Washington Holanda de Oliveira
RA: 112268

Desenvolvimento de um Sistema de Gerenciamento de Remédios Controlados

São José dos Campos - Brasil

Maio de 2019

Washington Holanda de Oliveira
RA: 112268

Desenvolvimento de um Sistema de Gerenciamento de Remédios Controlados

Relatório apresentado à Universidade Federal de São Paulo como parte dos requisitos para aprovação na disciplina de Sistemas Embarcados.

Docente: Prof. Dr. Sérgio Ronaldo Barros

Universidade Federal de São Paulo - UNIFESP

Instituto de Ciência e Tecnologia - Campus São José dos Campos

São José dos Campos - Brasil

Maio de 2019

Sumário

1 DESCRIÇÃO DO PROJETO 3
1.1 Descrição 3
1.2 Objetivo 3

2 FUNCIONAMENTO DO SISTEMA 5
2.1 Hardware 5
2.1.1 Real Time Clock RTC DS1307 5
2.1.2 Módulo Bluetooth HC-05 5
2.1.3 Display LCD 20x4 5
2.1.4 Buzzer 6
2.1.5 Light Emitting Diode(LED) 6
2.1.6 Esquemático do Sistema 7
2.2 Software 8
2.2.1 Bibliotecas 8
2.2.2 Declarações Iniciais 8
2.2.3 Configurações Iniciais 9
2.2.4 Funções 10
2.2.4.1 Função Horário 10
2.2.4.2 Função de Notificação 11
2.2.4.3 Função Atualiza 12
2.2.5 Rotina Principal 14

3 LISTA DE COMPONENTES 15

1 Descrição do Projeto

1.1 Descrição

O projeto visa desenvolver um sistema de gerenciamento de remédios controlados, em outras palavras, um sistema no qual um usuário de remédios controlados tenha controle de em qual remédio deve ser tomado e o horário exato que deve ser tomado, sendo notificado sempre que um remédio precisar ser tomado.

Para o controle desse sistema foi utilizado para a obtenção de informações como data e hora e todo controle de horários um módulo Real Time Clock RTC DS1307 que passava pro Arduino todas essas informações para o tratamento dos dados.

Para a comunicação entre o usuário e o sistema foi utilizado um display LCD 20x4 que exibia em modo stand-by a exata hora e data, além de mostrar informações como nome do medicamento e dosagem correta sempre que uma uma notificação de que estava na hora de tomar um remédio era apresentada. Além do display, pensando em um sistema final onde o usuário teria compartimentos com cada remédio que o sistema gerenciaria, foram utilizados LED's sinalizadores nos quais acendiam sempre que o remédio correspondente a devesse ser tomado. Um buzzer também foi usado para avisos sonoros do sistema sempre que uma notificação de remédio fosse lançada, além de dois botões nos quais o usuário comunicaria o sistema que o remédio notificado havia sido tomado, fazendo com que o sistema voltasse ao modo de stand-by e fizesse as devidas atualizações.

Além de todos os itens mencionados um módulo HC-05 (módulo Bluetooth) foi usado para que o sistema pudesse notificar o usuário em seu próprio smartphone, enviando um alerta para um aplicativo desenvolvido informando que um remédio deve ser tomado. Para uso do mesmo, foi utilizado alguns resistores de modo que um divisor de tensão fosse feito para controle correto da alimentação do módulo.

1.2 Objetivo

Notificar um usuário sobre o horário correto que certo medicamento controlado deve ser tomado, apresentando a ele informações como nome do remédio e dosagem correta que deve ser tomada, sendo essas notificações feitas por diversas vias, sejam elas por interfaces do próprio sistema ou via smartphone. Além disso, fazer todo o controle e gerenciamento de atualizações de horários em que cada remédio deve ser tomado, icentando o usuário de preocupações a cerca dos horários corretos.

2 Funcionamento do Sistema

2.1 Hardware

Esta sessão apresenta uma descrição detalhada de cada componente utilizado no projeto e como eles se interligam de maneira a gerar o funcionamento do sistema.

2.1.1 Real Time Clock RTC DS1307

O módulo utilizado para controle de informações como hora e data foi o RTC DS1307, um módulo que como o próprio nome já diz, funciona como um relógio em tempo real. Esse módulo faz uso de uma memória interna não-volátil de 56 bytes que guarda todas as informações de hora, data e dia da semana, podendo ser utilizado no modo 12h ou 24h, sendo o segundo o modo utilizado nesse projeto. Além disso o módulo também conta com um sensor de temperatura interno que não será utilizado. Para seu funcionamento no projeto foram utilizadas quatro de suas portas, sendo elas: as duas de alimentação (VCC e GND), que foram ligadas na alimentação de 5V do Arduino, e duas portas de comunicação I2C (SDA e SCL) ligadas nas portas de comunicação I2C do Arduino.

2.1.2 Módulo Bluetooth HC-05

O módulo para a comunicação com o smartphone do usuário foi o módulo HC-05 que foi utilizado no modo escravo, no qual ele apenas aceita pareamentos. Esse módulo possui 6 entradas das quais apenas 4 foram utilizadas, sendo elas a alimentação do módulo (VCC e GND) e as portas de comunicação serial (TX e RX). Apesar de a alimentação do módulo ser 5V, os pinos RX e TX utilizam sinais de 3.3V para se comunicar. Desta forma, foi necessário utilizarmos divisor de tensão para obter aproximadamente 3.3V a partir da saída de 5V do TX do Arduino. Para esse divisor de tensão foram utilizados três resistores de 1KΩ cada, sendo dois em série formando um resistor de 2KΩ para que pudesse ser alcançado o valor aproximado de 3.3V.

2.1.3 Display LCD 20x4

O display LCD utilizado possui 16 pinos de entrada dos quais, 8 são pinos (DB0-DB7) que fazem a troca de dados com o microcontrolador, porém nesse projeto o display foi utilizado no modo 4-bits, fazendo com que essa troca de dados seja feita apenas por 4 desses pinos (DB4-DB7). Os outros 8 pinos do display são usados para alimentação do módulo (VCC e GND) usando uma tensão de 5V, controle de contraste (V0) no qual foi

utilizado um potenciometro de 10K para que fosse possível variar o contraste da forma desejada, ativação da luz de fundo do display (A e K), e para o controle de funcionamento (RS,RW e E).

Além disso o módulo escolhido possuia, por necessidade de quantidade de informações a serem mostradas no projeto, quatro linhas de vinte caracteres cada, sendo cada um desses caracteres descrito por uma matriz de 8x5 bits.

2.1.4 Buzzer

O buzzer foi utilizado para sinalizações sonoras do sistema, ele possui internamento componentes piezoelétricos que vibram gerando o ruído que emite o sinal sonoro desejado. Para controle do som emitido foi utilizado um resistor de 300 ohms no qual controlava o nível da corrente que era mandada ao componente, fazendo com que o som emitido fosse mais baixo.

2.1.5 Light Emitting Diode(LED)

Foram utilizados três LED's para efeito de testes do projeto, porém nesse sistema a função do LED é apenas indicar um medicamento em específico, acendendo quando aquele medicamento deve ser tomado, portanto o número de LED's utilizados pode variar de acordo com o número de medicamentos que serão inseridos no sistema. Para cada LED foi utilizado também um resistor de 300 ohms que fazia o controle da corrente que chegava a cada LED, controlando a intensidade do brilho de cada um e evitando possíveis sobrecargas.

2.1.6 Esquemático do Sistema

O esquemático do projeto, contendo todos os componentes necessários para sua implementação, está apresentado na Figura1.

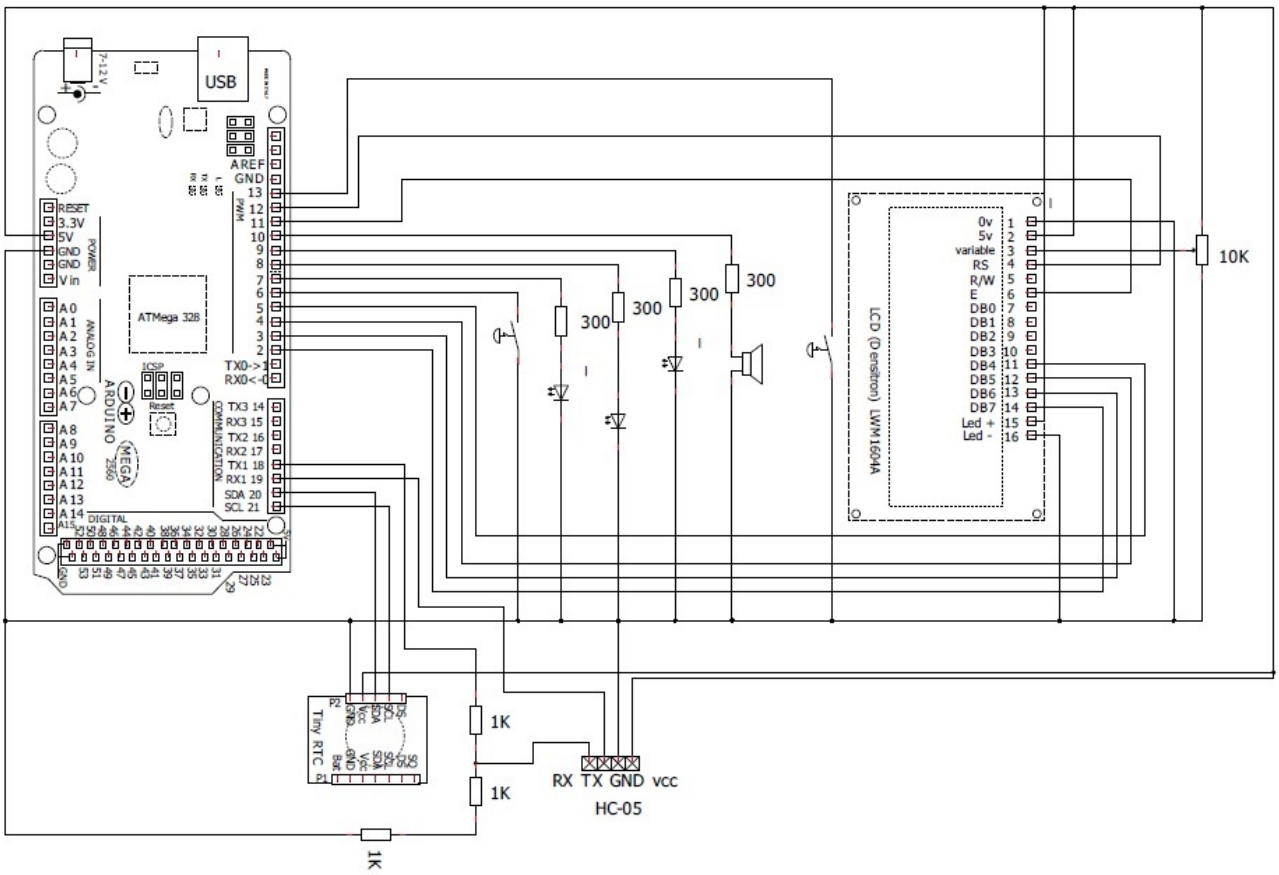

Figura 1 – Esquemático do circuito

Fonte: O Autor; software: QEletroTech

2.2 Software

Nesta sessão, serão apresentados os principais pontos no que diz respeito ao software do projeto, desde as bibliotecas utilizadas até as principais funções do código fonte.

2.2.1 Bibliotecas

No que diz respeito as bibliotecas utilizadas, além das já integrada LiquidCrystal,foi necessário importar a biblioteca externa DS1307 que é a biblioteca que faz todo o controle do módulo RTC. Suas principais funções no geral dizem respeito a como programar inicialmente o módulo ou como retirar informações mais específicas sobre hora ou data(como por exemplo os minutos) para utilização no código fonte do Arduino.

```
// --- Bibliotecas ---
#include <DS1307.h>             //Inclui a biblioteca do DS3231 Shield
#include <LiquidCrystal.h>      //Biblioteca para o display LCD
```

Figura 2 – Bibliotecas Utilizadas

Fonte: O Autor

2.2.2 Declarações Iniciais

Para o funcionamento correto do código foram declaradas diversas variáveis auxiliares, mas principalmente foram declarados os valores para cada pino do Arduino que seria utilizado no código.

Além disso foi declarada uma estrutura na qual seriam armazenadas as informações de cada remédio. Como pode ser visto na Figura 3, essa estrutura armazena informações como: nome do medicamento, dosagem que deve ser tomada, horário em que a primeira dose foi tomada e o período entre cada dose. São essas informações que serão usadas para todas as verificações e atualizações do sistema.

```
// --- Variáveis ---
Time T;
int hora;
int minuto;
int segundo;
int estado = 1;
int buzzerPin = 10;
int buttonPin_tomouNaHora = 13;
int buttonPin_atraso = 6;
int vetLedPin[3] = {9,8,7};
int control = 0;
int notf=0;
int cont_aux=0;

// --- Declaração dos Compartimentos de Remédios ---
typedef struct{
  char nome[30];
  char dosagem[20];
  int hora;
  int minuto;
  int segundo;
  int periodo_entre_doses[2];
  int atraso;
}Medicamento;

Medicamento remedio[3] = {{"Dipirona","30 Gotas",23,58,0,{0,6},0},{"Paracetamol","1 Comprimido",23,12,0,{0,5},0}
{"Tilenol","2 Comprimidos",22,42,0,{0,5},0}};
```

Figura 3 – Declarações do Código-fonte

Fonte: O Autor

2.2.3 Configurações Iniciais

Nesta sessão será mostrado como foi configurado o Arduino de acordo com as necessidades do circuito montado. Na Figura 4 abaixo é mostrado como são configurados os pinos que serão ligados os leds, botões, buzzer, além de mostrar também as configurações da comunicaçõa serial e do LCD usados no circuito.

O destaque nessa parte vai para as configurações iniciais do módulo RTC, que possui a necessidade de ser configurado com as informações do inicio do funcionamento do circuito, após isso o próprio módulo por possuir uma memória interna e uma alimentação auxiliar, não necessita uma segunda configuração, apenas se a alimentação do módulo seja cortada.

```
// --- Configurações Iniciais ---
void setup(){
  Serial.begin(9600);   //Inicia comunicações Serial em 9600 baud rate
  Serial1.begin(9600);   //Inicia comunicações Serial 1 em 9600 baud rate
  pinMode(buzzerPin, OUTPUT); //Inicializa o pino do buzzer como de saída
  pinMode(vetLedPin[0], OUTPUT);  //Inicializa o pino do Led 1 como de saída
  pinMode(vetLedPin[1], OUTPUT);  //Inicializa o pino do Led 2 como de saída
  pinMode(vetLedPin[2], OUTPUT);  //Inicializa o pino do Led 3 como de saída
  pinMode(buttonPin_tomouNaHora, INPUT);      //Inicializa o pino do botão como de entrada
  pinMode(buttonPin_atraso, INPUT);      //Inicializa o pino do botão como de entrada
  rtc.halt(false);     //Aciona o relogio
  lcd.begin(20,4);     //Inicializa LCD 20 x 4
  lcd.clear();         //Limpa LCD

// --- Configuração Inicial da Data e Hora do Sistema ---
  //Descomentar as linhas a seguir para configurar o horário,
  //após comente e faça o upload novamente para o Arduino
  //rtc.setDOW(THURSDAY);     // Set Day-of-Week to SUNDAY
  //rtc.setTime(22,12, 0);     // Set the time to 12:00:00 (24hr format)
  //rtc.setDate(02,05,2019);
} //end setup
```

Figura 4 – Configurações Iniciais

Fonte: O Autor

2.2.4 Funções

Nesta sessão serão apresentadas todas as funções que fazem a rotina completa do código-fonte do sistema.

2.2.4.1 Função Horário

Uma função simples, na qual tem como função principal gerar um modo standby do sistema, ou seja, quando não há nenhuma notificação sendo apresentada o sistema entra nesse modo que nada mais é que um modo onde o LCD apresentará informações como data, hora e dia da semana atuais, sendo atualizados a cada segundo.

```
// --- Imprime a hora/data ---
void horario(){
  lcd.clear();
  lcd.setCursor(0,0);                 //Posiociona cursor na coluna 1, linha 2
  lcd.print(" Medicine Controller");  //Imprime a data
  lcd.setCursor(6,1);                 //Posiciona cursor na coluna 1, linha 1
  lcd.print(rtc.getTimeStr());        //Imprime horário
  lcd.setCursor(5,2);                 //Posiociona cursor na coluna 1, linha 2
  lcd.print(rtc.getDateStr(2,1,'/')); //Imprime a data
  lcd.setCursor(6,3);                 //Posiociona cursor na coluna 1, linha 2
  lcd.print(rtc.getDOWStr());         //Imprime a dia da semana
} //end horario
```

Figura 5 – Função Horário

Fonte: O Autor

2.2.4.2 Função de Notificação

Outra função simples do código-fonte que basicamente configura todas as notificações que o usuário precisa receber quando estiver na hora de um remédio, seja sonora, seja luminosa, visual (lcd) ou até mesmo no celular. Como podemos ver na Figura 6, há uma verificação no começo da função, essa verificação serve para manter a notificação ativa por 3 segundos antes de voltar para a função principal, isso se fez necessário pois caso o usuário tenha que ser notificado sobre 2 remédios ao mesmo tempo o sistema mostrará no display cada um dos remédios durante 3 segundos, alternando entre cada medicamento que deve ser notificado.

```
// --- Envia a notificação pro usuário ---
void notificacao(int i){
  long tempo;

  tempo=millis();
  while(millis()-3000<tempo){
    digitalWrite(buzzerPin, HIGH); //  Toca Buzzer de alerta
    digitalWrite(vetLedPin[i], HIGH); //  Acende Luz de alerta
    lcd.clear();
    lcd.setCursor(2,0);
    lcd.print("HORA DO REMEDIO!");
    lcd.setCursor(5,1);
    lcd.print(remedio[i].nome);   //  Imprime o nome do remédio
    lcd.setCursor(4,2);
    lcd.print(remedio[i].dosagem);  //  Imprime a dosagem do remédio
    delay(100);
  }
  if(!cont_aux){  //Envia notificação no celular do usuário
    Serial1.println(22);
    cont_aux=1;
  }

} //end notificação
```

Figura 6 – Função Notificação

Fonte: O Autor

2.2.4.3 Função Atualiza

Essa função é a principal função do código, visto que os horários de cada remédios devem ser atualizados corretamente seguindo a prescrição médica. Podemos dividir essa função em duas partes, sendo a primeira delas para atualizações caso tenha ocorrido algum atraso no horário em que o remédio devia ter sido tomado, ou seja, caso exista esse atraso ele deve ser considerado na hora de atualizar o novo horário (Figura 7).

```
void atualiza(int i){
    T = rtc.getTime();  //Variável que armazena o horário atual para fazer as comparações adequad
    hora = T.hour;
    minuto = T.min;
    segundo = T.sec;

    cont_aux=0;
    control=0;
    digitalWrite(vetLedPin[i], LOW); //apaga o led do remédio correspondente

    //  Atualização do horário da próxima notificação caso haja atraso
    if(remedio[i].atraso){
      remedio[i].hora = remedio[i].periodo_entre_doses[0] + hora;
      if(remedio[i].hora>=24){
        remedio[i].hora= remedio[i].hora - 24;
      }
      remedio[i].minuto = remedio[i].periodo_entre_doses[1]+ minuto;
      if(remedio[i].minuto>=60){
        remedio[i].minuto= remedio[i].minuto - 60;
        remedio[i].hora= remedio[i].hora + 1;
      }
      remedio[i].atraso=0;
    }
```

Figura 7 – Função Atualiza para o caso de ocorrência de atrasos

Fonte: O Autor

Já a segunda parte diz respeito a atualizações caso não tenha ocorrido atrasos e o usuário tomou o remédio no horário (ou na tolerância de 2 minutos estabelecida). Abaixo podemos verificar que a atualização é feita com base no horário que estava armazenado inicialmente no remédio, diferentemente do caso de atrasos que a atualização é feita considerando o horário atual.

```
    else{ //  Atualização do horário da próxima notificação caso não haja atraso
      remedio[i].hora = remedio[i].periodo_entre_doses[0] + remedio[i].hora;
      if(remedio[i].hora>=24){
        remedio[i].hora= remedio[i].hora - 24;
      }
      remedio[i].minuto = remedio[i].periodo_entre_doses[1]+ remedio[i].minuto;
      if(remedio[i].minuto>=60){
        remedio[i].minuto= remedio[i].minuto - 60;
        remedio[i].hora= remedio[i].hora + 1;
      }
      remedio[i].atraso=0;
    }
}
```

Figura 8 – Função Atualiza para o caso de não ocorrer atrasos

Fonte: O Autor

2.2.5 Rotina Principal

A rotina principal do código é bem simples, ela basicamente consiste em um looping de verificações constantes se há algum remédio a ser tomado, caso haja se ele já foi tomado e se passar do tempo de tolerância se o botão de atraso já foi pressionado. O fluxograma da Figura **??** abaixo ilustra seu funcionamento de maneira mais simples.

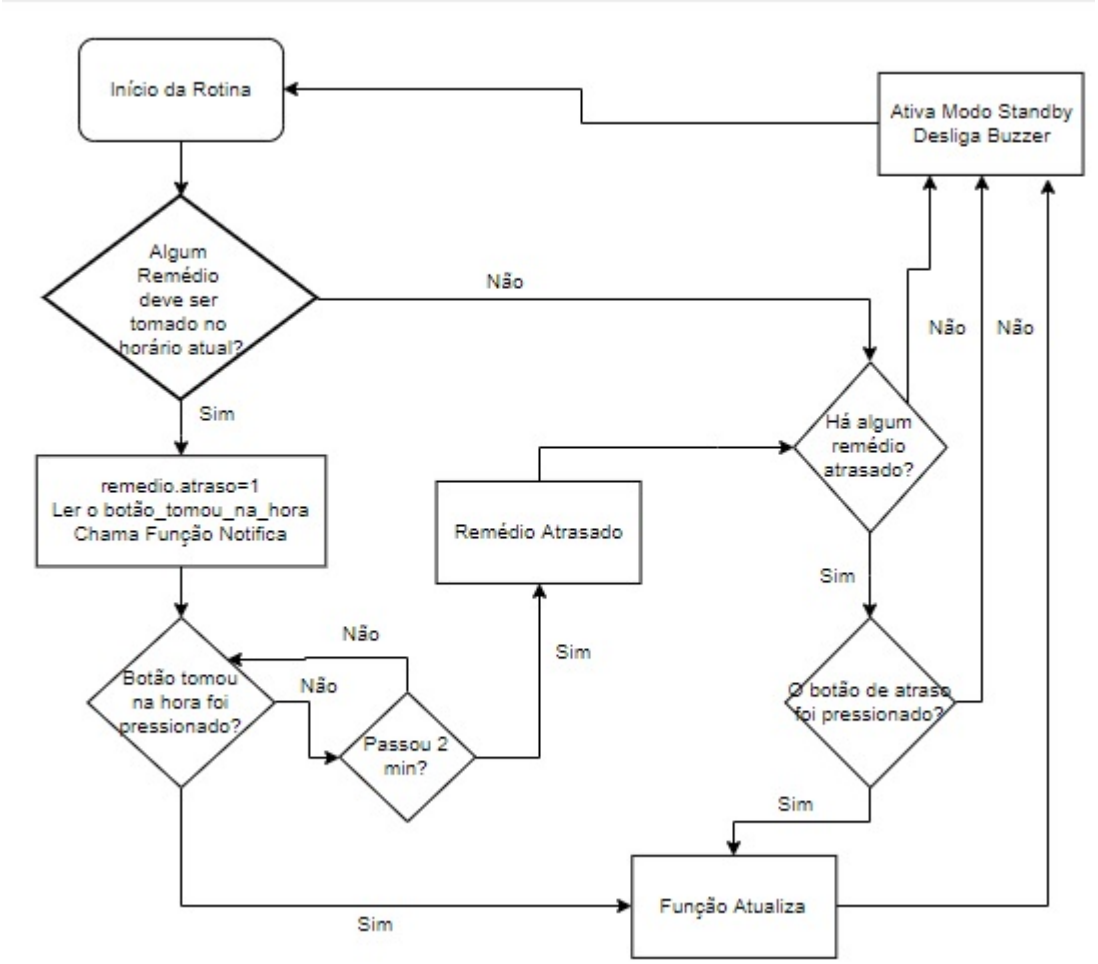

Figura 9 – Rotina Principal do Código-fonte

Fonte: O Autor

3 Lista de Componentes

- Arduino Mega 2560 – Microcontrolador;
- Shield Arduino - Módulo RTC de Precisão DS1307 – Relógio;
- Shield Arduino - Módulo Bluetooth HC-05 – Comunicação com Smartphone;
- Display LCD 20x4 Azul– Interface visual;
- Buzzer – Alarme sonoro;
- Leds - Sinalização Luminosa;
- Resistores de 300Ω - Controle de corrente dos LED's;
- Potenciometro de 10KΩ - Controle de contraste do LCD;
- Push Botton - Controle do usuário;
- Protoboard - Matriz para prototipagem do projeto;
- Jumpers - Conectores básicos.