**Autor: Leonardo Bonifácio Vieira Santos**

**Passos necessários:**

* 1. Ter o vs code instalado e todas as extensões necessárias para o projeto(extensões de C/C++, cmake, raspberry pico, wokwi , etc)
* 2. Instalar e configurar o gcc(que é o compilador de C/C++ no windows), colocando ele no caminho das variáveis de ambiente para que seja acessado por todo o computador.

Link para download: <https://sourceforge.net/projects/gcc-win64/>

* 3. Instalar e configurar uma variável de ambiente para o compilador Arm gcc.

Link para download: <https://developer.arm.com/-/media/Files/downloads/gnu/13.3.rel1/binrel/arm-gnu-toolchain-13.3.rel1-mingw-w64-i686-arm-none-eabi.exe>

* 4. Configurar as variáveis de ambiente do sdk do pico após instalar o sdk mais recente(2.1.0) pelo vs code ao inicializar o projeto com a extensão do raspberry pico(new project c/c++ project, escolher versão mais recente e escolher a raspberry pico w, já que é a placa microcontroladora presente na bitdoglab).
* 5. Desenvolver o código necessário para criar um sinal de SOS em código morse utilizando 1 led na GPIO13 da Raspberry pico W.
* 6. Configurar sua licença gratuita de 30 dias do wokwi pelo vs code.
* 7. Criar os arquivos necessários para usar o wokwi(.toml e .json)
* 8. Configurar os dois arquivos wokwi.toml(colocando a versão e caminhos para os arquivos uf2 e elf após já ter compilado o arquivo.c e gerado os mesmos) e diagram.json(colocando a versão,autor,editor,quais partes você irá utilizar e que conexões serão necessárias)
* 9. Caso queira ‘Rodar’ o projeto na BitDogLab basta plugar ela no pc colocar em modo bootsel e arrastar o arquivo uf2 no explorador de arquivos em cima do nome da bitDogLab que aparecerá como um flash drive

**Imagens abaixo**





