UNIVERSIDADE FEDERAL DE ITAJUBÁ
Instituto de Engenharia e Sistemas Tecnologia - IESTI
ECOP4 - Progamação Embarcada

Lucas José Nunes Martins
2020004584

Projeto Final de Programação Embarcada
Estacaum

Ouro Preto - Minas Gerais
01 de Agosto de 2021

Conteúdo

Resumo

*Neste relatório será apresentado o desenvolvimento de um programa para controle de um sistema com enfoque em alimentação e prática de atividades para cachorros, nomeado de **"Estacaum"**. Este sistema foi desenvolvido na disciplina ECOP04, com auxílio dos professores Otávio De Souza Martins Gomes e Rodrigo Maximiano Antunes de Almeida.*

1 Introdução

O objetivo deste projeto foi realizar o desenvolvimento de um programa para simular o controle de uma estação de almentação e de exercícios para cachorros. Para tal feito, foram utilizados alguns recursos presentes na placa PICGenios do PICSimLab, sendo esses os seguintes:

- **LEDs:** Os leds foram utilizados para indicar os níveis de ração e água nos potes de comida dos cachorros, sendo que cada pote irá possuir dois LEDs, um indicando que o pote não está mais vazio, e o outro que o pote está lotado.
- **Relés:** Foram utilizados dois relés, que irão simular a abertura de válvulas, essas válvulas irão liberar a passagem da ração ou da água, vindos do tanque de armazenamento para os potes em que os cacchorross se alimentarão.
- **Display LCD:** O display LCD será responsável por realizar a comunicação entre a Estacaum e o usuário do produto, informando os possíveis comandos e o que está acontecendo na Estacaum. Ele possui 4 possíveis menus, cada um com uma determinada atividade implementada.
- **Teclado:** O teclado também fará parte da nossa HMI(Interface de comunicação homem máquina), sendo utilizado para alternar entre as funções da Estacaum e para entrar com comandos na mesma.
- **Cooler:** O cooler será responsável por simular um motor, o qual iria girar uma esteira para que o cachorro possa correr sobre ela e brincar/exercitar. Essa esteira poderá ter sua velocidade controlada.
- **Potênciometro:** O potênciometro irá realizar o controle da velocidade da esteira, podendo variar essa velocidade entre 0% e 100%.

Com esses recursos desenvolvemos as funcionalidades para controlar a Estacaum, que pode ser visualizada segundo o seguinte protótipo:

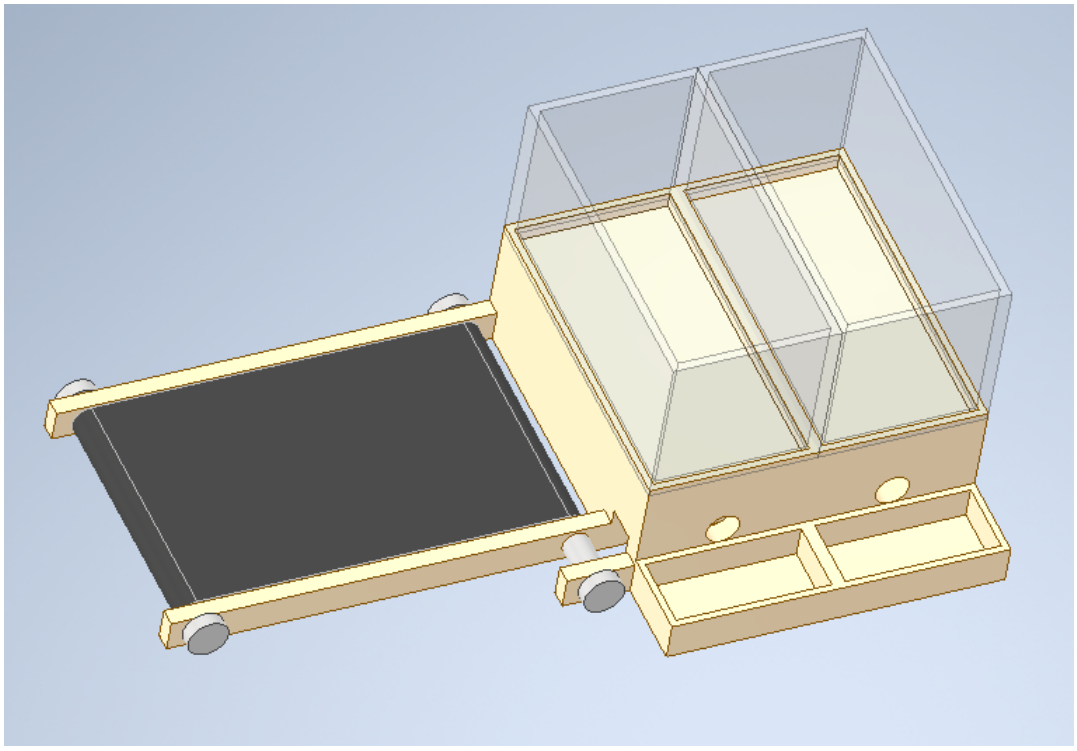


Figura 1: Protótipo da Estacaum.

Os códigos para implementação desses recursos serão descritos nos próximos tópicos desse relatório.

2 Funcionalidades e Códigos Implementados - Menus

Agora, iremos descrever todas as funcionalidades que foram implementadas no projeto Estacaum, e os códigos para que as mesmas funcionassem.

Sendo assim, primeiro é importante destacar que o programa foi feito como uma máquina de estados, onde teremos uma flag (declarada como uma variável char 'op') que irá decidir qual dos menu deverá ser utilizado em um determinado momento, e para alternar entre esses menus, utilizamos a tecla '1'.

2.1 Data e Hora

O menu inicial da Estacaum permite a visualização da data e hora da mesma pelo display LCD. Conforme mostra a imagem

Figura 2: 1º menu da Estacaum.

Essas funções são importantes, pois a data (mais especificamente o dia da semana e o horário) serão utilizados para que a Estacaum seja programada para servir ração aos cachorros em um horário determinado.

O código para impressão desse 1º menu foi feito nos arquivos showNow.h e showNow.c, onde foi implementada a função **showNow**, que é chamada quando *op = 0*:

```
//Mostra a hora atual se op = 0
if (op == 0) {
    if (cont % 300 == 0) {
        rtc_r();
        showNow(0xC4, (char *) time, 0x
         (char *) date, 0x80, (char) da
    }
}
```

```
void showNow(char posTime, char time[10],
        char posDate, char date[10], char
        posDay, char day){
    lcdCommand(CLR);

    switch(day - 48){
        case 0:
            printDay("dom", posDay);
            break;
        case 1:
            printDay("seg", posDay);
            break;
        case 2:
            printDay("ter", posDay);
            break;
        case 3:
            printDay("qua", posDay);
            break;
        case 4:
            printDay("qui", posDay);
            break;
        case 5:
            printDay("sex", posDay);
            break;
        case 6:
            printDay("sab", posDay);
            break;
    }

    lcdCommand(posDate);
    for(char i=0; i<8; i++){
        lcdData(date[i]);
    }

    lcdCommand(posTime);
    for(char i=0; i<8; i++){
        lcdData(time[i]);
    }
}

void printDay(char dia[4], char pos){
    lcdCommand(pos);
    for(char i=0; i<3; i++){
        lcdData(dia[i]);
    }
}
```

Figura 3: Codigos no picsimlab para implementação do 1º menu.

Assim, quando op = 0, a função showNow é chamada, passando como parâmetros a posição onde queremos que as horas apareçam no LCD, em seguida do vetor de horas lido na função rtc_r(), e da mesma forma para a posição da data, a data, a posição do dia da semana, e o dia da semana.

Então, na função showNow verificamos qual o dia da semana, e de acordo com o dia imprimimos as três primeiras letras que representam esse dia com a utilização de um switch. Depois, imprimimos a Data e a Hora com a utilização de dois for.

OBS: na função showNow é chamada a printDay, que irá receber o dia da semana e a posição onde se deseja escrevê-lo no display LCD, e então, escrever esse dia na posição desejada.

2.2 Inserção de horário para refeição

Nesse menu podemos adicionar as refeições que queremos que a Estacaum sirva ao cachorro. Conforme segue:

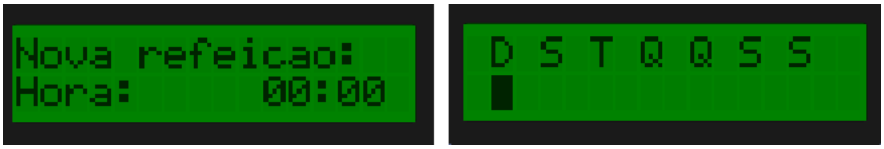


Figura 4: 2º menu da estacaum

Como um dos objetivos do projeto é diminuir os custos, temos que ele só utilizará 4 teclas para controle do mesmo e 1 para ligar/desligar.

Assim, como devemos selecionar hora, minutos e dias para a inserção de uma nova refeição, utilizamos a tecla **'2'** para alternar entre o que está sendo selecionado, alterando uma variável denominada *flagOp* no nosso código, conforme segue:

```
if (bitTst(tecla, 7)) {
    if (op == 1) {
        flagOp++;
        if (flagOp > 2) {
            insertMeal(dia, hora, minuto);

            hora[0] = '0';
            hora[1] = '0';
            minuto[0] = '0';
            minuto[1] = '0';
            dia = 0;
            lcdCommand(OFF);
            flagOp = -1;
            lcdInit();
            newTime(flagOp);
        }
        lcdCommand(CLR);

        newTime(flagOp);
        if (flagOp == 2) {
            newDay(contDay);
        }
    }
```

```
if (bitTst(tecla, 2)) {
    //Ajusta as horas
    if (op == 1 && flagOp == 0) {
        changeHours(hora, 0);
        newTime(flagOp);
    }

    //Ajusta os minutos
    if (op == 1 && flagOp == 1) {
        changeMinutes(minuto, 0);
        newTime(flagOp);
    }

    //Ajusta os dias da semana
    if (op == 1 && flagOp == 2) {
        bitFlp(dia, contDay);
        printDaysSelected(dia);
        lcdCommand(ON);
        lcdCommand((char) (L1 + 1
                + contDay * 2));
    }
```

```
if (bitTst(tecla, 6)) {
    //muda horas
    if (op == 1 && flagOp == 0) {
        changeHours(hora, 1);
        newTime(flagOp);
    }

    //muda minutos
    if (op == 1 && flagOp == 1) {
        changeMinutes(minuto, 1);
        newTime(flagOp);
    }

    //muda dias
    if (op == 1 && flagOp == 2) {
        contDay++;
        if (contDay > 6) {
            contDay = 0;
        }
        newDay(contDay);
    }
```

Figura 5: Utilização da flagOp para selecionar horas, minutos e dias.

Dessa forma, tendo op = 1, o que indica que estamos nesse 2º menu, iremos verificar a variável flagOp:

- Para flagOp = 0 ou flagOp = 1, mudaremos as horas e os minutos respectivamente, com o uso das funções **changeHours** e **changeMinutes** que serão chamadas ao pressionarmos as teclas **4** para diminuir e **5** para aumentar, e são definidas nos nossos arquivos meal.h e meal.c. Além disso, iremos exibir o que está sendo feito na tela com a função newTime da mesma biblioteca.

- Para flagOp = 2, iremos selecionar os dias em que se deseja que a refeição seja servida durante a semana. Para isso, utilizamos a variável dia, e as teclas **4** e **5**, sendo que a tecla 5 servirá para movimentar entre os dias, e a tecla 4 para selecionar ou desselecionar o dia. Sendo assim, ao apretarmos a tecla 4 em um determinado dia, realizaremos um bitFlip no bit correspondente à esse dia da semana na variável dia, e então, caso a variável dia tenha o bit correspondente igual a 1, indica que nesse dia a refeição deve ser servida, caso contrário, indica que não deve. E assim, a função **printDaySelected** imprime os dias selecionados.

- Para flagOp > 2, todos os dados foram coletados, e assim, podemos adicionar o novo horário de refeições a nossa lista de refeições, utilizando a função **insertMeal**, que adicionará o novo elemnto a nossa lista estática. Essa por sua vez pode ter no máximo 16 elementos.

Seguem as funções citadas dos arquivos meal.h e meal.c:

Figura 6: Funções em meal.c.

A função insert estará no próximo subtópico, junto as funções do nosso arquivo list.c.

2.3 Varificação dos horáios para refeição

No 3º menu da Estacaum é possível verificar nossa lista estática que possui as refeições que a nossa Estacaum deve servir aos cachorros.

Esse menu mostra o horário das refeições e os dias que ela deve ser servida, sendo que cada dia está na posição determinada dele. Isto é, começando de domingo até sábado, os dias são exibidos pulando sempre um digito do display LCD entre eles, e assim, começando do digito 1 ele mostra o D (de domingo), no 3 o S (de segunda) até o 13 onde mostra o S (de sábado). Assim temos a seguinte tela:

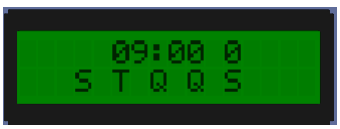
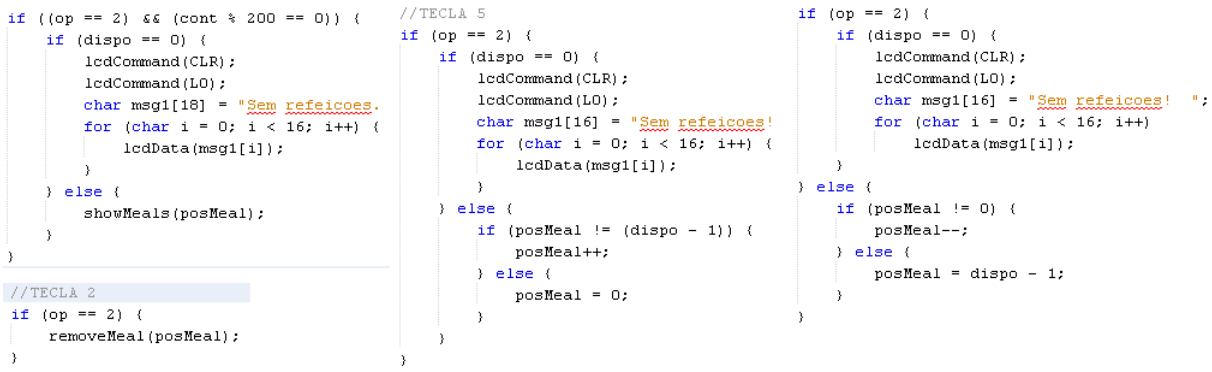


Figura 7: 3º menu da Estacaum, mostrando a refeição de número 0 da lista.

Como é possível ver, essa refeição deve ser servida às 9 horas dos dias segunda, terça, quarta, quinta e sexta. Além disso, o 0 ao lado das horas mostra que ela é a primeira refeição da lista, que possui as refeições odernadas em ordem crescente em relação as horas e para alternarmos entre as refeições basta utilizar as teclas 4 e 5. Caso queiramos remover uma refeição, devemos somente clicar na tecla 2, e ela será removida.

Para implementação dessas funcionalidades, foram utilizados os códigos abaixo:



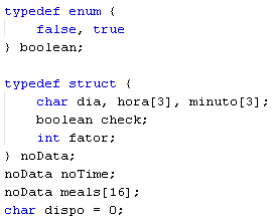
```
if ((op == 2) && (cont % 200 == 0)) {
    if (dispo == 0) {
        lcdCommand(CLR);
        lcdCommand(L0);
        char msg1[18] = "Sem refeicoes.
        for (char i = 0; i < 16; i++) {
            lcdData(msg1[i]);
        }
    } else {
        showMeals(posMeal);
    }
}

//TECLA 2
if (op == 2) {
    removeMeal(posMeal);
}
```

```
//TECLA 5
if (op == 2) {
    if (dispo == 0) {
        lcdCommand(CLR);
        lcdCommand(L0);
        char msg1[16] = "Sem refeicoes!
        for (char i = 0; i < 16; i++) {
            lcdData(msg1[i]);
        }
    } else {
        if (posMeal != (dispo - 1)) {
            posMeal++;
        } else {
            posMeal = 0;
        }
    }
}
```

```
if (op == 2) {
    if (dispo == 0) {
        lcdCommand(CLR);
        lcdCommand(L0);
        char msg1[16] = "Sem refeicoes!  ";
        for (char i = 0; i < 16; i++)
            lcdData(msg1[i]);
    }
} else {
    if (posMeal != 0) {
        posMeal--;
    } else {
        posMeal = dispo - 1;
    }
}
```

Figura 8: Trechos da função main para manipulação do 3º menu.

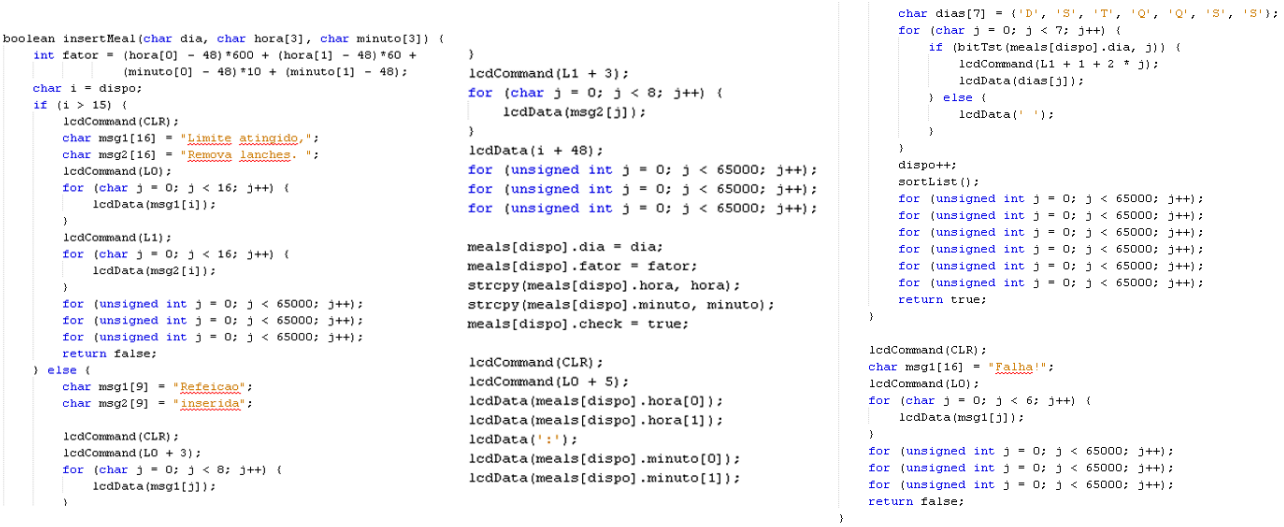
Ademais, foram utilizadas as funções presentes nos nossos arquivos list.h e list.c, além da estrutura noData, para que fizessemos a inserção, remoção e organização da lista com nossas refeições, sendo essas implementadas conforme é mostradao abaixo:



```
typedef enum {
    false, true
} boolean;

typedef struct {
    char dia, hora[3], minuto[3];
    boolean check;
    int fator;
} noData;
noData noTime;
noData meals[16];
char dispo = 0;
```

Figura 9: Variáveis utilizadas para manipulação da lista.

Dessas variáveis, criamos o tipo boolean para utilização na função **searchList** que busca um elemento na lista. O tipo noData para armazenar nossas refeições, a variável noTime para transformarmos as horas vindas das variáveis da biblioteca rtc.h em uma variável do tipo noData, através da função **turnIntoTime**, e assim podermos compará-la com nossos horáios de refeição. O vetor meals que aloca espaço para as 16 possíveis refeições. E a variável dispo, que indica a primeira posição vazia do vetor meals.



```
boolean insertMeal(char dia, char hora[3], char minuto[3]) {
    int fator = (hora[0] - 48)*600 + (hora[1] - 48)*60 +
                (minuto[0] - 48)*10 + (minuto[1] - 48);
    char i = dispo;
    if (i > 15) {
        lcdCommand(CLR);
        char msg1[16] = "Limite atingido,";
        char msg2[16] = "Remova lanches. ";
        lcdCommand(L0);
        for (char j = 0; j < 16; j++) {
            lcdData(msg1[i]);
        }
        lcdCommand(L1);
        for (char j = 0; j < 16; j++) {
            lcdData(msg2[i]);
        }
        for (unsigned int j = 0; j < 65000; j++);
        for (unsigned int j = 0; j < 65000; j++);
        for (unsigned int j = 0; j < 65000; j++);
        return false;
    } else {
        char msg1[9] = "Refeicao";
        char msg2[9] = "inserida";

        lcdCommand(CLR);
        lcdCommand(L0 + 3);
        for (char j = 0; j < 8; j++) {
            lcdData(msg1[j]);
        }
        lcdCommand(L1 + 3);
        for (char j = 0; j < 8; j++) {
            lcdData(msg2[j]);
        }
        lcdData(i + 48);
        for (unsigned int j = 0; j < 65000; j++);
        for (unsigned int j = 0; j < 65000; j++);
        for (unsigned int j = 0; j < 65000; j++);

        meals[dispo].dia = dia;
        meals[dispo].fator = fator;
        strcpy(meals[dispo].hora, hora);
        strcpy(meals[dispo].minuto, minuto);
        meals[dispo].check = true;

        lcdCommand(CLR);
        lcdCommand(L0 + 5);
        lcdData(meals[dispo].hora[0]);
        lcdData(meals[dispo].hora[1]);
        lcdData(':');
        lcdData(meals[dispo].minuto[0]);
        lcdData(meals[dispo].minuto[1]);
        char dias[7] = {'D', 'S', 'T', 'Q', 'Q', 'S', 'S'};
        for (char j = 0; j < 7; j++) {
            if (bitTst(meals[dispo].dia, j)) {
                lcdCommand(L1 + 1 + 2 * j);
                lcdData(dias[j]);
            } else {
                lcdData(' ');
            }
        }
        dispo++;
        sortList();
        for (unsigned int j = 0; j < 65000; j++);
        for (unsigned int j = 0; j < 65000; j++);
        for (unsigned int j = 0; j < 65000; j++);
        for (unsigned int j = 0; j < 65000; j++);
        for (unsigned int j = 0; j < 65000; j++);
        for (unsigned int j = 0; j < 65000; j++);
        return true;
    }

    lcdCommand(CLR);
    char msg1[16] = "Falha!";
    lcdCommand(L0);
    for (char j = 0; j < 6; j++) {
        lcdData(msg1[j]);
    }
    for (unsigned int j = 0; j < 65000; j++);
    for (unsigned int j = 0; j < 65000; j++);
    for (unsigned int j = 0; j < 65000; j++);
    return false;
}
```

Figura 10: Função para inserção de refeição na lista.

A função **insertMeal** insere refeições na lista as ordenando de forma crescente em relação as horas pela chamada da função **sortList**, que realiza um bubble sort na lista.

```
void removeMeal(char pos) {
    for(char i = pos; i < dispo; i++){
        meals[i] = meals[i + 1];
    }
    dispo--;
    lcdCommand(CLR);
    char msg1[12] = "Refeicao   ";
    char msg2[12] = "removida.  ";

    lcdCommand(L0);
    for(char i=0; i < 11; i++){
        lcdData(msg1[i]);
    }

    lcdCommand(L1);
    for(char i=0; i < 11; i++){
        lcdData(msg2[i]);
    }

    for (unsigned int j = 0; j < 65000; j++);
    for (unsigned int j = 0; j < 65000; j++);
    for (unsigned int j = 0; j < 65000; j++);
}

void turnIntoTime(char hora[10], char dia) {
    noTime.dia = 0;
    noTime.dia |= (1 << (dia - 48));

    for (char j = 0; j < 2; j++) {
        noTime.hora[j] = hora[j];
        noTime.minuto[j] = hora[j + 3];
    }
    noTime.fator = (noTime.hora[0] - 48)*600 +
    (noTime.hora[1] - 48)*60 + (noTime.minuto[0]
    - 48)*10 + (noTime.minuto[1] - 48);
}
```

```
void showMeals(unsigned char pos) {
    char dias[7] = {'D', 'S', 'T', 'Q', 'Q', 'S', 'S'};

    if (pos < dispo) {
        lcdCommand(CLR);
        lcdCommand(L0 + 5);
        lcdData(meals[pos].hora[0]);
        lcdData(meals[pos].hora[1]);
        lcdData(':');
        lcdData(meals[pos].minuto[0]);
        lcdData(meals[pos].minuto[1]);
        lcdData(' ');
        lcdData(pos+48);

        for (char i = 0; i < 7; i++) {
            if (bitTst(meals[pos].dia, i)) {
                lcdCommand(L1 + 1 + 2 * i);
                lcdData(dias[i]);
            } else {
                lcdData(' ');
            }
        }
    }else{
        lcdCommand(CLR);
        lcdData(pos+48);
        lcdData('>');
        lcdData(dispo+48);
    }
}

void listInit() {
    for (char j = 0; j < 16; j++) {
        meals[j].fator = -1000;
    }
}
```

```
void sortList(){
    noData aux;
    for(char i=0; i<dispo; i++){
        for(char j=0; j < dispo-1-i; j++){
            if(meals[j].fator > meals[j+1].fator){
                aux = meals[j];
                meals[j] = meals[j+1];
                meals[j+1] = aux;
            }
        }
    }
}

boolean searchList(noData time){
    char lim = dispo, i = 0, aux;
    while(i < lim){
        aux = (lim + i) / 2;
        if(meals[aux].fator > time.fator){
            i = aux + 1;
        }else if(meals[aux].fator < time.fator){
            lim = aux - 1;
        }else if((meals[aux].fator == time.fator) &&
                (meals[aux].dia & time.dia)){
            return true;
        }else{
            return false;
        }
    }
    return false;
}
```

Figura 11: Outras funções da biblioteca list.

Das funções restantes, a **removeMeal** remove uma refeição da lista. A **showMeals** mostra a refeição referente ao indice que lhe é passado como parâmetro. E a **listInit** inicializa a lista colocando um valor absurdo no fator de todas as posições, variável que é responsável por transformar as horas em um valor comparável, e será sempre positivo se utilizarmos horas reais.

2.4 Controle de velocidade da esteira

Para controle de velocidade da esteira, temos o 4º menu da Estacaum, que será onde poderemos aumentar e diminuir a velocidade da mesma pelo potênciometro. Esse menu tem a seguinte aparência:

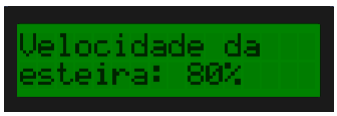


Figura 12: 4º menu da Estacaum.

A velocidade pode variar entre 0% e 99%, de acordo com a rotação do potênciometro.
Para implementação desse menu foram utilizados os códigos abaixo:

```
if (op == 3) {
    if (cont % 500 == 0) {
        itoa(((adc_amostra(0) * 10)
                / 204), str);
        pwm = (str[0] - 48)*20 +
                (str[1] - 48)*2;
        pwmSet(pwm);
        showPWM(pwm);
    }
}
```

```
void showPWM(unsigned char pwm) {
    char msg1[16] = "Velocidade da    ";
    lcdCommand(CLR);

    lcdCommand(L0);
    for (char i = 0; i < 16; i++) {
        lcdData(msg1[i]);
    }

    char msg2[16] = "esteira: ";
    lcdCommand(L1);
    for (char i = 0; i < 16; i++) {
        lcdData(msg2[i]);
    }

    lcdCommand(L1 + 9);
    if(pwm > 99){
        pwm = 99;
    }
    lcdData((char)(pwm/10 + 48));
    lcdData((char)(pwm%10 + 48));
    lcdData('%');
    for(char i=0; i<5; i++){
        lcdData(' ');
    }
}
```

Figura 13: Código para implementação do menu 4.

As funções utilizadas são da biblioteca pwm.h e pwm.c, onde a única função implementada foi a **showPWM**, que mostra a porcentagem de velocidade com que a esteira está funcionando.

Vale destacar que a esteira continua funcionando com a velocidade configurada mesmo nos outros menus, mas sua velocidade só pode ser alterada nesse menu. E assim, encerramos nossos menus.

3 Outras funcionalidades implementadas

Algumas outras funcionalidades foram implementadas na nossa Estacaum, sendo elas as seguintes.

3.1 Indicadores de nível

Os indicadores de nível de ração e água nos potes para alimentação dos cachorros são LEDs, sendo que possuimos dois LEDs para cada um dos potes, um indicando que o pote não está vazio, e o outro indicando que ele está cheio. Os leds utilizados foram os seguintes:

Figura 14: LEDs de indicação de nível

Os LEDs de incicação de nível são divididos da seguinte forma: LED B4 - vasilha de ração com comida; LED B5 - vasilha de ração cheia; LED B6 - vasilha de água com água; LED B7 - vasilha de água cheia.

E os códigos para simulação desses sensores de nível no PICSimlab foram os seguintes:

```
if (cont % 2000 == 0) {
    rtc_r();
    turnIntoTime(time, day);

    if (searchList(noTime)) {
        PORTCbits.RC0 = 1;
        if (!bitTst(PORTB, 5)) {
            if (meal == 0) {
                mealFigure();
            }
            meal++;
        }
    }
    if (meal > 1 && meal < 4) {
        bitSet(PORTB, 4);
    } else if (meal > 3) {
        bitSet(PORTB, 5);
        PORTCbits.RC0 = 0;
    }
}
```

```
if (cont % 1000 == 0) {
    if (water < 3) {
        PORTEbits.RE0 = 1;
    }
    if(bitTst(PORTE, 0)) water++;

    if (water > 3 && water < 8) {
        bitSet(PORTB, 6);
    } else if (water > 8) {
        bitSet(PORTB, 7);
        PORTEbits.RE0 = 0;
    }
}
// Simula que o pote de racao ficou vazio ///
if (bitTst(tecla, 1)) {
    bitClr(PORTB, 4);
    bitClr(PORTB, 5);
    meal = 0;
}
if (bitTst(tecla, 5)) {
    bitClr(PORTB, 6);
    bitClr(PORTB, 7);
```

Figura 15: Código dos LEDs.

Nesses códigos temos que a água do cachorro será reposta sempre que sua vasilha se esvaziar, e que a ração só será colocada nos horários que estão presentes na lista de horários, no nosso vetor meals. Além disso, temos a utilização das teclas 7 e 8, para simular que as vazilhas foram esvaziadas, o que não seria necessário em uma implementação real.

3.2 Relés

Nos códigos do subtópico anterior podemos ver também a utilização dos relés nos bits RC0 e RCE. Que irão indicar o acionamento das válvulas para que a água ou a ração seja liberada.

Esse acionamento ocorrerá da forma explicada anteriormente. Ou seja, a água será reposta sempre que seu nível estiver baixo e a ração nos horários selecionados. E ambas serão desligadas quando os potes respectivos a cada uma das válvulas estiverem vazios.

3.3 Desnhos

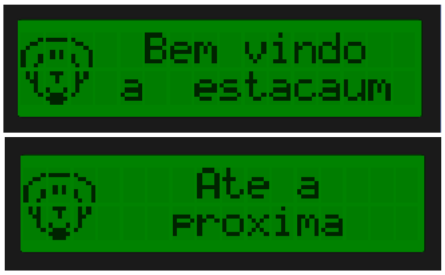
Dois desenhos foram implementados, sendo eles:

- O de um pote de ração, que aparecerá nos momentos em que a Estacaum servir ração para o cachorro, como abaixo:


```
char food[48] = {
    0b00000, 0b00000, 0b00000, 0b00000, 0b00001, 0b00010, 0b00100, 0b01000,
    0b00000, 0b00000, 0b00010, 0b10101, 0b01000, 0b00101, 0b10000, 0b01001,
    0b00000, 0b00000, 0b00000, 0b00000, 0b10000, 0b01000, 0b00100, 0b00010,
    0b01111, 0b01110, 0b01100, 0b01100, 0b01110, 0b01111, 0b01111, 0b00000,
    0b11111, 0b01110, 0b00000, 0b00000, 0b01110, 0b11111, 0b11111, 0b00000,
    0b11110, 0b01110, 0b00110, 0b00110, 0b01110, 0b11110, 0b11110, 0b00000
};
```

Figura 16: Figura mostrada no momento de servir o chachorro e vetor com linhas que formam a figura.

- O de um cachorro, que aparecerá nos momentos em que a Estacaum iniciar e desligar, da seguinte forma:




```
char dog[48] = {
    0b00000, 0b00001, 0b00110, 0b01000, 0b10000, 0b10001, 0b10001, 0b10010,
    0b00000, 0b11111, 0b00000, 0b00000, 0b01010, 0b01010, 0b00000, 0b00000,
    0b00000, 0b10000, 0b01100, 0b00010, 0b00001, 0b10001, 0b10001, 0b01001,
    0b01010, 0b01010, 0b00110, 0b00010, 0b00010, 0b00001, 0b00000, 0b00000,
    0b01110, 0b00100, 0b00100, 0b00000, 0b10001, 0b01110, 0b11111, 0b01110,
    0b01010, 0b01010, 0b01100, 0b01000, 0b01000, 0b10000, 0b00000, 0b00000
};
```

Figura 17: Telas de incício e fim do programa, e vetor com as linhas que formam a imagem.

4 Resultados

Os resultados obtidos com esse projeto foram apresentados em um video, que pode ser acessado no seguinte link.

Videos explicativo: https://youtu.be/SdiLew646v4

E todos os arquivos desse projeto explicado no relatório e no video, podem ser acessados através do link abaixo.

Arquivos do projeto: https://github.com/lucasjnmartins/Projeto$_Final_ECOP04.git$

5 Conclusão

Através desse projeto final da disciplina ECOP04 foi possível ter noções práticas do desenvolvimento de softwares para embarcados, enfrentando as dificuldades que esses tipos de sistema podem trazer para sua implementação.

Foi possível trabalhar a criatividade e realizar a construção de um programa que pode ser utilizado para uma aplicação real, com auxílio da IDE **MPLabX** e do simulador **PICSimLab**.

Sendo assim, o projeto possibilitou a aquisição de novos conhecimentos e prática dos já adquiridos, tanto em questões técnicas quanto incentivando a resiliência para solução dos problemas.