**Relatório do Projeto: Código Morse SOS com LED**

**Implementação na Placa BitDogLab**

1. Visão Geral do Projeto

Este relatório documenta a implementação de um sinal SOS em código Morse utilizando um LED na placa de desenvolvimento BitDogLab. O projeto demonstra conceitos fundamentais de programação de sistemas embarcados utilizando a linguagem C, incorporando controle de GPIO e requisitos precisos de temporização.

2. Requisitos Técnicos

O projeto implementa as seguintes especificações de temporização para o sinal SOS em código Morse:

- Duração do ponto (.): 0,2 segundos LED ACESO

- Duração do traço (-): 0,8 segundos LED ACESO

- Intervalo entre elementos da mesma letra: 0,125 segundos

- Intervalo entre letras: 0,25 segundos

- Intervalo de repetição da mensagem: 3,0 segundos

3. Detalhes da Implementação

3.1 Configuração do Projeto

O projeto foi construído em cima de um projeto de exemplo chamado blink do Raspberry Pi Pico W, todo o desenvolvimento foi realizado a partir desse boilerplate.

3.2 Configuração do Wokwi

Para rodar o Wokwi no VSCode, foi necessária a criação dos arquivos diagram.json e wokwi.toml. Depois de criado o diagram.json bastou abrir para montar o diagrama visualmente. Já o wokwi.toml foi configurado para apontar os arquivos blink.uf2 e blink.elf do projeto.

3.3 Configuração de Hardware

A implementação utiliza o pino GPIO 13 para controle do LED, embora o código suporte pinos alternativos (11 ou 12) através de uma simples modificação de constante. O sistema é projetado para ser compatível tanto com placas Pico padrão quanto com variantes Pico W, implementando compilação condicional apropriada.

3.4 Arquitetura de Software

A implementação do software segue uma abordagem modular, utilizando várias funções principais:

1. Função de Inicialização:

1. int pico\_led\_init(void)

Esta função gerencia a inicialização do GPIO e garante a configuração adequada para as variantes Pico padrão e Pico W.

2. Função de Controle do LED:

1. void pico\_set\_led(bool led\_on)

Esta função fornece controle abstraído do LED, tratando as diferenças entre GPIO padrão e implementações Pico W.

3. Funções de Elementos do Código Morse:

1. void pico\_morse\_dot()

2. void pico\_morse\_dash()

Estas funções implementam os elementos fundamentais do código Morse com controle preciso de temporização.

4. Função Principal do Sinal:

1. void pico\_sos()

Esta função orquestra o padrão completo do sinal SOS, gerenciando todos os requisitos de temporização e sequências de sinais.

4. Componentes Principais do Código

4.1 Constantes de Temporização

1. #define MORSE\_DOT\_MS 200

2. #define MORSE\_DASH\_MS 800

3. #define MORSE\_SPACE\_MS 250

4. #define MORSE\_REPEAT\_MS 3000

Estas constantes garantem temporização consistente em todos os elementos do código Morse, atendendo precisamente aos requisitos do projeto.

4.2 Loop Principal do Programa

1. int main() {

2. int rc = pico\_led\_init();

3. hard\_assert(rc == PICO\_OK);

4. while (true) {

5. pico\_sos();

6. }

7. }

O programa principal implementa um loop infinito que transmite continuamente o sinal SOS com os atrasos apropriados.

5. Testes e Verificação

A implementação foi testada em múltiplos ambientes:

1. Simulador Wokwi Web

2. Integração Wokwi com VS Code

3. Placa de Desenvolvimento BitDogLab física

Todos os ambientes de teste confirmaram a temporização adequada e geração do sinal, com o LED produzindo com sucesso o padrão SOS (... --- ...) de acordo com as especificações.

6. Modularidade e Manutenibilidade do Código

A implementação demonstra boas práticas de engenharia de software através de:

- Separação clara de funções e responsabilidades

- Constantes bem definidas para parâmetros de temporização

- Tratamento de compatibilidade multiplataforma

- Comentários claros e documentação

- Estilo de codificação e formatação consistentes

7. Conclusão

O projeto implementa com sucesso todas as especificações necessárias para o gerador de sinal SOS em código Morse. O código é bem estruturado, manutenível e adequadamente documentado. A implementação demonstra o uso adequado de conceitos e práticas de programação de sistemas embarcados, mantendo a compatibilidade entre diferentes ambientes de desenvolvimento e teste.

O design modular permite modificações fáceis e potenciais melhorias, como suporte a padrões adicionais de código Morse ou ajuste de parâmetros de temporização. O projeto serve como uma base sólida para a compreensão de conceitos de programação de sistemas embarcados usando a placa de desenvolvimento BitDogLab.

8. Código-fonte

*blink.c*

1. #include "pico/stdlib.h"

2.

3. // Pico W devices use a GPIO on the WIFI chip for the LED,

4. // so when building for Pico W, CYW43\_WL\_GPIO\_LED\_PIN will be defined

5. #ifdef CYW43\_WL\_GPIO\_LED\_PIN

6. #include "pico/cyw43\_arch.h"

7. #endif

8.

9. // Define the delay for the LED in milliseconds if not already defined

10. #ifndef LED\_DELAY\_MS

11. #define LED\_DELAY\_MS 125

12. #endif

13.

14. // Define the default LED pin for the Pico device

15. #define PICO\_DEFAULT\_LED\_PIN 13

16.

17. // Define the timing for Morse code elements in milliseconds

18. #define MORSE\_DOT\_MS 200

19. #define MORSE\_DASH\_MS 800

20. #define MORSE\_SPACE\_MS 250

21. #define MORSE\_REPEAT\_MS 3000

22.

23. // Perform initialisation of the LED

24. int pico\_led\_init(void) {

25. #if defined(PICO\_DEFAULT\_LED\_PIN)

26.     // A device like Pico that uses a GPIO for the LED will define PICO\_DEFAULT\_LED\_PIN

27.     // so we can use normal GPIO functionality to turn the led on and off

28.     gpio\_init(PICO\_DEFAULT\_LED\_PIN);

29.     gpio\_set\_dir(PICO\_DEFAULT\_LED\_PIN, GPIO\_OUT);

30.     return PICO\_OK;

31. #elif defined(CYW43\_WL\_GPIO\_LED\_PIN)

32.     // For Pico W devices we need to initialise the driver etc

33.     return cyw43\_arch\_init();

34. #endif

35. }

36.

37. // Turn the LED on or off

38. void pico\_set\_led(bool led\_on) {

39. #if defined(PICO\_DEFAULT\_LED\_PIN)

40.     // Just set the GPIO on or off

41.     gpio\_put(PICO\_DEFAULT\_LED\_PIN, led\_on);

42. #elif defined(CYW43\_WL\_GPIO\_LED\_PIN)

43.     // Ask the wifi "driver" to set the GPIO on or off

44.     cyw43\_arch\_gpio\_put(CYW43\_WL\_GPIO\_LED\_PIN, led\_on);

45. #endif

46. }

47.

48. // Function to blink a Morse code dot

49. void pico\_morse\_dot() {

50.     pico\_set\_led(true);

51.     sleep\_ms(MORSE\_DOT\_MS);

52.     pico\_set\_led(false);

53.     sleep\_ms(MORSE\_DOT\_MS);

54. }

55.

56. // Function to blink a Morse code dash

57. void pico\_morse\_dash() {

58.     pico\_set\_led(true);

59.     sleep\_ms(MORSE\_DASH\_MS);

60.     pico\_set\_led(false);

61.     sleep\_ms(MORSE\_DOT\_MS);

62. }

63.

64. // Function to blink SOS in Morse code

65. void pico\_sos() {

66.     pico\_morse\_dot();

67.     sleep\_ms(LED\_DELAY\_MS);

68.     pico\_morse\_dot();

69.     sleep\_ms(LED\_DELAY\_MS);

70.     pico\_morse\_dot();

71.

72.     sleep\_ms(MORSE\_SPACE\_MS);

73.     pico\_morse\_dash();

74.     sleep\_ms(LED\_DELAY\_MS);

75.     pico\_morse\_dash();

76.     sleep\_ms(LED\_DELAY\_MS);

77.     pico\_morse\_dash();

78.

79.     sleep\_ms(MORSE\_SPACE\_MS);

80.     pico\_morse\_dot();

81.     sleep\_ms(LED\_DELAY\_MS);

82.     pico\_morse\_dot();

83.     sleep\_ms(LED\_DELAY\_MS);

84.     pico\_morse\_dot();

85.     sleep\_ms(MORSE\_SPACE\_MS);

86.

87.     sleep\_ms(MORSE\_REPEAT\_MS);

88. }

89.

90. // Main function to initialize the LED and blink SOS in an infinite loop

91. int main() {

92.     int rc = pico\_led\_init();

93.     hard\_assert(rc == PICO\_OK);

94.     while (true) {

95.         pico\_sos();

96.     }

97. }

98.

*diagram.json*

1. {

2. "version": 1,

3. "author": "Anonymous maker",

4. "editor": "wokwi",

5. "parts": [

6. { "type": "board-pi-pico-w", "id": "pico", "top": 0, "left": 0, "attrs": {} },

7. {

8. "type": "wokwi-led",

9. "id": "led1",

10. "top": 34.8,

11. "left": -149.8,

12. "attrs": { "color": "red" }

13. },

14. {

15. "type": "wokwi-resistor",

16. "id": "r1",

17. "top": 139.2,

18. "left": -163.75,

19. "rotate": 90,

20. "attrs": { "value": "1000" }

21. }

22. ],

23. "connections": [

24. [ "pico:GP0", "$serialMonitor:RX", "", [] ],

25. [ "pico:GP1", "$serialMonitor:TX", "", [] ],

26. [ "led1:C", "r1:1", "green", [ "v0" ] ],

27. [ "r1:2", "pico:GND.4", "green", [ "h0" ] ],

28. [ "led1:A", "pico:GP13", "green", [ "v0" ] ]

29. ],

30. "dependencies": {}

31. }