Universidade de Brasília

Faculdade de Ciências e Tecnologias em Engenharia

Engenharia Eletrônica

Relatório Final de Projeto

Desenvolvimento de Firmware para Elevador Inteligente com Algoritmo SCAN (PIC16F1827)

Grupo 1:

Arthur Pinheiro Marinho – 231011112

Fernando de Melo Colli – 231026349

Gabriel Celestino de Almeida – 231027079

Professor: Guillermo Alvarez Bestard

Disciplina: Eletrônica Embarcada

Brasília, DF

Dezembro de 2025

1 Introdução

O controle de sistemas eletromecânicos de transporte vertical, como elevadores, apresenta desafios significativos em termos de lógica de escalonamento, tempos de resposta e segurança operacional. Sistemas tradicionais baseados em lógica sequencial simples (First-In, First-Out) tendem a ser ineficientes energeticamente, pois realizam viagens completas para atender uma única chamada, ignorando solicitações intermediárias.

Este relatório descreve o desenvolvimento, a arquitetura de software e a validação de um firmware para um sistema de controle de elevador de 4 andares, utilizando o microcontrolador PIC16F1827. O objetivo central do projeto foi a implementação do algoritmo de escalonamento SCAN (também conhecido como "algoritmo do elevador"), que prioriza requisições no sentido do deslocamento atual. Essa abordagem permite que o elevador atenda múltiplos passageiros em uma única viagem (lógica de "Carona"), otimizando o percurso e reduzindo o desgaste do motor.

O sistema foi projetado de forma modular, integrando drivers de baixo nível para controle de hardware (PWM, ADC, Timers), interfaces de comunicação externa (UART/Bluetooth e Display SPI) e uma lógica de controle robusta baseada em Máquina de Estados Finitos (FSM). A validação foi realizada em ambiente de simulação (MPLAB X), exigindo adaptações específicas no código para emular o comportamento físico em tempo real.

2 Arquitetura e Divisão de Atividades

O desenvolvimento do firmware seguiu uma arquitetura em camadas (Layered Architecture), dividida entre os membros do Grupo 1 para simular um fluxo de trabalho de engenharia profissional.

2.1 Camada de Hardware (HAL) - Arthur Pinheiro Marinho

Esta camada é responsável pela abstração dos periféricos do microcontrolador, garantindo que a lógica superior não precise manipular diretamente os registradores. As principais implementações foram:

- **Controle de Tração (PWM e Ponte H):** Foi configurado o módulo *Enhanced PWM* (Módulo CCP3) com resolução de 10 bits. O controle de velocidade utiliza a modulação por largura de pulso para acionar uma Ponte H (modelo TB6612FNG), permitindo controle suave de aceleração. A reversão de sentido é controlada pelo pino de direção (RA7), definindo a polaridade da tensão no motor.

- **Sensoriamento de Posição:** A leitura dos sensores de fim de curso (S1 a S4) foi implementada de forma híbrida para aproveitar os recursos do PIC16F1827. Os sensores dos andares extremos (S1/S2) utilizam entradas digitais com interrupções por mudança

de estado (IOC - *Interrupt On Change*). Já os sensores intermediários (S3/S4) são lidos através dos Comparadores Analógicos internos (C1OUT e C2OUT), permitindo a detecção precisa mesmo com sinais analógicos ruidosos.

- **Telemetria Física:** O Timer 0 foi configurado no modo contador de pulsos externos (Counter Mode) para ler o encoder óptico acoplado ao eixo do motor. Isso permite o cálculo de posição em milímetros e velocidade instantânea. Adicionalmente, o ADC lê periodicamente um sensor térmico (LM35) no canal AN2 para monitorar a temperatura da ponte de potência.

2.2 Camada de Interface (HMI) - Gabriel Celestino de Almeida

Responsável pela comunicação entre o sistema embarcado e o mundo externo, implementando protocolos de visualização e comando.

- **Interface Visual (SPI):** Desenvolvimento do driver para o controlador de display MAX7219. A comunicação via SPI (*Serial Peripheral Interface*) foi otimizada para atualizar uma matriz de LEDs 8x8. Foram criadas tabelas de busca (`LUT_Andar` e `LUT_dir`) no arquivo `comm.c` para renderizar o número do andar atual e animações de setas (subindo/descendo), além de indicar visualmente as chamadas pendentes na linha inferior da matriz.

- **Comunicação Serial (UART):** Implementação de um *parser* de comandos via Bluetooth utilizando a EUSART a 19200 baud. O protocolo define pacotes no formato `$OD\r` (Origem, Destino), que são decodificados e inseridos na fila de solicitações. Também foi implementado o envio periódico de telemetria em formato CSV (Posição, Velocidade, Temperatura) a cada 300ms para monitoramento em terminal remoto.

2.3 Camada de Aplicação e Lógica - Fernando de Melo Colli

A camada de aplicação, desenvolvida pelo integrante responsável pela integração, atua como o "cérebro" do sistema, orquestrando os drivers dos demais integrantes.

- **Integração (Super Loop):** O arquivo `main.c` foi estruturado como um loop infinito não-bloqueante. Tarefas críticas (como leitura de sensores) rodam a cada ciclo (10ms), enquanto tarefas lentas (telemetria UART) utilizam divisão de tempo (*Time Slicing*) para não travar o controle do motor.

- **Algoritmo SCAN:** Diferente de uma fila FIFO tradicional, foram implementados dois vetores booleanos de memória: `chamadas_subida[4]` e `chamadas_descida[4]`. A lógica verifica, a cada andar, se existe uma chamada no vetor correspondente à direção atual.

  - *Lógica de Carona:* Se o elevador está subindo e há uma chamada de subida no andar atual, ele para imediatamente para atender o passageiro extra.
  - *Look-ahead:* Se não há chamada no andar, a função `Existe_Chamada_Acima` verifica se o motor deve continuar ligado ou iniciar a frenagem.

- **Máquina de Estados (FSM):** O sistema opera em 5 estados exclusivos para garantir segurança:

  1. **PARADO:** Aguarda solicitações e decide a direção inicial.
  2. **SUBINDO / DESCENDO:** Controla o PWM e verifica sensores.
  3. **ESPERA_PORTA:** Um temporizador lógico de 200 ciclos (2 segundos) que simula o embarque.
  4. **REVERSÃO:** Estado de segurança crítica. Impõe um atraso de 500ms com o motor desligado antes de inverter a marcha, protegendo a Ponte H contra correntes de curto-circuito (Shoot-through).

3 Metodologia de Simulação

A validação do firmware foi realizada utilizando o simulador integrado do MPLAB X. Como o simulador não possui física (gravidade, inércia ou sensores reais), foi necessário desenvolver uma metodologia de "injeção de estímulos"e realizar adaptações no código fonte.

3.1 Adaptações Técnicas no Código

Para que a simulação fosse viável e representasse fielmente o comportamento temporal, duas correções de engenharia foram aplicadas:

1. **Otimização de Latência no Timer (motor.c):** A versão inicial da função `SENSORES_CalcularVelocidade` realizava uma média móvel com atrasos bloqueantes (`delay_ms`) dentro da interrupção do Timer 4. Isso causava o travamento do processador no simulador, pois o tempo de execução da interrupção excedia o período do timer. A função foi reescrita para realizar amostragem única a cada 100ms (Timer 4 configurado com PR4=0xEF e Prescaler 1:64), eliminando loops de espera e liberando a CPU para processar a lógica principal no `main.c`.

2. **Bypass de Hardware SPI (comm.c):** O simulador do MPLAB não emula o tempo de transmissão do periférico SPI, fazendo com que a flag de conclusão (`SSP1IF`) nunca fosse setada pelo hardware virtual. Para a simulação, as linhas de espera ativa (o comando `while(!PIR1bits.SSP1IF)`) foram comentadas temporariamente no arquivo `comm.c`, permitindo que a lógica de visualização rodasse sem travar o sistema em loops infinitos.

3.2 Procedimento de Teste

Os testes foram conduzidos manipulando diretamente a memória RAM do microcontrolador através da ferramenta *Watches*:

- **Estímulo de Entrada:** A variável `andar_atual` foi alterada manualmente passo-a-passo para simular o movimento físico do elevador passando pelos sensores. Os vetores de chamada (`chamadas_subida`) foram forçados para `true` para simular o pressionamento dos botões.
- **Verificação de Saída:** O sucesso do teste foi determinado pela observação da transição correta da variável `estado_atual` (de `SUBINDO` para `ESPERA`) e pelo "zeramento"automático das flags de chamada, indicando que o pedido foi atendido. Adicionalmente, verificou-se o acionamento do driver de motor durante os estados de movimento.

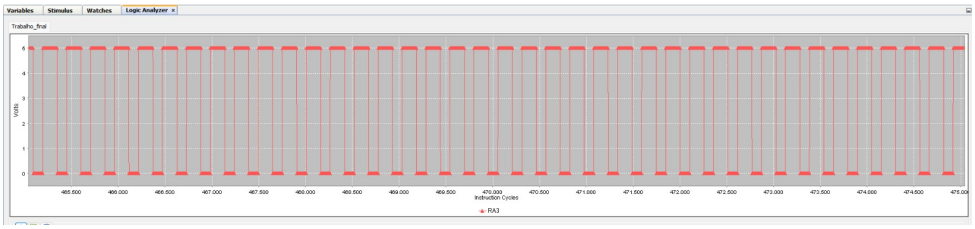

Figura 1 - Verificação do sinal de controle: Gráfico representativo do PWM gerado quando o sistema entra no estado `SUBINDO` ou `DESCENDO`, validando o acionamento da Ponte H com Duty Cycle de aproximadamente 60%.

4 Análise dos Resultados Obtidos

Foram executados dois cenários de teste fundamentais para comprovar a eficiência do algoritmo SCAN.

4.1 Cenário 1: Eficiência de Percurso

Objetivo: Demonstrar que o elevador é capaz de otimizar o transporte atendendo passageiros no meio do caminho. **Configuração:** O elevador partiu do Térreo (0) com destino final ao 3º andar. Durante o trajeto de subida, foi inserida uma solicitação de subida no 1º andar (`chamadas_subida[1]=1`).

Análise do Resultado: Conforme observado na Figura 2, ao alterar manualmente a variável `andar_atual` para 1, o sistema reagiu imediatamente. O `estado_atual` transicionou de `0x01` (SUBINDO) para `0x03` (ESPERA_PORTA). Simultaneamente, a variável `chamadas_subida[1]` foi alterada para 0, confirmando que a lógica atendeu a solicitação intermediária. Isso comprova a economia de energia, evitando que o elevador subisse até o topo para depois voltar buscar o passageiro.

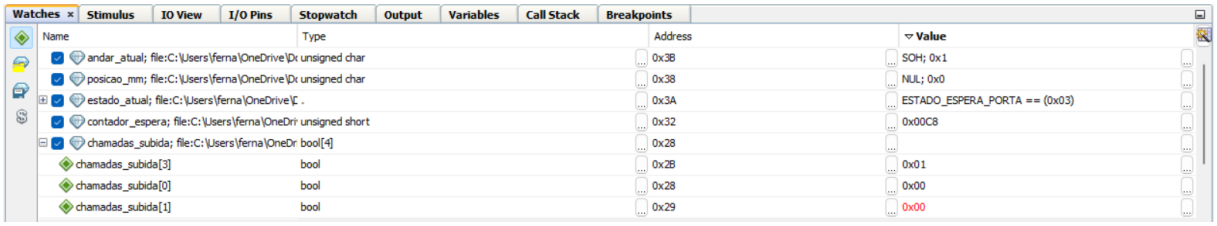

Figura 2 - Simulação do Cenário 1: O *Watch* mostra o elevador parado no Andar 1 (Estado 0x03) para atender uma carona. Note que a chamada do andar 1 foi zerada (atendida), mas a do andar 3 permanece ativa.

4.2 Cenário 2: Prioridade de Sentido

Objetivo: Verificar a robustez do algoritmo em situações de conflito de direção. **Configuração:** O elevador estava em movimento de subida do andar 0 para o 3. Ao passar pelo 2º andar, foi inserida uma chamada de **descida** (`chamadas_descida[2]=1`).

Análise do Resultado: Na Figura 3, observa-se que, ao atingir o andar 2 (`andar_atual=2`), o `estado_atual` permaneceu inalterado em `0x01` (SUBINDO). A lógica SCAN identificou corretamente que atender a chamada de descida naquele momento seria ineficiente. O sistema priorizou levar os passageiros atuais até o topo (andar 3) e manteve a chamada do andar 2 pendente para o trajeto de retorno.

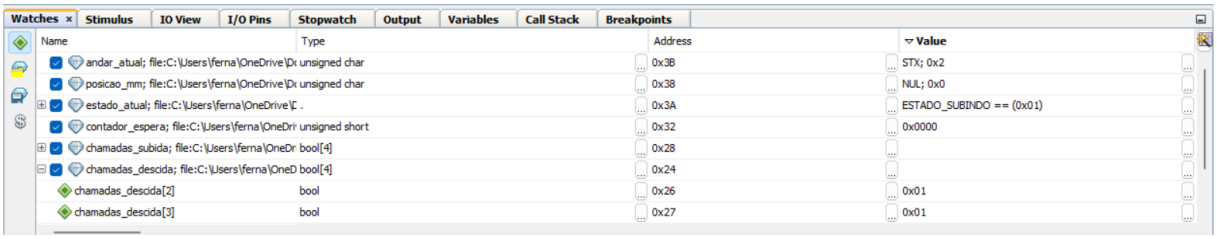

Figura 3 - Simulação do Cenário 2: O sistema ignora a chamada de descida no Andar 2 (Flag True) e mantém o motor em movimento de subida (Estado 0x01), otimizando o fluxo.