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.
- Reflexão sobre as decisões de projeto envolvendo escolha de microcontroladores, recursos de hardware e software, otimização energética e desafios de implementação.

Requisitos do projeto

Parte 1 - Projeto usando a placa ESP32 Devkit (Realizar como tarefa usando simulador Wokwi – e testar na placa durante na última aula)

Implementar um projeto para controle PWM e comunicação serial 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 do ESP32 por meio da Espressif IDE (software Eclipse desktop instalado nos PCs do laboratório), ou Arduino IDE (disponível nos computadores do laboratório), ou por meio da plataforma VSCode (extensão da Espressif IDE ou PlataformIO), ficando a critério do aluno qual framework usar. Com base na documentação e exemplos disponibilizados na página da biblioteca LED Control PWM (LEDC), elaborar um programa para realizar a modulação de brilho 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);
- 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 e diferente a cada cor do LED RGB (por exemplo: R = incremento*2; G = incremento; B = incremento*3 – outra lógica poderá ser usada desde que o incremento seja aplicado com valores distintos em cada cor).
- Exibir uma mensagem no Monitor Serial/terminal (UART – baud: 115.200) que indique o valor de incremento e duty cycle.
- Tarefa: Elaborar previamente o programa e testar/simular o projeto no Wokwi (o qual dispõe de LED RGB). OBS**:** não será necessário a entrega da parte realizada no Wokwi. No entanto, sua realização é obrigatória, de modo que, na aula subsequente, o programa já esteja funcional, restando apenas gravar na placa e testar o funcionamento, montando o circuito do LED na protoboard e realizando as devidas ligações. Não haverá tempo suficiente durante a aula para

desenvolver todo o programa sendo, portanto, obrigatório que ele seja previamente implementado no Wokwi, possibilitando a resolução de eventuais problemas de funcionamento antes da aula prática. Durante a aula, no laboratório, deve-se focar em montar o circuito da Figura 1 (protoboard e componentes serão disponibilizados), fazendo as ligações na GPIO do ESP32 DevKit e em demonstrar o funcionamento do programa. O não atendimento deste critério acarretará desconto de pontuação ou desconsideração da Parte 1.

- **Desafio proposto 1 (opcional - pontuação diferenciada):** tentar implementar o controle do duty cycle de cada cor do LED RGB (ou escolher uma delas) via comunicação sem fio no programa, a ser testado na placa durante a aula. A implementação pode ser realizada por meio de um aplicativo de terminal Bluetooth comum disponível em smartphone, de forma a permitir que se envie valores numéricos que serão interpretados como incrementos para o duty cycle de cada canal (R, G, B) pelo programa por meio do módulo Bluetooth da ESP32. A implementação é opcional e faz parte do desafio que o/a aluno(a) estude por conta própria o uso de bibliotecas de comunicação serial Bluetooth com ESP32, a integração desse recurso ao programa base e a realização da interação prática via aplicativo de terminal no celular, visando explorar recursos mais avançados disponíveis em sistemas embarcados.

Formato de entrega da Parte 1: Apresentar em um documento o programa desenvolvido com as partes principais devidamente comentadas. Apresentar fotos da montagem prática (microcontrolador e circuito montado na protoboard com modulação de cores do LED RGB; prints do monitor serial.

Parte 2 – Aplicação final de escolha do(a) aluno(a) (**Realizar somente no Wokwi**).

A aplicação final deverá envolver o uso de controle PWM para o comando de um motor, podendo incluir controle de movimento, rotação, velocidade ou posicionamento em motores de passo, servo, motor DC, entre outros. A escolha deve ser compatível com os itens disponíveis no Wokwi, permitindo que o projeto seja totalmente simulado neste aplicativo. Não será necessário a implementação física, devido à ausência de componentes no laboratório. O projeto deve ser diferente do programa de Servo Motor solicitado na Atividade Semanal 11. Outras funcionalidades e recursos podem ser implementados, a critério do aluno, considerando que se trata de um projeto final. Ou seja, o ideal é que sejam utilizados todos os recursos estudados ao longo do semestre, além de explorar componentes e funcionalidades disponíveis no Wokwi, como sensores (temperatura, acelerômetros etc.), atuadores, Buzzers, uso de interrupções de GPIO para botões (se pertinente), integração com outros módulos como temporizadores/contadores, ADC/DAC etc. Para a parte de controle PWM, recomenda-se utilizar a biblioteca MCPWM (via Arduino ou IDE nativa). Além disso, é obrigatório incluir algum tipo de comunicação serial e exibição dos resultados. Por exemplo: Exibição em um display OLED (via comunicação serial I2C) e/ou apresentação de dados no monitor serial ou terminal via UART.

Formato de Entrega da Parte 2: Apresentar em documento o programa desenvolvido, destacando as principais partes do código devidamente comentadas. Incluir uma discussão textual explicando o projeto escolhido e os principais conceitos envolvidos, bibliotecas utilizadas, além de outros recursos eventualmente empregados (como sensores, interrupções, conversores ADC/DAC, timers etc.) explicando como e por que cada um foi utilizado. Apresentar o diagrama e fotos da montagem prática (microcontrolador e circuito montado no Wokwi), com prints da simulação realizada mostrando, por exemplo, valores no display, monitor serial etc. Por fim, incluir na documentação uma discussão sobre os seguintes aspectos: Justificativa da escolha e do uso adequado de recursos de hardware (por que seria necessário usar um microcontrolador de 32 bits; quais os impactos e diferenças em relação a um de 8 bits; quando ou não utilizar); e qual o impacto do consumo de energia e formas de otimização desse consumo em sistemas embarcados quando se considera microcontroladores de 32 bits.

Entrega final:

- Apresentar as Partes 1 e 2 em único arquivo PDF ou link para documentação em repositório no GitHub (enviar o link por meio de um arquivo).
- Enviar separadamente os códigos fonte dos programas desenvolvidos nas Partes 1 e 2. Os scripts podem ser enviados 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 (neste caso, enviar link para o repositório).
- 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 ou monitores.

Configurações

Figura 1 - ESP32 Pinout

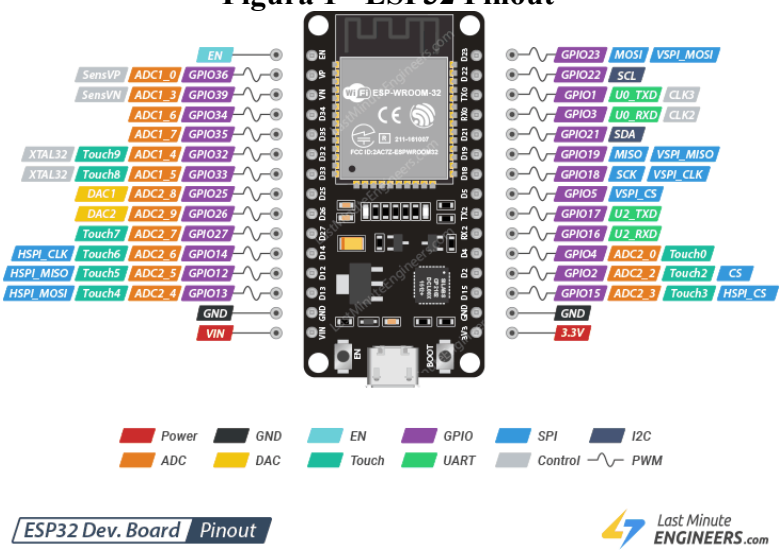


Figura 2 - Ligação do LED RGB de catodo comum (escolher 3 pinos da ESP e fazer as ligações no Wokwi - não esquecer resistores)

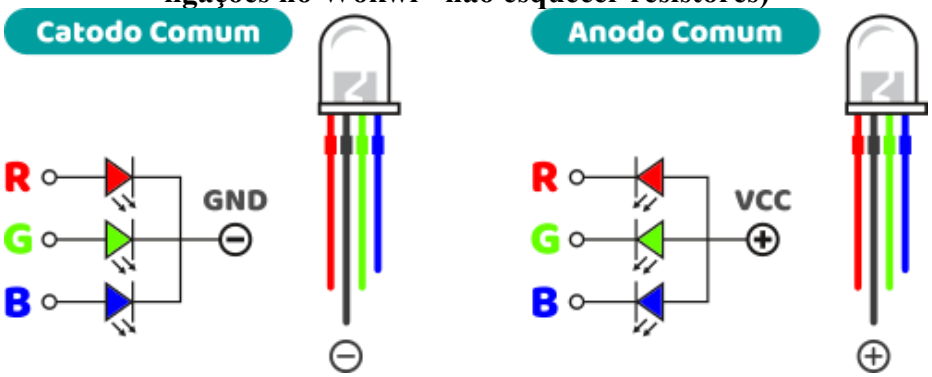


Figura 3 - Wokwi - ESP32 e Servo Motor

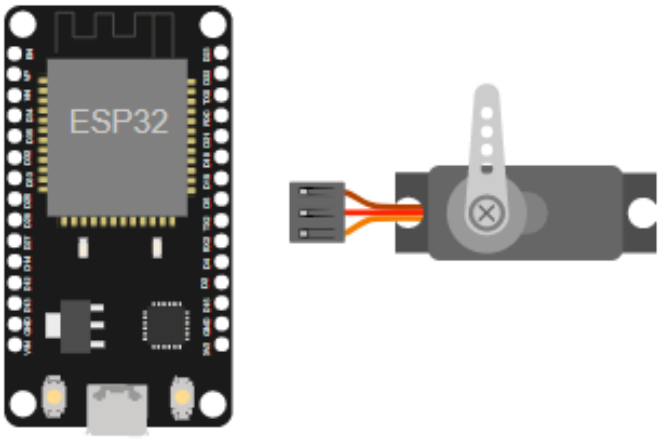
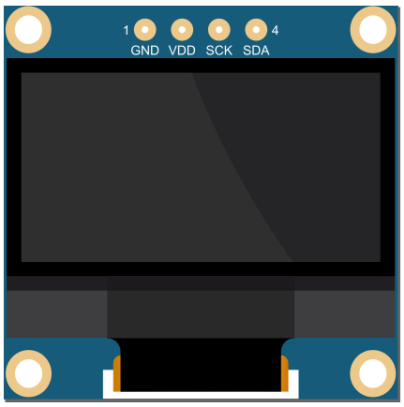


Figura 4 - Configurações do Display OLED: GND, VDD, SCK, e SDA



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	**1**
Correção lógica do programa: atendimento ao enunciado, uso das bibliotecas solicitadas; Projeto implementado na prática (Parte 1); Aplicação final desenvolvida no Wokwi (Parte 2) e atendimento a documentação solicitada	**8**

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

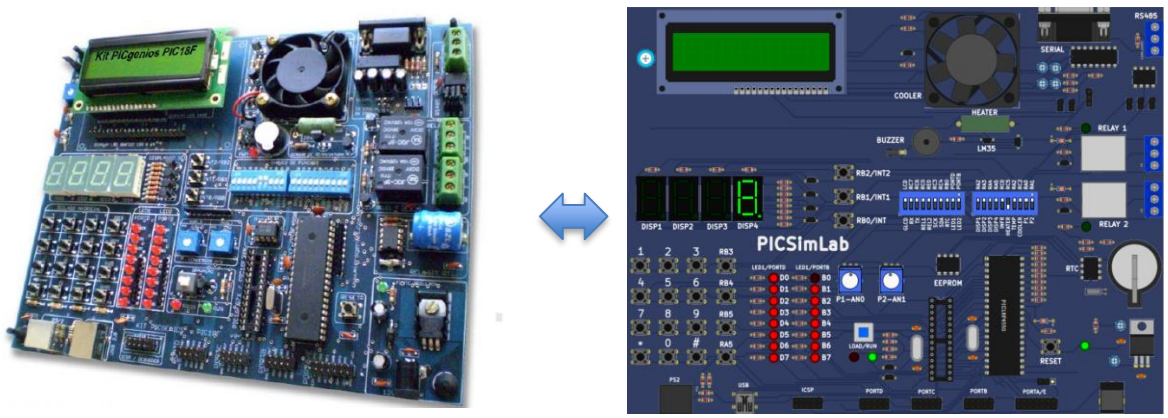
Esta proposta trata-se também de um desafio (opcional) já que o recurso PWM não foi especificamente explorado no PIC. O funcionamento do PWM no PIC, no entanto, segue conceitos semelhantes aos já abordados nos projetos anteriores, como I/O digital, timers, interrupções e ADC. Ou seja, a utilização do 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 realizado nas aulas anteriores 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.

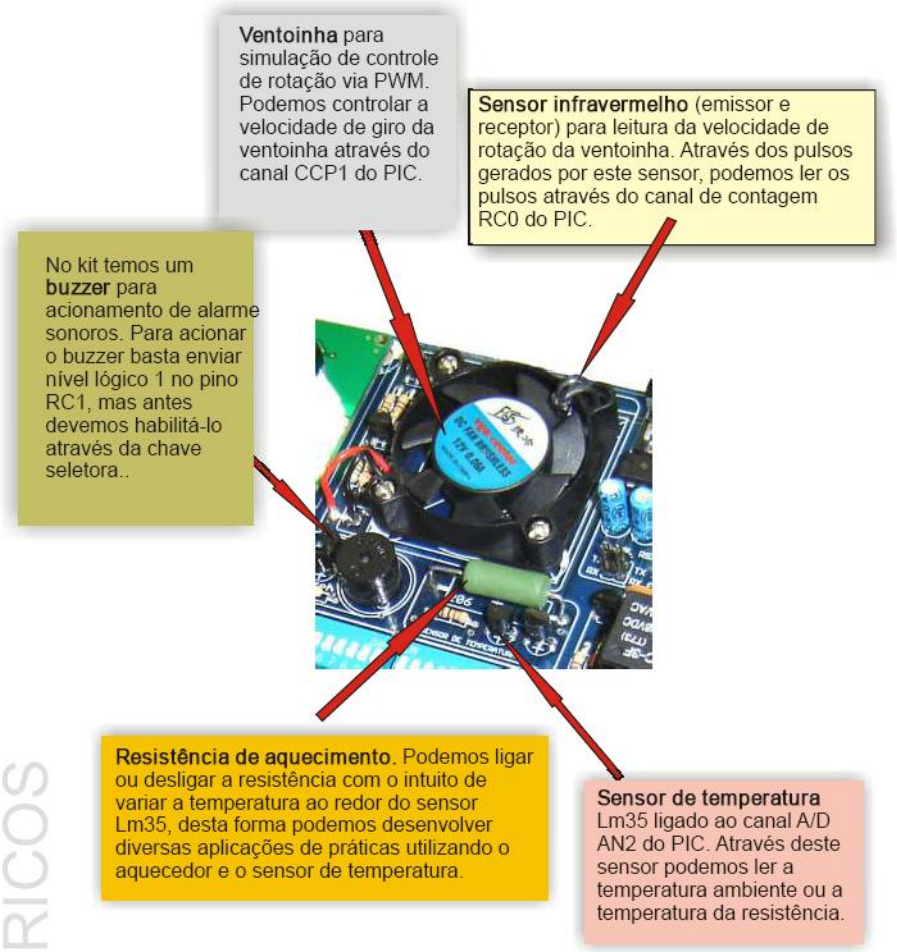
Hardware: O projeto deve ser baseado na placa PIC Genios (e não no kit EasyPIC v7 usado nos projetos anteriores) conforme ilustrado na Figura 5 (consulte o manual completo da placa aqui). Esta placa é virtualizada no simulador PIC Sim Lab (disponível para download aqui), que permite carregar o arquivo 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



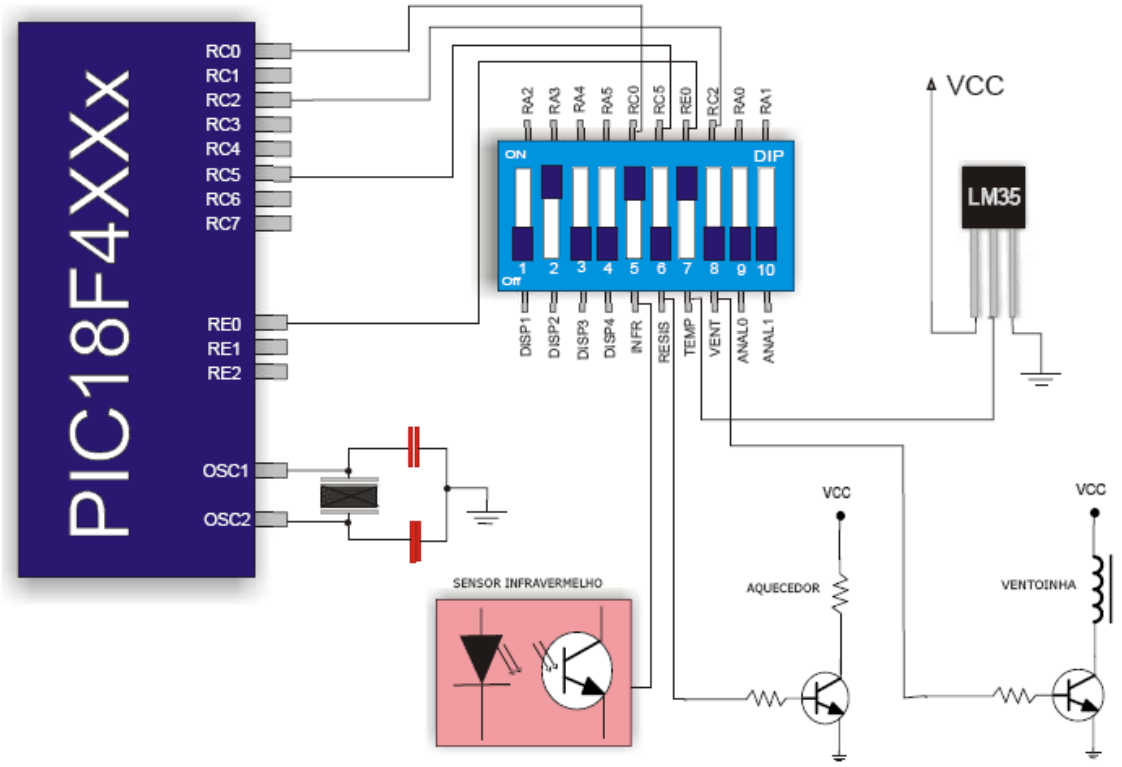
Conforme ilustrado na Figura 7, a ventoinha está conectada ao pino RC2 (canal PWM) do PIC18F4550, que será controlado para ajustar a velocidade da ventoinha (saída). O sensor infravermelho está conectado ao pino RC0 (T13CKI), que está vinculado ao Timer1 ou Timer3 do PIC. Este pino será configurado como 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).

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).

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)

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).