Sistemas Microprocessados

Projeto Final de Sistemas Microprocessados

Caio Henrique de Oliveira Coelho - 11086815
Felipe Azzolino Varella - 11201722225
Gabriel Zolla Juarez - 11201721446
Guilherme Eduardo Pereira - 11201720498

25 de abril de 2023

1 Introdução

Uma interface humano-máquina, também conhecida como IHM, é um sistema que permite a comunicação entre um indivíduo e um computador de forma interativa, seja por interface gráfica, por voz, gestos, e até mesmo realidade virtual. O principal foco nesse âmbito de pesquisa é visar ao máximo a facilidade na interação, usabilidade e eficiência do sistema, e podendo também ser adaptada à diferentes tipos de usuário, como indivíduos com deficiências visuais, que necessitam de funcionalidades distintas para que a navegação e uso seja feito da melhor forma possível.

Em sistemas microprocessados, a presença da IHM é corriqueira. Ela existe para construir sistemas com finalidade de controlar dispositivos com botões, displays, teclados, interfaces de voz, sensores, entre outros. Nesse contexto, os microcontroladores são programados para receber entradas do usuário por meio das interações do mesmo, e decidem, a partir da lógica implementada, o que deve fazer com tal informação e como deve dispor o resultado ao usuário.

No projeto em questão, o Arduino, frequentemente utilizado em projetos eletrônicos, realiza o papel do microcontrolador, e pode, por conseguinte, se aliar a uma estrutura interativa para transformar o sistema de forma a transformá-lo em uma IHM. Essa tarefa será atribuída principalmente ao Keypad Shield LCD, cujos botões pressionados pelo usuário enviarão comandos ao Arduino, para que este tome o comportamento adequado e ambos em conjunto formem um sistema semelhante ao que se tem nos micro-ondas atuais. O usuário pode, por exemplo, querer alterar o tempo de funcionamento do dispositivo, então ele interage com os botões para fazê-lo. O sistema deve reconhecer esse tempo, disponibilizá-lo na tela para que o indivíduo se certifique do que está fazendo, e armazená-lo para quando o sistema iniciar, o tempo ser decrementado e após percorrer esses instantes, tomar uma nova decisão correspondente à finalização do funcionamento.

Vale ressaltar que é uma das práticas fundamentais da área de interface humano-máquina garantir que o usuário possa interagir com o sistema de forma fácil e intuitiva, o que foi visado ao máximo no projeto de simulação do micro-ondas.

2 Objetivos

2.1 Proposta

A proposta do projeto é desenvolver uma Interface Homem-Máquina (IHM) que simule o funcionamento de um micro-ondas. Para tal utilizar-se-á o Arduino aliado a um Shield com display LCD, alguns dispositivos como LED, buzzer, bem como aproveitando os conceitos da

disciplina de Sistemas Microprocessados. Através desse sistema, poder-se-á definir o tempo de funcionamento do micro-ondas, que será definido pelo próprio usuário a partir do display LCD do Shield e dos botões que o acompanham, e os demais componentes auxiliares servem para informar o usuário sobre o estado de funcionamento do mesmo. Por exemplo, o LED será utilizado para indicar quando o micro-ondas está ligado ou desligado, enquanto o buzzer para emitir sons de alerta quando o tempo de aquecimento chegar ao fim.

2.2 Lista de materiais

Qtd.	Materiais
1	Display Lcd Keypad Shield 16x2 com teclado
1	Arduino UNO
1	Led RGB
1	Buzzer
1	Resistor 550Ω
1	Protoboard
1	Motor CC 3-6V
8	Jumpers macho-fêmea

Tabela 1: Lista de materiais

Os materiais utilizados podem ser observados abaixo, na Imagem 1:

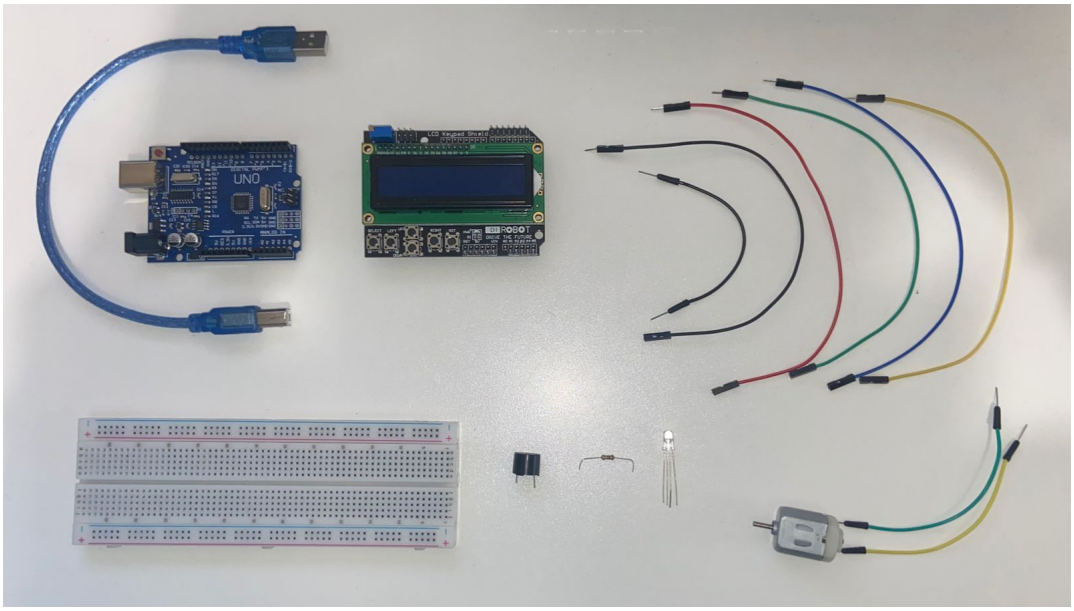

Figura 1: Materiais utilizados

As especificações do motor, por sua vez, são as seguintes:

- Tensão de funcionamento: 3 a 6V;
- Corrente aproximada sem carga em 3 V: 150mA;
- Corrente aproximada sem carga em 6 V: 200mA;
- Velocidade aproximada em 3V: 3000RPM;
- Velocidade aproximada em 6V: 6000RPM;
- Diâmetro do eixo : 2.0mm;
- Saída do comprimento do eixo: 75mm;
- Dimensões: 38mm x 20mm x 15mm;
- Peso: 16g.

3 Desenvolvimento

3.1 Fundamentação Teórica

Para o desenvolvimento deste projeto, os conhecimentos adquiridos durante o curso de Sistemas Microprocessados foram de suma importância. Como alguns conceitos acabaram estando mais presentes, abordaremos estes em mais detalhes.

3.1.1 Timers

Os timers são circuitos eletrônicos integrados a microcontroladores que contam com uma série de contadores que funcionam como temporizadores, nos permitindo executar tarefas com base em intervalos de tempos. O microprocessador ATmega328 (presente na placa Arduino UNO) tem três timers diferentes:

- TIMER 0: De 8 bits, utilizado em algumas funções integradas como delay e millis.
- TIMER 1: De 16 bits, utilizado por padrão pela biblioteca Servo para o controle de servomotores.
- TIMER 2: De 8 bits, sendo utilizado por padrão na biblioteca Tone.

Apesar de não ter sido necessária a configuração manual dos Timers, como a função delay (utilizada em nosso programa) utiliza o Timer 0 para gerar interrupções em intervalos bem definidos, o conhecimento dos conceitos que rondam timers foram de suma importância.

3.1.2 Leituras analógicas e Conversor Analógico-Digital (DAC)

Apesar de ser um dispositivo digital, por meio de um conversor analógico-digital de 10 bits integrado ao microcontrolador, o Arduino UNO nos permite realizar leituras analógicas. Assim, os valores lidos podem variar de 0 a 1023, onde 0 corresponde a uma entrada de 0V e 1023 a uma entrada de 5V, sendo 5.5V o limiar máximo suportado pela placa antes que a sua porta de leitura analógica seja prejudicada. Em suma, podemos inferir a tensão lida na porta por meio da equação:

$$V(N) = \frac{5}{1023} \cdot N \tag{1}$$

onde $N = 0, 1, \ldots, 1023$ representa o valor lido.

O Shield utilizado neste projeto (que acopla um display LCD e um conjunto de botões) faz uso deste conceito para mapear qual botão foi pressionado. O esquemático do sistema utilizado pelo Shield encontra-se na Figura 2.

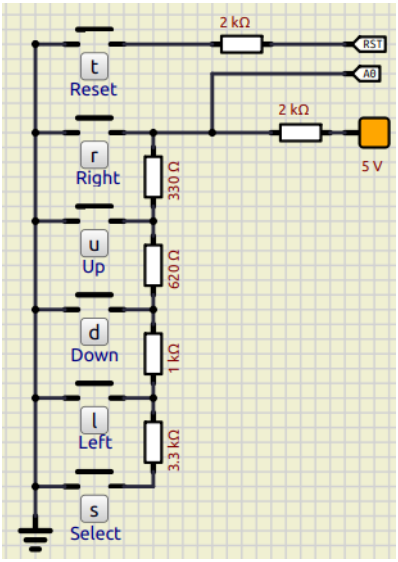

Figura 2: Esquemático da leitura analógica realizada pelo Shield

Considerando que apenas um botão será pressionado por vez, podemos deduzir a tensão lida em A0 por meio do conceito de divisor de tensão conforme, representado na equação 2:

$$V_{A0}(R_{eq}) = \frac{R_{eq}}{R_{eq} + 2 \cdot 10^3} \cdot 5 \tag{2}$$

onde R_{eq} corresponde a resistência equivalente entre $A0$ e o terra. Substituindo 2 em 1:

$$\begin{aligned} \frac{R_{eq}}{R_{eq} + 2 \cdot 10^3} \cdot 5 &= \frac{5}{1023} \cdot N \\ \therefore N(R_{eq}) &= \frac{R_{eq}}{R_{eq} + 2 \cdot 10^3} \cdot 1023 \end{aligned} \tag{3}$$

que nos permite deduzir o seguinte mapeamento entre botões pressionados para valor lido (N):

$$\begin{cases} \text{Se nenhum botão for pressionado} \Rightarrow R_{eq} \to \infty\ \Omega \therefore N = 1023 \\ \text{Se "s"for pressionado} \Rightarrow R_{eq} = 3.3k + 1k + 620 + 330\ \Omega \therefore N \approx 741 \\ \text{Se "l"for pressionado} \Rightarrow R_{eq} = 1k + 620 + 330\ \Omega \therefore N \approx 505 \\ \text{Se "d"for pressionado} \Rightarrow R_{eq} = 620 + 330\ \Omega \therefore N \approx 330 \\ \text{Se "u"for pressionado} \Rightarrow R_{eq} = 330\ \Omega \therefore N \approx 145 \\ \text{Se "r"for pressionado} \Rightarrow R_{eq} = 0\ \Omega \therefore N = 0 \end{cases} \tag{4}$$

Como os valores podem variar dentro de uma faixa especificada, o grupo optou por analisar intervalos de valores ao invés de fazer uma comparação exata, culminando na seguinte relação representada em 5:

$$\begin{cases} \text{Se } N \in [0, 80] \Rightarrow \text{"r"foi pressionado} \\ \text{Se } N \in [81, 200] \Rightarrow \text{"u"foi pressionado} \\ \text{Se } N \in [201, 400] \Rightarrow \text{"d"foi pressionado} \\ \text{Se } N \in [401, 600] \Rightarrow \text{"l"foi pressionado} \\ \text{Se } N \in [601, 800] \Rightarrow \text{"s"foi pressionado} \\ \text{Caso contrário, nenhum botão foi pressionado} \end{cases} \tag{5}$$

que nos permite determinar qual foi o botão pressionado com base no valor lido na porta analógica A0.

3.1.3 Tipos de memórias

Quando estamos trabalhando na parte de desenvolvimento voltada para microcontroladores, um dos principais pontos de atenção deve ser o uso de memória, visto que, diferente dos notebooks e desktops, estes dispositivos tendem a ter uma quantidade muito mais reduzida de memória volátil (e.g. SRAM) e não-volátil (e.g. Flash). No caso do Arduino UNO,

temos controle sobre o uso de dois tipos de memória:

- Flash: Uma memória não-volátil de 32kB responsável por armazenar constantes e o programa em si.
- SRAM: Uma memória volátil de 2kB responsável por armazenar as variáveis utilizadas durante a execução do programa.

Como a escolha dos tipos de dados afeta diretamente no comportamento do sistema, podendo até levá-lo a travamento ou reinicialização em situações onde é necessário alocar mais memória do que se tem disponível, durante o desenvolvimento do projeto buscou-se designar cautelosamente a cada dado o tipo correto bem como a sua natureza enquanto constante ou variável.

3.2 Desenvolvimento do Software

Primeiramente, é interessante destacar a arquitetura do sistema como um todo. A fim de visar uma melhor organização do código, principalmente para permitir um trabalho concomitante dos membros do grupo no processo e melhor compreensão do código escrito, optou-se por uma estrutura baseada em múltiplos arquivos, em que cada um destes representa um componente do sistema, como, por exemplo, o LED, o Shield, entre outros.

Para tal, utilizou-se o conceito de headers da linguagem C++, que basicamente são interfaces que definem as funções e atributos (públicos e privados) de uma classe, e permite que esta seja referenciada em outros arquivos do código, tal como é possível fazer com bibliotecas do Arduino por meio do include. A estrutura do código dessas interfaces têm o seguinte formato, tomando como base a classe do Buzzer:

```
#ifndef BUZZER_H
#define BUZZER_H

#include <Arduino.h>

class Buzzer {
    public:
        Buzzer(byte buzzerPin);
        // Metodos
        void buzz(int smallerBuzzPeriod, int biggerBuzzPeriod);

    private:
        // Atributos
```

```
        byte buzzerPin;
    };

#endif
```

Código 1: Exemplo de um Header correspondente à interface da classe Buzzer

Além disso, optou-se pelo paradigma da programação orientada a objeto no desenvolvimento do projeto, que também é fundamental para uma maior abstração do código: as classes representam as entidades do sistema e os seus respectivos métodos e atributos descrevem o comportamento e as características das mesmas. Assim, dentre outras características, a orientação a objeto provê, dentre diversas particularidades, algumas como:

- Modularidade: o sistema é dividido em subprogramas menores, onde cada um destes é um módulo independente que pode ser modificado e testado sem depender e/ou interferir no funcionamento dos demais.
- Encapsulamento: a programação orientada a objeto evita o acesso indevido a alguns objetos, seus atributos e funções, visando principalmente a segurança do sistema
- Herança: é possível criar classes a partir de outras já existentes e herde seus atributos e métodos, o que contribui muito quando houver classes que compartilham comportamentos comuns, o que torna o código mais eficiente e fácil de manter.

No arquivo main.ino, tem-se todas as declarações dos objetos que serão utilizados no funcionamento do sistema, com suas respectivas pinagens e nuances. Entretanto, o principal módulo do sistema é o Microwave.cpp, que é justamente invocado a partir da main e recebe como parâmetros de seu construtor todos os objetos citados anteriormente. Nesta classe, há toda a lógica que controla os dispositivos da aplicação, mais especificamente na função turnOnMicrowave.

O método responsável por chamar turnOnMicrowave, por sua vez, é o resolveInput, que recebe um char (caractere) correspondente ao mapeamento realizado para cada botão pressionado no Shield, como 'u' para Up, 'l' para Left, e 's' para Start. Com isso, os botões de Left e Right servem para movimentar o cursor entre cada dígito de unidade e dezena de segundos e minutos, e com o auxílio do Up e Down, é possível realizar o incremento e decremento do tempo de funcionamento do micro-ondas. Por fim, quando o botão Start é pressionado, o sistema começa a funcionar durante o tempo especificado.

Vale ressaltar que, ainda na classe do Microwave existem algumas funções auxiliares denominadas resolve que servem principalmente para realizar as validações dos valores possíveis para cada casa de minuto e segundo, bem como armazenar esse valor nos atributos da mesma

classe. Ainda há um método que traduz o tempo inserido para segundos, tendo em vista que o formato do display é da forma mm:ss. Por fim, o método de countDown realiza o decremento do tempo à medida que o micro-ondas percorre o tempo de funcionamento, e, simultaneamente realiza a atualização do tempo disposto no display.

Dentre os dispositivos que compõe o sistema, aquele que possui o método um pouco mais complexo é o LcdKeypad, que corresponde ao Shield. Foi necessário o desenvolvimento de uma função que traduz cada botão pressionado, tendo em vista que esse dispositivo emite uma saída analógica na qual cada botão é mapeado através de uma faixa de valores conforme explicado na subseção 3.1.2. A função print, ainda na classe LcdKeypad, disponibiliza no display as frases desejadas (que são passadas por parâmetro pela classe Microwave), como "INSIRA O TEMPO:", bem como o tempo de execução que será manipulado pelo usuário. Como adicional, implementou-se uma função blink, que é responsável por piscar a posição em que o cursor está, o que facilita a interação do indivíduo com o sistema, aprimorando a experiência do usuário no mesmo.

Os demais dispositivos, como o LED RGB, o Buzzer e o Motor, possuem códigos mais simples em suas respectivas classes, cujos métodos são responsáveis basicamente por ligar e desligar o componente através das funções disponibilizadas pela própria biblioteca do Arduino, como o digitalWrite, o delay, entre outras.

3.3 Simulação

Além do projeto físico, o grupo optou por realizar uma simulação através do SimulIDE, sugerido pelo professor durante as aulas. Felizmente, o software possui uma estrutura que lida bem com a orientação a objeto, então bastou inserir o código do main.ino no editor do software e especificar o caminho onde os arquivos header se encontravam. Além disso, destaca-se-á que a interface do software possui todos os componentes exatos que utilizamos no projeto, inclusive o LCD Shield[1], o que corroborou imensamente para o desenvolvimento simplificado da simulação.

Finalizada a simulação (vide Figura 3) e realizado alguns testes, verificou-se que o software desenvolvido pelo grupo estava de performando conforme esperado, nos possibilitando partir para a montagem física do sistema.

[1]Vale destacar que o mapeamento de pinos do LCD Shield presente no software de simulação era diferente do Shield real, sendo necessário editar o código fonte deste componente para que fosse possível ter compatibilidade total entre a simulação e o projeto real.

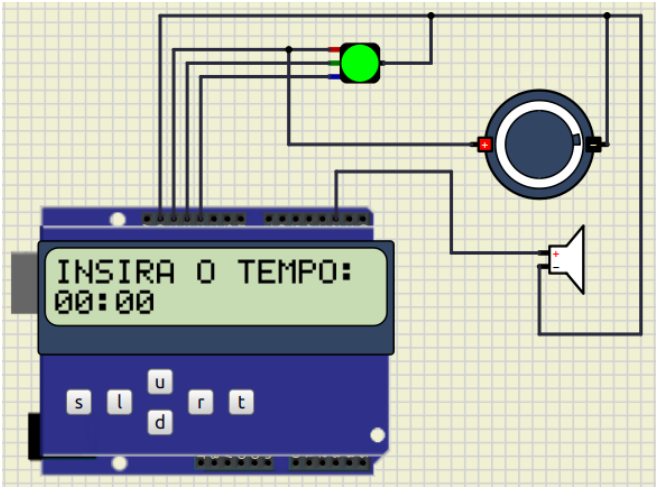

Figura 3: Simulação realizada no SimulIDE

3.4 Resultados Práticos

Após realizar diversos testes no sistema físico, foi constatado que não seria possível utilizar o motor na configuração desejada, uma vez que o mesmo não poderia ser conectado diretamente ao Arduino. Isso se deve ao fato de que a corrente necessária para o funcionamento do motor é de 150mA a 3V ou 200mA a 6V, enquanto a corrente máxima fornecida por porta do Arduino UNO é de apenas 40mA. Sendo assim, seria necessário alimentar o motor com uma fonte externa. No entanto, o grupo não dispunha deste componente, o que impossibilitou o uso do motor no projeto físico. Dessa forma, o sistema final acabou sendo representado pela figura 4.

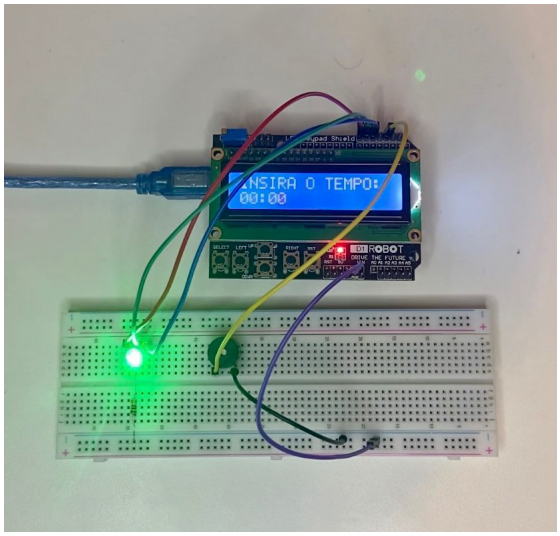

Figura 4: Sistema físico montado

Apesar de não ter sido possível a utilização do motor, o sistema atendeu as expectativas, performando conforme projetado e desejado pelo grupo.

4 Conclusão

O presente projeto teve como objetivo desenvolver uma IHM para um forno de micro-ondas, de forma que fosse possível viabilizar a interação entre um indivíduo e a referida máquina.

Para isso, inicialmente foram consideradas a utilização de um Arduino UNO, um Shield com display LCD e botões, um LED RGB, um Buzzer e um motor DC, de forma que por meio do Shield fosse possível obter os inputs do usuário (no caso o tempo de funcionamento desejado), pelo LED sinalizar ao usuário o estado do sistema (emitindo cor verde quando estivesse pronto para uma nova instrução e vermelha caso já estivesse em funcionamento), pelo Buzzer emitir um som para indicar a finalização da execução do programa, pelo motor rotacionar prato da parte interna do forno de micro-ondas e pelo Arduino controlar os dispositivos referidos e orquestrar a execução dos comandos. No entanto, devido ao grupo não dispor dos requisitos necessários para acoplar o motor ao sistema, optou-se pela remoção deste.

Apesar da impossibilidade da utilização do motor, o sistema atendeu as expectativas, visto que os objetivos foram alcançados com sucesso e os conceitos estudados durante a disciplina de Sistemas Microprocessados puderam ser aplicados com êxito.

5 Apêndice

- O repositório do projeto pode ser encontrado no Github através do link: `https://github.com/azzolinovarella/microwave-hmi`.
- O vídeo exemplo de funcionamento do sistema pode ser encontrado em: `https://youtu.be/cdiGSZVyDLo`