INSTITUTO FEDERAL
Santa Catarina
Câmpus Florianópolis

Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina
Campus Florianópolis
Departamento de Metal Mecânica
Bacharelado em Engenharia Mecatrônica
Eletrônica Digital II - Valdir Noll

Máquina de Café - ATMEGA328p

Breno Cabral

Murilo Bloch

Pedro Saraiva

FLORIANÓPOLIS, julho de 2022

Sumário

Introdução

Café é uma bebida indispensável na vida cotidiana, é uma bebida multicultural e, ainda assim, consegue ser única por sua maneira. Seja em casa, na universidade, em comércios, em escritórios de engenharia, o café se apresenta em inúmeras formas, uma bebida versátil e apreciada por muitos. Opções de preparo se tornam tão amplas quanto as opções de combinação com outras bebidas, não é incomum uns gostam mais de café puro, enquanto outros preferem o famoso “pingado” com leite, ou uma bebida mais requintada, como um *capuccino* para trazer um adocicado extra à boca.

Dado esta gama de variedades, um projeto de uma máquina de café utilizando o microcontrolador ATMEGA328p foi desenvolvido, utilizando a linguagem C pelo software Microchip Studio. Testes e simulações foram realizados utilizando o software de engenharia Proteus e outros artifícios adotados pelos autores durante o desenvolvimento, como simulações com componentes físicos e pela plataforma *online Tinkercad.*

Como já mencionado, a lógica do sistema foi programada em baixo nível, configurando, registradores e seus registros, utilizando assim, recursos do chip como interrupções externas, temporizadores e comunicação serial.

Todos os processos e técnicas envolvidas são exploradas ao longo do desenvolvimento deste documento.

2. Desenvolvimento

Durante o desenvolvimento serão apresentadas abordagens da equipe para o problema, como os recursos disponibilizados foram utilizados e também como o problema apresentado foi resolvido. Nos seguintes tópicos tais considerações são feitas.

Problema

O problema apresentado é desenvolver uma máquina de café com as seguintes características:

- 3 botões para interação: On/Off, Enter e Select;
- 1 sensor de presença para copo, xícara e/ou caneca;
- 1 visor de LED 7 segmentos para visualização da bebida selecionada;
- 1 visor LCD para interação com o usuário;
- 1 monitor serial para interação com técnico da máquina;
- 3 válvulas, uma para cada ingrediente, indicadas por LEDs;
- 1 motor para moer café, indicado por LED;
- 1 LED para indicar estado ligado;
- 1 microcontrolador ATMEGA328p para controle do sistema.

Requisitos de projeto

Algumas características da cafeteira são especificadas como requisitos para serem cumpridos, entre elas, a interação com usuário fica limitada à comunicação com o LCD e o visor de 7 segmentos utilizando os botões Select, Enter e o sensor de copo. O display de 7 segmentos deve apresentar valores de 1 a 7, referentes a cada bebida possível. Além disso, o display LCD deve apresentar o tipo de bebida e valor, seguindo os valores apresentados na tabela a seguir.

Imagem 1 - Tabela LCD

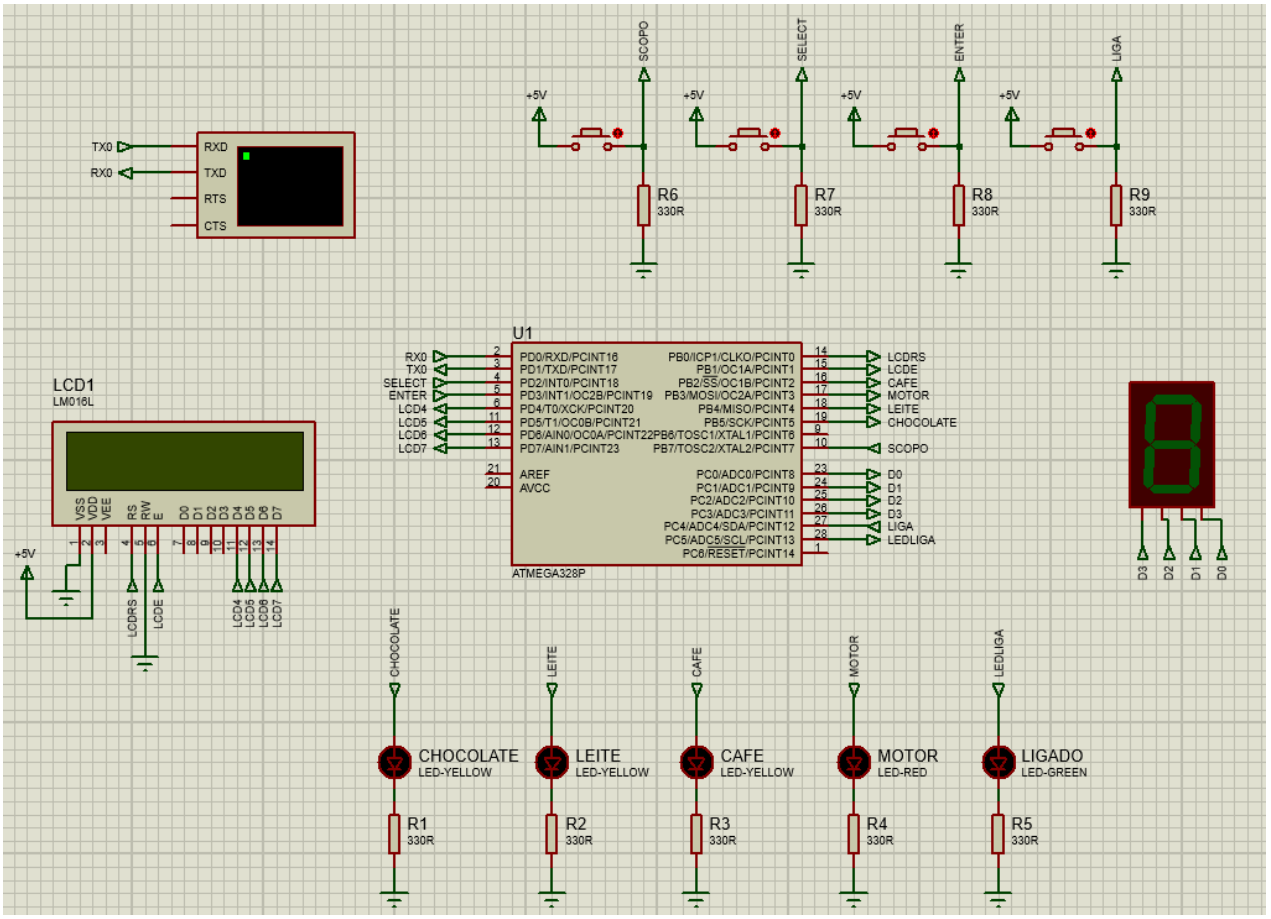
Tipo	Café	Leite	Chocolate	Valor R$
1	x			1,50
2	x	x		2,00
3	x	x	x	2,75
4		x		1,00
5		x	x	1,80
6			x	1,50
7	x		x	2,10

É definido para o sistema requisitos de tempo de preparo para cada bebida, bebidas com um único ingrediente deve ter o tempo de preparo de 6 segundos, com dois ingredientes, 3 segundos e com os três ingredientes, 2 segundos. Todas as bebidas que contêm café, devem acionar o motor por três segundos antes do preparo, para moer o grão. E as válvulas dos ingredientes utilizados devem ser acionadas simultaneamente.
Além da preparação, um modo de reabastecimento também deve ser implementado, este é ativado ao pressionar os botões Enter e Select simultaneamente. Este modo prevê o reabastecimento dos ingredientes, apresenta um relatório de lucro e porções.

Recursos utilizados

Para a programação do chip ATMEGA328p a linguagem C foi utilizada pela interface Microchip Studio, utilizando bibliotecas e funções para auxiliar a programação bit a bit. Para a simulação do código desenvolvido, o software de engenharia Proteus foi utilizado de acordo com o modelo apresentado.

Imagem 2 - Proteus



Proteus é uma ótima ferramenta, já que permite a simulação do projeto sem a necessidade de componentes reais, além de também permitir uma depuração do código em conjunto com seu desenvolvimento, a fim de identificar bugs e potenciais problemas que possam afetar outros módulos do sistema.
A seguir, cada componente utilizado será discutido, mostrando sua integração com o sistema e implementação no firmware.

Botões - Interrupções Externas

O protótipo do sistema apresenta 4 botões no total, a tabela com índices a seguir auxilia na descrição de cada botão.

Tabela 1 - Botões

Índice	Título do botão	Descrição
1	SCOPO	Representa o sensor do copo
2	SELECT	Representa o botão Select
3	ENTER	Representa o botão Enter
4	LIGA	Representa o botão On/Off

O botão 1 (SCOPO) foi aplicado no protótipo como uma representação do sensor do copo, uma vez que o sensor possui comportamento equivalente ao acionamento de um botão, alterando o estado do pino para alto quando acionado e baixo quando desligado.
Os botões 2 e 3 (SELECT e ENTER) representam os dois botões de interação com o usuário, o botão Select tem como objetivo atualizar o visor de 7 segmentos e o Enter confirmar a bebida selecionada para confecção. Funções secundárias também são atribuídas aos botões, como por exemplo, o modo de reabastecimento é ativado ao pressionar os botões 2 e 3 simultaneamente, e para sair deste modo, aperta-se o botão Enter.
O botão 4 (LIGA) simula o botão on/off, para ligar e desligar a máquina, sem outras atribuições.
Para configuração dos botões, a verificação do estado do pino é feita para os botões 1 e 4 (SCOPO e LIGA), já que estes trabalham via retenção, ou seja, os pinos são ativados apenas quando o botão está pressionado.
Já para os botões 2 e 3 (SELECT e ENTER), as interrupções INT0 e INT1 foram configuradas para que estes possam conferir às atribuições sempre que solicitados e a lógica permitir.
Os registradores EICRA e EIMSK foram configurados de acordo; para fins da lógica utilizada, a INT0 (Select) foi configurada para acionamento por mudança de estado lógico, enquanto a INT1 (Enter), foi configurada para acionamento por borda de descida.

Imagem 3 - Interrupções

```
// Interrupcao INT0 -- SELECT --
ISR(INT0_vect)...

// Interrupcao INT1 -- ENTER --
ISR(INT1_vect)...
```

```
// Interrupções Externas
EICRA = 0b00001001; // Habilita interrupções int1 e int0  = enter e select

sei(); // Habilita interrupções globais
```

```
EIMSK = 0b00000011; // Habilita mascara das interrupções int0 e int1
```

Display de 7 Segmentos

O Display de 7 segmentos, atualizado pelo botão select, opera no modo BCD, e a seguinte tabela foi utilizada para a atribuição de valores.

Imagem 4 - Tabela 7 segmentos

D0	D1	D2	D3	
1	0	0	0	1
0	1	0	0	2
1	1	0	0	3
0	0	1	0	4
1	0	1	0	5
0	1	1	0	6
0	0	1	0	7

Sua atualização é feita durante a interrupção do botão Select, limitada para ocorrer durante os estados da máquina ligada ou com copo. Sua lógica é baseada em um sistema de switch case, onde uma variável auxiliar é incrementada toda vez que o botão select é pressionado, alterando a disposição dos LEDs para o valor desejado.

Imagem 5 - Lógica 7 segmentos

```
if (_estado == ComCopo || _estado == Ligado){
    aux++;
    switch(aux){
        case 1:
        set_bit(PORTC, D0); //7 segmentos 1
        clr_bit(PORTC, D1);
        clr_bit(PORTC, D2);
        clr_bit(PORTC, D3);
```

Display Cristal Líquido (LCD)

O LCD é um dos meios de visualização do usuário, por ele é possível mostrar informações sobre as bebidas, como se há ou não algum ingrediente e também um indicativo do estado corrente da máquina.
O LCD utilizado foi o LM016L, o qual possui 2 linhas e a possibilidade de usar 16 caracteres em cada uma delas.O seu comando é simples utilizando a biblioteca LCD, para escrever algo, basta dizer em qual linha o cursor deve começar e o seu conteúdo. Sendo escrito desta forma:

Imagem 6 - Comando LCD

```
// LCD
cmd_LCD(0x80,0);
escreve_LCD("Preparando          ");
cmd_LCD(0xC0,0);
escreve_LCD("Aguarde...          ");
```

LEDs - Temporizadores

Com exceção do botão de liga/desliga, acionado via sinal alto para sinalizar que a cafeteira está ligada, todos os demais LEDs, que representam as válvulas e o motor, têm o acionamento baseado em temporizadores. Isso devido aos requisitos de tempo de preparo de cada bebida. Os temporizadores também acionam interrupções para realizar a contagem de tempo, os utilizados no projeto foram o TC1 eTC 2.

Timer 1 - TC1

O TC1 foi utilizado como um temporizador “global” que realiza uma interrupção a cada segundo. Parâmetros para alcançar tal medida foram configurados em seus registradores durante o setup do programa como mostra a imagem a seguir.

imagem 7 - Registradores TC1

```
// Timer 1 1s
TCCR1A = 0; //limpa registrador
TCCR1B = 0; //limpa registrador
TCNT1  = 0; //zera temporizado

OCR1A = 0x3D09;           // carrega registrador de comparação: 16MHz/1024/1Hz = 15625 = 0X3D09
TCCR1B |= (1 << WGM12);   // modo CTC prescaler de 0
TIMSK1 |= (1 << OCIE1A);  // habilita interrupção por igualdade de comparação
```

O timer é acionado durante o preparo da bebida para fazer o controle do tempo, e assim, definir quando as válvulas e o motor devem ser acionados e desabilitados.

imagem 8 - Acionamento TC1

```
void preparoBebida(){
    TCCR1B = 0b00001101;   // prescaler de 1024: CS12 = 1 e CS10 = 1
    int tempoPreparo = 0;
    segundos = 0;
```

Sua interrupção baseia-se na incrementação de uma variável do tipo volatile int, que indica a quantidade de segundos contados.

imagem 9 - Interrupção TC1

```
// Timer 1
ISR(TIMER1_COMPA_vect){
    segundos++;
}
```

Timer 2 - TC1

O timer 2 foi configurado numa frequência de 1kHz para simular a frequência de modulação de onda PWM do motor, indicado por um LED. Assim como o TC1, seus registradores foram configurados durante a função de setup do código e seu acionamento foi feito quando necessário de acordo com a lógica do código. As imagens a seguir demonstram como isso foi aplicado no programa.

imagem 10 - Registradores TC2

```
// Timer 2 1kHz
TCCR2A = 0; //limpa registrador
TCCR2B = 0; //limpa registrador
TCNT2  = 0; //zera temporizador

OCR2A = 125;           // carrega registrador de comparação: 16MHz/2*64/1kHz = 125
TCCR2A = (1 << COM2A0)|(1 << WGM21); // modo CTC
TCCR2B = 0;    // prescaler de 0
```

imagem 11 - Acionamento motor

```
switch(bebidaSelecionada)
{
    case 1:
        setMotor(1);    //liga o motor
        while(segundos < 3){}; //espera 3s
        setMotor(0); //desliga o motor
```

imagem 12 - Função motor

```
void setMotor(int condicao){
    if (condicao == 1)
        TCCR2B = 0b00000100; // prescaler de 64 Liga Motor
    if (condicao == 0)
    {
        TCNT2  = 0;    //zera temporizador
        TCCR2B = 0;    // prescaler de 0 Desliga Motor
    }
}
```

Comunicação Serial

Para a comunicação serial, foi utilizado uma taxa de 9600 bits/s, pacotes com 8 bits, sem paridade e 1 stop bit. Para sua configuração, é necessário o cálculo do UBRRO, e depois a definição dos registradores.

Imagem 13 - Registradores do baud rate

```
//configurar a serial para 9600 bits/s
UBRR0H = CALCULO_UBRRO >> 8;
UBRR0L = CALCULO_UBRRO;

//configura a serial p trabalhar com 8 bits, sem paridade, 1 stop bit ... 8N1
UCSR0C = (1<<UCSZ00) | (1<<UCSZ01);

// Habilita a transmissao, recepcao por interrupcao serial
UCSR0B = 0b10011000;
```

Para o envio de mensagens pela serial, é necessário converter a mensagem para um formato em string. Para isso, foi utilizado a função sprintf, que é utilizada para formar uma

string e guardar o resultado em um array, no caso msg[], desta forma, é possível fazer o envio correto da mensagem via serial. Abaixo vemos um exemplo de uma mensagem transmitida.

Imagem 14 - Comandos Serial

```
// Envio de LOG para Serial
char msg[50];
sprintf(msg,"Simulacao Proteus Iniciada. Realizando Setup\r\n");
transmiteStringSerial(msg);
```

Lógicas Desenvolvidas

Para desenvolver o protótipo funcional, a equipe projetou um sistema de lógicas e funções baseado em um diagrama de máquina de estados finitos (FSM). A FSM foi construída em cima de 5 estados básicos do sistema; a partir de cada estado uma lógica própria foi construída integrando os recursos e módulos disponíveis utilizando funções e variáveis auxiliares durante o processo.

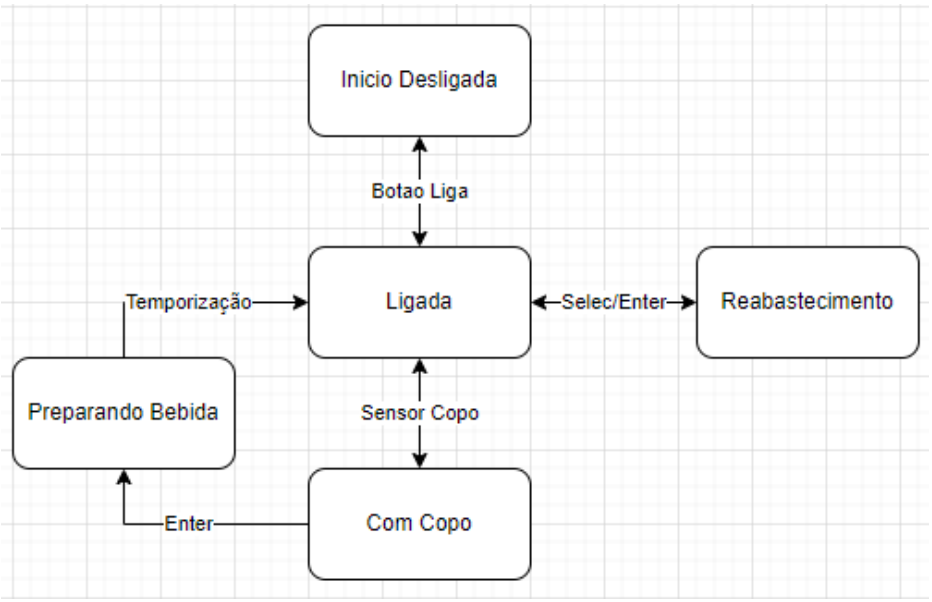
Máquina de estados finitos

Como mencionado, a FSM foi construída a partir de 5 estados básicos do sistema:

- Estado Desligada;
- Estado Ligada;
- Estado em Reabastecimento;
- Estado Com Copo;
- Estado Preparando Bebida.

O diagrama de estados a seguir apresenta quais módulos integram e como essa integração de estados é feita.

Imagem 15 - Máquina de estados finitos



Integração dos estados

Durante o programa desenvolvido no ambiente do Microchip Studio, a FSM foi enumerada, controlada por um switch case em loop e convertida em funções para que a integração entre os estados pudesse ser realizada.

Imagem 16 - Switch case FSM

```
// Loop
while(1){
    switch (_estado){
        case Desligado:
            desligado();
        break;

        case Ligado:
            ligado();
        break;

        case ComCopo:
            comCopo();
        break;

        case PreparoBebida:
            preparoBebida();
        break;

        case Reabastecimento:
            reabastecimento();
        break;
    }
```

Cada função chamada em seu respectivo estado possui uma integração com o estado seguinte baseado na lógica do sistema e no diagrama de FSM. O estado inicial da máquina é desligado, então ao apertar o botão on/off, a máquina deve seguir para o estado Ligado, e tal verificação é o que ocorre na função desligado().

Imagem 17 - Função desligado()

```
void desligado(){
    clr_bit(PORTC, LED_ONOFF);
    FLAG_Select = 0;

    cmd_LCD(0x80,0);
    escreve_LCD("Desligada          ");
    cmd_LCD(0xC0,0);
    escreve_LCD("                   ");
    if(firstTimeOff == 0)
    {
        // Envio de LOG para Serial
        char msg[50];
        sprintf(msg,"Máquina Desligada \r\n");
        transmiteStringSerial(msg);
        firstTimeOff = 1;
    }
    if (rd_bit(PINC, BT_ONOFF) > 0 ){
        firstTime = 0;
        _estado = Ligado;
        return;
    }
}
```

Durante esta função, o sistema verifica se o botão on/off foi pressionado, e caso positivo, altera o estado para Ligado.
No modo ligado, além de várias verificações e rotinas para lógica, também há a verificação caso o botão seja pressionado para desligar a máquina, ou, caso o copo seja inserido para acessar o estado Com Copo.

Imagem 18 - Função ligado()

```
if (rd_bit(PINC, BT_ONOFF) ==0 ){
    //Muda estado para Desligado
    firstTimeOff = 0;
    _estado = Desligado;
}

if(rd_bit(PINB, S_COPO) > 0){
    // Envio de LOG para Serial
    sprintf(msg,"Maquina Estado - Com copo. Aguardando o botao ENTER ser pressionado. \r\n");
    transmiteStringSerial(msg);
    //Muda estado para comCopo
    _estado = ComCopo;
}
```

Caso vá para o estado Com Copo, verifica-se se não ocorre a retirada do mesmo, resultando no retorno ao estado Ligado.

Imagem 19 - Função comCopo()

```
void comCopo(){
    ;
    if(rd_bit(PINB, S_COPO) == 0 )
    {
        _estado = Ligado;
    }
}
```

Para acessar os demais estados, as interrupções via os botões Select e/ou Enter devem ser acionadas, acessando o estado PreparoBebida ou Reabastecimento.

Imagem 20 - Interrupção Enter, PreparoBebida

```
if (rd_bit(PINB, S_COPO)> 0){
    int var = VerificaIngredientes();

    sprintf(msg,"%d",var);
    transmiteStringSerial(msg);

    FLAG_Select = 0;
    switch(var)
    {
        case 0: //tudo ok
            // Envio de LOG para Serial

            sprintf(msg,"Todos os ingredientes disponiveis. Preparando bebida \r\n");
            transmiteStringSerial(msg);

            _estado = PreparoBebida;
        break;
```

Durante a interrupção INT1 (Enter), há diversas rotinas e verificações para a preparação (ou não) da bebida, a lógica de verificação de ingredientes retorne que tudo está OK, a máquina acessa o estado PreparoBebida.

Imagem 21 - Função preparoBebida()

```
void preparoBebida(){
    TCCR1B = 0b00001101;   // prescaler de 1024: CS12 = 1 e CS10 = 1
    int tempoPreparo = 0;
    segundos = 0;
        // LCD
        cmd_LCD(0x80,0);
        escreve_LCD("Preparando          ");
        cmd_LCD(0xC0,0);
        escreve_LCD("Aguarde...          ");

    switch(bebidaSelecionada)
    {
        case 1:
            setMotor(1);    //liga o motor
            while(segundos < 3){}; //espera 3s
            setMotor(0); //desliga o motor

            segundos = 0; //zera o parametro de tempo

            tempoPreparo = 6;
            setValvula(1,1); //Abre valvula cafe
            // Balanceio
            segundosCafe -= tempoPreparo;
            Quantidades[0] += 1;
            valorArrecadado+= 1.50;
        break;
```

Caso o botão Select também seja pressionado junto ao Enter, a máquina altera o estado para Reabastecimento.

Imagem 22 - Interrupção Enter, Reabastecimento

```
if (FLAG_Select){

    //LCD
    cmd_LCD(0x80,0);
    escreve_LCD("MODO REBASTECIMENTO");
    cmd_LCD(0xC0,0);
    escreve_LCD("                    ");

    // Envio de LOG para Serial
    sprintf(msg,"Entrando no modo reabastecimento  \r\n");
    transmiteStringSerial(msg);
    ShowAbastecimento = 0;

    _estado = Reabastecimento;
    return;
}
```

3 - Conclusão

Testes do projeto foram realizados ao longo de sua programação utilizando o software Proteus, isso permitiu aos autores uma verificação quase instantânea de possíveis erros e bugs que ocorriam ao longo do processo e com isso correções puderam ser implementadas com mais agilidade e praticidade.

Com o desenvolvimento do projeto, é possível ver o quão complexo um código tende a se tornar. A cadeia, quase exponencial, de influência que uma linha pode gerar em todo o código é notável; dificultando a rastreabilidade de problemas que surgem. O entendimento dos processos envolvidos, além do bom uso das linhas de código, gera um código limpo e eficiente. Objetivo que torna-se extremamente distante sem tal organização.

Testes em Simulação

Inúmeros testes e correções foram realizados ao longo do processo já que a programação e a simulação foram feitas simultaneamente, sempre que alguma nova função ou lógica era implementada, novos testes eram realizados, os resultados verificados e as correções, ou novas funcionalidades, implementadas. Essa metodologia empregada impossibilitou a criação de versões diferentes da máquina, uma vez que novas atualizações eram sempre adaptadas à integração.

Como resultado final, a equipe definiu que o preenchimento de todos os requisitos era o objetivo, e a partir de tal ponto, a correção de bugs e implementação de features mais detalhadas poderiam ser feitas.

O resultado foi alcançado. A máquina final cumpre todos os requisitos e os testes apresentam bom desempenho, apesar de conter bugs e espaços para implementação de funcionalidades complementares que podem incrementar no funcionamento geral do sistema.

Referências bibliográficas

Charles Borges de Lima; Marcos V. M. Villaça. AVR e Arduino: Técnicas de Projetos. 2ª Edição.

Fábio Souza. Timers do ATmega328 no Arduino. Embarcados, 2015. Disponível em: <https://embarcados.com.br/timers-do-atmega328-no-arduino/>. Acesso em: 08/07/2022.