SEL0614
APLICAÇÃO DE MICROPROCESSADORES

Parte 3 – Introdução aos Microcontroladores de 32 bits

Projeto 4 (final): Controle PWM e Comunicação

Objetivos

- Exercitar a programação em alto nível para microcontroladores de 32 bits
- Desenvolver um projeto prático com foco em 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 sendo opcional uso do framework nativo Espressif IDE e programação em ling. C/C++)

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 comunicação serial 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, ao ser pressionado, coloca o sistema em modo manual. Nesta condição o duty cycle (de 0 a 100 %) será incrementado de forma manual pelo usuário(a) por meio de um potenciômetro. Ou seja, o sinal analógico de tensão do potenciômetro será lido pelo ADC da ESP32 e convertido para valores de ângulo, controlando diretamente a posição do servo motor a partir da variação do duty cycle de forma manual pelo potenciômetro.
- Um terceiro botão deve desligar o sistema (servindo para parar o servo quando estiver no modo automático – loop).
- Utilize a técnica PWM com a biblioteca "**ESP32Servo**", ou "**MCPWM**" (Arduino) considerando que a especificação de controle do servo motor escolhido é um sinal modulado com frequência de 50 Hz, 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 (usar as bibliotecas "Adafruit" específicas para controle deste display) conectado à ESP32 via I2C e no monitor serial (UART) com 115.200 de baud rate. Utilizar a biblioteca "wire" para comunicação serial I2C.

- 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) e conceitos dos projetos anteriores.
- 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. Os resultados também devem ser exibidos em um display OLED bem como no monitor serial/terminal via UART. 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 (obrigatório usar o framework nativo da Espressif IDE ESP-IDF – e programação em ling. C/C++ - Realizar como tarefa usando simulador Wokwi – não será necessário testar na placa durante a aula)

Criar um projeto utilizando a placa ESP32 Dev Module. Conectar um LED RGB (catodo comum) na GPIO da placa conforme esquemático ilustrado na Figura abaixo, utilizando resistores de 220 ohms, jumpers e protoboard. Programar utilizando bibliotecas nativas do ESP32 por meio da Espressif IDE (software Eclipse desktop instalado nos PCs do laboratório), ou por meio da plataforma VSCode (eextensão da Espressif IDE ou PlataformIO) ou ainda com ESP-IDF via terminal (toolchain para WindowsPowerShell ou para Linux/MacOS/WSL).

- Com base na documentação e exemplos disponibilizados na página da biblioteca LED Control PWM (LEDC) por meio da programação nativa (não usar Arduino IDE), 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 a partir de 8 bits (256 níveis) (pode ser escolhido uma resolução maior); Dessa forma cada pino GPIO associado aos terminais R, G e B deve ser vinculado à um canal PWM individual da ESP32, utilizando a biblioteca LEDC (LED Control PWM)."
- 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, por exemplo).
- 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.

- Elaborar o programa e testar/simular o projeto no Wokwi (o qual dispõe de LED RGB) (OBS. não é necessária testar na placa ESP32 DevKit durante a aula).

- **Desafio 1 (opcional)** – Tentar desenvolver o projeto utilizando a linguagem de programação Rust ao invés da programação nativa ESP-IDF (implicando em consultar por conta própria a documentação, IDEs para execução e debugging, toolchain para ESP32 ESP-IDF, recursos do *Rust Analyzer* e do *Cargo*, etc.)

Formato de entrega

- Apresentar em um documento os programas desenvolvidos (Programa 1 - projeto no Wokwi; Programa 2- projeto com a placa ESP32 Devkit) com as partes principais devidamente comentadas.
- 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 o funcionamento do programa (modulação de cores do LED RGB)

- Apresentar os itens acima em único arquivo PDF ou link para documentação em repositório no GitHub (enviar o link por meio de um arquivo).
- Além da documentação acima referida, enviar também os códigos fonte dos programas desenvolvidos (Programa 1 e Programa 2). Os scripts podem ser enviados separadamente em arquivo compactado ou, caso escolher o GitHub para documentar o projeto, colocar os códigos diretamente no repositório, sendo opcional o envio separado pelo e-Disciplinas neste último caso.
- Realizar o upload dos arquivos na respectiva tarefa atribuída no e-Disciplinas até a data especificada. A atividade poderá ser feita preferencialmente em duplas.
- Qualquer dúvida sobre o formato de envio ou sobre a implementação da atividade prática, entrar em contato com o professor.

Configurações

Figura 1 - ESP32 Pinout

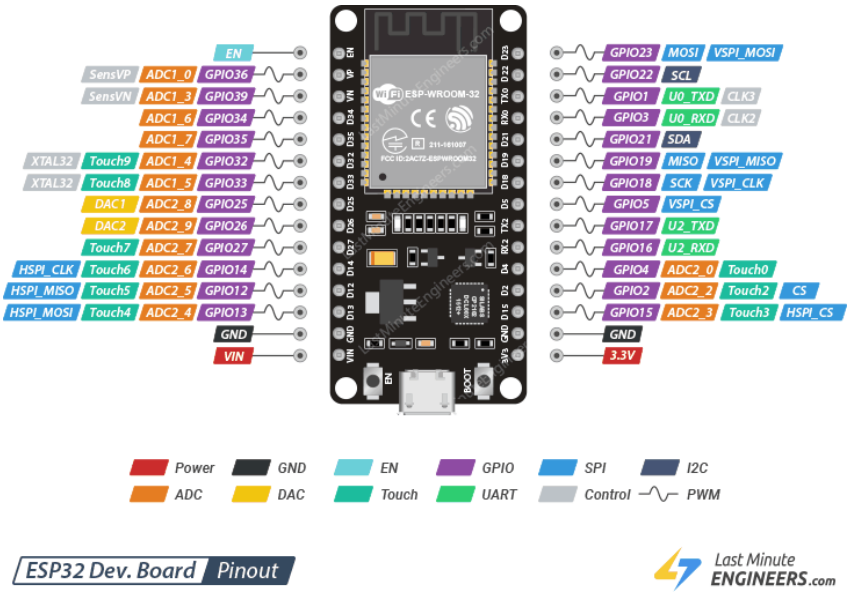


Figura 2 - Ligação do LED RGB de catodo comum (escolher 3 pinos da ESP e fazer as ligações no Wokwi - não esquecer resistores)

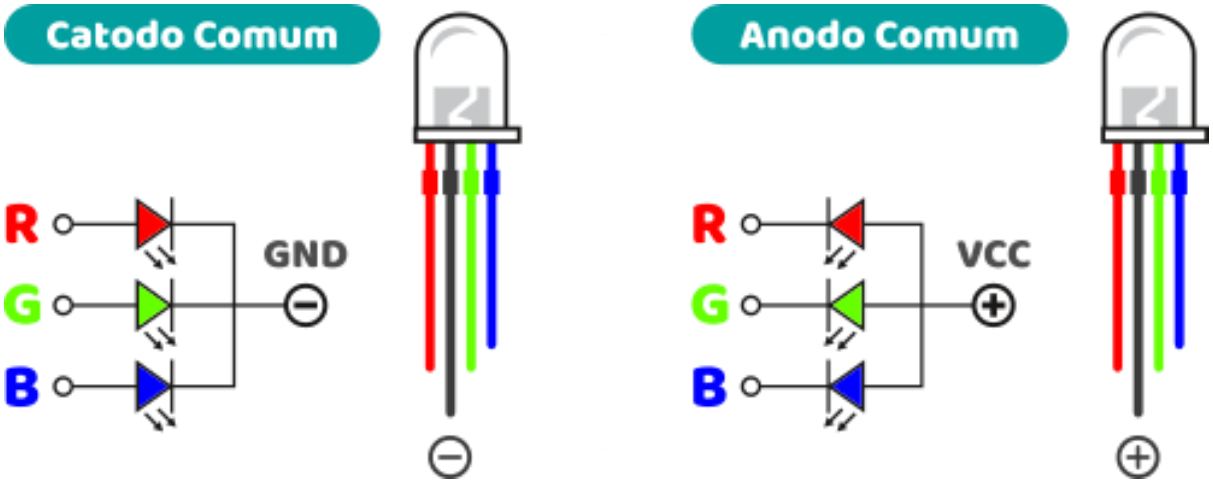


Figura 3 - Wokwi - ESP32 e Servo Motor

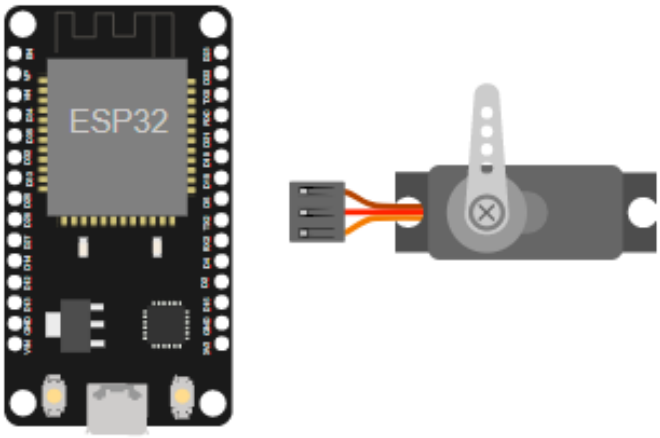
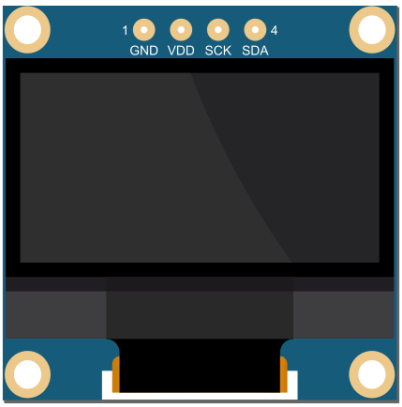


Figura 4 - Configurações do Display OLED: GND, VDD, SCK, e SDA



Critérios de avaliação

Item	Pontuação
Formato: entrega no formato solicitado (arquivo PDF com programa, discussão/diagramas + arquivos fonte do projeto ou essa documentação no GitHub)	**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**

Desafio 2 (projeto opcional – pontuação diferenciada*) – PWM e Sensor de Velocidade com Microcontrolador PIC

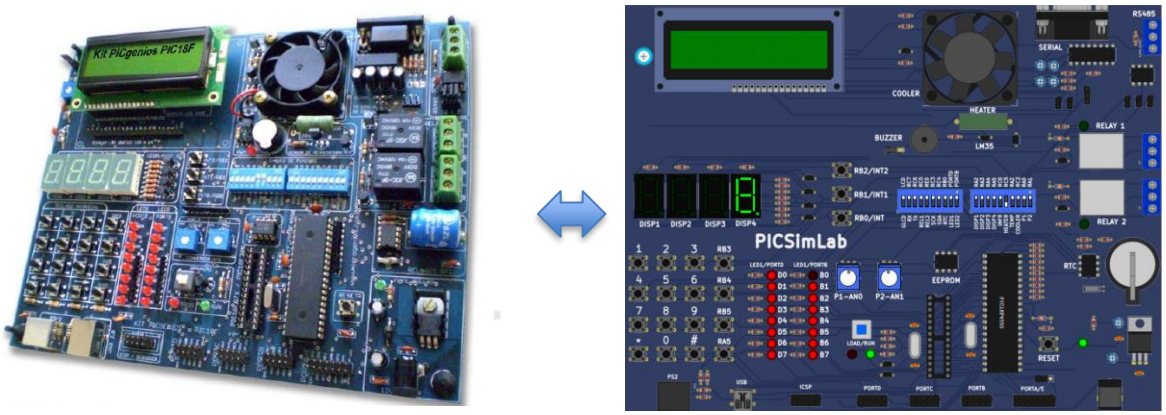
Esta proposta trata-se também de um desafio (opcional) já que o recurso PWM não foi especificamente explorado no PIC. O funcionamento do PWM no PIC, no entanto, segue conceitos semelhantes aos já abordados nos projetos anteriores, como I/O digital, timers, interrupções e ADC. Ou seja, a utilização do PWM envolve estudar o datasheet do microcontrolador (no caso o módulo CCP -PWM) e configurar os parâmetros nos registradores relativos ao controle PWM, assim como foi feito para os outros periféricos. O desafio visa dar a oportunidade de aprofundar o estudo do PWM em um nível mais baixo e de abstração menor. Existe possibilidade de obter uma pontuação extra ou diferenciada para o projeto 4, caso aceite o desafio. O projeto é opcional por demandar carga de trabalho extra, porém, com propósito de instigar.

Objetivo: realizar o controle PWM (Pulse Width Modulation) de uma ventoinha, em 5 velocidades diferentes utilizando o microcontrolador PIC18F4550. Diferente do controle PWM aplicado a um servo motor, que controla o posicionamento ou o ângulo, o controle neste projeto será voltado para a variação da velocidade do motor do ventilador. Deve-se ainda calcular a velocidade de rotação da ventoinha (RPM - rotações por minuto) para cada uma das 5 velocidades a partir de um sensor de velocidade por infravermelho. Os resultados devem ser exibidos em um display LCD.

Ferramentas: PIC18F4550, compilador MikroC Pro for PIC; simulador PIC Sim LAB (usando a placa virtual PIC Genios que possui a ventoinha, permitindo o controle PWM desejado); e simulador Simul IDE para validação do projeto.

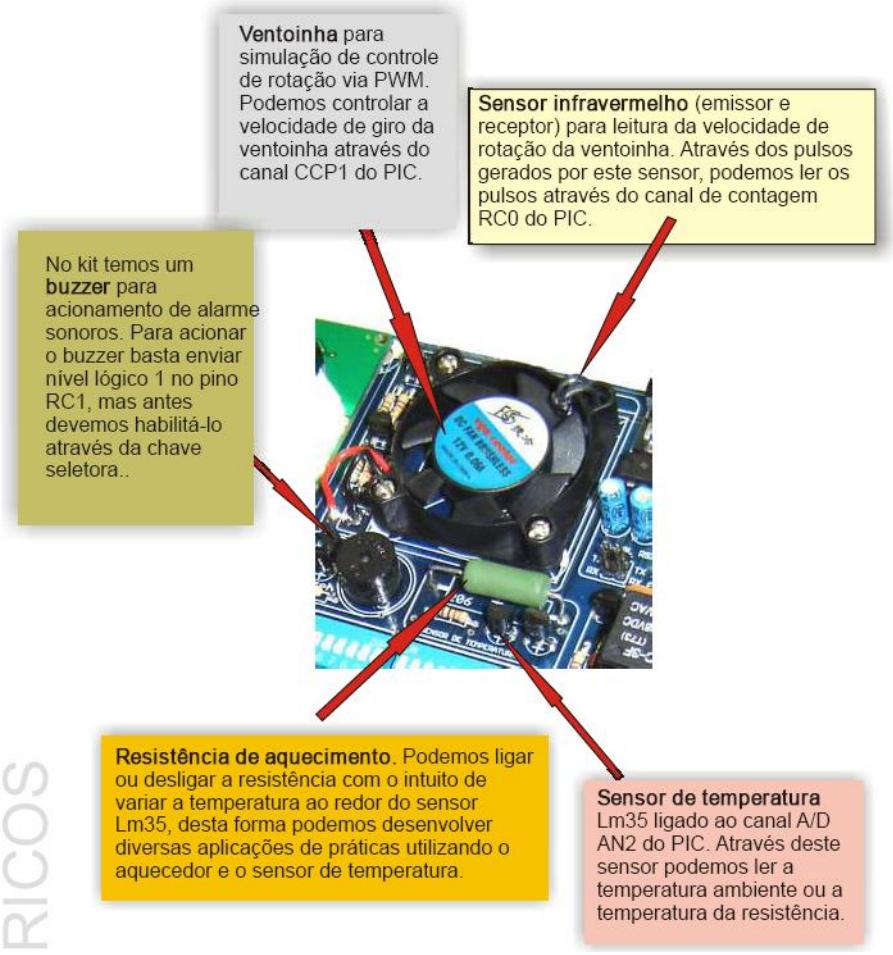
Hardware: O projeto deve ser baseado na placa PIC Genios (e não no kit EasyPIC v7 usado nos projetos anteriores) conforme ilustrado na Figura 5 (consulte o manual completo da placa aqui). Esta placa é virtualizada no simulador PIC Sim Lab (disponível para download aqui), que permite carregar o arquivo hex gerado após a compilação do programa no MikroC (o mesmo processo utilizado no Simul IDE**)**.

Figura 5 - Kit PIC Genios- placa física e virtual (PIC Sim Lab)



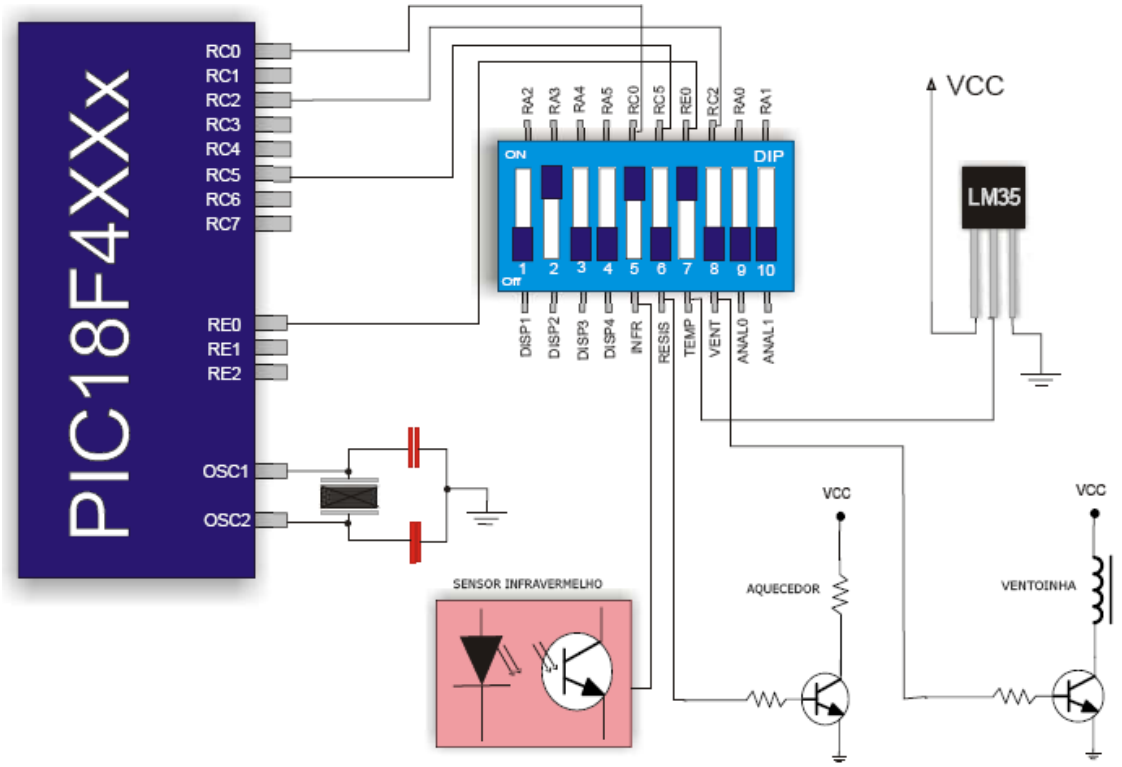
Conforme Figura 6, a placa possui uma ventoinha de 7 pás, equipada com um sensor de velocidade infravermelho. O sensor emite pulsos cada vez que uma das pás passa pela luz do sensor, funcionando como um tacômetro (contador de rotações). Os outros periféricos do kit demostrados na imagem, como o sensor de temperatura, buzzer e resistência, não serão utilizados neste projeto e podem ser ignorados.

Figura 6 – Detalhes de alguns periféricos do kit



Conforme ilustrado na Figura 7, a ventoinha está conectada ao pino RC2 (canal PWM) do PIC18F4550, que será controlado para ajustar a velocidade da ventoinha (saída). O sensor infravermelho está conectado ao pino RC0 (T13CKI), que está vinculado ao Timer1 ou Timer3 do PIC. Este pino será configurado como entrada para contar os pulsos gerados pelo sensor, a cada vez que uma pá da ventoinha passar pela luz, incrementando o valor do timer. Mais detalhes podem ser encontrados no manual do kit PIC Genios aqui).

Figura 7 – Ligações dos periféricos externos no microcontrolador



Pino	Descrição
RE0	Sensor de temperatura LM35
RC2	Ventoinha (cooler)
RC5	Resistência de aquecimento
RC1	Buzzer - via Jumper
RC0	Sensor infravermelho (tacometro)

Para o desafio, no entanto, utilizar a versão virtual da placa disponível no simulador PIC Sim Lab. Segue: a documentação do simulador, manual, e instalador referente ao uso da placa virtual PIC Genios. Também é possível realizar o projeto na placa física se assim for do seu interesse (temos uma unidade do kit PIC Genios disponível para empréstimos).

Software: Módulo CCP = Capture, Compare, PWM. Periférico presente em alguns microcontroladores PIC (consultar documentação no datasheet no PIC18F4550):

- **Modo Capture:** contagem de tempo entre dois eventos ocorridos no pino do PIC (borda de descida ou subida)
- **Modo Compare**: contagem de tempo entre dois eventos ocorridos no pino do PIC e comparação com um valor pré-determinado

- **Modo PWM**: geração de um pulso PWM no pino do PIC. Pode gerar interrupção. Utiliza os temporizadores do PIC para geração da base de tempo (no PIC18F4550 o modo PWM é vinculado ao Timer2 e seus registradores envolvidos – ver datasheet e exemplos nos fóruns relacionados).

Requisitos: Estudar o funcionamento do kit PIC Genios por meio da documentação indicada anteriormente, bem como estudar o módulo CCP (Capture/Compare/PWM) e os registradores associados à configuração PWM no datasheet do PIC18F4550, além das bibliotecas PWM fornecidas pelo compilador MikroC Pro for PIC (consultar o help do software). Em seguida, desenvolver um programa em linguagem C para atender aos requisitos abaixo.

- **Parte 1**: Utilizar 5 chaves para (quando pressionadas) controlar a velocidade da ventoinha via PWM, nos seguintes valores de duty cycle (o valor do duty cycle selecionado deverá ser exibido no display LCD):
    - 0%
    - 25%
    - 50%
    - 75%
    - 100%
- **Parte 2**: O programa também deverá ler um sensor infravermelho para calcular a velocidade de rotação da ventoinha em RPM (rotações por minuto). Para isso, utilizar os temporizadores do PIC18F4550 no modo contador de eventos. O sensor será conectado ao pino de entrada T13CKI (pino RC0). Neste modo, o timer (1 ou 3) não contará o tempo baseado no clock do sistema, mas será incrementado com cada evento de borda gerado pelo sensor, sempre que uma das pás da ventoinha passar pela luz do sensor infravermelho (isto é, o timer será usado no modo contador de eventos). A ventoinha possui 7 pás, portanto, cada volta completa será registrada após 7 pulsos. Isso significa que, após 7 eventos de borda, teremos uma volta da ventoinha. Usar interrupções do timer para garantir a contagem dos pulsos com precisão. O valor do RPM (rotações por minuto – ou seja, número de voltas que teremos da ventoinha por minuto), que pode ser calculado por meio de um timer, também deverá ser exibido no display LCD.

Testes e Validação: compilar o programa e carregar o **firmware** na placa virtual PIC Genios do simulador **PIC Sim Lab** para testar o funcionamento do controle de velocidade da ventoinha. Verificar se a funcionalidade do sensor de velocidade infravermelho está operando corretamente na versão virtual da placa (caso não esteja, testar apenas a Parte 1 e validar a Parte 2 no Simul IDE).

Validação no SimulIDE: monte um circuito com o PIC18F4550, cinco chaves (usar switches e evitar *push buttons*) e um LED para representar a ventoinha (o brilho do LED simula a variação de velocidade de 0 a 100% - ajustar o programa para normalizar o *duty cycle* caso necessário). Utilize um gerador de pulso para simular o sensor infravermelho. Cada pulso gerado representará uma pá da ventoinha, incrementando o Timer conforme configurado. Exibir *duty cycle* e RPM no display LCD e verificar o sinal PWM e as variações de ciclo de trabalho por meio de osciloscópio virtual que o simulador disponibiliza (conectá-lo na saída PWM).

Entrega: arquivos fontes e documentação com programa comentado, circuitos/diagramas e prints da simulação realizada, demonstrando o funcionamento (se preferir, enviar um vídeo).