MINISTÉRIO DA DEFESA

EXÉRCITO BRASILEIRO
DEPARTAMENTO DE CIÊNCIA E TECNOLOGIA
INSTITUTO MILITAR DE ENGENHARIA
CURSO DE GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

Ten CARLOS HENRIQUE PIMENTEL PAIVA
Ten RAFAEL SOUZA DA SILVEIRA
Ten JULIO CÉSAR LEITÃO ALBUQUERQUE DE FARIAS

A CASA INTELIGENTE: MODELAGEM E IMPLEMENTAÇÃO DE UM SISTEMA DE APRENDIZADO DAS PREFERÊNCIAS DOS MORADORES UTILIZANDO INTELIGÊNCIA ARTIFICIAL

Rio de Janeiro
2015

INSTITUTO MILITAR DE ENGENHARIA

Ten CARLOS HENRIQUE PIMENTEL PAIVA
Ten RAFAEL SOUZA DA SILVEIRA
Ten JÚLIO CÉSAR LEITÃO ALBUQUERQUE DE FARIAS

A CASA INTELIGENTE: MODELAGEM E IMPLEMENTAÇÃO DE UM SISTEMA DE APRENDIZADO DAS PREFERÊNCIAS DOS MORADORES UTILIZANDO INTELIGÊNCIA ARTIFICIAL

Projeto de Final de Curso apresentado ao Curso de Graduação em Engenharia de Computação do Instituto Militar de Engenharia.
Orientador: Sandro Santos de Lima – Ms.C

Rio de Janeiro
2015

INSTITUTO MILITAR DE ENGENHARIA
Praça General Tibúrcio, 80 – Praia Vermelha
Rio de Janeiro – RJ CEP: 22290-270

A Casa Inteligente: Modelagem e Implementação de um Sistema de Aprendizado das Preferências dos Moradores utilizando Inteligência Artificial. Carlos Henrique Pimentel Paiva; Júlio César Leitão Albuquerque de Farias; Rafael Souza da Silveira. - Rio de Janeiro: Instituto Militar de Engenharia, 2014.
Projeto de Final de Curso – Instituto Militar de Engenharia, 2015.

INSTITUTO MILITAR DE ENGENHARIA

Ten CARLOS HENRIQUE PIMENTEL PAIVA
Ten RAFAEL SOUZA DA SILVEIRA
Ten JÚLIO CÉSAR LEITÃO ALBUQUERQUE DE FARIAS

A CASA INTELIGENTE: MODELAGEM E IMPLEMENTAÇÃO DE UM SISTEMA DE APRENDIZADO DAS PREFERÊNCIAS DOS MORADORES UTILIZANDO INTELIGÊNCIA ARTIFICIAL

Projeto de Final de Curso apresentado ao Curso de Graduação em Engenharia de Computação do Instituto Militar de Engenharia.

Orientador: Sandro Santos de Lima – Ms.C.

Aprovada em 28 de maio de 2015 pela seguinte Banca Examinadora:

Prof. Sandro Santos de Lima – Ms.C, do IME

Prof. Ronaldo Ribeiro Goldschmidt – D.C, do IME

Prof. Julio Cesar Duarte – D.C, do IME

Rio de Janeiro
2015

RESUMO

A Domótica, também referenciada como Automação Residencial, permite que os usuários de uma casa, de um apartamento ou até mesmo de um ambiente corporativo possuam maior conforto. Ela permite automatizar processos rotineiros da casa e permite o controle remoto de dispositivos do ambiente. A Automação Residencial, porém, só é capaz de atuar em algum dispositivo da casa quando um usuário da casa solicita, ou seja, ele não é capaz de antecipar o que o usuário deseja que a casa faça. Nesse contexto surge o conceito de Domótica. A Domótica visa não só a automação, mas também a introdução de inteligência artificial no controle de dispositivos da casa.

Esse trabalho tem por finalidade o desenvolvimento de um sistema domótico que se adapte às preferências dos usuários de uma casa na medida que esses a utilizem. O sistema deve aprender as preferências através da análise de uma base de dados que expressem a utilização da casa pelos moradores. A utilização da casa pelos usuários pode ser expressa pelo estado em que a casa se encontra em diferentes momentos do dia e com diferentes usuários nos diversos cômodos da casa. O sistema é responsável por gerenciar os dados referentes à utilização da casa, perceber alteração de rotina dos usuários da casa e gerenciar ações que devem ser tomadas pela casa visando o bem estar dos usuários.

Esse trabalho visa contribuir na àrea de Domótica à medida que introduz Inteligência Artificial ao controle da casa, permitindo que essa atue sobre si mesma sem a necessidade de intervenção do morador. Esse trabalho mostrou-se importante uma vez que abrange várias áreas de Engenharia, como Computação e Eletrônica.

ABSTRACT

The home automation process provides more comfort to users of a house, an apartment or even a business environment. It allows you to automate routine processes of the house and allows remote control environmental devices. However, is only able to act on any device in the house when a user requests the house, that is, it can not anticipate what the user wants the house to do. In this context arises the concept of Home Automation. It aims not only automation but also the introduction of artificial intelligence in control of home devices.

This study aims to develop a home automation system that suits the preferences of users of these homes as home use. The system must learn the preference by analyzing a database which expresses the use by the users of the house. The use of the house by the users can be expressed by the state in which the house is at different times of day and with different users in different rooms of the house. The system is responsible for managing the data regarding the use of the house, noticing the change in routine users of home and manage actions to be taken by the house for the well being of the users.

This work aims to contribute in Home automation since it uses artificial intelligence in control of the house, allowing it to act on itself without the intervention of a user. This work proved to be important as it covers various areas of engineering, such as Computer and Electronics.

Sumário

1. Introdução

Desde os primórdios da humanidade, a implementação de sistemas de automação de tarefas realizadas por humanos sempre foi o grande motor de grandes mudanças tanto na área econômica quanto tecnológica. A automação proporcionou à humanidade a evolução do seu processo de agricultura, produção de bens de consumo e até de prestação de serviços. Dado que os equipamentos eletrônicos modernos estão convergindo para um preço mais baixo e também para uma capacidade de integração cada vez maior, espera-se que a automação residencial inteligente esteja dentre os grandes próximos saltos da evolução da automação. Com ela, o conforto, a segurança e a qualidade de vida serão substancialmente melhorados através de uma residência que possa realizar diversas tarefas, dentre elas monitorar o estado de saúde do morador, garantir a sua segurança, melhorar no gasto de recursos energéticos e tomar decisões de acordo com a preferência dos usuários.

1.1. Motivação

Trabalhos relacionados à automação residencial são motivados pela crescente demanda de segurança e conforto pela população em geral. Além disso, há uma grande necessidade de reduzir-se gastos energéticos de uma construção, sendo esse um prédio comercial, uma residência ou um ambiente corporativo. Podemos destacar também a tendência de envelhecimento da população em âmbito mundial. Com isso, novas tecnologias estão sendo produzidas visando prover melhoria no bem estar da população em geral.

1.2. Objetivo

Esse trabalho tem por finalidade o desenvolvimento de um sistema domótico capaz de automatizar processos da casa inteligente. Esse sistema deve ser capaz, também, de se adequar às preferências dos usuários.

1.3. Justificativa

Projetos na área de domótica inteligente provêm além de conforto, economia de energia e segurança para os usuário de uma casa. O campo da domótica vem se desenvolvendo em um ritmo cada vez maior. A situação atual da tecnologia é tão avançada que já estamos entrando na era de tecnologias vestíveis. Aprendizado de máquina está em amplo desenvolvimento. E nesse plano vem a ideia, há pouco tempo futurística, mostrada no famoso desenho *The Jetsons* onde encontra-se interação homem-máquina entre seus eletrodomésticos residenciais que proporcionam maior comodidade aos moradores por atuarem de forma concatenada.

Além do grande atrativo de maior conforto que a domótica pode conceder, a interação entre os eletrodomésticos conduz a uma possível redução de gastos energéticos em residências e também mais segurança.

1.4. Metodologia

Inicialmente será realizado um estudo referente a conceitos relacionados com domótica inteligente, suas aplicações, seu contexto atual e suas expectativas para o futuro.

A segunda parte deste trabalho está relacionada com o estudo das tecnologias responsáveis por tornar possível a existência de um sistema domótico inteligente.

Utilizou-se como base para a escolha das tecnologias de hardware aquelas que foram estudadas no trabalho Iniciação à Pesquisa (MARLIÈRE et al., 2013).

Ainda na segunda parte, em relação às tecnologias de software, são mostradas e justificadas quais ferramentas auxiliaram em tarefas tais como emitir comandos para os dispositivos da casa, receber estado da casa, utilizar aprendizado de máquina e prover interface com o usuário.

Uma vez conhecidas as ferramentas, o foco do trabalho parte para a abstração do problema necessária para a sua implementação. São descritas as entidades pertencentes ao sistema, os fluxos de funcionamento da casa, os diagramas de caso de uso, os diagramas de sequência e a arquitetura do sistema em geral. Logo após o entendimento do problema e tendo o comportamento de sua solução bem explicado, parte-se para a implementação do sistema em Python.

1.5. Estrutura

No capítulo 2 deste trabalho será realizado um estudo referente a conceitos relacionados com domótica inteligente, suas aplicações, seu contexto atual e suas expectativas para o futuro. Além disso, serão apresentadas as tecnologias de hardware presentes no trabalho. Dentre estas tecnologias, estão inclusos todos os dispositivos utilizados para a construção de uma rede de comunicação.

No capítulo 3 serão discutidos conceitos relacionados com a inteligência do sistema e outras tecnologias de *software*. Estão inclusas todas as frameworks utilizadas para o funcionamento sistema domótico.

No capítulo 4 deste trabalho será dada uma modelagem formal ao problema do aprendizado, abstraindo-se as classe necessárias para seu funcionamento e especificando-se a estrutura do software. Além disso são expostos trabalhos anteriores

sobre domótica e por fim os requisitos adotados pela equipe para a construção do sistema domótico proposto neste trabalho.

No capítulo 5 apresenta-se o processo de implementação da solução descrita pelo capítulo anterior, bem como uma apresentação do produto final do projeto.

Nos capítulos 6 e 7 estão, respectivamente, a conclusão e as referências bibliográficas do trabalho.

2. Domótica e o Hardware

A domótica pode ser definida com a automatização das atividades que os humanos fazem dentro de uma residência. Tal automatização ocorre através de equipamentos que possuem a capacidade de detectar informações do ambiente, seja por sensores diretamente ligados a eles ou por algum sistema mais complexo, para executar ações que substituem as dos humanos.

Pelo fato de ser capaz de realizar as atividades humanas automaticamente, ela provê conforto e segurança para os usuários no ambiente. Além disso, ela também pode prover economia de energia uma vez que os usuários de uma casa casualmente esquecem de desligar algum dispositivo desnecessário do lugar onde estão. Dentre as possibilidades mais comuns de controle realizado pela domótica, estão: controle de luminosidade, temperatura, abertura de cortinas, trancamento de portas, áudio, vídeo e alarmes de segurança.

Neste trabalho será abordado o ramo da domótica inteligente. Nela há a implementação de uma rede de dispositivos que se comunicam para que eles possam decidir algum acionamento com base nos conhecimentos prévios registrados. No decorrer do trabalho será descrita a especificação dos requisitos de uma domótica inteligente para que se possa ter uma noção do que usar para implementá-lo.

Dentre os elementos básicos da domótica inteligente, temos:

i) Sensores: são dispositivos capazes de captar alguma informação do ambiente para comunicar ao dispositivo central. São a peça chave para se conhecer o estado da casa e eles podem ser acoplados ao dispositivo fim (por exemplo uma lâmpada que também contém um sensor de luminosidade) ou independentes de dispositivos (por exemplo um sensor ultrassom com a finalidade de detectar presença).

ii) Controlador: é o elemento central da inteligência de um sistema domótico. Ele recebe todas as informações dos sensores e tem a capacidade de utilizar aprendizado de máquina para que ele mesmo emita algum comando de mudança de estado.

iii) Atuadores: são os elementos que fazem diretamente uma alteração no estado de um dispositivo final. Alguns atuadores, por exemplo, são responsáveis por fazer passar corrente elétrica em algum dispositivo (luz) ou então emitir algum sinal (infravermelho de ar condicionado).

2.1. Hardware

Após estudos do trabalho da disciplina de Iniciação à Pesquisa (IP), disponível em (MARLIÈRE et al., 2013), concluiu-se que seria adequado usar o protocolo CAN Bus para a comunicação entre os dispositivos, cujos motivos serão relembrados ao longo desse capítulo. Para poder concatenar todo o sistema, fez-se necessário também a utilização do protocolo de comunicação serial UART.

Para a implementação prática da rede CAN foi escolhido o microcontrolador PIC18F2580 e o dispositivo responsável por receber os dados, processá-los e tomar decisões para dispositivos finais é o Raspberry Pi.

2.1.1. Protocolo CAN

Na década de 80, a empresa alemã Robert BOSCH desenvolveu o CAN Bus (Controller Area Network) (GUIMARÃES, 2014). O CAN é um protocolo de comunicação serial síncrono que suporta controle distribuido de tempo real com um alto grau de segurança (BOSCH, 1991).

Robusto frente a falhas de comunicação e interferências externas. Possui mecanismos bastante eficientes de detecção e correção de erros na mensagem, no bus e nos nós. Permite apenas uma mensagem transmitida por vez no bus, selecionada através dos bits de prioridade. Faz o broadcasts de mensagem, destinadas a um ou mais destinatários, dependendo de seus endereços. Possui mecanismos de hierarquização e priorização. Funciona em emergência em modo de transmissão Single-Wire. Criado originalmente para a indústria automotiva para diagnóstico remoto e utilizado até hoje para essa finalidade com bastante êxito.

2.1.2. Rede CAN e o controlador central

Para a implementação da rede CAN foi utilizado o microcontrolador PIC18F2580, o transceiver mcp2551 e o Raspberry Pi. Conforme (MARLIÈRE et al., 2013), houve dificuldades de utilizar o protocolo CAN com o Raspberry Pi. A solução escolhida foi utilizar o protocolo UART entre o esse último e um PIC18F2580, o qual foi usado como mestre em uma rede CAN.

O microcontrolador que está presente na rede CAN BUS utilizado foi o PIC18F2580 desenvolvido pela Microship inc. Os motivos principais são o preço e disponibilidade do PIC e do gravador, já que a compra de outros hardwares iria atrasar o projeto, e a familiarização do PIC.

Já o microcontrolador que fez o controle central do sistema foi o Raspberry Pi. Conforme (MARLIÈRE et al., 2013), esse hardware torna possível a implementação de uma camada de rede em alto nível através de um servidor local ao mesmo tempo que controla sensores e atuadores analógicos e digitais. Sem contar o suporte a várias linguagens de programação e a execução de códigos sem a necessidade de programação do dispositivo, como exemplo python, que foi a linguagem de programação para implementar os programas que manipularão os pinos do UART.

2.1.3. Comunicação entre dispositivos

A arquitetura dos protocolos está dividida em três partes, no mais alto, o Raspberry Pi que se comunica via UART com o nó master, esse último com um nó secundário e que, por fim, manda mensagens para um atuador. A Figura 1 expressa o esquema de camadas de protocolos:

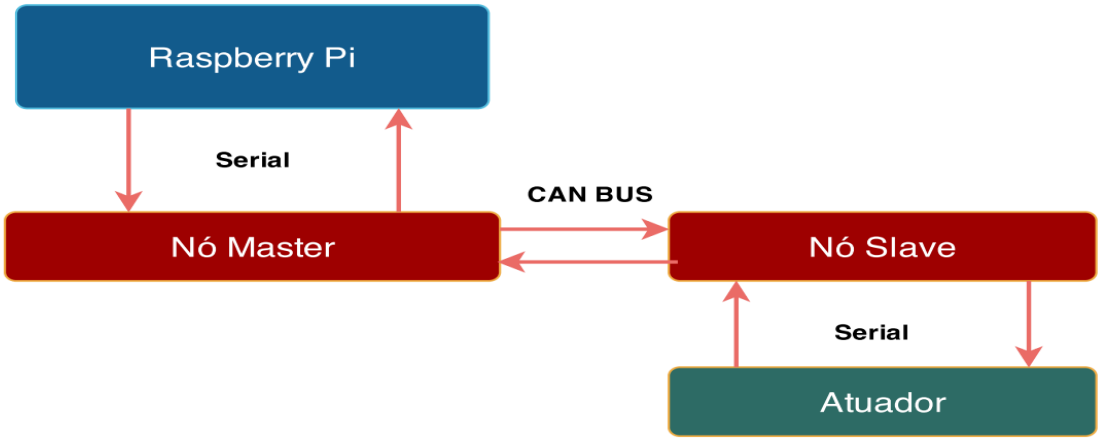

Figura 1: Camadas de protocolos.

Quando o Raspberry Pi deseja se comunicar com a casa, seja para executar uma ação ou para buscar os estados dos dispositivos nessa, é enviada uma mensagem via comunicação serial para o Nó Master. Essa comunicação ocorre através dos pinos GPIO oferecidos pelo Raspberry Pi que estão ligados diretamente ao Nó Master.

A mensagem enviada ao Nó Master pode ser de duas formas:

i) mensagem de mudança de estado: contém o tipo de mensagem (mudança de estado), a identificação do dispositivo final e o estado desejado no qual ele deve estar.

ii) mensagem de busca pelos estados: contém o tipo de mensagem (busca de estados).

No momento em que o Nó Master recebe alguma mensagem, ele verifica qual é o seu tipo e se comunica com o(s) Nó(s) Slave(s) que estão conectados com o(s) dispositivo(s) fim.

Quando o Nó Slave recebe alguma mensagem, ele verifica o tipo desta e solicita ao atuador a ação necessária. Se for uma mensagem de atuar no dispositivo conectado ao Atuador, esse executa a mudança. Caso seja de buscar o estado do dispositivo, ele obtém essa informação e envia ao nó master os dados necessários, que são: um aviso de que é uma resposta ao pedido de busca por estado, o tipo de nó (pois no caso geral pode-se definir varios tipos, ex.: iluminação, temperatura, cortina), a identificação do nó e o estado (ou os estados) do nó.

Para padronizar os valores de tais variáveis (referentes ao tipo de mensagem (setar ou buscar estados), estado do dispositivo, tipo do dispositivo) e definir como armazenar a informação a ser transmitida fêz-se necessário utilizar gramáticas, que foram construidas após estudo dos programas disponíveis em (HORTA et al., 2014) e modificação para atender as propostas de interface com o Raspberry Pi. Tais gramáticas são explicadas a seguir.

Caso seja uma mensagem de Request Update, o Raspberry Pi necessitaria somente um byte para informar que é um pedido de requerimento e um outro byte para dispor a numeração do nó a ser requisitado sobre os estados dos dispositivos finais, entretanto, para o código de transmição de mensagem e recepção, padronizar que sempre são enviados quatro bytes mais um de confirmação, tanto para a mensagem de setar quanto para a de requerimento de estado, facilita a implementação, portanto, são enviados dois bytes concatenados com dois bytes aleatórios.

2.1.4. Nós

A rede CAN Bus é disposta em topologia anel e os nós são definidos como pontos de conexão dessa rede com dispositivo eletrônico ativo capaz de enviar ou receber mensagens. Os nós podem se comunicar entre si via CAN Bus ou com dispositivos fora da rede utilizando variados tipos de protocolos que o PIC consiga implementar.

Tendo em vista as variadas aplicações em automação, optou-se primeiro por construir um sistema de iluminação autônomo que mostrasse que a rede CAN Bus funciona, bem como o código de processamento de dados fazendo o projeto adaptável a mais dispositivos finais.

Nós podem ser do tipo Master ou Slave, o primeiro controla a rede, envia mensagens de ajustes direcionadas aos outros nós, os nós tipo Slave, que devem obedecer comandos, sejam de prover informação ou de executar alguma tarefa fora da rede (agir nos dispositivos finais).

Nesse trabalho fez-se a implementação prática de um nó Slave, responsável pela iluminação, e um Master, entretanto, houve um grande esforço para funcionar um segundo nó Slave responsável por acionar dispositivos comandados por receptores de infravermelho.

2.1.5. Nó master

O nó Master deve receber tarefas (mensagens) de uma fonte e repassar para os Slaves utilizando o protocolo CAN Bus. Nesse projeto, essa fonte é o Raspberry Pi, que

tem dados a processar e funções a enviar para o Master poder redirecionar aos Slaves. Para manter tal conexão entre um PIC (nó) e o Raspberry Pi foi necessário, além do oscilador e seus capacitores, um transceiver, todos serão detalhados a seguir.

2.1.6. PIC

O nó, como dito, necessita enviar ou receber mensagens, no caso desse trabalho, os dois, para isso, utilizou-se o PIC 18F2580, um microcontrolador da Microchip que contém biblioteca direcionada ao protocolo CAN, além disso tem baixo custo de mercado e é comumente utilizado na cadeira de introdução aos microcontroladores lecionada no IME.

Para estabelecer a comunicação entre o PIC e o Raspberry Pi utilizou-se a porta USART e a comunicação entre dois PICs foi feita pelas portas CAN Tx e CAN Rx, necessitou-se ligar ao transceiver que será estudado adiante.

2.1.7. Oscilador e Capacitores

O oscilador em um microcontrolador é responsável por manter a frequência do clock de funcionamento para que a instruções ocorram de maneira correta. O oscilador escolhido foi um cristal de frequência 8.000 kHz, uma frequência comum no uso de microcontroladores.

Existem alguns esquemas de dispor o oscilador para o microcontrolador. É bastante comum usar a estrutura XT, na qual o cristal é ligado ao terra por meio de dois

capacitores de forma a estabilizar a sua frequência. A escolha desses capacitores é direcionada no data sheet do PIC. Foram utilizados capacitores de 22pF.

2.1.8. Transceiver

Transceiver, em português, transceptor, é usado para adequar os níveis de sinais elétricos distintos, no caso, adequa os níveis do barramento CAN Bus ao PIC. O transceptor escolhido foi o MCP2551, que é usado geralmente com o PIC18F2580 e é de mesmo fabricante.

A folha de dados informa que pode suportar até 1 Mbps que é suficiente para esse trabalho, além disso, possui operação com baixa energia, que é importante para esse trabalho.

2.1.9. Nó slave

O nó Slave deve receber e transmitir informações ao Master e atuar nos dispositivos finais. Semelhante ao nó Master, o nó Slave contém um PIC18F2580, seu oscilador e capacitores e transceiver. Nesse nó fez-se o acionamento de uma lâmpada, para isso foi necessário um banco de relés.

2.1.10. Banco de Relés

Os relés foram utilizados para chavear o circuito com a lâmpada. Foi escolhido um banco de relés do modelo srd-05vdc-sl-c, de fabricante Songle, por ser disponível a um dos integrantes do grupo e atender às especificações necessárias de voltagem de chaveamento (máxima de 250VAC) e amperagem (máxima de 10 amperes).

2.1.11.Firmware

O firmware foi desenvolvido na IDE mikroC PIC, em linguagem C, após estudo de programas disponíveis no Trabalho de Fim de Curso (HORTA et al., 2014) e modificações necessarias para interfaciar a rede CAN Bus e o Raspberry Pi.

O código contém parâmetros comuns aos PICs (Slave e Master), tais como TYPE_LAMP e TYPE_IR (o qual na prática não foi possível implementar), os tipos de instrução e os parâmetros do CAN.

Por não poder ver o código rodando em um microprocessador, tomou-se leds como auxílio, uma prática muito comum, para entender a posição em que o código está na hora da execução.

Para o Master, a função main() deve ser capaz de manter o PIC disponível a receber ou enviar dados via UART ou CAN Bus. A palavra recebida pode ter dois formatos, caso seja de setar parâmetros, será uma concatenação de SET + NodeID + StatusNumber + value. No caso de uma instrução de Request_Update é necessário a concatenação de REQ_UPDATE + END, entretanto, para facilitar o código de recebimento no Slave, as palavras enviadas têm o mesmo tamanho e deve-se enviar também mais dois bytes que não serão usados, dessa forma a palavra enviada em uma instrução de requerimento (REQ_UPDATE) também tem quatro bytes.

Para o nó Slave, além dos parâmetros comuns há parâmetros específicos do nó que definem sua identificação (MY_ID) e seu tipo (TYPE_LAMP). Sua função main() deve manter o microcontrolador disponível a receber instruções do Master, sejam de UPDATE ou de SET, e agir corretamente atuando nos dispositivos finais.

Para ambos os parâmetros CAN são usados na biblioteca própria do PIC, que facilitou bastante a implementação.

3. Inteligência do Sistema

A inteligência do sistema foi baseada em Aprendizado de Máquina. Aprendizado de Máquina é uma área de conhecimento da Inteligência Artificial. Ela estuda problemas nos quais um computador deve ser capaz de fazer a análise de dados e determinar padrões com a finalidade de gerar conhecimento de forma automática e, assim, aumentar o desempenho de determinadas tarefas.

Foram estudados algoritmos de Aprendizado de Máquina que pudessem ser utilizados pelo sistema na geração de regras que servissem de apoio às decisões tomadas pelo controlador central.

3.1. Algoritmo de Aprendizado de Máquina

É fundamental que o sistema não se limite apenas a automatizar processos. Ele também deve ser capaz de aprender com interações com os usuários da casa e tornar o meio adaptado às preferências dos usuários. Para que isso seja possível, será utilizado um algoritmo de Aprendizado de Máquina que será responsável por analisar dados e detectar padrões de comportamento. O algoritmo a ser trabalhado será o ID3, algoritmo de aprendizado supervisionado que utiliza árvores de decisão na construção de regras.

3.2. Árvore de Decisão

Existem várias ferramentas que permitem, a partir da análise de dados, o aprendizado de máquina. A árvore de decisões é uma dessas ferramentas. Árvores de decisões são utilizadas para definir respostas do sistema para determinado contexto (Pozzer, 2006). Podemos representar um contexto através de uma série de variáveis. A partir da raiz de uma Árvore de Decisão, fazem-se consecutivos testes e dependendo do resultado de cada teste percorre-se um caminho diferente na árvore. Cada nó da árvore representa um teste de um determinado atributo dentre aqueles que representam o contexto. Ao alcançar-se uma folha da árvore, tem-se o valor procurado para determinada variável. Cabe ressaltar que as variáveis representam características do ambiente e podem possuir valores contínuos e discretos.

3.3. Algoritmo ID3

O primeiro algoritmo de Aprendizado de Máquina estudado foi o ID3. Esse algoritmo é utilizado na construção de árvores de decisão (Quinlan, 1983). O algoritmo consiste em dividir um conjunto de dados divididos em um conjunto de variáveis. O primeiro passo é definir a métrica a ser utilizada no algoritmo. Introduz-se assim, o conceito de entropia. A entropia é o grau de homogeneidade dos exemplos de um determinado conjunto de dados[1].

O objetivo do algoritmo é priorizar as variáveis do conjunto de exemplos de acordo com ganho de informação de cada uma. Com isso, a árvore de decisão

[1] Definição extraída de notas de aula de Inteligência Computacional de Willian P. Amorim, professor da Universidade Federal da Grande Dourados.

montada é otimizada, sendo necessárias menos consultas até que se chegue em um nó da árvore (PENG; CHEN; ZHOU, 2013).

Seja um conjunto D de exemplos de treino com N variáveis, e seja p a probabilidade de ocorrência de determinada variável. A Entropia do conjunto de dados é dada por:

$$E(D) = -\sum_{i=1}^{N} P_i * \log_2 P_i \quad (1)$$

A priorização dos atributos é feita com base no ganho de informação. Esse ganho de informação é feito para cada variável V do conjunto D. Esse ganho é a redução de entropia resultante da escolha da variável V para dividir a árvore em duas sub-árvores. O ganho de informações pode ser dado pela expressão:

$$G(D,V) = E(D) - \sum_{i=1}^{N} \frac{|D_i|}{|D|} * E(D_i) \quad (2)$$

Onde |D| é a quantidade de elementos do conjunto D, e é o subconjunto de S tal que a variável V vale i.

O algoritmo ID3 é formalizado a seguir:

Função ID3 (R: conjunto não categórico, C: atributo categórico, S: conjunto de treino): árvore de decisão

Início
Se S está vazio retorna um nó com valor negativo (falha);
Se S consiste em registros com o mesmo valor para o atributo categórico, retorna um nó com esse valor;
Se R é vazio, então retorna um nó com o valor mais frequente do atributo categórico.
Seja D o atributo com o maior ganho de informação dentro de R;
Seja os valores dos atributos de D { dj | j = 1,2,...,m }
Seja { sj | j = 1,2,...,m} o subconjunto de S consistindo respectivamente de registros com valor dj para o atributo D;
Retorna uma árvore com raiz rotulada D e arcos rotulados d1, d2, ..., dm indo respectivamente para as árvores ID3(R-{D}, C, S1), ID3(R-{D}, C, S2), .., ID3(R-{D}, C, Sm);
Fim.

3.4. Software de Aprendizado de Máquina

Primeiramente será mostrado o que se espera por parte do software para que ele possa realizar o Aprendizado de Máquina e, em seguida, será justificada a framework escolhida com base nesses requisitos. Após isso será apresentado um exemplo passo-a-passo de como se deve utilizá-la para o Aprendizado de Máquina.

3.5. Funcionalidades esperadas do Software de Aprendizado

A parte do software responsável pelo Aprendizado de Máquina deve ser capaz de receber um conjunto de entradas X e um conjunto Y das respectivas observações para essas entradas para que se possa gerar uma árvore de decisão. Tal árvore deve ser capaz de deduzir o valor de uma observação dado que se sabe o estado Xi no qual o sistema se encontra. Espera-se também que o objeto que representa uma árvore de decisão possa ser persistido em algum banco ou sistema de arquivos bem como seja visualizado.

Por exemplo, na situação da figura mostrada abaixo, temos a simulação do aprendizado para um exemplo de decisão para um dispositivo de luz. Nas figuras 2 e 3, X é o conjunto das tuplas de com atributos “hora” e “usuário” e Y é o conjunto dos valores observados para a luz. Para valores de luz iguais a 0 temos que a luz está desligada e nos valores iguais a 1 temos que a luz está ligada. Pode-se ver na figura 2 que X = [[6, 1], [12, 0], [18, 1], [23, 1], [6, 2], [12, 2] , [18, 2]] e Y = [1,0, 1, 1, 0, 0, 1]. A partir de X e Y é construída a ávore de decisão e ela deve ser capaz de deduzir, para uma entrada E = [a, b], sendo *a* e *b* valores quaisquer para hora e usuário respectivamente, qual a observação Y.

Estados		Observações
X		Y
hora	usuário	luz
6	1	1
12	0	0
18	1	1
23	1	1
6	2	0
12	2	0
18	2	1

Figura 2: Exemplo de estados e observações.

Apresenta-se outro exemplo na figura 3 no qual tomam-se as mesmas tuplas da figura 2 e adicionam-se três novas tuplas com valores que irão interferir no aprendizado. Com esses novos valores espera-se que a árvore de decisão gerada por esse novo exemplo seja diferente:

Estados		Observações
X		Y
hora	usuário	luz
6	1	1
12	0	0
18	1	1
23	1	1
6	2	0
12	2	0
18	2	1
18	1	0
18	1	0
18	1	0

Figura 3: Exemplo de estados e observações.

3.6. Scikit-learn

A framework scikit-learn, disponível em *scikit-learn.org*, foi escolhida por ser open-source, fácil de usar, bem documentada e ser a mais utilizada em trabalhos de Aprendizado de Máquina em python. Ela fornece implementações de diversos problemas de aprendizado supervisionado e não-supervisionado.

Para a persistência dos objetos do Aprendizado de Máquina fez-se o uso da framework serializadora Pickle para que se possa salvar os objetos em um banco ou em um sistema de arquivos. Além disso, para visualização, a framework scikit-learn fornece a transformação dos objetos do aprendizado em arquivos pdf.

Em relação ao seu desempenho, pode-se dizer que a framework scikit-learn é superior ao de uma framework escrita puramente em python, pois ela faz internamente chamadas a outras frameworks compiladas em c (NumPy e SciPy) para poder realizar manipulação de vetores e matrizes.

3.7. Exemplo de Aplicação do scikit-learn

Nesta seção será mostrado um exemplo de aprendizado utilizando os valores mostrados anteriormente na figura 2:

i) Preparando o conjunto de treino: inicialmente preparam-se os valores das entradas X, e das observações Y:

```
>>> X = [ [6, 1], [12, 0], [18, 1], [23, 1], [6, 2], [12, 2], [18, 2]]
>>> Y = [ 1, 0, 1, 1, 0, 0, 1].
```

ii) Realizando o aprendizado: instancia-se o construtor de árvore de decisão e insere-se X e Y:

```
>>> from sklearn import tree
>>> arvoreDec = tree.DecisionTreeRegressor(criterion=”entropy”)
>>> arvoreDec = arvoreDec.fit(X, Y)
```

iii) Utilizando o aprendizado: nesta etapa já se tem a árvore gerada no objeto *clf*. Para se realizar uma dedução de um valor a ser observado faz-se:

```
>>> resultado = arvoreDec.predict([[23, 1]])
>>> print resultado
array([1])
```

Ao utilizar o *predict*, obtivemos um *array* que contém os valores deduzidos. Nesse caso, o valor foi 1, o que significa que a observação deduzida é igual a 1.

iv) Persistindo a árvore de decisão: persiste-se o objeto serializado pelo método *dump* do Pickle:

```
>>> import pickle
>>> nomeDoArquivo = ‘arvore_X’ + ‘.plk’
>>> with open(nomeDoArquivo, ‘wb’) as f:
… pickle.dump(arvoreDec, f)
```

v) Recuperando uma árvore de decisão: recuperar o arquivo do objeto serializado pelo método *load* do Pickle:

```
>>> arvoreDec = {}
>>> with open(nomeDoArquivo, ‘rb’) as f:
… arvoreDec = pickle.load(f)
… f.close()
>>> print arvoreDec.predict([23, 1])
array([1])
```

vi) Exportando os resultados para pdf: Para salvar um arquivo pdf com a árvore de decisão converte-se o objeto para um arquivo .dot (formato utilizado para gravar objetos que representam grafos). Após a conversão altera-se o formato de .dot para .pdf:

```
>>> nomeDoArquivoEmDot = 'arvore_X' + '.dot'
>>> with open(ilename,'w') as f:
… f = tree.export_graphviz(arvoreDec, out_file = f, feature_names =
['hora','usuario'])
        >>> import os
        >>> os.system("dot -Tpdf arvore_X.dot -o arvore_X.pdf")
        >>> os.unlink(nomeDoArquivoEmDot)
```

A seguir encontram-se os arquivos em formato pdf gerados para os exemplos mostrados acima. Nele, os nós internos da árvore de decisão testam as variáveis de treino ("hora" e "usuário"). Estes nós estão conectados aos debaixo com setas para especificar as possíveis saídas dos testes. Como se trata de um problema de classificação, os nós folha da árvore representam as possíveis classes. Nas amostras de treino utilizadas, os valores de observação eram somente iguais a 0 ou 1. Vale notar que os valores observados para as possíveis classes são 0 ou 1 e que a framework apresenta os resultados de "value" de forma ordenada para o valor das classes. Dessa forma, a framework scikit-learn representa o resultado da seguinte maneira: se value = [0, 3] significa que houve nenhuma ocorrência para a classe de índice 0 e três para a de índice 1. Portanto, para a folha que contém value = [0, 3] o valor da observação a se deduzir é 1, ou seja, para os casos onde a hora é maior que 15 a luz deve estar ligada (no estado 1).

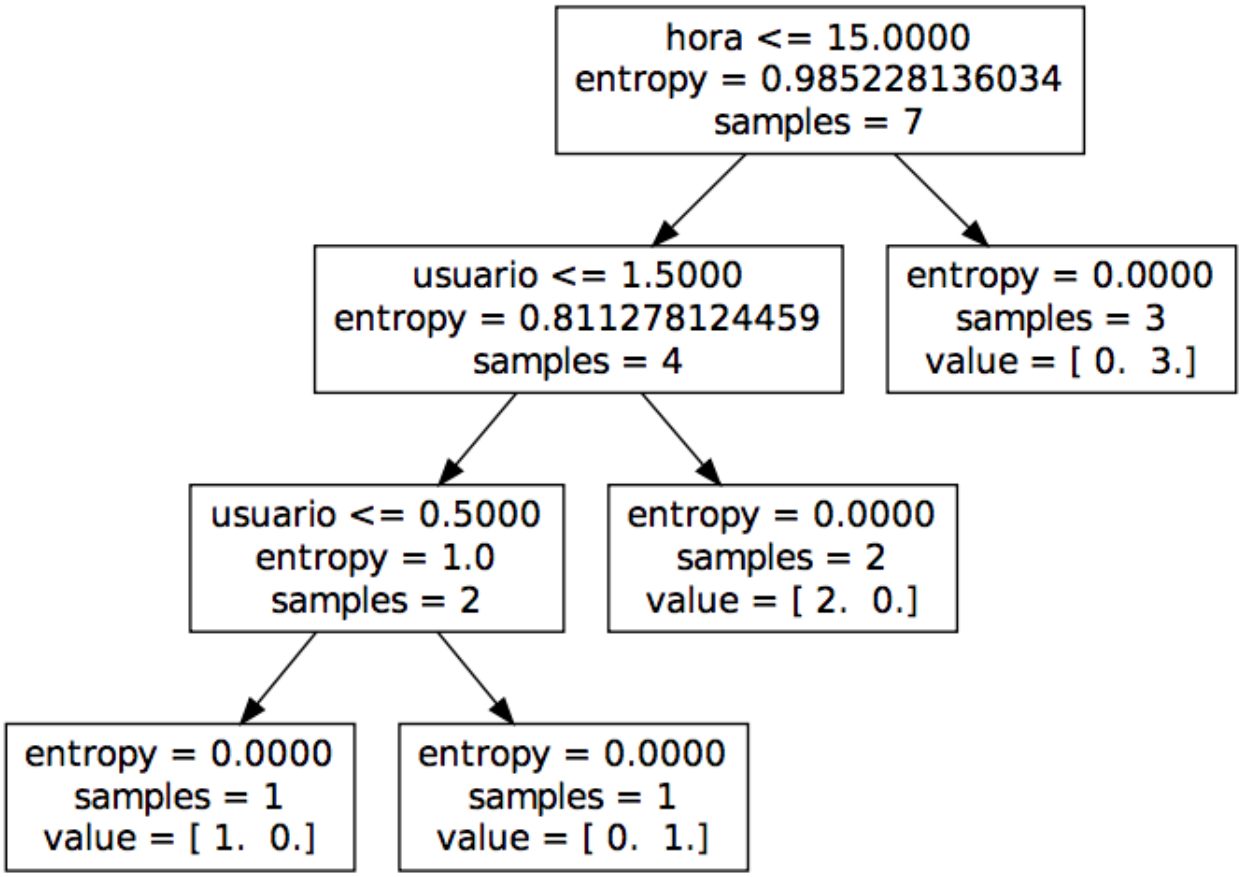

Figura 4: Árvore de decisão gerada para o exemplo genérico apresentado.

Uma vez que novos eventos ocorrem, a árvore de decisão deve ser capaz de sofrer alterações de acordo com esses novos eventos. O exemplo da figura 5 confirma que, caso seja adicionado um novo comportamento, que no caso foi de a luz estar apagada quando se está no estado cuja hora e usuário são 18 e 1 respectivamente. A seguir encontra-se a árvore gerada para o segundo exemplo.

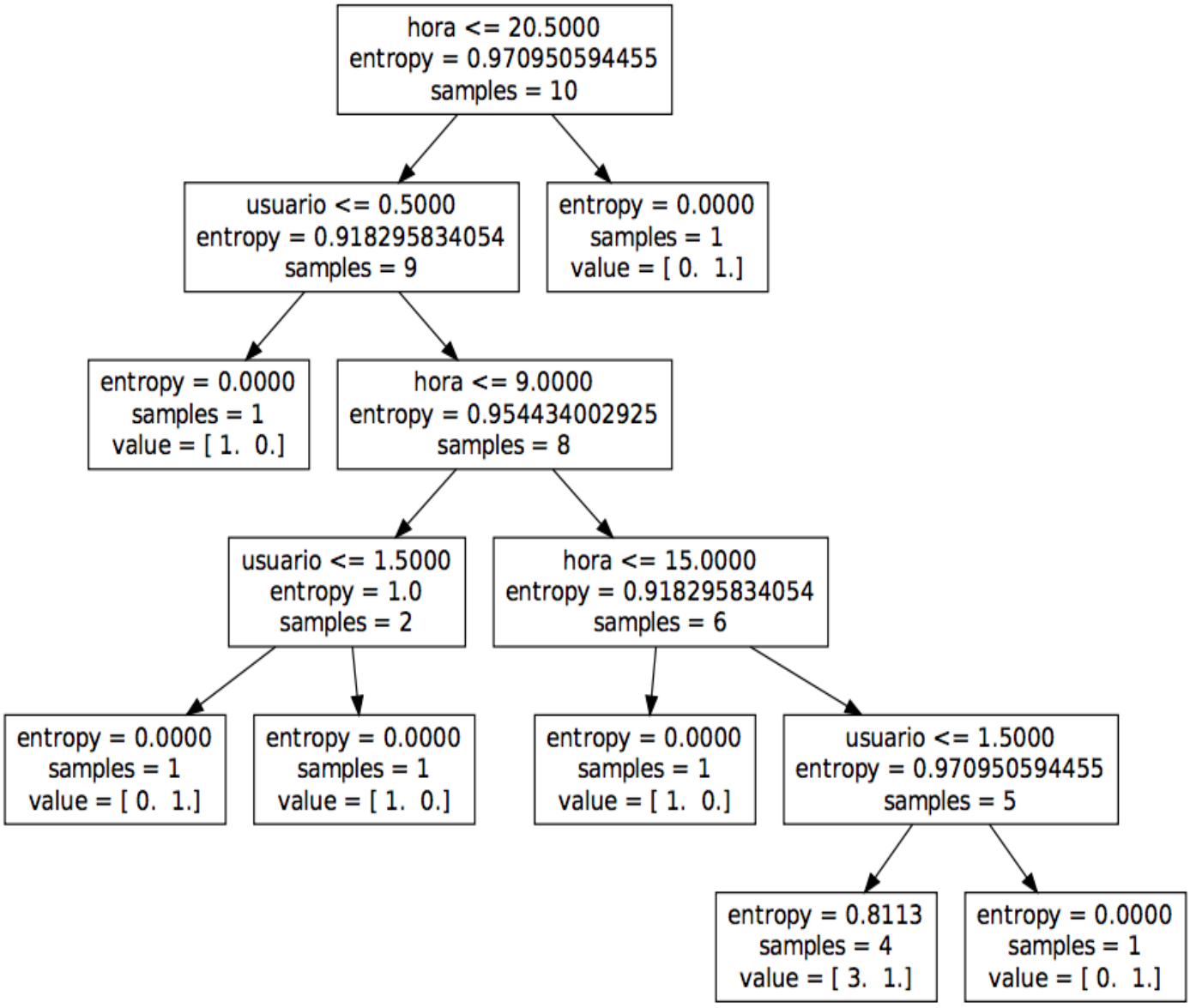

Figura 5: Árvore de decisão gerada para o exemplo genérico apresentado.

4. Proposta

4.1. Trabalhos anteriores sobre Domótica

Neste tópico são descritos os sistemas domóticos estudados anteriormente pelos membros desta equipe(PAIVA; SILVEIRA, 2013). A seguir estão descritas as ferramentas utilizadas por tais sistemas bem como suas limitações e vantagens. Tal abordagem torna-se útil para se ter uma base de como sistemas anteriores foram implementados e por fim para a especificação dos requisitos do sistema proposto neste trabalho.

i) Sistema ABC (Tonidandel; Takiuchi; Melo, 2004): no sistema de Arquitetura Baseada em Comportamento (sistema ABC) o aprendizado é supervisionado e dá-se construção de árvores de decisão com o algoritmo ID3 (QUINLAN, 1986) com o critério de entropia. As árvores de decisão são eliminadas após um certo tempo e, basicamente o sistema prevê o que a casa deve fazer para cada situação. Uma limitação desse sistema é o fato de que ele não é capaz de prever ações em sequência uma vez que a utilização de árvores de decisão fornece apenas como a casa deve estar para uma dada situação.

ii) Sistema ABC+ (Sgarbi, 2007): utiliza o algoritmo C4.5 para a construção das árvores de decisão. Toma como base o Sistema ABC e procura eliminar suas deficiências através da tentativa de prever eventos contínuos e de um sistema de pontuação de regras. Em relação às regras, são estabelecidos dois estados (regras ativas ou regras embrionárias) para que elas possam ser avaliadas pelo sistema de pontuação e enfim serem utilizadas pela casa ou descartadas.

iii) Sistema ACHE (Mozer, 1998): utiliza cadeias de markov e árvores de decisão para a tomada de decisão. A proposta deste sistema tem como fazer a tomada de decisão levando-se em conta dois fatores: conforto do usário versus economia de energia e gás. Nele, é proposta uma equação que procura determinar, dentre os

estados possíveis para cada evento, a menor soma de desconforto e custo energético. Também são introduzidos pesos de desconfortos conforme a aceitação do usuário.

iv) Sistema ADR-SPLDA (Chikhaouia; Pigot, 2012): utiliza aprendizado não-supervisionado a partir de Cadeias de Markov e Modelos Ocultos de Markov e Alocação Latente de Dirichlet. Foca em registrar as tarefas de um usuário em forma de sequência para tentar descobrir se existem blocos de ações que se repetem. As ações repetidas são tratadas como estados durante a formação de Cadeias de Markov. Uma vez conhecida as sequências de tarefas dos usuários, o sistema é capaz de avaliar os registros de eventos e verificar se o usuário esqueceu de fazer alguma tarefa importante. Com isso, o sistema torna-se útil para pessoas com deficiência em memória e também pode prevenir acidentes domésticos causados por falha humana.

4.2. Especificações do Sistema

O sistema a ser desenvolvido deve possuir as determinadas características:

- O sistema deve ser escalável, ou seja, deve ser capaz de se adaptar à novas funcionalidades e inserção de periféricos;
- O sistema deve ser o mais independente o possível de tecnologias, tornando mais fácil a manutenção futura, seja essa preventiva ou corretiva;
- O sistema deve possuir bom desempenho, de modo que o tempo de resposta a eventos deve ser o mínimo o possível, e assim tornar mais cômoda a experiência do usuário;
- O sistema deve ser seguro, ou seja, deve ter alguma forma de identificar e autenticar o usuário da casa;
- O sistema deve ser o mais automatizado o possível, de modo que o usuário necessite interagir o mínimo possível com a casa.

Em relação à interação com o usuário residente, o sistema deve possuir as seguintes características:

- O sistema deve ser capaz de controlar algum dispositivo da casa. Esse dispositivo pode ser responsável, por exemplo, pela luminosidade e/ou pela temperatura de determinado cômodo;
- O sistema deve ser capaz de aprender as preferências do usuário conforme ele utiliza a casa;
- O sistema deve partir de um padrão de preferências para novos usuários;
- O sistema deve priorizar as preferências do usuário que está a mais tempo no cômodo nos casos em que há mais de um usuário no mesmo cômodo;
- O sistema deve ser capaz de identificar quem é o usuário que está interagindo com a casa;
- O sistema deve ser capaz de detectar qual é o cômodo em que o usuário da casa se encontra.

5. Modelagem do Problema

5.1. Fluxo de funcionamento da casa

Quando um morador entra em um cômodo da casa inteligente, ele deve usar o celular com acesso à Internet para acessar uma página Web na qual ele irá informar à casa que ele se encontra em determinado cômodo. A casa irá procurar no banco de dados se há alguma preferência daquele morador para aquele horário naquele cômodo. Caso haja, a casa deve automaticamente alterar seu estado visando se adequar às preferências do usuário.

O sistema deve verificar, periodicamente, o estado de cada cômodo da casa. Se o sistema verificar que o cômodo alterou seu estado em relação à última verificação que foi realizada, esse novo estado da casa deve ser salvo em um banco de dados. As informações do banco de dados serão utilizadas na aprendizagem das preferências dos usuários. Esta verificação não é necessária quando não houver nenhum morador em casa.

5.2. Arquitetura do sistema

Será utilizado um webserver no qual estão presentes todos os módulos de lógica para o aprendizado adaptativo da casa. Neste webserver, o algoritmo de tomada de decisão pela casa é acionado de duas maneiras: uma delas ocorre quando o usuário muda de cômodo e a outra ocorre a cada 30 minutos para ver se a casa deve mudar de estado. Ainda há uma terceira opção, que é acionar a tomada de decisão assim que o estado de qualquer dispositivo mude, porém, nesse caso, uma simples mudança

como o usuário ligar e desligar uma lâmpada iria acionar o algoritmo de tomada de decisão e o tratamento do aprendizado torna-se mais complexo.

A arquitetura da aplicação foi feita para acionar a tomada de decisão somente nos dois primeiros casos citados acima: a primeira é quando o usuário muda de cômodo e a segunda é verificar a cada 30 minutos o estado da casa e as posições dos usuários para verificar se o algoritmo decisor deve fazer algo. Ainda nessa rotina de 30 minutos, salvam-se as posições dos usuários e o estado da casa no banco de dados. Dessa forma, garante-se com que sejam salvas essas informações pelo menos a cada 30 minutos. Para gerar o aprendizado, a cada 12 horas é disparada uma rotina que busca todos os eventos e as posições dos usuários no banco para se sejam criadas as regras de decisão.

5.3. Casos de uso

Com base no fluxo de funcionamento da casa, foi elaborada uma especificação do funcionamento da casa com objetivo de identificar entidades do sistema. A seguir encontra-se o diagrama de casos de uso obtido:

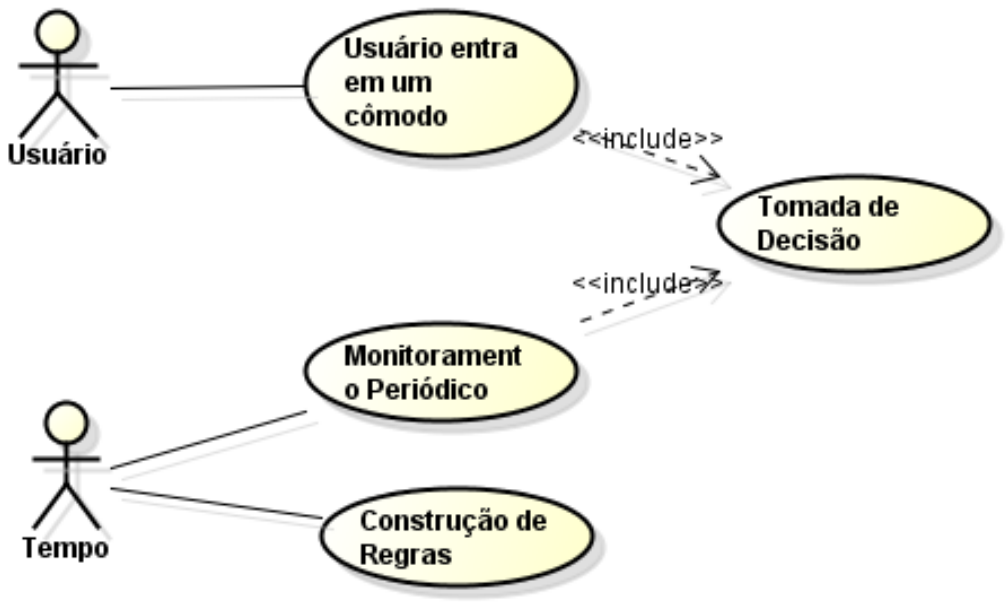

Figura 6: Diagrama dos Casos de Uso

Os casos de uso estão no apêndice 1, com excessão do caso de uso de construção de regras que está a seguir:

Tabela 1: Caso de uso de criação de regras

Título	Construção de Regras
Ator	Tempo
Pré-Condição	Não Há
Fluxo Principal	1. A cada 12 horas o agendador de rotinas deve requisitar a rotina de construção de regras do gerenciador de regras. 2. O gerenciador de regras requisita ao serviço de acesso ao banco de dados o histórico de todos os estados dos cômodos. 3. O gerenciador de regras cria as novas regras utilizando o histórico de estados dos cômodos da casa. 4. O gerenciador de regras requisita ao serviço de banco de dados que salve as regras obtidas
Fluxo Secundário	Não há
Pós-Condição	O sistema criou e salvou as novas regras baseado no histórico de estados da casa.
Regras de Negócio	RN 01: O estado de um cômodo da casa é constituido por: cômodo, usuários que se encontram no cômodo, hora da gravação, temperatura do ar condicionado, se a lâmpada está ligada ou desligada e se a cortina está aberta ou fechada. Caso o ar condicionado esteja desligado, pode ser atribuido um valor padrão. RN 02: Uma regra é constituida por usuário, cômodo, hora e se é dia de semana ou fim de semana. A partir disso, é possível determinar a temperatura do ar e se a luz deve ser ligada ou desligada.

5.4. Diagramas de Iterações

Para a confecção dos diagramas de iterações, foi utilizada a versão gratuita de um software de modelagem de software, o Astah. A imagem a seguir é referente ao diagrama de iterações obtido a partir do caso de uso de Construção de Regras. Os demais diagramas encontram-se no Apêndice 2.

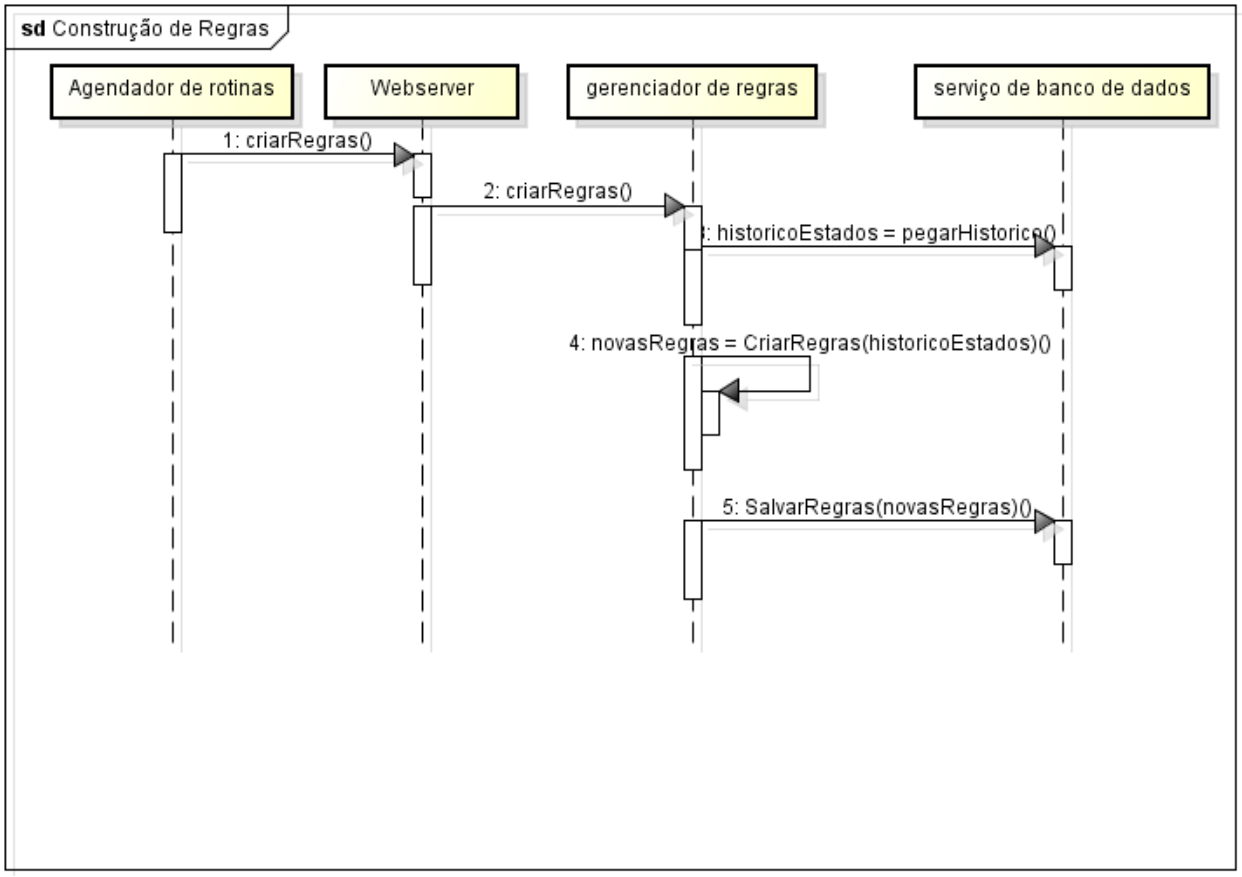

Figura 7: Diagrama de sequência de construção de regras

6. Desenvolvimento do Sistema

6.1. Entidades do software

Nesta parte são mostradas as entidades que a aplicação utilizará durante o seu funcionamento. Inicialmente serão apresentados os modelos responsáveis por carregar as informações no software e então os módulos que utilizam esses modelos bem como suas responsabilidades.

6.1.1. Modelo de Domínio

i) *RoomState*: armazena o estado de um cômodo da casa. Nele estão presentes os seguintes atributos:

- *room*: indica qual é o cômodo representado. Assume valor 0 caso seja o cômodo 1 e valor 1 caso seja o cômodo 2 e assim por diante.

- *hour*: indica qual é a hora em que o estado foi lido. Assume valores determinados no conjunto 0, 0.5, 1, … , 23 e 23.5. Os valores são sempre arredondados para a menor fração.

- *users*: indica quais são os usuários presentes no cômodo. Assume valor 0 caso seja o usuário 0, valor 1 caso seja o usuário 1 e assim por diante.

- *light*: indica se a luz do cômodo está acesa ou apagada. Assume valor 0 caso esteja acesa e valor 1 caso esteja apagada.

- *temperature*: indica a temperatura em que o cômodo está. Assume valores inteiros que variam de 18 até 28.

- *curtain*: indica se a cortina do cômodo está fechada ou aberta. Assume valor 0 caso esteja fechada e valor 1 caso esteja aberta.

ii) *RoomStateRule*: armazena como deverá estar o estado de um cômodo na casa. Os valores deduzidos através do aprendizado de máquina são postos nesta classe.

- *room*: indica qual é o cômodo em que aquela regra é válida. Assume valor 0 caso seja o cômodo 1 e valor 1 caso seja o cômodo 2 e assim por diante.

- *hour*: indica qual é a hora em que aquela regra é válida. Assume valores determinados no conjunto 0, 0.5, 1, … , 23 e 23.5. Os valores são sempre arredondados para a menor fração.

- *user*: indica qual é o usuário a qual a regra se refere. É importante destacar que dois usuários podem ter regras distintas para o mesmo cômodo e horário e o tratamento desse caso será exposto mais adiante. Assume valor 0 caso seja o usuário 0, valor 1 caso seja o usuário 1 e assim por diante.

- *light*: indica se a luz do cômodo deve estar acesa ou apagada. Assume valor 0 caso esteja acesa e valor 1 caso esteja apagada.

- *temperature*: indica a temperatura em que o cômodo deve estar. Assume valores inteiros que variam de 18 até 28.

- *curtain*: indica se a cortina do cômodo deve estar fechada ou aberta. Assume valor 0 caso esteja fechada e valor 1 caso esteja aberta.

6.1.2. Módulos do Sistema

O sistema foi dividido em módulos de forma a separar responsabilidades, diminuir o acoplamento e permitir uma maior manutenibilidade do código. Os módulos são:

- *Webcontroller*: é onde chegam as requisições ao webserver. Provê todos os métodos para permitir a interação de usuários e o do sistema operacional com o *webserver*. Devido a características da framework de desenvolvimento web Flask, está localizado no mesmo arquivo de iniciação do sistema.
- *Database Service*: oferece serviços de acesso ao banco de dados, como salvar um modelo.
- *Decision Service*: módulo responsável pela tomada de decisão. Decide se a casa vai tomar alguma ação. Se sim, ele aciona os serviços necessários para atualização do estado da casa.
- *Devices Control*: módulo responsável pela comunicação do webserver com a rede de dispositivos locais. Tanto emite ordens para os atuadores agirem na casa quanto pede o estado da casa à rede de dispositivos.
- *House State Manager*: módulo responsável por gerenciar o estado atual de cada cômodo. Também é responsável pelas regras de negócio dos modelos de domínio.
- *House State Rules Manager*: módulo responsável por gerenciar as regras da casa. É responsável pelo algoritmo de criação de regras e manipulação do histórico de estados salvos.

6.2. Implementação do Software

Uma vez modeladas e identificadas todas as entidades e responsabilidades, foi implementada a parte do sistema referente ao *webserver*. Foi implementada, também, uma interface que permite usuários da casa controlarem dispositivos e informarem caso entrem ou saiam de algum cômodo. Foi implementada uma interface para apenas um cômodo. Entretanto, é fácil escalar para mais cômodos, uma vez que todos possuem as mesmas características. A Figura 4 é referente à interface de interação do usuário com o sistema:

Figura 8: Interface de interação do usuário com o sistema

6.3. Implementação do Hardware

Foi implementada a rede para interligar os dispositivos e que permitisse o tráfego bidirecional de mensagens na rede, tanto para emitir ordens para os dispositivos quanto para conseguir o estado dos dispositivos. A imagem a seguir é referente ao esquema da rede implementada.

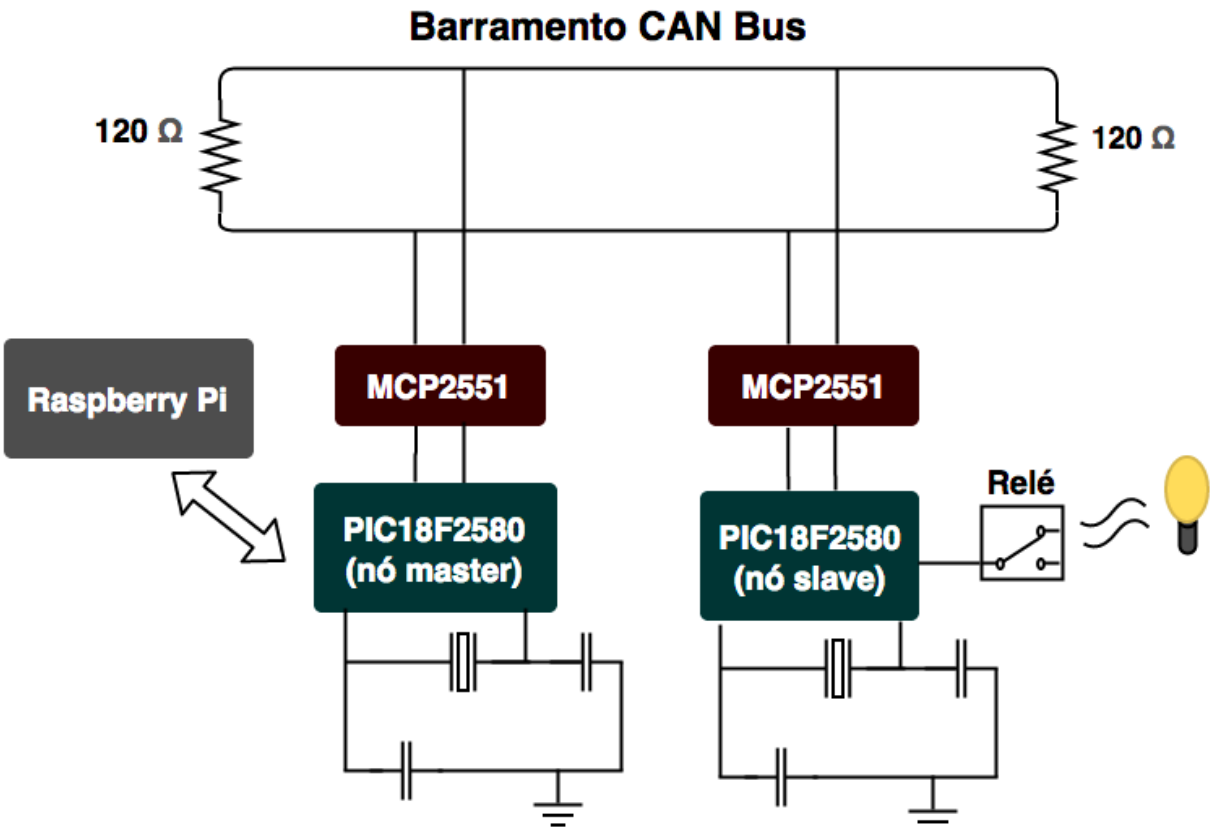

Figura 9: Representação da implementação da rede.

6.4. Desafios encontrados e soluções adotadas

Durante a execução da parte prática deste trabalho, a equipe enfrentou dificuldades que variam desde problemas de instalação de bibliotecas até problemas com a utilização do material. A seguir estão documentados os desafios e também as soluções adotadas ou sugeridas pela equipe:

i) Migração de código entre microcontroladores.

O problema ocorreu ao tentar passar o algoritmo responsável pelo controle infravermelho que foi utilizado anteriormente no trabalho de Iniciação à Pesquisa disponível em (MARLIÈRE et al., 2013). Tal algoritmo estava implementado para um microcontrolador Atmel AVR e, quando implementado no PIC 18F2580, não foi possível emitir os pulsos com precisão, o que resultou em uma não comunicação em infravermelho.

Apesar dos esforços, não foi possível implementar um segundo nó Slave, que seria responsável por atuar no ar condicionado.

ii) Permitir o webserver ser acessível na rede

Uma vez com a aplicação sendo executada na rede da Raspberry Pi, deveria ser possível fazer com que ela fosse acessível por outros dispositivos conectados na mesma rede. A solução encontrada foi colocar o webserver para executar no endereço IP 0.0.0.0, que faz a aplicação ser executada em algum endereço de IP válido na rede.

iii) Obtenção de dispositivos

Esse desafio ocorreu devido à dificuldade de se encontrar no comércio local os dispositivos corretos para rede CAN Bus. A obtenção de nós slaves de uma rede CAN Bus teve que ser feita através de compras no exterior e pelo fato de que a equipe não era experiente em tais dispositivos, não havia a certeza de que eles seriam adequados ao nosso problema.

Ao final foram utilizados dois nós CAN Bus, um para uso como master e um como slave. Com isso, o controle de dispositivos da casa restringiu-se a apenas um dispositivo periférico (lâmpada ligada ao nó slave).

iv) Utilização da comunicação da porta serial na Raspberry Pi

O primeiro problema encontrado foi com a habilitação dos pinos de comunicação UART na Raspberry Pi. Por padrão, eles estão habilitados para uso do sistema operacional para fins de debug. Foi necessário configura-los para uso externo a partir da leitura de uma discussão online (ROBOTWORKSHOP).

O segundo problema foi saber se a comunicação estava sendo bem sucedida. Para solucioná-lo foi escrito um script em python que abria as portas, enviava e recebia as mensagens trocadas.

7. Conclusão

A automação residencial surgiu com a finalidade de prover mais conforto e segurança para as pessoas, contribuindo assim para melhoria na qualidade de vida. Nesse contexto, a domótica surgiu como um avanço ainda maior: a introdução de inteligência artificial na automação residencial.

Este trabalho teve por objetivo a construção de um sistema domótico a com adição de recursos de inteligência artificial. Para isso, o sistema proposto teve que ser construído com a capacidade de se adaptar a mudança de comportamento de seus usuários. Foi necessário o estudo de sistemas domóticos, das ferramentas utilizadas para o aprendizado de máquina e das formas de comunicação entre os dispositivos periféricos e a aplicação responsável pelo aprendizado.

Para alcançar os objetivos, começou-se pelo processo de automação e controle de dispositivos da casa. Em seguida foi implementada uma interface gráfica para o usuário. Tal interface comunicava-se com um servidor web que processava as requisições e comunicava-se com a rede de dispositivos periféricos. Em seguida foi implementada a inteligência do sistema, cuja finalidade era prever as ações do usuário na casa.

Ainda cabem no sistema desenvolvido diversas melhorias. Podemos citar a modificação do layout da interface com objetivo de melhorar a experiência do usuário. O sistema também pode verificar automaticamente que determinado usuário entrou em um cômodo, tirando essa responsabilidade do usuário da aplicação. Em relação à forma do aprendizado, ainda pode ser adicionado um sistema de pontuação de regras aos moldes dos estudados nos outros trabalhos vistos nesta pesquisa. Outro trabalho futuro pode ser a mudança no modo como as regras são geradas, pois também poderia ter-se utilizado árvores de regressão para deduzir o comportamento do usuário.

8. Referências Bibliográficas

BOSCH. **CAN SPECIFICATION 2.0.** 1991. Disponível em: <http://www.bosch-semiconductors.de/media/pdf_1/canliteratur/can2spec.pdf>. Acesso em: 01 nov. 2014

CATSOULIS, J. Designing: Embedded Hardware. O'Reilly, 2005. GUEDES, G. T. A. UML 2 - Uma Abordagem Prática. NOVATEC, 2011. IBRAHIM, D. Controller Area Network. Elektor, 2011.

CONDUTA, Bruno Custódio; MAGRIN, Diego Henrique. APRENDIZAGEM DE MÁQUINA. Universidade Estadual de Campinas – Unicamp, Faculdade de Tecnologia - Ft, Limeira – Sp,2010. Disponível em: <http://www.ft.unicamp.br/liag/wp/monografias/monografias/2010_IA_FT_UNICAMP_aprendizagemMaquina.pdf>. Acesso em: 14 fev. 2014.

D-robotics uk - dht11 humidity and temperature sensor, Mai 2014. Disponível em: <http://www.micro4you.com/files/sensor/DHT11.pdf>. Acesso em: 20 agosto 2014.

FUSSEK, R. A. J. DESENVOLVIMENTO DE UM CLASSIFICADOR AUTOMÁTICO DE PESSOAS PARA SISTEMAS DE AUTOMAÇÃO RESIDENCIAL INTELIGENTE, 2010. Dissertação (Mestrado em Engenharia Elétrica) - Centro Universitário da FEI, São Bernardo do Campo, 2010.

GUIMARÃES, Alexandre de A.. **CAN BUS:** Barramento Controller Area Network. Disponível em: <http://www.pcs.usp.br/~laa/Grupos/EEM/CAN_Bus_Parte_2.html>. Acesso em: 29 out. 2014.

HALMENSCHLAGER, Carine. Um algoritmo para indução de árvores e regras de decisão. 2002. 109 f. Dissertação (Mestrado) - Curso de Computação, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2002. Disponível em: <http://www.lume.ufrgs.br/bitstream/handle/10183/2755/000325797.pdf?sequence=1>. Acesso em: 18 maio 2014.

LINS, V.; MOURA, W. Domótica: Automação Residencial UNIBRATEC , Recife, 2010 , Disponível em: <http://www.unibratec.edu.br/tecnologus/wp-content/uploads/2010/12/lins_moura.pdf>. Acesso em 30 set 2013.

MARLIÈRE, F. M. C.; FARIAS, J. C. L. A. A casa inteligente: o desenvolvimento de um sistema de controle de temperatura. Monografia (Iniciação a Pesquisa) Rio de Janeiro: Instituto Militar de Engenharia, 2014.

MATTOS, Wajeria. O Caminho Para a Automação. Lumière eletric. São Paulo, Edição 170, p. 20-23, junho. 2012.

MITCHELL, T. M. Machine Learning. Boston: McGraw-Hill, 1997.

MONARD, M. C.; BARANAUSKAS, J.A. Conceitos sobre aprendizado de máquina. In: REZENDE, S. O. (Ed.). Sistemas Inteligentes: fundamentos e aplicações. São Carlos: Manole, 2003. p. 89-114. cap. 4.

NILSSON, N. PRINCIPLES OF ARTIFICIAL INTELLIGENCE. Palo Alto, California, 1980.

OSÓRIO, F. S. Tutorial: Redes Neurais - Aprendizado Artificial, 1999. In: SGARBI, J.A. Domótica inteligente: automação residencial baseada em comportamento. 2007. Dissertação (Mestrado em Engenharia Elétrica) – Centro Universitário da FEI, São Bernardo do Campo.

PAIVA, C. H. P.; SILVEIRA, R. S. D. Estado da Arte da Casa Inteligente. Monografia (Iniciação a Pesquisa). Rio de Janeiro: Instituto Militar de Engenharia, 2014.

PEDREGOSA (Ed.). **Scikit-learn: Machine Learning in Python.** 2011. JMLR 12, pp. 2825-2830. Disponível em: <http://scikit-learn.org/stable/>. Acesso em: 10 maio 2015.

PENG, Wei; CHEN, Juhua; ZHOU, Haiping. An Implementation of ID3 --- Decision Tree Learning Algorithm. Sydney. Disponível em: <http://cis.k.hosei.ac.jp/~rhuang/Miccl/AI-2/L10-src/DecisionTree2.pdf>. Acesso em: 10 fev. 2014.

POZZER, C. T. Aprendizado por Árvores de Decisão. Universidade Federal de Santa Maria, Departamento de Eletrônica e Computação, 2006.

QUINLAN, J. R. Induction of decision trees. Machine Learning Vol 1. Sydney: University of Sydney, 1986. 1 v.

QUINLAN, J. R. C4.5: Programs for Machine Learning. San Mateo, California, EE.UU, Morgan Kauffmann Publishers. 1993.

Simple and intuitive web interface for your raspberry, Mai 2014. Tutorial (Fórum Online). Disponível em: <http://www.instructables.com/id/Simple-and-intuitive-web-interface-for-your-Raspbe/?ALLSTEPS>. Acesso em: 18 agosto 2014.

SGARBI, J.A. Domótica inteligente: automação residencial baseada em comportamento. 2007. Dissertação (Mestrado em Engenharia Elétrica) - Centro Universitário da FEI, São Bernardo do Campo, 2007

TONIDANDEL, F; TAKIUCHI, M.; MELO, E. Domótica Inteligente: Automação baseada em comportamento. CONGRESSO BRASILEIRA DE AUTOMÁTICA, 2004.

APÊNDICE 1 – Casos de Uso do Sistema

Tabela 2: Caso de uso de usuário entrando em um cômodo

Título	O usuário entra em um cômodo
Ator	Usuário
Fluxo Principal	1. O usuário da casa interage com uma interface de usuário e informa que entrou em determinado cômodo da casa; 2. A interface de usuário informa ao gerenciador do estado da casa que determinado usuário entrou em determinado cômodo; 3. O gerenciador de estado da casa salva o estado atual do cômodo; 4. O gerenciador de estado da casa aciona o serviço de tomada de decisão. 5. Incluir Tomada de Decisão pela casa.
Fluxo Secundário	Em 3, já existe alguém no cômodo quando o usuário entra no cômodo. 1. O sistema não faz nada se já existe alguém no cômodo quando o usuário entra no cômodo.
Pós-Condição	O sistema atualizou o estado atual do comodo e acionou o serviço de tomada de decisão.

Tabela 3: Caso de uso do monitoramento da casa

Título	Monitoramento periódico da casa
Ator	Tempo
Fluxo Principal	1. A cada 30 minutos, o agendador de rotinas aciona uma rotina do gerenciador de estado da casa. 2. O gerenciador de estado da casa requisita ao serviço de acesso ao banco de dados que salve o estado corrente da casa. 3. O gerenciador de estado da casa aciona o serviço de tomada de decisão. 4. Incluir Tomada de Decisão pela casa
Fluxo Secundário	Não Há.
Pós-Condição	Foi salvo no banco o estado atual de todos os cômodos da casa e foi acionado o serviço de tomada de decisão.
Requisito Não Funcional	A hora salva no banco deve obedecer o seguinte formato: caso os minutos sejam menores que 30, então os minutos são ignorados; caso contrario, será considerada a hora atual mais meia hora. Exemplos: 19 horas, 19,5 horas, 20 horas. A finalidade é discretizar o tempo em apenas alguns determinados possíveis valores e acelerar o aprendizado do sistema.

Tabela 4: Caso de uso de tomada de decisão

Título	Tomada de decisão pela casa
Pré-Condição	Foi acionado o serviço de tomada de decisão
Fluxo Principal	1. O serviço de tomada de decisão requisita ao gerenciador de regras as regras existentes; 2. O serviço de tomada de decisão requisita ao gerenciador de estado da casa o estado atual dos cômodos da casa. 3. O serviço de tomada de decisão verifica qual a regra dentre as regras existentes está relacionada com o cômodo, o usuário e a hora. Ou seja, quais os valores esperados para lâmpada, cortina e ar condicionado dados o usuário, o cômodo e a hora. 4. O serviço de tomada de decisão compara o estado esperado com o estado atual. 5. O serviço de tomada de decisão requisita ao atuador da casa que altere o estado atual da casa. 6. O serviço de tomada de decisão requisita que o gerenciador de estado da casa atualize o estado atual.
Fluxo Secundário	Fluxo Secundário 1: no passo 4, o estado atual é igual ao estado esperado. 1. O serviço de tomada de decisão não continua o fluxo principal e o caso de uso termina com sucesso. Fluxo Secundário 2: no passo 3, o serviço de tomada de decisão não encontra uma regra que se aplique a determinado cômodo, usuário e hora. 1. O serviço de tomada de decisão não continua o fluxo principal e o caso de uso termina com sucesso.

APÊNDICE 2 – Diagramas de iterações do sistema

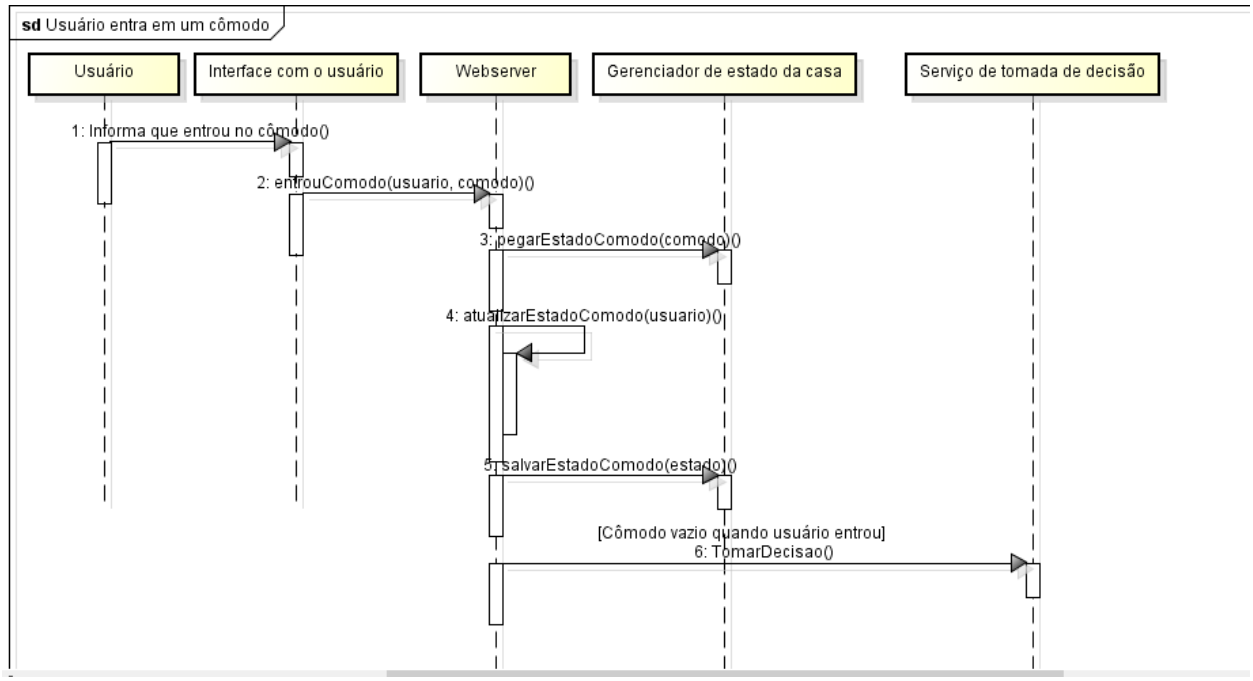

Figura 10: Diagrama de sequência de usuário entrando em um cômodo

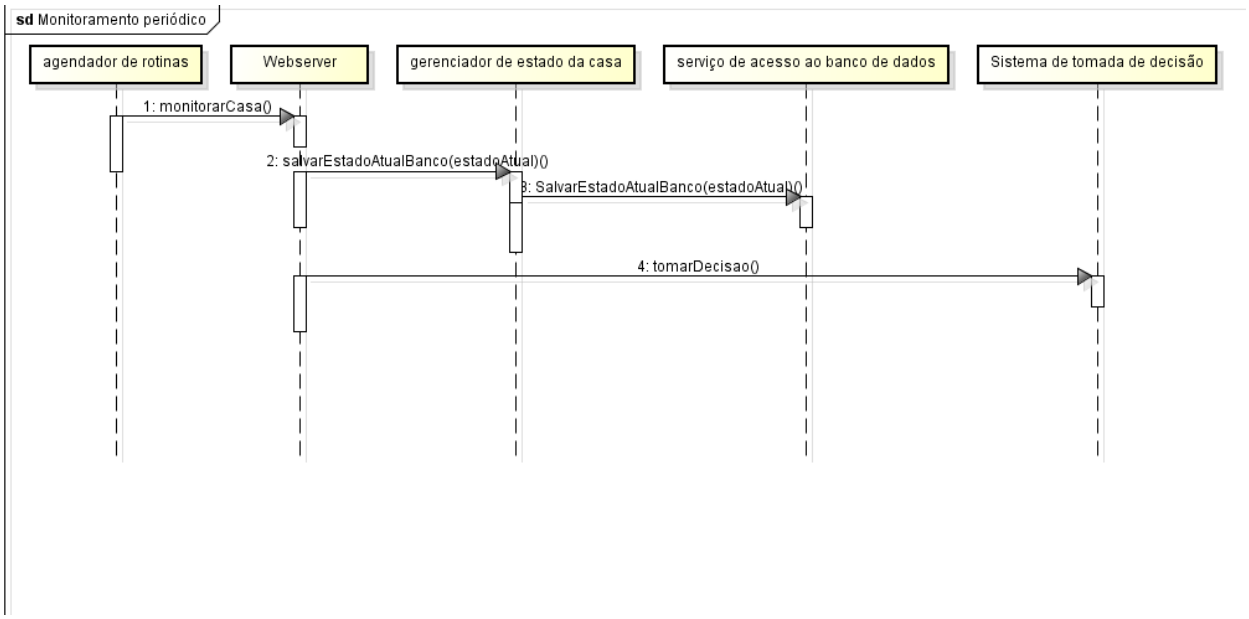

Figura 11: Diagrama de sequência de monitoramento periódico

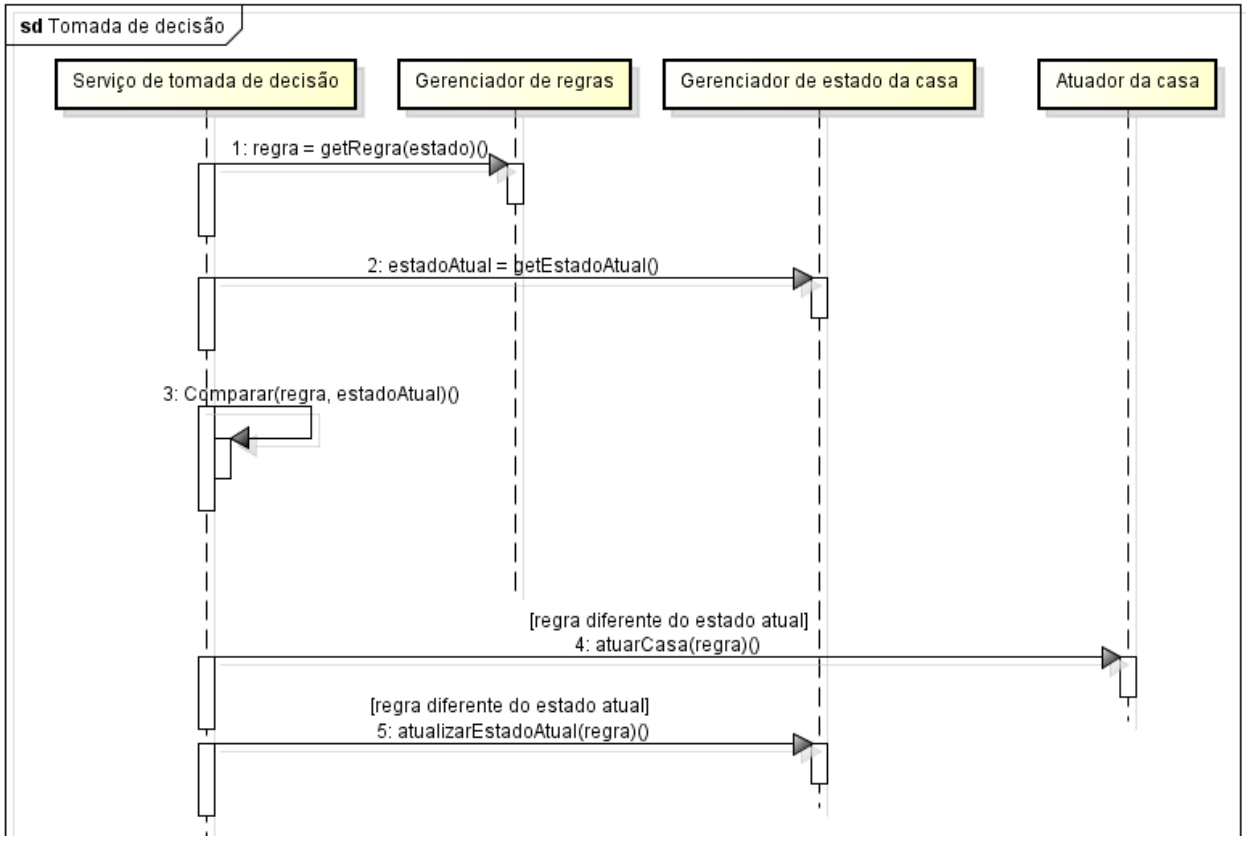

Figura 12: Diagrama de sequência de tomada de decisão