|  |  |
| --- | --- |
|  |  |
| EmbarcaTech  Projeto Final | C:\Users\jvpdo\Downloads\LOGO_JVPO.png |
|  |  |
|  |  |
|  | João Victor Pomiglio de Oliveira |

Sumário

[Item 1 3](#_Toc189154147)

[Item 2 3](#_Toc189154148)

[Item 3 3](#_Toc189154149)

[Item 4 4](#_Toc189154150)

[Item 5 6](#_Toc189154151)

[Item 6 7](#_Toc189154152)

# a) Escopo do projeto (2 pontos no total)

* Apresentação do projeto – breve descrição sobre o que trata o projeto. (0,5 ponto)

O projeto visa desenvolver um dispositivo vestível baseado em IoT para monitorar níveis de ruído em tempo real em ambientes de construção civil, alertando trabalhadores e equipes de segurança quando os limites de exposição segura são ultrapassados.

* Título do projeto – pequeno título para o projeto.

SISMA - Sistema de Monitoramento de Intensidade Sonora

* Objetivos do projeto – descreva os objetivos do projeto. (0,25 ponto)

Monitorar níveis de ruído em tempo real.

Alertar trabalhadores com sinais visuais e sonoros.

Armazenar dados para análise pós-exposição.

Calcular tempo seguro de exposição conforme normas NIOSH/OSHA.

* Principais requisitos – podem ser estipulados pelo usuário ou cliente, ou até por você (fazendo o papel de um deles). (0,25 ponto)

Sensor de áudio com saída analógica/digital.

Microcontrolador com capacidade de processamento e comunicação IoT.

Display OLED para interface visual.

Alerta sonoro (buzzer) e visual (LEDs).

Armazenamento de dados em cartão SD.

* Descrição do funcionamento – descreva as funcionalidades do projeto. (0,25 ponto)

Leitura contínua de ruído ambiente.

Conversão de sinais analógicos para dB.

Acionamento de alarmes ao ultrapassar limites pré-definidos (85 dB, 88 dB, etc.).

Exibição de dados em tempo real (nível de dB, tempo de exposição).

Registro de dados para análise offline.

* Justificativa – mostre que a execução do projeto se justifica. (0,25 ponto)

A perda auditiva é uma das doenças ocupacionais mais comuns na construção civil. Soluções existentes não integram IoT para monitoramento em tempo real e análise de dados, tornando este projeto relevante para prevenção.

* Originalidade – mostre através de uma pesquisa que existem projetos correlatos, mas não iguais. (0,5 ponto)

Embora existam medidores de ruído convencionais, a integração com IoT, alertas personalizados por faixa de dB e cálculo automático de tempo seguro de exposição são diferenciais. Projetos correlatos focam em sensores básicos, sem análise contextual (ex.: Abbasianjahromi et al., 2021).

# b) Hardware (2,5 pontos no total)

* Diagrama em blocos – diagrama mostrando os blocos e sua interligação. (0,25 ponto)
* Função de cada bloco – descreva a função que cada bloco terá no projeto.  
  (0,25 ponto)

Sensor de Áudio (KY-037): Captura sinais sonoros e converte para tensão.

ADC: Converte sinais analógicos do sensor para digital.

Microcontrolador (RP2040): Processa dados, gerencia alarmes e comunicação.

Display OLED: Exibe dB, tempo de exposição e alertas.

Buzzer/LEDs: Alertas sonoros (100 dB+) e visuais (LED vermelho para >85 dB).

Cartão SD: Armazena dados para análise posterior.

* Configuração de cada bloco – descreva a configuração usada em cada bloco.  
  (0,25 ponto)
* Especificações – descreva como as especificações técnicas atendem os requisitos  do cliente ou usuário. (0,5 ponto)
* Lista de materiais – incluindo descrição e quantidade. (0,25 ponto)

Lista de Materiais:

Componente Quantidade

Raspberry Pi Pico 1

Sensor KY-037 1

Display OLED 1

Buzzer 1

LEDs (RGB) 3

Cartão SD 1

* Descrição da pinagem usada – quais pinos do hardware foram usados e sua função. Você pode usar uma tabela para indicar como os pinos do microcontrolador se conectam com os periféricos. OBS: Apesar de você estar usando a BitDogLab e estas conexões já estarem estabelecidas, monte um mapa do seu projeto só com aquilo que está usando. (0,5 ponto)

Pinagem:

Pino RP2040 Função

GPIO28 Entrada ADC (MIC)

GPIO14 I2C SDA (OLED)

GPIO15 I2C SCL (OLED)

GPIO21 PWM Buzzer

GPIO13 LED Vermelho

GPIO11 LED Verde

* Circuito completo do hardware – faça o desenho do circuito completo do hardware. (0,5 ponto)

# c) Software (2,5 pontos no total, sendo 0,25 cada item)

* Blocos funcionais – mostre um diagrama das camadas do software e suas funções.
* Descrição das funcionalidades – breve descrição das funções dos blocos de software.
* Definição das variáveis – descreva as principais variáveis usadas.
* Fluxograma – desenhe o fluxograma completo do software.
* Inicialização – descreva o processo de inicialização do software.
* Configurações dos registros – descreva as funções de configuração dos registros.
* Estrutura e formato dos dados – descreva os dados específicos usados no seu software.
* Organização da memória – descrição dos endereços de memória que você usou.
* Protocolo de comunicação – descreva o protocolo, se existir.
* Formato do pacote de dados – descreva a formação dos pacotes, se existir.

# d) Execução do projeto (3 pontos no total)

* Metodologia – descrição da execução das etapas do projeto: pesquisas realizadas, escolha do hardware, definição das funcionalidades do software, inicialização da IDE, programação na IDE, depuração. (1 ponto)
* Testes de validação – descreva os testes realizados para validação do funcionamento.  (0,5 ponto)
* Discussão dos Resultados – analise os resultados e conclua sobre a confiabilidade e aplicabilidade do projeto. (0,5 ponto)
* Faça um vídeo de no máximo 3 minutos mostrando seu projeto funcionando. Inclua o link do vídeo no seu relatório de entrega. (1 ponto)
  + O vídeo deve ser enviado na forma de um link do YouTube. Se o aluno desejar que o vídeo não seja público, pode ser um vídeo **não listado**.

**Dicas:**

O uso de recursos avançados como PIOs e DMA valorizam o seu projeto.

Priorize utilizar o hardware da BitDogLab, pois a avaliação será focada nisto.  Tome cuidado com a inclusão de componentes externos, não complique! Na segunda fase, vamos explorar além!

**Atenção: Se você não receber a sua placa até 10 de fevereiro**, faça uma declaração de que não a recebeu e inclua no seu relatório. Além disso, envie a mesma declaração para o e-mail: [capacitacao.hbr@hardware.org.br](mailto:capacitacao.hbr@hardware.org.br). Desta forma, você poderá entregar a parte prática do seu projeto usando o simulador Wokwi. Neste caso, não haverá prejuizo na sua nota.

# e) Referências

* Enumere as referências científicas e material consultado usados, por você, durante a execução do projeto.

# Item 1

Elabore um programa para acionar um LED quando o botão A for pressionado 5 vezes, utilizando o temporizador como contador. Quando o valor da contagem atingir 5 vezes, um LED deve ser piscar por 10 segundos na frequência de 10 Hz.

Para resolver a questão utilizei o exemplo de botão com interrupção aprensentado nos materias de estudo como base, onde implementei a função **blink\_led** para para fazer o led piscar e uma função **for** reponsavel pela excução da função o numero de vezes necessário. Para facilitar a operação tame foi inserido as constantes **hz** e **time\_led\_ms** para configurar o modo de operação do led. Por último foram implementadas mesagens de depuração no console para averiguar o funcionamento correto do código.

<https://wokwi.com/projects/419977229297470465>

# Item 2

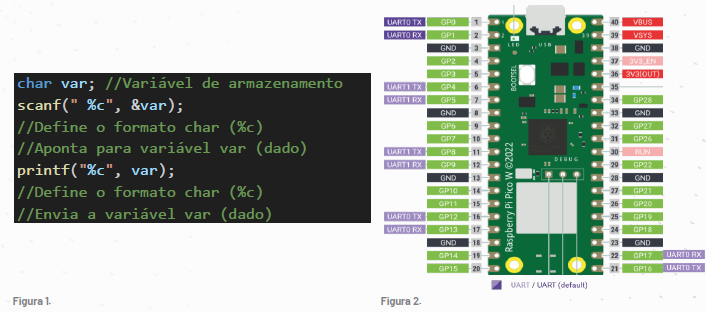
Na questão anterior, implemente o botão B, para mudar a frequência do LED para 1 Hz.

Foi reutilizado o código do item 1 e implementado as configurações básicas de um novo botão B para o controle da frequência do LED. Para aplicar a lógica de funcionamento foi inserido função **if** dentro do callback do timer o qual é responsavel por alternar o valor do **hz** (frequência do LED) entre 10 Hz e 1 Hz.

<https://wokwi.com/projects/420009794510920705>

# Item 3

Elabore um código utilizando a interfaces UART0 e conecte os fios TX e RX atribuídos à essa interface entre. Essa estrutura envia dados e recebe os dados na mesma interface, apenas para verificar seu funcionamento. Utilize a função scanf da biblioteca stdio para enviar via console um dado à placa, em seguida, transmita da UART0 para a UART1, e por fim, transmita o dado recebido para o console utilizando o printf.



O pinos tx e rx de cada UART forma definidos e configurados pela função **config\_uart.** O programa inicia solicitando uma entrada no console de no maximo 99 caracteres para ser enviado da **UART0** para **UART1**, chama a função **clear\_stdin** que limpa o buffer do console para garantir que apenas a mensagem certa seja enviada. Em seguida a mensagen é transferida e após ser recebida é mostrada no console e finalmente o programa volta ao estado inicial.

<https://wokwi.com/projects/420782572934566913>

# Item 4

Já para a comunicação I2C, iremos utilizar o DS1307, que é um Real Time Clock – RTC disponível no simulador Wokwi. O endereço I2C do DS1307 é 0x68. Um RTC é um hardware que garante a contagem de tempo na unidade de segundos. Muitos microcontroladores possuem RTC internos, mas alguns fazem uso de hardware externos. Para ler os valores, é necessário inicialmente configurar um valor de data e hora que deve, por exemplo, ser configurado manualmente pelo usuário. Nessa questão você deverá configurar o RTC para 24/09/2024 –13:27:00 e em seguida, realizar a leitura do mesmo a cada 5 segundos, e imprimir na tela do console (Serial USB) o valor lido. Na tabela a seguirão apresentados os principais endereços do RTC DS1307. 

Fonte: tabela elaborada pelo Prof. Pedro Henrique Almeida Miranda

Inicialmente foi definido os pinos do I2C utilizado além de duas funções utilitárias para converter de decimal para BCD (Binary-Coded Decimal) e vice-versa para facilitar a inserção e retirada dos dados do RTC (Real-Time Clock).

A primeira função criada a ser chamada é a **init\_i2c** responsável por inicializar a comunicação I2C com o RTC. Após isso é chamada a **setRTCDateTime** que altera as informações internas do RTC para as indicadas no enunciado e por fim foi feito um loop que a cada 5 segundos chama a função **readRTCDateTime** que lê os valores do RTCe mostra no console.

<https://wokwi.com/projects/420808642849582081>

# **Item 5**

Modifique o exemplo de código apresentado na videoaula (reproduzido abaixo) para controlar os LEDs RGB da placa BitDogLab usando o módulo PWM e interrupções, seguindo as orientações a seguir:

1. O LED vermelho deve ser acionado com um PWM de 1kHz.
2. O LED verde deve ser acionado com um PWM de 10kHz.
3. O Duty Cycle deve ser iniciado em 5% e atualizado a cada 2 segundos em incrementos de 5%. Quando atingir o valor máximo, deve retornar a 5% e continuar a incrementando. - Fazer isso para ambos os LEDs: azul e vermelho.
4. Tente controlar frequência e o Duty Cycle do LED azul de forma independente do que fez nos LEDs vermelho e verde. Você consegue? Por que não?



Fonte: tabela elaborada pelo Prof. Pedro Henrique Almeida Miranda

Inicialmente foi definido os valores de configuração para os PWMs onde para os períodos de cada LED utilizem 125MHz (clock interno) divido pela frequência desejada. Uma função **setup\_pwm** foi criada para configurar as saídas PWM e criar a interrupção **pwm\_irq\_handler** responsável por altera o duty cicle dos LEDs a cada 2 segundos.

Como os Pinos Vm(GPIO13) e Az(GPIO12) dividem o mesmo slice do PWM ambos devem obrigatoriamente possuir a mesma frequência, porém ainda é possível modificar o Duty Cicle de cada separadamente.

<https://wokwi.com/projects/420996921742550017>

# **Item 6**

Refaça o Programa Prático 00 presente no Ebook do Capítulo de ADC (Capítulo 8), mas implementando no código a conversão da unidade de medida da temperatura de Celsius para Fahrenheit.

A Tabela 1 mostra um lembrete das conexões de GPIO da placa BitDogLab.



Fonte: tabela elaborada pelo Prof. Pedro Henrique Almeida Miranda

Para esse item foi utilizado o código sugerido no enunciado, com a adição da nova função **celsius\_to\_fahrenheit** que transforma o valor obtido em graus celsius em graus fahrenheit. Ambos os valores são exibidos no console.

<https://wokwi.com/projects/421321807702993921>