RESPOSTAS DAS ATIVIDADES DA UNIDADE 4 (MOODLE)

Nome: José Adriano Filho
Programa: EmbarcaTech

Obs.: Tarefa 1 foi anulada.

Tarefa 2:

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 piscar por 10 segundos na frequência de 10 Hz.

Link do WOKWI: https://wokwi.com/projects/419879507629645825

a) Configurações de GPIO:

- O LED está conectado ao gpio 13.
- O botão está conectado ao gpio 5.
- O LED é configurado como saída e o botão como entrada com pull-up.

b) Interrupções e Timers:

- Utilizei timers repetitivos para monitorar o botão e piscar o LED.
- A função “monitora_botao_callback” verifica o estado do botão a cada 100 ms.
- A função “pisca_led_callback” controla o piscar do LED com uma frequência de 10 Hz e duração de 10 segundos.

c) Lógica Principal:

- O botão deve ser pressionado cinco vezes para ativar o LED.
- O LED pisca por 10 segundos a uma frequência de 10 Hz quando o botão é pressionado cinco vezes.
- Se o LED já estiver piscando, ele será desligado e o contador será reiniciado.

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

Link do WOKWI: https://wokwi.com/projects/419899992530262017

a) Botões e Frequências:

- O LED continua a piscar a cada 5 vezes que o botão A é pressionado.
- O botão B alterna a frequência de piscar do LED entre 10 Hz (frequência padrão) e 1 Hz.
- Quando o LED está piscando, pressionar o botão B não muda a frequência; a frequência só pode ser alterada quando o LED não está piscando.

b) Configuração de GPIO:

- O LED está conectado ao gpio 13.
- O botão A está conectado ao gpio 5.
- O botão B está conectado ao gpio 6.
- Os botões são configurados como entradas com pull-up, e o LED é configurado como saída.

c) Funções Adicionais:

- Função “monitora_botao_B_callback” para monitorar o estado do botão B e alternar a frequência do LED.
- A lógica de debounce é aplicada para evitar leituras falsas dos botões.

d) Timers Repetitivos:

- Dois timers repetitivos são usados: um para monitorar o botão A e outro para o botão B.
- O timer de monitoramento do botão A controla a contagem de pressões e o início do piscar do LED.
- O timer de monitoramento do botão B alterna a frequência de piscar do LED entre 10 Hz e 1 Hz.

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.

Link do WOKWI: https://wokwi.com/projects/420100405860259841

Este código utiliza o SDK do Raspberry Pi Pico W. Ele faz o seguinte:

a) Inicializa as UARTs e os pinos correspondentes.
b) Lê um caractere do console usando scanf.
c) Limpa o buffer de recepção da UART1.
d) Transmite o caractere da UART0 para a UART1.
e) Recebe o caractere transmitido pela UART1.
f) Exibe o caractere recebido no console.

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 imprimindo na tela do console (Serial USB) o valor lido.

Link do WOKWI: https://wokwi.com/projects/420152505726086145

a) Configuração do RTC DS1307:

- Define a data e a hora inicial como 24/09/2024 - 13:27:00.
- Converte valores decimais para BCD e vice-versa.
- Configura os registradores internos do DS1307 com os valores de data e hora.

b) Leitura do RTC:

- A função “get_rtc_time” lê a data e a hora do DS1307 e as imprime no console.

c) Configuração do I2C:

- Inicializa o I2C com pinos SDA no gpio 16 e SCL no gpio 17 com uma velocidade de 100kHz.

d) Timer Repetitivo:

- Define um timer repetitivo que executa a função “repeating_timer_callback” a cada 5 segundos para ler e exibir a data e hora.

e) Funções Auxiliares:

- “decimal_bcd”: Converte decimal para BCD.
- “bcd_decimal”: Converte BCD para decimal.
- “set_rtc_time”: Configura a data e hora no DS1307.
- “init_i2c”: Inicializa a comunicação I2C.

5. Modifique o exemplo de código apresentado na videoaula (reproduzido abaixo) para controlar os três LEDs RGB da placa BitDogLab usando o módulo PWM e interrupções, seguindo as orientações a seguir:
A - O LED vermelho deve ser acionado com um PWM de 1kHz.
B - 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%.
O LED azul deve ser acionado com um PWM de 10kHz.

Link do WOKWI: https://wokwi.com/projects/420265147747004417

a) **Frequências de PWM**:

- LED vermelho: 1 kHz
- LED azul: 10 kHz
- LED verde: 4 kHz (como não foi indicado na questão, foi utilizado 4 kHz)

a) **Duty Cycle**:

- O duty cycle inicia em 5% e incrementa de 5% a cada 2 segundos.
- Quando atinge 100%, o duty cycle retorna a 5%.

b) **Configuração de GPIO e PWM**:

- Cada LED é configurado para usar a função PWM nos pinos especificados.
- O divisor do clock PWM é definido como 10.

- O valor máximo do contador PWM (wrap) é configurado para cada frequência específica.

c) **Interrupções de PWM**:

- A função “pwm_irq_handler” é responsável por atualizar o duty cycle dos LEDs a cada 2 segundos.

Aqui estão os principais pontos do código:

a) **Função “**setup_pwm”:

- Configura o PWM para cada LED.
- Define o nível inicial do PWM para os LEDs.
- Habilita as interrupções do PWM.

b) **Função “**pwm_irq_handler”:

- Atualiza o duty cycle dos LEDs a cada 2 segundos.
- Incrementa o duty cycle em 5%.
- Retorna o duty cycle para 5% quando atinge 100%.

6. Refaça o programa pratico 01 presente no Ebook do Capítulo de ADC, mude a unidade de medida da temperatura de celsius para fahrenheit.

Link do WOKWI: https://wokwi.com/projects/420510764170441729

a. Definições e Bibliotecas:

- Biblioteca padrão de entrada e saída.
- Biblioteca para funções básicas do Pico (GPIO e temporização).
- Biblioteca para funções do conversor ADC.
- O canal ADC 4 é utilizado para o sensor de temperatura interno.

b. Funções de Conversão:

- “adc_to_temperature(uint16_t adc_value)”: Converte o valor lido do ADC para temperatura em graus Celsius, usando a equação fornecida no datasheet do RP2040.
- “celsius_to_fahrenheit(float temperature_celsius)”: Converte a temperatura de Celsius para Fahrenheit.

c. Configuração Inicial:

- Inicializa a comunicação serial.
- Inicializa o módulo ADC e habilita o sensor de temperatura interno no canal 4.

d. Leitura e Exibição da Temperatura:

- Lê o valor do ADC no canal do sensor de temperatura.
- Converte o valor lido para temperatura em Celsius e depois para Fahrenheit.
- Imprime a temperatura em Fahrenheit na comunicação serial.
- Utiliza um atraso de 1 segundo entre as leituras.

7. Como o ADC converte sinais analógicos do joystick em valores digitais no exemplo 02?

Como sabemos um **ADC** faz a conversão de sinais analógicos em valores digitais, no **RaspberryPi pico w** temos um conversor de 12 bits, assim podemos variar os valores de 0 a 4095, por exemplo a tensão do microcontrolador que vai de 0 a 3,3V, pode ser dividida em 4096 valores discretos.

No exemplo 02, onde utilizamos um joystick como entrada, podemos fazer leituras analógicas dos eixos x e y. Os movimentos do joystick alteram a tensão nos pinos do eixo X (VRX) e do eixo Y (VRY), essas tensões analógicas são então convertidas para valores digitais pelo ADC do Pico, por exemplo, se o joystick estiver no centro, os valores do ADC podem ser aproximadamente metade do valor máximo (por exemplo, cerca de 2048 para um ADC de 12 bits), movendo o joystick para uma extremidade aumentará a tensão e resultará em um valor de ADC mais próximo do valor máximo (por exemplo, 4095).

Basicamente temos três fases no processo:

1. **Posição do Joystick -> Tensão Analógica**: O movimento do joystick altera a resistência, modificando a tensão medida nos pinos VRX e VRY.

2. **Tensão Analógica -> Sinal Digital (ADC)**: O ADC amostra e quantiza essa tensão, convertendo-a em um valor digital.

3. **Valor Digital**: O valor digital resultante pode ser usado pelo microcontrolador para determinar a posição do joystick.

Desta forma temos a conversão de valores do mundo real para o mundo digital.