Sistema de Controle e Acesso a Ambientes Restritos

Leonardo Silva Oliveira[1], Pedro Henrique de Araújo Siqueira[1]

[1]Instituto Federal de Mato Grosso do Sul – Três Lagoas-MS

leonardo.oliveira11@estudante.ifms.edu.br, pedro.siqueira@ifms.edu.br

Resumo

Soluções de automação residencial têm crescido muito nos últimos anos, devido ao barateamento da produção dos dispositivos eletrônicos e à redução de seus tamanhos. Dentre as soluções procuradas, estão as de proteção e segurança. O objetivo desse trabalho foi desenvolver um sistema de controle e acesso a ambientes restritos. Esse sistema possibilita o acesso por meio de impressão digital ou por senha, e também mantém o registro de horário de todos que fizeram o acesso. Para a realização do trabalho, foi utilizado o minicomputador Raspberry Pi como controlador dos dispositivos, a linguagem de programação Python para desenvolver o software, e o banco de dados SQLite para gravação dos dados e usuários . Ao final do projeto, alcançou-se um protótipo funcional do sistema de segurança.

Palavras-chave: raspberry pi, leitor biométrico, fechadura elétrica

Introdução

A proposta do projeto ao qual esse trabalho faz parte foi desenvolver uma solução de segurança eficiente para uso em ambientes restritos compartilhados, como laboratórios ou escritórios, que possuem conteúdos sensíveis ou de valor. A solução apresentada é uma fechadura elétrica a ser instalada na porta do ambiente que se quer manter protegido, que é acionada por um teclado através de senha ou por um leitor biométrico através da leitura de digital. O objetivo deste trabalho foi desenvolver o software que controla a fechadura eletrônica e registra as tentativas de acesso ao ambiente protegido.

Para realizar esse controle e registro, o software, a fechadura, o teclado e o leitor biométrico são acoplados a um microprocessador. A essa combinação de módulos é dado o nome de sistema embarcado, que é um sistema computacional dedicado a uma função mecânica específica (Heath, 2002).

Para o desenvolvimento do software, há várias tecnologias possíveis de serem utilizadas. Conforme o índice de linguagens de programação mais utilizadas da IEEE Spectrum (2020), dentre as linguagens para sistemas embarcados, o Python e o C ocupam as primeiras posições. Mas a linguagem Javascript, já sendo praticamente ubíqua no que diz respeito à internet, vem crescendo muito em ambientes embarcados também (Okoi, 2019).

Sara Santos (2018) desenvolveu um sistema embarcado com sensor biométrico usando o microprocessador Arduino e a biblioteca Adafruit Fingerprint Sensor Library, escrita na linguagem C. Esse é um projeto simples apenas para fins didáticos em que ela mostra como registrar e buscar por impressões digitais usando as tecnologias citadas.

Bastian Raschke (2015) desenvolveu um sistema embarcado com sensor biométrico usando o Raspberry Pi e a biblioteca pyfingerprint. O Raschke escreveu essa biblioteca na linguagem Python e se baseou na biblioteca Adafruit em C para fazê-la. Esse também é um projeto didático para ensinar a usar a biblioteca.

Paulo Abreu (2016) desenvolveu um sistema embarcado com fechadura elétrica que usa o Raspberry Pi e a biblioteca onoff do Node.js, escrita em Javascript. Nesse projeto, a fechadura elétrica é ativada por meio de uma requisição http a um servidor web sendo executado no Raspberry Pi.

Metodologia

Para o sistema embarcado, foi escolhido o Raspberry Pi 3B+, que não é apenas um microcontrolador, mas também um minicomputador. A sua escolha se deu pela facilidade de se desenvolver nele, que por já ser um minicomputador, não é necessário um outro computador para fazer a ponte com o microcontrolador, tudo é desenvolvido diretamente com ele. O Raspberry Pi 3B+ funciona com o sistema operacional Linux.

Como linguagem de programação, foi escolhido o Python por ser a linguagem mais utilizada em sistemas embarcados, logo possui uma maior quantidade de bibliotecas e funções para serem aproveitadas, bem como uma boa comunidade e suporte para uso. Além de que Python é uma linguagem mais fácil de ser aprendida comparada ao C, por exemplo.

Como esse é um sistema embarcado simples, sem milhares de transações paralelas ocorrendo no banco de dados, optou-se por usar o sistema de gerenciamento de banco de dados SQLite, por ser leve, prático e rápido para essas situações.

A metodologia empregada para desenvolver o software foi o modelo incremental. Neste modelo, são entregues várias versões do sistema, sendo que a cada versão, mais funcionalidades vão sendo incorporadas ao sistema (Sommerville, 2011).

Resultados e Discussão

Para o banco de dados, foram criadas duas tabelas, uma representa os usuários que possuem permissão de acesso ao

ambiente, e outra representa os horários em que cada usuário acessou o ambiente. A figura a seguir apresenta o diagrama entidade-relacionamento das tabelas do banco de dados.

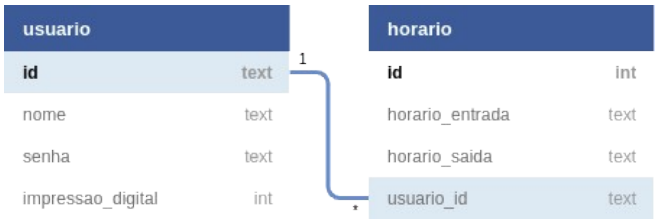

Figura 1: Diagrama Entidade Relacionamento

A chave estrangeira `usuario_id` da tabela `horario` faz ligação muitos-para-um com a chave primária `id` da tabela `usuario`. Cada vez que o usuário informa sua senha ou impressão digital, é registrado um novo horário, seja de saída ou de entrada, com o `id` do usuário.

Todas as impressões digitais ficam salvas na memória interna do leitor biométrico, e cada uma delas possui um valor numérico que representa a posição onde se encontra na memória. A coluna `impressao_digital` da tabela usuário faz referência à posição onde a impressão digital do usuário é salva na memória interna do leitor biométrico.

O Sistema de Controle e Acesso a Ambientes Restritos (SCAR) possui dois programas. Um deles se chama `Manager`, e é responsável por criar, editar, apagar ou listar os usuários do banco de dados. O outro programa se chama `scar`, e controla a fechadura elétrica e registra os horários que os usuários acessam o ambiente.

Seguindo o paradigma orientado a objetos para melhor estruturar o código, cada entidade do sistema representa um objeto nos programas. A seguir é apresentado o diagrama de classes do sistema:

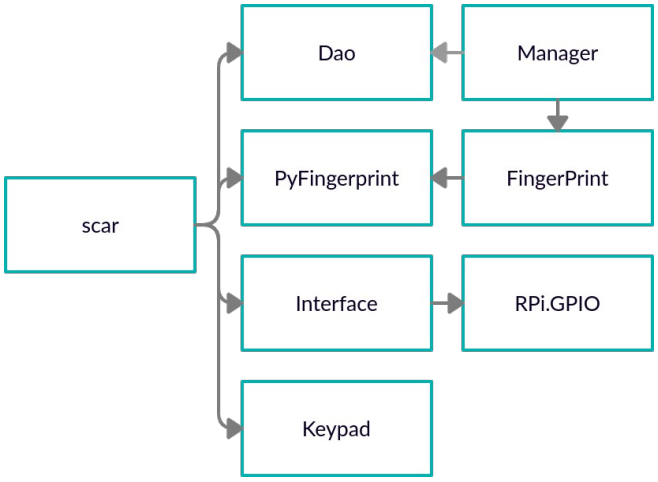

Figura 2: Diagrama de Classes

O objeto `Dao`, de *data access object*, é responsável por acessar o banco de dados. Este objeto realiza as conhecidas operações C-R-U-D, que, respectivamente, criam, leem, atualizam e apagam os dados do banco de dados.

Como o objeto `Dao` pode modificar apenas o banco de dados, há o objeto `FingerPrint`, capaz de modificar o leitor biométrico. Esse objeto encapsula operações do objeto `PyFingerprint` para salvar e apagar impressões digitais da memória do leitor biométrico. O objeto `PyFingerprint` faz parte da biblioteca fornecida por Bastian Raschke (2015) para acessar o leitor biométrico. Essas operações específicas foram encapsuladas no objeto `FingerPrint` pois o `PyFingerprint` não as possui implementadas conforme a necessidade do programa. Tanto o objeto `Dao` quanto o `FingerPrint` são usados pelo programa `Manager`.

O objeto `Keypad` representa o teclado usado para o usuário informar a senha. Cada vez que o usuário informa uma senha, esse objeto devolve a senha ao programa `scar`, que verifica com o objeto `Dao` se é uma senha válida. Caso a senha seja válida, o `Dao` devolve uma mensagem positiva ao `scar`, além de registrar o horário em que o usuário solicitou o acesso, mas caso seja uma senha inválida, devolve uma mensagem negativa.

Semelhantemente, o objeto `PyFingerprint` representa o leitor biométrico, e cada vez que o usuário informa sua digital, esse objeto devolve ao programa `scar` a posição em que a impressão digital se encontra na memória, ou uma posição inválida se a impressão digital não foi encontrada. O `scar`, por sua vez, envia ao `Dao` a posição recebida pelo `PyFingerprint`. O `Dao`, então, verifica se a posição é válida, registrando o horário de acesso ao banco e devolvendo uma mensagem positiva, mas se a posição for inválida, devolve uma mensagem negativa ao `scar`.

Tanto no caso da senha quanto no caso da impressão digital, quando o `Dao` devolve uma mensagem positiva ou negativa ao `scar`, que representa acesso concedido ou negado, o `scar` transmite essa mensagem ao objeto `Interface`. Esse objeto acessa o *led* de *feedback* com o usuário e a fechadura elétrica. Há quatro possibilidades de *feedback*, conforme a tabela a seguir:

Tabela 1: Feedback com o usuário

Acesso	Por senha	Por digital	Fechadura
Concedido	Led *acende* luz verde por dois segundos	Led *pisca* luz verde por dois segundos	É destrancada
Negado	Led *acende* luz vermelha por dois segundos	Led *pisca* luz vermelha por dois segundos	Permanece trancada

Por fim, o objeto Interface acessa o led e a fechadura através do objeto `RPi.GPIO`, disponível na biblioteca do Python, que controla as portas de entrada e saída do Raspberry Pi, que são conectadas ao led e à fechadura.

Considerações Finais

Esse projeto se propôs a fazer um sistema de segurança por meio de leitura biométrica e senha, usando um Raspberry Pi. Ao final do projeto foi desenvolvido um protótipo funcional do sistema de segurança.

Como trabalhos futuros, tem-se: criar uma senha de administrador para o programa `Manager`, pois como ficou até o momento, qualquer um conseguiria acessar o banco e apagar as informações; o `Manager` só consegue alterar informações da tabela `usuario`, pode ser interessante também permitir alterações na tabela `horario`.

Agradecimentos

Agradecemos ao Instituto Federal de Mato Grosso do Sul por nos proporcionar a satisfação de realizar esse projeto.

Referências

IEEE Spectrum. Interactive: The Top Programming Languages. IEEE Spectrum, 2020. Disponível em < https://spectrum.ieee.org/static/interactive-the-top-programming-languages-2020>. Acessado em 4 de setembro de 2020.

Okoi, Martins D. 10 Best Programming Languages for Embedded Systems. Fossmint, 23 de abril de 2019. Disponível em < https://www.fossmint.com/programming-languages-for-embedded-systems>. Acessado em 4 de setembro de 2020.

Heath, Steve. Embedded Systems Design, 2nd Edition. Newnes, outubro de 2002. p. 2.

Santos, Sara. Guide to Fingerprint Sensor Module with Arduino (FPM10A). randomnerdtutorials.com, janeiro de 2018. Disponível em https://randomnerdtutorials.com/fingerprint-sensor-module-with-arduino/. Acessado em 4 de setembro de 2020.

Raschke, Bastian. Fingerprint sensor for Raspberry Pi and Debian Linux (Update 8). sicherheitskritisch.de, 17 de março de 2015. Disponível em < https://sicherheitskritisch.de/2015/03/fingerprint-sensor-fuer-den-raspberry-pi-und-debian-linux-en/>. Acessado em 4 de setembro de 2020.

Abreu, Paulo. Controle de Fechadura Elétrica com Raspberry Pi. Github.com, novembro de 2016. Disponível em < https://github.com/abreups/fechadura_eletrica>. Acessado em 4 de setembro de 2020.

Sommerville, Ian. Software Engineering, 9th edition. Pearson, 2011. p. 32.