Sistema Iot para Alerta de Emergência em Ambientes Internos

Francisco Laurindo Costa Júnior, Filipe A. Lira, Jessé P. Coutinho, Erik J. F. Nascimento, Josué B. Mota, Gilmar P. B. Bezerra, Sandro S. C. Juca

Eixo Tecnológico Da Computação¸ Instituto Federal do Ceará (IFCE) – Campus Maracanaú
Av. Parque Central – 61919-140 – Maracanaú – CE – Brazil

laucostajr@hotmail.com, filipe.al2015@gmail.com, jessecdm@gmail.com, erikjohnesf@gmail.com, josuebatistam1@gmail.com, gilmardepontes@hotmail.com, sandro.juca@gmail.com

Abstract. *The objective of this article is to describe the construction and operation of an emergency warnings prototype on dangerous situations. The device sends a predefined message to Telegram, containing the address and a message: -Emergency Alert Activated. The prototype contains a bluetooth low energy(BLE) tracker that detects the proximity of one or more people using any bluetooth device that was previously registered on the system. The MQTT server registers each event every time that the button is pressed. The economic energy mode is deactivated when the button is pressed and soon after sending the emergency warning, it returns to deep sleep mode. The system was implemented in a low cost microcontroller, the ESP32.*

Resumo. *O objetivo deste artigo é relatar a construção e funcionamento de um protótipo para alertas de emergência em situações de risco. O dispositivo envia uma mensagem predefinida para o Telegram, contendo o endereço e uma mensagem: - Alerta de Emergência Ativado. O protótipo contém um rastreador bluetooth low energy (BLE) que detecta a proximidade de uma ou mais pessoas usando qualquer dispositivo bluetooth, previamente cadastrado no sistema. O servidor MQTT (Message Queuing Telemetry Transport) registra cada evento cada vez que o botão é pressionado. O modo de economia de energia é desativado quando o botão é pressionado e logo depois de enviar o alerta de emergência, retorna ao modo de deepsleep. O sistema foi implementado em um microcontrolador de baixo custo, o ESP32.*

1. Introdução

Um sistema de alertas de emergências, viável e de baixo custo, pode aumentar a segurança das pessoas. Um ponto importante que justifica a criação de um sistema IoT (*Internet of Things)* de baixo custo é a disseminação da tecnologia ao alcance e benefício de todos. Crianças, adultos e idosos facilmente podem entender o funcionamento do sistema, o pressionar de um botão pode ser simples, mas possui grande relevância em inúmeras situações de risco.

Os acidentes domésticos estão intimamente relacionados com o comportamento da família e rede social, com o estilo de vida, com fatores educacionais, econômicos, sociais e culturais. Dentre as pessoas em situação de vulnerabilidade, as crianças por estarem na fase de desenvolvimento, caracterizada pela curiosidade aguçada e contínuo aprendizado, são vítimas dessas tentativas de descobertas, por isso que. na faixa etária

de um a cinco anos, os principais casos ocorridos no domicílio são representados pelas quedas, queimaduras, aspirações ou introduções de corpos estranhos e intoxicações exógenas [Souza, L. J. E. X, Barroso, M.G.T. 1998]. Com a implementação das novas tecnologias, parentes, amigos e vizinhos poderão receber mensagens de alerta sobre um determinado amigo ou parente via Telegram.

A tecnologia *bluetooth low energy* (BLE) integrada ao ESP32 foi aproveitada para implementar um rastreador de raio curto, de aproximadamente dez metros, com o objetivo de identificar pessoas que possuam qualquer dispositivo BLE registrado previamente no sistema, guardando informações sobre o números de pessoas naquele local, auxiliando autoridades quando necessário. O sistema possui o auxilio do modo econômica para usar somente a energia necessário para solicitar ajuda, podendo usar duas pilhas comuns e, ainda assim, durar até 2 meses de funcionamento.

2. Fundamentação Teórica

Este capítulo contém uma breve introdução aos conceitos e tecnologias importantes para o desenvolvimento do protótipo. Ao longo do capítulo serão apresentados os principais conceitos dentro do âmbito dos sistemas que auxiliam o homem em situações de risco. Será explicado a comunicação do ESP32 com o servidor MQTT, *bluetooth* e o *bot* no aplicativo Telegram, assim como o desenvolvimento do software e o modo econômico *DeepSleep*.

Baseado nos projetos citados na Tabela 1, foi desenvolvido um sistema para emitir avisos de emergência para um atendimento rápido a qualquer individuo.

Referência	Microcontrolador	Dados
[Neves, L. A. P. 2019]	ESP32	RFID+PHP
[Araújo, P. H. M 2019]	ESP8266	MQTT+RFID
[Palani, S. 2018]	ESP32	Bluetooth
[Ringe, A. 2019]	Raspberry Py	MQTT
[Taştan, M. 2018]	ESP8266	DHT22+Blink

O sistema de alerta de emergência desenvolvido possui três características principais: o algoritmo de acionamento do botão, o *DeepSleep*, busca usuários com dispositivos *bluetooth* registrados *no sistema* e a comunicação com o servidor MQTT. É o servidor MQTT que registra todos os eventos e o envio da mensagem de socorro para o grupo do aplicativo Telegram, através de um *bot* criado no próprio Telegram. Em aplicações envolvendo a Internet das Coisas, o MQTT é um dos protocolos mais utilizados devido a sua definição de qualidade de serviço, especificações de segurança, implementação simples e garantia de utilização da banda de uso moderada [Lampkin, V. 2012]. O envio de mensagem do ESP32 para Telegram está associado ao servidor MQTT. Este servidor armazena e envia o evento que possui protocolo HTTP (HyperText Transfer Protocol) , que transfere a informação ao Telegram. O *DeepSleep* é um modo que mantém o ESP32 com o mínimo de consumo de energia possível para seu funcionamento.

Pode-se entender automação pela “substituição do trabalho do homem, manual, por sistemas previamente programados que se auto-controlam, regulam e realizam uma série de operações em velocidade superior à capacidade humana” [Blogdacurto 2018].

Estima-se que até o final de 2020 existam mais de 212 bilhões de dispositivos IoT espalhados pelo mundo [Al-Fuqaha, A. 2015]. Mais além, em 2022, a previsão é de que 45% de todo o tráfego da Internet seja constituído por comunicações *Machine-to-Machine* (M2M) [Xiao, Y. 2016].

O MQTT foi criado para ser um protocolo de mensagem usado sobre o protocolo TCP/IP (Transmission Control Protocol) e baseia-se em um esquema de *subscriber/publisher* [Michael Yuan. 2017]. É executado na camada de aplicação (TCP/ IP). As mensagens MQTT são trocadas entre um cliente, que pode ser um publicador ou assinante (*publisher/subscriber)* de mensagens e um intermediário (*Broker)* de mensagens (por exemplo, Mosquitto MQTT).

Os clientes do Telegram possuem código aberto, porém seus servidores são proprietários. O serviço também providencia APIs (*Application* Programming Interface) para desenvolvedores independentes, ou seja, possui ferramentas que facilitam a construção de sistemas com IoT, possibilitando um controle por troca de mensagens entre plataformas móveis, como *tablets* e/ou *smartphones*, e plataformas microcontroladas [TELEGRAM Home Page. 2019].

Em [Souza, D. José. 2005] um microcontrolador é definido como um componente eletrônico de pequeno porte que possui inteligência programável para o controle de processos lógicos [Cortelleti, Daniel. 2006]. O ESP32 possui 18 entradas analógicas para conversão digital, as quais fornecem uma resolução de 12 bits na escala de 0 a 3,3V. Estas entradas serão utilizadas para a aquisição dos dados dos sensores que vem em um formato analógico, convertendo-os para uma escala digital entre 0 e 4095 (12 bits) para 32 posteriormente serem manipulados no software [Intermec, T. C. 2007].

Para programar o ESP32, pode-se utilizar o IDE (Ambiente de Desenvolvimento Integrado) do Arduíno, um software livre no qual se programa na linguagem C. O IDE é instalado em um PC com plataforma Windows ou Linux, ele permite que você escreva instruções para controlar as portas do ESP32, das quais você faz o *upload* para o *chip* [Neves, L. A. P. 2019].

3. Materiais e Métodos

Para que o sistema fosse desenvolvido foi necessário fragmentá-lo em três etapas distintas: (I) Manipulação de Hardware, onde ocorre a montagem, desenvolvimento e estruturação física de todo o circuito eletrônico para seu funcionamento pleno; (II) Manipulação lógica, a implementação do código de programação na linguagem C com a IDE do Arduíno; (III) Integração, responsável por acoplar as duas etapas anteriores em um protótipo funcional.

Para desenvolver (I) foram utilizados: *protoboard* para realizar organização inicial do circuito; LED verde (L1) para indicar o status na conexão com a Internet; LED amarelo (L2) para indicar o *status* na conexão com o servidor MQTT; LED branco (L3) para indicar o *status* do envio da mensagem ao aplicativo de mensagens Telegram; uma chave *push* (C1) junto com um resistor igual ou inferior a 390 kΩ (R1) para acionamento do envio da mensagem ao Telegram ao ser pressionada.

Em (II), o desenvolvimento é realizado com a linguagem de programação C, usando a IDE do Arduíno configurada para o dispositivo ESP32. Esta é a etapa onde as requisições MQTT e HTTP são feitas e organizadas para que todo o processo de envio de mensagem possa fluir normalmente. A função de enviar a mensagem ao Telegram atribuída ao pressionar a chave *push*, utiliza protocolo HTTP para comunicar-se ao Telegram. A função do MQTT é registrar cada evento e logo após enviar a mensagem de alerta de emergência ao Telegram. O *bot* responsável por enviar mensagens ao grupo foi configurado a partir do próprio aplicativo Telegram que disponibilizou o *token* para requisição HTTP junto com o *chatID*, que são necessários para a comunicação plena. A função de escanear um ou mais dispositivos *bluetooth* (BLE) verifica se os dispositivos próximos são dispositivos registrados no sistema. Assim, ajudando a identificar usuários quando for acionado o alerta.

Em (III), a integração ocorreu através da compilação do código desenvolvido na plataforma de desenvolvimento Arduíno. Foi necessário suprimir a função do microcontrolador ESP32 OTA, que realiza atualizações, já que essa função não foi necessária para o funcionamento do projeto em questão.

O sistema foi montado em uma *protoboard*, onde foram conectados os componentes: LED verde; LED amarelo; LED branco; chave *push* e um resistor. A saída GPIO 22 foi usada para atuar no LED verde que pode piscar, indicando a tentativa do módulo de conectar-se a internet. LED aceso representa que esta conectado a internet. A saída GPIO 21 foi usada para atuar no LED amarelo que pode piscar, indicando que o módulo está tentando conectar-se a internet, e acender, representando que houve conexão ao MQTT. A entrada GPIO 33 foi usada para conectar a chave *push.* Ela acorda o dispositivo para realizar sua tarefa e logo depois volta ao *DeeSleep*. A saída GPIO 4 atua no LED branco, que será aceso quando a mensagem for enviada. O fluxograma representado na Figura 1 mostra todo o funcionamento do sistema.

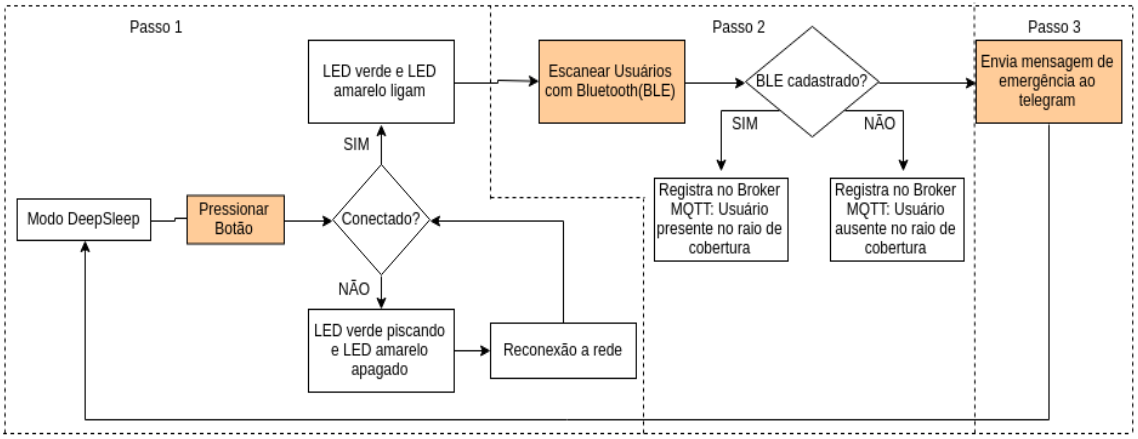

Figura 1. Fluxograma

4. Resultados e Discussões

O protótipo construído teve a finalidade de validar as simulações, diagramas de circuitos e lógica de algoritmos desenvolvidos para cumprir os requisitos do projeto. A montagem ficou simples com a inserção dos componentes em uma *protoboard*. A Figura 2 ilustra o protótipo idealizado.

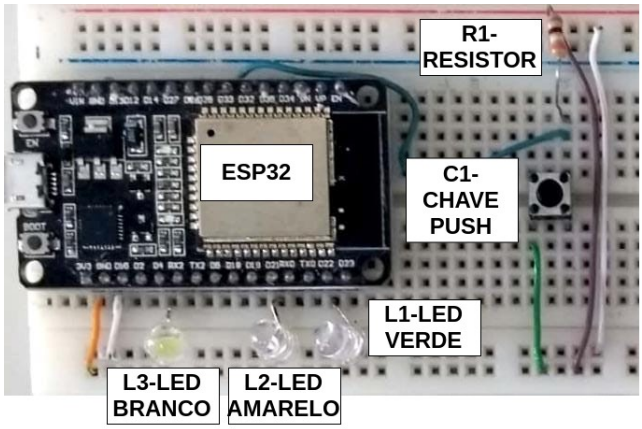

Figura 2. Projeto idealizado

Durante a etapa de teste foi observado que a função de economia de bateria funcionou com êxito, dando uma autonomia de 14 dias com uma bateria de 200mah. O envio de mensagens não ocorreu como previsto inicialmente. Depois de mais de 30 horas ligado e algumas oscilações na rede, o dispositivo não realizava mais o envio de mensagens. Este problema foi corrigido com a inclusão do complemento das funções de conexão que restabelecem as conexões sempre que a rede não responder. Não houve mais falhas após as correções. O protótipo ficou eficiente demonstrando uma resposta aproximadamente de quatro segundos para cumprir todo processo e voltar ao estado inicial em uma rede estável como foi ilustrado no diagrama da Figura 1. A função de rastreamento *bluetooth* funcionou bem, mostrou-se relevante para identificar pessoas em um ambiente onde esteja o protótipo com raio de 10 metros. O tempo de envio da mensagem de emergência pode variar se a rede local não for estável ou o sinal de internet estiver com atenuações que prejudiquem o envio de dados.

5. Considerações Finais

Este trabalho abordou a problemática da assistência a pessoas em situações de risco em ambientes *internos*, levando em consideração inúmeros fatores de risco que podem ocorrer dentro do ambiente doméstico.

Os resultados derivados da aplicação dos testes orientado ao usuário usando o sistema de alerta de emergência, conseguiu diminuir ou inibir as consequências danosas quando solicitado ajuda pelo dispositivo. Dessa forma, foi possível fazer inferências de situações de risco. No entanto, algumas variáveis podem causar dificuldades para pressionar um botão, como por exemplo: quedas que causem desmaios ou ambiente fragmentado devido a uma causa natural. Essas variáveis indicam necessidades de melhorias futuras. Desta forma abre espaço para novas aplicações e outros avanços.

Referências

Souza, L. J. E. X, Barroso, M.G.T. (1998) “Envenenar é mais perigoso: uma abordagem etnográfica.”,Cogitare Enfermagem.

Lampkin, V. (2012) “Building smarter planet solutions with mqtt and ibm websphere mq telemetry”, IBM Redbooks.

Araújo, P. H. M., Jucá, S. C. S., Lima, D., Gonçalves, C., da Silva, V. F., Soares, R. I., & Pereira, S. A. D. S. (2019) “SCloud-based RFID access control using lightweight

messaging protocol", IJAERS.

Cortelleti, Daniel. (2006) "Dossiê Técnico: Introdução à programação de microcontroladores Microchip PIC.", Centro Tecnológico de Mecatrônica, SENAI-RS .

Souza, D. José. (2005), Desbravando o PIC: Ampliado e Atualizado para PIC 16F628A, 8th ed, Érica.

Michael Yuan. (2017), Getting to know MQTT.

Darnell, L. (2015). The Internet of Things: A Look at Real-World Use Cases and Concerns, Kindle Edition.

M.A.E. Mowad, A. Fathy and A. Hafez. (2014) "International Journal of Scientific & Engineering Research", vol. 5, Issue. 5, pp 935-939, International Journal of Scientific & Engineering Research.

Intermec, T. C. (2007) "Supply Chain RFID: How It Works and Why It Pays.",Intermec, USA

Neves, L. A. P. (2019) "Smart door: gerenciamento e acesso remoto de portas.", UFP, Monografia.

Blogdacurto (2018) "Conhecendo o ESP32",curtocircuito.com.br.

Palani, S. (2018) ""Who's There?": Designing Sensor-Aided Wearable Assistive Technology for the Visually-Impaired, dissertação doutorado, University of California, San Diego.

Ringe, A., Dalavi, M., Kabugade, S., & Mane, P. P. (2019) "IoT Based Smart Refrigerator Using Raspberry Pi.", Vol,6, Issue 2, IJRAR.

Taştan, M., & Gökozan, H. (2018), An Internet of Things Based Air Conditioning and Lighting Control System for Smart Home, cientific Research Journal for Engineering, Technology, and Sciences (ASRJETS), 50(1), 181-189..

Al-Fuqaha, A. (2015). Internet of things: A survey on enabling technologies, protocols, and applications. Communications Surveys & Tutorials, IEEE, IEEE, USA, v. 17, n. 4, p. 2347–2376.

Xiao, Y. (2019) "Full-duplex machine-to-machine communication for wireless-powered internet-of-things.", IEEE, 2016. p. 1–6..

Palani, S. (2018) ""Who's There?": Designing Sensor-Aided Wearable Assistive Technology for the Visually-Impaired, dissertação doutorado, University of California, San Diego.

Ringe, A., Dalavi, M., Kabugade, S., & Mane, P. P. (2019) "IoT Based Smart Refrigerator Using Raspberry Pi.", Vol,6, Issue 2, IJRAR.

TELEGRAM Home Page.(2019) Telegram: a new era of messaging.