Tradutor do Código da Batida

por Rodrigo Batista de Moraes

Para o meu projeto final de programação embarcada eu precisava fazer um programa destinado a rodar em um PIC18F4520, que usasse múltiplos periféricos diferentes.

Eu decidi fazer um decodificador de mensagens. Mais especificamente, um tradutor do código da batida. Como a parte fundamental do projeto é utilizar os diferentes periféricos da placa, usarei o teclado matricial para ler as teclas, os display 7 segmentos para exibir o números pressionados, o lcd para exibir a mensagem traduzida, e o buzzer para fazer bip ao pressionar os botões.

Cada display 7 segmentos deve ser constantemente operado pela placa, e o teclado deverá ser lido a todo instante para que a interface entre o programa e o usuário seja a mais suave possível. Por isso o programa deve ser capaz de multiplexar essas duas partes.

O código da batida e o quadrado de polybius

O código da batida é um jeito de criptografar mensagens letra a letra de uma maneira bastante simples, usando uma sequência de batidas que vão se traduzir em coordenadas. Já foi usado por prisioneiros que se comunicavam através de batidas nas grades das celas, por exemplo.

A parte mais fundamental do código da batida, é um quadrado de polybius, que basicamente é uma tabela 5x5, preenchida por letras, de modo que cada letra pode ser representado por dois números de 1 a 5:

	1	2	3	4	5
1	A	B	C/K	D	E
2	F	G	H	I	J
3	L	M	N	O	P
4	Q	R	S	T	U
5	V	W	X	Y	Z

Como existem 26 letra, mas apenas 25 posições no quadrado, um par de letras terá a mesma posição. No meu caso, como C e K têm sons parecidos, e K não é muito utilizado em português, ambas as letras são representadas pela mesma posição.

Logo para codificar uma letra, basta convertê-la para um par de números. Em seguida, para transmitir esses números, basta fazer barulhos de batida. Por exemplo, para codificar a letra M você a converte para 3, 2 (linha 3, coluna 2), e a transmite batendo 3 vezes, esperando um pouco e batendo mais 2 vezes.

Para decodificar basta contar o número de batidas em cada sequência, e localizá-la na tabela.

Para codificar uma frase inteira, se codifica cada letra separadamente, e transmite todos os números. Para decodificar, basta separar os números em pares, e decodificar cada letra.

Convertendo para código

O funcionamento do programa é o seguinte: o programa, em loop, detecta se houve alguma tecla foi pressionada, e caso haja, o armazena. Quando um par de teclas for armazenado, ele o traduz, usando o quadrado de polybius, e o exibe no lcd.

O código é basicamente:

```
void main() {
    const static unsigned char polibiouSquare[5][5] = {
        {'A', 'B', 'C', 'D', 'E'},
        {'F', 'G', 'H', 'I', 'J'},
        {'L', 'M', 'N', 'O', 'P'},
        {'Q', 'R', 'S', 'T', 'U'},
        {'V', 'W', 'X', 'Y', 'Z'},
    };
    unsigned char input[2];
    unsigned char input_num = 0;
    bool holding = false;
    TRISD = 0x0F; // Use 4 bits do PORTD como entrada (para o teclado)
    for(;;) {
        while (input_num < 2) {
            unsigned char a = tc_tecla();
            if (a > 0 && a <= 5 && !holding) {
                input[input_num] = a;
                input_num++;
            }
            holding = a != 255;
        }
        input_num = 0;
        //adiciona a respectiva letra do quadrado de polybius  no lcd
        lcd_dat(polibiouSquare[input[0]-1][input[1]-1]);
    }
}
```

No caso o quadrado de polybius é uma matriz 5x5, a função tc_tecla() retorna a tecla pressionada ou 255 caso não haja, e lcd_dat(char), adiciona um caractere ao lcd.

Usando o Display Sete Segmentos

Com apenas isso o programa já faz a sua função principal. Mas agora para utilizar o display 7 segmentos para exibir a tecla pressionada, teremos que entrelaçar a execução do código responsável por controlar o display 7 segmentos, e ler a tecla.

Para deixar tudo mais organizado, vou fazer uma função para ler o teclado e uma para exibir o display, e vou transformar as variáveis input e input_num em globais, e variável holding em uma estática dentro da função de ler o teclado.

Agora o loop fica:

```
    for(;;) {
        while (input_num < 2) {
            display_input();
            ler_input();
        }
        input_num = 0;
        //adiciona a respectiva letra do quadrado de polybius  no lcd
        lcd_dat(polibiouSquare[input[0]-1][input[1]-1]);
    }
```

E a função display_input():

```
void display_input() {
    // combinação de segmentos para cada número. 0 é vazio.
    unsigned int values7seg[] = {
        0x00, 0x06, 0x5b, 0x4f, 0x66, 0x6d,
        0x7d, 0x07, 0x7f, 0x67, 0x77, 0x7C,
        0x39, 0x5E, 0x79, 0x71,
    };
    TRISA = 0x00; // define PORTA como saida
    TRISD = 0x00; // define PORTD como saida
    int i, j, d;
    for(i = 0; i<25;i++) {
        PORTA = 0;
        PORTD = values7seg[input[0]];
        PORTA = 1 << 4; // liga o dis3
        for(j = 0; j < 40; j++); // mantem ligado por um tempinho
        PORTA = 0;
        PORTD = values7seg[input[1]];
        PORTA = 1 << 5; // liga o dis4
        for(j = 0; j < 40; j++); // mantem ligado por um tempinho
    }
    PORTA = 0;
}
```

Mas como eu tenho 4 display, eu também posso exibir o par anterior de números. Portanto, eu renomeei a variável input para input_atual, e crio uma variável igual com nome input_ant. Agora eu exibo no display_input o par anterior nos diplays 1 e 2, e assim que eu ler o primeiro número do par atual, e movo os inputs em input_atual para input_ant:

```
void ler_input() {
    static bool holding = false;
    TRISD = 0x0F;
    unsigned char a = tc_tecla();
    TRISD = 0x00;
    if (a > 0 && a <= 5 && !holding) {
        if (input_num == 0) {
            input_ant[0] = input_atual[0];
            input_ant[1] = input_atual[1];
            input_atual[1] = 0;
        }
        input_atual[input_num] = a;
        input_num++;
    }
    holding = a != 255;
}
```

Usando o Buzzer

Agora, para fazer um bip ao pressionar uma tecla, podemos usar o buzzer. Na placa PIC genius, para ligar o buzzer basta desligar o 2º bit do PORTC. Para que o som do buzzer ligue por apenas um instante, vou criar uma variável global chamada buzzer delay, e vou decrementar em 1 a cada loop do programa, e quando chegar a 0, ele desliga o buzzer. Logo só preciso a função:

```
void start_buzzer() {
    PORTC &= ~0b10;
    buzzer_delay = BUZZER_TIME;
}
```

chamar essa função no ler_input, e modificar o loop principal:

```
    for (;;) {
        if (input_num < 2) {
            display_input();
            ler_input();
        } else {
            input_num = 0;
            add_char(polibiouSquare[input_atual[0]-1][input_atual[1]-1]);
        }
        if (buzzer_delay > 0) {
            buzzer_delay--;
            if (buzzer_delay == 0) {
                PORTC |= 0b10;
            }
        }
    }
```

Usando o LCD

Para finalizar eu posso melhorar como o caractere é exibido no lcd ao permitir que o cursor pule para a segunda linha, e implementando um scroll. No caso eu crio uma nova função que mantém em memória os conteúdos da segunda linha, e a posição do cursor. Quando o cursor chegar ao final da primeira linha, o cursor pula para a segunda, e caso ele chegue no final da segunda linha, a segunda linha é copiada para a primeira, e cursor volta para o início da linha:

```
void add_char(unsigned char c) {
    // uma copia do que está escrito na segunda linha
    static char linha_buffer[17] = "                ";
    lcd_dat(c);
    cursor += 1;
    if (cursor == 16) { // a primeira linha encheu
        lcd_set_cursor(1, 0);
    } else if (cursor > 16) { // escrevendo na segunda linha
        linha_buffer[cursor - 17] = c;
        if (cursor == 32) { // segunda linha encheu
            lcd_cmd(L_CLR); //limpa
            lcd_str(linha_buffer);
            lcd_set_cursor(1, 0);
            cursor = 16;
        }
    }
}
```

Outras melhorias

Além do que já foi feito, eu também adicionei umas mensagens de instrução ao iniciar o código, um botão para apagar o que já tenha sido escrito no LCD.

Caso queira ver como isso foi implementado, ou ver como tudo foi feito em mais detalhes, você pode ter acesso ao código pelo meu repositório no github:

https://github.com/Rodrigodd/CodigoDaBatida