SEL0433/SEL0336/SEL0614
APLICAÇÃO DE MICROPROCESSADORES

Roteiros de Aulas, Atividades e Projetos
Parte 3 – Microcontroladores de 32 bits e ESP32 DevKit

Motivação e objetivos: será abordado a programação de periféricos em microcontroladores de 32 bits (ESP32), tais como: comunicação serial, PWM (modulação por largura de pulso) e introdução a comunicação sem fio.

Instruções e datas para entrega de atividades/projetos: todas as atividades podem ser realizadas em grupos de três pessoas. Não é permitido realizá-las de forma individual. Instruções específicas e datas de entrega estarão contidas em documento específico na tarefa correspondente atribuída no e-Disciplinas (Moodle).
Material de apoio: além dos links disponibilizados diretamente no presente documento, o material principal estará também disponível no e-Disciplinas.
Instruções para as aulas: As ferramentas computacionais utilizadas no curso estão disponíveis nos computadores do laboratório. Durante o horário da aula, os computadores são de uso exclusivo para realização das atividades pertencentes à disciplina e não devem ser utilizados para outros fins. **Importante:** ao final da aula ou ao término do uso dos computadores, não esqueça de fazer logout de contas pessoais que eventualmente tenha acessado durante seu uso (email, e-Disciplinas etc.), desligar os computadores e organizar mesas e cadeiras que utilizou.

Recomendações importantes: caso necessário, recomenda-se que a utilização do ESP32 DevKit durante as aulas seja revezada entre as três pessoas, possibilitando a cada aluno(a) a oportunidade de reproduzir as atividades, tais como manipular a plataforma e realizar configurações, gravar o programa no kit etc., em razão da limitação de unidades. Recomenda-se cuidado e atenção ao manipular as placas, pois seus componentes são extremamente sensíveis a danos. Em caso de dúvidas, sempre perguntar antes de prosseguir com alguma ação. Mantendo esse cuidado, será possível que outras turmas também utilizem os kits em boas condições. Durante as aulas, o monitor estará à disposição para ajudar a resolver problemas de uso do kit. Ao término da aula, procure devolver o kit da mesma forma que o recebeu: guardado na embalagem junto ao cabo USB.

Monitoria: será possível agendar esclarecimentos de dúvidas e atendimento para auxílio nas atividades/projetos em outros horários, com o professor e com monitores. Da mesma forma, podem utilizar os laboratórios (LEI Maior ou Lab. de Microprocessadores) em horários que não estão sendo usados para aulas para realizar atividades da disciplina (caso necessitem de computadores, uso das ferramentas computacionais, kits etc.), preferencialmente agendando previamente com técnicos responsáveis, monitores e/ou com professor.

Roteiros

Aula 10

Pré-aula

- **Material de aula: "**Cap. 9 – Introdução aos Microcontroladores de 32" (**disponível no e-Disciplinas**).
    - **Objetivos da aula:** conhecer famílias e recursos de microcontroladores de 32 bits (arquiteturas e microprocessadores ARM), olhando brevemente a família STM32 (STM32F103C8T6 e placa BluePill) e especialmente o ESP32 e placa DevKit.

Pós-aula

- Testar no Wokwi o exemplo demostrado em aula referente ao blink LED na placa BluePill (microcontrolador STM32F103), observando a programação no framework Arduino e na IDE nativa (Cube IDE) conforme tutoriais nos slides do Cap. 9;
- Demostrar como acionar a saída PC13 (LED onboard na BluePill) a partir da programação "bare-metal".

Aula 11

Pré-aula

- **Material de aula: "**Cap. 10 – Programação para Microcontroladores de 32 bits" (disponível no e-Disciplinas).
- **Objetivos da aula:** introdução a programação em linguagem C/C++ para ESP32; programação de periféricos e projetos usando I/O Digital, comunicação serial, sensores, ADC/DAC etc.
- **Recursos:** simulador Wokwi, placa DevKit + cabo USB, Arduino IDE; pasta com programas de exemplo disponível no e-Disciplinas.

Roteiro e tutoriais durante a aula

Passo 1 - Abrir a Arduino IDE e seguir as instruções no material de aula (Cap. 10) referente à parte: "**Configurando Ambiente de Programação**" disponível no e-Disciplinas, visando instalar e configurar o framework da ESP32 e bibliotecas relacionadas.

Passo 2 - Compilar e gravar na placa ESP32 Devkit ou no Simulador Wokwi os seguintes programas disponibilizados na pasta de exemplo (tutoriais a partir do slide 35 do Cap. 10):

- "**sketch_LED_blink**" (escrever durante a aula, compilar e gravar na placa usando Arduino IDE)
- "**sketch_timer_interrupt**" (montar o projeto no simulador Wokwi e testar o programa).

- "**sketch_Touch**" (abrir na Arduino IDE). Explorar o recurso "touch" por meio do pino 13 da placa e o uso da UART por meio do monitor serial e serial plotter da Arduino IDE.

- "**sketch_DisplayOLED_I2C**" **(**montar o projeto no simulador Wokwi e testar o programa – inserir as bibliotecas necessárias). Explorar o recurso de comunicação I2C por meio do display OLED.

- "**sketch_ADC_DAC_Pot_LCD" (**montar o projeto no simulador Wokwi e testar o programa inserir as bibliotecas necessárias). Explorar o recurso de comunicação I2C por meio do display LCD I2C e de leituras analógicas com ADC e DAC da ESP32.

Pós-aula

- Realizar as simulações acima no Wokwi e estudar as aplicações (procurando modificar os programas para testar outras funcionalidades e explorar outras ideias de projetos com base nos recursos disponíveis no simulador).
- A reprodução destes exemplos e criação dos projetos no simulador Wokwi servirão como tutorial para o projeto final.

Aula 12

- **Material de aula: "**Cap. 11 – PWM e Introdução a Comunicação Sem Fio" (**disponível no e-Disciplinas**).
- **Objetivos da aula:** programação de periféricos como PWM (modulação por largura de pulsos) para controle de brilho, posição e velocidade; Introdução a comunicação wireless Wi-Fi e Bluetooth com aplicações Web e controle por dispositivo móvel; Considerações finais sobre a disciplina e assuntos tratados em outros cursos.

Roteiro e tutoriais durante a aula

Compilar e gravar na placa ESP32 Devkit ou no Simulador Wokwi os seguintes programas disponibilizados na pasta de exemplo:

- Reproduzir o exemplo "LEDC Fade" disponibilizado na documentação da biblioteca LEDC PWM ESP32 Arduino: https://espressif-docs.readthedocs-hosted.com/projects/arduino-esp32/en/latest/api/ledc.html (abrir na Arduino IDE e gravar na placa). Utilizar o LED onboard para verificar o controle PWM do brilho em 8 bits de resolução.

- "**sketch_PWM_servo_manual**" (montar o projeto no simulador Wokwi e testar o programa). Escolher um servo motor, realizar as ligações e verificar o controle PWM da posição do servo.
- "**sketch_Touch_WiFi**" (abrir na Arduino IDE e gravar na placa). Explorar o recurso "touch" por meio do pino 13 da placa e verificar uma saída acionada (qualquer outro pino) de forma remota, via Wi-Fi, acessando pelo browser o endereço da ESP32 informado no monitor serial da Arduino IDE. Adicionar as bibliotecas necessárias na Arduino IDE por meio do tutorial a partir do slide 26 do Cap. 11. Adicionar usuário e senha da rede Wi-Fi LabMicros.

- "**sketch_ Bluetooth_exemplo_01_Dabble**" (abrir na Arduino IDE e gravar na placa)**.** Instalar o aplicativo Dabble e a biblioteca Dabble ESP32 na Arduino IDE – ver tutorial no slide 31 do Cap. 11. Conectar na ESP32 via Bluetooth e realizar o controle da GPIO 2 (LED onboard) por meio do API "LED Brightness Control".

- "**sketch_ Bluetooth_exemplo_02_Dabble"** (abrir na Arduino IDE e gravar na placa). Realizar o mesmo procedimento adotado no exemplo anterior.

Pós-aula

- Realizar as simulações acima no Wokwi e estudar as aplicações (procurando modificar os programas para testar outras funcionalidades e explorar outras ideias de projetos com base nos recursos disponíveis no simulador).
- A reprodução destes exemplos e criação dos projetos no simulador Wokwi servirão como tutorial para o projeto final.

Projeto final: Controle PWM, Comunicação Serial e Wireless

Objetivos

- Exercitar a programação em alto nível para microcontroladores de 32 bits
- Desenvolver um projeto prático com foco em comunicação sem fio, comunicação serial e PWM (pulse width modulation).
- Usar bibliotecas otimizadas para programação de periféricos e controle de GPIO, UART, I2C, Display OLED, Touch, PWM, Wi-Fi e Bluetooth por meio da plataforma ESP32 Devkit e do ambiente de simulação virtual Wokwi.

Requisitos do projeto

Programa 1 - Projeto usando o simulador Wokwi (ESP32 com framework Arduino IDE – Realizar como tarefa): Criar um projeto no Wokwi utilizando o template da ESP32 e conectar um servo motor a um pino da GPIO (PWM) e a alimentação (GND e 3V3) da placa. Conectar também à placa um Display OLED, atentando-se para a ligação correta dos pinos SDA e SCL disponíveis para I2C na ESP32.

- Escrever um programa para controle PWM do servo motor, começando em 0° e movendo-se continuamente em intervalos (delays) de 100 ms até 180°. Quando o servo motor chegar em 180°, aguardar um delay de 1 segundo e mover continuamente para

0° (sentido aposto) em intervalos de 200 ms. Quando chegar novamente em 0°, repetir todo o processo desde o início (loop).

- O processo da linha anterior (modo automático em loop) deve somente acontecer após um botão ser pressionado (botão loop);
- Um segundo botão deve parar o servo e desligar o sistema.
- Utilize a técnica PWM com a biblioteca "**ESP32Servo**", considerando que a especificação de controle do servo motor escolhido é um sinal modulado com frequência de 50Hz, com referências de posicionamento inicial e final entre 500 e 2500 microsegundos.
- O ângulo do servo e o sentido da rotação deverão ser exibidos em um display OLED ou LCD I2C (usar as bibliotecas específicas para controle do display conforme exemplos) conectado à ESP32 via I2C e no monitor serial/terminal (UART) com 115.200 de baud rate. Utilizar a biblioteca "wire" para comunicação serial
- Caso deseje implementar outras funcionalidades e recursos, sinta-se à vontade como, por exemplo, o uso de interrupção de GPIO para os botões; integração com os outros recursos (sensores e atuadores disponíveis no Wokwi). Tais implementações terão caráter complementar e de aprimoramento para a estrutura mínima solicitada acima.

- Para a parte referente ao controle PWM, poderá ser aceito algum projeto alternativo ao solicitado acima, desde que envolva controle PWM de atuadores (preferencialmente de servo motor ou motor de passo) e seja compatível ou esteja em sintonia ao que foi solicitado (por exemplo, um projeto integrador desenvolvido em outra disciplina que tenha envolvido esses conceitos ou alguma aplicação prática de controle de movimento/posição em mecanismos, atuadores, motores ou robótica). Os resultados também devem ser exibidos em um display OLED ou LCD I2C bem como no monitor serial/terminal via comunicação serial. Caso escolha essa opção, justificar no documento de entrega a aplicação escolhida e como os conceitos de controle PWM estão envolvidos.

Programa 2 - Projeto usando a placa ESP32 Devkit (framework Arduino IDE – Realizar durante a aula e testar no Laboratório): Criar um projeto na Arduino IDE utilizando a placa ESP32 Dev Module. Conectar um LED RGB (catodo comum) na GPIO placa conforme esquemático ilustrado na Figura abaixo, utilizando resistores de 220 ohms, jumpers e protoboard. Tentar utilizar o App Dabble para comunicação Bluetooth com a ESP32 (ou algum, outro App alternativo de comunicação Bluetooth).

- Com base na documentação e exemplos disponibilizados na página da biblioteca "LEDC PWM Arduino-ESP32" neste link: https://espressif-docs.readthedocs-hosted.com/projects/arduino-esp32/en/latest/api/ledc.html. Elaborar um programa para realizar a modulação de cores em um LED RGB utilizando a técnica PWM.
- O PWM deverá controlar o brilho individual de cada cor " R- Red", "G- Green" e "B - Blue" do LED, com resolução de 8 bits (256 níveis); isto é, cada pino GPIO associado aos terminais R, G e B deve ser vinculado à um canal PWM individual da ESP32.
- Utilizar a biblioteca Led Control PWM da ESP32: "ledc"
- O duty cycle deve variar de 0 a 100% em loop, com frequência de 5kHz, com valor de incremento pré-definido (por exemplo: se definir o valor "5", o controle/resolução do brilho será feito de 5 em 5, até 255).
- O valor de incremento será aplicado de forma individual a cada cor do LED RGB da seguinte forma: R = incremento*2; G = incremento; B = incremento*3;
- Exibir uma mensagem no Monitor Serial/terminal (UART – baud: 115.200) que indique o valor de incremento e duty cycle.
- **Comunicação sem fio:**
  - Tentar instalar o App Dabble no seu smartphone. No PC, manipular diretamente o exemplo "02 Terminal", disponibilizado pela biblioteca Dabble

(acessar na Arduino IDE em: Arquivos> Exemplos > DabbleESP32 > "02 - Terminal" - ou na pasta de exemplos disponibilizada no e-Disciplina, que contém este exemplo com explicações adicionais), adicionando uma variável que recebe o valor digitado no terminal do app Dabble para ser tratada como incremento no programa. Incorporar essa estrutura no programa principal que realiza o controle PWM do LED RGB de forma que o incremento do duty cycle seja feito por meio desta variável.

- Dessa forma, após gravar o programa na placa, abrir o app Dabble por meio do exemplo "Terminal", conectar-se a ESP32 e digitar o valor do incremento desejado para o duty cycle (ou seja, se digitar 5, o controle do brilho será feito de 5 em 5 até 255).
- É importante lembrar que o valor digitado no terminal do App Dabble deve ser reconhecido como string e posteriormente deve ser convertido para "inteiro" para ser tratado como valor de incremento pelo programa na Arduino IDE.
- Essa parte poderá ser opcional caso a integração com o app Dabble não funcione por algum fator externo.

- **Tarefa sugerida**: elaborar previamente o programa e testar/simular o projeto no Wokwi (exceto a parte de comunicação sem fio Bluetooth) antes de implementar fisicamente na placa ESP32 devkit durante as aulas (OBS. não é necessária a entrega desta simulação realizada no Wokwi, sendo apenas uma sugestão que vai permitir você adiantar uma parte da tarefa e corrigir possíveis erros de código antes de testar na placa propriamente).

Formato de entrega

- Apresentar em um documento os programas desenvolvidos (Programa 1 - projeto no Wokwi; Programa 2- projeto com a placa ESP32 Devkit) devidamente comentado.
- Apresentar o diagrama e fotos da montagem prática (microcontrolador e circuito montado no Wokwi para o Programa 1, com prints da simulação realizada mostrando, por exemplo, valores no display, monitor serial, e circuito montado; bem como, para o Programa 2, apresentar uma fotografia da montagem prática realizada com a placa da ESP32 Devkit, LED RGB, protoboard, print do monitor serial da Arduino IDE, print da tela do App Dabble etc.). Identificar as imagens com numeração de figuras e comentários sucintos explicando do que se trata.
- Apresentar os itens acima em único arquivo PDF.
- Além do arquivo em pdf acima referido, enviar também o código fonte dos programas desenvolvidos (Programa 1 e Programa 2), salvos na extensão ".ino".
- Portanto, a **entrega consistirá no envio de apenas 3 arquivos avulsos ('documento em PDF' e dois arquivos ".ino")**.
- Fazer o upload dos arquivos na respectiva tarefa atribuída no e-Disciplinas até a data especificada. A atividade poderá ser feita em grupos, nos moldes dos trabalhos anteriores.
- Qualquer dúvida sobre o formato de envio ou sobre a implementação da atividade prática, entrar em contato com o professor ou com um dos monitores.

Configurações

Figura 1 - ESP32 Pinout

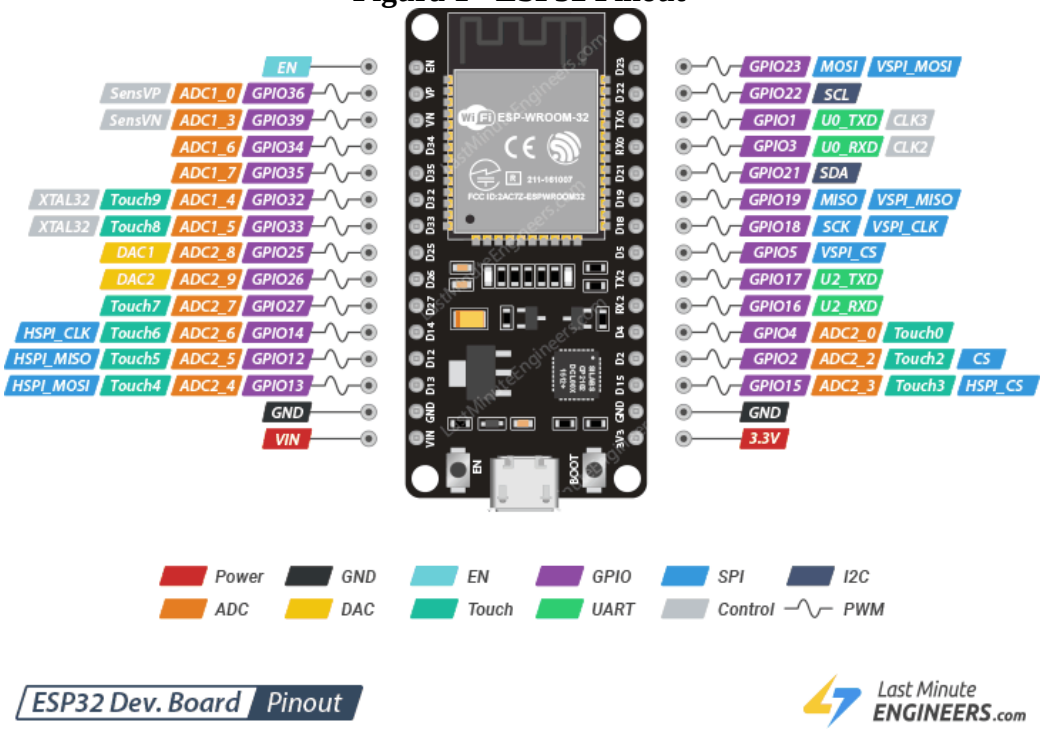


Figura 2 - [Wokwi - ESP32](#) e Servo Motor

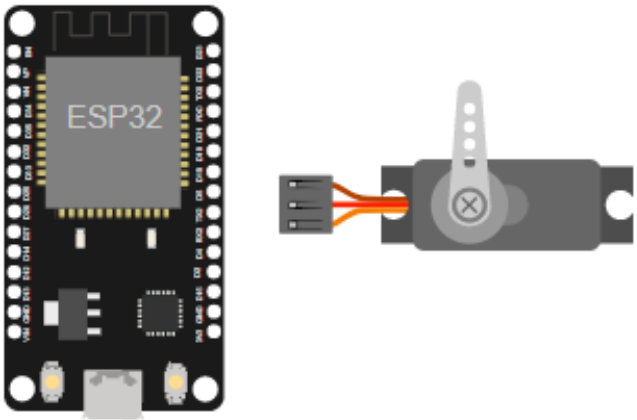
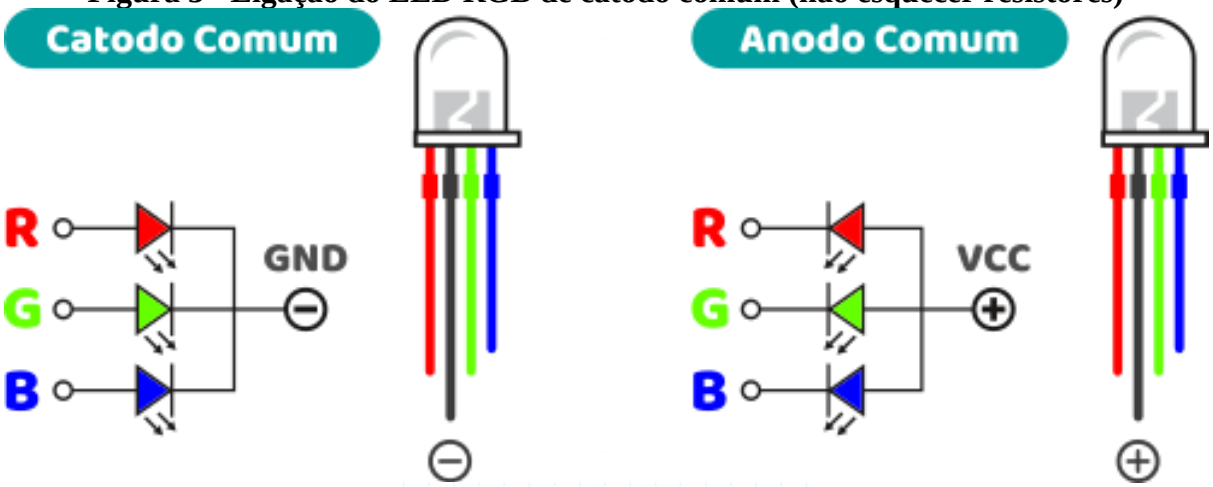


Figura 3 - Ligação do LED RGB de catodo comum (não esquecer resistores)



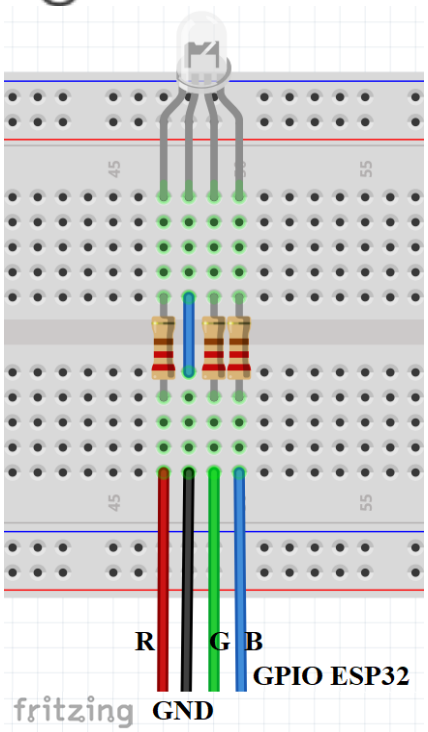
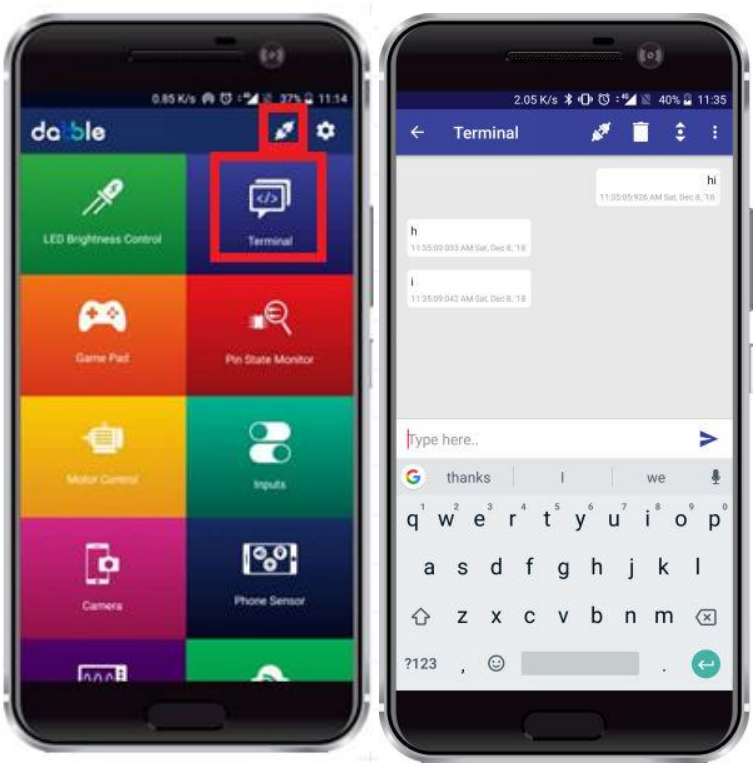


Figura 4 - Uso do Terminal do App Dabble



Critérios de avaliação

Item	Pontuação
Formato: entrega no formato solicitado (arquivo PDF com programa, discussão/diagramas + arquivos avulsos do projeto: ".ino")	**1**
Boas práticas de programação: programa com as linhas de código devidamente comentadas; diagramas e prints solicitados	**2**
Correção lógica do programa: atendimento ao enunciado, uso das bibliotecas solicitadas, projeto desenvolvido no Wokwi (Programa1), projeto implementado na prática (Programa 2)	**7**