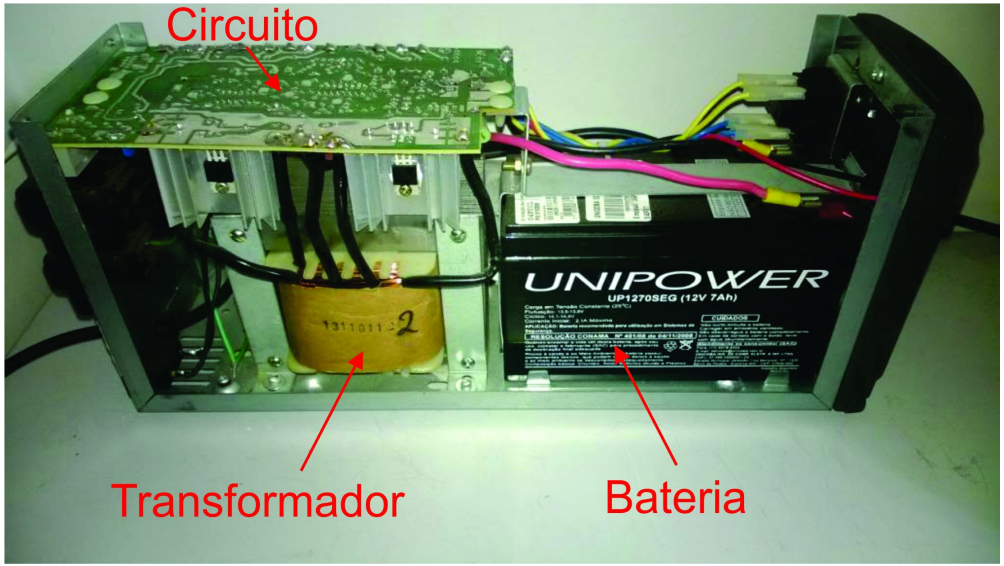
UPS com planta supervisionada
Apresentação final

Aluno: João Guilherme de Oliveira Jr.

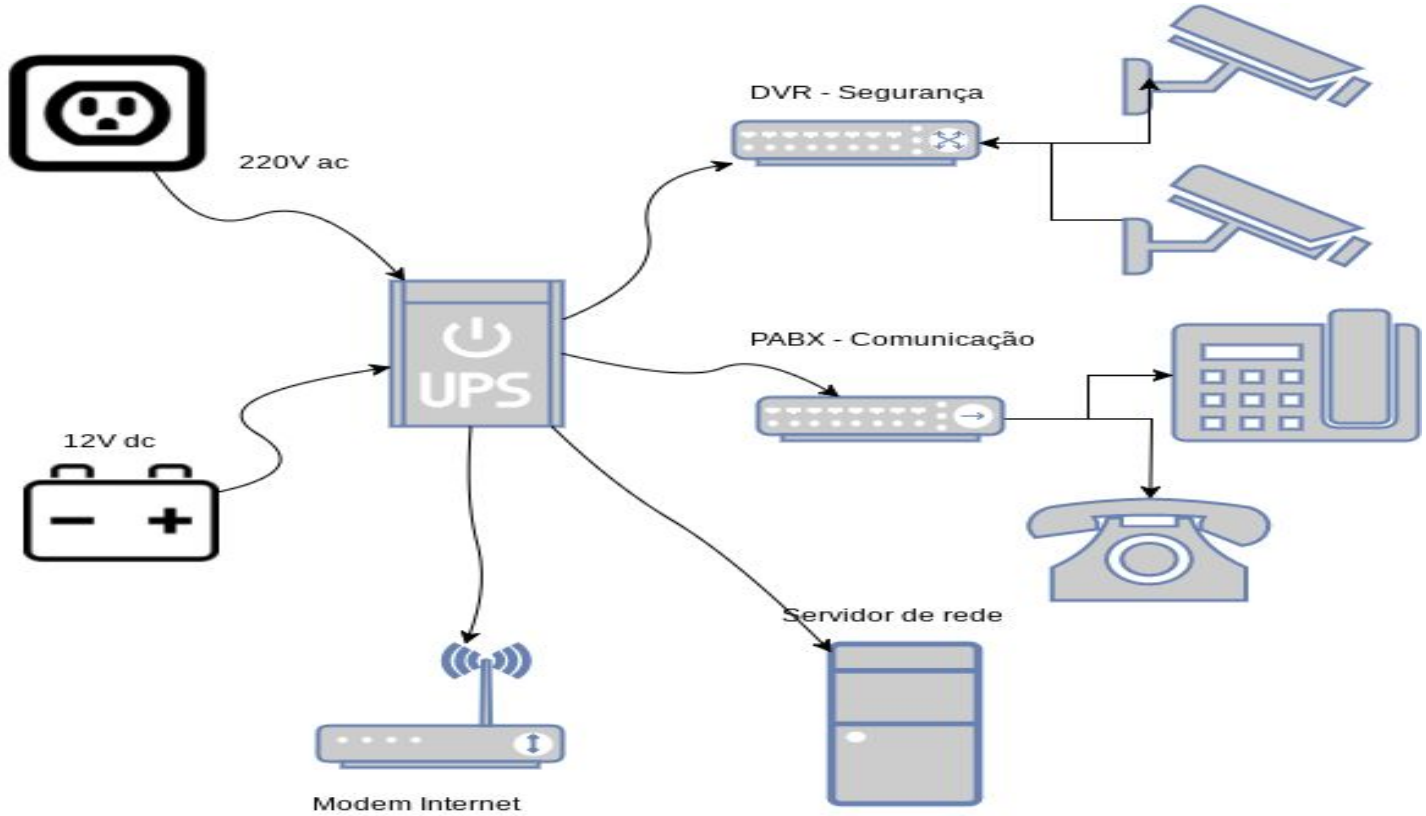
Motivação

UPS (Uninterrupted power source) ou mais conhecido como “no-break” é um dispositivo que mantém ligada todos os dispositivos nele alimentados na ocasião de falta de energia.



Motivação - continuação

Diagrama dos dispositivos ligados no UPS atualmente:



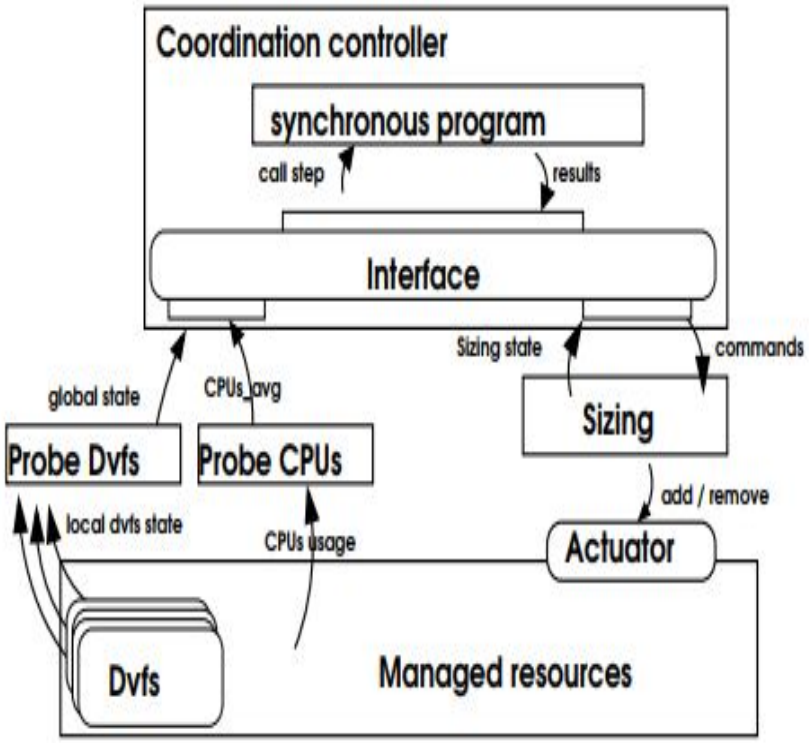
Motivação - continuação

- O problema com o esquema atual é que sem supervisão o no-break não tem como saber se um dispositivo precisa ser mesmo alimentado, vamos supor que a falta de energia ocorra a noite nessa situação provavelmente manter o computador ligado não trará nenhum benefício pois nenhum usuário está acordado para utilizá-lo. O computador é o dispositivo que mais consome o recurso da bateria do no-break então ser capaz de desligar ele ou outros dispositivos mediante políticas pré estabelecidas pode economizar este recurso fazendo que a planta continue alimentada por mais tempo, de preferência até a energia ser restabelecida, momento este que todos os dispositivos podem voltar à operação normal.

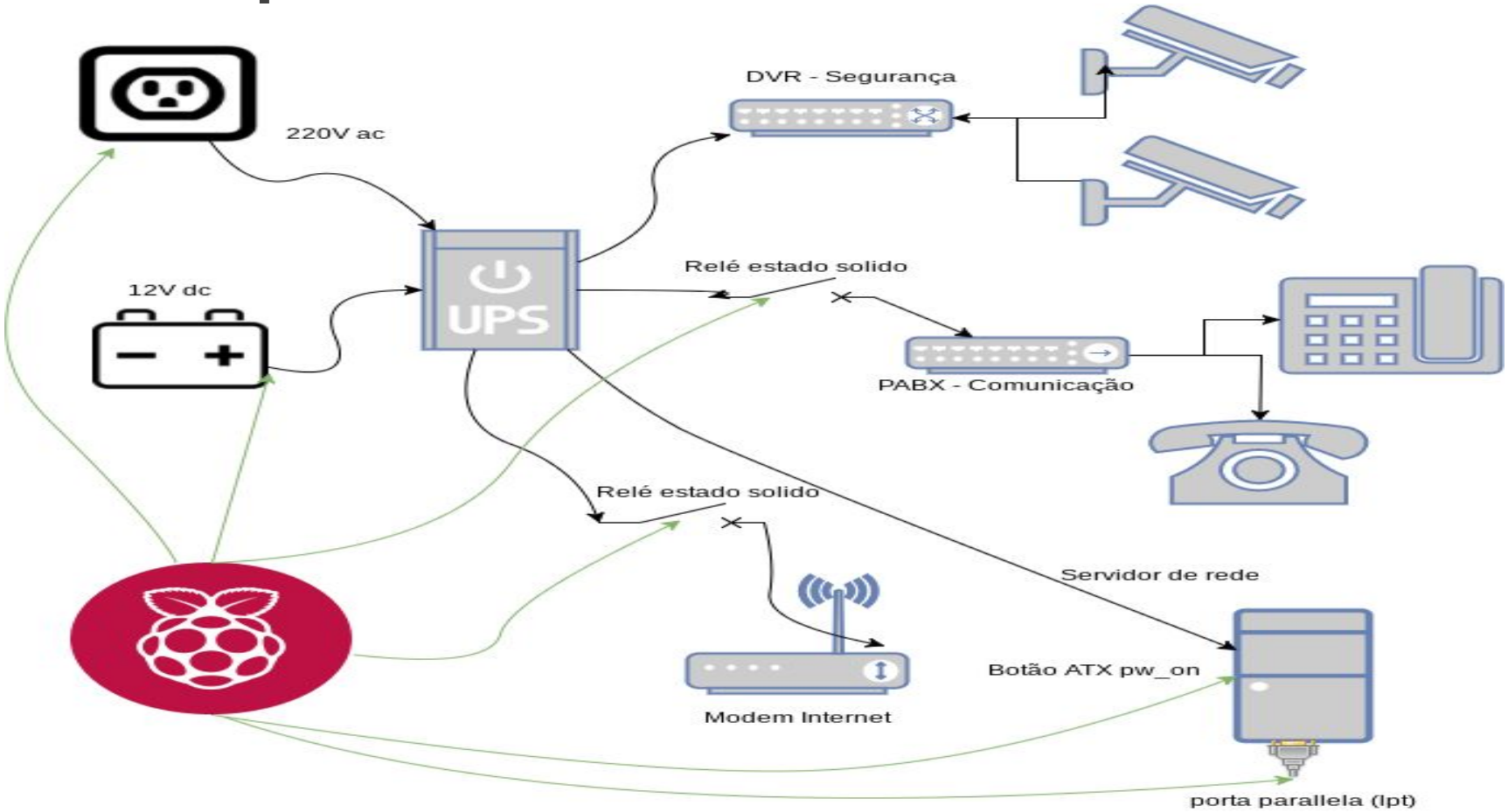
Artigos relacionados

Artigo “Coordinating Energy-aware Administration Loops Using Discrete Control”

De forma bem similar ao que proponho os autores desse artigo francês utilizaram a síntese de controlador discreto na linguagem bzr para gerenciar em uma malha de loop fechado aspectos de impactam na performance energética de um sistema informatizado atuando sobre configurações na máquina como velocidade do processador (via dvfs) e sizing management (provisões de replicação de nodes), após o controlador ter sido gerado



Proposta


220V ac
12V dc
UPS
DVR - Segurança
Relé estado solido
PABX - Comunicação
Relé estado solido
Servidor de rede
Botão ATX pw_on
Modem Internet
porta parallela (lpt)

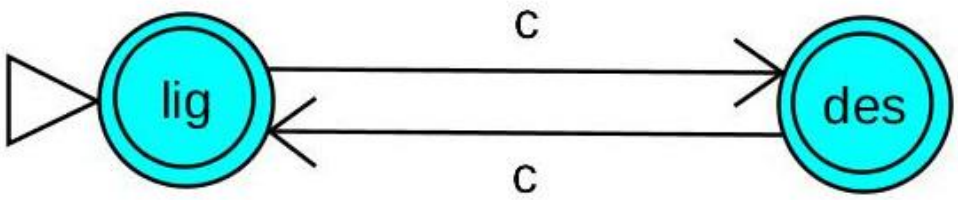
Automatos dos atuadores

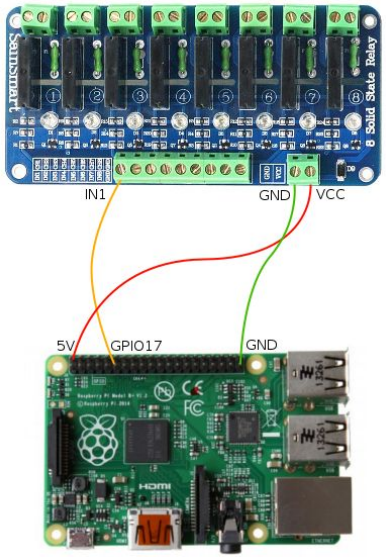
Relé de estado solido:

Será utilizado no pabx e no modem wifi.

Transições

C - Sinal proveniente do controlador





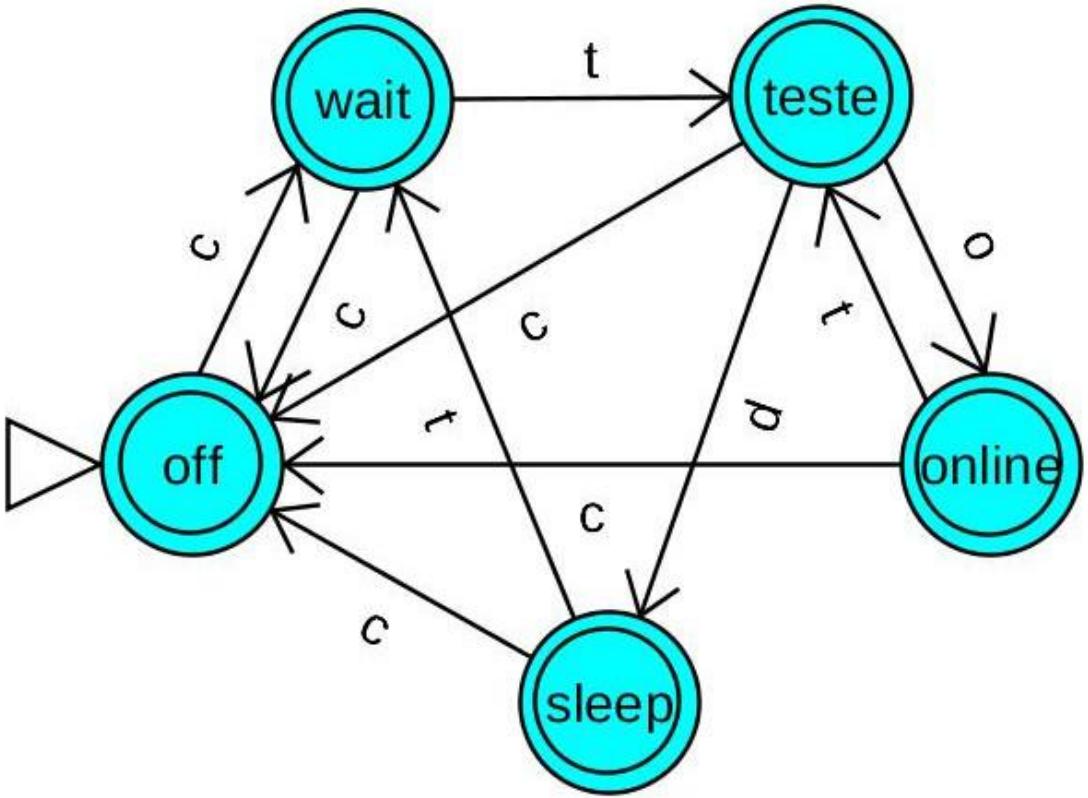
Autômato do modem adsl

c - evento controlado capaz de colocar o modem em estado de desligado ou ligado.

t - evento temporizado, um timer vai disparar após um determinado (por experimentação) tempo em segundos.

o - evento que apenas ocorre quando o teste é bem sucedido

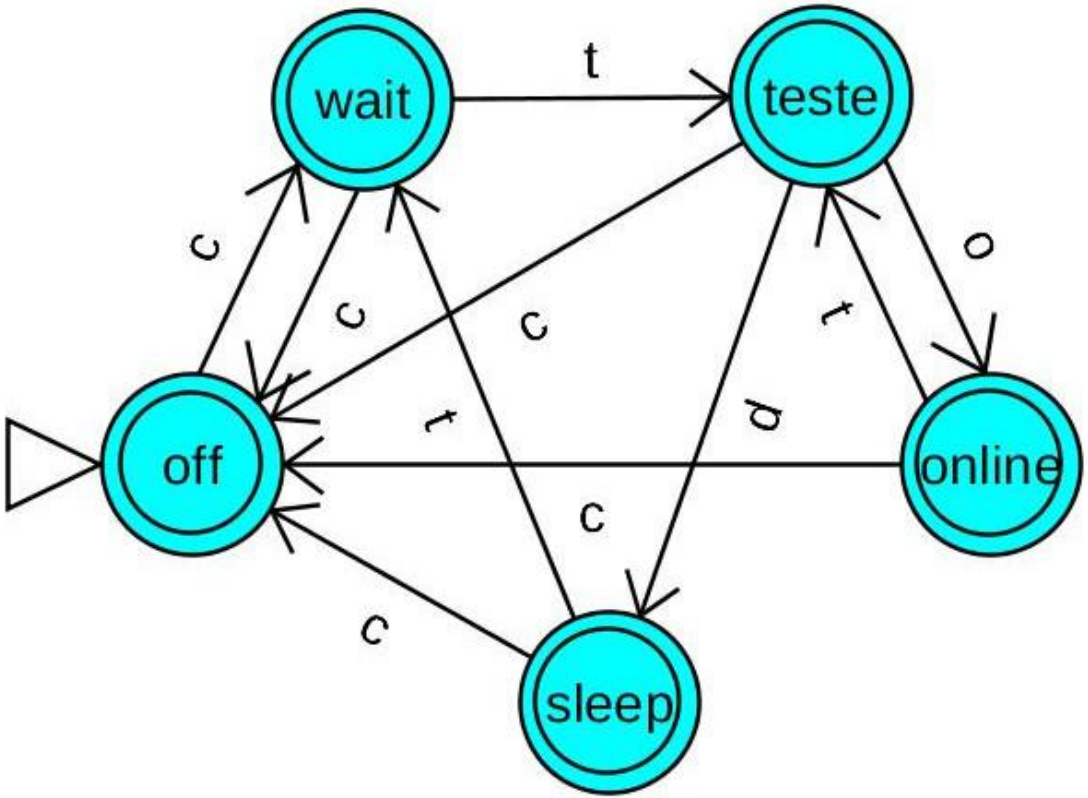
d - evento que ocorre quando o teste é mau sucedido, ou seja é a negação de o (!o)



Autômato do modem adsl

node modem_adsl(c, t, o :bool)
returns (atuador: bool;
estado:int)

d = !o



Autômato do computador

Computador:

Transições

A: SWATX

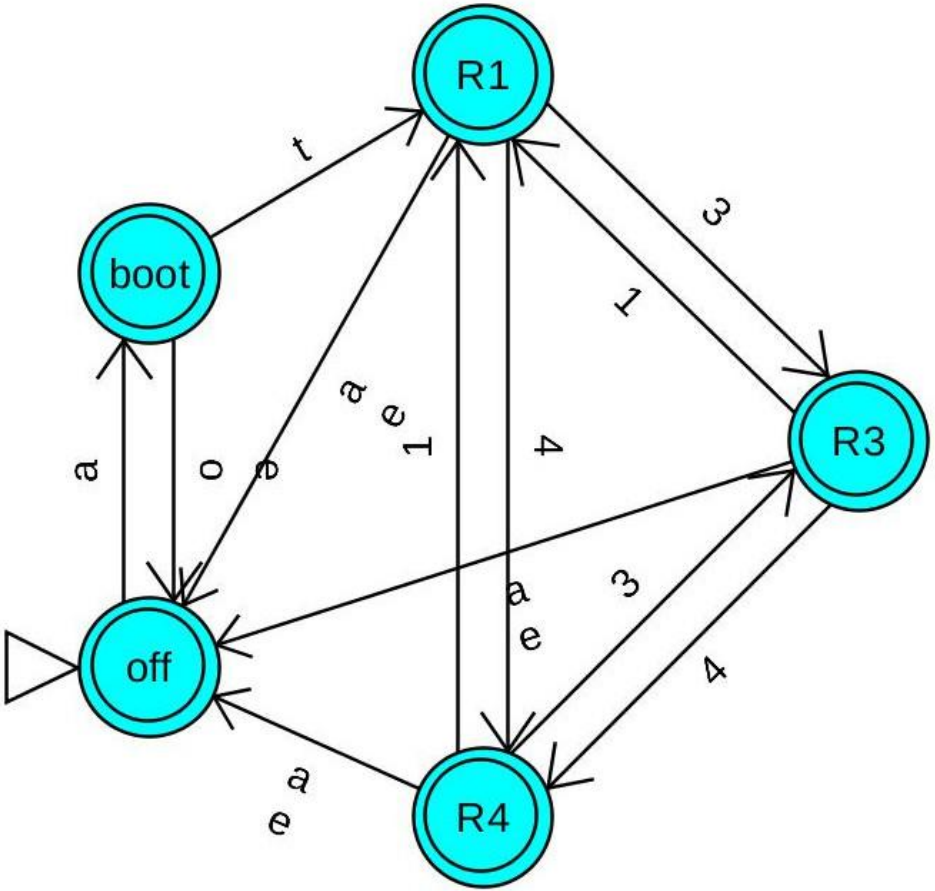
E: EMERG_OFF

O: FORCE_OFF

1: SWR1

3: SWR3

4: SWR4



Autômato do computador

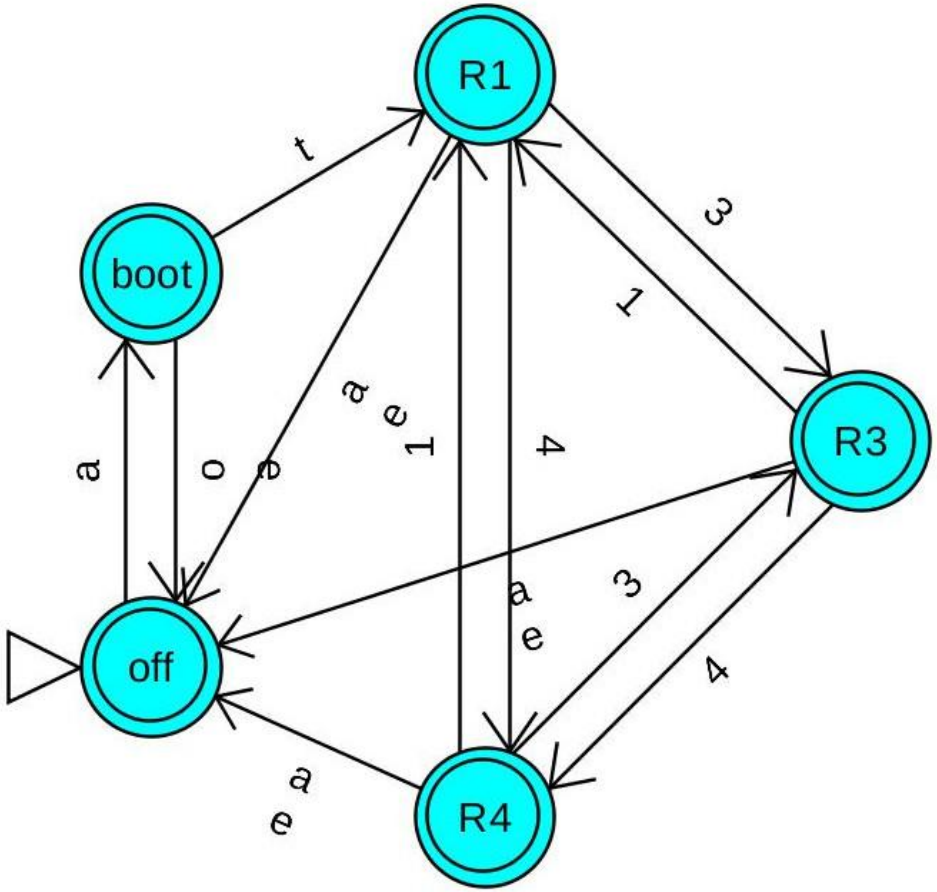
a - evento controlado capaz de colocar o pc em estado de desligado ou ligado.

e - evento de desligamento emergencial.

o - evento que pode desligar de maneira forçada o pc.

t - evento que finaliza o término do processo de boot.

1, 3 e 4 - eventos que alteram o runlevel do computador para R1, R3 e R4 respectivamente.



Autômato da bateria

Bateria:

Transições

A: !B_FULL & B_L1 & B_L2 & B_LOW

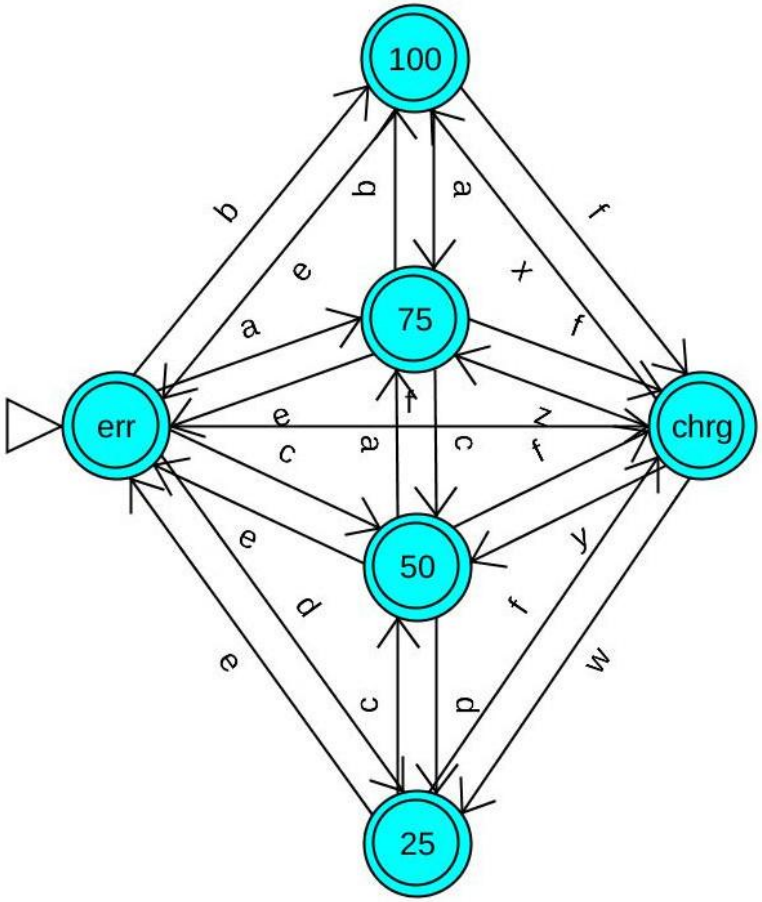
B: B_FULL & BL1 & BL2 & B_LOW

C: !B_FULL & !BL1 & BL2 & B_LOW

D: !B_FULL & !BL1 & !BL2 & B_LOW

E: !A & !B & !C & !D

F: AC_IN;



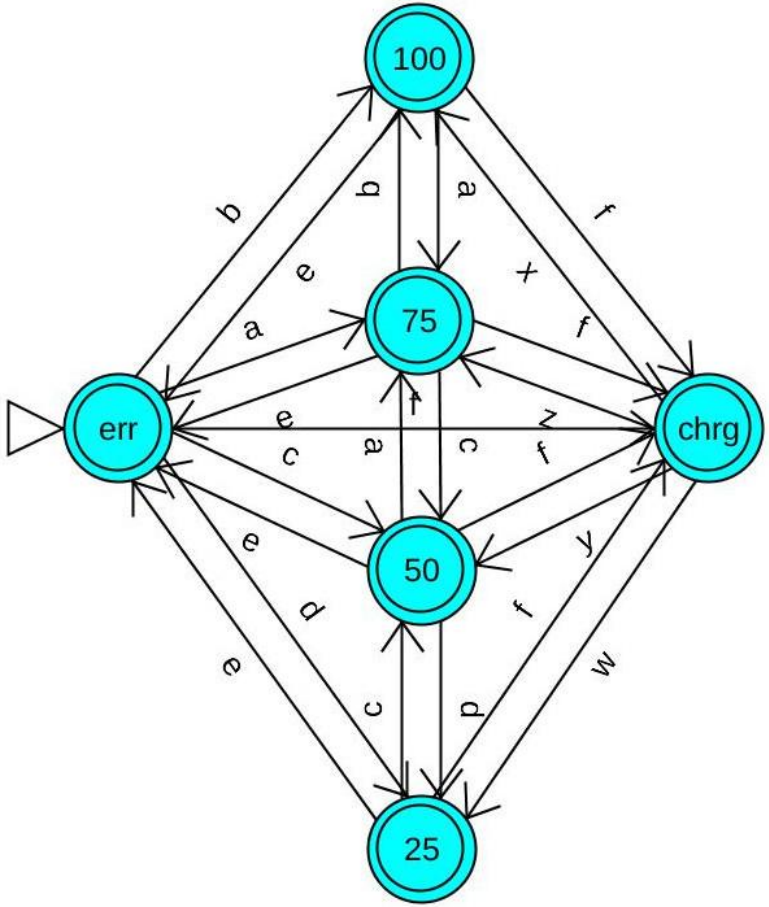
Autômato da bateria

a, b, c, d - eventos que são criados a partir da combinação dos sinais provenientes dos sensores da bateria.

e - evento de erro que é composto a partir dos sinais dos sensores da bateria.

f - evento não controlado que ocorre quando há falta de energia elétrica.

w, x ,y , z - eventos compostos pelo eventos a, b, c, d e f.



Autômato da bateria

type bat_states = BERR | B100 | B75 | B50 | B25 | BCHR

node bateria(ac, cp, l1, l2, v:bool) returns (estado_bat:bat_states)

e = not a & not b & not c & not d;

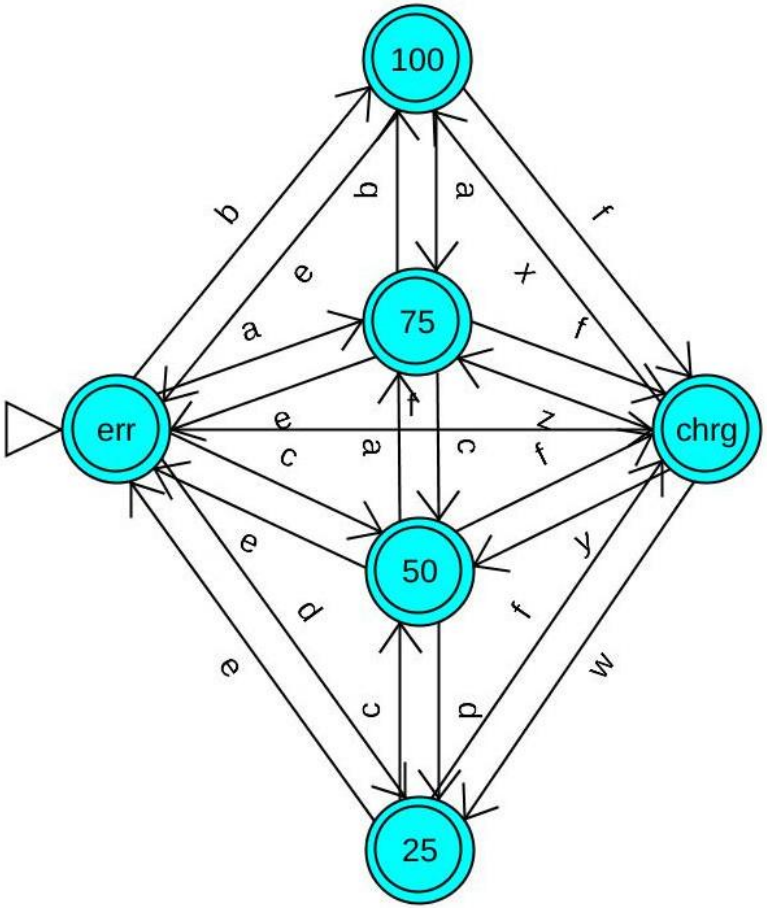
f = ac

z = not f & a

x = not f & b

y = not f & c

w = not f & d



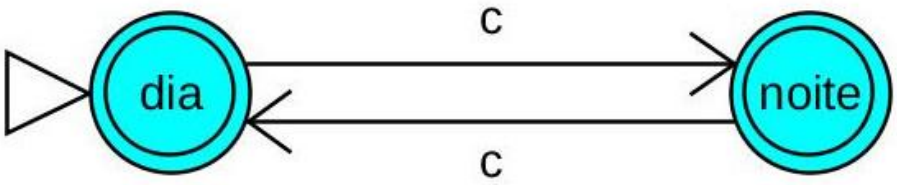
Sinais da bateria

Nível de energia da bateria:		Nível lógico dos sinais do sensor
100%		C = True L1 = True L2 = True V = True
75%		C = False L1 = True L2 = True V = True
50%		C = False L1 = False L2 = True V = True
25%		C = False L1 = False L2 = False V = True

Autômato do período do dia

Relógio:

A transição C será gerada por software a cada 12h a partir da 8h da manhã, ou seja às 8h e às 20h respectivamente, todos os dias.



Políticas - DIA

Durante o dia e quando o estado da bateria for 100:

- Computador ligado e pode estar em qualquer runlevel;
- Pabx ligado;
- Modem ligado;
- DVR ligado.

Durante o dia e quando o estado da bateria for 75:

- Computador ligado ou podendo estar em qualquer runlevel abaixo do R4;
- Pabx ligado;
- Modem ligado;
- DVR ligado.

Durante o dia e quando o estado da bateria for 50:

- Computador pode ser desligado ou estar no runlevel R1;
- Pabx desligado;
- Modem ligado;
- DVR ligado.

Durante o dia e quando o estado da bateria for 25:

- Computador desligado;
- Pabx desligado;
- Modem desligado;
- DVR ligado.

Políticas - NOITE

Durante a noite e quando o estado da bateria for 100:

- Computador ligado e pode estar em qualquer runlevel;
- Pabx ligado;
- Modem ligado;
- DVR ligado.

Durante a noite e quando o estado da bateria for 75:

- Computador desligado;
- Pabx desligado;
- Modem ligado;
- DVR ligado.

Durante a noite e quando o estado da bateria for 50:

- Computador desligado;
- Pabx desligado;
- Modem ligado;
- DVR ligado.

Durante a noite e quando o estado da bateria for 25:

- Computador desligado;
- Pabx desligado;
- Modem desligado;
- DVR ligado.

Políticas adicionais

1 - Quando voltar a energia todos os equipamentos devem ficar ligados;

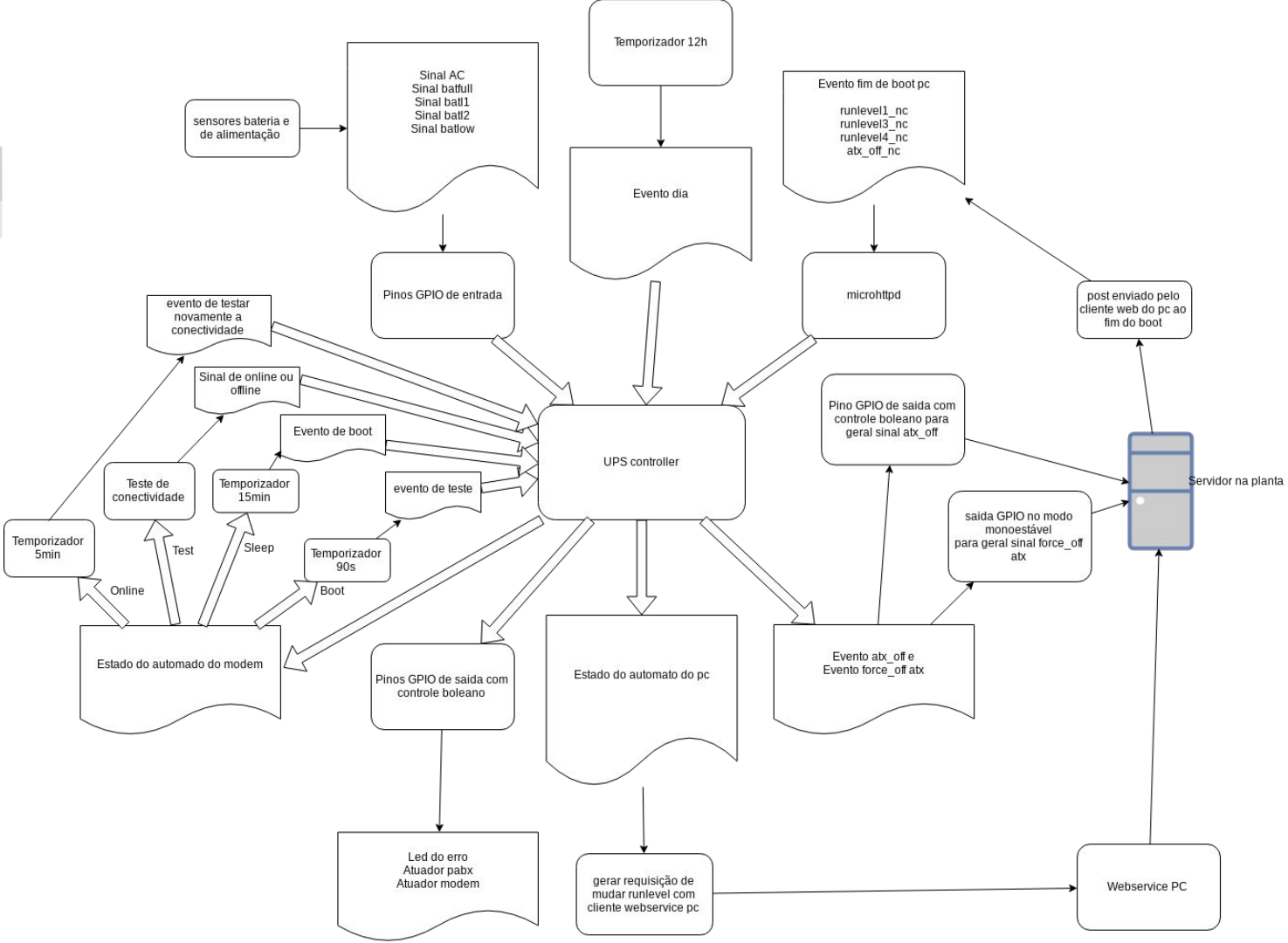
2 - O controlador só permite que a planta opere se a bateria não estiver com problemas;

3 - Se a bateria estiver com problema, o led indicador de troca de bateria deve permanecer ligado.

4 - Não manter o modem ligado se não houver conectividade com internet (falha do provedor), desligando-o em intervalos de 15 minutos.

O Controlador

A interface do controlador a ser utilizada no protótipo foi implementada em C contando com sincronização inicial da planta. A planta se comunicará com o controlador através dos atuadores e leds conectados às saídas GPIO. As entradas se darão via webservice rodando na porta local 8082 e polling de sinais provenientes dos sensores da bateria por pinos GPIO de entrada que deverá ser realizado a cada segundo, sendo que no protótipo este tempo foi reduzido para 10 segundos para fins de depuração.


Temporizador 12h
sensores bateria e de alimentação
Sinal AC
Sinal batfull
Sinal batl1
Sinal batl2
Sinal batlow
Evento fim de boot pc
runlevel1_nc
runlevel3_nc
runlevel4_nc
atx_off_nc
Evento dia
Pinos GPIO de entrada
microhttpd
post enviado pelo cliente web do pc ao fim do boot
evento de testar novamente a conectividade
Sinal de online ou offline
Evento de boot
UPS controller
Pino GPIO de saida com controle boleano para geral sinal atx_off
Servidor na planta
Teste de conectividade
Temporizador 15min
evento de teste
saida GPIO no modo monoestável para geral sinal force_off atx
Temporizador 5min
Test
Sleep
Temporizador 90s
Online
Boot
Estado do automado do modem
Pinos GPIO de saida com controle boleano
Estado do automato do pc
Evento atx_off e Evento force_off atx
Led do erro
Atuador pabx
Atuador modem
gerar requisição de mudar runlevel com cliente webservice pc
Webservice PC

Lições aprendidas

Este projeto foi um interessante e instigante desafio, com um desenvolvimento que se deu por completo em acompanhamentos durante as aulas que em primeiro momento se destinaram a esclarecer a teoria por trás das operação realizadas pelo linguagem BZR/Heptagon e a operação de composição paralela de autômatos, mas que em segundo momento dedicou-se a mostrar as implicações práticas dessas operações e suas dificuldades na concepção de toda modelagem assim como detalhes que podem trazer problemas na implementação do modelo.

- Aprendizado de uma nova linguagem para síntese de um controlador discreto;
- Aprendizado sobre o processo de compilação de um projeto C para uma arquitetura diferente (ARM);
- Aprendizado em codificação em C puro. Toda a interface do controlador gerado pela BZR/heptagon com o mundo real, bem como implementações de timers e funções auxiliares foi escrita em C e compilada junto com o controlador gerado para resultar no controlador final.

Obrigado!

Código e arquivos atualizados em:
https://github.com/joaogojunior/ufrpe_tmc.git