Faculdade de Engenharia Elétrica e de Computação - EA871 (R)

Projeto de integração – Aplicativo de Datilografia

Aluno: Mateus Henrique Silva Araújo (184940)
Data: 23/06/2023

1) Proposta e descrição de funcionalidades:

Para este projeto de integração final, propõe-se o desenvolvimento de um teste de destreza em forma de aplicativo, cujo objetivo será não só colocar à prova, mas também aprimorar a habilidade de digitação do usuário.

O aplicativo em questão decorrerá da seguinte forma: ao longo de 8 níveis (com dificuldade crescente), o usuário deverá, por meio da digitação (capturada a partir de um terminal de comunicação serial com o computador hospedeiro utilizando protocolo UART) em ordem dos caracteres das sequências de letras apresentadas no LCD do *Shield FEEC*, limpar por completo o *display* desse dispositivo. No início de cada nível, a parte inferior do *display* será preenchi com caracteres aleatórios que variam nos intervalos [A-Z] e [a-z]. Enquanto o jogador não for capaz de completar sua tarefa, periodicamente (com um intervalo de tempo entre a inserção de duas letras que diminui com o avanço dos níveis), mais letras aleatórias serão adicionadas à sequência atual. O teste termina se o usuário completar todos os 8 níveis de dificuldade ou caso, durante um desses níveis, o LCD for completamente preenchido.

Além disso, com intuito de facilitar a visualização do nível no qual o jogador se encontra, os LEDs vermelhos associados ao *Latch 74573* do *Shield FEEC* serão empregues. Tal dispositivo será configurado de modo que, a cada nível completo, um novo LED se acenderá (da esquerda para direita). Outro ponto de destaque é que, visando expandir a interação sensorial da aplicação, um *buzzer* será utilizado para gerar notas musicais com tonalidades que variam conforme o *display* LCD é preenchido (a cada 1/4 de preenchimento do *display*, aumenta-se o tom da nota emitida).

Por fim, vale destacar que, entre as transições de níveis, deverão ser apresentadas mensagens que comuniquem ao usuário sua progressão e preparem-no para o próximo teste. Em especial, no início de cada nível, será realizada uma contagem regressiva de 5 segundos para aumentar a atenção do usuário ao início do teste.

2) Periféricos e módulos utilizados:

Na implementação deste projeto, serão utilizados os seguintes periféricos: *display* LCD disponível no *Shield FEEC* (empregue para comunicação com o usuário e como interface gráfica do aplicativo), LEDs vermelhos associados ao *Latch 74573* do *Shield FEEC* (utilizados para indicar o nível de dificuldade no qual o usuário se encontra), UART (para possibilitar a comunicação com o teclado do computador hospedeiro) e *buzzer* (com intuito de indicar sonoramente a progressão do usuário em um nível).

Os principais módulos empregues para desenvolver as funcionalidades desejadas serão: UART0 (para configurar e permitir a conexão UART), PORTC/GPIOC (para a comunicação com os periféricos do *Shield FEEC*), TPM1 (produção do sinal de PWM do

buzzer e geração de valores aleatórios), TPM0 (para implementação de temporizadores), bem como NVIC (para configuração e tratamento de interrupções).

Para maiores detalhes quanto aos dispositivos empregues, veja o diagrama de componentes apresentado na figura 1.

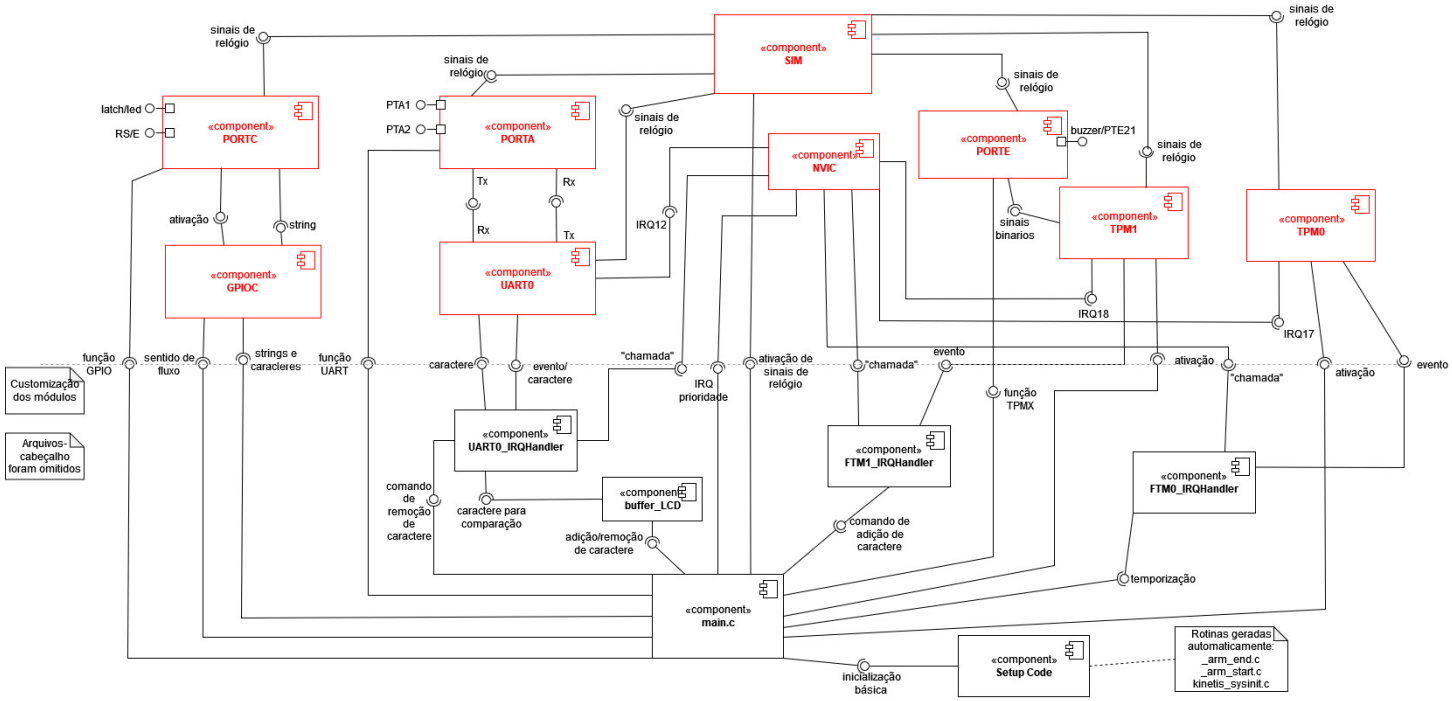


***Figura 1**: Diagrama de componentes do projeto "Aplicativo de Datilografia"*

3) Máquina de Estados da aplicação:

Uma descrição detalhada dos estados da máquina implementada neste projeto, bem como das regras de transições existentes entre eles é feita abaixo. O diagrama de máquina de estado corresponde à descrição feita é exposto na figura 2.

- **Estado ESPERA_INICIO:**
  - Exibe no LCD uma mensagem de introdução, requisitando ao usuário que aperte a tecla *enter* para iniciar o jogo;
  - Produz o valor a ser escrito no *Latch 74573* correspondente ao nível 1;
  - Aumenta o nível atual do aplicativo (que inicia no nível zero);
  - Espelha os caracteres digitados pelo usuário no terminal até que ele envie o caractere '\r';
  - **Regras de transição:**
    1. Chaveia para o estado **PREPARA_LCD** quando recebe o caractere '\r' pelo terminal UART.

- **Estado PREPARA_LCD:**
  - Atualiza os LEDs vermelhos para corresponder ao nível atual do jogo;
  - Gera a pilha (*buffer*) associado ao LCD com 16 letras do alfabeto (intervalo A-Z ou a-z) geradas aleatoriamente;
  - Exibe, no *display* do LCD , uma mensagem indicando o início do nível atual em 5 segundos;

- Configura o temporizador para gerar um intervalo de espera de 5 segundos, no qual será realizada uma contagem regressiva;
- **Regras de transição:**
  1. Chaveia para o estado **CONTAGEM_REGRESSIVA** após configurar o intervalo de tempo dessa contagem.

➢ **Estado CONTAGEM_REGRESSIVA:**

- Este estado deve implementar uma contagem regressiva que ocorrerá mostrando a passagem dos segundos contados no *display* do LCD;
- A cada segundo, reduz o número de segundos que se faltam para o início do nível atual apresentado no *display*;
- Quando a contagem regressiva finaliza, a primeira linha do LCD é limpa, a sequência aleatória gerada é impressa na segunda linha do *display* (provocando também o acionamento do *buzzer* no tom adequado) e o temporizador é ajustado para gerar os intervalos de adição de letras correspondente ao nível de teste atual;
- **Regras de transição:**
  1. Chaveia para o estado **EXECUTA_LVL** após fim da contagem regressiva, impressão da pilha de caracteres, acionamento do *buzzer* e ajuste do temporizador;

➢ **Estado EXECUTA_LVL:**

- Aguarda o esvaziamento ou preenchimento completo do *buffer* associado ao LCD;
- Espelha os caracteres digitados pelo usuário no terminal do computador hospedeiro e, caso seja enviado um caractere que corresponde ao último elemento da sequência aleatória, chaveia para o estado **APAGA_LETRA** para eliminá-lo;
- Periodicamente, com um intervalo de tempo que diminui conforme os níveis aumentam, chaveia para o estado **ADICIONA_LETRA** para inserir uma letra na sequência de caracteres apresentadas no *display*;
- **Regras de transição:**
  1. Se o caractere correspondente à primeira letra exibida no LCD for digitado, chaveia para o estado **APAGA_LETRA**;
  2. Se o intervalo de tempo associado à dificuldade do nível se esgotar, chaveia para o estado **ADICIONA_LETRA**.

➢ **Estado APAGA_LETRA:**

- Apaga a primeira letra exibida no LCD e desempilha o último elemento do *buffer* associado;
- Ajusta a frequência da nota musical gerada pelo *buzzer*, podendo diminui-la ou mantê-la;
- Se o desempilhamento fizer com que a pilha fique vazia, desabilita o temporizador usado no nível e o som do *buzzer*, chaveando em sequência para o estado **LVL_FINALIZADO;**
- **Regras de transição:**
  1. Se a pilha de letras não estiver vazia, chaveia para o estado **EXECUTA_LVL** após a eliminação do caractere;
  2. Se a pilha de letras estiver vazia, chaveia para o estado **LVL_FINALIZADO**.

- **Estado ADICIONA_LETRA:**
  - Gera um caractere aleatório e o empilha caso haja espaço suficiente no buffer associado ao LCD. Caso contrário, desabilita o temporizador usado no nível e o som do *buzzer*, chaveando em sequência para o estado **LVL_FINALIZADO**;
  - Caso o empilhamento seja bem-sucedido, renderiza a nova letra no LCD e ajusta a frequência da nota musical gerada pelo *buzzer*, podendo aumentá-la ou mantê-la;
  - **Regras de transição:**
    1. Se o empilhamento for bem-sucedido, chaveia para o estado **EXECUTA_LVL**;
    2. Se o empilhamento não ocorrer, chaveia para o estado **LVL_FINALIZADO**.

- **Estado LVL_FINALIZADO:**
  - Caso o nível atual do jogo seja menor do que 8 e a pilha associada ao LCD estiver vazia, renderiza mensagem de parabenização pela conclusão do nível e incrementa a variável associada ao nível de jogo;
  - Caso o nível atual do jogo igual a 8 e a pilha associada ao LCD estiver vazia, renderiza mensagem de parabenização especial pela conclusão do jogo;
  - Caso a pilha associada ao LCD estiver cheia, renderiza uma mensagem de fim de jogo;
  - Nos dois casos de fim de jogo, reinicia a variável associada ao nível de jogo;
  - Em todos os casos, configura um temporizador para gerar um intervalo de tempo de leitura das mensagens produzidas pelo usuário;
  - **Regras de transição:**
    1. Chaveia para o estado **LEITURA** após exibição da mensagem característica, ajuste do nível de jogo e configuração do tempo de leitura;
    2. Caso o *buffer* esteja vazio e o nível de jogo atual seja menor que 8, chaveia para o estado **PREPARA_LCD**;
    3. Caso contrário, chaveia para o estado **ESPERA_INICIO**.

- **Estado LEITURA:**
  - Estado de espera, cuja funcionalidade é gerar um intervalo de tempo para que o usuário leia a mensagem exibida no LCD a qual foi gerada pelo estado anterior;
  - Quando o tempo configurado de espera se esgota, desativa o temporizador e chaveia para o próximo estado;
  - **Regras de transição:**
    1. Se o nível do aplicativo tiver sido reiniciado (igual a zero), chaveia para o estado **ESPERA_INICIO**;
    2. Se o nível do aplicativo não tiver sido reiniciado (diferente de zero), chaveia para o estado **PREPARA_LCD**.

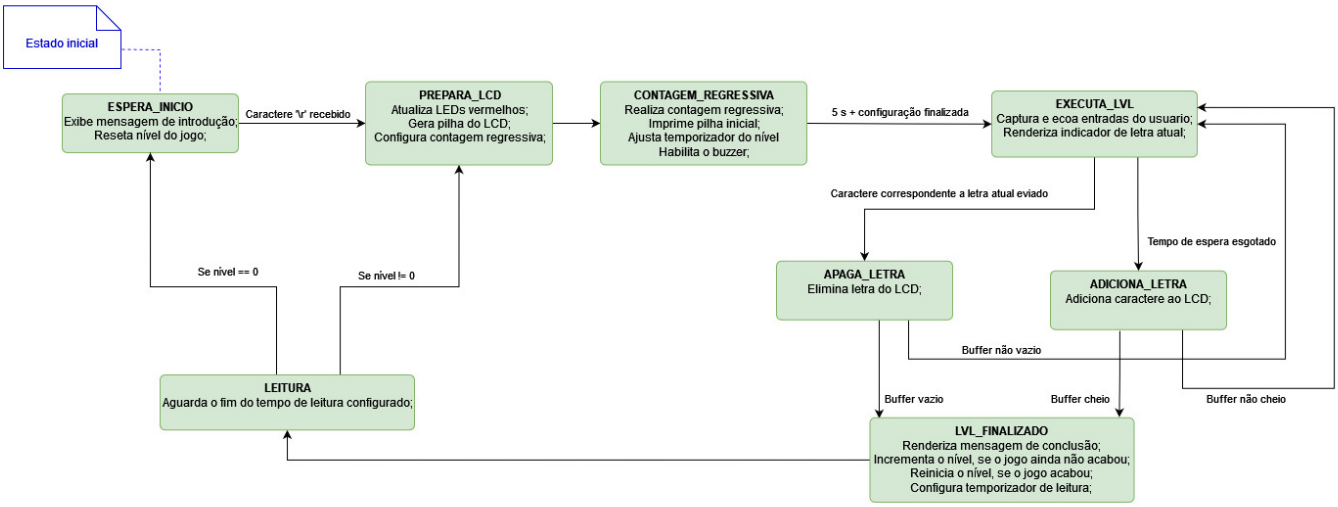


***Figura 2**: Diagrama de máquina de estados do projeto "Aplicativo de Datilografia"*

4) Alterações durante implementação:

Neste tópico, serão tanto destacadas quanto justificadas as diferenças existentes entre a primeira proposta de implementação do projeto e sua versão final. De início, vale destacar que o nome do projeto foi alterado de "Jogo de Digitação" para "Aplicativo de datilografia" devido ao fato do sistema projetado se aproximar mais de uma ferramenta útil para teste e aprimoramento da habilidade de digitação do usuário do que de uma forma lúdica de entretenimento.

Outra diferença marcante existente entre a proposta e o projeto final foi a inserção dos estados **CONTAGEM_REGRESSIVA** e **LEITURA** na máquina de estados do sistema, bem como algumas pequenas mudanças nas funcionalidades de cada estado para que estes se adaptassem ao novo modelo. Tais alterações foram necessárias, pois , no momento de construção da proposta, não se notou que não seria prático nem mesmo coeso a construção dos estados **PREPARA_LCD** e **LVL_FINALIZADO** com todas as funcionalidades que a eles foram delegadas. Dessa forma, parte das funcionalidades antigas destes foram transferidas para aqueles.

Além disso, também destaca-se a alteração do modo de uso do *buzzer* na aplicação. Na proposta, este dispositivo seria empregue de modo a produzir ruídos cuja intensidade (controlada pelo ciclo de trabalho do sinal de PWM enviado ao dispositivo) aumentaria conforme o *display* do LCD fosse preenchido com letras e o usuário se aproxima-se de falhar no teste. Entretanto, os experimentos com esse modo de operação mostraram que a diferença na intensidade do ruído gerado não era adequadamente perceptível pelo usuário, fato o qual pode ser atribuído não só a baixa qualidade do *buzzer* disponível para uso, mas também a pequena variação gerada na adição/remoção de cada letra (que poderia ser imperceptível para a audição do usuário). Dessa forma, para contornar esse empecilho e manter o uso do dispositivo proposto, optou-se por passar a se realizar alterações na tonalidade das notas geradas pelo som do *buzzer* conforme caracteres fossem inseridos ou retirados do *display*. Em especial, para evitar que a alteração das notas não fosse percebida pelo usuário em virtude de um gradiente pequeno e constante, os endereços do LCD foram agrupados em 8 grupos de 4 caracteres e, para cada um desses grupos, uma nota musical foi atribuída (sendo as mais agudas ligadas aos endereços mais próximos do início do *display*).

Por fim, uma mudança de menor porte realizada foi a inserção do módulo TPM0 para facilitar a geração de intervalos periódicos utilizados para implementar os temporizadores

responsáveis por gerar tanto os tempos de contagem regressiva e leitura de mensagens quanto os períodos entre a inserção de duas letras. Outra alteração desse tipo foi o abandono da ideia de se renderizar periodicamente um indicador sobre a primeira letra do *display*. Esta atitude foi tomada, pois, com o aumento dos níveis e a consequente diminuição do intervalo de tempo entre a inserção de um novo caractere no LCD, o indicador passaria a atrapalhar o usuário ao invés de ajudá-lo.

5) Destaques de implementação:

Como destaque do modo de implementação das funcionalidades desejadas para esse projeto, ressalta-se o desenvolvimento de uma estrutura de dados específica ("*BufferPilha_type"*) e rotinas especiais para a sua manipulação (disponíveis no arquivo pilha_LCD.c) com o intuito de não só facilitar a comunicação do programa com o LCD integrado ao *Shield FEEC*, mas também de tornar o código mais simples e coeso.

Ao observar as instruções das rotinas do arquivo mencionado, nota-se que as alterações realizadas na variável que identifica a pilha associada ao LCD têm a possibilidade de causar efeitos diretos nesse dispositivo, o que torna o fluxo de execução principal mais enxuto e compreensível.

Além disso, vale destacar que, com a mesma intenção de simplificar o código da rotina principal, usou-se do arquivo ISR.c para armazenar as variáveis de controle mais importantes.

6) Testes realizados:

Vale ressaltar que durante todo o processo de construção do projeto, testes de unidade foram executados para ratificar separadamente o funcionamento adequado de cada módulo e estrutura utilizado no programa final. Tais testes variaram desde a verificação do comportamento das estruturas de dados implementadas por meio da análise das variáveis em tempo de execução, até testes de desvio de fluxo para as rotinas de tratamento de interrupção desenvolvidas, bem como testes de estresse (forçando situações anômalas para certificar o modo de atuação do sistema). O procedimento adotado foi somente continuar o desenvolvimento do sistema principal após a certificação que a parcela implementada estava funcional.

Nesse sentido, destaca-se que os principais erros identificados durante os testes efetuados estavam relacionados a impressões em endereços inadequados do LCD, cálculo inapropriado do valor a ser escrito nos LEDs do *Latch 74573* e não limpeza da pilha gerada em situações necessárias.

Ademais, uma vez finalizado o projeto, não só vários testes funcionais foram executados, mas também os colegas de sala foram convidados a experimentarem o aplicativo. Tais eventos permitiram confirmar que o sistema se condizentemente com o especificado.

7) Manual do usuário:

Para poder utilizar do aplicativo desenvolvido, o usuário deve primeiro carregar o programa no microcontrolador com o *Shield FEEC* devidamente associado, sendo o *buzzer* ligado ao pino 3 (PTE 21) do header H5 dele. Em seguida, um terminal de comunicação serial com protocolo UART deve ser configurado no computador hospedeiro. As configurações desta conexão são as seguintes: *baud rate* de 38400, caractere de 8 *bits*, com 2 *stop bits* e sem *bit* de amostragem.

Após isso, o usuário já conseguirá interagir com a aplicação, sendo necessário que ele pressione a tecla *enter* para se iniciar a sequência de testes. Feito tal pressionamento, uma

mensagem indicando o nível atual será exibida, uma contagem regressiva será feita na tela do LCD e, após o seu término, a linha inferior do *display* será preenchida com uma sequência de caracteres aleatórios.

A partir deste momento, o usuário deverá digitar a primeira letra exibida na tela (considerando o início dela sendo em seu canto superior direito), fazendo com que ela se apague, até que o LCD seja completamente limpo. Durante esse processo, periodicamente (em um intervalo de tempo que diminui conforme o nível do teste aumenta), serão inseridas mais letras aleatórias no *display*, as quais também deverão ser eliminadas com o intuito de se prosseguir para o próximo nível.

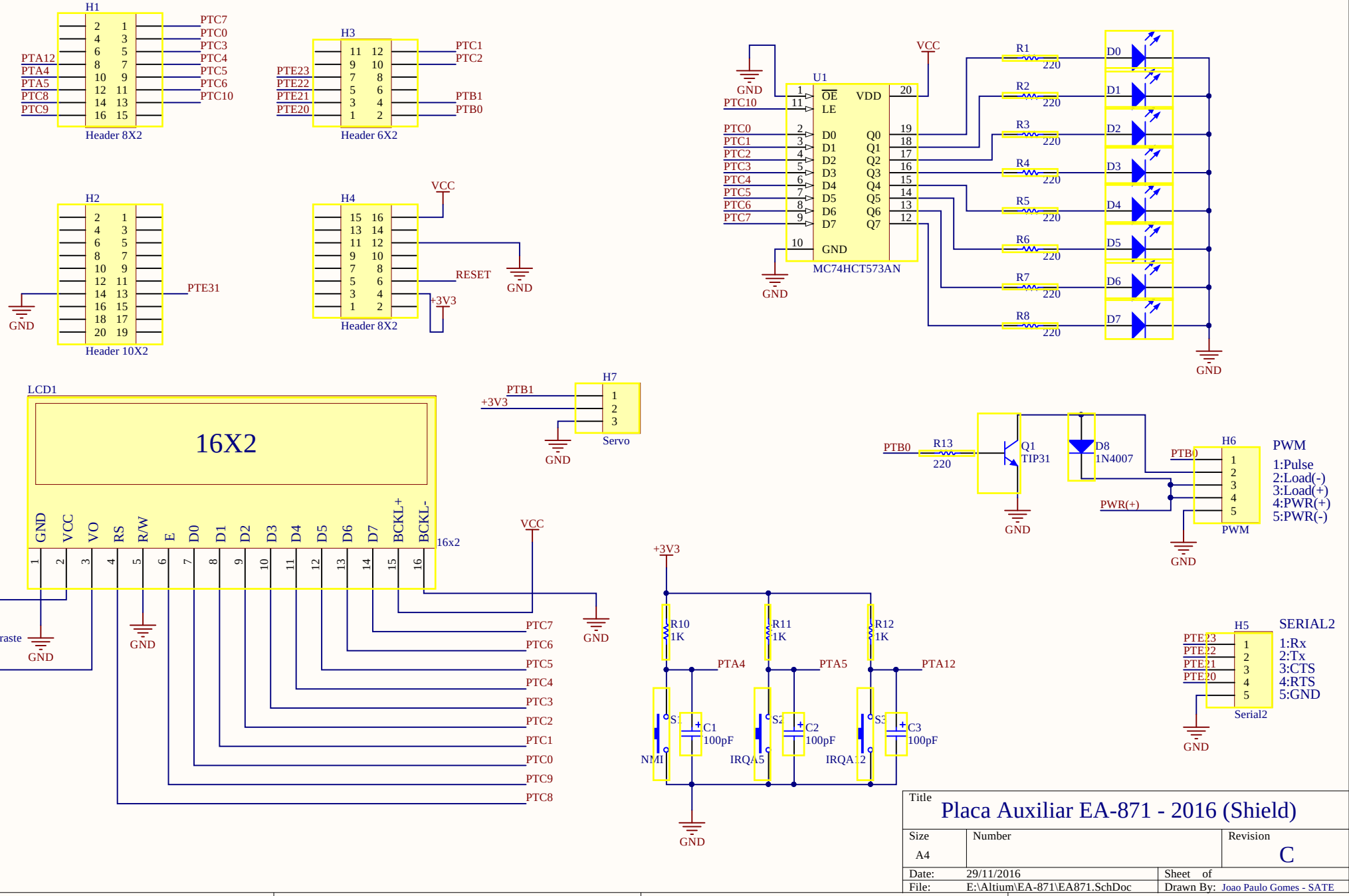
Caso o usuário seja capaz de limpar a tela do LCD por completo, o nível se encerra, uma mensagem de parabenização é renderizada durante alguns segundos e, se o nível completo não for o oitavo, inicia-se, novamente, o preparo para o próximo estágio. Se o nível terminado for o oitavo, uma mensagem de parabenização especial pelo término de todos os testes é renderizada e o aplicativo retorna para o estado inicial. Procedimento análogo a este ocorre se, durante algum dos níveis, o *display* for completamente preenchido, com exceção de que a mensagem exibida nesse caso indica que o teste falhou. Uma vez que o sistema retorna para o estado inicial, o usuário pode reiniciar as sequências do mesmo modo que feito inicialmente.

8) Conclusões:

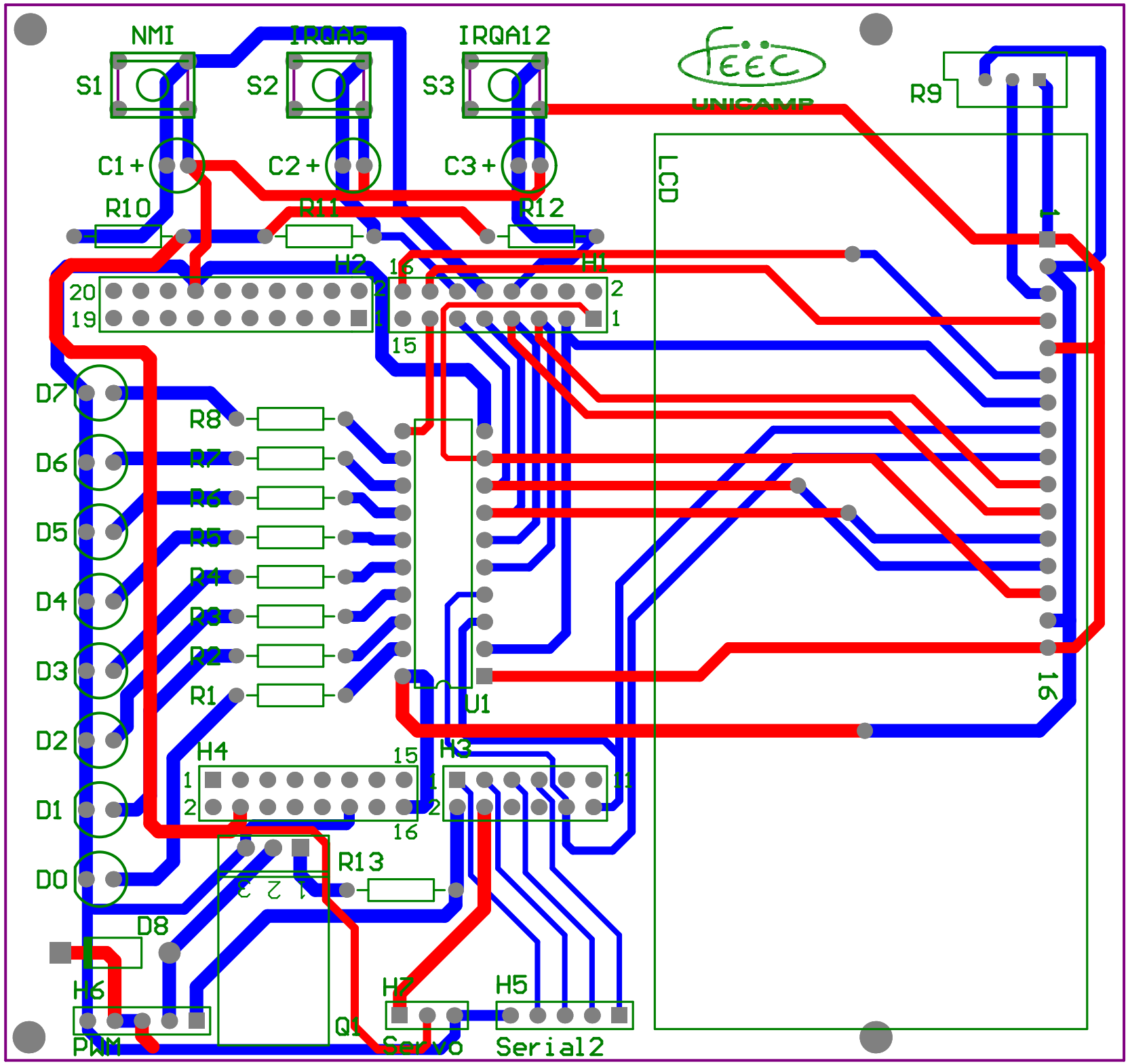
Dessa forma, é possível concluir que o projeto desenvolvido foi bem-sucedido. Afinal, não só os pontos principais e característicos da proposta inicial foram devidamente implementados e testados, mas também os dispositivos estudados durante o semestre tiveram suas funcionalidades exploradas, bem como o resultado obtido foi um aplicativo funcional e consistente.

9) Anexos:

Em anexo, um esquemático do *Shield FEEC* e as conexões realizada para sua comunicação com o microcontrolador foi inserido. Vale ressaltar novamente que o *buzzer* deve ser ligado ao pino 3 (PTE 21) do header H5 deste com o intuito de que a sonorização do aplicativo funcione adequadamente.



Title	Placa Auxiliar EA-871 - 2016 (Shield)	
Size: A4	Number:	Revision: C
Date: 29/11/2016		Sheet of
File: E:\Altium\EA-871\EA871.SchDoc		Drawn By: Joao Paulo Gomes - SATE


NMI
IRQA5
IRQA12
S1
S2
S3
FEEC
UNICAMP
R9
C1+
C2+
C3+
LCD
R10
R11
R12
H2
H1
20
19
16
15
2
1
D7
D6
D5
D4
D3
D2
D1
D0
R8
R7
R6
R5
R4
R3
R2
R1
U1
H4
H3
H6
H7
H5
R13
D8
Q1
PWM
Servo
Serial2