Análise de métodos de atualização firmware over-the-air (FOTA) para sistemas embarcados

Felipe dos Santos[1], **Otávio de Souza Martins Gomes**[2]

[1] Instituto de Engenharia de Sistemas e Tecnologia da Informação (IESTI)
Universidade Federal de Itajubá (UNIFEI) – Itajubá, MG – Brasil

{felipedsi,otavio.gomes}@unifei.edu.br

Abstract. *Embedded devices are present in different contexts and in increasing numbers. The evolution of areas such as electronics and telecommunications drive the implementation of new technologies such as over-the-air updating. Although some devices such as cell phones, for example, already have such technology, there are other devices that have restrictions and present challenges for the application of technology. This article aims to analyze over-the-air update methods, discuss memory, communication and security challenges, and propose the implementation of a bootloader software on an Espressif ESP32 development board that allows over-the-air update.*

Resumo. *Os dispositivos embarcados estão presentes em diversos contextos e em quantidade crescente. A evolução de áreas como eletrônica e telecomunicações impulsionam a implementação de novas tecnologias como a atualização over-the-air. Embora alguns dispositivos como celulares, por exemplo, já apresentam tal tecnologia, existem outros dispositivos que possuem restrições e apresentam desafios para a aplicação da tecnologia. Este artigo tem como objetivo analisar métodos de atualização over-the-air, discutir desafios de memória, comunicação e segurança, e propor a implementação de um software bootloader em uma placa de desenvolvimento Espressif ESP32 que permita a atualização over-the-air.*

1. Introdução

Os computadores embarcados são computadores embutidos em dispositivos que não são vendidos como computadores. Os computadores embarcados, às vezes chamados de microcontroladores ou ECUs *(Eletronic Control Unit)*, gerenciam os dispositivos e controlam a interface do usuário. Os microcontroladores são encontrados em uma grande variedade de dispositivos diferentes como dispositivos médicos, brinquedos, eletrodomésticos, veículos etc. Sistemas embarcados, muitas vezes, têm restrições físicas em termos de tamanho, peso, consumo de bateria e outros limites elétricos e mecânicos [Tanenbaum 2016]. Assim como computadores convencionais, esses dispositivos possuem um software, comumente chamado de firmware, e podem necessitar de atualizações, seja para adicionar novas funcionalidades ou corrigir possíveis erros de funcionamento. As atualizações podem ser realizadas por meios guiados *(Ethernet, JTAG)* ou por meio não guiados *(WiFi, Bluetooth)*. Convencionalmente, FOTA (*Firmware Over-The-Air*) ou OTA (*Over-The-Air*), refere-se ao processo de atualização do firmware do dispositivo através de um meio de comunicação sem fio. O firmware é pré-carregado nos dispositivos

remotos ou na rede dos sensores para dispositivos que podem ocasionalmente precisar ser atualizados por diversas razões, incluindo correções de bugs pós-implantação, segurança, patches, introdução de novos recursos e melhorias de desempenho. Com isso, OTA tem sido considerado mais apropriado, especialmente quando se lida com implantação de redes de sensores e dispositivos de ponta inacessíveis fisicamente e em grande escala. Cenários onde os métodos manuais seriam muito caros ou apresentariam dificuldades físicas e/ou temporais [Pule and Abu-Mahfouz 2019].

A atualização over-the-air pode ser vista como um método para programar código de computador em uma plataforma remota por meio de um dispositivo local. O método inclui receber o código de computador de tal modo que o código seja dividido em pacotes e são fornecidos por transmissão sem fio. O próximo passo envolve armazenar os pacotes de código em uma área alternativa de memória flash do dispositivo local de modo que os pacotes armazenados representem um código de computador. O dispositivo local então reconhece a recepção da cópia completa do código de computador e verifica a integridade do código recebido [Quadri and Sidek 2014].

2. Desafios

Como os dispositivos IoT serão sempre projetados para serem simples e acessíveis de forma a manter a sua tradicional forma de fator pequena, eles são inerentemente limitados por recursos em termos de poder computacional, energia/duração da bateria, comunicação, largura de banda e memória [Pule and Abu-Mahfouz 2019].

Com relação aos dispositivos embarcados, [Sen 2010] categorizou as principais restrições de dispositivos IoT em restrições de energia, limitações de memória, comunicação não confiável, alta latência de comunicação e proteção física do dispositivo. Com relação ao consumo de energia, é possível separar a energia em três campos chave, sendo a energia utilizada pelo sensor transdutor, energia uti- lizada para comunicação e energia utilizada pelo microprocessador para executar instruções. Além disso, a fatores como segurança e latência estão atrelados ao consumo energético. Algoritmos de criptografia mais robustos exigem maior capacidade de processamento, consequentemente afetam o consumo energético. A proteção física do dispositivo é um grande desafio, visto que de acordo com a aplicação, o dispositivo pode ser instalado em uma região onde não é possível realizar o monitoramento físico do dispositivo, permitindo a prática de ações maliciosas como o tampering físico.

Para o processo de atualização OTA, [Brown 2018] categorizou três desafios chave. O primeiro desafio diz respeito à memória. A solução de software deve organizar o novo aplicativo de software na memória volátil ou não volátil do cliente dispositivo para que possa ser executado quando o processo de atualização for concluído. A solução deve garantir que uma versão anterior do software seja mantida como um aplicativo alternativo caso o novo software tenha problemas. Também é necessário manter o estado do dispositivo cliente entre reinicializações e ciclos de energia, como a versão do software que está em execução no momento e onde ele está na memória. O segundo grande desafio é a comunicação. O novo software deve ser enviado do servidor para o cliente em pacotes discretos, cada um visando um endereço específico na memória do cliente. O esquema para empacotamento, a estrutura do pacote e o protocolo usado para transferir todos os dados devem ser considerados no design do software. O último major desafio

é a segurança. Com o novo software sendo enviado sem fio de o servidor para o cliente, deve-se garantir que o servidor seja uma parte confiável. Esse desafio de segurança é conhecido como autenticação. Também é necessário garantir que o novo software seja ofuscado para qualquer observador, pois pode conter informação sensível. Esse desafio de segurança é conhecido como confidencialidade. O elemento final da segurança é a integridade, garantindo que o novo software não seja corrompido quando for enviado pelo ar.

Além dos desafios do processo de atualização no geral, existem outros desafios particulares para determinadas aplicações. No ramo automotivo, Ken Tindell publicou em seu blog [Tindell] alguns problemas que podem ocorrer caso o método over-the-air seja adotado na indústria. O primeiro problema é a falha catastrófica, isto é, caso uma atualização possua uma falha, isso seria replicado em milhões de veículos nas ruas. A liberação de atualização por partes pode resolver esse problema. O segundo problema também cita uma falha, porém no processo de atualização, o que pode necessitar que o carro seja atualizado fisicamente para corrigir o problema. O terceiro e quarto problema citados por ele está relacionado a segurança da infraestrutura e a segurança do processo de gravação do firmware no dispositivo. A segurança está sendo abordada em parte pela indústria automotiva, por meio de um módulo de segurança de hardware (HSM). A indústria definiu o padrão Secure Hardware Extensions (SHE) e os chips projetados para aplicações automotivas incluem esses módulos. Eles fornecem não apenas funções criptográficas (criptografia, autenticação, números aleatórios), mas também distribuição e armazenamento seguros de chaves. Para veículos rodoviários, a ISO 26262 rege as regras para o funcionamento seguro dos veículos relacionado às partes elétricas e eletrônicas.

3. Objetivos

Muitos sistemas embarcados são implantados em locais difíceis para um operador humano acessar. Isso é especialmente verdadeiro para aplicações de Internet das Coisas (IoT), que normalmente são implantados em grandes quantidades e com vida útil limitada da bateria. Alguns exemplos seriam sistemas embutidos que monitoram a saúde de uma pessoa ou de uma máquina. Esses desafios, juntamente com o rápido ciclo de vida do software, fazem com que muitos sistemas exijam suporte para atualizações over-the-air (OTA) [Brown 2018]. Sendo assim, o artigo em questão tem o objetivo de levantar informações e discutir diferentes abordagens do processo de atualização over-the-air.

A modificação do formato do arquivo a ser enviado, modelo de comunicação e características do bootloader será realizada a fim de obter dados de desempenho e efetuar comparações em tabelas, para posteriormente validar prós e contras das diferentes abordagens implementadas. A utilização de um Sistema Operacional de Tempo Real também está dentro das possibilidades de implementação. É importante que o bootlolader a ser desenvolvido apresente formas de mitigar problemas que possam ocorrer durante o processo de atualização de dispositivos. Alguns problemas incluem falta de energia durante o processo, arquivo de atualização corrompido, dentre outros. A abordagem a ser utilizada que garante que, caso o sistema venha a falhar durante esse processo, o dispositivo conseguirá se recuperar inclui a abordagem de um bootloader de dois estágios.

O bootloader de inicialização principal é um aplicativo de software que reside permanentemente no microcontrolador na memória somente leitura. A região da memória

que o bootloader primário reside é conhecido como espaço de informações e ,às vezes, não é acessível aos usuários. Este aplicativo é executado toda vez que uma redefinição ocorre, geralmente executando algumas inicializações de hardware essenciais, e pode carregar o software do usuário na memória. No entanto, se o microcontrolador contém memória não volátil no chip, como memória flash, o bootloader não precisa fazer nenhum carregamento e simplesmente transfere o controle para o programa na memória flash. Se o carregador de inicialização principal não tiver nenhum suporte para atualizações OTA, é necessário ter uma inicialização do segundo estágio do bootloader. Como o bootloader primário, o segundo estágio será executado toda vez que um reset ocorra, mas implementará uma parte do processo de atualização OTA. A abordagem é ilustrada na Figura 1 [Brown 2018].

Ao final do trabalho, espera-se obter dados que possam auxiliar no design de bootloaders para sistemas embarcados de acordo com a aplicação e requisitos, e também fornecer a implementação de um modelo de bootloader funcional e versátil.

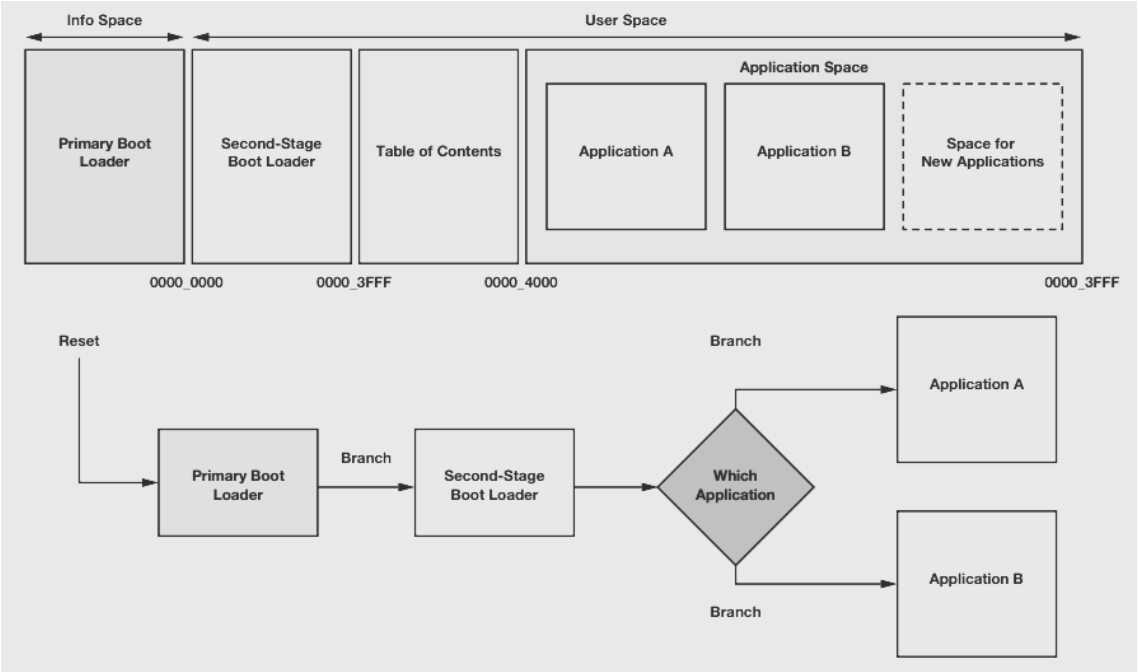

Figura 1. Mapa de memória e fluxo de inicialização com bootloader de dois estágios

4. Metodologia

As pesquisas bibliográficas são utilizadas para verificar o estado da arte da tecnologia na comunidade acadêmica, assim como fornecer aporte teórico para diferentes abordagens a serem implementadas. A implementação dos softwares de sistema serão realizadas em uma placa de desenvolvimento com o microcontrolador ESP32 da Espressif.

Conforme citado por [Beningo 2015], cada bootloader tem seu próprio conjunto exclusivo de requisitos com base no tipo de aplicação, no entanto, ainda existem alguns requisitos gerais comuns a todos os bootloaders. Esses requisitos podem ser divididos em sete grupos fundamentais de requisitos que são comuns ao bootloader.

1. Capacidade de alternar ou selecionar o modo de operação (aplicativo ou bootloader)

2. Requisitos de interface de comunicação (USB, CAN, I2C, USART etc)
3. Requisito de análise de registro (S-Record, hex, intel, toeff etc)
4. Requisitos do sistema Flash (apagar, gravar, ler, localizar)
5. Requisitos de EEPROM (particionar, apagar, ler, escrever)
6. Soma de verificação do aplicativo (verificar se o aplicativo não está corrompido)
7. Segurança do código (protegendo o carregador de inicialização e o aplicativo

Pretende-se concluir a implementação do bootloader com base nos requisitos citados acima. Alguns dos requisitos como, por exemplo, segurança do código, podem apresentar diferentes abordagens, o que será utilizado para realizar comparações com diferentes métodos de garantir a segurança do código e como isso impacta no desempenho. Outro ponto a ser explorado é como os dados serão enviados ao microcontrolador, com relação ao protocolo de comunicação e no formato dos dados a serem enviados. A idéia é garantir um equilíbrio entre desempenho, consumo energético e utilização de memória.

5. Conclusão

Projetar uma porção de código responsável por carregar o software de inicialização e, caso necessário, atualizá-lo com mecanismos de software para mitigar problemas que possam inviabilizar o dispositivo, envolve diversos desafios e *trade-off*. Algumas aplicações como, por exemplo, automóveis e dispositivos médicos, podem necessitar de grande alocação de recursos na segurança. Para sensores em uma rede IoT, o consumo energético pode ser crucial. A atualização over-the-air é uma tecnologia que foi possibilitada devido à evolução tecnológica e está presente em diversos dispositivos como celulares, televisores, dentre outros. Espera-se que este trabalho proporcione questões teóricas e práticas que colaborem para que a tecnologia over-the-air seja adotada em outras áreas e tipos de dispositivos.

Referências

Beningo, J. (2015). Bootloader design for microcontrollers in embedded systems. *Embedded Software Design Techniques*.

Brown, B. B. (2018). Over-the-air (ota) updates in embedded microcontroller applications: Design trade-offs and lessons learned. *Analog Dialogue Technical Journal*, 52:52–11.

Pule, M. and Abu-Mahfouz, A. M. (2019). Firmware updates over the air mechanisms for low power wide area networks: A review. In *2019 International Multidisciplinary Information Technology and Engineering Conference (IMITEC)*, pages 1–7. IEEE.

Quadri, A. and Sidek, B. O. (2014). An introduction to over-the-air programming in wireless sensor networks. *Int J Comput Sci Netw Solut [Internet]*, 2:33–49.

Sen, J. (2010). A survey on wireless sensor network security. *arXiv preprint arXiv:1011.1529*.

Tanenbaum, A. S. (2016). *Structured computer organization*. Pearson Education India.

Tindell, K. Digital health, medical devices and ota software updates.