UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

MAINARA CRISTINA LORENCENA

CONTRIBUIÇÕES AO MANEJO AVÍCOLA USANDO TÉCNICAS DE CONTROLE SUPERVISÓRIO E MONITORAMENTO REMOTO

TRABALHO DE CONCLUSÃO DE CURSO

PATO BRANCO
2015

MAINARA CRISTINA LORENCENA

CONTRIBUIÇÕES AO MANEJO AVÍCOLA USANDO TÉCNICAS DE CONTROLE SUPERVISÓRIO E MONITORAMENTO REMOTO

Trabalho de Conclusão de Curso de graduação, apresentado à disciplina de Trabalho de Conclusão de Curso 2, do Curso de Engenharia de Computação do Departamento Acadêmico de Informática - DAINF - da Universidade Tecnológica Federal do Paraná - UTFPR, Câmpus Pato Branco, como requisito parcial para obtenção do título de Engenheiro.

Orientador: Prof. Dr. Marcelo Teixeira

Coorientador: Prof. MSc. Ricardo Bernardi

PATO BRANCO

2015

MINISTÉRIO DA EDUCAÇÃO
Universidade Tecnológica Federal do Paraná
Câmpus Pato Branco
Departamento Acadêmico de Informática
Curso de Engenharia de Computação

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

TERMO DE APROVAÇÃO

Às 10 horas e 30 minutos do dia 03 de dezembro 2015, na sala V106, da Universidade Tecnológica Federal dão Paraná, Câmpus Pato Branco, reuniu-se a banca examinadora composta pelos professores Marcelo Teixeira (orientador), Ricardo Bernardi (coorientador), Cesar Rafael Claure Torrico e Marco Antonio de Castro Barbosa para avaliar o trabalho de conclusão de curso com o título **Contribuições ao manejo avícola usando técnicas de controle supervisório e monitoramento remoto**, da aluna **Mainara Cristina Lorencena**, matrícula 01159135, do curso de Engenharia de Computação. Após a apresentação a candidata foi arguida pela banca examinadora. Em seguida foi realizada a deliberação pela banca examinadora que considerou o trabalho aprovado.

Marcelo Teixeira Orientador (UTFPR)	Ricardo Bernardi Coorientador (UTFPR)
Cesar Rafael Claure Torrico (UTFPR)	Marco Antonio de Castro Barbosa (UTFPR)
Beatriz Terezinha Borsoi Coordenador de TCC	Marco Antonio de Castro Barbosa Coordenador do Curso de Engenharia de Computação

A Folha de Aprovação assinada encontra-se na Coordenação do Curso.

Dedico este trabalho aos donos de todo meu amor e admiração, ao meu pai Idelmar Lorencena, à minha mãe Neide Lorencena e à minha querida irmã Andrieli Lorencena, por todo apoio e compreensão em cada momento da minha vida.

Há mais mistérios entre o céu e a Terra do que sonha a nossa vã filosofia.

William Shakespeare

AGRADECIMENTOS

Sou imensamente grata aos meus pais, Idelmar Lorencena e Neide Lorencena, por terem acreditado nos meus sonhos e na minha capacidade de realizá-los. A vocês, que tantas vezes deixaram a si mesmos de lado para dar todo apoio e suporte que eu precisava, não há maneiras de agradecer, apenas recebam meu amor e reconhecimento por tudo. Agradeço também minha melhor amiga, aquela que posso afirmar com absoluta certeza que para sempre estará comigo, minha irmã Andrieli Lorencena. Obrigada por ter sido tão presente, pelos conselhos, advertências. Obrigada por ter compartilhado suas experiências comigo, para que eu aprendesse antes de errar, e também por ter me ajudado mesmo errando. São gestos que me fazem amá-la e respeitá-la, não só como uma irmã, mas principalmente como uma verdadeira amiga. Ao meu namorado, Osvaldo Herpich, pela compreensão nos momentos difíceis da graduação e o apoio de sempre. Aos meus amigos, de faculdade e da vida, obrigada por terem feito meus dias mais felizes, pelas risadas, estudos, conversas. Vocês são muito especiais e permanecerão em meu coração.

Ao meu orientador, Marcelo Teixeira, e coorientador, Ricardo Bernardi, pela orientação, ensinamentos e por estarem sempre prontos para me atender, sanando minhas dúvidas e contribuindo para o meu crescimento profissional e pessoal.

Por último, cito a figura que, em minha vida, está em primeiro lugar: Deus. Obrigada, Senhor, por todas as bênçãos recebidas por mim e por minha família, por manter nossa união que tanto me dá força, e por me permitir concluir a graduação, alcançando o primeiro de muitos objetivos. És o mentor de todas as minhas conquistas.

RESUMO

A avicultura é uma atividade que se fortaleceu muito ao longo dos últimos anos, tornando-se uma prática importante para a economia nacional, o que faz com que as exigências por qualidade na produção de carne de frangos de corte sejam cada vez maiores. A qualidade da carne depende diretamente do bem-estar animal, que é obtido através do manejo adequado das aves no que diz respeito à higienização, saúde, alimentação, abastecimento de água e condições climáticas. A condição climática do aviário é um dos principais fatores que influenciam no desenvolvimento das aves, sendo a temperatura e a umidade, duas das variávies mais importantes a serem controladas no interior do galpão. É desejável que se mantenham os valores mais próximos possíveis dos ideais, além da máxima homogeneização destas variáveis ao longo do ambiente. Este trabalho apresenta as etapas que envolvem a modelagem, a implementação e o monitoramento de um sistema para o controle automático da temperatura e da umidade em processos avícolas. A abordagem de modelagem e síntese do controlador é inspirada na Teoria de Controle Supervisório, utilizando autômatos finitos. A etapa de implementação traduz o controlador para o ambiente operacional, cujo código é implementado no microcontrolador MSP430G2553, da *Texas Instruments*. Por fim, a etapa de monitoramento remoto da temperatura e da umidade comunica, via protocolo Modbus RTU, os comandos do microcontrolador com o ScadaBR, uma interface gráfica utilizada em ambientes industriais para a observação e a interferências sobre processos. A interface pode ser acessada através da Internet e mostra em tempo real a condição climática do ambiente monitorado, permitindo ainda que os usuários atuem remotamente sobre os componentes do sistema.

Palavras-chave: Avicultura, Controle Supervisório, Microcontrolador, Protocolo Modbus RTU, ScadaBR.

ABSTRACT

Aviculture is an activity that has strengthened a lot for the last few years, becoming an important practice for the national economy, which makes increasing the quality requirements in the production of chicken meat. The quality of meat depends directly on animal welfare, which is achieved through the proper handling of the birds in terms of hygiene, health, food, water and weather conditions. The aviary climatic condition is one of the main factors that influence the birds development. The temperature and humidity are the most important variables to be controlled inside of aviary. It's desirable to conserve the values as close as possible to the ideals, besides to keep homogenization along the environment. This work presents the modeling, implementation and monitoring phases for obtaining a automatic control system for temperature and humidity in poultry farming. The modeling approach and control synthesis is inspired on the Supervisory Control Theory using finite-state automata. The implementation step translates the automaton controller to the operational environment whose resulting code is transfered to a MSP430G2553 microcontroller, from *Texas Instruments*. Finally, the monitoring stage implements a remote web system that communicates, via RTU protocol, the commands from the microcontroller with a SCADA framework, the ScadaBR, a graphical interface commonly used in industrial environments to observe and maybe interfere on process. The interface can be accessed through the Internet and show the climatic condition in real time for users, also allowing them to remotely interfere on the system components.

Keywords: Aviculture, Supervisory Control, Microcontroller, Modbus RTU protocol, ScadaBR.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

Este capítulo apresenta as considerações iniciais a respeito da prática da avicultura, as demandas existentes nesta área, o cenário da produção de frangos de corte atualmente no Brasil, as tecnologias existentes nos aviários e alguns problemas identificados no modelo tradicional utilizado para realizar o monitoramento e o controle da temperatura e da umidade no ambiente de criação das aves.

Em face desses problemas, apresenta-se também uma proposta contendo algumas alternativas para resolvê-los.

1.1 CONSIDERAÇÕES INICIAIS

O cenário atual da indústria e comércio no mundo todo, incluindo a agricultura, a criação de animais de corte e outras atividades do ramo, é de constante inovação e a tecnologia está cada vez mais presente nos mais diversos setores.

Por vários anos, a criação de frangos de corte foi considerada uma fonte alternativa de renda ou consumo próprio das famílias do campo. No início do século passado iniciou-se no Brasil, mais precisamente nos estados de São Paulo, Rio de Janeiro e Minas Gerais, o processo de aperfeiçoamento dessa atividade através da implantação de recursos tecnológicos visando melhorar o rendimento da produção. Nos anos 90, principalmente com a abertura econômica e com a estabilização da inflação, a agroindústria passou para a era da competitividade (SANDI, 2015).

No campo da avicultura, a criação de frangos de corte conquistou

novos mercados e a exigência pela qualidade da produção e otimização de custos aumentou gradativamente. O setor tem hoje relevante importância social e econômica. São 3,6 milhões de empregos diretos e indiretos, que agrega produtores, frigoríficos e exportadores, gerando mais de 300 mil empregos de fábrica. Os negócios que envolvem o segmento avícola geram um movimento de R$ 36 bilhões e participação de 1,5% no PIB o que supera todas as riquezas produzidas por países como El Salvador, Trinidad, Tobago e Bolívia (UBABEF, 2012).

A avicultura é representada por dezenas de milhares de produtores integrados, centenas de empresas beneficiadoras e dezenas de empresas exportadoras, o que significa que essa atividade se fortaleceu ao longo dos anos em todo o país, concentrando-se mais nas regiões Sul e Sudeste (ABPA, 2015). Tal organização leva a avicultura brasileira a ser reconhecida hoje como das mais desenvolvidas do mundo (UBABEF, 2008).

Tecnicamente, a qualidade na produção de frangos de corte está diretamente relacionada ao bem-estar animal, o que é obtido mantendo-se o ambiente interno do galpão com as condições climáticas, higiênicas, de alimentação e abastecimento de água, ideais para o desenvolvimento das aves. Ao se garantir a estabilidade da temperatura e da umidade (não apenas, mas sobretudo), sempre respeitando os valores indicados para cada fase da vida do frango, há uma tendência de que se obtenha melhores índices de conversão alimentar (relação entre consumo de ração em um período de tempo e ganho de peso) e de aceleração do desenvolvimento animal.

Sendo a temperatura e a umidade as variáveis do ambiente que mais influenciam no processo de produção de frangos de corte, é fundamental que elas sejam homogêneas, ou seja, é desejável que se observem os mesmos valores (ou mais próximos possíveis) em todos os pontos de medição de dados no interior do aviário, e é fundamental que os pontos de medição se distribuam ao longo de toda a extensão do galpão. Atualmente, existem sistemas de con-

trole capazes de capturar a condição climática interna do galpão por meio de sensores, e, com base nisso, automaticamente manipular a temperatura e a umidade acionando automaticamente dispositivos atuadores (exaustores, ventiladores, aquecedores, entradas de ar, etc.). Como resultado, a tendência é a de que o ambiente se torne mais estável e agradável para as aves (BANG *et al.*, 2014; SOIN *et al.*, 2013; SAMER, 2009).

1.2 CONTEXTUALIZAÇÃO DA PESQUISA

Mesmo se dispondo de tecnologia moderna, e cara, atualmente não se observa a tão desejável homogeneização do ambiente interno de um aviário. Uma das principais barreiras é a tradicional arquitetura adotada para os galpões, na qual os exaustores se localizam em uma das extremidades do aviário e são responsáveis pela exaustão de todo o ambiente, o que dificulta homogeneizar a temperatura e a umidade interna, pois, naturalmente, a ação é mais intensa nas proximidades dos equipamentos do que na extremidade oposta a eles.

Um efeito imediato desse tipo de configuração é o surgimento dos chamados ”pontos mortos”, que são zonas internas com menor fluxo de ar. Em outras zonas, por outro lado, o ar externo é canalizado com maior intensidade, causando discrepância nos níveis medidos de temperatura. Ademais, o próprio funcionamento dos exaustores não é homogêneo, uma vez que eles são acionados individualmente, conforme a necessidade de exaustão, atuando ao longo do comprimento do aviário. Assim se, por exemplo, um exaustor for acionado na posição mais na lateral do aviário, é esperado que ele configure o fluxo de ar conforme a sua posição, desfavorecendo a exaustão da zona lateral oposta, devido ao fato de que os exaustores existentes não conseguem atingi-la.

A partir de entrevistas e de visitas técnicas a avicultores da região de Pato Branco, Paraná, constatou-se, de fato, uma diferença significativa na temperatura dois extremos do aviário. Conforme relatos, o problema se agrava em dias de temperaturas baixas, o que é bastante típico em boa parte do ano

na região sul do Brasil. Nesses dias, se por um lado é necessário manter o conforto térmico das aves (desligando a exaustão), por outro lado é fundamental eliminar concentrações de amônia[1] (ligando exaustão).

Nesse sentido, o problema central abordado neste trabalho é fazer uso de técnicas de controle supervisório para a criação de um controlador ótimo capaz de coordenar os componentes considerados no sistema. Para isso, assume-se uma redistribuição dos componentes do aviário, principalmente em relação às entradas de ar, que agora passam a ser laterais, e em relação à disposição dos sensores de medição, que agora ocupam estrategicamente toda a extensão do galpão. Ambos os pressupostos são factíveis, na prática. Por fim, como fechamento do trabalho, propõe-se complementar o sistema de controle com uma interface de monitoramento e integração com o usuário, utilizando um sistema SCADA.

1.3 OBJETIVOS

Esquematicamente, os objetivos da pesquisa podem ser assim apresentados:

1.3.1 OBJETIVO GERAL

- Desenvolver um sistema automático de supervisão e controle de temperatura e umidade, que promova o melhoramento das condições climáticas e da circulação de ar no interior de um aviário, sobre uma arquitetura com ventilação transversal, além de permitir o monitoramento remoto e eventuais intervenções do usuário sobre certos equipamentos, via software.

[1] **Amônia é um gás incolor gerado a partir da decomposição microbiana de dejetos, que causa significativas perdas econômicas na criação das aves (OLIVEIRA, 2003)**

1.3.2 OBJETIVOS ESPECÍFICOS

- Modelar o comportamento individual e integrado dos componentes internos do aviário em malha aberta de controle (planta);
- Modelar os requisitos comportamentais (especificações) a serem impostos ao sistema em termos de comportamento em malha fechada de controle;
- Sintetizar um controlador ótimo para o ambiente do aviário, com base na planta e nas especificações predefinidas, utilizando técnicas da Teoria de Controle Supervisório;
- Integrar o sistema de controle com uma ferramenta SCADA através do protocolo de comunicação Modbus RTU;
- Monitorar as variáveis do ambiente, como nível de temperatura e umidade, exibindo-as remotamente e em tempo real através de um software que permita ao usuário visualizar e intervir nos dispositivos, em caso de falha do sistema ou quando julgar conveniente conforme seu domínio empírico.

O alcance desses pressupostos parte de algumas premissas apresentadas no Capítulo 2, a seguir, enquanto que os resultados alcançados são efetivamente ilustrados no Capítulo 3.

2 REFERENCIAL TEÓRICO

Este capítulo traz um apanhado técnico dos principais fundamentos que estruturam este trabalho.

2.1 A AVICULTURA NO BRASIL

A produção de frangos de corte impacta na economia da maioria dos estados, com uma tendência de expansão para a região centro-oeste (). Aproximadamente 90% das aves são produzidas, processadas, abatidas e distribuídas pela empresa a que se destinam. Isso gera empregos, renda, fixação do homem no campo e viabiliza a pequena propriedade (UBABEF, 2008)

Em 2011 a produção brasileira atingiu a marca histórica de 13,058 milhões de toneladas, garantindo ao Brasil uma posição entre os três maiores produtores mundiais de carne de frango, com Estados Unidos e China, e, desde 2004, mantém a posição de maior exportador mundial. No final de 2011, atingiu a marca de 3,9 milhões de toneladas embarcadas para mais de 150 países (ABPA, 2015).

Junto ao crescimento do setor, vieram também as exigências do mercado quanto à qualidade da produção, como o bem-estar dos animais, que além de ser de importância para o criador das aves, também passou a ser exigência dos importadores, das redes de supermercados, das cadeias de *fast-food*, etc. Dessa forma, a implantação de programas de qualidade e bem-estar animal passou a ser fundamental (UBABEF, 2008).

Com relação ao efeito da temperatura nos frangos de corte, resultados apontam para a existência de uma zona de conforto térmico que deve ser

mantida para que o animal se sinta bem, tal que o desconforto térmico pode acarretar problemas como a queda no consumo de ração, uma menor taxa de crescimento, um maior consumo de água, a aceleração no ritmo cardíaco, alterações na conversão alimentar, entre outros (FILHO, 2004).

Além da preocupação com o conforto térmico, a qualidade do ar em galpões avícolas também é fundamental. Pesquisas mostram que um alto nível de concentração de amônia pode implicar na produção de 5% a 10 % de carcaças com tamanhos abaixo do padrão (frangos que não atingem o tamanho ideal no período do lote), além de ocasionar doenças respiratórias tanto nos frangos quanto em seres humanos (OLIVEIRA, 2003) Assim, fica evidente que a sanidade das aves é indispensável para o emergente cenário da produção de frangos de corte no Brasil, cenário sobre o qual os avanços tecnológicos se mostram cada vez mais decisivos.

2.2 DESCRIÇÃO DO PROCESSO AVÍCOLA

Dentro da vasta gama de equipamentos disponíveis para as granjas avícolas, os mais comuns são os sistemas de distribuição de ração, de fornecimento de água e de climatização. Nesses sistemas, o controle adequado da temperatura e da umidade são variáveis que se destacam e desafiam a correta implementação das leis de controle.

Basicamente, o funcionamento desses sistemas baseia-se na leitura de sensores instalados em alguns pontos do aviário, predominantemente 2, para uma extensão que varia de 100 a 200 metros. Ao detectar uma temperatura muito alta, automaticamente os exaustores são acionados até que se atinja o valor desejado. Do contrário, ao detectar uma temperatura menor que o parâmetro, aberturas de aquecimento são acionadas.

Diariamente, o avicultor se desloca até o painel em que se encontra o controlador e fornece os parâmetros desejados de temperatura e umidade como entrada ao sistema. Pode também ativar ou desativar manualmente o

funcionamento dos exaustores. O sistema automático se encarrega de efetuar as medições de temperatura e umidade, e atuar ligando mais ou menos exaustores, e também acionando alarmes. A Figura 1 mostra um exemplo de painel de usuário.

Figura 1: Típico painel de controle de um sistema de automatização de aviário. Exemplo 1. Fotografia: Autoria própria.

A Figura 2 mostra outro exemplo de painel de controle e os botões utilizados para realizar intervenções manuais nos atuadores do aviário.

No display do painel de controle é possível visualizar informações em tempo real sobre a temperatura e a umidade no interior do aviário, atualizar os parâmetros do controlador ou acionar/desligar equipamentos manualmente. Apesar da aparente eficiência, algumas limitações são perceptíveis na tecnologia empregada, as quais desafiam a demanda por produtividade. A Figura 3 mostra a arquitetura mais comum encontrada em galpões de criação de frangos de corte que possuem um sistema de controle automático de temperatura e umidade.

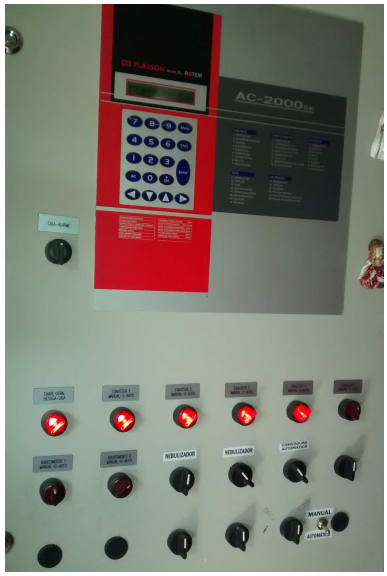

Figura 2: Típico painel de controle de um sistema de automatização de aviário. Exemplo 2. Fotografia: Autoria própria

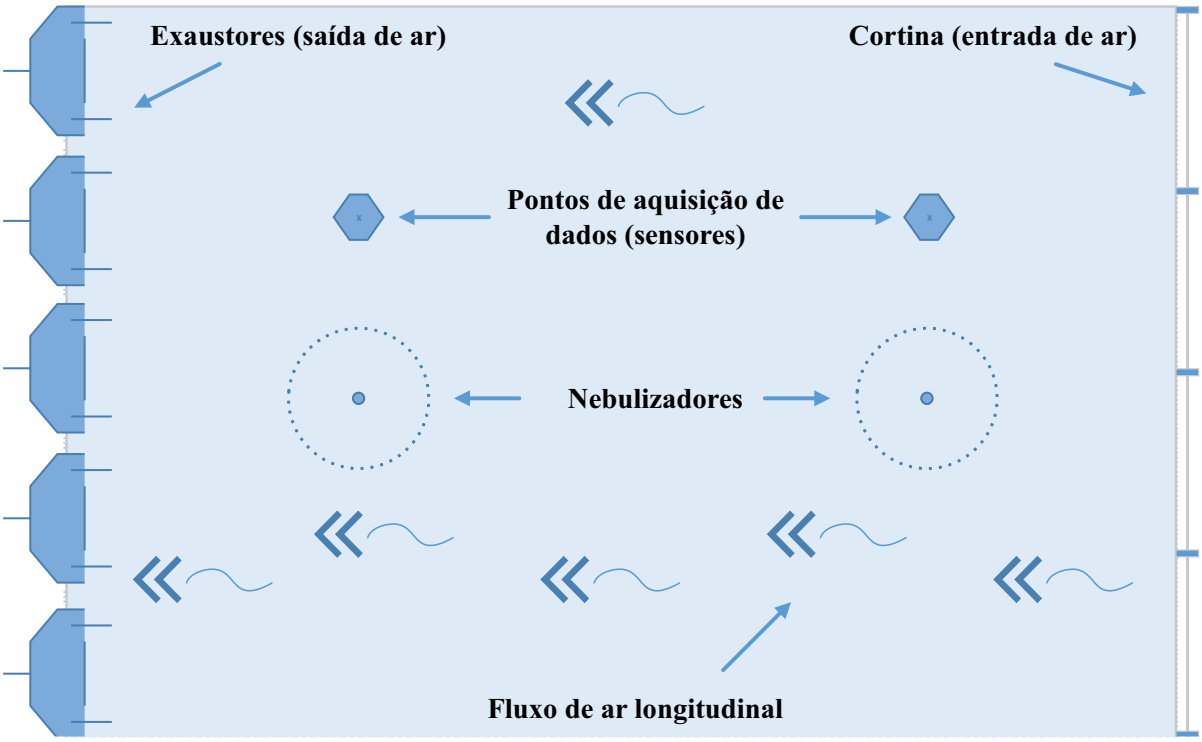

Figura 3: Arquitetura tradicional de um aviário automatizado.

No modelo de arquitetura mostrado acima, os exaustores localizam-

se em uma das extremidades do galpão, e a entrada de ar localiza-se na extremidade oposta a eles. O sistema de controle de temperatura e umidade conta com a leitura de um ou dois sensores distribuídos no interior do aviário, sendo que os controladores suportam, em geral, em torno de 3 pontos de sensoriamento.

O ar que circula no interior do galpão entra através das aberturas da cortina, e sai através dos exaustores, gerando um fluxo linear que passa por toda a extensão do aviário. Isso leva a uma característica indesejável, que é a diferença encontrada nos valores de temperatura medidos nas extremidades do galpão, que se dá pelo aquecimento do ar enquanto percorre o ambiente interno.

Uma alternativa para tal problema, inclusive admitida por avicultores entrevistados e sinalizada por tecnologias emergentes para aviário, seria promover a entrada e a saída lateral de ar. Porém, nesse caso, o sistema de controle atual não seria diretamente escalável. Ocorre que apenas um ou dois sensores não são suficientes para identificar discrepâncias nos valores medidos ao longo do aviário, uma vez que as construções possuem medidas que variam de 10m a 14m de largura e de 100m a 200m de comprimento, o que representa uma vasta área suscetível a variações climáticas em seu interior. Uma rede de sensores surge então como alternativa natural, mas requer que seja implementada em conjunção com as transformações da arquitetura.

2.3 QUALIFICAÇÃO TÉCNICA DE UM PROCESSO AVÍCOLA

Esta seção estabelece o enquadramento técnico do processo avícola descrito na seção anterior. Essa dinâmica sugere que um SED seja mais naturalmente representado por um diagrama de transições do que, por exemplo, através de uma equação diferencial, como é o caso dos sistemas cuja dinâmicas são regidas pelo tempo. Usar um diagrama de transição (uma máquina de estados por exemplo) permite que a modelagem captureA automatização

de um processo avícola depende basicamente da leitura de sensores e da tomada de um conjunto de ações conforme essa leitura. O disparo de sinal de sensor é claramente um estímulo esporádico, assíncrono e independente do tempo. Aos sistemas que compartilham essa dinâmica, ou seja, que possuem um conjunto enumerável de estados e cuja dinâmica de transição é guiada por estímulos discretos no tempo, dá-se o nome de *Sistemas a Eventos Discretos* (SEDs) (CASSANDRAS; LAFORTUNE, 2008). Exemplos de SEDs incluem sistemas robóticos, de manufatura, de supervisão de tráfego, de logística, gerenciadores de bases de dados, entre outros.

Na prática, os SEDs têm em comum a maneira pela qual percebem as ocorrências no ambiente a sua volta, o que se dá pela recepção de estímulos, denominados *eventos*. Exemplos de eventos incluem o início e o término de uma tarefa, a percepção de uma mudança de estado em um sensor, etc. Entre a ocorrência de dois eventos consecutivos o sistema permanece num determinado estado. A ocorrência de um evento causa então uma transição de estado, de forma que sua evolução no tempo pode ser representada pela trajetória percorrida unicamente no seu espaço de estados (CURY, 2001).

tão somente o comportamento que é relevante em um contexto discreto.

2.4 MODELAGEM DE SEDS

A etapa de modelagem de um SED se justifica por nem sempre ser possível ou seguro interferir experimentalmente sobre a sua estrutura real. Além disso, um modelo facilita a compreensão e a manipulação do sistema, etc. (TEIXEIRA, 2013). Neste trabalho, um modelo para o processo avícola é definido e utilizado para a síntese de um controlador para o processo. Argumenta-se que a definição de todas as sequências operacionais do controlador poderia ser de elevado dispêndio caso fosse conduzida de maneira empírica, além de

levar a um resultado cuja consistência precisaria ainda ser checada.

A estrutura discreta utilizada neste trabalho para modelar o processo avícola é centrada na *Teoria dos Autômatos e Linguagens*. Basicamente, *Linguagens* são formalismos que podem ser usados para descrever comportamentos discretos (CASSANDRAS; LAFORTUNE, 2008). A base de uma linguagem são os *eventos*, tomados de um *alfabeto* finito Σ, onde Σ^* denota o conjunto de todas as *cadeias* finitas de eventos compostas a partir de Σ, incluindo a *cadeia vazia* ε. Finalmente, a linguagem pode ser formalmente introduzida como um subconjunto $L \subseteq \Sigma^*$. O *prefixo-fechamento* de uma linguagem L é $\overline{L} = \{ s \in \Sigma^* \mid st \in L \text{ para algum } t \in \Sigma^* \}$.

Se L pode ser expressa por um número finito de recursos, então ela é dita ser *regular*, do contrário ela é *irregular*. Na prática industrial, regularidade é uma característica de interesse, já que nesse caso a linguagem pode ser representada por uma estrutura finita de modelos que, portanto, podem ser considerados adequados para o tratamento computacional.

Uma das estruturas que podem ser usadas para o reconhecimento formal de linguagens são os *Autômatos Finitos (determinísticos)* (AFs). Os AFs possuem como principal característica o fato de serem simples e intuitivos, ao mesmo tempo em que preservam o rigor formal. Um AF pode ser usado não apenas para reconhecer linguagens, como também para definir e operar sobre as suas propriedades.

Formalmente, um AF é uma 5-tupla $A = \langle \Sigma, Q, q^\circ, Q^\omega, \rightarrow \rangle$, em que:

- Σ é o alfabeto de eventos;
- Q é o conjunto de estados;
- $q^\circ \in Q$ é o estado inicial;
- $Q^\omega \subseteq Q$ é o subconjunto de estados marcados, associados a tarefas completas; e

- $\rightarrow \subseteq Q \times \Sigma \times Q$ é a relação de transição.

Para dois estados $q_1, q_2 \in Q$, denota-se

$$\mathsf{q}_1 \xrightarrow{\sigma} \mathsf{q}_2$$

uma transição do estado q_1 para o estado q_2 com o evento $\sigma \in \Sigma$. $A \xrightarrow{s} q$ denota que uma cadeia s é possível em A.

Duas linguagens podem ser definidas a partir de A:

$$L(A) = \{s \in \Sigma^* \mid A \xrightarrow{s} q \in Q\}$$

e

$$L^{\omega}(A) = \{s \in \Sigma^* \mid A \xrightarrow{s} q \in Q^{\omega}\}.$$

$L(A)$ é a *linguagem gerada*, que contém todas as cadeias possíveis em A, enquanto $L^{\omega}(A) \subseteq L(A)$ é a *linguagem marcada*, que representa o subconjunto de cadeias de $L(A)$ que possuem a característica de conduzir a estados marcados.

Dois AFs $A = \langle \Sigma_A, Q_A, q_A^{\circ}, Q_A^{\omega}, \rightarrow_A \rangle$ e $B = \langle \Sigma_B, Q_B, q_B^{\circ}, Q_B^{\omega}, \rightarrow_B \rangle$, podem ser associados por composição síncrona, definida tal que:

$$A \parallel B = \langle \Sigma_A \cup \Sigma_B, Q_A \times Q_B, (q_A^{\circ}, q_B^{\circ}), Q_A^{\omega} \times Q_B^{\omega}, \rightarrow \rangle,$$

em que:

- $(q_A, q_B) \xrightarrow{\sigma} (q'_A, q'_B)$, se $\sigma \in \Sigma_A \cap \Sigma_B$;
- $(q_A, q_B) \xrightarrow{\sigma} (q'_A, q_B)$, se $\sigma \in \Sigma_A \setminus \Sigma_B$;
- $(q_A, q_B) \xrightarrow{\sigma} (q_A, q'_B)$, se $\sigma \in \Sigma_B \setminus \Sigma_A$.

Na operação $\parallel$, os eventos compartilhados pelos dois AFs que estão sendo compostos são sincronizados, i.e., as transições são fundidas para o

mesmo evento. Do contrário, ou seja, caso os eventos não forem compartilhados pelos dois AFs, então eles sofrem o *interleaving*, que nada mais é do que a não-fusão da transição tal que o modelo do sistema evolui em qualquer ordem.

A Figura 4 ilustra um exemplo simples de um SED composto por dois subsistemas, M_1 e M_2, modeladas respectivamente por G^1 e G^2, tal que $G = G^1 \parallel G^2$.

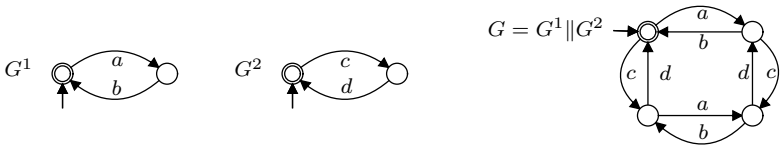

Figura 4: Modelagem da planta de um SED.

Assume-se que o comportamento das máquinas M_1 e M_2 é descrito em termos de eventos de início e final de operações. Os eventos a e c modelam os respectivos inícios, enquanto b e d modelam os finais de operações. Quando compostos, G^1 e G^2 resultam no modelo global da planta, G, tal que $L(G)$ representa o comportamentos gerado em malha aberta e $L^{\omega}(G)$ o seu comportamento marcado, representando tarefas completas pelo sistema.

Uma das principais vantagens da composição síncrona é permitir que os modelos sejam tratados de maneira mais simples e individual, gerando automaticamente o modelo completo do sistema. Um AF (ou uma composição de AFs) pode ser usado para representar o comportamento de um sistema, ao qual dá-se o nome de *planta*. Uma planta representa o comportamento irrestrito do sistema, conhecido como *malha aberta*. Na Figura 4, o autômato G representa o sistema em malha aberta.

Na prática, um sistema em malha aberta precisa ser restrito a certos padrões de comportamento para quando estiver em funcionamento. Logo, o modelo do sistema precisa ser complementado com uma estrutura denominada *especificação*. Uma especificação pode ser pensada como a representação de uma ação proibitiva no sistema. Quando associada ao modelo da planta, ela interfere adequando seu comportamento aos requisitos predefinidos, o que gera

um comportamento da planta em *malha fechada*. Um sistema pode possuir uma ou um conjunto de especificações, e cada uma pode ser representada individualmente através de um AF. A Figura 5 mostra o exemplo de um modelo de uma especificação que controla a planta apresentada na Figura 4.

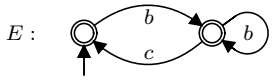

Figura 5: Modelagem de uma especificação.

Independentemente de relevância prática, a especificação E tem por objetivo controlar a planta G de maneira a evitar o início de operação da máquina M_2 (evento c proibido no estado inicial), sem que antes M_1 tenha finalizado uma operação (evento b).

2.5 CONTROLE SUPERVISÓRIO DE SISTEMAS A EVENTOS DISCRETOS

A construção da planta de um SED (G) e a inclusão de especificações de controle (E) resulta em um modelo composto $K = G \| E$ que representa o sistema em malha fechada. Sob certas condições, K pode ser utilizado para a implementação de um controlador ótimo para o sistema, ou seja, um controlador que detém uma lógica capaz de supervisionar o sistema e interferir no seu comportamento de forma minimamente restritiva e não-bloqueante.

Nesse caso, porém, estaria sendo assumido que todos os eventos possíveis de ocorrer na planta são passíveis de controle, i.e., eles são *controláveis*. Na prática, porém, esse pressuposto pode não ser totalmente adequado para o controle de processos industriais, já que é comum que alguns eventos possuam a característica de não permitirem uma interferência direta do controlador.

A perda de um sinal de comunicação, a quebra de um equipamento, a chegada de uma resposta remota, a conclusão de uma tarefa, por exemplo, caracterizam eventos cuja ocorrência é involuntária e independe de qualquer

política de controle. Logo, ao modelar tais eventos é necessário levar em conta esses aspectos, sob pena de ocasionar uma inconsistência semântica entre o sistema e sua solução de controle.

Uma abordagem formal que lida com a controlabilidade parcial de eventos em modelos de SEDs é provida pela *Teoria de Controle Supervisório* (TCS) (RAMADGE; WONHAM, 1989). A ideia básica é particionar o conjunto de eventos da planta de um SED tal que $\Sigma = \Sigma_{\mathrm{c}} \cup \Sigma_{\mathrm{u}}$, em que Σ_{c} é o conjunto de *eventos controláveis*, cuja ocorrência pode ser evitada na planta, e Σ_{u} é o conjunto contendo os *eventos não-controláveis*, os quais não podem ser diretamente desabilitados.

O elemento da TCS que efetivamente implementa as ações de controle na planta é denominado *supervisor*. Um supervisor é um mapa $S : L(G) \rightarrow 2^{\Sigma}$, associado a uma linguagem $L_{\mathrm{S}} \subseteq L^{\omega}(G)$ que, após qualquer cadeia $s \in L(G)$, observa eventos possíveis na planta e informa, dentre eles, quais devem ser, de fato, habilitados. Assim, a ação de controle de S sobre G (denotada por S/G) consiste em habilitar eventos do conjunto $S(s) \subseteq \Sigma$. A estrutura de supervisão que interage com a planta de forma a fechar a malha de controle, é ilustrada na Figura 6.

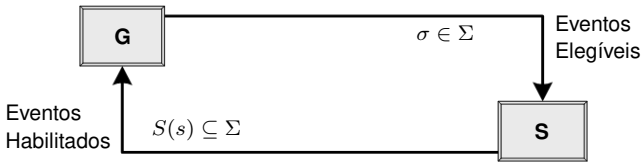

Figura 6: Fluxo de Controle

Assume-se que eventos $\sigma \in \Sigma$ ocorrem espontaneamente na planta. Quando σ é observado, após uma cadeia qualquer $s \in L(G)$, o supervisor S atualiza o conjunto $S(s)$ de eventos habilitados. Os eventos possíveis na planta que não pertencem a $S(s)$ são justamente aqueles que estão sendo inibidos pela ação de controle.

O conjunto de cadeias de $L(G)$ que sobrevive sob controle representa o *comportamento gerado em malha fechada*, e é dado pela linguagem

$L(S/G)$. O *comportamento marcado em malha fechada*, por sua vez, é dado por $L^{\omega}(S/G) = L(S/G) \cap L_{\mathrm{S}}$.

S é chamado *não-bloqueante* quando $L(S/G) = \overline{L^{\omega}(S/G)}$. Ou seja, sempre que uma cadeia sobrevive sob controle, essa cadeia é um prefixo de uma cadeia marcada. Assim, garante-se que a evolução do sistema sob controle sempre leva a completar uma tarefa e, logo, o sistema controlado nunca bloqueia.

A partir dessas definições, o *Problema de Controle Supervisório* (PCS) (RAMADGE; WONHAM, 1989) é tal que dada uma planta G, com eventos em Σ, e uma especificação $E \subseteq \Sigma^*$, definindo um comportamento desejado $K = E \cap L^{\omega}(G)$, encontre um supervisor não-bloqueante S tal que $L^{\omega}(S/G) \subseteq K$.

Encontrar uma solução para o PCS passa, sobretudo, pelo conceito de *controlabilidade*. Uma linguagem $K \subseteq \Sigma^*$ é dita ser *controlável* em relação a L quando

$$\overline{K}\Sigma_{\mathrm{u}} \cap L \subseteq \overline{K}.$$

Ou seja, após qualquer prefixo de K, se um evento não-controlável é observado em L, a cadeia resultante continua sendo um prefixo de K. Garante-se, então, que nenhum evento não-controlável, possível em L, esteja sendo desabilitado pela ação de controle.

A controlabilidade de uma linguagem não vazia $K \subseteq L^{\omega}(G)$ é uma condição necessária e suficiente para a existência de um supervisor marcador não-bloqueante S, tal que $L^{\omega}(S/G) = K$. Nesse caso, S, tal que $L^{\omega}(S/G) = K$, pode ser implementado por um autômato V, tal que $K = L^{\omega}(V) \cap L^{\omega}(G)$ e $\overline{K} = L(V) \cap L(G)$. A ação de controle de S é implementada pela desabilitação de eventos possíveis em $L(G)$, os quais não são possíveis em $L(V)$, após uma cadeia $s \in L(S/G)$. A função de marcação corresponde a marcar cadeias que são marcadas em ambos V e G.

Quando K não atende à condição de controlabilidade, torna-se necessário que se calcule a máxima sub-linguagem controlável, i.e., a linguagem que mais se aproxima de K. Obter tal linguagem é um processo iterativo que remove de K os estados a partir dos quais a controlabilidade não é observada. O resultado é um elemento supremo denotado por $\sup\mathcal{C}(K, G)$, representando a máxima linguagem controlável. Assim, $\sup\mathcal{C}(K, G)$ pode ser associado ao comportamento menos restritivo possível de ser implementado por um supervisor não-bloqueante S sobre G, de maneira a respeitar o conjunto de especificações.

2.6 IMPLEMENTAÇÃO DO CONTROLADOR

Obtido o controlador, a sua implementação passa naturalmente pela conversão do AF que representa $\sup\mathcal{C}(K, G)$ para código interpretável por hardware (microcontroladores, CLPs, etc.). Existem ferramentas que auxiliam muito este processo, desde a concepção do problema, a modelagem e até a implementação do controlador. O software Supremica é uma delas, sendo tipicamente utilizado para construir um modelo de sistema a ser controlado, construir especificações de como o sistema de controle em malha fechada deverá se comportar, gerar um supervisor que restringe o sistema ainda não controlado de tal modo que cumpra as especificações quando em malha fechada, etc. Oferece ainda recursos de geração de códigos para determinados dispositivos.

O Supremica possui uma interface que permite representar graficamente os modelos e as especificações levantados para um sistema e realizar a simulação do seu funcionamento. Este software é capaz de realizar a operação de composiçao síncrona sobre os autômatos, construindo um modelo controlável.

Outra ferramenta utilizada nesta aplicação é o Deslab, software que também permite a representação de autômatos, recebendo como entrada a descrição das transições e estados de cada modelo individualmente e, após re-

alizar a operação de composição síncrona de todos os modelos e especificações, em pares, gera um controlador ótimo para o sistema.

Um diferencial do software Deslab é a geração de códigos em linguagem C, para microcontroladores, que implementam fisicamente o comportamento do sistema antes simulado. O código gerado é uma estrutra genérica de máquina de estado que representa o autômato supervisor do sistema. O software permite que o usuário escolha entre a implementação do código em máquina de Mealy e máquina de Moore, em linguagem C.

As máquinas de Mealy e Moore diferem-se uma da outra na implementação, porém, seus comportamentos são idênticos. Na máquina de Mealy, cada estado do autômato contém apenas sua identificação (1,2,3...) e cada transição representa o valor das entradas e das saídas do sistema, ou seja, ao ocorrer um evento não controlável, o sistema executa uma ou mais ações na transição que o leva para o próximo estado. Já na máquina de Moore, cada estado contém sua identificação e também o valor das saídas do sistema correspondentes a ele, enquanto suas transições representam determinados eventos não controláveis a elas associados, executando então suas ações em cada estado, e não nas transições (MANTOVANI, 2010).

O software Deslab oferece a opção de código baseado em listas encadeadas e máquinas de estados com estruturas ”case-if”, dependendo do microcontrolador escolhido. Neste trabalho, a opção escolhida foi microcontrolador do tipo MSP430Gxx e geração de código baseado em listas encadeadas, utilizando uma máquina de Mealy.

O código exportado pelo Deslab pode ser utilizado para programar o microcontrolador que fará o gerenciamento das ações de controle, intermediando a comunicação do supervisor com os atuadores no meio físico.

2.7 SISTEMA DE SUPERVISÃO COM SCADABR

Implementado fisicamente o controlador, entra-se na fase de monitoramento do sistema sob controle. A importância dessa fase é decisiva na prática, pois implica diretamente em custos financeiros. Não são raros relatos de perdas de lotes inteiros de frango devido a problemas momentâneos de controle. Uma simples interrupção do sistema de exaustão, por exemplo, por alguns minutos, é suficiente para comprometer um lote.

Ademais, desde a chegada das aves no aviário até o dia em que são entregues para o abate, é necessário o monitoramento empírico das variáveis ambientes de maneira a se certificar de que elas estejam ideais para a respectiva fase de desenvolvimento dos frangos, que são animais muito sensíveis às condições climáticas. Por este motivo, a presença do avicultor é indispensável, o que caracteriza o manejo avícola uma atividade exaustiva e minuciosa.

Em entrevistas feitas com avicultores da região, foram relatados episódios em que os responsáveis pelos aviários tiveram que deslocar-se aos galpões várias vezes durante o dia e a noite, ou permanecerem próximos à propriedade, sem poderem se ausentar. Isso se deve ao fato de que, mesmo utilizando sistemas automáticos de controle de temperatura e de umidade, os parâmetros do processo devem ser alimentados manualmente e acompanhados constantemente.

Os avicultores argumentam ainda sobre a necessidade de estarem presentes no aviário para que, com sua experiência, analisem o comportamento das aves e, empiricamente, verifiquem o seu bem-estar. Alguns trabalhos recentes, inclusive, investigam a análise de imagens e sons para determinar o bem-estar das aves, que é visivelmente distinto para cada parâmetro climático (FILHO, 2004; PEREIRA *et al.*, 2013) resultando maior em movimentação, aglomeração, retração, etc.

Através de um sistema de controle e monitoramento remoto, é possível

realizar o monitoramento por meio de um computador ou smartphone.

O ScadaBr é um recurso computacional da família dos sistemas SCADA (*Supervisory Control and Data Acquisition*), nacional e de código aberto, que permite desenvolver esta tarefa. Através deste framework, o comportamento dos frangos pode ser acompanhado pela transmissão da imagem de câmeras instaladas no interior dos galpões, pela coleta dos parâmetros de temperatura e umidade via sensores, etc. Além disso, o ScadaBR conta com uma estrutura de alarmes que alertam de maneira eficaz para eventuais defeitos em equipamentos ou mau funcionamento do sistema. Também há a possibilidade de atuação via interface remota. Em conjunção, esses aspectos tornam mais eventual a presença do avicultor no galpão.

No escopo deste trabalho, o ScadaBR será utilizado para monitorar a comunicação entre um microcontrolador e os dispositivos de atuação dispostos no aviário. Com base nessa comunicação, é implementada uma interface que identifica de maneira intuitiva os valores medidos nos pontos de aquisição de dados e a ação em tempo real dos equipamentos de atuação. Além disso, a interface provê um esquema de interferência manual sobre o processo automático. Essa comunicação é realizada através do protocolo Modbus RTU.

2.8 PROTOCOLO MODBUS RTU

O protocolo Modbus foi desenvolvido pela *Modicon Industrial Automation Systems*, a atual *Schneider Eletric*, como um meio de comunicação entre dispositivos, sendo eles um mestre e um ou mais escravos. Embora seja utilizado normalmente sobre conexões seriais padrão RS-232, ele também pode ser usado como um protocolo da camada de aplicação de redes industriais, tais como TCP/IP sobre Ethernet e MAP (FILHO, 2002).

Este protocolo define uma estrutura de mensagem que os dispositivos reconhecem e utilizam, independentemente do tipo de rede sobre a qual se comunicam. Ele descreve a maneira como um controlador realiza pedidos

e responde outros dispositivos, segundo alguma função pré-definida, além de como os erros de envio e recepção de mensagens são detectados e notificados, estabelecendo um formato comum para o layout e conteúdo dos campos das mensagens (MODICON, 1996).

Segundo a documentação fornecida pela Modicon, os controladores que utilizam este protocolo de comunicação, em redes Modbus padrão, podem ser configurados para se comunicar utilizando um dos dois modos de transmissão existentes: ASCII e RTU. Os usuários podem implementar um dos dois modos, juntamente com os parâmetros da porta de comunicação serial, como identificação da porta serial, taxa de transmissão, modo de paridade, etc. O modo de transmissão e os parâmetros de configuração serial devem ser os mesmos para todos os dispositivos presentes na rede Modbus. O Modbus padrão, no qual existem os modos ASCII e RTU, define o conteúdo dos campos de mensagens, que são arranjos de bits transmitidos de maneira serial, e como as informações são organizadas nos campos do pacote de dados, além de como serão decodificadas pelo dispositivo que as receberem.

A comunicação entre dispositivos utilizando o protocolo Modbus é baseada no modelo mestre-escravo, onde um único dispositivo da rede, o mestre, pode iniciar transações, ou troca de mensagens, e os demais, chamados escravos, respondem, suprindo os dados requisitados pelo mestre, ou executando uma ação por ele comandada (FILHO, 2002). Quando o mestre envia uma mensagem endereçada a um escravo, apenas o dispositivo endereçado retorna uma resposta, como mostra a Figura 7.

O mestre inicia a comunicação enviando um byte com o endereço do escravo para o qual se destina a mensagem. Ao enviar a resposta, o escravo também inicia o telegrama com o seu próprio endereço. O campo “código da função” também contém um único byte, onde o mestre especifica o tipo de serviço ou função solicitada ao escravo (leitura, escrita, etc.). De acordo com o protocolo, cada função é utilizada para acessar um tipo específico de dado. As

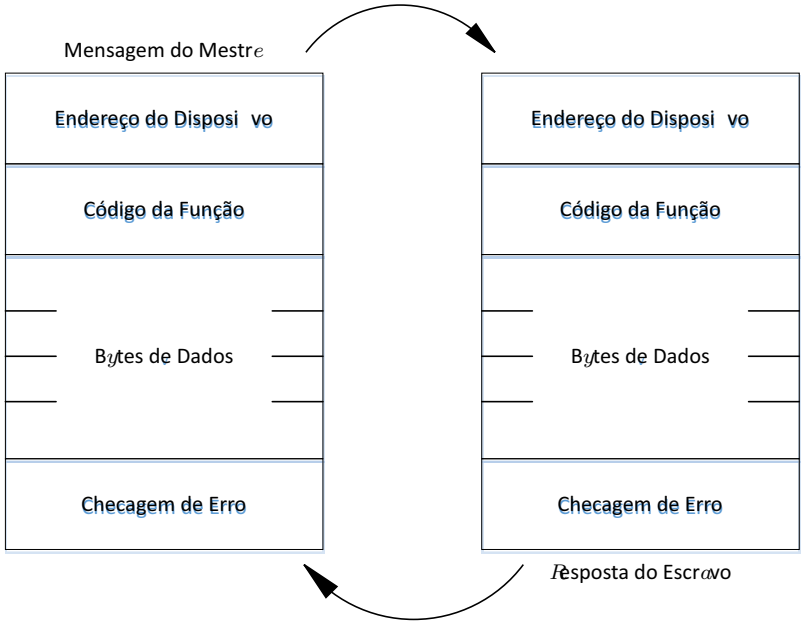

Figura 7: Ciclo de mensagens genéricas do Modbus trocadas entre mestre e escravo.

funções são pré-definidas para este protocolo. O campo “bytes de dados” possui tamanho variável e o formato e conteúdo deste campo dependem da função utilizada e dos valores transmitidos (WEG, 2013). Por último, há um mecanismo de checagem de erro, finalizando a construção do pacote.

Neste trabalho, o protocolo de comunicação utilizado para troca de informações entre o microcontrolador MSP430G2553 e o sistema ScadaBR, foi o Modbus RTU. A Figura 8 mostra o formato das mensagens trocadas entre os dispositivos sob este protocolo.

PACOTE GENÉRICO MODBUS RTU

Início	Endereço	Função	Dados	CRC	Fim
Tempo de espera 3 a 5 chars	8 bits	8 bits	N x 8 bits	16 bits	Tempo de espera 3 a 5 chars

Figura 8: Composição do pacote de informações do Modbus RTU.

Entre as mensagens é necessário um espaço de tempo de 3 a 5 *chars*, ou seja, o tempo necessário para transmitir de 3 a 5 bytes. O cálculo deve ser realizado baseado na taxa de transmissão escolhida.

O campo “endereço” pode assumir valores entre 0 e 247 (em hexadecimal, 0x00 a 0xf7), sendo que os dispositivos são endereçados com valores entre 1 e 247. O endereço zero é reservado para ser utilizado como endereço *broadcast*, ou seja, as mensagens destinadas ao endereço zero são reconhecidas por todos os elementos da rede. O campo “função” recebe o código da função que será utilizada para realizar a comunicação. O valor pode variar de 1 a 255 (0x01 a 0xff, em hexadecimal), porém, serão reconhecidos apenas valores que variam entre 1 e 127 (0x01 e 0x7f, em hexadecimal). Isso deve-se ao fato de que o bit mais significativo indica um código de diagnóstico de falha ou sucesso na comunicação. O campo “dados” pode variar conforme a função e o tipo da mensagem, requisição ou resposta. No caso de escrita em algum registrador, este campo pode representar o valor que se deseja escrever. Já o campo “CRC” (*Cyclical Redundancy Check*) contém o resultado de um mecanismo de checagem de erros (FILHO, 2002).

O campo CRC verifica o conteúdo de toda a mensagem. Ele é aplicado independentemente de qualquer método de verificação de paridade utilizado para cada um dos caracteres da mensagem. O CRC ocupa dois bytes do pacote, contendo um valor binário de 16 bits. Seu valor é calculado pelo dispositivo de transmissão, que acrescenta o CRC à mensagem. O dispositivo receptor refaz a operação durante a recepção da mensagem, e compara o valor calculado para o valor real que recebeu no campo CRC. Se os dois valores não forem iguais, ocorrerá um erro (MODICON, 1996).

A listagem a seguir apresenta um exemplo em linguagem C para o cálculo do CRC (CONTROL, 2013). A função recebe como entrada a variável que armazena o pacote de dados do Modbus RTU e seu comprimento. O retorno da função é o valor do CRC para o pacote de dados dado como parâmetro.

Neste exemplo, o valor de retorno é composto por dois bytes invertidos entre si, ou seja, para que a comparação de CRC esteja correta, é necessário trocar suas posições.

```
1
2  // Exemplo de código em C para cálculo do CRC
3  UInt16 ModRTU_CRC(byte[] buf, int len)
4  {
5    UInt16 crc = 0xFFFF;
6
7    for (int pos = 0; pos < len; pos++) {
8      crc ^= (UInt16)buf[pos];          // XOR byte into least
           sig. byte of crc
9
10     for (int i = 8; i != 0; i--) {    // Loop over each bit
11       if ((crc & 0x0001) != 0) {      // If the LSB is set
12         crc >>= 1;                    // Shift right and XOR
               0xA001
13         crc ^= 0xA001;
14       }
15       else                            // Else LSB is not set
16         crc >>= 1;                    // Just shift right
17     }
18   }
19   // Note, this number has low and high bytes swapped, so
        use it accordingly (or swap bytes)
20   return crc;
21 }
```

Durante a geração do CRC por meio do código apresentado, cada caractere de 8 bits passa pela operação XOR (lógica “ou exclusivo”) com o conteúdo atual do registro. Em cada uma das vezes, o resultado é deslocado na direcção do bit menos significativo (LSB), com um zero preenchido na posição

de bit mais significativo (MSB). O LSB é extraído e examinado. Se seu valor lógico for 1, o registro passa pela operação XOR com um valor fixo predefinido (0xA001). Se o LSB tiver valor lógico 0, ocorre apenas o deslocamento à direita novamente. Este processo é repetido até que oito mudanças sejam realizadas.

O protocolo Modbus RTU pode ser utilizado para comunicar o sistema ScadaBR com dispositivos como microcontroladores e CLP's.

3 RESULTADOS

Este capítulo é dividido em quatro macro seções, *Modelagem*, *Síntese*, *Implementação* e *Monitoramento*, cada uma descrevendo uma das etapas que constituem o sistema proposto no trabalho.

Para a implementação de um sistema de controle de temperatura e umidade baseado na TCS aplicado ao processo avícola, é ideal que as variáveis que se deseja controlar sejam monitoradas em vários pontos do ambiente. Na arquitetura tradicional, discutida na Seção 2.2, apenas 2 sensores cobrem uma vasta área de 100 a 200 metros de extensão, dificultando uma ação de controle objetiva. Nesse sentido, é fato que, quanto mais sensores forem distribuídos ao longo do aviário, melhor será o monitoramento climático como um todo e mais objetiva tende a ser a ação do controlador. Assim, uma tarefa imediata desse trabalho é aumentar o número de sensores ao longo da extensão do galpão.

Ao posicionar uma quantidade maior de sensores, é possível identificar em cada ponto de medição os valores de temperatura e umidade. Porém, sob o ponto de vista do sistema de controle, não basta apenas conhecer os pontos críticos do ambiente (em que as variáveis monitoradas não encontram-se em valores ideais), mas sim viabilizar ações baseadas nessas informações, manipulando variáveis até que o sistema alcance um estado térmico desejado.

Portanto, a tarefa de reposicionamento e expansão da quantidade de sensores pressupõe a divisão do espaço físico do galpão em setores de ventilação, onde cada setor recebe, não só um ponto de aquisição de dados, mas também atuadores locais. Cada setor é visto isoladamente, tornando o sistema facilmente escalável e mais robusto, uma vez que problemas em equi-

pamentos de um setor não afetarão os demais. A Figura 9 ilustra a forma assumida pela nova organização dos componentes do sistema.

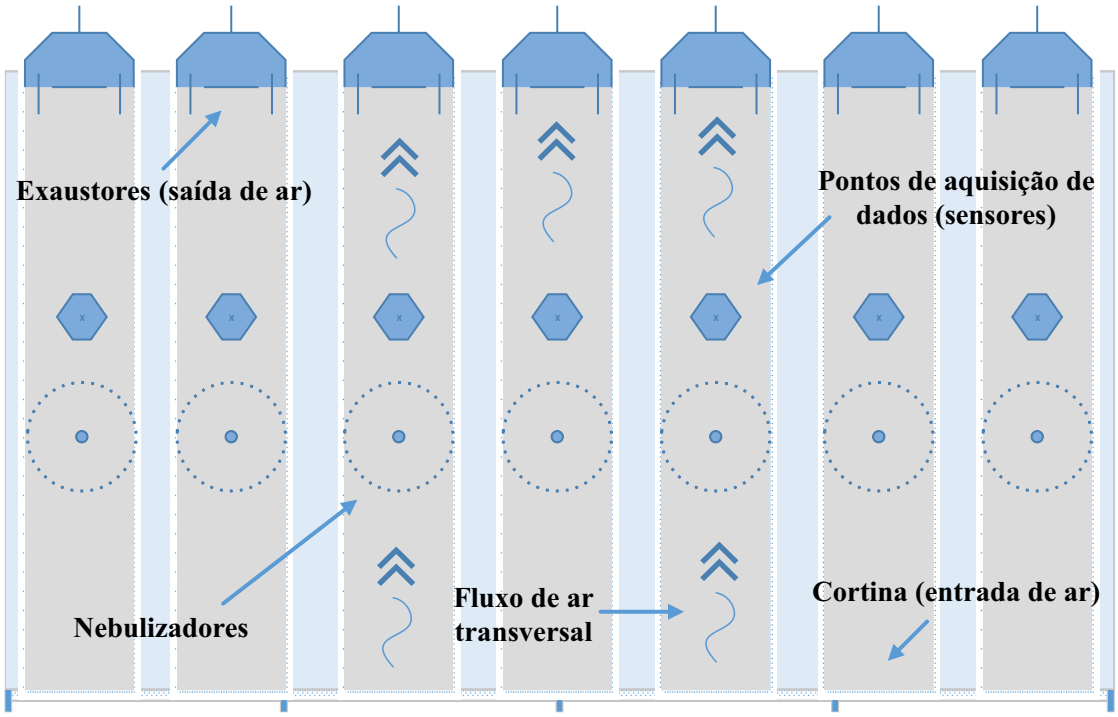

Figura 9: Modelo de arquitetura proposta para o novo sistema de controle do aviário.

Note que o fluxo de circulação interna de ar passa a ser transversal, o que sugere um melhoramento na homogeneização da temperatura e da umidade, uma vez que a distância percorrida pelo ar entre a entrada e a saída é menor em relação ao modelo tradicional, o que diminui seu superaquecimento.

Salienta-se ainda que, nessa nova arquitetura, a atuação local dos componentes do sistema pode interferir no comportamento dos componentes vizinhos. Entretanto, essa interferência é absorvida pelo sistema de controle proposto, que será apresentado na sequência.

3.1 ETAPA DE MODELAGEM

A partir da arquitetura assumida, o sistema de controle a ser implementado toma claramente um formato modular, em que cada setor contará com uma estrutura de controle idêntica ao dos demais setores. Desse modo, a partir daqui, assume-se a implementação de um único controlador, para um único setor do aviário. Naturalmente, os resultados podem ser replicados a quantos forem os setores.

3.1.1 MODELAGEM DA PLANTA DO SISTEMA

Os seguintes componentes (subsistemas) passam a compor cada setor do sistema: um *exaustor*, um *sensor de temperatura*, um *sensor de umidade* e um *nebulizador*. A seguir são apresentados os modelos de cada subsistema.

A Figura 10 mostra o modelo que representa o comportamento de um exaustor, cuja ação assume três níveis de potência: mínima, intermediária e máxima, o que evita a alteração brusca da circulação interna de ar no ambiente e evita ainda o desperdício de potência nos casos em que a exaustão é mínima, o que é bastante frequente, especialmente em dias frios. Estes níveis de ventilação dependem da implantação de mecanismos apropriados nos exaustores que permitam seu funcionamento em diferentes fases. Os eventos que compõem o modelo do exaustor são descritos a seguir:

- Lex_1 - Liga exaustor no primeiro nível de ventilação (de menor intensidade);
- Lex_2 - Liga exaustor no segundo nível de ventilação (com intensidade intermediária);
- Lex_3 - Liga exaustor no terceiro nível de ventilação (em sua máxima intensidade);

- Dex_3 - Desliga exaustor, passando-o do nível máximo para o intermediário;
- Dex_2 - Desliga exaustor, passando-o do nível intermediário para o primeiro nível;
- Dex_1 - Desliga exaustor.

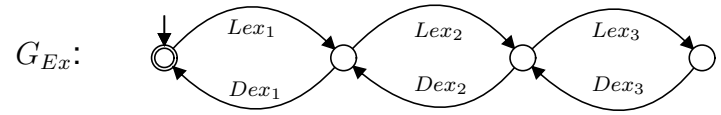

Figura 10: Modelo do exaustor.

De acordo com o modelo apresentado, cada fase do exaustor será ligada incrementalmente conforme o aumento da temperatura, ou seja, os eventos Lex_1, Lex_2 e Lex_3 serão associados com a respectiva leitura do sensor de temperatura. Ideia análoga se aplica aos eventos de redução da potência e desligamento do exaustor.

Outro dispositivo atuador presente no sistema de climatização é o nebulizador, e o modelo que descreve suas possíveis ações é mostrado na Figura 11, onde:

- n_x - Liga nebulizador;
- n_{off} - Desliga nebulizador.

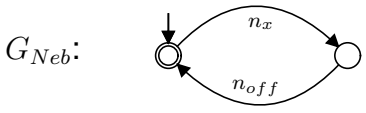

Figura 11: Modelo do nebulizador.

O comportamento útil observável do nebulizador é composto apenas pelos eventos de ligar e desligar o equipamento. Na prática, esse equipamento só é acionado em casos extremos, em que o sistema de exaustão alcança a potência máxima de atuação e, ainda assim, a temperatura não cede. Nesse caso, a umidade associada ao fluxo de ar, reduz drasticamente a temperatura,

em poucos segundos. Em outros casos, essa técnica não é aconselhável, pois ela implica no aumento significativo da umidade interna, o que pode ser nocivo se não controlada.

A conjunção do exaustor e do nebulizador, modelados anteriormente, compõe o sistema de atuação que será controlados pelo sistema proposto nesse trabalho. Para que as ações de controle aconteçam efetivamente, no entanto, elas necessitam de eventos que as desencadeiem. Esses eventos são os sinais dos sensores de temperatura e umidade, que, na prática, são lidos periodicamente, fornecendo informações nas quais o sistema se baseia para a tomada de decisões. A seguir são apresentados os modelos de cada sensor.

A Figura 12 mostra o modelo do sensor de temperatura, onde:

- t_x - Temperatura com valor maior que o ideal;
- t_{xx} - Temperatura com valor maior que tx;
- t_{xxx} - Temperatura com valor maior que txx;
- t_{ok} - Temperatura ideal.

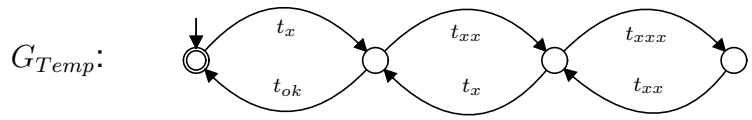

Figura 12: Modelo do sensor de temperatura.

O modelo deste sensor transitará entre os estados à medida em que atingir os limites do intervalo definido para cada medição de temperatura, indicando que a mesma está se elevando gradativamente, ou diminuindo. Partindo do estado inicial, irá para o próximo estado quando a temperatura subir (evento t_x). Se a temperatura continuar subindo, mudará de estado sucessivamente ao alcançar o limite da faixa, por exemplo, 26ºC, 28ºC e 30ºC. Tais parâmetros serão informados automaticamente, mas, sem perda de generalidade na ação

de controle, poderão ser reconfigurados pelo usuário. O retorno ao estado inicial segue ideia análoga.

Agora, a Figura 13 apresenta o modelo do sensor de umidade, onde:

- u_{baixa} - Umidade abaixo do valor ideal;
- u_{ok} - Umidade com valor aceitável.

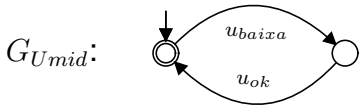

Figura 13: Modelo do sensor de umidade.

Este modelo mostra que o sensor de umidade terá dois pontos de transição de estados, o primeiro é a detecção de umidade baixa no ambiente, onde passará para o próximo estado, e retornará ao estado inicial ao detectar que o valor medido está em uma margem aceitável. Neste trabalho, não considera-se a modelagem de um problema envolvendo índices de umidade alta. Por se tratar de um ambiente controlado e coberto, subentende-se que a mera ação dos exaustores já implica na "secagem" do ambiente.

3.1.1.1 ESTATÍSTICA DE MODELAGEM DO SISTEMA

Na modelagem do sistema composto pelos submodelos apresentados anteriormente, foi assumido o particionamento do conjunto de eventos de maneira tal que

$$\Sigma_{\mathrm{c}} = \{Lex_1, Lex_2, Lex_3, Dex_1, Dex_2, Dex_3, n_{on}, n_{off}\}$$

e

$$\Sigma_{\mathrm{u}} = \{t_x, t_{xx}, t_{xxx}, t_{ok}, u_{baixa}, u_{ok}\}.$$

Após obter os modelos individuais dos subsistemas, os mesmos fo-

ram compostos entre si para formar a planta G do sistema, tal que

$$G = G_{Ex} \| G_{Neb} \| G_{Temp} \| G_{Umid}.$$

G corresponde a um autômato com 64 estados e 320 transições, e modela o comportamento do sistema em malha aberta. Para fechar a malha de controle, são modeladas especificações para o sistema.

3.1.2 MODELAGEM DAS ESPECIFICAÇÕES DE CONTROLE

Conforme a TCS, as especificações são modeladas, também por autômatos, e então são combinadas ao sistema por composição síncrona. O efeito disso é a restrição do sistema, a fim de evitar que situações indesejadas, ou ações proibidas, aconteçam. Dessa composição também deriva-se a entrada para o algoritmo de síntese ótima, conforme a TCS.

Abaixo são apresentadas as descrições textuais das oito especificações determinadas para o sistema de controle do aviário.

- E_1 - Proíba ligar o primeiro nível do exaustor (Lex_1) enquanto a temperatura não atingir o nível t_x;
- E_2 - Proíba ligar o segundo nível do exaustor (Lex_2) enquanto a temperatura não atingir o nível t_{xx};
- E_3 - Proíba ligar o terceiro nível do exaustor (Lex_3) enquanto a temperatura não atingir o nível t_{xxx};
- E_4 - Proíba o desligamento do exaustor enquanto a temperatura não atingir o nível t_{ok};
- E_5 - Proíba o desligamento do segundo nível do exaustor enquanto a temperatura não atingir o nível t_x;

- E_6 - Proíba o desligamento do terceiro nível do exaustor enquanto a temperatura não atingir o nível t_{xx};
- E_7 - Proíba o acionamento do nebulizador enquanto a temperatura não atingir o nível t_{xxx} e a umidade estiver baixa (u_{baixa}), em qualquer ordem de ocorrência;
- E_8 - Proíba o desligamento do nebulizador até que uma das condições para tal seja satisfeitas.

Agora, a semântica de cada uma das especificações é expressa por um autômato, conforme mostra a Figura 14.

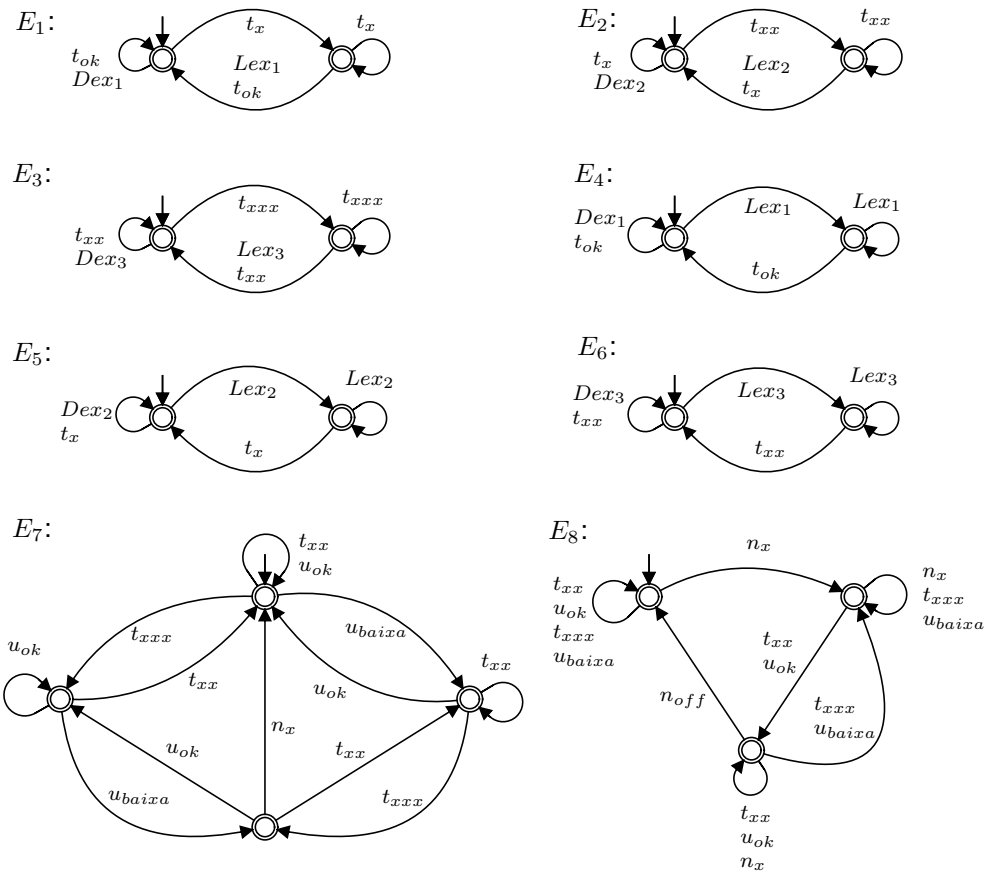

Figura 14: Modelos para as especificações do sistema.

As especificações devem ser interpretadas como ações restritivas ao sistema. Por exemplo, a especificação E_1, quando em seu estado inicial, não permite que o exaustor ligue (evento Lex_1). Este evento só estará habilitado a ocorrer após t_x, que indica que a temperatura está acima do normal. As especificações E_2, E_3, E_4, E_5 e E_6 são interpretadas de maneira análoga.

Também tem-se modelos diferentes, o que é o caso da E_7. Este modelo assume que o nebulizador só poderá ser ligado (evento n_x), após a ocorrência de t_{xxx} e u_{baixa}, em qualquer ordem, que são as duas condições necessárias para habilitar este equipamento. Já a E_8 é responsável por garantir que o nebulizador só poderá ser desligado (evento n_{off}), se o sistema identificar que a temperatura diminuiu t_{xx}, ou que a umidade está em um valor aceitável u_{ok}.

3.2 SÍNTESE DE CONTROLE

Após ter modelado a planta G e ter obtido os modelos individuais das especificações $E_i,\ i = 1, \cdots, 8$, pode-se proceder com a síntese de controle. Para isso, toma-se como especificação um modelo global

$$E = \mathop{\|}_{i=1}^{8} E_i.$$

E corresponde a um autômato com 560 estados e 4344 transições, e pode ser combinado à planta para gerar o comportamento do sistema em malha fechada, denotado por K, tal que $K = E\|G$. K corresponde a um autômato com 503 estados e 1606 transições.

A partir de K, foi sintetizado um supervisor $\sup\mathcal{C}(K, G)$, o qual corresponde a um autômato com 503 estados e 1606 transições. Observe que K corresponde a um autômato com o mesmo número de estados do autômato que representa $\sup\mathcal{C}(K, G)$. De fato, ao testar a controlabilidade de K, percebeu-se que se tratava de um modelo controlável. Portanto, K foi usado para a

implementação de um controlador ótimo para o sistema proposto.

3.3 IMPLEMENTAÇÃO

O controlador do sistema, $\mathrm{sup}\mathcal{C}(K, G)$, possui a estrutura discreta que pode ser representada por um autômato finito. Um autômato é uma máquina de estados, pois é composto por estados e transições, lendo e manipulando entradas e saídas.

A implementação do autômato supervisor em microcontrolador torna-se muito trabalhosa e inviável, dependendo do seu tamanho. Para auxiliar neste processo existem ferramentas computacionais que geram estruturas de código que descrevem os estados e transições do controlador automaticamente. Neste trabalho, dois softwares foram utilizados como ferramentas de validação e implementação do controlador: o Supremica e o Deslab.

O supremica foi utilizado para a simulação e validação dos modelos. Através dos testes, é possível afirmar que os mesmos atendem às necessidades identificadas para o controle de temperatura e umidade no processo avícola. Após essa etapa, foi desenvolvido o código para microcontrolador na linguagem C, e para isso, o software Deslab foi utilizado.

A descrição do autômato que representa cada modelo foi fornecida como parâmetros de entrada no Deslab, que, através da operação de composição síncrona, chegou ao autômato supervisor do sistema. O $\mathrm{sup}\mathcal{C}(K, G)$ foi exportado em formato de código para microcontrolador, na linguagem C, pelo modelo de máquina de Mealy.

A escolha pela máquina de Mealy ao invés de Moore é viável pois o $\mathrm{sup}\mathcal{C}(K, G)$ do sistema contém um número muito menor de eventos associados às transições do que de estados (503 estados e 14 eventos que desencadeiam as transições). Pela máquina de Mealy, as ações do controlador acontecem nas transições do sistema associadas a determinados eventos, já pela máquina de

Moore, as ações do controlador acontecem nos estados. Portanto, a máquina de estados se torna menor e menos trabalhosa, uma vez que 14 eventos precisam ser programados, no lugar de 503 estados.

O código gerado pelo Deslab utiliza estruturas em listas encadeadas e destina-se ao microcontrolador MSP430G2553, da *Texas Instruments*. Este microcontrolador foi escolhido por ser considerado de baixo custo e baixo consumo de energia, além de ter capacidade de hardware suficiente para a atual aplicação. O software utilizado para a gravação do código no microcontrolador foi o *Code Composer Studio V. 6.1.1.00022*, também da *Texas Instruments*.

Para construção de um protótipo em *protoboard*, com a finalidade de testar o funcionamento do sistema de controle, foram utilizados *leds*, resistores e *push buttons*, como mostra a Figura 15. As leituras de temperatura e umidade realizadas pelo sistema são simuladas através dos *push buttons*. Cada *push button*, quando acionado, indica a ocorrência de um evento não controlável, ou seja, a leitura de um sensor, enquanto os *leds* indicam as saídas físicas do sistema, como equipamento ligado ou desligado.

A imagem acima mostra os componentes do protótipo, onde:

- L1 - Se 1 = Lex_1; se 0 = Dex_1;
- L2 - Se 1 = Lex_2; se 0 = Dex_2;
- L3 - Se 1 = Lex_3; se 0 = Dex_3;
- L4 - Se 1 = n_{on}; se 0 = n_{off};
- B1 - Representa o evento t_x;
- B2 - Representa o evento t_{xx};
- B3 - Representa o evento t_{xxx};
- B4 - Representa o evento t_{ok};

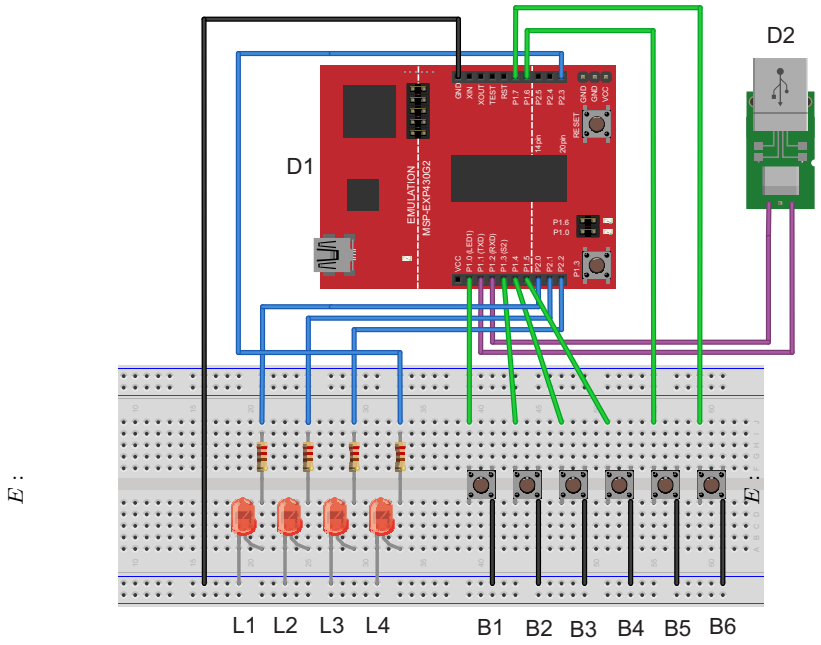

Figura 15: Protótipo construído em *protoboard*.

- B5 - Representa o evento u_{alta};
- B1 - Representa o evento u_{ok};
- D1 - MSP430G2553;
- D2 - Conversor USB-Serial RS485.

Uma característica importante do Deslab, é que ele interpreta o tipo de cada evento do sistema através de seu número de identificação. Os eventos ímpares são eventos controláveis, enquanto os eventos pares são eventos não controláveis. Cada vez que um botão é pressionado, o sistema entende que um evento não controlável ocorreu, e, através da identificação do botão, alguma ação é realizada. Para que a saída física seja mostrada através dos leds, o microcontrolador foi configurado como mostra a listagem de código abaixo.

```
1 if(mealy_output) //Se o evento ocorrido for válido, então
     imprimir saída física
2 {
```

```
3         switch(occur_event)
4         {
5                 //eventos controláveis
6                 case(1):        //Adicionar Ação para o
                     Evento 1;
7                       P2OUT |= BIT0; //Seta o BIT0 da
                           Porta 2
8                 break;
9                 case(3):        //Adicionar Ação para o
                     Evento 3;
10                      P2OUT |= BIT1; //Seta o BIT1 da
                           Porta 2
11                break;
12                case(5):        //Adicionar Ação para o
                     Evento 5;
13                      P2OUT |= BIT2; //Seta o BIT2 da
                           Porta 2
14                break;
15                case(7):        //Adicionar Ação para o
                     Evento 7;
16                      P2OUT &= ~BIT2; //Reseta BIT2 da
                           Porta 2
17                break;
18                case(9):        //Adicionar Ação para o
                     Evento 9;
19                      P2OUT &= ~BIT1; //Reseta BIT1 da
                           Porta 2
20                break;
21                case(11):       //Adicionar Ação para o
                     Evento 11;
22                      P2OUT &= ~BIT0; //Reseta BIT0 da
                           Porta 2
```

```
23                  break;
24                  case(13):       //Adicionar Ação para o
                       Evento 13;
25                        P2OUT |= BIT6; //Seta o BIT6 da
                             Porta 2
26                  break;
27                  case(15):       //Adicionar Ação para o
                       Evento 15;
28                        P2OUT &= ~BIT6; //Reseta BIT6 da
                             Porta 2
29                  break;
30
31                  //Aqui são listados os eventos não
                       controláveis (eventos pares)
32
33          }//fim switch
34          mealy_output = 0;
35          occur_event = -1;
36  }//fim if(mealy_output)
```

Os eventos configurados no código acima estão divididos da seguinte forma:

- Evento 1 - Lex_1
- Evento 3 - Lex_2
- Evento 5 - Lex_3
- Evento 7 - Dex_3
- Evento 9 - Dex_2
- Evento 11 - Dex_1

- Evento 13 - n_x
- Evento 15 - n_{off}

Cada evento, ou cada saída, está associada a um pino (ou bit) da porta 2 do microcontrolador. A configuração apresentada permite que o usuário visualize as ações de controle através dos leds, que representam o exaustor e o nebulizador.

3.4 MONITORAMENTO

Até o momento, as ações de controle já podem ser vistas pelo usuário do sistema através do protótipo construído em *protoboard*. A partir de agora, será mostrado o sistema de monitoramento e controle remoto, desenvolvido através da comunicação entre o software ScadaBR e o microcontrolador, pelo protocolo Modbus RTU.

O protocolo de comunicação foi implementado no microcontrolador e configurado no ScadaBR utilizando os mesmos parâmetros. O Modbus RTU funciona através da troca de mensagens pela interface serial. Para isso, foi utilizado o conversor USB-Serial Rs232 Ttl Pl2303, da *Prolific*.

O ScadaBR pode ler dados de posições de registradores ou escrever nestas posições, dependendo da função pré-definida pela qual está se comunicando. Para isso, é definido no microcontrolador o registrador que será lido (neste caso, os 8 bits da porta 2 do microcontrolador).

A configuração do *Data Source* no ScadaBR é mostrada na Figura 16.

Conforme a figura, o *Data Source* recebe o nome de *Modbus Serial*, e será atualizado a cada 500 milissegundos (ms). Ou seja, a cada período de tempo, os valores presentes no registrador vinculado à comunicação serão atualizados. O campo *timeout* representa um período de tolerância para a