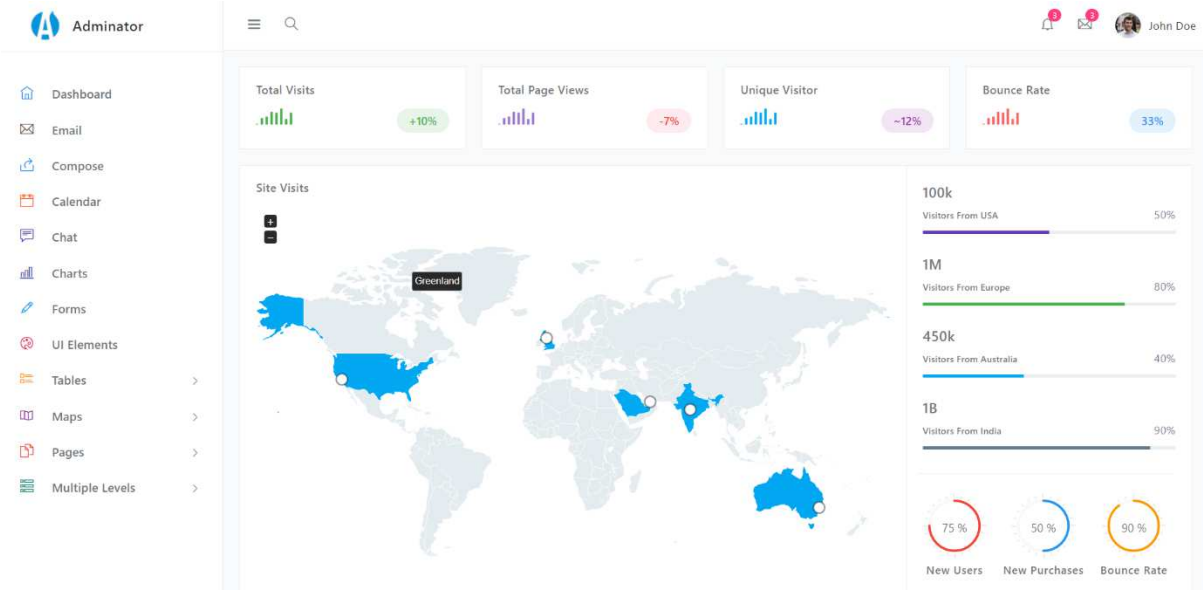
Com o módulo para ordenhadeira pronto, e o protótipo pronto para fazer a coleta de dados dos processos de ordenha, parte-se para o desenvolvimento da aplicação web. Através desta aplicação, os dados gerados durante a simulação podem ser analisados. A partir desta análise, o produtor tem o acompanhamento de sua produção, podendo tomar medidas, quando julgar necessário, para uma melhor rentabilidade. Esta também permite o cadastro e alteração dos animais salvos no banco de dados, permitindo assim o gerenciamento do rebanho.

A escolha por uma aplicação web foi feita, pois a mesma tem um desenvolvimento fácil, além de poder ser aberta facilmente em diversos dispositivos. Como esta opera diretamente no navegador, pode ser aberta em qualquer computador, sem a necessidade de instalação de qualquer software terceiro. Isso também possibilita que a aplicação seja acessada por dispositivos móveis com facilidade, independente da marca ou modelo do aparelho.

Para iniciar o desenvolvimento da aplicação web, fez-se uso do modelo Adminator, disponibilizado pela companhia Colorlib. Este modelo está disponível gratuitamente, e ao fazer o download do mesmo, todos os recursos e ferramentas necessárias para o seu desenvolvimento são automaticamente instalados.

O modelo utilizado tem seu desenvolvimento feito utilizando as linguagens HTML, CSS e JavaScript, e assim como a API, este também é executado no ambiente Node.js. A Figura 47 apresenta o modelo original, conforme disponibilizado.

Figura 47 – Aplicação Adminator, conforme disponibilizada.



Fonte: Adaptado de Colorlib, 2019.

Feita a instalação das ferramentas de desenvolvimento, é iniciado o desenvolvimento da aplicação. Inicia-se pela limpeza do modelo, removendo todos os recursos que não são necessários e não serão utilizados. Após isso, são modificados os arquivos de estilo, utilizando a linguagem CSS, modificando cores e temas, alterando assim a aparência da aplicação.

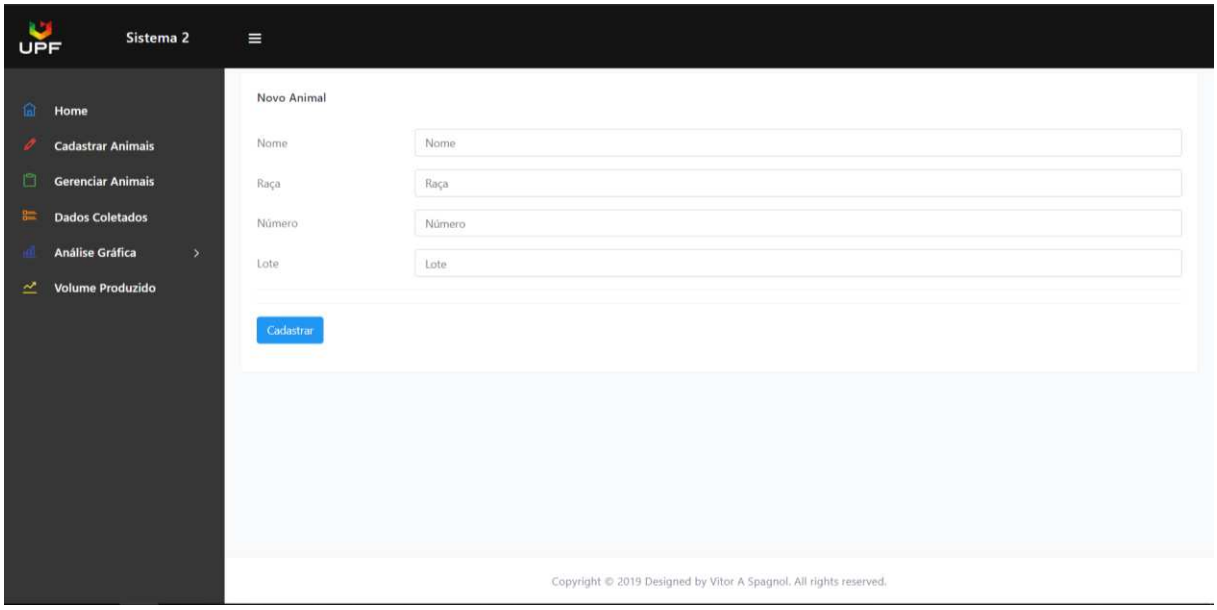
Com as configurações iniciais realizadas, parte-se então para a criação e implementação das páginas que irão compor e dar funcionalidade à aplicação.

3.5.1 Página Cadastrar Animais

Esta página é criada para possibilitar ao usuário a inclusão de novos animais no banco de dados.

A estrutura da página é criada utilizando a linguagem HTML, fazendo uso de um formulário, onde o usuário entra com o nome do animal a ser cadastrado, raça do mesmo, número de identificação e número do lote. A Figura 48 apresenta a página pronta.

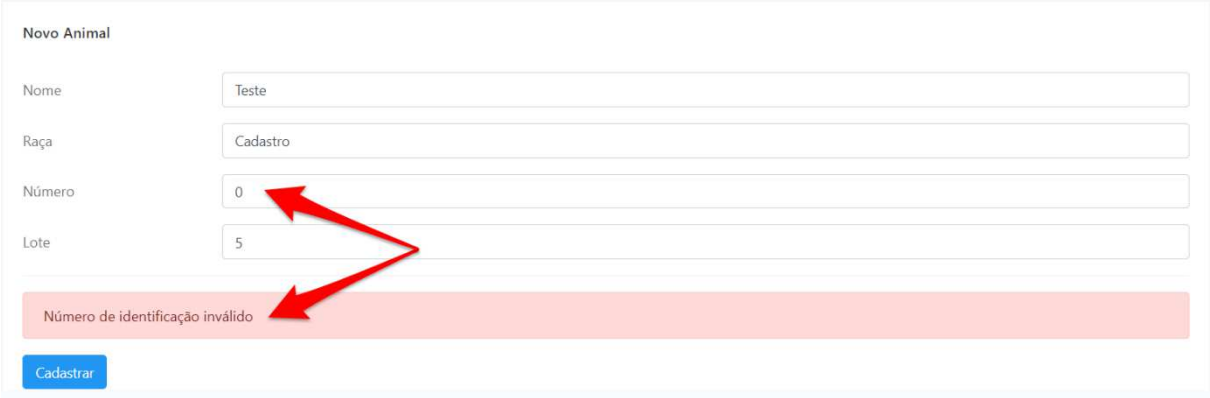
Figura 48 – Página Cadastrar Animais.



Fonte: O autor, 2019.

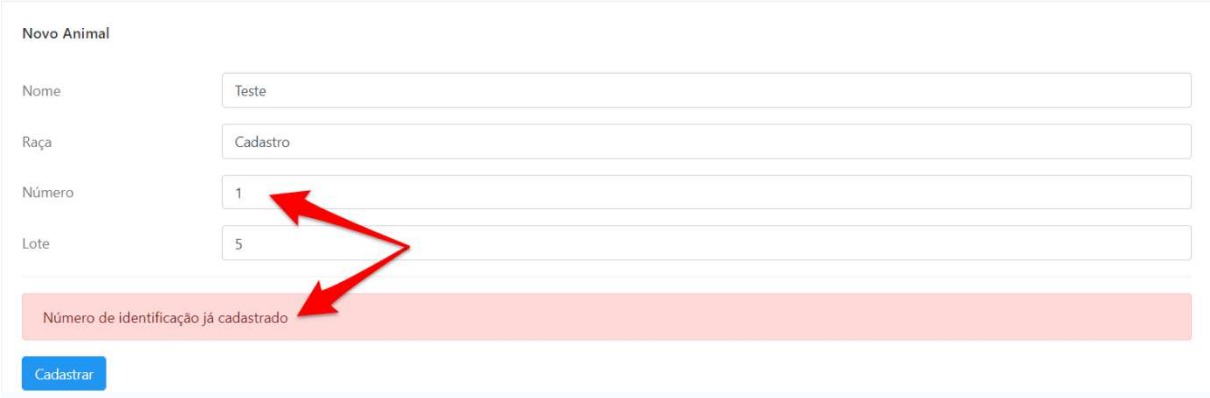
Ao clicar no botão Cadastrar, é chamada uma função JavaScript desenvolvida. Esta função é responsável por pegar os dados informados pelo usuário, e enviá-los para o banco de dados. O envio é feito gerando uma requisição do tipo POST para o endereço /newAnimal, e usando os dados informados como corpo da requisição. A API fará então a verificação dos dados, e responderá, informando sucesso ou erro. De acordo com a resposta da API, é informado ao usuário se o animal foi ou não cadastrado. As Figuras 49, 50 e 51 apresentam as mensagens de erro e sucesso possíveis durante o cadastro.

Figura 49 – Mensagens de erro no cadastro de animais, número inválido.



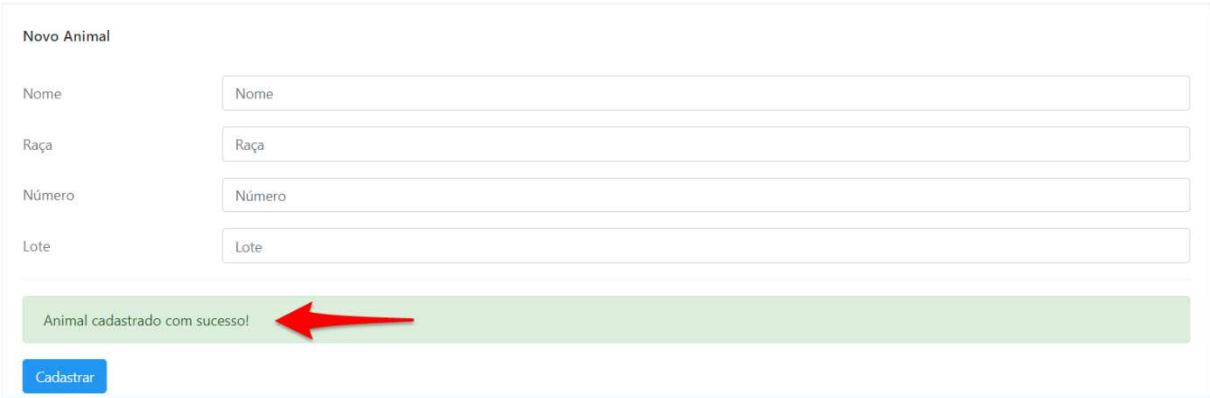
Fonte: O autor, 2019.

Figura 50 – Mensagens de erro no cadastro de animais, número já existente.



Fonte: O autor, 2019.

Figura 51 – Mensagens de sucesso no cadastro de animais.



Fonte: O autor, 2019.

3.5.2 Página Gerenciar Animais

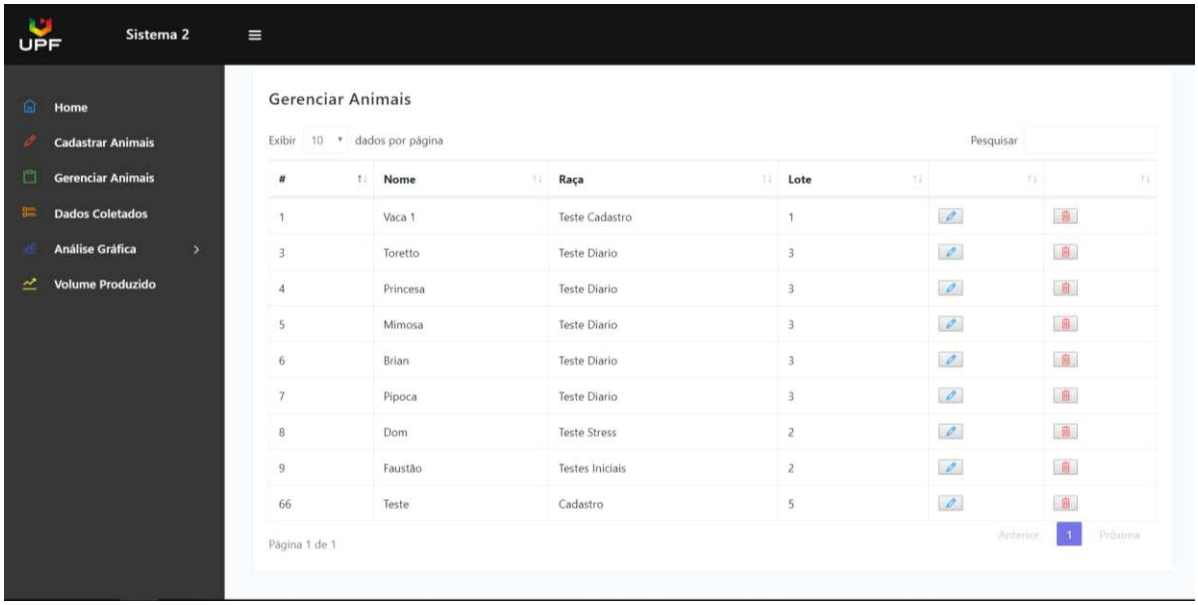
Através desta página, o usuário pode visualizar os animais que se encontram cadastrados no banco de dados. Aqui, também, é possível fazer a alteração dos animais cadastrados, além de poder fazer a exclusão dos mesmos.

Esta página é implementada na forma de uma tabela. Para fazer a construção desta tabela é utilizado o pacote Datatables para JavaScript. Este módulo facilita a criação de tabelas, além de adicionar funções interessantes a estas, como permitir que o usuário ordene os dados de diferentes formas, pesquise por dados específicos, e faça a exibição dos mesmos em diferentes páginas.

A renderização da tabela de animais é feita através de uma função em JavaScript. Esta função é ativada no momento em que a página é carregada.

Dentro da função para montar a tabela, o primeiro passo é a geração de uma requisição do tipo GET para o endereço /animals/. Ao receber esta requisição, a API retorna todos os dados escritos no arquivo `animais.json`, ou seja, retorna as informações sobre os animais cadastrados no banco de dados. Estes dados são então passados como parâmetro para o pacote Datatables, que cria a tabela. A Figura 52 apresenta a página Gerenciar Animais.

Figura 52 – Página Gerenciar Animais.



Fonte: O autor, 2019.

Nesta página, pode ser visto que para cada animal cadastrado existem dois botões, um botão para excluir e um para editar.

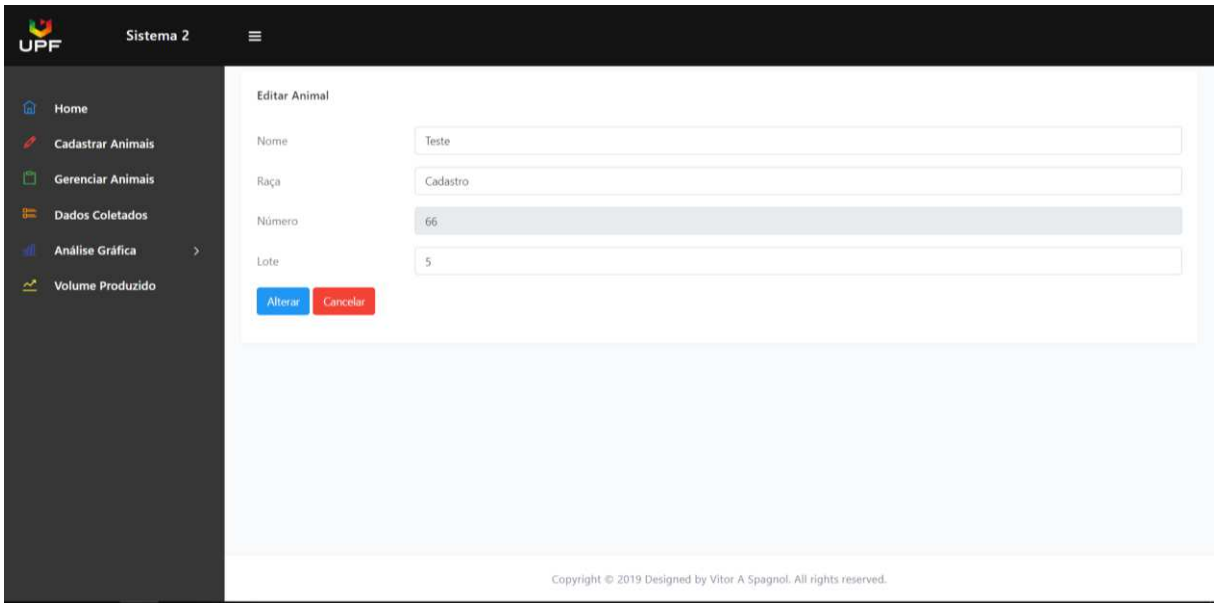
Ao pressionar o botão Excluir, é chamada uma segunda função JavaScript. Esta função pega o número de identificação do animal que está na mesma linha do botão pressionado. Este número de identificação é então usado na geração de uma requisição do tipo DELETE para o endereço /deleteAnimal/:id. Através desta requisição, a API excluirá o animal do banco de dados. A página Gerenciar Animais é então recarregada, agora já sem o animal recentemente excluído.

Pressionando o botão Editar, uma terceira função JS é chamada. Assim como a função do botão excluir, esta pega o número de identificação que está na mesma linha do botão pressionado, e redireciona o usuário para a página de edição.

3.5.2.1 Página Edição de Animais

Esta página, da mesma maneira que na página Cadastrar Animais, é implementada com o uso de um formulário. A Figura 53 apresenta página para edição dos animais cadastrados.

Figura 53 – Página Editar Animais.



Fonte: O autor, 2019.

Ao ser carregada, esta página exibe quatro campos, previamente preenchidos com os dados atuais, que podem ser alterados pelo usuário pelos valores desejados. Dos quatro dados relacionados ao animal, o número de identificação é o único que não pode ser alterado. Ao final do formulário são adicionados dois botões, Alterar e Cancelar.

Pressionando o botão cancelar, todos os dados do formulário são descartados, nenhuma modificação é feita no banco de dados, e aplicação é redirecionada para a página Gerenciar Animais.

Caso o botão Alterar seja pressionado, chama-se uma função JS que realiza as alterações. Esta função trabalha de forma similar a função utilizada no cadastro de animais, pegando os valores passados nos campos, e gerando uma requisição para a API. A requisição deve ser do tipo PUT, para o endereço /modifyAnimal/:id. Recebendo esta requisição a API executará a rotina de alteração do banco de dados. Após isso, a aplicação retorna para a página Gerenciar Animais, agora já com o animal atualizado.

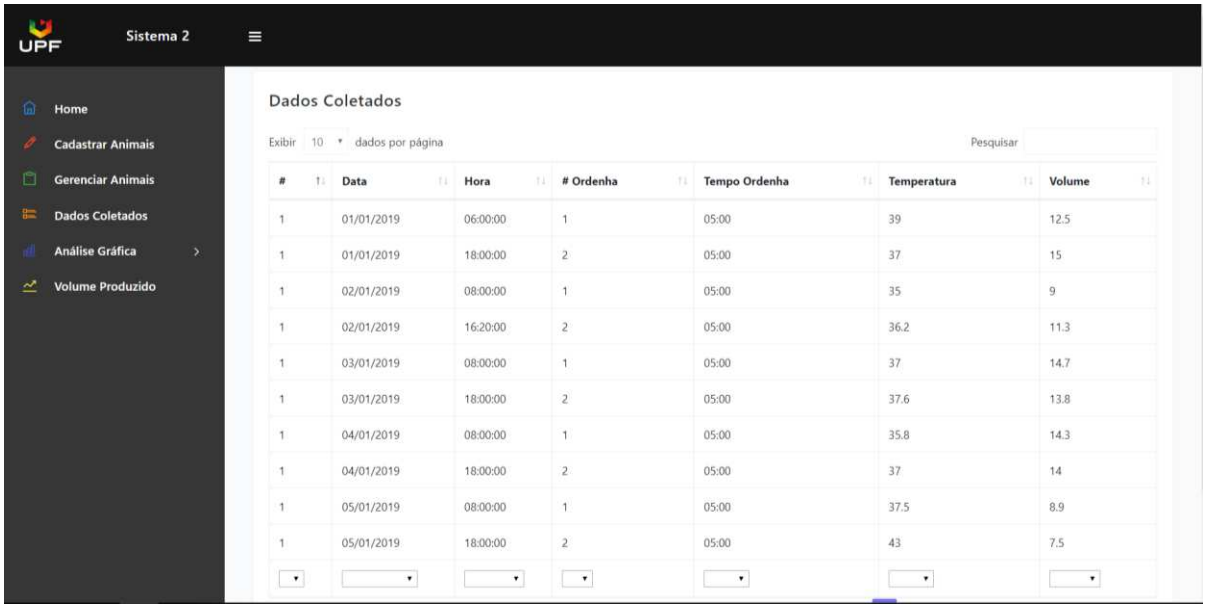
3.5.3 Página Dados Coletados

Através desta página podem ser acessados todos os dados já coletados no sistema. Novamente esta página é implementada através de uma tabela, fazendo uso para isso do pacote Datatables.

Para criar esta tabela é usada uma função JavaScript, função esta que é automaticamente chamada no momento em que a página é carregada.

Dentro desta função, são pegos todos os dados já coletados, disponíveis no banco de dados. Para isso, primeiro é gerado uma requisição do tipo GET para o endereço /idCadastrados, pegando o número de identificação de todos os animais do sistema. Após, usando um laço *for*, várias requisições do tipo GET são geradas, para o endereço /animalData/:id, onde :id assume o valor dos números de identificação recebidos na requisição /idCadastrados. Ao final da execução do laço, a API terá retornado todos os dados disponíveis no banco. Esses dados são então usados pelo módulo Datatables para criar a tabela de dados coletados. A Figura 54 apresenta esta página.

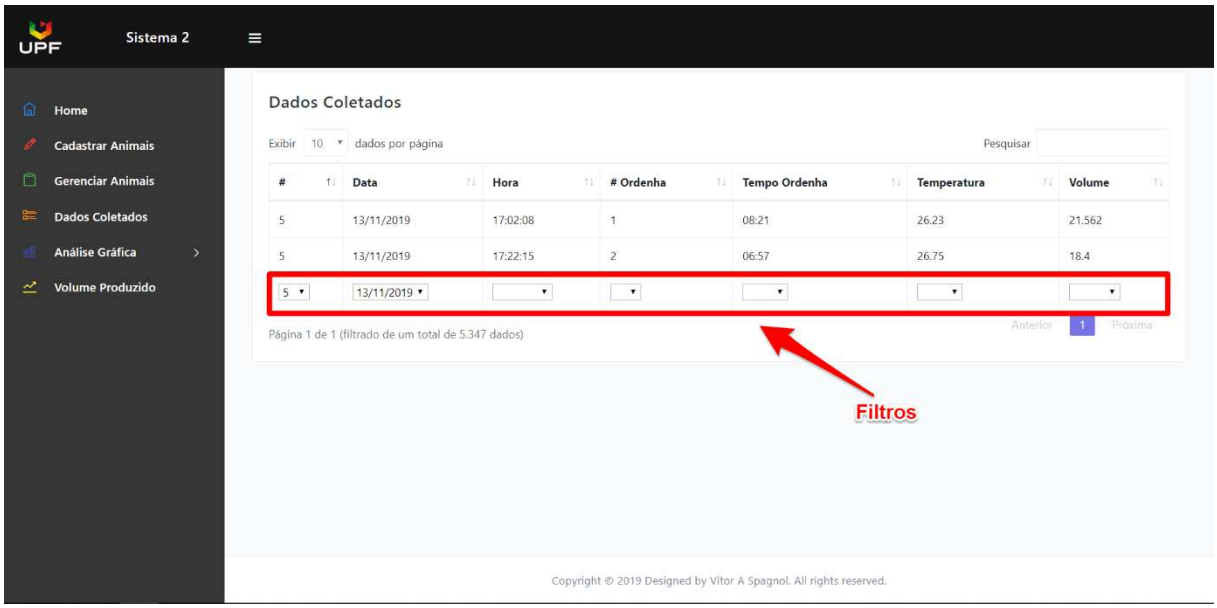
Figura 54 – Página Dados Coletados.



Fonte: O autor, 2019.

Para permitir que dados específicos fossem encontrados mais facilmente, foi implementado um filtro na tabela gerada. Através deste, é possível filtrar por qualquer dado coletado disponibilizado na tabela. A implementação deste filtro foi feita utilizando linhas de código disponibilizadas pelos criadores do módulo Datatables. A Figura 55 apresenta estes filtros em funcionamento. Nela, os dados coletados foram filtrados, sendo exibidos apenas os dados referentes ao animal número 5, e apenas os que foram coletados no dia 13/11/2019.

Figura 55 – Filtros da página Dados Coletados em funcionamento.



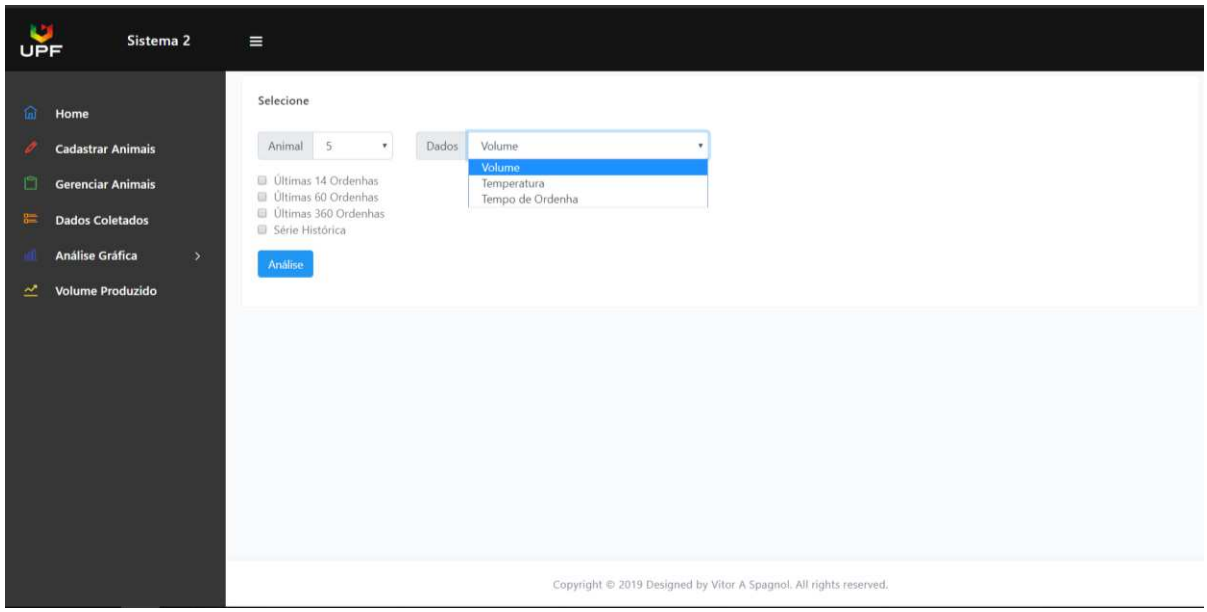
Fonte: O autor, 2019.

3.5.4 Página Análise Gráfica

Esta página foi pensada para permitir que o usuário possa analisar os dados de ordenha através de gráficos. Para a criação dos gráficos desta página, fez-se uso do módulo Chart.js. Este é um módulo simples e flexível que permite a renderização de gráficos com a linguagem JavaScript. Através do Chart.js é possível a criação de diversos tipos de gráficos, como gráficos de linhas e barras por exemplo.

A Figura 56 apresenta a página Análise Gráfica.

Figura 56 – Página Análise Gráfica.



Fonte: O autor, 2019.

No topo desta página foram adicionados campos de seleção. No campo Animal, o usuário seleciona o número de identificação do animal que deseja analisar. Para saber os números de identificação que podem ser selecionados, ao acessar esta página uma função JS é chamada. Esta uma função gera uma requisição do tipo GET para o endereço /idCadastrados, recebendo os números de identificação cadastrados no banco e renderizando-os na caixa de seleção.

O campo Dados é usado para selecionar qual dado pretende-se usar. Aqui é possível selecionar entre Volume, Temperatura e Tempo de Ordenha.

Após selecionar o número de identificação e os dados utilizados para análise, ao clicar no botão Análise, uma nova função JS é chamada, responsável por criar os gráficos. Neste momento, quatro gráficos serão renderizados, um gráfico contendo as últimas 14 ordenhas,

um para as últimas 60 ordenhas, um para as últimas 360 e um para todos os dados já coletados. Estes números de ordenhas a serem exibidas foi definido, pois a maioria dos produtores realizam duas ordenhas por dia. Desta maneira, na maioria dos casos, as últimas 14 ordenhas representarão a última semana, as últimas 60 ordenhas representarão o último mês, e as últimas 360 os últimos 6 meses.

Ao chamar a função JS que cria os gráficos, a primeira ação realizada pela mesma é remover quaisquer gráficos que já estejam renderizados na tela. Isto é necessário para evitar interferências que podem ocorrer entre os gráficos, quando mais de um é criado. Após, a função pega o número de identificação selecionado pelo usuário, e gera uma requisição do tipo GET para o endereço /animalData/:id, pegando todos os dados já coletados para este animal. Os dados recebidos são então separados, em dados de volume, dados de temperatura e dados de tempo.

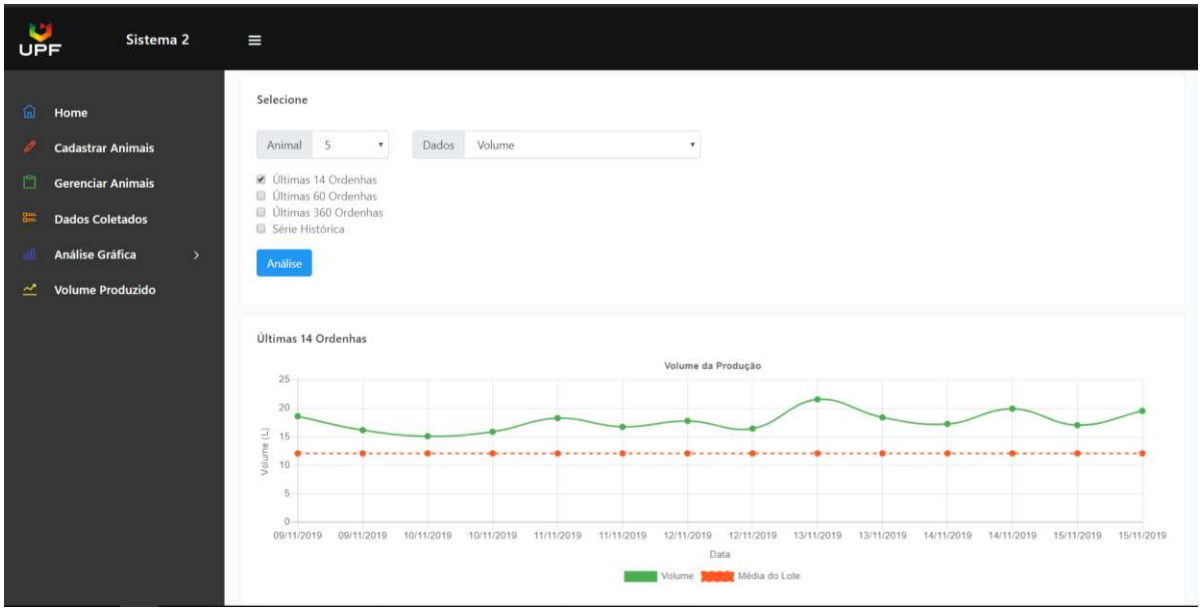
Tendo os dados separados por tipo, agora são separados os últimos 360 dados de cada tipo, para criar o gráfico das últimas 360 ordenhas. Caso não haja 360 dados coletados até o momento, todos os dados disponíveis são utilizados. Este processo é repetido para as últimas 60 e últimas 14 ordenhas. Os dados estão agora prontos para serem utilizados na criação dos gráficos.

Além dos dados coletados, nos gráficos criados também são exibidos os valores médios coletados para o lote em que o animal em análise se encontra. Para pegar estes dados médios, quatro requisições GET são geradas em sequência, para os endereços /loteAvg/:lote, /lote360/:lote, /lote60/:lote e /lote14/:lote. Os dados recebidos dessas requisições são separados em variáveis, e estão prontos para serem usados em seus respectivos gráficos.

Agora, com todos os dados preparados, estes são passados como parâmetro para o módulo Chart.js, que faz a renderização dos gráficos. Apenas o gráfico selecionado é renderizado, assim, se o usuário seleciona o gráfico de Volume, apenas este é renderizado. Os dados de Tempo e Temperatura também são recebidos da API, porém, estes não são utilizados.

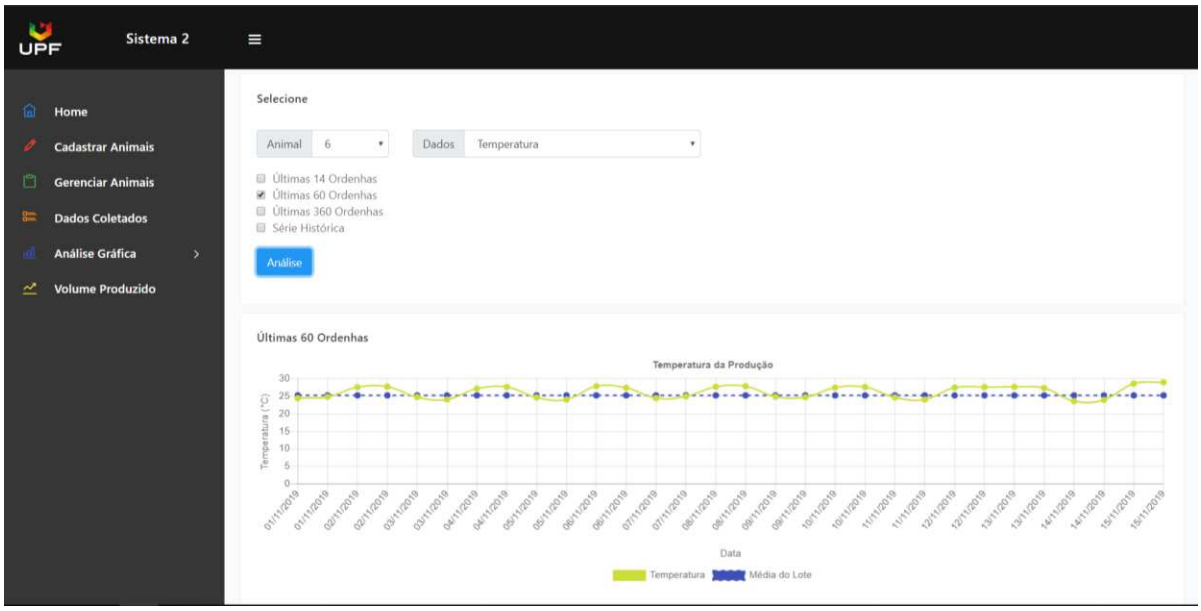
A Figuras 57, 58 e 59 apresentam alguns dos gráficos gerados. Na Figura 57 é demonstrado o gráfico de volume das últimas 14 ordenhas do animal número 5. A Figura 58 apresenta o gráfico de temperatura das últimas 60 ordenhas para o animal número 6, e a Figura 59 apresenta o gráfico de duração da ordenha, de todos os dados já coletados para o animal número 6.

Figura 57 – Gráfico de Volume, gerado na página Análise Gráfica.



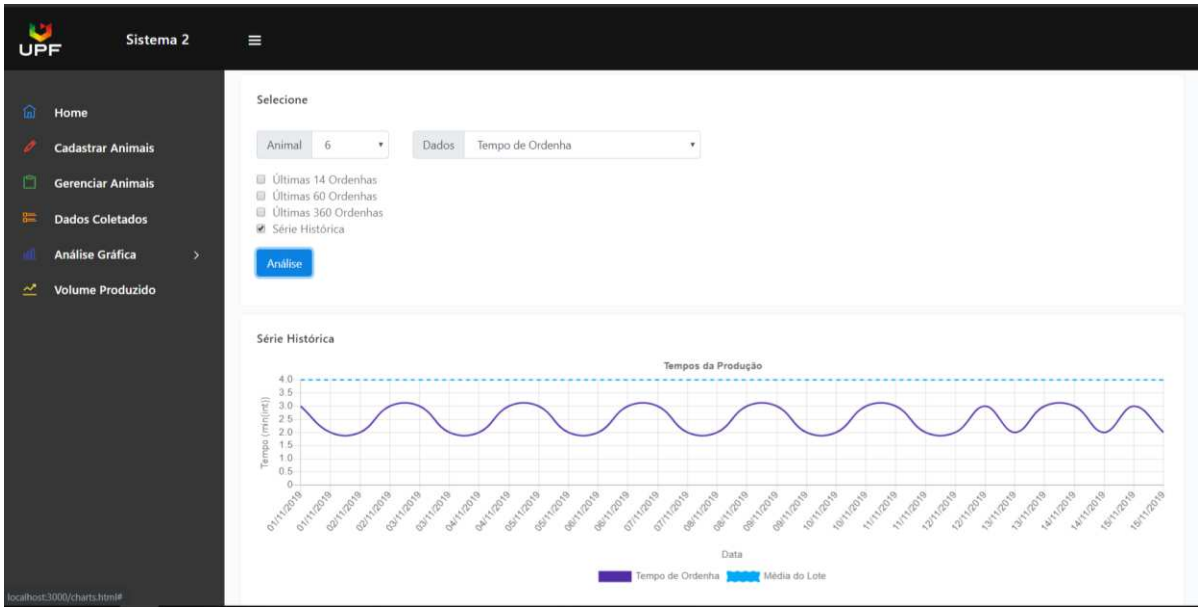
Fonte: O autor, 2019.

Figura 58 – Gráfico de Temperatura, gerado na página Análise Gráfica.



Fonte: O autor, 2019.

Figura 59 – Gráfico do Tempo de Ordenha, gerado na página Análise Gráfica.



Fonte: O autor, 2019.

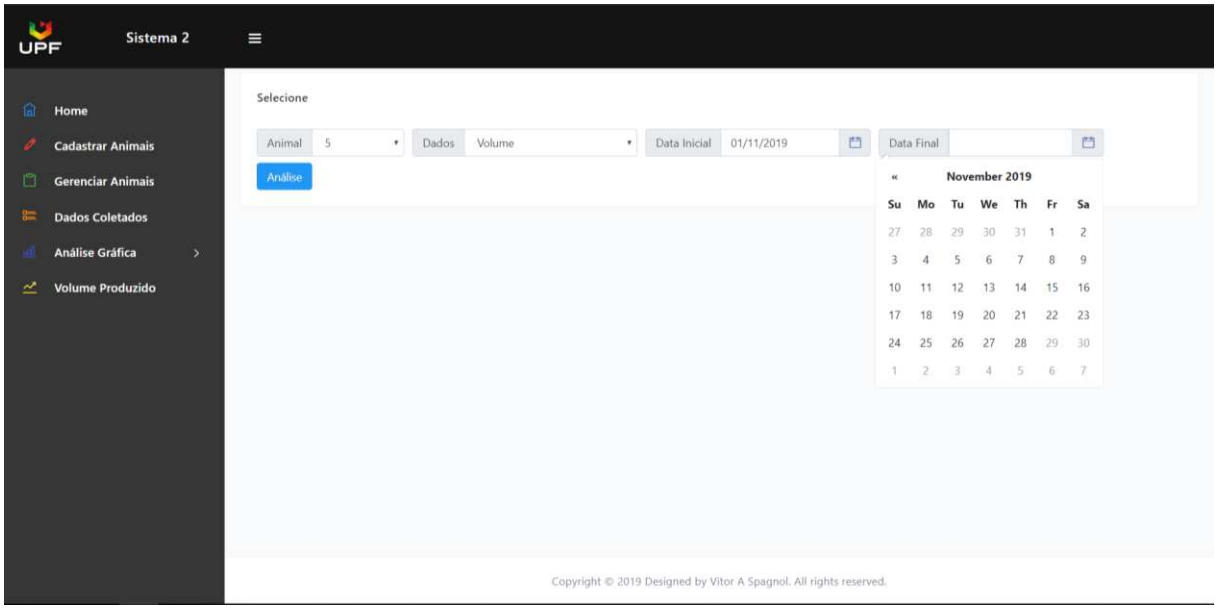
Pensando em facilitar o uso do sistema, foram adicionadas checkboxes na seleção dos gráficos, assim, o usuário pode selecionar apenas os gráficos que deseja ver. Para a implementação destas checkboxes, a função JS que renderiza os gráficos também precisou sofrer algumas modificações. Agora, além dos procedimentos realizados anteriormente, esta função verifica quais checkboxes estão marcadas, fazendo a renderização apenas dos gráficos que foram selecionados.

3.5.5 Página Análise Gráfica Personalizada

Com a implementação da análise personalizada, permite-se que o usuário tenha maior liberdade na criação dos gráficos.

Na seleção destes gráficos, além dos dados selecionados na análise normal, previamente apresentada, também deve ser selecionado o período de análise, informando uma data inicial e uma data final. A Figura 60 apresenta esta página.

Figura 60 – Página Análise Gráfica Personalizada.



Fonte: O autor, 2019.

A seleção do animal e do tipo do gráfico ocorre da mesma maneira que nos gráficos predefinidos. Para a seleção das datas inicial e final, foi utilizado o pacote Bootstrap Datepicker, que adiciona um calendário as caixas de seleção, facilitando a indicação das datas.

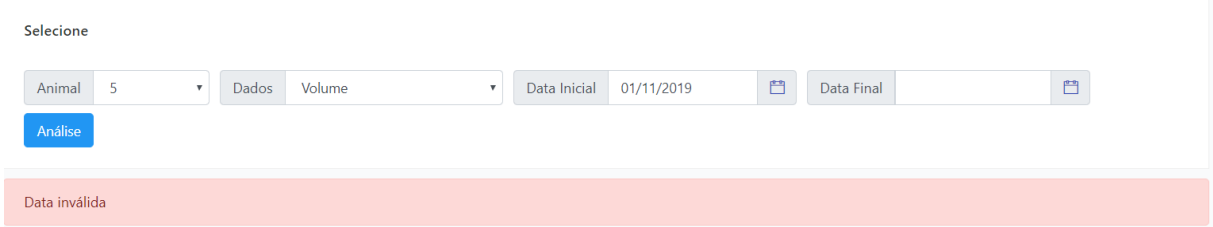
Pressionando o botão Análise, uma função JS é executada. Essa função pega os dados passados pelo usuário nos campos de seleção.

As datas inicial e final informadas são convertidas para dias, de acordo com a Equação 7.

$$dias = ano \cdot 365 + mês \cdot 30 + dia \tag{7}$$

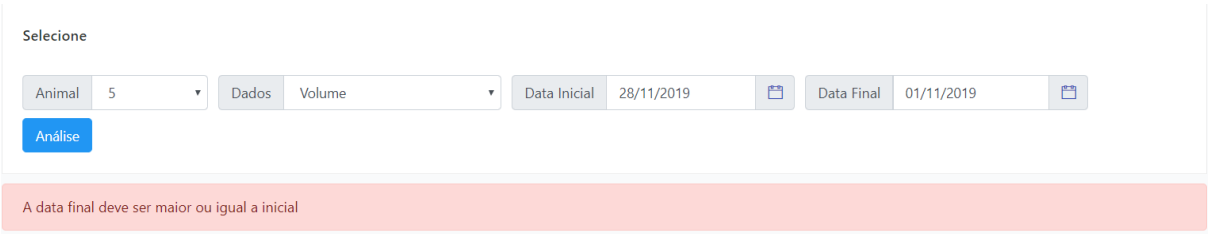
As datas informadas pelo usuário são então avaliadas, verificando se as mesmas são válidas. Caso haja algum erro com as datas, uma mensagem de erro é exibida para o usuário. As Figuras 61 e 62 apresentam as mensagens de erro possíveis.

Figura 61 – Mensagem de erro nas datas informadas, data inválida.



Fonte: O autor, 2019.

Figura 62 – Mensagem de erro nas datas informadas, ordem inválida.



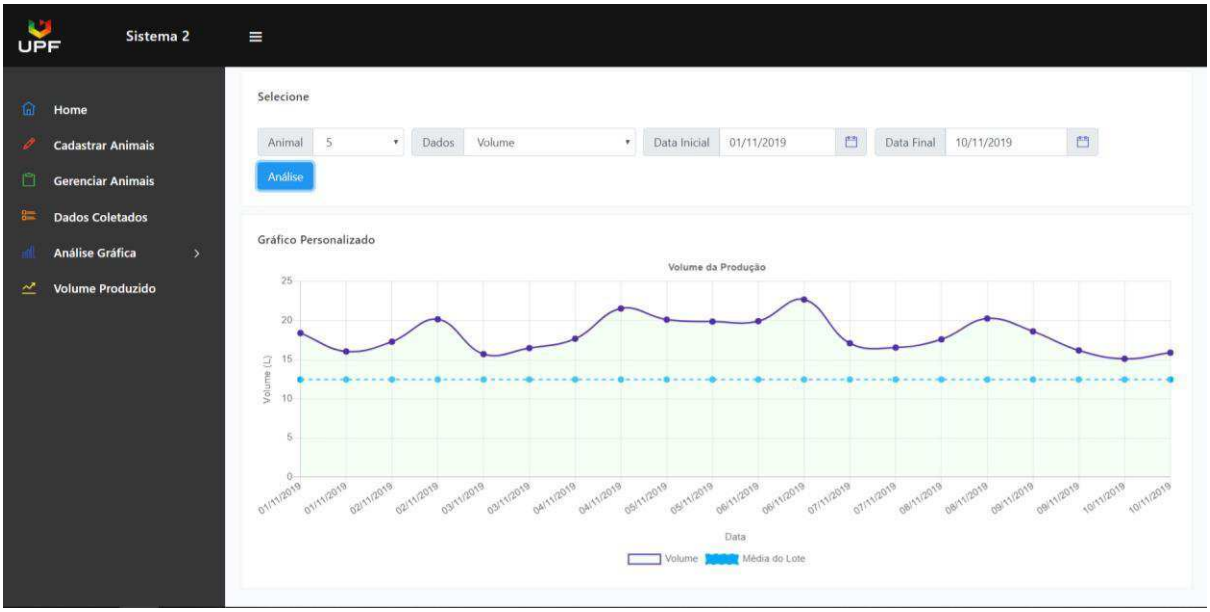
Fonte: O autor, 2019.

Verificando que as datas informadas são válidas, os dados selecionados pelo usuário são enviados para a API através de uma requisição do tipo POST, para o endereço customData/:id. A API processa a requisição e devolve os dados coletados dentro do período selecionado. Os dados recebidos são então separados em dados de volume, de temperatura e de tempo.

Após, uma nova requisição do tipo POST é gerada para o endereço /customAvg/:id, que retorna como resposta as médias de volume, temperatura e tempo para o lote em que está o animal em análise.

Tendo todos os dados necessários, a ferramenta Chart.js é usada para fazer a renderização do gráfico, sendo que apenas o gráfico selecionado é renderizado, e os dados recebidos não utilizados são ignorados. A Figura 63 apresenta um gráfico personalizado. Neste gráfico são exibidos os dados de Volume do animal número 5, coletados entre os dias 01/11/2019 e 10/11/2019.

Figura 63 – Gráfico Personalizado.

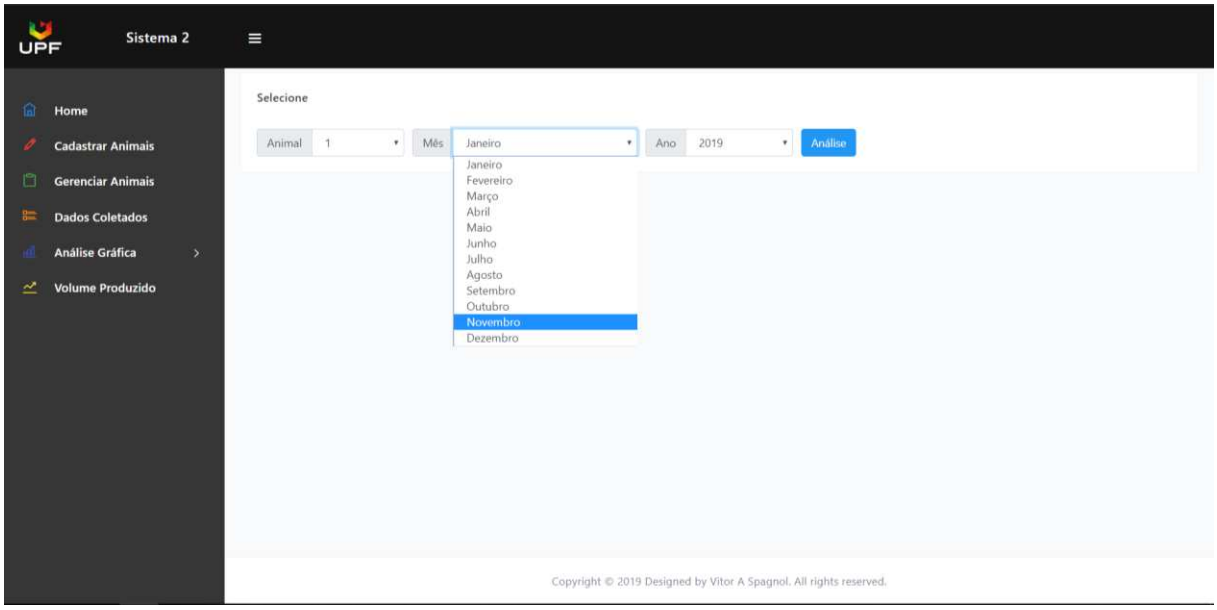


Fonte: O autor, 2019.

3.5.6 Página Volume Produzido

Esta página foi criada para permitir que o produtor tenha uma análise do volume mensal de leite produzido, permitindo uma comparação entre os animais do mesmo lote. A Figura 64 apresenta a página Volume Produzido.

Figura 64 – Página Volume Produzido.



Fonte: O autor, 2019.

No topo da página são apresentadas ao usuário as opções de seleção do animal para análise, além do mês e do ano desejados. Os números de identificação dos animais para seleção são gerados da mesma maneira feita na página Análise Gráfica.

Ao pressionar o botão análise uma função JavaScript é executada. A primeira ação desta função é remover quaisquer gráficos que estejam na tela, evitando posteriores erros. Após isso, os dados selecionados pelos usuários são utilizados na geração de uma requisição para a API, que retorna a evolução da produção ao longo do mês selecionado.

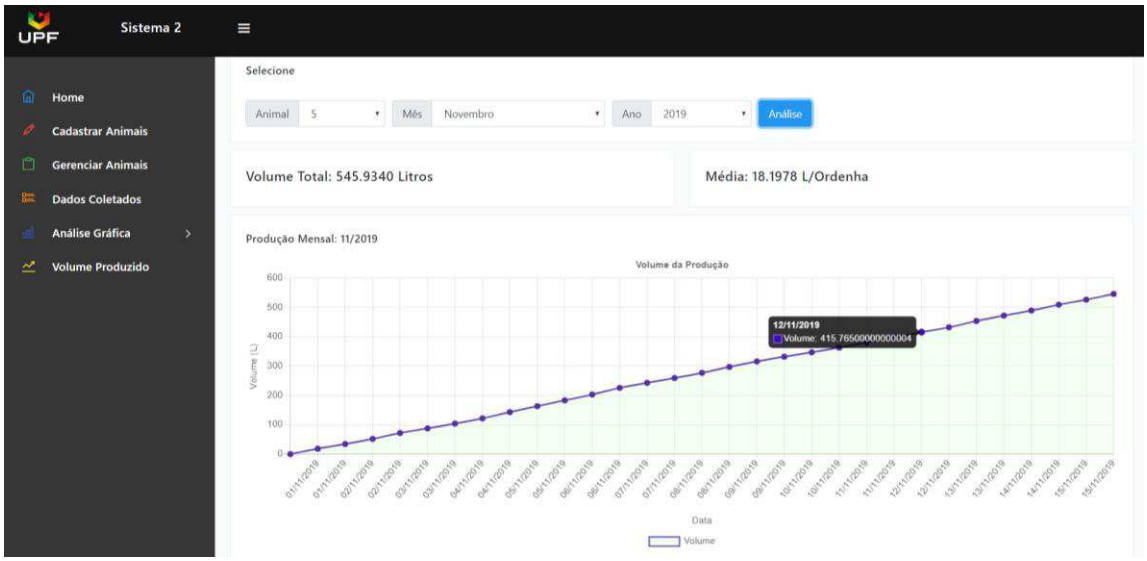
Tendo os dados da evolução da produção, é apresentado ao usuário a produção total do mês, em litros. Também é informada a produção média por ordenha, dividindo a produção total pelo número de ordenhas realizadas no mês.

Após, é utilizada a ferramenta Chart.js para renderizar o gráfico da evolução da produção ao longo do mês. Sendo este um gráfico de evolução, é esperado que o mesmo apresente um crescimento constante.

Posterior a isso, é gerada nova requisição para a API, desta vez para receber o volume médio por ordenha de cada animal cadastrado no mesmo lote que o animal em análise. Recebendo os dados, o módulo Chart.js é novamente utilizado para renderizar um segundo gráfico na mesma página. Desta vez é criado um gráfico de barras, com a produção média por ordenha para cada animal do lote. Através da análise deste gráfico, pode-se visualizar quais animais elevam ou abaixam a produção do lote.

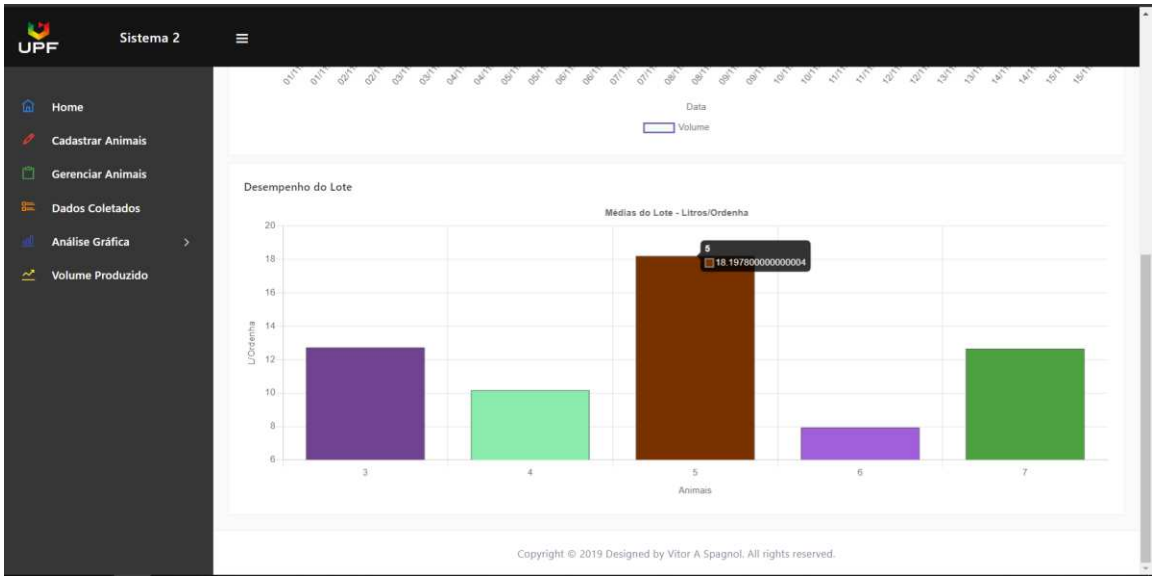
A Figuras 65 e 66 apresentam a página Volume Produzido com todos os gráficos e informações renderizadas.

Figura 65 – Gráfico da evolução da produção, na página Volume Produzido.



Fonte: O autor, 2019.

Figura 66 – Gráfico de comparação do lote, na página Volume Produzido.

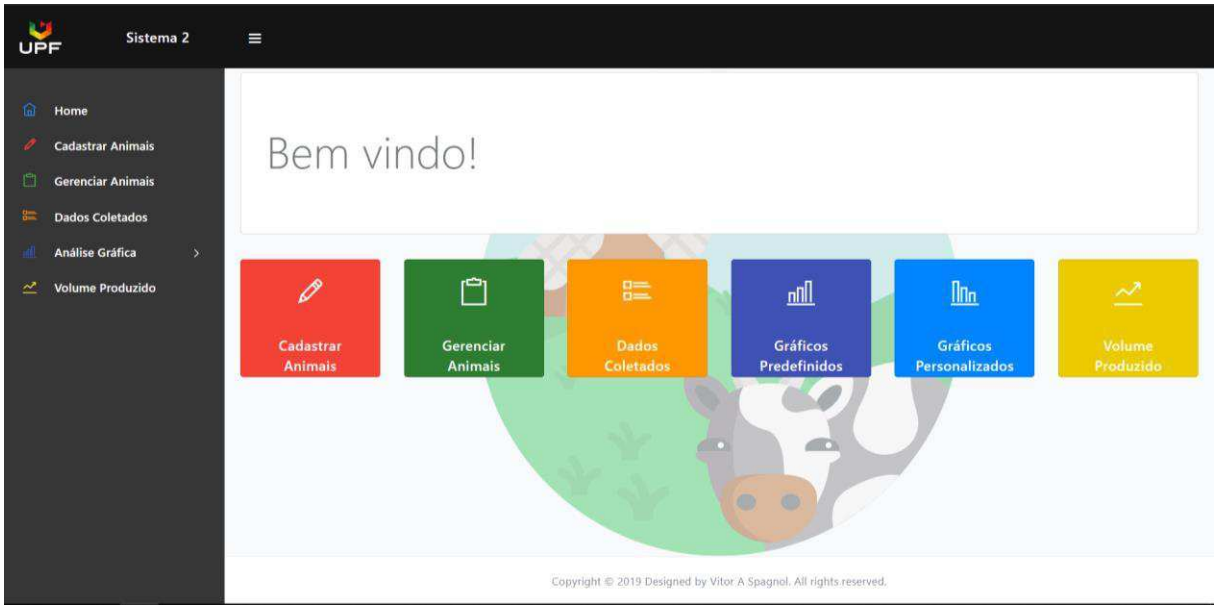


Fonte: O autor, 2019.

3.5.7 Página Home

Após a finalização de todo o resto da aplicação web, foi então pensado na página Home. Para esta página nenhuma função foi implementada, apenas foram criados atalhos, para facilitar o acesso as funções do sistema. A página Home é apresentada na Figura 67.

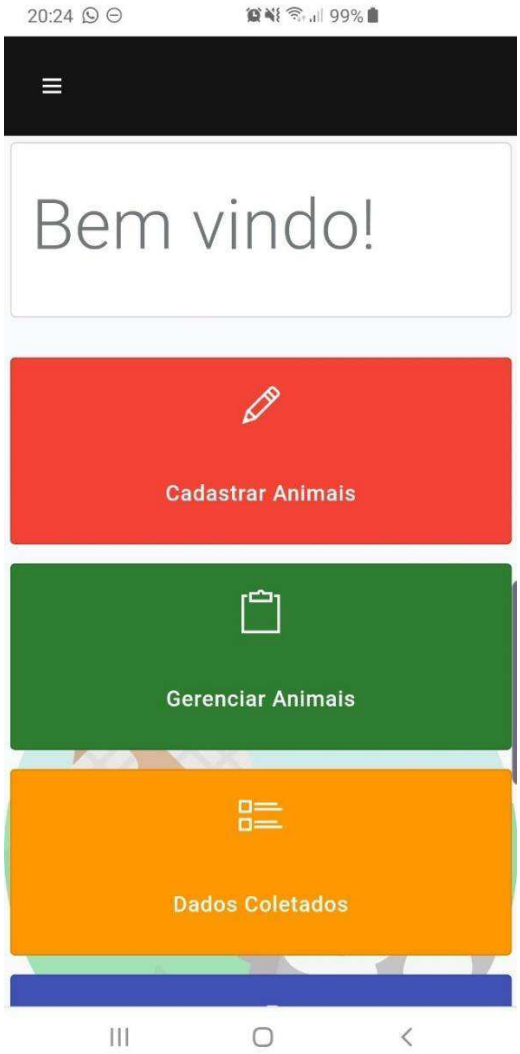
Figura 67 – Página Home.



Fonte: O autor, 2019.

Em primeiro momento, ao acessar esta página utilizando um computador, como demonstrado na Figura 67, os atalhos na Home parecem desnecessários, pois os mesmos podem ser facilmente acessados através do menu lateral. Porém, essa página se mostra interessante quando a aplicação está sendo acessada através de um dispositivo móvel, como visto na Figura 68.

Figura 68 – Página Home vista em um smartphone.



Fonte: O autor, 2019.

Acessando a página por um smartphone, o menu lateral sempre carregará colapsado, sendo assim, os atalhos na Home apresentam uma facilidade na navegação nestes casos.

4 RESULTADOS E DISCUSSÃO

Neste capítulo estão apresentados os principais testes realizados e resultados obtidos para este sistema de monitoramento de produção de leite.

4.1 SELEÇÃO DO SENSOR DE FLUXO

Inicialmente não foram imaginadas dificuldades na escolha de um sensor de fluxo para este projeto, entretanto a escolha de um sensor apto para trabalhar no ambiente de produção de leite acabou apresentando problemas. Para este tipo de ambiente é recomendado o uso de peças em aço inox, porém, sensores deste tipo acabam apresentando custos elevados, assim, inviabilizando o projeto, que tem como objetivo ser um sistema acessível ao produtor.

Desta maneira, optou-se pelo uso do sensor YF-S401, mesmo sabendo que o mesmo não oferece as características ideais. Assim, infelizmente, não foi possível fazer a instalação deste projeto em um ambiente real de ordenha.

Portanto, houve a necessidade da criação de um protótipo, que permitisse a simulação do processo de ordenha, permitindo os testes do projeto com este sensor não ideal.

4.2 TESTES E AJUSTES COM O PROTÓTIPO PARA SIMULAÇÃO

Para poder realizar as coletas de dados com o protótipo, inicialmente foram realizados testes para verificar as pressões de trabalho em que este pode operar, relacionando-as com o fluxo produzido pelas mesmas. Para verificação da pressão de trabalho, a válvula reguladora possui um manômetro, apresentado na Figura 69.

Figura 69 – Manômetro da válvula reguladora de fluxo.

Fonte: O autor, 2019.

O protótipo foi testado com as pressões de 1, 2, 3 e 4 bar. Para fazer este teste, o protótipo é regulado em uma das pressões de teste, e é feita a coleta em um recipiente de todo o volume produzido pelo protótipo, durante 1 minuto. Após a passagem deste minuto, o volume produzido é então medido, utilizando para isto um copo medidor, apresentado na Figura 70.

Tendo a medida do volume total, sabe-se o fluxo, em litros por minuto, que foi produzido pelo protótipo. Estes testes foram repetidos três vezes para cada pressão. Os resultados obtidos estão apresentados na Tabela 8.

Figura 70 – Copo Medidor.

Fonte: O autor, 2019.

Tabela 8 – Resultados dos testes de pressão de trabalho.

Pressão (bar)	Fluxo (L/min)
1	1,45
1	1,44
1	1,46
2	1,91
2	1,93
2	1,93
3	2,34
3	2,33
3	2,31
4	2,67
4	2,69
4	2,69

Fonte: O autor, 2019.

Mesmo a bomba escolhida podendo operar em até 4,8 bar, aqui não foram realizados testes acima de 4 bar, pois elevando a pressão acima disso, para cerca de 4,5 bar, já era suficiente para que a bomba começasse a apresentar problemas no seu funcionamento.

A realização dos outros testes, e também das coletas de dados, foi feita então regulando o sistema em 4 bar, já que o fluxo produzido nesta pressão está dentro do suportado pelo sensor, e este também está próximo ao fluxo produzido em uma ordenha real, que deve ficar em torno de 3,0 a 3,9 litros por minuto.

Após a verificar as pressões de operação do sistema, foram feitos testes para verificar o funcionamento do sensor de fluxo.

De acordo com o datasheet do sensor, este deve produzir 5880 pulsos para cada litro de água que passa por ele. Para fazer esta verificação, o sistema é ligado, e o volume que passa pelo sensor é coletado em um copo medidor, o mesmo apresentado na Figura 70. Tendo coletado 1 litro neste copo, o sistema é desligado, e a quantidade de pulsos produzidas pelo sensor é verificada. Este teste foi repetido 20 vezes, sempre trabalhando na pressão de 4 bar. A Tabela 9 apresenta os resultados obtidos.

Tabela 9 – Resultados dos pulsos gerados pelo sensor de fluxo.

Amostra	Pulsos	Amostra	Pulsos
1	2758	11	2663
2	2679	12	2706
3	2730	13	2690
4	2653	14	2675
5	2683	15	2578
6	2649	16	2621
7	2611	17	2655
8	2628	18	2641
9	2632	19	2610
10	2651	20	2632

Fonte: O autor, 2019

Como pode ser visto, os pulsos gerados ficaram muito longe dos 5880 informados pelo datasheet. Para calibrar o programa, permitindo a correta medição do volume, foi realizada a média dos pulsos coletados, eliminando o maior e menor valor obtidos. O cálculo dessa média é feito fazendo a soma dos pulsos, e dividindo por 18, resultando em 2656,056.

O firmware do sistema é então ajustado, para o valor de 2656 pulsos por litro.

4.3 COLETA DE DADOS COM O PROTÓTIPO

Para a simulação de um ambiente de ordenha, foram cadastrados no sistema 5 animais, apresentados na Figura 71, sendo que estes seriam ordenhados duas vezes ao dia. Estas ordenhas foram simuladas durante 20 dias, obtendo assim ao final 40 dados para cada animal.

Figura 71 – Animais para testes diários.

3	Toretto	Teste Diario	3		
4	Princesa	Teste Diario	3		
5	Mimosa	Teste Diario	3		
6	Brian	Teste Diario	3		
7	Pipoca	Teste Diario	3		

Fonte: O autor, 2019.

4.3.1 Dados de Volume

Para que cada animal produzisse um volume diferente, o tempo de ordenha de cada um também precisou ser diferente, já que todos os dados foram coletados utilizando 4 bar como pressão. As Figuras 72, 73 e 74 apresentam os dados coletados para alguns dos animais cadastrados.

Figura 72 – Análise Gráfica do Animal 5.

Fonte: O autor, 2016.

Figura 73 – Análise Gráfica do Animal 3.

Fonte: O autor, 2016.

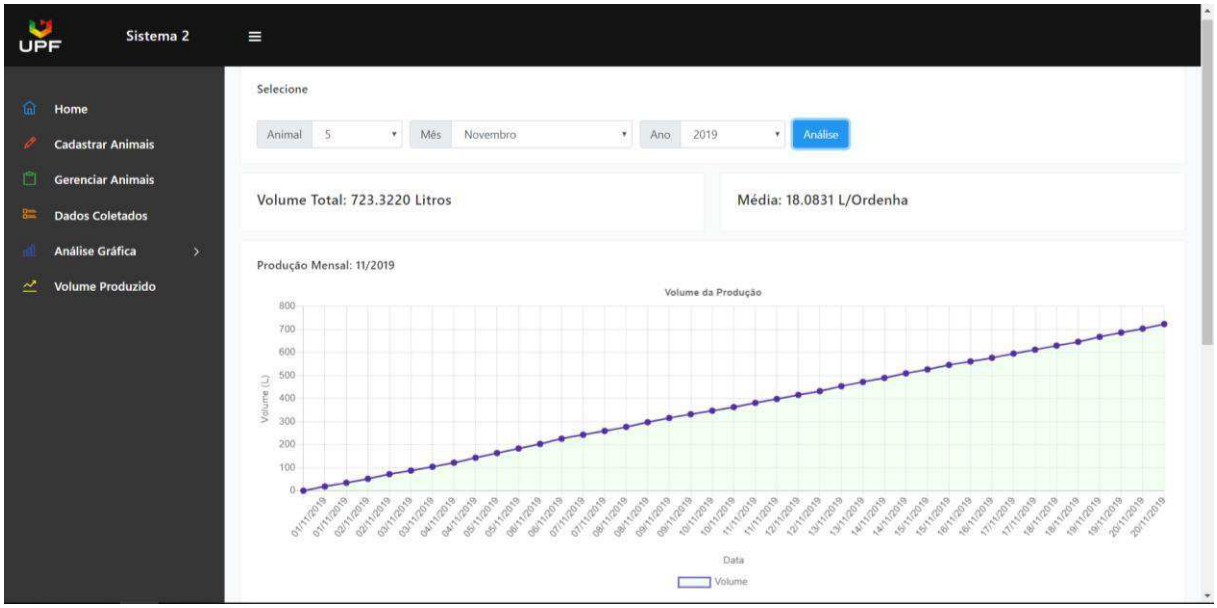
Figura 74 – Análise Gráfica do Animal 6.

Fonte: O autor, 2016.

Os dados apresentados já são exibidos utilizando a aplicação web desenvolvida. Na Figura 72, pode-se perceber claramente que o animal 5 apresenta produtividade superior a seu lote. A Figura 73 apresenta um animal que está na média, e na Figura 74 percebe-se um animal que está produzindo menos que a média de seu lote.

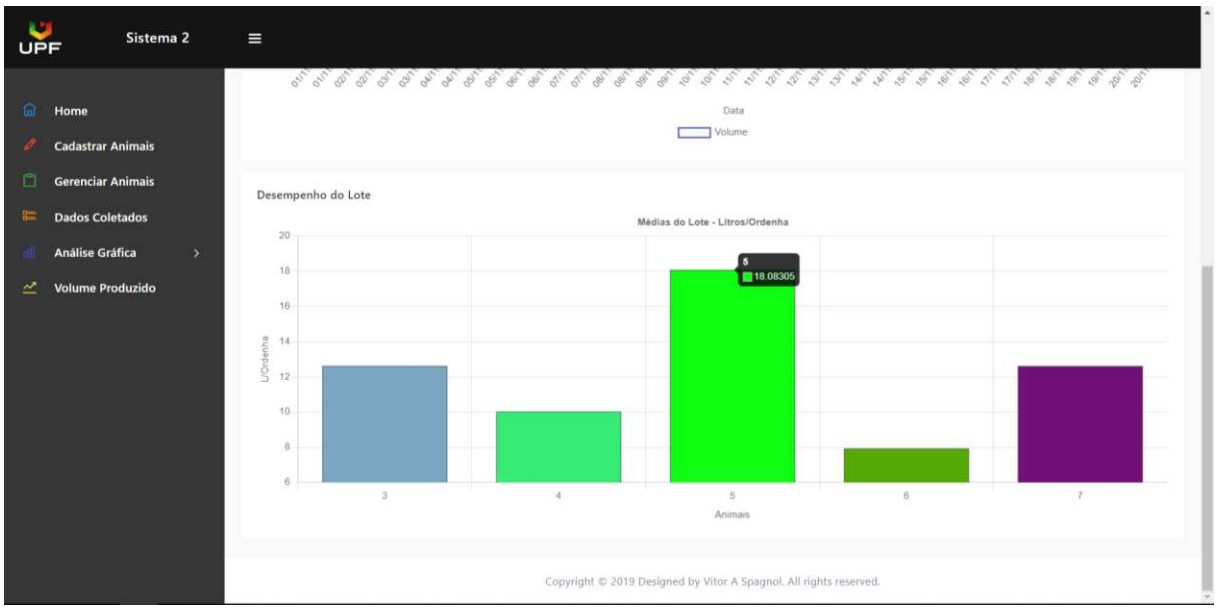
A Figura 75 apresenta a página da aplicação web Volume Produzido. Aqui pode-se perceber a produção total e média do animal 5. Também é possível ver a constância no volume produzido pelo mesmo, observando o gráfico linear da evolução da produção. Ao final da página, demonstrada na Figura 76, exibe-se o gráfico de barras, que compara a produção média por ordenha deste animal com os demais presentes no lote.

Figura 75 – Volume Produzido para o animal 5.



Fonte: O autor, 2019.

Figura 76 – Comparação entre animais no mesmo lote.

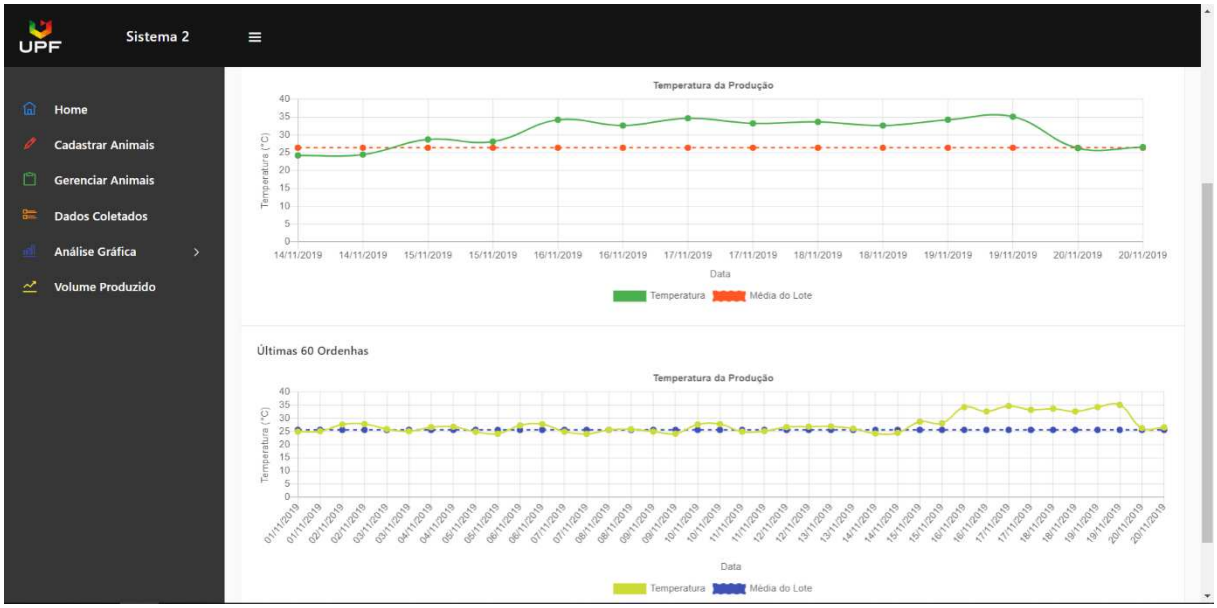


Fonte: O autor, 2019.

4.3.2 Dados de Temperatura

Da mesma forma que nos dados de volume, na Figura 77 estão apresentados os dados de temperatura coletados para o animal 7. Estes dados novamente estão apresentados utilizando as páginas de análise da aplicação web desenvolvida.

Figura 77 – Gráfico de Temperatura para o animal 7.



Fonte: O autor, 2019.

Para gerar uma diferença de temperatura entre as ordenhas, algumas ordenhas foram realizadas em temperatura ambiente, enquanto que para outras foi utilizado uma resistência para aquecimento de água, colocada dentro do reservatório, elevando assim a temperatura do líquido que circula no sistema.

Através destes gráficos pode-se perceber alterações na temperatura do animal, alertando assim para mudanças biológicas no mesmo.

4.3.3 Dados de Tempo

Aqui são apresentados os dados coletados sobre a duração das ordenhas. Novamente é utilizada para isso a aplicação web desenvolvida.

A Figura 78 demonstra os dados de duração da ordenha para o animal 6.

Figura 78 – Análise gráfica da duração da ordenha para o animal 6.

Fonte: O autor, 2019.

Os dados de tempo de ordenha salvos no banco de dados estão todos escritos em uma string, no formado “mm:ss”. Para a criação dos gráficos de duração da ordenha, estes dados precisam ser convertidos para números. Esta conversão é feita utilizando a função parseInt(), nativa do JavaScript. Porém, esta função converterá apenas os minutos para valores inteiros, dessa forma, nos dados apresentados nos gráficos de tempo, são desconsiderados os segundos dos tempos de duração das ordenhas.

4.4 SOBRE O BANCO DE DADOS

Um problema possível, quando utilizando arquivos JSON para o banco de dados, é que com o aumento da quantidade de dados, estes podem vir a causar lentidão no momento de abri-los para exibição na aplicação web.

Para verificar se o aumento de dados resultaria em problemas, foi cadastrado o animal número 8. Para este animal, foram adicionados manualmente ao banco de dados 5.000 dados de ordenha.

Tendo agora um arquivo com todos estes dados, sua abertura foi testada, onde pode-se perceber que, mesmo com a enorme quantidade de dados contidos no mesmo, este continua sendo processado de maneira extremamente rápida, não causando problemas na sua manipulação ou visualização.

Ainda, caso este número de dados aumente, passando dos 5.000, pode-se facilmente adicionar a API uma rotina, de maneira que apenas os últimos 5.000 dados sejam mantidos, por exemplo, eliminando os dados mais antigos. Considerando duas ordenhas diárias, nestes 5.000 pontos mantidos estariam contidos mais de 6 anos de dados de ordenha sobre o animal.

4.5 SOBRE O MÓDULO PARA ORDENHADEIRA

Para o módulo para ordenhadeira, foram testadas todas as rotinas automáticas, sendo estas a verificação da conexão com a rede Wi-Fi a cada 60 segundos, e o sincronismo com o servidor a cada 5 minutos, recebendo os animais cadastrados no sistema e enviando quaisquer dados pendentes salvos na memória EEPROM.

O módulo também foi testado quando há perda repentina da conexão, ou ocorre algum erro na API. Ao ocorrer qualquer um destes problemas, o módulo continua operando normalmente, agora trabalhando no modo offline, utilizando a memória EEPROM para salvamento dos dados.

Também, foram feitos testes do módulo operando sem conexão, onde todos os dados são armazenados na EEPROM, e, ao recuperar a conexão, estes são enviados para o banco de dados.

4.5.1 Memória EEPROM

O uso da memória EEPROM permite que o módulo funcione offline. Nela são reservados 100 endereços para os números de identificação dos animais cadastrados no banco, e o resto dos endereços são utilizados para permitir ordenhas offline, sendo que da maneira como foi implementada, esta permite 37 ordenhas deste tipo.

Quando comparadas as 37 ordenhas offline com os possíveis 100 animais cadastrados, este número acaba sendo pequeno. Para alterar esta situação, pode-se fazer uma troca, diminuindo o número de animais cadastrados, e aumentando o número de ordenhas offline. Outra possibilidade seria a implementação de uma memória EEPROM externa, de tamanho maior, possibilitando mais ordenhas sem conexão.

4.5.2 Acesso a Rede Wi-Fi

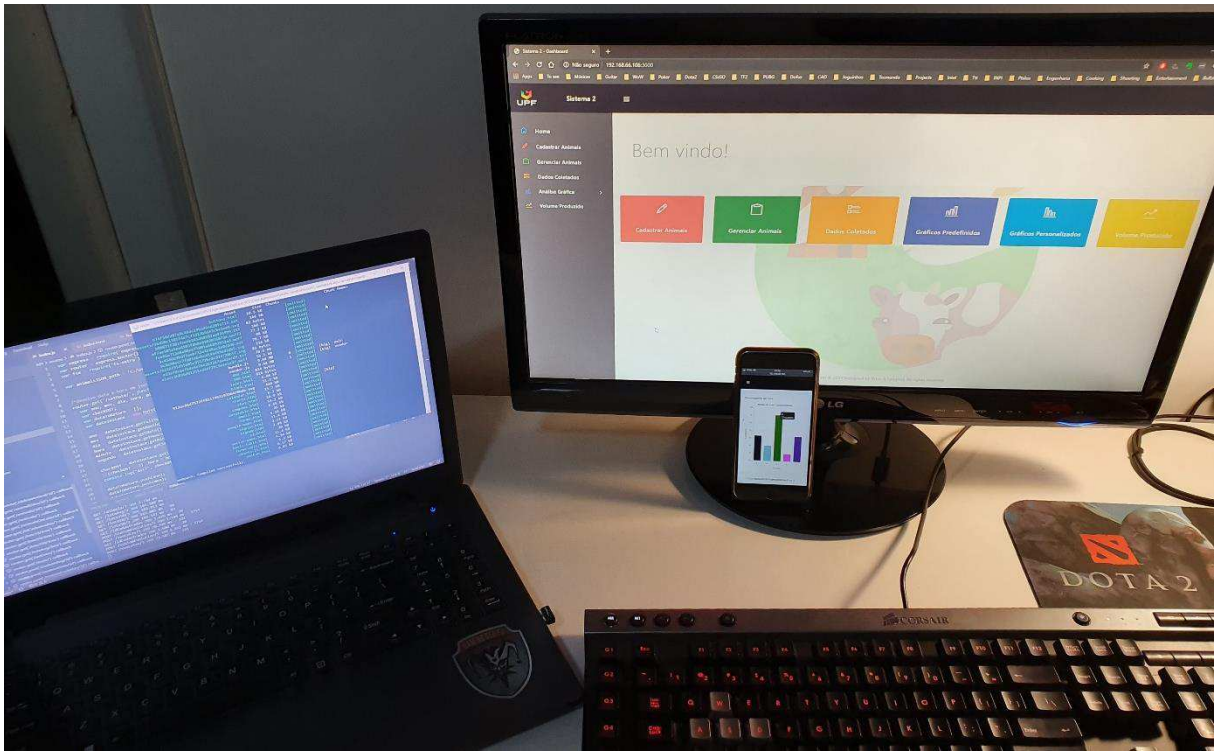
Como neste projeto, o nome de identificação do ponto de acesso ao qual o módulo se conecta, bem como a senha do mesmo são cadastradas diretamente na programação do microcontrolador, caso este sistema seja instalado em uma sala de ordenha real, há a necessidade de que o mesmos seja instalado em conjunto com um roteador, que terá seu uso exclusivo para o sistema de monitoramento, e não poderá ter seu nome ou senha alterados.

4.6 SOBRE A APLICAÇÃO WEB

Como visto no subcapítulo 4.3, a aplicação web se mostrou eficaz na análise dos dados coletados.

Todas as figuras apresentadas foram feitas com o uso de um computador para acessar a aplicação, porém, como já falado, por se tratar de uma aplicação web, e não de um aplicativo, esta também pode ser acessada de outros dispositivos. A Figura 79 apresenta a aplicação acessada de diferentes locais. Aqui, ela está rodando no notebook, e sendo acessada simultaneamente por um computador e um smartphone.

Figura 79 – Aplicação web acessada de diferentes dispositivos.



Fonte: O autor, 2019.

4.7 ESTIMATIVA DE CUSTOS

Considerando que o projeto busca trazer uma solução acessível ao produtor, a Tabela 10 apresenta os custos estimados para implementação do sistema em um ambiente real.

Tabela 10 – Custos estimados para implementação do sistema.

Item	Custo (R$)
LaunchPad TM4C123G	55,00
Módulo Wi-Fi ESP8266	26,00
Sensor de Temperatura DS18B20	18,00
RTC DS1307	16,00
Fonte 12V/1A	20,00
Sensor de Fluxo	~500,00
Outros Componentes (LEDs, resistores, ...)	20,00
Peças para instalação dos sensores	65,00
Confecção PCI	7,00
Roteador Wi-Fi	90,00
Hospedagens (API, Banco de dados e Aplicação Web)	0,00
Valor Total	817,00

Fonte: O autor,2019.

Para fazer a avaliação dos custos do projeto, foram utilizados valores de varejo. Caso o mesmo venha a ser produzido, estes custos podem ser diminuídos.

Para análise dos custos de hospedagem, foram utilizados como base os custos do serviço Google Cloud. Através deste serviço, é possível fazer a hospedagem do banco de dados a um custo mensal de US$0,30 por gigabyte. A API também pode ser hospedada neste serviço, de maneira gratuita, desde que a mesma não receba mais de 2 milhões de requisições por mês. Ainda, é possível fazer a hospedagem da aplicação web, pagando, mensalmente, US$0,060 por gigabyte que a mesma utiliza para ser implementada. (GOOGLE CLOUD, 2019).

Desta forma, considerando que o sistema utilizaria poucos recursos, e que estes recursos apresentam um baixo valor, os custos de hospedagem são desconsiderados.

O valor do sensor de fluxo aqui informado é um valor estimado, já que o utilizado no projeto não poderia ser implementado em um produto comercial. Se o custo do sensor ficar em torno de R$500,00, o projeto pode ser considerado viável para comercialização.

5 CONSIDERAÇÕES FINAIS

Tendo apresentado os dados que demonstram a baixa produtividade de leite brasileira, e o acesso à tecnologia dificultado aos pequenos e baixos produtores, é demonstrado que há a necessidade de que estes produtores se modernizem, devido ao fato de os mesmos serem responsáveis por grande parte da produção brasileira.

Este trabalho tem uma importância significativa, pois representa avanços na busca por uma solução que se adequa a realidade da cadeia produtiva de leite brasileira.

Através do monitoramento da temperatura do leite é possível saber se o animal está entrando no estado de cio, pois este faz com que a temperatura se eleve. Também, se a temperatura permanece elevada por diversas ordenhas consecutivas, é indicativo para alguma doença. Como demonstrado nos resultados deste trabalho, o método utilizado para coletar a temperatura do leite produzido de mostrou eficaz, e a aplicação para web permite a análise com facilidade.

Uma diminuição repentina no volume de leite também pode indicar o cio do animal, já uma diminuição gradual pode indicar o início de doenças como a mastite.

Infelizmente, o método para coleta de volume utilizado neste trabalho não possibilita que o mesmo possa ser empregado em uma ordenha real. Para que o sistema desenvolvido possa ser disponibilizado no mercado, há a necessidade de uma maior pesquisa, buscando encontrar ou desenvolver um sensor que se adeque as necessidades, apresentando um baixo custo de produção.

Juntando as informações de volume e temperatura, além de uma análise da produção, pode-se ter uma análise básica do estado clínico dos animais.

Assim, este é um importante passo, avançando o desenvolvimento de um sistema que pode atender ao produtor, que atualmente não é especializado, provendo um acompanhamento eficiente de sua produção, se a necessidade de um alto investimento. Com a continuação de pesquisas, principalmente na parte da medição do fluxo de leite, pode-se abrir um novo mercado, que buscará trazer tecnologias baratas e eficientes, voltadas a parte da cadeia de leite que mais apresenta problemas atualmente.

REFERÊNCIAS

AFIMILK. **Site da Empresa**. Disponível em: <https://www.afimilk.com/>. Acesso em: 18 maio 2019.

BALBINOT, Alexandre; BRUSAMARELLO, Valner João. **Instrumentação e fundamentos de medidas vol. 1**. 2. ed. Rio de Janeiro: Ltc, 2011.

BALBINOT, Alexandre; BRUSAMARELLO, Valner João. **Instrumentação e fundamentos de medidas vol. 2**. 2. ed. Rio de Janeiro: Ltc, 2011.

BARBOSA, Pedro Franklin et al. **Embrapa Gado de Leite: Sistemas de Produção**. 2002. Disponível em: <https://sistemasdeproducao.cnptia.embrapa.br/FontesHTML/Leite/LeiteSudeste/importancia.html>. Acesso em: 13 maio 2019.

CAMPOMORI, Cleber. **REST não é simplesmente retornar JSON: indo além com APIs REST**. Disponível em: <https://www.treinaweb.com.br/blog/rest-nao-e-simplesmente-retornar-json-indo-alem-com-apis-rest/>. Acesso em: 18 nov. 2019

CANAL RURAL. **Expointer Universitária: entenda as características de uma vaca leiteira.** 2012. Disponível em: <https://canalrural.uol.com.br/sites-e-especiais/expointer-universitaria-entenda-caracteristicas-uma-vaca-leiteira-36624/>. Acesso em: 16 maio 2019.

CANALTECH. **O que é API?** Disponível em: <https://canaltech.com.br/software/o-que-e-api/>. Acesso em: 18 nov. 2019.

CARIA, Maria; TODDE, Giuseppe; PAZZONA, Antonio. Evaluation of automated in-line precision dairy farming technology implementation in three dairy farms in Italy. **Frontiers Of Agricultural Science And Engineering**. Sassari, p. 181-187. 07 mar. 2019. Disponível em: <http://journal.hep.com.cn/fase/EN/10.15302/J-FASE-2019252>. Acesso em: 28 maio 2019.

CODEX ALIMENTARIUS OFFICIAL STANDARDS. **GENERAL PRINCIPLES OF FOOD HYGIENE**. 1969. Disponível em: <https://www.dairyinfo.gc.ca/index_e.php?s1=fil-idf&s2=codex&s3=stan-norm>. Acesso em: 15 maio 2019.

Colorlib. **Adminator**. Disponível em: <https://colorlib.com/polygon/adminator/index.html>. Acesso em: 28 nov. 2019.

COMER, Douglas E.; STEVENS, David L. **Internetworking with TCP/IP**: Client - Server Programming and Applications. 3. ed. Englewood Cliffs: Prentice Hall, 1993. Disponível em: <https://archive.org/details/internetworkingw00come_0/page/n7>. Acesso em: 18 nov. 2019.

DELAVAL. **Farm Management**. Disponível em: <https://www.delaval.com/pt-br/our-solutions/gerenciamento-de-rebanho/>. Acesso em: 18 maio 2019.

DELAVAL. **Indicadores de Fluxo**. Disponível em: <https://www.delaval.com/pt-br/our-solutions/milking/at-the-milking-point/milk-meters/>. Acesso em: 18 maio 2019.
DELAVAL. **Uso da extração automática em equipamentos de ordenha**. 2018. Disponível em: <https://www.milkpoint.com.br/canais-empresariais/delaval/uso-da-extracao-automatica-em-equipamentos-de-ordenha-211801/>. Acesso em: 17 maio 2019.

DESENVOLVIMENTO REGIONAL SUSTENTÁVEL: Bovinocultura de Leite. Brasília: Fundação Banco do Brasil, v. 8, n. 1, 2010. Disponível em: <https://www.bb.com.br/docs/pub/inst/dwn/Vol1BovinoLeite.pdf>. Acesso em: 13 maio 2019.

DFROBOT. **Water Flow Sensor - 1/8" Datasheet**. Disponível em: <https://media.digikey.com/pdf/Data%20Sheets/DFRobot%20PDFs/SEN0216_Web.pdf>. Acesso em: 18 nov. 2019.

DEVMEDIA. **Guia Completo de JavaScript**. Disponível em: <https://www.devmedia.com.br/guia/javascript/34372>. Acesso em: 18 nov. 2019.

DEVMEDIA. **Primeiros passos no Node.js.** Disponível em: <https://www.devmedia.com.br/nodejs/>. Acesso em: 18 nov. 2019.

EBAY. **Afimilk MPC Control Box**. Disponível em: <https://www.ebay.com/itm/AfiMilk-MPC-Control-Box-/222212050226?_ul=BR>. Acesso em: 21 maio 2019.

ESPRESSIF SYSTEMS. **ESP8266 AT Instruction Set**. 2019. Disponível em: <https://www.espressif.com/sites/default/files/documentation/4a-esp8266_at_instruction_set_en.pdf>. Acesso em: 25 nov. 2019.

FAÇANHA, Débora Andréa Evangelista et al. Variação anual de características morfológicas e da temperatura de superfície do pelame de vacas da raça Holandesa em ambiente semiárido. **Revista Brasileira de Zootecnia**, São Paulo, v. 39, n. 4, p.837-844, jan. 2010.

FAGNANI, Rafael. **Resumão das INs 76 e 77: elas estão chegando!** Disponível em: <https://www.milkpoint.com.br/colunas/rafael-fagnani/resumao-das-ins-76-e-77-elas-estao-chegando-212785/>. Acesso em: 15 maio 2019.

FILIPEFLOP. **Módulo WiFi ESP8266 ESP-12E**. Disponível em: <https://www.filipeflop.com/produto/modulo-wifi-esp8266-esp-12e/>. Acesso em: 18 nov. 2019.

FILIPEFLOP. **Sensor de Temperatura DS18B20 a Prova D'água**. Disponível em: <https://www.filipeflop.com/produto/sensor-de-temperatura-ds18b20-a-prova-dagua/>. Acesso em: 28 maio 2019.

Food and Agriculture Organization of the United Nations. **The Global Dairy Sector: Facts**. 2016. Disponível em: <https://www.fil-idf.org/wp-content/uploads/2016/12/FAO-Global-Facts-1.pdf>. Acesso em: 13 maio 2019.

GEA. **Software de gerenciamento de rebanho Dairy Plan C21**. Disponível em: <https://www.gea.com/pt/products/dairy-plan-c21.jsp>. Acesso em: 18 maio 2019.

GEA. **Unidades de Controle de Ordenha**. Disponível em: <https://www.gea.com/pt/products/milking-detacher-metatron-easyde-demax-dematron.jsp>. Acesso em: 18 maio 2019.

GOFF, Douglas. **The Dairy Science and Technology eBook**. Disponível em: <https://www.uoguelph.ca/foodscience/book-page/dairy-science-and-technology-ebook>. Acesso em: 17 maio 2019.

GOOGLE CLOUD. **Preços Cloud Filestore**. Disponível em: <https://cloud.google.com/filestore/pricing>. Acesso em: 03 dez. 2019.

GOOGLE CLOUD. **Preços do App Engine**. Disponível em: <https://cloud.google.com/appengine/pricing>. Acesso em: 03 dez. 2019.

GOOGLE CLOUD. **Preços e cotas Cloud Endpoints**. Disponível em: <https://cloud.google.com/endpoints/pricing-and-quotas>. Acesso em: 03 dez. 2019.

GRUNN. **DMIK**. Disponível em: <https://grunn.com.br/meister-tecnologia-em-medicao-de-vazao/medidores-de-vazao/dmik-sensor-de-fluxo-para-liquidos/>. Acesso em: 28 maio 2019.

GRUPO AGUIAR. **Ordenha Mecânica: Composição e função**. Disponível em: <http://www.grupoapoiar.com/ordenha-mecanica-composicao-e-funcao/>. Acesso em: 18 maio 2019.

HAMBY DAIRY SUPPLY. **Metatron P21 Control System by GEA WestfaliaSurge**. Disponível em: <https://hambydairysupply.com/metatron-p21-front-panel-with-control-system-by-gea-westfaliasurge/>. Acesso em: 21 maio 2019.

HOSTINGER. **O que é CSS? Guia Básico para Iniciantes.** Disponível em: <https://www.hostinger.com.br/tutoriais/o-que-e-css-guia-basico-de-css/>. Acesso em: 18 nov. 2019.

IMPACTA. **Desenvolvimento de aplicações web: tire suas dúvidas sobre o assunto!** Disponível em: <https://www.impacta.com.br/blog/2018/01/05/desenvolvimento-de-aplicacoes-web-tire-suas-duvidas-sobre-o-assunto/>. Acesso em: 18 nov. 2019.

INCOMAGRI. **Ordenhadeira Canalizada**. Disponível em: <http://www.incomagri.com.br/ordenhadeira_canalizada.php>. Acesso em: 20 maio 2019.

INTERNATIONAL DAIRY FEDERATION. **Facts & Figures**. 2016. Disponível em: <https://www.fil-idf.org/about-dairy/facts-figures/>. Acesso em: 13 maio 2019.

Instituto Mineiro de Agropecuária. **Normas higiêncio-sanitárias e tecnológicas para leite e produtos lácteos**. Disponível em: <http://www.ima.mg.gov.br/component/content/article/324-legislacao/415-legislacao-inspecao>. Acesso em: 15 maio 2019.

IRIAS, Anderson. **API HTTP + REST – Conceito e exemplo em Node.js**. Disponível em: <https://imasters.com.br/back-end/api-http-rest-conceito-e-exemplo-em-node-js>. Acesso em: 18 nov. 2019.

MAIA, Guilherme Baptista da Silva et al. Produção leiteira no Brasil. **BNDES Setorial**, Rio de Janeiro, v. 1, n. 37, p.371-398, mar. 2017. Disponível em: <https://web.bndes.gov.br/bib/jspui/handle/1408/1514>. Acesso em: 13 maio 2019.

MAXIM INTEGRATED. **1-WIRE COMMUNICATION THROUGH SOFTWARE**. 2009. Disponível em: <https://www.maximintegrated.com/en/design/technical-documents/app-notes/1/126.html>. Acesso em: 25 nov. 2019.

MAXIM INTEGRATED. **DS18B20 - Datasheet**: Programmable Resolution 1-Wire Digital Thermometer. Disponível em: <https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf>. Acesso em: 28 maio 2019.

MAXIM INTEGRATED. **MAX6675 - Datasheet**. Disponível em: <https://img.filipeflop.com/files/download/Datasheet_MAX6675.pdf>. Acesso em: 28 maio 2019.

MCGEE, Harold. **On Food and Cooking**: The Science and Lore of the Kitchen. New York: Scribner, 2004.

MDN WEB DOCS. **CSS**. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/CSS>. Acesso em: 18 nov. 2019.

MDN WEB DOCS. **Guia JavaScript**. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide>. Acesso em: 18 nov. 2019.

MDN WEB DOCS. **HTML: Linguagem de Marcação de Hipertexto**. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/HTML>. Acesso em: 18 nov. 2019.

MDN WEB DOCS. **JavaScript básico**. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Aprender/Getting_started_with_the_web/JavaScript_basico>. Acesso em: 18 nov. 2019.

MDN WEB DOCS. **Métodos de requisição HTTP**. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Methods>. Acesso em: 18 nov. 2019.

MDN WEB DOCS. **Uma visão geral do HTTP**. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Overview>. Acesso em: 18 nov. 2019.

MILKPOINT. **IBGE: produção total de leite cai 0,5% em 2017**. 2018. Disponível em: <https://www.milkpoint.com.br/noticias-e-mercado/panorama-mercado/ibge-producao-total-de-leite-cai-05-em-2017-210514/>. Acesso em: 13 maio 2019.

MINISTÉRIO DA AGRICULTURA. **Saem novas regras para a produção de leite**. 2018. Disponível em: <http://www.agricultura.gov.br/noticias/saem-novas-regras-para-a-producao-de-leite>. Acesso em: 15 maio 2019.

NETTO, Francelino Goulart da Silva; BRITO, Luciana Gatto; FIGUEIRÓ, Marivaldo Rodrigues. **A ordenha da vaca leiteira**. Porto Velho: Embrapa, 2006.

OLIVEIRA NETO, Aroldo Antonio de. **Pecuária Leiteira: Análise dos Custos de Produção e da Rentabilidade nos anos de 2014 a 2017**. 16. ed. Brasília: Companhia Nacional de Abastecimento, 2018. (Compêndio de Estudos Conab).

OPUS SOFTWARE. **Node.js – O que é, como funciona e quais as vantagens.** Disponível em: <https://www.opus-software.com.br/node-js/>. Acesso em: 18 nov. 2019.

ORDEMAX. **Catálogo Geral de Acessórios 2019**. Disponível em: <http://ordemax.net.br/>. Acesso em: 15 mar. 2019.

ORDEMAX. **Máquina de Ordenhar Transferidor de Leite Plus**. 2019. Disponível em: <http://ordemax.net.br/index.php?id_product=133&controller=product>. Acesso em: 28 maio 2019.

PROCREARE. **A Raça Holandesa**. 2016. Disponível em: <https://procreare.com.br/a-raca-holandesa/>. Acesso em: 16 maio 2019.

RIBEIRO, Marlice Teixeira; CARVALHO, Armando da Costa. **Ordenha Mecânica**. Disponível em: <https://www.agencia.cnptia.embrapa.br/Agencia8/AG01/arvore/AG01_63_217200392359.html>. Acesso em: 18 maio 2019.

ROSA, Marcelo Simão da et al. **Boas Práticas de Manejo - Ordenha**. Jaboticabal: Funep, 2009.

RURAL PECUÁRIA. **Raça Holandesa**. Disponível em: <http://ruralpecuaria.com.br/tecnologia-e-manejo/racas-gado-de-leite/raca-holandesa.html>. Acesso em: 16 maio 2019.

SANTOS, Marco Veiga. **Avaliação de desempenho de salas de ordenha.** 2007. Disponível em: <https://www.milkpoint.com.br/colunas/marco-veiga-dos-santos/avaliacao-de-desempenho-de-salas-de-ordenha-34334n.aspx>. Acesso em: 17 maio 2019.

SUPER INTERESSANTE. **Quais são os alimentos mais consumidos no mundo?** 2011. Disponível em: <https://super.abril.com.br/mundo-estranho/quais-sao-os-alimentos-mais-consumidos-no-mundo/>. Acesso em: 13 maio 2019.

TEXAS INSTRUMENTS. **ARM® Cortex®-M4F Based MCU TM4C123G LaunchPad™ Evaluation Kit.** Disponível em: <http://www.ti.com/tool/EK-TM4C123GXL#descriptionArea>. Acesso em: 28 maio 2019.

TEXAS INSTRUMENTS**. ARM® Cortex®-M4F-Based MCU TM4C1294 Connected LaunchPad™ Evaluation Kit**. Disponível em: <http://www.ti.com/tool/EK-TM4C1294XL#1>. Acesso em: 28 maio 2019.

USINA INFO ELETRÔNICA E ROBÓTICA. **Sensor de Fluxo de Água 0,3-6 l/min**. Disponível em: <https://www.usinainfo.com.br/sensor-de-fluxo-arduino/sensor-de-fluxo-de-agua-03-6-lmin-2841.html>. Acesso em: 18 nov. 2019.

VARGAS, Anderson Daniel Freitas et al. Estimação de parâmetros genéticos para a produção de leite no dia do controle e em 305 dias para primeiras lactações de vacas da raça Holandesa. **Revista Brasileira de Zootecnia**, São Paulo, v. 35, n. 5, p.1959-1965, jan. 2006.

VERTIGO TECNOLOGIA. **O que é API? Entenda de uma maneira simples.** Disponível em: <https://vertigo.com.br/o-que-e-api-entenda-de-uma-maneira-simples/>. Acesso em: 18 nov. 2019.

WEIZUR. **Site da Empresa**. Disponível em: <http://www.weizur.com.br/home>. Acesso em: 18 maio 2019.