Tamagotchi embarcado no Raspberry Pi

'

Pedro Eugênio Machado de Lima
Engenharia Eletrônica
Universidade de Brasília - Campus Gama
Brasília, Brasil
peugenio95@gmail.com

Pedro Henrique Trindade Andrade
Engenharia Eletrônica
Universidade de Brasília - Campus Gama
Brasília, Brasil
pedrohenriqueatrindade@gmail.com

***Resumo*—Esse trabalho se propõe a apresentar uma versão contemporânea de um popular brinquedo eletrônico da década de 90 usando um Raspberry Pi como plataforma de desenvolvimento e implementação.**

Palavras-chave—Raspberry Pi, Tamagotchi.

I. INTRODUÇÃO

A-REVISÃO BIBLIOGRÁFICA

O Tamagotchi foi um brinquedo eletrônico lançado inicialmente pela Bandai em 1996 no japão, não levou muito até tomar conta de todo o mundo. O príncipio básico do brinquedo se baseia em criar e cuidar de um animal de estimação virtual, que se relaciona com o usuário a partir de uma tela geralmente de cristal líquido (LCD) e botões coloridos. A criatura digital possui necessidades diárias como fome, sono , felicidade e disciplina, podendo até morrer se negligenciadas. Todos esses marcadores definem a saúde geral do animal, criando uma sensação de progressividade e uma jogabilidade de alta liberdade ao jogador, fazendo cada gameplay criado especial e único.

O Raspberry Pi foi criado pela instituição de caridade Raspberry Pi Foundation com o objetivo de fornecer para estudantes e hobistas um computador de baixo custo e grande eficiência [1]. O Raspberry Pi é, de fato, um computador e, portanto, apresenta quase todas as funcionalidades de um computador desktop, como por exemplo embarcar um sistema operacional por exemplo um Linux [2].

Um sistema operacional como o Linux é formado pelo Kernel, drivers e outras aplicações como a interface gráfica e outros. Atualmente, existem diversas distribuições de linux facilmente acessíveis e com características específicas para cada aplicação., Apesar da capacidade do Raspberry ser semelhante a de um desktop, ele ainda apresenta limitações, como pouca memória RAM, logo Algumas versões adaptadas do linux foram criadas para funcionarem melhor na placa [2].

B-PROPOSTA DE PROJETO

Justificativa:

O Tamagotchi foi uma febre dos anos 90 mesmo com seu sistema limitado tecnicamente, pois a própria tecnologia da época limitava que o jogo fosse muito mais avançado, porque isso aumentaria o preço comercial, inviabilizando a produção. Com o passar do tempo a capacidade de computação e processamento evoluíram exponencialmente, como já é esperado se baseando na lei de moore para os equipamentos eletrônicos, já a popularidade do Tamagotchi diminuiu, o que abre a oportunidade para a inovação e a criação de um sistema ainda melhor com mais funcionalidades e interação, aproveitando tanto um público já familiarizado com o funcionamento do jogo original, aproveitando a nostalgia, quanto apresentando para um público mais jovem a mesma frenesi que levou uma geração inteira a comprar milhões de unidades desses jogos eletrônicos.

Objetivo:

Desenvolver uma versão mais interativa e mais moderna do tamagotchi usando um sistema embarcado no Raspberry Pi.

Requisitos funcionais:

- Desenvolver ferramentas de interação.
- Implementar em um Raspberry Pi.
- Criar uma comunicação com um aplicativo.
- Garantir entretenimento e diversão.
- Apresentar uma interface amigável.
- Garantir a robustez do sistema.

Requisitos não funcionais:

- Raspberry Pi 3.
- Sistema operacional
- Sensores

Benefícios:

Com a conclusão do projeto, os estudantes irão adquirir os conhecimentos relativos à sistemas embarcados bem como terão a oportunidade de revitalizar um dispositivo eletrônico de entretenimento que foi marcante para uma geração inteira.

II. DESENVOLVIMENTO

A. Visão geral

O sistema se baseia no modelo de cliente/servidor, onde o cliente é o computador pessoal do usuário e o Raspberry é o servidor. Para o acesso do usuário foi desenvolvida uma interface gráfica usando a linguagem C#. O código do cliente recebe as informações da interface e as passa para o servidor usando o protocolo TCP/IP. O servidor recebe os dados e os válida ou registra no banco de dados. Com a permissão de acesso, o usuário pode realizar ações que serão enviadas para o servidor e processadas, retornando novos status. Além disso o servidor possui um algoritmo que representa a inteligência artificial do jogo, variando os status de forma independente e gerando respostas diferentes até para ações iguais do usuário.A figura 1 apresenta a visão geral do projeto.

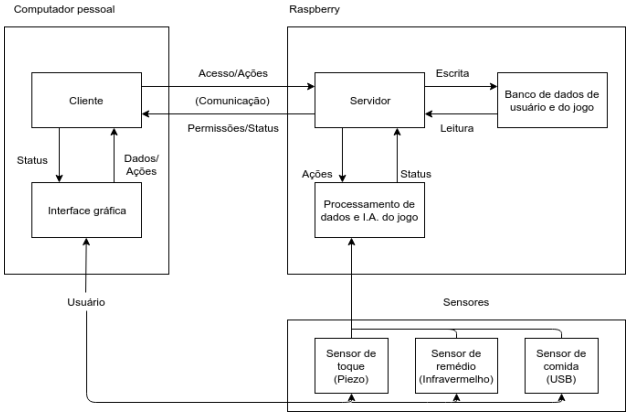


Figura 1: Diagrama de blocos do Tamagotchi.

B. *Hardware*

O desenvolvimento do hardware para o jogo se baseia em cria formas de interação lúdica com o bicho, assim como informar a ocorrência de eventos específicos.

Para a entrega do ponto de controle 3, optou-se em adicionar alguns elementos mais simples que pudessem ser controlados usando apenas os conhecimentos básicos sobre o GPIO do Raspberry.

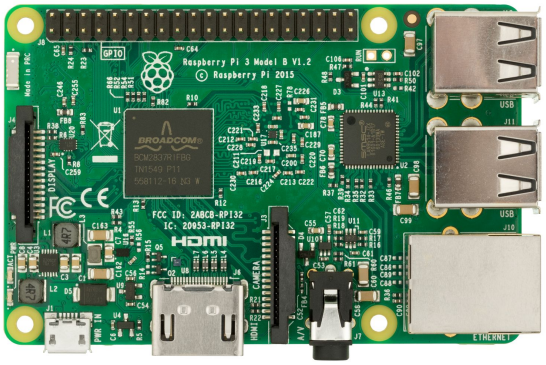
Inicialmente adicionou-se um LED para a sinalização de solicitações recebidas . Optou-se em colocar um LED RGB para sinalizar as atualizações dos *status*, como os pinos de entrada e saída do Raspberry só funcionam com sinais digitais, o LED RGB, apresenta as cores azul, verde, vermelho e branco, para indicar quando cada *status* é modificado pelas solicitações do cliente.

Outro elemento importante é a utilização de comunicação sonora. Para isso usou-se um sensor piezoelétrico que ao sofrer uma pressão mecânica gera uma tensão correspondente que é interpretada de forma binária pelo Raspberry. Caso o Raspberry receba o valor ‘1’ no pino de leitura do piezo, o servidor iniciar a reprodução de um áudio que até o momento é uma música que será reproduzida por um período de tempo previamente definido.

Os elementos apresentados são apenas a base teórica da interatividade do projeto, já planeja-se formas mais dinâmicas e lúdicas e aplicá-las, além disso, com o refinamento do projeto, novos elementos interativos serão adicionados e que já estão sendo testados.

Lista de materiais:

1. Raspberry Pi 3



O raspberry é uma plataforma de desenvolvimento de baixo custo que permite criar aplicações para sistemas embarcado. Com este processador é possível criar códigos com muita facilidade e flexibilidade bastando um monitor e um teclado sobressalentes, ou mesmo remotamente usando algumas das ferramentas disponíveis na *internet*.

O raspberry possui um sistema operacional embarcado que permite o desenvolvimento de softwares em alto nível. Ele permite desenvolver projetos usando uma interface gráfica comum.

Alguns aspectos interessantes deste processador é o conjunto de ferramentas que já foram desenvolvidas exclusivamente para ele, como Raspbian, Sonic Pi, Chromium, entre outras. Além disso outras ferramentas já vem integradas nela como a linguagem de programação Python, que permite desenvolvimento em alto nível, e o jogo Minecraft, que permite o entretenimento e é bastante comum entre os usuários.

Neste projeto o Raspberry foi usado como principal plataforma de desenvolvimento. Os códigos foram escritos todos em linguagem C, às vezes remotamente às vezes não. Ele é o elemento principal da comunicação do software, pois ele representa o servidor.

Para o controle dos sensores usou-se os pinos de GPIO. Os sensores apresentam tensões nos pinos de acordo com o uso e , em nível de código, o *raspberry* lê os valores que o GPIO que são armazenados em arquivos e os processa no software.

2. Sensor piezoelétrico

O sensor piezoelétrico se baseia no efeito piezoelétrico, ou seja, ele transforma tensões mecânicas em tensões elétricas. Isso permite criar uma interação física com o objeto, você pressiona um ponto dele e um sinal é enviado para o processador gerar uma reação.

Neste projeto, o piezo foi usado para simular um contato físico com o *tamagotchi*, algo simular a dar carinho a um animal de estimação.

3. Sensor infravermelho

O sensor infravermelho é composto de um emissor e um receptor. O Emissor será usado para simular o uso de uma seringa para representar cuidados médicos. O receptor fará parte da estrutura do tamagotchi, ou seja, será necessário aproximá-la de um ponto específico dele para que funcione e assim o processador receba um sinal para reagir de acordo.

Para o projeto usou-se o sensor para simular uma seringa médica, ou seja, o receptor ficaria no *tamagotchi* e o emissor em um componente a parte que o usuário deve usar para medicar virtualmente o *tamagotchi*.

4. *Players* e caixas de som

Para uma maior interatividade, optou-se pela reprodução de áudios pré-definidos para que o processador informe ao usuário as necessidades do jogo. A melhor forma de implementar isso seria usando uma caixa de som dentro do *tamagotchi*, mas não obteve-se os equipamentos necessários, outras opções são o uso de alguns reprodutor por *blutooth* ou pela própria entrada para fones de ouvido do *Raspberry*

5. LED's

Para um melhor acompanhamento do funcionamento do processador alguns LED's foram usados para indicar o status do processador, como por exemplo, quando o Tamagotchi solicita uma ação ele espera o usuário realizá-la, então um LED permanece acesso até que a ação seja executada.

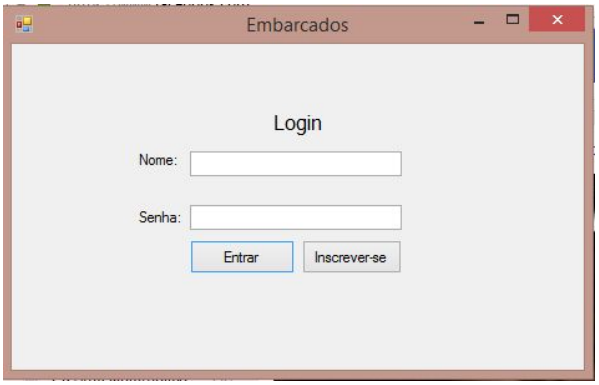
6. Pen-drive

Visando uma maior interatividade do jogo, decidiu-se usar um *pen-drive* para simular a forma de alimentação do Tamagotchi, ou seja, quando o processador solicita ao usuário comida, ele deve inserir o *pen-drive*, então o software verifica se existe um arquivo corresponde a comida simulada no dispositivo e reage de acordo.

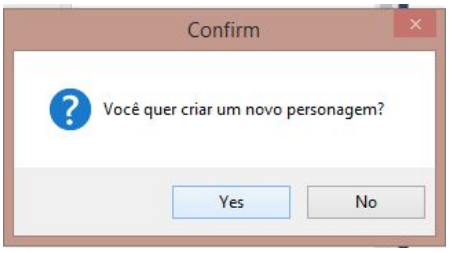
C. *Software*

Interface Gráfica:

Tela 1 (login):

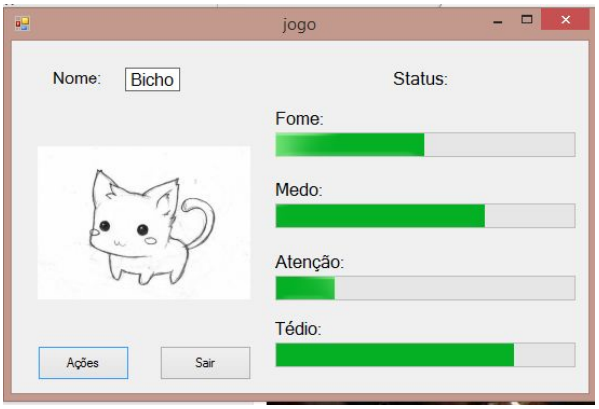


Essa é primeira tela que aparece para usuário, nessa tela ele possui duas opções, o usuário pode tanto colocar seu nome e senha nos campos correspondentes e clicar "Entrar" , assim o sistema mandará informação via tcp para o server checar se o nome e senha do usuário se encontram no banco de dados (olhar tópico "comunicação"), caso esteja ele irá diretamente para Tela 2 (status) , caso contrário aparecerá um aviso para o usuário se registrar. A outra opção possível é clicar em "Registro", nesse caso aparecerá o seguinte aviso:



Em caso de sim, o sistema avisará ao server que o usuário pretende apagar as informações do animal anterior e criar um novo do começo, indo para Tela 3. Se não, o sistema apenas registra o nome e senha desse novo usuário que quer se cadastrar no banco de dados.

Tela 2 (status):

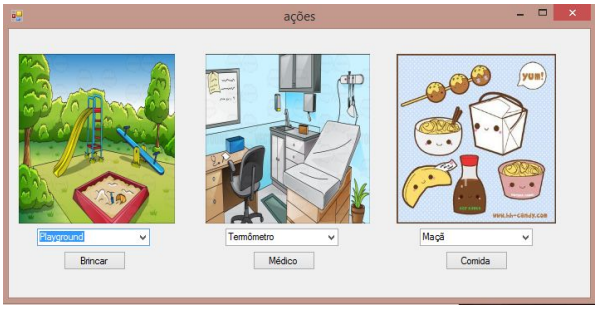


Nessa tela o usuário poderá receber as informações dos status principais do tamagotchi, recebidas pelo server. clicando no botão ações, o usuário será levado para a Tela 4.

Tela 3 (Registro):

Nessa tela o usuário escolhe o nome do animal e o tipo, sendo informações importantes para o resto do funcionamento do programa, já que essas informações serão guardadas no banco de dados para acesso do cliente no futuro. Após clicar registrar o usuário irá para Tela 2 (status).

Tela 4 (ações):



Nessa tela o usuário poderá escolher entre 3 tipos de ações (Brincar, ir ao médico e dar comida ao bichinho). em cada uma dessas ações existem diversos tipos de modalidades da mesma ação, da qual aparecerá uma lista para o usuário. Por exemplo, ele poderá alimentar o bicho com feijoada, batata ou maçã. tipos diferentes de comida, com consequências diferentes.

Ao escolher sua ação e clicar o botão, o cliente acionará o server que computará a consequência da ação e responderá com uma atualização de status. o programa voltará para a Tela 2.

Comunicação cliente-servidor:

A comunicação do cliente com o servidor se dará da seguinte forma: Existem três ações principais que o usuário poderá fazer no software para acionar o cliente e essas são: Entrar,Registrar e Ação do bicho. O cliente antes de tomar qualquer ação, conectará com o servidor e mandará qual ação ele pretende fazer, em forma de um inteiro 32 bits, sendo esses codificados da seguinte forma:

0 - Entrar
1 - Registrar
2 - Ação do bicho

Ele esperará o servidor responder com o status da conexão, sendo esse um inteiro 0 ou 1. Se o servidor responder com 0 ou não responder de forma alguma, significará que o sistema não está conectado, acionando uma mensagem de erro. Se ele responder com 1 sabemos que a conexão está normal, Passando para o próximo estágio.

No processo de entrar e registrar o cliente enviará o nome de usuário e a senha, com um buffer de 120 caracteres. por exemplo:
nome: Fulano
senha: teste

O cliente enviará:
Fulano\0\0\0\0\0\... teste\0\0\0\0\0...

Esses barras-zeros representam o não uso total do buffer, o espaço vazio, por sobrarem mais caracteres que poderiam ser usados.

No caso da ação do bicho, o cliente enviará dois inteiros, um representando o tipo da ação e o outro representando o sub-tipo. Codificado da seguinte forma:

Primeiro inteiro:
0 - Brincadeira
1 - Medico
2 - Comida

Segundo inteiro:
Brincadeira: 0 - Playground
1 - Abraço
2 - Pista de skate

Médico: 0 - Termômetro
1 - Psicólogo
2 - Raio-x

Comida: 0 - Maçã
1 - Batata Frita
2 - Feijoada

Exemplo:
Usuário alimentou o bicho com batata frita.
O cliente enviará:
2 1

Tanto ao entrar, registrar e na ação do bicho, a resposta de informação server será da seguinte forma, em ordem:
-60 chars para o nome do bicho
-depois mais 1 int para o tipo do bicho
-depois 4 ints de status (para atualizar a tela)
-um inteiro para dizer se ação foi bem sucedida ou não

Servidor:

O servidor é responsável por receber as solicitações do cliente e através da lógica do jogo responder adequadamente a cada caso. Ele também é responsável pelo banco de dados dos jogadores, onde são armazenados os dados necessários para cadastro e

acesso dos usuários, bem como algumas informações que conferem ao jogo uma ilusão de memória.

Para processar as solicitações dos clientes, o servidor cria um *socket* para receber essas solicitações seguindo o protocolo TPC. Após o recebimento dos dados do cliente, no caso as ações realizadas pelo jogador, o servidor modifica os *status* seguindo uma lógica pré estabelecida de valores em conjunto um fator de aleatoriedade. Em seguida, os novo *status* são armazenados em arquivos binários e enviados para o cliente.

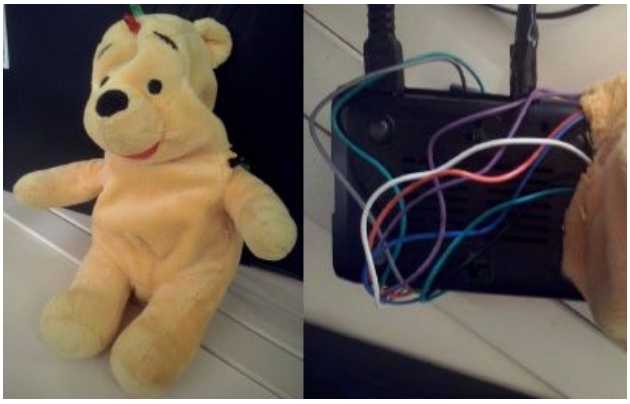
Outra função do servidor é a lógica de inteligência artificial do jogo, ou seja, o modo como os *status* variam sozinhos para simular as necessidades do bicho. Para isso usa-se um sinal que com uma certa frequência chama uma função que modifica os *status* seguindo uma lógica fixa.

Além disso, esse sinal gera aleatoriamente solicitações de ação do usuário de acordo com o modelo de necessidades do jogo (fome, tédio, atenção). Essas solicitações são feitas a partir de áudios pré-gravados, um para cada caso. Assim que a ação é realizada, seja por *hardware* ou seja pela comunicação de internet, outro áudio é reproduzido em resposta.

A medida que os status se modificam novas funções são solicitadas para processar condições específicas, um exemplo disso é a função *morte()*. Essa função é chamada assim que o valor de um dos status atinge o seu limite e o bicho é considerado morte, assim o servidor informa a causa da morte e o jogador deve reiniciar o jogo criando um novo bicho.

III. Resultados

A figura a seguir apresenta o protótipo do t*amagotchi* funcionando.



É possível observar como o raspberry ficou disposto dentro do ursinho de pelúcia. Pela natureza do *raspberry*, o processador apresenta um fácil aquecimento, para poder colocá-lo em um ursinho de pelúcio sem correr riscos, utilizou-se uma proteção para *raspberry* com um *cooler* embutido.
Alguns problemas foram encontrados durante a implementação final, entre eles:

- A necessidade de usar um fonte de energia externa prejudica a autonomia do jogo, mantendo o ursinho de pelúcio sempre conectado a rede elétrica..
- A reprodução dos áudios para a comunicação com o usuário não estavam funcionando por adversidades na conexão por *Bluetooth*.
- Não conseguiu-se reparar o funcionamento dos sensores ao da interface, ou seja, para realizar uma ação no hardware ainda era necessário enviar uma confirmação pela interface.

Entretanto, muitas coisa estavam funcionando bem, entre elas:

- A comunicação por *sockets*;
- O processamento dos sensores que envolvem o uso do GPIO e de processos em paralelo, ou seja, a função *fork()*;
- A lógica de necessidades do *Tamagotchi* e o uso de reprodução de música em nível de código;
- O controle dos pinos de entrada digital;
- Os sinais de alarme usados para chamadas de funções;
- A interface gráfica;
- O controle do *raspberry* de forma remota.

IV. Conclusão

O projeto possibilitou o aprendizado de muitas formas. Os conteúdos referentes à disciplina de sistema embarcados foram amplamente usados no desenvolvimento do projeto, desde manipulação de arquivos até o uso do *hardware* do *raspberry*.

Inicialmente, o projeto foi bastante fundamentado na comunicação por *sockets* ignorando as possibilidades de uso do *hardware*, logo na tentativa de integrá-los e aproximar o projeto do comportamento original do *tamagotchi*, muitas falhas ocorreram e limitaram a expansão do projeto, criando uma bagunça entre o que o *hardware* e o *software* deveriam fazer.

Aprendeu-se, então, que o uso de *sockets*, sinais, processos e outros conhecimentos de sistemas embarcados deveriam ser usados para criar novas possibilidade para o uso do *tamagotchi* e não apenas para tentar recriá-los usando esses recursos.

A seguir é proposta uma lista de algumas melhorias que podem ser realizadas para uma melhor implementação do projeto no futuro.

- Iniciar o projeto desenvolvendo o *tamagotchi* totalmente em *hardware* e posterior melhorá-lo usando novas ferramentas para criar novas possibilidade de uso;
- Utilizar uma fonte de energia móvel que possa ser colocada junto ao processador dentro do ursinho de pelúcia;
- Solucionar alguns problemas menores como a reprodução de áudios por *bluetooth*;
- Oferecer formas de interação se simulem melhor ações reais.

V. REFERÊNCIAS

[1] Raspberry Pi Brasil. **"O que é Raspberry Pi?"**. Disponível em: <http://raspberrypibra.com/o-que-e-raspberry-pi-4.html> acesso em 1º de abril de 2017.

[2] RICHARDSON, Matt; WALLACE, Shawn. **"Primeiros Passos com o Raspberry Pi"**. Disponível em: <http://static.novatec.com.br.s3.amazonaws.com/capitulos/capitulo-9788575223451.pdf> acesso em 1º de abril de 2017.

[3] MAITA, aki. **"Who came up with Tamagotchi?**". Disponivel em: <http://www.mimitchi.com/html/q10.htm> acesso em 1º de abril de 2017.

[4] Mimitchi. **"Tamagotchi angel instructions".** Disponivel em: <http://www.mimitchi.com/html/tainst.htm> acesso em 1º de abril de 2017.