



COLÉGIO DE SÃO MIGUEL

**CURSO CIENTÍFICO-TECNOLÓGICO DE
INFORMÁTICA
ANO LETIVO 2020/2021**

Luminosa

Relatório De Prova de Aptidão Tecnológica

Aluno: Daniel Ferreira Rodrigues

Nº5 12ºD

Professor Orientador: Paulo Figueira

Fátima, 7 de junho de 2021

ÍNDICE

Introdução	1
Objetivos	2
Material Usado	4
Escrever título do capítulo (nível 2).....	4
Escrever título do capítulo (nível 3).....	5
Software Usado.....	6
Arduino IDE.....	7
C++.....	8
Android Studio.....	9
Java	10
Tinkercad.....	10
Fritzing	10
Desenvolvimento.....	11
Etapa 1 – Levantamento de Ideias e Material	11
Etapa 2 – Programação do Relógio e Alarme.....	13
Etapa 3 – Programação do LED's	20
Etapa 4 - Botões.....	27
Etapa 5 – Estrutura do Projeto	33
Etapa 6 – Bluetooth Arduino.....	34
Etapa 7 - Aplicação.....	37
Netgrafia.....	16
Objetivos	17
Agradecimentos.....	18
Objetivos	19

Introdução

A PAT (Projeto de Aptidão Tecnológica) é um projeto pessoal onde podemos usar a nossa imaginação, os nossos conhecimentos adquiridos ao longo do curso e poder um pouco mais além através de pesquisas na internet para que possamos sair também um pouco da área de conforto e tentarmos aprender um pouco mais por nós mesmos através de pesquisas.

Para a minha PAT escolhi algo que me interessou, um candeeiro com arduino e que pudesse, não só usá-lo manualmente, mas também com controlo “remoto”, através de uma aplicação android programada por mim.

...

Objetivos

Para este projeto usei conhecimentos na eletrônica(arduino), desenho 3D(Tinkercad) para desenhar a estrutura e programação (arduino e android).

O objetivo deste projeto é fazer um candeeiro com relógio e alarme. O candeeiro irá ser iluminado através de Leds endereçáveis, com alguns efeitos já programados como por exemplo uma sequência de luzes que faça parecer um arco íris. Para além disso o usuário terá um potenciômetro que irá funcionar para acelerar a velocidade com que a luz passa na fita de Leds. O usuário para além de uma sequência de efeitos que estão prontos para se usar, irá poder ter um relógio e um alarme que irá tocar com a ajuda de um Buzzer.

....

Material Usado

Para a parte da eletrônica usei algum do material que o kit arduino já trazia e outros que tive de mandar encomendar por via Internet já que não eram incluídos no kit.

- Bluetooth module HC-06;
- RTC DS1307/DS3231;
- Placa Breadboard;
- TM1637 Display 4-digits;
- Buzzer;
- Arduino Uno(usado para testes);
- Arduino nano;
- fita de 10 LED's;
- Potenciômetro;
- 4 botões.

Bluetooth

O Módulo Bluetooth HC-06 permite que um dispositivo possa comunicar com outro dispositivo através de rede sem fios, usando o Bluetooth. Tem um alcance de 10 metros, para o envio de informação. Tem uma conexão padrão de UART de 2 pinos (Tx e Rx) que o torna muito fácil de conectar à sua placa microcontroladora.



Figura – Bluetooth HC-06

RTC DS1307/3231

É um relógio de tempo real onde armazena na sua pilha, funcionando mais ou menos como a pilha da motherboard.

Para além disto o DS3231 tem um sensor de temperatura incorporado e caso ocorra uma falha de energia ele aciona automaticamente a bateria que acompanha o módulo para evitar perda de dados.



Figura – RTC DS3231

TM1637 Display 4-digits

O módulo TM1637 é um display de 7 segmentos e 4 dígitos, ideal para projetos envolvendo relógios, contadores e outros equipamentos que necessitem de um display de fácil visualização.



Figura – TM1637

LED's WS2811

Estes LED's são LED's endereçáveis compatíveis com o arduino para os programar, podem ser usados para fazer tela de led, parede de led, quadro de publicidade e amplamente aplicado a hotéis, KTV, bares, letreiros de publicidade externa, iluminação de festas de casamento, decoração de carros e

casas, horizonte de cidade, contorno de edifícios e assim por diante.



Figura – LED's WS2811

Arduino Nano

O Arduino Nano é a menor placa existente para construir projetos.

Possui mais ou menos a mesma funcionalidade do Arduino Duemilanove, mas em um pacote diferente.

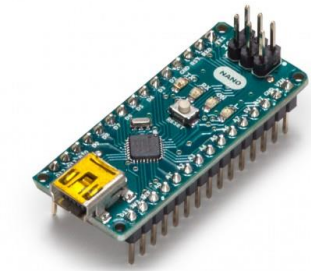


Figura – Arduino Nano

Potenciômetro

O potenciômetro é um componente para fazer o controle de diversas funcionalidades no arduino, no meu caso eu irei usar-lo para o controle dos LED's e o seu brilho.



Figura - Potenciômetro

Software Usado

Neste projeto usei 4 softwares diferentes: Fritzing, Arduino IDE, Tinkercad e o AndroidStudio.

Arduino IDE

Para a programação do arduino utilizei o Arduino IDE, desenvolvido pela mesma empresa que criou o Arduino.

Neste software usamos C++, e através deste software é possível compilar e enviar o código para qualquer placa que seja compatível com o Arduino.

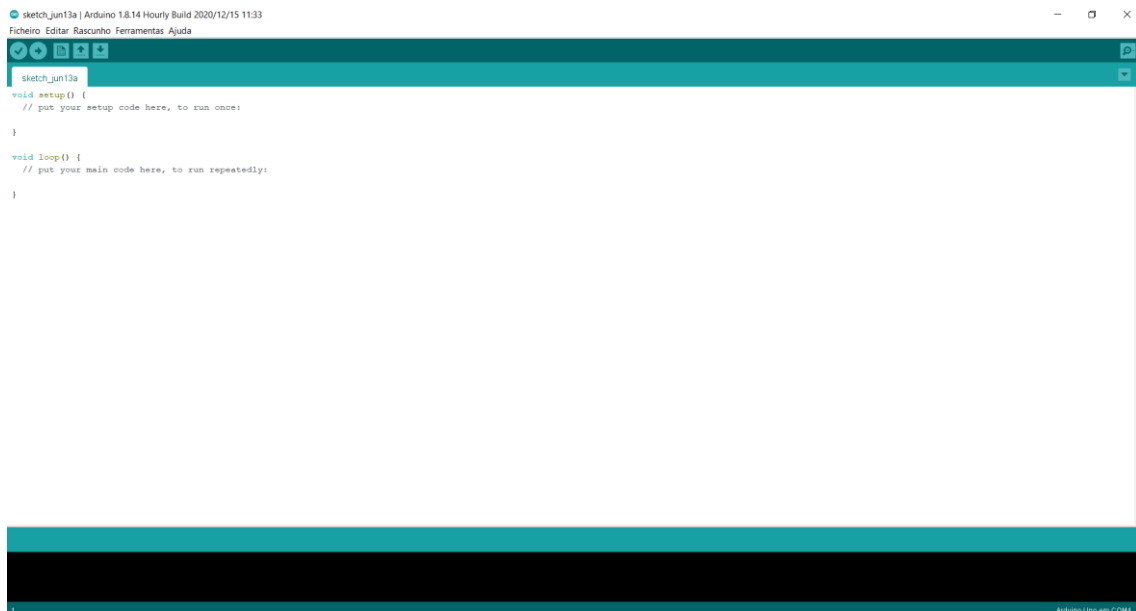


Figura – Arduino IDE

Libraries usadas

Para este projeto usei algumas libraries para auxílio do funcionamento de componentes que usei.

Para o controle dos LED's usei a FastLED; para a utilização e configuração do RTC usei a RTCLib e para o controle do TM1637 usei a TM1637Display.

O SoftwareSerial serve para simular qualquer um dos pinos digitais da placa que eu quiser como se fossem pinos de serial que são usados para comunicação serial; já que na placa só existem 2 pinos serial o pino 0 e 1 que são usados para fins de depuração.



Figura - LibrariesAndroid

C++

C++ é uma linguagem de programação compilada multi-paradigma, usada por este software. Esta linguagem de programação para além da programação orientada a objetos, serve para tudo no geral.

Android Studio

Este software é usado para desenvolver aplicações/jogos no Android.

É um software bastante prático de se usar e bastante organizado.

Este software usa as linguagens Java e Kotlin.

Um aspeto bastante prático deste software é o facto de ele nos fornecer um emulador de telemóvel, sem termos que ligar toda a hora o telemóvel ao computador.

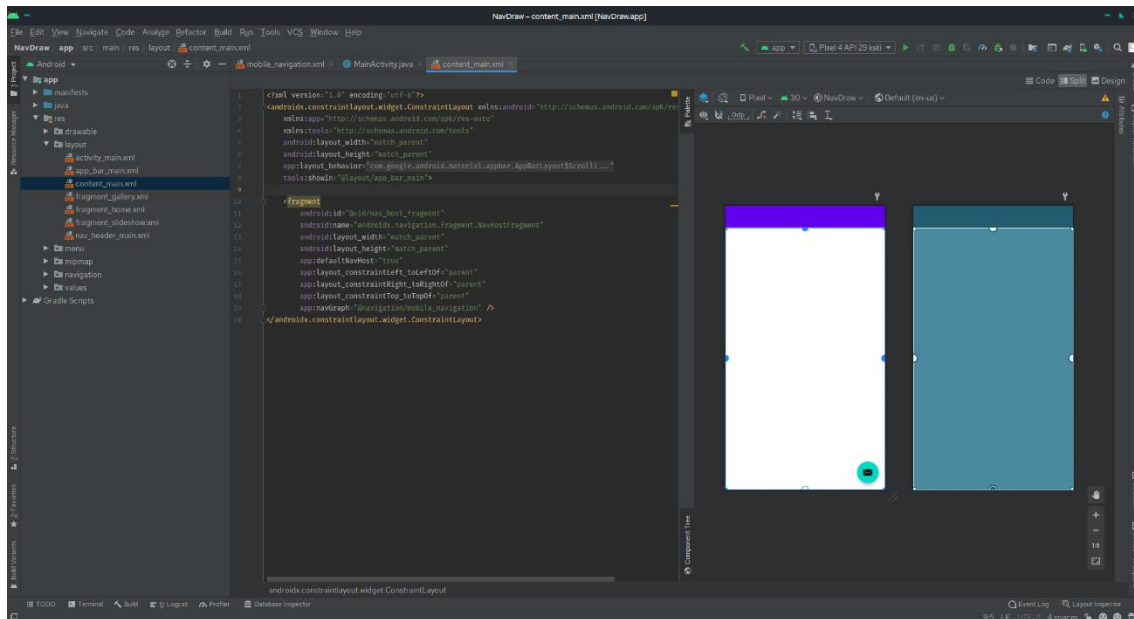


Figura - AndroidStudio

Java

Java é uma linguagem de programação orientada a objetos bastante popular neste Software é bastante usada também para o desenvolvimento de jogos, calculadoras e rádios.

Diferente de outras linguagens, Java é compilada para um bytecode que é interpretado por uma máquina virtual.

Há uma facilidade de internacionalização, suportando nativamente caracteres Unicode.

Tinkercad

Tinkercad é um software bastante prático para o desenho de objetos 3D e para o desenho de circuitos, desenvolvido pela Autodesk.

Este software é um software grátis que roda num browser qualquer.

Com este software pôde desenhar toda a estrutura do meu projeto.

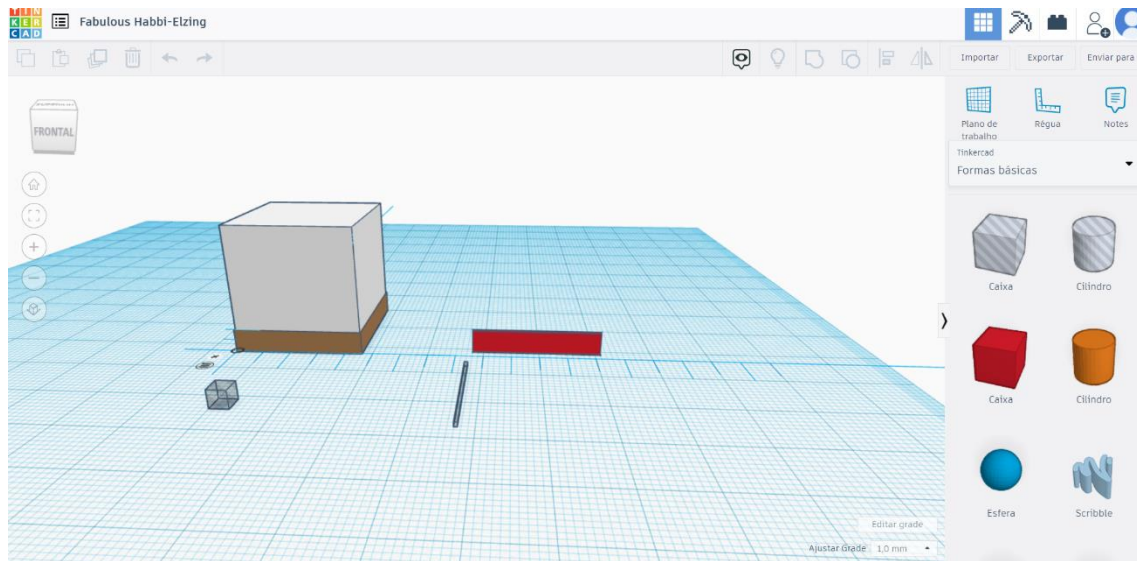


Figura – Tinkercad

Fritzing

Fritzing é um software desenvolvido para design de Hardware eletrônico, foi desenvolvido na Universidade de Ciências Aplicadas de Potsdam.

Para o desenho do meu circuito para testes utilizei este software bastante prático e fácil de usar, este software é bastante parecido com o tinkercad mas

no entanto tem mais recursos.

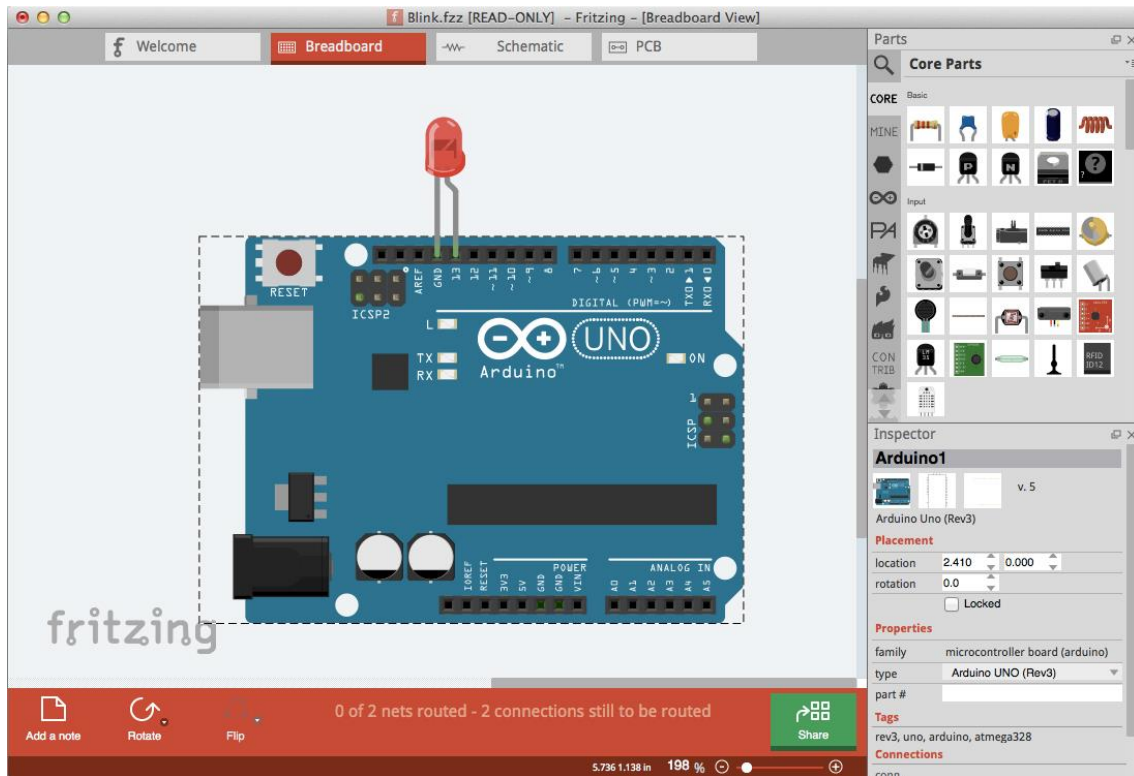


Figura – Fritzing

Desenvolvimento

Etapa 1 – Levantamento de ideias e material

Primeiramente pensei em todas as funcionalidades que queria para o meu projeto, feito isso fiz uma lista de tudo o que iria precisar desde todo o material eletrônico necessário até aos softwares.

Enquanto esperava pelo material tentei aprender, como funcionavam os circuitos nas placas breadboard.

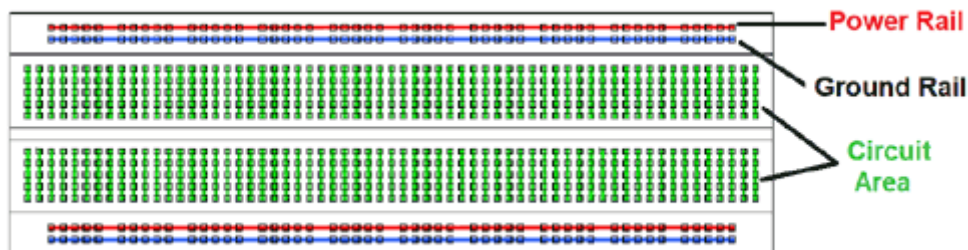


Figura – BreadBoardExplicação

Também desenhei um circuito para testes do meu projeto com auxilio do Fritzing.

Nota: O potenciômetro não tem pinos específicos para o GND e o VCC.

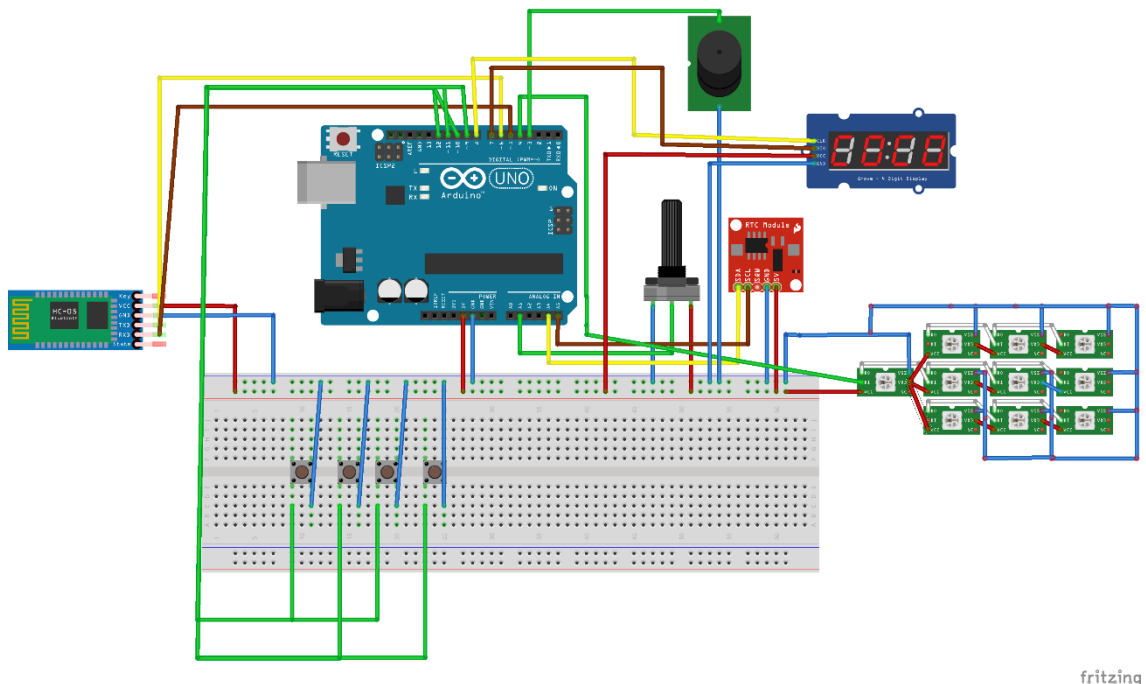


Figura – Circuito Testes

Etapa 2 – Programação do Relógio e Alarme

Assim que recebi o material, primeiramente verifiquei através dos exemplos das bibliotecas que o arduino nos disponibiliza se o material estava a funcionar corretamente.

Verificado tudo, configurei o RTC com a ajuda de um exemplo da RTCLib.

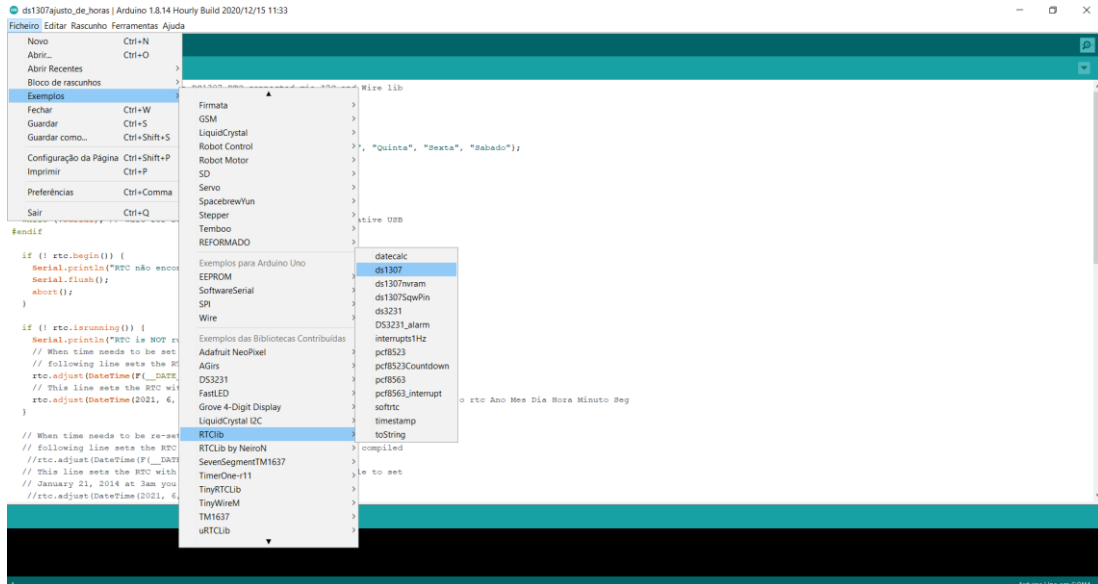
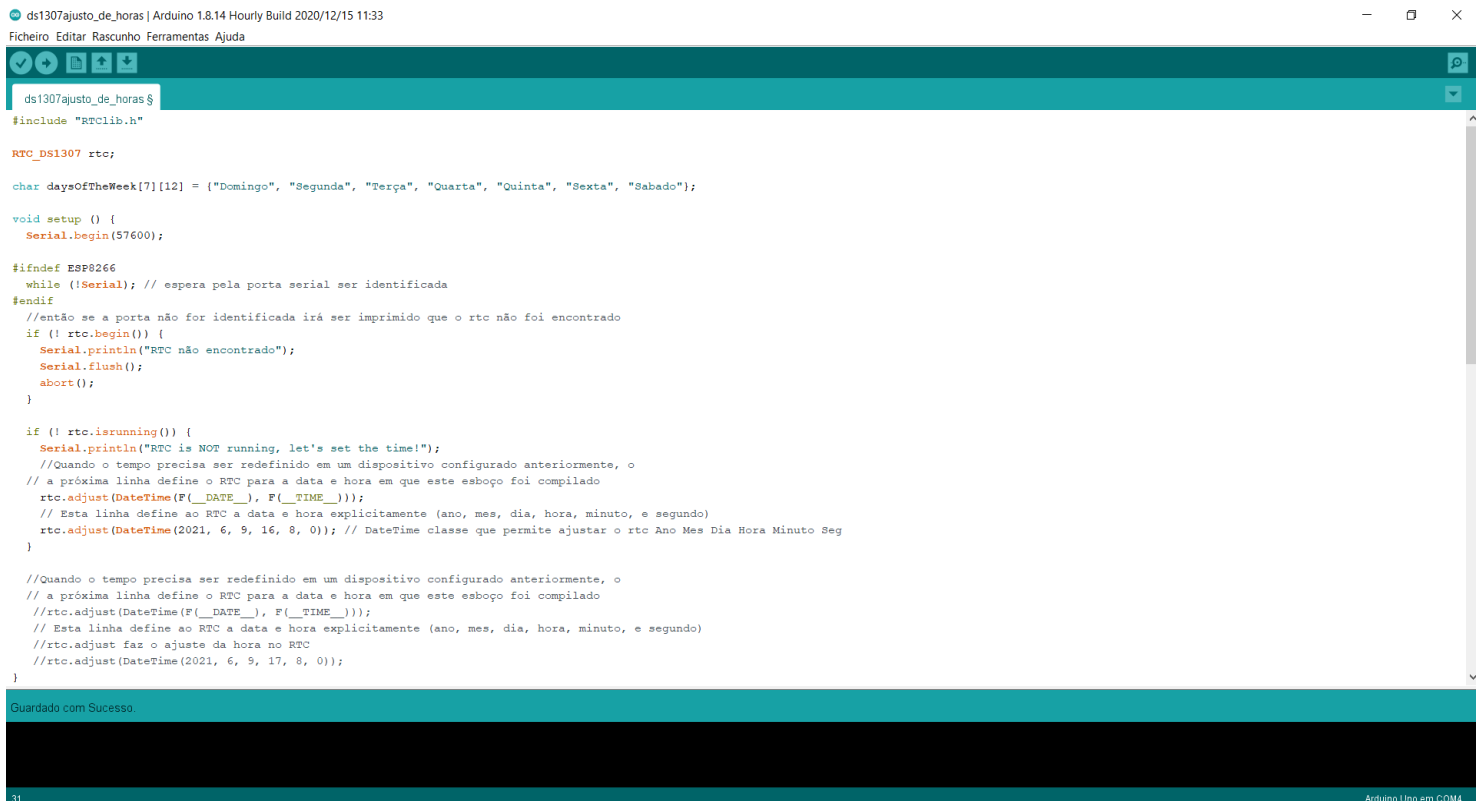


Figura – ExemplosRTC

Sempre que quiser ajustar o RTC basta apenas tirar os comentarios abaixo de comando e definir as horas como quiser, depois só mandar para o RTC e ele irá guardar.



```

ds1307ajusto_de_horas | Arduino 1.8.14 Hourly Build 2020/12/15 11:33
Ficheiro Editar Rascunho Ferramentas Ajuda

#include "RTCLib.h"

RTC_DS1307 rtc;

char daysOfTheWeek[7][12] = {"Domingo", "Segunda", "Terça", "Quarta", "Quinta", "Sexta", "Sabado"};

void setup () {
  Serial.begin(57600);

#ifdef ESP8266
  while (!Serial); // espera pela porta serial ser identificada
#endif
  //então se a porta não for identificada irá ser imprimido que o rtc não foi encontrado
  if (!rtc.begin()) {
    Serial.println("RTC não encontrado");
    Serial.flush();
    abort();
  }

  if (!rtc.isrunning()) {
    Serial.println("RTC is NOT running, let's set the time!");
    //Quando o tempo precisa ser redefinido em um dispositivo configurado anteriormente, o
    // a próxima linha define o RTC para a data e hora em que este esboço foi compilado
    rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
    // Esta linha define ao RTC a data e hora explicitamente (ano, mes, dia, hora, minuto, e segundo)
    rtc.adjust(DateTime(2021, 6, 9, 16, 8, 0)); // DateTime classe que permite ajustar o rtc Ano Mes Dia Hora Minuto Seg
  }

  //Quando o tempo precisa ser redefinido em um dispositivo configurado anteriormente, o
  // a próxima linha define o RTC para a data e hora em que este esboço foi compilado
  //rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
  // Esta linha define ao RTC a data e hora explicitamente (ano, mes, dia, hora, minuto, e segundo)
  //rtc.adjust faz o ajuste da hora no RTC
  //rtc.adjust(DateTime(2021, 6, 9, 17, 8, 0));
}

Guardado com Sucesso.
31 Arduino Uno em COM4

```

Figura – Configuração do RTC

De seguida tentei ligar o TM1637 com as horas que tinha definido no RTC.

Como tenho 3 ecrãs para 3 funções diferentes Relógio, Alarme e ajuste de Horas, sempre que apenas eu chamar a função Relógio, as outras serão consideradas falsas e apresentará as horas atuais no Display, e para outros ecrãs do mesmo método.

As funções now.hour e now.minute vão buscar as horas atuais ao rtc e enviam essa informação para o Display para serem apresentadas.

Nota: a classe now sempre que for chamada com a library RTCLib ela vai buscar as horas atuais.

Antes de definirmos as posições usamos uint8_t position[] = { 0, 0, 0, 0 }; para resetar e criar o array position[] que armazena os valores pretendidos.

Position 0 corresponde à primeira linha do display, este display é dividido por 4 dígitos e as “positions”, correspondem aos 4 dígitos.

O código position[1] = position[1] + 128; apenas serve para meter aquele separador “ : ” entre as horas e minutos.

Feito isso configurando a ordem que queremos, chamamos `display.setSegments(positions);` para que possa organizar as informações que enviamos para o display.

```

experimento
void display_current_time(DateTime now, int alarm_hour, int alarm_minute) {

    //////////////// Horas Atuais:
    if (!set_alarm && !set_hour_minute_flag) {
        if (!set_alarm) {
            display.setBrightness(7);
            uint8_t position[] = { 0, 0, 0, 0 };
            position[0] = display.encodeDigit(now.hour() / 10); // divide hora por 10 e coloca a parte inteira do resultado no primeiro display
            position[1] = display.encodeDigit(now.hour() % 10); // divide hora por 10 e coloca o resto da divisao no segundo display
            position[2] = display.encodeDigit(now.minute() % 60 / 10); // divide minuto por 10 e coloca a parte inteira do resultado no terceiro display
            position[3] = display.encodeDigit(now.minute() % 60 % 10); // divide minuto por 10 e coloca o resto da divisao no quarto display
            position[1] = position[1] + 128; // Coloca o separador ":"
            display.setSegments(position); //organiza/atualiza as informações no display
        }
    }

    //////////////// Mostrar alarme

    if (set_alarm) {

        if (millis() > time_last_showed && millis() < time_last_showed + 500) {

            // brilho
            display.setBrightness(7);

            // Zera e cria o array position[] para armazenar os valores do alarme
            uint8_t position[] = { 0, 0, 0, 0 };
            // Calculo individual para posição no display
            position[0] = display.encodeDigit(alarm_hour / 10); // divide hora por 10 e coloca a parte inteira do resultado no primeiro display
            position[1] = display.encodeDigit(alarm_hour % 10); // divide hora por 10 e coloca o resto da divisao no segundo display
            position[2] = display.encodeDigit((alarm_minute % 60) / 10); // divide minuto por 10 e coloca a parte inteira do resultado no terceiro display
            position[3] = display.encodeDigit((alarm_minute % 60) % 10); // divide minuto por 10 e coloca o resto da divisao no quarto display
            position[1] = position[1] + 128; // Coloca o separador ":"
        }
    }
}

```

Figura – Código do Display

```

experimento $
        position[1] = position[1] + 128; // Coloca o separador ":"
        display.setSegments(position); //organiza/atualiza as informações no display
    }

    if (millis() >= time_last_showed + 500) {
        display.clear();
        time_last_showed = millis() + 250;
    }
}

// Novas horas e minutos
if (set_hour_minute_flag) {

    if (millis() > time_last_showed && millis() < time_last_showed + 500) {

        // Set brightness
        display.setBrightness(7);

        // Zera e cria o array position[] para armazenar os novos valores das horas
        uint8_t position[] = { 0, 0, 0, 0 };
        // Calculo individual para posição no display
        position[0] = display.encodeDigit(new_hour / 10); // divide hora por 10 e coloca a parte inteira do resultado no primeiro display
        position[1] = display.encodeDigit(new_hour % 10); // divide hora por 10 e coloca o resto da divisao no segundo display
        position[2] = display.encodeDigit((new_minute % 60) / 10); // divide minuto por 10 e coloca a parte inteira do resultado no terceiro display
        position[3] = display.encodeDigit((new_minute % 60) % 10); // divide minuto por 10 e coloca o resto da divisao no quarto display
        // Coloca o separador ":"
        position[1] = position[1] + 128;
        display.setSegments(position); //organiza/atualiza as informações no display
    }

    if (millis() >= time_last_showed + 500) {
        display.clear();
        time_last_showed = millis() + 250;
    }
}

```

Figura – Código do Display

Para definir as horas novas criei uma função void `get_current_time_once()`, onde ele define `new_minute` igual ao minuto atual e `new_hour` igual à hora atual.

Quando o utilizador usar o botão up, ele irá adicionar 1 valor à new_hour, que por sua vez irá atualizar de seguida os valores do relógio.

Quando set_hour_minute for definido ele irá fazer essa atualização usando a função rtc.adjust(...,new_hour, new-minute,...) para que possa armazenar os novos dados diretamente no rtc.

```
void get_current_time_once() {
    DateTime now_to_change = rtc.now();

    new_minute = now_to_change.minute();
    new_hour = now_to_change.hour();
}
```

Figura – Código da atualização das horas no rtc

```
// Alterar o valor das horas no relógio
if (set_hour_minute_flag) {

    new_hour = new_hour + 1;

    if (new_hour >= 24) {
        new_hour = 0;
    }
}
```

Figura – Código da atualização das horas no rtc

```
//alterar hora e minutos do rtc
void set_hour_minute() {

    rtc.adjust(DateTime(now.year(), now.month(), now.day(), new_hour, new_minute, now.second())); // (ano), (mes), (dia), (hora), (minutos), (segundos)
}

void update_my_time_now() {
    DateTime now = rtc.now();
}
```

Figura – Código da atualização das horas no rtc

Após ter conseguido mudar as horas usando os botões então comecei a programar o alarme.

O alarme diferentemente das horas não é gravado no RTC, mas sim na EEPROM um chip incorporado na placa arduino.

Primeiramente criei 2 variáveis alarm_time_hour e alarm_time_minute para que estas sejam lidas e enviem os seus valores para a EEPROM.

```
// Definir alarme:
int alarm_time_hour;
int alarm_time_hour_address = 0;
int alarm_time_minute;
int alarm_time_minute_address = 1;
```



```

////////////////////////////////////////// alarme

alarm_time_hour = EEPROM.read(alarm_time_hour_address); //EEPROM(chip incorporado não no rtc mas no arduino) irá ler o valor que iremos dar ao alarme para depois armazenar
alarm_time_minute = EEPROM.read(alarm_time_minute_address);

}
////////////////////////////////////////// RTC

```

Figura – Alarme

Programar o alarme não foi muito diferente do ajuste de horas já que o raciocínio é idêntico, a diferença é que o alarme é armazenado na EEPROM.

Então para o ajuste do alarme quando o ajuste do alarme for chamado “set_alarm” sempre que o utilizador clicar no botão up ele irá adicionar um valor às horas.

Nota: sempre que os valores das horas passarem de 24 ele volta ao número 0 e nos minutos sempre que passar dos 60 minutos ele retorna ao número 0.

Esse EEPROM.write irá ler os valores

```

void button_up_update_state() {

    if (digitalRead(button_up) == LOW) { // Se o botão up for pressionado

        button_up_startPressed = millis();
        button_up_idleTime = button_up_startPressed - button_up_endPressed;

        //Mudar o valor das horas no alarme até 24
        if (set_alarm) {

            alarm_time_hour = alarm_time_hour + 1;

            EEPROM.write(alarm_time_hour_address, alarm_time_hour);

            if (alarm_time_hour >= 24) {
                alarm_time_hour = 0;
                EEPROM.write(alarm_time_hour_address, alarm_time_hour);
            }

        }

    }
}

```

Após isso pensei em criar uma funcionalidade podia ligar e desligar o alarme sempre que quisesse sem ter que ajustar sempre ou tirar o alarme de um “loop infinito”.

Este alarm_on só irá ser chamado quando o utilizador pressionar no botão clock durante 3 segundos ou mais e se o utilizador pretender desliga-lo basta pressionar de novo durante 3 segundos.

Se o alarme_on for ligado então quando as horas definidas pelo alarme se coincidirem com as horas do RTC (as horas atuais), então a função play_alarm() começa.

Esta que por sua vez irá usar um Buzzer que irá servir como apito.

```

////////////////////////////////////verificação de alarme

if (alarm_on) {

    if (sunshine_check(now, alarm_time_hour, alarm_time_minute)) {
        luzes_on = true;
        luzes_mode = 100;
        clock_on = true;
    }
    else {

        time_begin_sunshine = millis();
        time_end_sunshine = time_begin_sunshine + (240000);
        time_end_alarm = time_begin_sunshine + (900000);

    }

    // Verifica se o alarme coincide com as horas do rtc (now)
    if (wake_up_check(now, alarm_time_hour, alarm_time_minute)) {

        set_alarm = false;
        sleeping = false;
        play_alarm(0);

    }

}

}

```

Figura - Alarme

experimento

```

void play_alarm() {

    if (millis() > time_last_played) {
        tone(pinBuzzer, 600, 350);
        time_last_played = millis() + beep_delay;
        times_beeped += 1;
        times_beeped_absolute += 1;
    }

    if (times_beeped >= 15) {

        if (beep_delay >= 600) {
            beep_delay -= 500;
        }
        else {
            beep_delay = 400;
        }

        times_beeped = 0;
    }

    if (millis() >= time_end_alarm) { //Desligar alarm
    }

    alarm_on = false;
    times_beeped = 0;
    times_beeped_absolute = 0;
    beep_delay = 2000;
}

```

Figura – Alarme

```

    ✓ ↻ 📄 ⬆️ ⬇️
    experimento $
    bool sunshine_check(DateTime now, int alarm_time_hour, int alarm_time_minute) {

        // Compara horas e minutos da hora definida para o alarme com a hora atual

        String compare_alarm = "";

        if (alarm_time_hour == 0) {
            compare_alarm += "00";
        }
        else {
            compare_alarm += alarm_time_hour;
        }

        if (alarm_time_minute <= 9) {
            compare_alarm += "0";
            if (alarm_time_minute == 0) {
                compare_alarm += "0";
            }
            else {
                compare_alarm += alarm_time_minute;
            }
        }
        else {
            compare_alarm += alarm_time_minute;
        }

        String compare_now = " ";

        if (now.hour() == 0) {
            compare_now += "00";
        }
        else {
            compare_now += now.hour();
        }

        if (now.minute() <= 9) {

```

Figura - Alarme

```

    experimento $
    if (now.hour() == 0) {
        compare_now += "00";
    }
    else {
        compare_now += now.hour();
    }

    if (now.minute() <= 9) {
        compare_now += "0";
        if (now.minute() == 0) {
            compare_now += "0";
        }
        else {
            compare_now += now.minute();
        }
    }
    else {
        compare_now += now.minute();
    }

    int compare_alarm_int = compare_alarm.toInt();
    int compare_now_int = compare_now.toInt();

    if ((compare_alarm_int - compare_now_int) <= 5 && (compare_alarm_int - compare_now_int) >= 0) {

        return true;
    }
    else {
        return false;
    }
}

bool wake_up_check(DateTime now, int alarm_time_hour, int alarm_time_minute) {

    // Compara horas e minutos da hora definida para o alarme com a hora atual

```

Figura - Alarme

```
bool wake_up_check(DateTime now, int alarm_time_hour, int alarm_time_minute) {

    // Compara horas e minutos da hora definida para o alarme com a hora atual

    if (alarm_time_hour == now.hour() && alarm_time_minute == now.minute()) {
        return true;
    }
    else {
        return false;
    }

}
```

Figura – Alarme

Etapa 3 – Programação dos LED's

Para o meu projeto criei 6 modos de efeitos diferentes: rainbow; chama (cores mais quentes); moon (cores mais escuras); color pick (cor à escolha usando o potenciômetro); uma sequência de cores mais relaxantes e outra mais movimentada.

Primeira coisa foi fazer a configuração dos LED's neste caso LED's WS2811, com 10 LED's, até um limite de 255 de brilho.

```
//////////////////////////////////// LED WS2811
// Configuração dos LEDs
FastLED.addLeds<LED_TYPE, DATA_PIN, COLOR_ORDER>(leds, NUM_LEDS).setCorrection(TypicalLEDStrip).setDither(BRIGHTNESS < 255);
FastLED.setBrightness(BRIGHTNESS); // controle do brilho

// Potenciômetro
pinMode(A1, INPUT); // pino de entrada
```

Figura - Efeitos

```
//////////////////////////////////// Modos de luz efeitos
////////////////////////////////////

byte cor_sunshine_R = 0xff;
byte cor_sunshine_G = 0xbf;
byte cor_sunshine_B = 0x00;

void sunshine() {

    float r, g, b;

    unsigned long fator = map(millis(), time_begin_sunshine, time_end_sunshine, 0, 1000);

    if (fator > 1000) {
        fator = 1000;
    }

    r = cor_sunshine_R * fator / 1000;
    g = cor_sunshine_G * fator / 1000;
    b = cor_sunshine_B * fator / 1000;
    setAll(r, g, b);
    showStrip();
}
```

Figura - Efeitos

```

exprimento$
//modo personalizado/my color
void myColor(int pot_val) {

    CRGB color1 = CHSV(map(pot_val, min_pot_val, max_pot_val, 0, 255), 255, 255);
    //o CRGB irá representar o verde/vermelho/verde
    //o CHSV serve para alterar/configurar o brilho

    fadeTowardColor(leds, NUM_LEDS, color1, 20);
    FastLED.show();

}

bool comecar = true;
bool respirar = false;
bool segurar = false;
bool expirar = false;
bool segurar2 = false;

unsigned long time_begin;
unsigned long time_to_hold;
unsigned long time_to_let_go;
unsigned long time_to_let_go2;
unsigned long time_to_begin_again;

byte cor_respiro_R = 0xff;
byte cor_respiro_G = 0x55;
//byte cor_respiro_B = 0x16;
byte cor_respiro_B = 0x00;

```

Figura – Efeitos

```

exprimento
//modo relaxante
void Respiro() {

    //cores com uma sintonia mais longa

    float r, g, b;
    //definir variaveis para depois conseguir fazer a tal de sintonia
    //assim que iniciar irá começar a função respirar assim que passar o tempo então as cores desaparecem e vai para inspirar
    if (comecar) {
        time_begin = millis();
        time_to_hold = millis() + 4000;
        time_to_let_go = time_to_hold + 3000;
        time_to_let_go2 = time_to_let_go + 300;
        comecar = false;
        respirar = true;
    }

    if (respirar) {

        //Inspira
        //o map que vai ajudar a planificar um número de um intervalo para outro
        unsigned long fator = map(millis(), time_begin, time_to_hold, 0, 1000);

        r = cor_respiro_R * fator / 1000;
        g = cor_respiro_G * fator / 1000;
        b = cor_respiro_B * fator / 1000;
        setAll(r, g, b);
        showStrip();

        if (millis() >= time_to_hold) {

            respirar = false;

        }
    }
}

```

Figura – Efeitos

```

if (millis() >= time_to_hold) {
    //time_to_let_go = millis() + 80000;

    CRGB bgColor1(0, 100, 255);
    fadeTowardColor(leds, NUM_LEDS, bgColor1, 20);
    FastLED.show();
}

if (millis() >= time_to_let_go) {
    time_to_hold = millis() + 80000;

    CRGB bgColor2(255, 85, 0);
    fadeTowardColor(leds, NUM_LEDS, bgColor2, 20);
    FastLED.show();

}

if (millis() >= time_to_let_go2) {
    time_to_let_go = millis() + 80000;
    time_to_let_go2 = millis() + 80000;
    time_begin = millis();
    time_to_hold = millis() + 2500;
    expirar = true;

}

if (expirar) {
    // Expira

    unsigned long fator = map(millis(), time_begin, time_to_hold, 1000, 0);

    r = (cor_respiro_R * fator / 1200);
    g = (cor_respiro_G * fator / 1200);
    b = (cor_respiro_B * fator / 1200);
}

```

Figura - Efeitos

```

exprimto
if (expirar) {
    // Expira

    unsigned long fator = map(millis(), time_begin, time_to_hold, 1000, 0);

    r = (cor_respiro_R * fator / 1200);
    g = (cor_respiro_G * fator / 1200);
    b = (cor_respiro_B * fator / 1200);
    setAll(r, g, b);
    showStrip();

    if (millis() >= time_to_hold) {
        /*for (int i = 0; i < NUM_LEDS; i++) {
            leds[i] = CRGB::Black;
        }
        FastLED.show();*/
        time_to_begin_again = millis() + 3300;
        time_to_hold = millis() + 80000;
        expirar = false;
        segurar2 = true;
    }
}

if (segurar2) {
    for (int i = 0; i < NUM_LEDS; i++) {
        leds[i] = CRGB::Black;
    }
    FastLED.show();

    if (millis() >= time_to_begin_again) {
        comecar = true;
        segurar2 = false;
    }
}
}

```

Figura - Efeitos

```

|
void FadeInOut(byte red, byte green, byte blue, int speed) {
    float r, g, b;

    for (int k = 0; k < (256 * speed); k = k + 1) {
        r = (k / (256.0 * speed)) * red;
        g = (k / (256.0 * speed)) * green;
        b = (k / (256.0 * speed)) * blue;
        setAll(r, g, b);
        showStrip();
        if (check_if_buttons_were_pressed()) {
            break;
        }
    }


    for (int k = (255 * speed); k >= 0; k = k - 2) {
        r = (k / (256.0 * speed)) * red;
        g = (k / (256.0 * speed)) * green;
        b = (k / (256.0 * speed)) * blue;
        setAll(r, g, b);
        showStrip();
        if (check_if_buttons_were_pressed()) {
            break;
        }
    }
}

void rainbowCycle(int SpeedDelay) {
    byte* c;
    uint16_t i, j;

    for (j = 0; j < 256; j++) { // 5 cycles of all colors on wheel
        for (i = 0; i < NUM_LEDS; i++) {

```

Figura - Efeitos



```

void setPixel(int Pixel, byte red, byte green, byte blue) {

    // FastLED
    leds[Pixel].r = red;
    leds[Pixel].g = green;
    leds[Pixel].b = blue;

}

void setAll(byte red, byte green, byte blue) {
    for (int i = 0; i < NUM_LEDS; i++) {
        setPixel(i, red, green, blue);
    }
    showStrip();
}

void nblendU8TowardU8(uint8_t& cur, const uint8_t target, uint8_t amount)
{
    if (cur == target) return;

    if (cur < target) {
        uint8_t delta = target - cur;
        delta = scale8_video(delta, amount);
        cur += delta;
    }
    else {
        uint8_t delta = cur - target;
        delta = scale8_video(delta, amount);
        cur -= delta;
    }
}

// mistura uma cor CRGB com outra em determinada quantidade
// linear e com espaço de cores
CRGB fadeTowardColor(CRGB& cur, const CRGB& target, uint8_t amount)

```

Figura - Efeitos

```
// mistura uma cor CRGB com outra em determinada quantidade
// linear e com espaço de cores
CRGB fadeTowardColor(CRGB& cur, const CRGB& target, uint8_t amount)
{
    nblendU8TowardU8(cur.red, target.red, amount);
    nblendU8TowardU8(cur.green, target.green, amount);
    nblendU8TowardU8(cur.blue, target.blue, amount);
    return cur;
}

// Modifica a matriz dos pixels no local
void fadeTowardColor(CRGB* L, uint16_t N, const CRGB& bgColor, uint8_t fadeAmount)
{
    for (uint16_t i = 0; i < N; i++) {
        fadeTowardColor(L[i], bgColor, fadeAmount);
    }
}

CRGB color = CHSV(random8(), 255, 255);

unsigned long next_color = 0;
|
void jujuba(int pot_val) {

    if (millis() >= next_color) {
        color = CHSV(random8(), 255, 255);
        next_color = millis() + map(pot_val, min_pot_val, max_pot_val, 1000, 15000)
    }

    // todos os pixels
    fadeTowardColor(leds, NUM_LEDS, color, 1);
    FastLED.show();
}
```

Figura - Efeitos

```
    else {
        uint8_t delta = cur - target;
        delta = scale8_video(delta, amount);
        cur -= delta;
    }
}

// mistura uma cor CRGB com outra em determinada quantidade
// linear e com espaço de cores
CRGB fadeTowardColor(CRGB& cur, const CRGB& target, uint8_t amount)
{
    nblendU8TowardU8(cur.red, target.red, amount);
    nblendU8TowardU8(cur.green, target.green, amount);
    nblendU8TowardU8(cur.blue, target.blue, amount);
    return cur;
}

// Modifica a matriz dos pixels no local
void fadeTowardColor(CRGB* L, uint16_t N, const CRGB& bgColor, uint8_t fadeAmount)
{
    for (uint16_t i = 0; i < N; i++) {
        fadeTowardColor(L[i], bgColor, fadeAmount);
    }
}

CRGB color = CHSV(random8(), 255, 255);

unsigned long next_color = 0;
//modo fade
void jujuba(int pot_val) {

    if (millis() >= next_color) {
        color = CHSV(random8(), 255, 255);
        next_color = millis() + map(pot_val, min_pot_val, max_pot_val, 1000, 15000);
    }
}
```

Figura - Efeitos


```

void rainbowCycle(int SpeedDelay) {
    byte* c;
    uint16_t i, j;

    for (j = 0; j < 256; j++) { // 5 ciclos de todas as cores a percorrer os
        for (i = 0; i < NUM_LEDS; i++) {
            c = Wheel(((i * 256 / NUM_LEDS) + j) & 255);
            setPixel(i, *c, *(c + 1), *(c + 2));
            if (check_if_buttons_were_pressed()) {
                break;
            }
        }
        showStrip();

        if (check_if_buttons_were_pressed()) {
            break;
        }

        delay(SpeedDelay);
    }
}

byte* Wheel(byte WheelPos) {
    static byte c[3];

    if (WheelPos < 85) {
        c[0] = WheelPos * 3;
        c[1] = 255 - WheelPos * 3;
        c[2] = 0;
    }
    else if (WheelPos < 170) {

```

Figura - Efeitos

```

Linha 12345678910111213141516171819202122232425262728293031323334353637383940414243444546474849505152535455565758596061626364656667686970717273747576777879808182838485868788899091929394959697989910010110210310410510610710810911011111211311411511611711811912012112212312412512612712812913013113213313413513613713813914014114214314414514614714814915015115215315415515615715815916016116216316416516616716816917017117217317417517617717817918018118218318418518618718818919019119219319419519619719819920020120220320420520620720820921021121221321421521621721821922022122222322422522622722822923023123223323423523623723823924024124224324424524624724824925025125225325425525625725825926026126226326426526626726826927027127227327427527627727827928028128228328428528628728828929029129229329429529629729829930030130230330430530630730830931031131231331431531631731831932032132232332432532632732832933033133233333433533633733833934034134234334434534634734834935035135235335435535635735835936036136236336436536636736836937037137237337437537637737837938038138238338438538638738838939039139239339439539639739839940040140240340440540640740840941041141241341441541641741841942042142242342442542642742842943043143243343443543643743843944044144244344444544644744844945045145245345445545645745845946046146246346446546646746846947047147247347447547647747847948048148248348448548648748848949049149249349449549649749849950050150250350450550650750850951051151251351451551651751851952052152252352452552652752852953053153253353453553653753853954054154254354454554654754854955055155255355455555655755855956056156256356456556656756856957057157257357457557657757857958058158258358458558658758858959059159259359459559659759859960060160260360460560660760860961061161261361461561661761861962062162262362462562662762862963063163263363463563663763863964064164264364464564664764864965065165265365465565665765865966066166266366466566666766866967067167267367467567667767867968068168268368468568668768868969069169269369469569669769869970070170270370470570670770870971071171271371471571671771871972072172272372472572672772872973073173273373473573673773873974074174274374474574674774874975075175275375475575675775875976076176276376476576676776876977077177277377477577677777877978078178278378478578678778878979079179279379479579679779879980080180280380480580680780880981081181281381481581681781881982082182282382482582682782882983083183283383483583683783883984084184284384484584684784884985085185285385485585685785885986086186286386486586686786886987087187287387487587687787887988088188288388488588688788888989089189289389489589689789889990090190290390490590690790890991091191291391491591691791891992092192292392492592692792892993093193293393493593693793893994094194294394494594694794894995095195295395495595695795895996096196296396496596696796896997097197297397497597697797897998098198298398498598698798898999099199299399499599699799899910001001100210031004100510061007100810091010101110121013101410151016101710181019102010211022102310241025102610271028102910301031103210331034103510361037103810391040104110421043104410451046104710481049105010511052105310541055105610571058105910601061106210631064106510661067106810691070107110721073107410751076107710781079108010811082108310841085108610871088108910901091109210931094109510961097109810991100110111021103110411051106110711081109111011111112111311141115111611171118111911201121112211231124112511261127112811291130113111321133113411351136113711381139114011411142114311441145114611471148114911501151115211531154115511561157115811591160116111621163116411651166116711681169117011711172117311741175117611771178117911801181118211831184118511861187118811891190119111921193119411951196119711981199120012011202120312041205120612071208120912101211121212131214121512161217121812191220122112221223122412251226122712281229123012311232123312341235123612371238123912401241124212431244124512461247124812491250125112521253125412551256125712581259126012611262126312641265126612671268126912701271127212731274127512761277127812791280128112821283128412851286128712881289129012911292129312941295129612971298129913001301130213031304130513061307130813091310131113121313131413151316131713181319132013211322132313241325132613271328132913301331133213331334133513361337133813391340134113421343134413451346134713481349135013511352135313541355135613571358135913601361136213631364136513661367136813691370137113721373137413751376137713781379138013811382138313841385138613871388138913901391139213931394139513961397139813991400140114021403140414051406140714081409141014111412141314141415141614171418141914201421142214231424142514261427142814291430143114321433143414351436143714381439144014411442144314441445144614471448144914501451145214531454145514561457145814591460146114621463146414651466146714681469147014711472147314741475147614771478147914801481148214831484148514861487148814891490149114921493149414951496149714981499150015011502150315041505150615071508150915101511151215131514151515161517151815191520152115221523152415251526152715281529153015311532153315341535153615371538153915401541154215431544154515461547154815491550155115521553155415551556155715581559156015611562156315641565156615671568156915701571157215731574157515761577157815791580158115821583158415851586158715881589159015911592159315941595159615971598159916001601160216031604160516061607160816091610161116121613161416151616161716181619162016211622162316241625162616271628162916301631163216331634163516361637163816391640164116421643164416451646164716481649165016511652165316541655165616571658165916601661166216631664166516661667166816691670167116721673167416751676167716781679168016811682168316841685168616871688168916901691169216931694169516961697169816991700170117021703170417051706170717081709171017111712171317141715171617171718171917201721172217231724172517261727172817291730173117321733173417351736173717381739174017411742174317441745174617471748174917501751175217531754175517561757175817591760176117621763176417651766176717681769177017711772177317741775177617771778177917801781178217831784178517861787178817891790179117921793179417951796179717981799180018011802180318041805180618071808180918101811181218131814181518161817181818191820182118221823182418251826182718281829183018311832183318341835183618371838183918401841184218431844184518461847184818491850185118521853185418551856185718581859186018611862186318641865186618671868186918701871187218731874187518761877187818791880188118821883188418851886188718881889189018911892189318941895189618971898189919001901190219031904190519061907190819091910191119121913191419151916191719181919192019211922192319241925192619271928192919301931193219331934193519361937193819391940194119421943194419451946194719481949195019511952195319541955195619571958195919601961196219631964196519661967196819691970197119721973197419751976197719781979198019811982198319841985198619871988198919901991199219931994199519961997199819992000200120022003200420052006200720082009201020112012201320142015201620172018201920202021202220232024202520262027202820292030203120322033203420352036203720382039204020412042204320442045204620472048204920502051205220532054205520562057205820592060206120622063206420652066206720682069207020712072207320742075207620772078207920802081208220832084208520862087208820892090209120922093209420952096209720982099210021012102210321042105210621072108210921102111211221132114211521162117211821192120212121222123212421252126212721282129213021312132213321342135213621372138213921402141214221432144214521462147214821492150215121522153215421552156215721582159216021612162216321642165216621672168216921702171217221732174217521762177217821792180218121822183218421852186218721882189219021912192219321942195219621972198219922002201220222032204220522062207220822092210221122122213221422152216221722182219222022212222222322242225222622272228222922302231223222332234223522362237223822392240224122422243224422452246224722482249225022512252225322542255225622572258225922602261226222632264226522662267226822692270227122722273227422752276227722782279228022812282228322842285228622872288228922902291229222932294229522962297229822992300230123022303230423052306230723082309231023112312231323142315231623172318231923202321232223232324232523262327232823292330233123322333233423352336233723382339234023412342234323442345234623472348234923502351235223532354235523562357235823592360236123622363236423652366236723682369237023712372237323742375237623772378237923802381238223832384238523862387238823892390239123922393239423952396239723982399240024012402240324042405240624072408240924102411241224132414241524162417241824192420242124222423242424252426242724282429243024312432243324342435243624372438243924402441244224432444244524462447244824492450245124522453245424552456245724582459246024612462246324642465246624672468246924702471247224732474247524762477247824792480248124822483248424852486248724882489249024912492249324942495249624972498249925002501250225032504250525062507250825092510251125122513251425152516251725182519252025212522252325242525252625272528252925302531253225332534253525362537253825392540254125422543254425452546254725482549255025512552255325542555255625572558255925602561256225632564256525662567256825692570257125722573257425752576257725782579258025812582258325842585258625872588258925902591259225932594259525962597259825992600260126022603260426052606260726082609261026112612261326142615261626172618261926202621262226232624262526262627262826292630263126322633263426352636263726382639264026412642264326442645264626472648264926502651265226532654265526562657265826592660266126622663266426652666266726682669267026712672267326742675267626772678267926802681268226832684268526862687268826892690269126922693269426952696269726982699270027012702270327042705270627072708270927102711271227132714271527162717271827192720272127222723272427252726272727282729273027312732273327342735273627372738273927402741274227432744274527462747274827492750275127522753275427552756275727582759276027612762276327642765276627672768276927702771277227732774277527762777277827792780278127822783278427852786278727882789279027912792279327942795279627972798279928002801280228032804280528062807280828092810281128122813281428152816281728182819282028212822282328242825282628272828282928302831283228332834283528362837283828392840284128422843284428452846284728482849285028512852285328542855285628572858285928602861286228632864286528662867286828692870287128722873287428752876287728782879288028812882288328842885288628872888288928902891289228932894289528962897289828992900290129022903290429052906290729082909291029112912291329142915291629172918291929202921292229232924292529262927292829292930293129322933293429352936293729382939294029412942294329442945294629472948294929502951295229532954295529562957295829592960296129622963296429652966296729682969297029712972297329742975297629772978297929802981298229832984298529862987298829892990299129922993299429952996299729982999300030013002300330043005300630073008300930103011301230133014301530163017301830193020302130223023302430253026302730283029303030313032303330343035303630373038303930403041304230433044304530463047304830493050305130523053305430553056305730583059306030613062306330643065306630673068306930703071307230733074307530763077307830793080308130823083308430853086308730883089309030913092309330943095309630973098309931003101310231033104310531063107310831093110311131123113311431153116311731183119312031213122312331243125312631273128312931303131313231333134313531363137313831393140314131423143314431453146314731483149315031513152315331543155315631573158315931603161316231633164316531663167316831693170317131723173317431753176317731783179318031813182318331843185318631873188318931903191319231933194319531963197319831993200320132023203320432053206320732083209321032113212321332143215321632173218321932203221322232233223432253226322732283229323032313232323332343235323632373238323932403241324232433244324532463247324832493250325132523253325432553256325732583259326032613262326332643265326632673268326932703271327232733274327532763277327832793280328132823283328432853286328732883289329032913292329332943295329632973298329933003301330233033304330533063307330833093310331133123313331433153316331733183319332033213322332333243325332633273328332933303331333233333334333533363337333833393340334133423343334433453346334733483349335033513352335333
```

```
//modo chama
void fireplace(int SPARKING, int COOLING) {

    gPal = CRGBPalette16(CRGB::Black, CRGB::Red, CRGB::OrangeRed, CRGB::Yellow);

    // crgb:black vai emitindo cores mais escuras pretas
    // ciclo vai do preto ate ao amarelo parecendo um efeito de fogueira
    //gPal = HeatColors_p;

    // gPal = CRGBPalette16( CRGB::Black, CRGB::Red, CRGB::Yellow, CRGB::White

    // gPal = CRGBPalette16( CRGB::Black, CRGB::Blue, CRGB::Aqua, CRGB::White

    // gPal = CRGBPalette16( CRGB::Black, CRGB::Red, CRGB::White);

    // numeros random
    random16_add_entropy(random());

    // Array para ler temperatura
    static byte heat[NUM_LEDS];

    //reduzir luz
    for (int i = 0; i < NUM_LEDS; i++) {
        heat[i] = qsub8(heat[i], random8(0, ((COOLING * 10) / NUM_LEDS) + 2));
    }
}
```

Figura – Efeitos

Etapa 4 – Botões

Para o uso manual deste projeto irei usar botões para que consiga controlar o candeeiro, o alarme e o relógio.

Irei dar o nome ao 4 botões, o button_clock(usado para o menu do alarme e ajuste de horas), button_luzes(usado para ligar e desligar os LED's) e os button up e down que servirão para a funções seguinte e anterior, e controlar as horas e minutos.

```

-
// Botões
// Botão das horas
const int safe_button_press = 200;

bool read_button_clock = true;

const byte button_clock = 12; // pino 12
byte button_clock_state = 0;
unsigned long button_clock_last_state = 0; // momento que foi pressionado
unsigned long button_clock_startPressed = 0; // momento que foi solto
unsigned long button_clock_endPressed = 0; // quanto tempo ficou pressionado
unsigned long button_clock_holdTime = 0; // quanto tempo o botao teve inativo
//unsigned long time_before_play_alarm = 0;
unsigned long time_last_played;

int beep_delay = 2000;

int times_beeped = 0;
int times_beeped_absolute = 0;

unsigned long currentMillis;

unsigned long allow_change_clock = 0;

// Botao UP
const byte button_up = 11; // pino 11
byte button_up_state = 0;
byte button_up_last_state = 0;
unsigned long button_up_startPressed = 0; // momento que foi pressionado
unsigned long button_up_endPressed = 0; // momento que foi solto
unsigned long button_up_holdTime = 0; // quanto tempo ficou pressionado
unsigned long button_up_idleTime = 0; // quanto tempo o botao teve inativo

```

Figura - Botões

```

unsigned long button_up_startPressed = 0; // momento que foi pressionado
unsigned long button_up_endPressed = 0; // momento que foi solto
unsigned long button_up_holdTime = 0; // quanto tempo ficou pressionado
unsigned long button_up_idleTime = 0; // quanto tempo o botao teve inativo

////////// Botao DOWN
const byte button_down = 10; // pino 10
byte button_down_state = 0;
byte button_down_last_state = 0;
unsigned long button_down_startPressed = 0; // momento que foi pressionado
unsigned long button_down_endPressed = 0; // momento que foi solto
unsigned long button_down_holdTime = 0; // quanto tempo ficou pressionado
unsigned long button_down_idleTime = 0; // quanto tempo o botao teve inativo

////////// Botao das luzes
const byte button_luzes = 9; // pino 9
byte button_luzes_state = 0;
byte button_luzes_last_state = 0;
unsigned long button_luzes_startPressed = 0; // momento que foi pressionado
unsigned long button_luzes_endPressed = 0; // momento que foi solto
unsigned long button_luzes_holdTime = 0; // quanto tempo ficou pressionado
unsigned long button_luzes_idleTime = 0; // quanto tempo o botao teve inativo

```

Figura – Botões

Após isto imaginei como poderia criar um menu, o usuário só poderá mexer nos LED's para mudar a cor enquanto o alarme, ajuste de horas e a ativação do alarme forem consideradas falsas.

O button_clock irá ser o botão que vai decidir as opções do menu. Se o utilizador só clicar e tirar imediatamente o dedo, apenas irá desligar o Display. Se o utilizador premir de 1.5 a 3 segundos o button_clock irá chamar a função para o ajuste de alarme e assim que o utilizador quiser confirmar o ajuste de alarme deverá novamente manter premido o botão durante 1.5 a 3 segundos. De 3 a 8 segundos o utilizador poderá definir se quiser ativar ou não o alarme. De 8 a 10 segundos o utilizador pode ajustar as horas do RTC e a sua confirmação é de 8 a 10 segundos.

O button_luzes apenas servirá para desligar e ligar os LED's mas no entanto os botões up e down podem alterar os efeitos para seguinte ou anterior.

```

experimento$
void button_clock_updateCounter() {
  // quando for pressionado botao clock
  if (button_clock_state == LOW) {
    button_clock_holdTime = millis() - button_clock_startPressed;

    if (button_clock_holdTime > 1500 && button_clock_holdTime<3000 ) { // quando for pressionado 1.5 segundos até 3
      Serial.println("ok1.5");
      if (allow_change_set_alarm) {

        set_alarm = !set_alarm;
        Serial.println("SET ALARM FLAG CHANGED");

        allow_change_set_alarm = false;

        allow_change_alarm_on = true;
      }
    }

    if (button_clock_holdTime > 3000 && button_clock_holdTime<8000) { // luando for premido de 3 a 8 segundos
      Serial.println("ok3");
      if (allow_change_alarm_on) {

        alarm_on = !alarm_on;
        set_alarm = !set_alarm;
        allow_change_hour_minute = true;
        show_alarm_status(alarm_on);
        if (alarm_on) {
          Serial.println("O Alarme está ativo");
        }
        if (!alarm_on) {
          Serial.println("O alarme está desativado");
        }
        allow_change_alarm_on = false;
      }
    }
  }
}

```

Figura – Funções button_clock

```

        set_alarm = !set_alarm;
        allow_change_hour_minute = true;
        show_alarm_status(alarm_on);
        if (alarm_on) {
            Serial.println("O Alarme está ativo");
        }
        if (!alarm_on) {
            Serial.println("O alarme está desativado");
        }
        allow_change_alarm_on = false;
    }
}

if (button_clock_holdTime > 8000 && button_clock_holdTime<10000) { // If it was pressed for more than 8 seconds:
    Serial.println("ok8");
    if (allow_change_hour_minute) {

        Serial.println("As suas horas foram definidas");

        if (set_hour_minute_flag) {
            set_hour_minute();
        }

        set_hour_minute_flag = !set_hour_minute_flag;
        set_alarm = false;
        clock_on = true;

        get_current_time_once();

        alarm_on = !alarm_on;
        allow_change_hour_minute = false;
    }
}
else {
    button_clock_idleTime = millis() - button_clock_endPressed;
}

```

Figura – Funções button_clock

```

void button_clock_update_state() {

    allow_change_set_alarm = true; // mudar alarme

    if (digitalRead(button_clock) == LOW) { // If clock button is pressed:

        button_clock_startPressed = millis();
        button_clock_idleTime = button_clock_startPressed - button_clock_endPressed;

        clock_on = !clock_on; // liga e desliga o display se for falso !clock on liga clock_on e vice versa
    }
}

```

Figura – Funções button_clock

```

void button_luzes_update_state() {

    if (digitalRead(button_luzes) == LOW) { // quando o botão for solto

        button_luzes_startPressed = millis();
        button_luzes_idleTime = button_luzes_startPressed - button_luzes_endPressed;

        // ligar e desligar leds
        luzes_on = !luzes_on;

    }

    else {
        button_luzes_endPressed = millis();
        button_luzes_holdTime = button_luzes_endPressed - button_luzes_startPressed;
    }
}

```

Figura – Funções button_luzes

```
void button_up_update_state() {

    if (digitalRead(button_up) == LOW) { // se for pressionado

        button_up_startPressed = millis();
        button_up_idleTime = button_up_startPressed - button_up_endPressed;

        //Mudar o valor das horas no alarme até 24
        if (set_alarm) {

            alarm_time_hour = alarm_time_hour + 1;

            EEPROM.write(alarm_time_hour_address, alarm_time_hour);

            if (alarm_time_hour >= 24) {
                alarm_time_hour = 0;
                EEPROM.write(alarm_time_hour_address, alarm_time_hour);
            }

        }

    }
    else {
        // Alterar os efeitos irá andar o n° a frente

        luzes_mode += 1;

        if (luzes_mode > num_luzes_modes) {
            luzes_mode = 1;
        }

        button_up_endPressed = millis();
        button_up_holdTime = button_up_endPressed - button_up_startPressed;

    }

    // Alterar o valor das horas no relógio
```

Figura – Função botão up

```
EEPROM.write(alarm_time_hour_address, alarm_time_hour);

    if (alarm_time_hour >= 24) {
        alarm_time_hour = 0;
        EEPROM.write(alarm_time_hour_address, alarm_time_hour);
    }

}

else {
    // Alterar os efeitos irá andar o n° a frente

    luzes_mode += 1;

    if (luzes_mode > num_luzes_modes) {
        luzes_mode = 1;
    }

    button_up_endPressed = millis();
    button_up_holdTime = button_up_endPressed - button_up_startPressed;

}

// Alterar o valor das horas no relógio
if (set_hour_minute_flag) {

    new_hour = new_hour + 1;

    if (new_hour >= 24) {
        new_hour = 0;
    }

}
```

Figura – Função botão up

```

exprimento $
void button_down_update_state() {

    if (digitalRead(button_down) == LOW) { // se o botao down for pressionado

        button_down_startPressed = millis();
        button_down_idleTime = button_down_startPressed - button_down_endPressed;

        // mudar valores do alarme
        if (set_alarm) {
            alarm_time_minute = alarm_time_minute + 5;

            EEPROM.write(alarm_time_minute_address, alarm_time_minute);

            if (alarm_time_minute >= 60) {
                alarm_time_minute = 0;
                EEPROM.write(alarm_time_minute_address, alarm_time_minute);
            }
        }
        else {

            luzes_mode -= 1;

            if (luzes_mode <= 0) {
                luzes_mode = num_luzes_modes;
            }

            button_down_endPressed = millis();
            button_down_holdTime = button_down_endPressed - button_down_startPressed;

        }
    }
}

```

Figura – Função botão down

```

exprimento $
EEPROM.write(alarm_time_minute_address, alarm_time_minute);

    if (alarm_time_minute >= 60) {
        alarm_time_minute = 0;
        EEPROM.write(alarm_time_minute_address, alarm_time_minute);
    }

}

else {

    luzes_mode -= 1;

    if (luzes_mode <= 0) {
        luzes_mode = num_luzes_modes;
    }

    button_down_endPressed = millis();
    button_down_holdTime = button_down_endPressed - button_down_startPressed;

}

// Mudar o valor(relógio) dos minutos até 60
if (set_hour_minute_flag) {

    new_minute = new_minute + 1;

    if (new_minute >= 60) {
        new_minute = 0;
    }
}

}
}

```

Figura – Função botão down

Finalizado o sistema manual fotos dos testes.

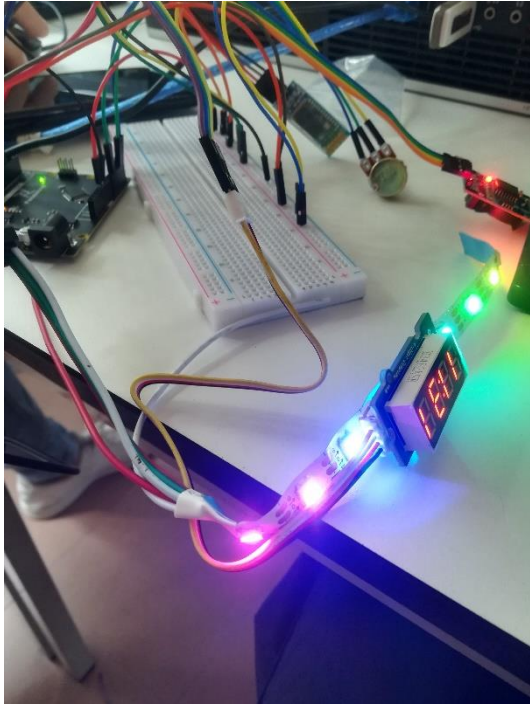


Figura - Testes

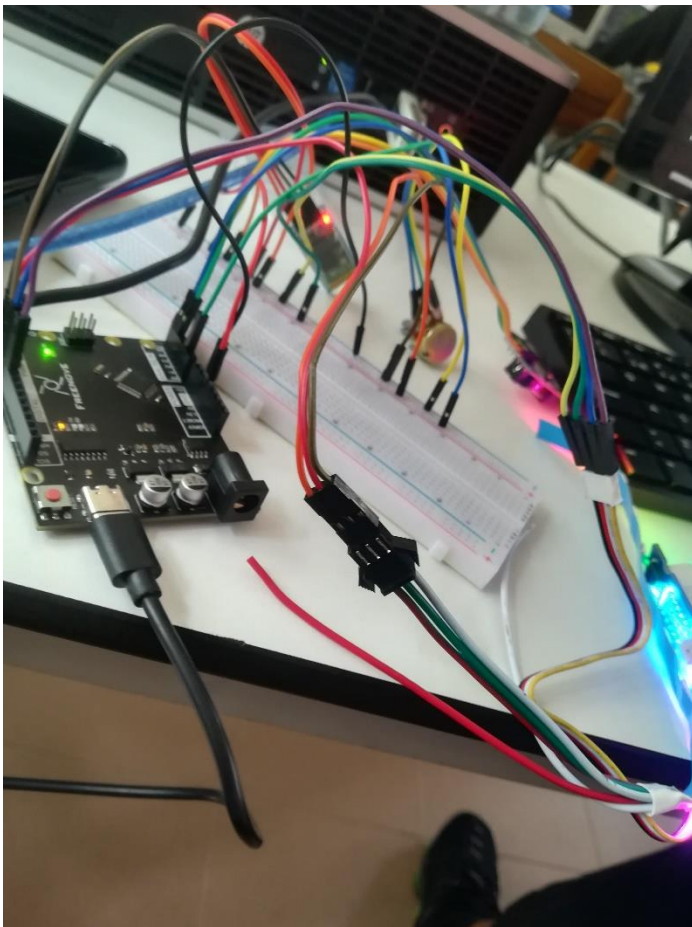


Figura - Testes

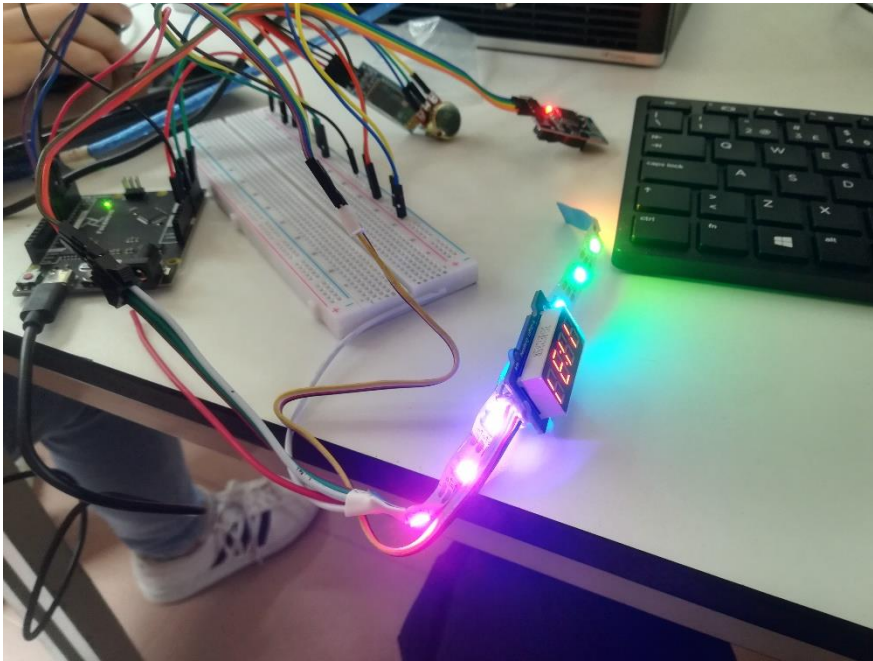


Figura - Testes

Etapa 5 – Estrutura do Projeto

Após programar o arduino com todas a funcionalidades para uso manual cumpridas, comecei a pensar na estrutura do meu projeto, algo como um cubo, onde desenhei no tinkercad.

Este cubo irá ser feito de acrílico para uma iluminação boa e que consiga esconder o circuito contido dentro da estrutura e MDF para a sua base.

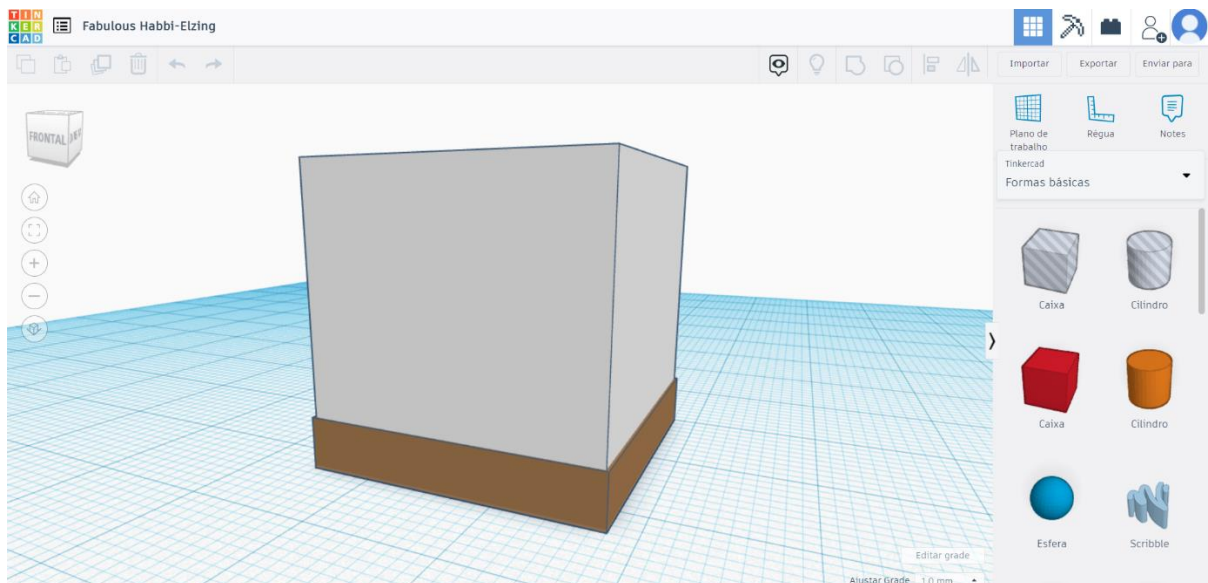


Figura – Ideia

Tiradas as ideias comecei a tirar medidas para o cubo acrílico terá uma base superior de 23 cm de cada lado, e para as laterais que são 4 terão 23 cm de comprimento e 19 cm de altura.

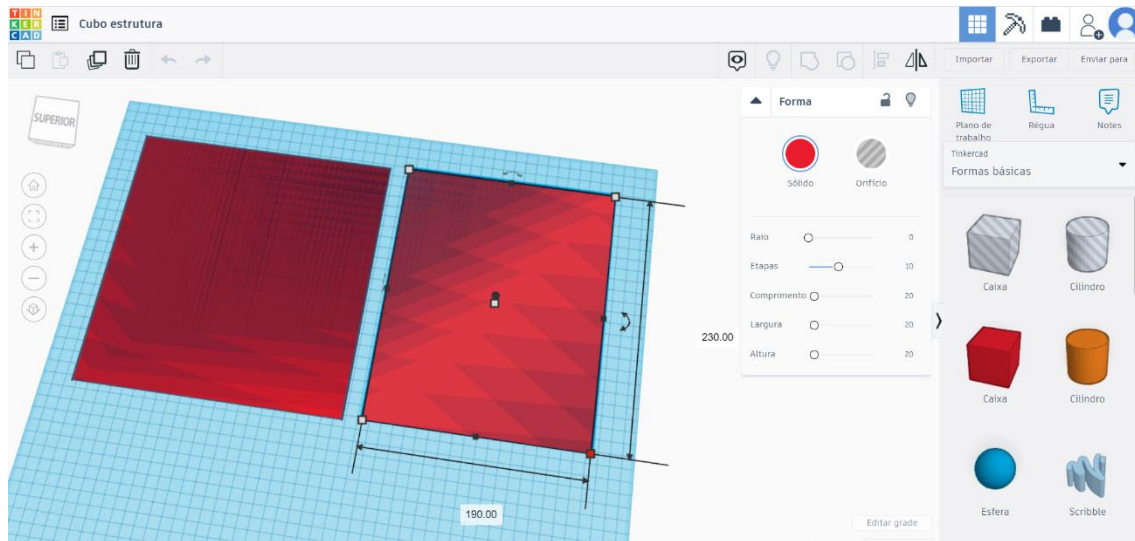


Figura – Estrutura Topo

Depois comecei a estruturar como queria que os cortes MDF fossem, uma placa para colocar o Display, outra para colocar os botões e o potenciômetro e outra para colocar o cabo de alimentação. Cada placa tem 23 cm de comprimento e 4 cm de altura, a base tem 23 cm de cada lado.

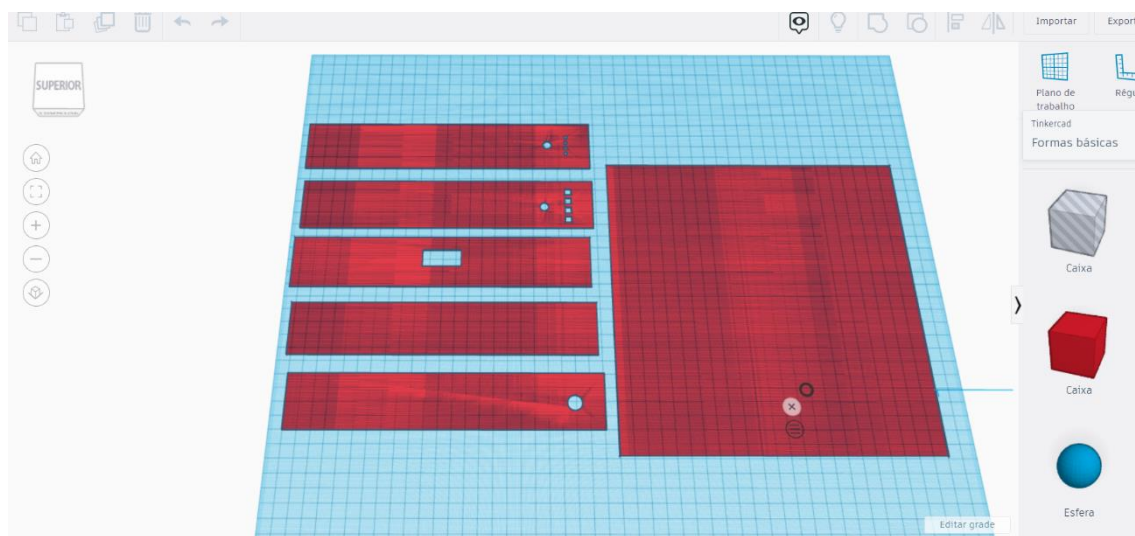


Figura – Estrutura MDF

Etapa 6 – Bluetooth Arduino

De seguida comecei a pensar na aplicação pesquisando um pouco de como iria ligar o telemóvel à aplicação e como ia programar a aplicação.

Primeiro comecei a programar no arduino a informação que o HC-06 irá receber e realizar.

Para a instalação do HC-06, usei o `#include <SoftwareSerial>`, assim posso simular um pino digital qualquer como pino de comunicação serial sem ter que usar os pinos 0 e 1 que devem ser usados para fins de depuração. O pino 6 corresponde ao TXD e o pino 5 ao RXD.

O char command servirá para armazenar os dados recebidos da aplicação.

```
#include <SoftwareSerial.h>
#include <TM1637Display.h>

//////////////////// Bluetooth pinos
SoftwareSerial bluetooth(6, 5);
```

Figura – Bluetooth Arduino

```
////////////////////Bluetooth
String comando_bt;
String data_bt = "";
char command;
int dados;
```

Figura – Bluetooth Arduino

Assim que for estabelecida uma ligação entre a aplicação e o arduino, é preciso uma variável que leia e armazene os dados recebidos.

As letras que são igualadas a command irão servir como meios de comunicação, isto é a aplicação manda X para o bluetooth e quando X for recebido e lido pelo bluetooth o aplicativo irá executar o que se pretender que X execute.

```

////////////////////////////////////Bluetooth

if (bluetooth.available()) {

    command = 'x';

    while (bluetooth.available()) {

        char char_bt = (bluetooth.read());

        comando_bt += char_bt;

        if (command == 'x') {
            command = char_bt;
        }

    }

    data_bt = comando_bt.substring(1);
}

while (command != 'x') {

    //ligar led
    if (command == 'J') {
        luzes_on = !luzes_on;
        command = 'x';
        break;
    }

    //ligar TM1637
    if (command == 'X'){
        clock_on = !clock_on;
        if (clock_on) {
            Serial.println("ecra on");
        }
    }
}

```

Figura – Bluetooth Arduino

```

| experimento
    }
    if (!clock_on) {
        Serial.println("ecra off");
    }
    command = 'x';
    break;
}

if (command == 'U'){
    luzes_mode = 1;
    command = 'x';
    break;
}

if (command == 'N'){
    luzes_mode = 2;
    command = 'x';
    break;
}

if (command == 'U'){
    luzes_mode = 3;
    command = 'x';
    break;
}

if (command == 'P'){
    luzes_mode = 4;
    command = 'x';
    break;
}

if (command == 'L'){
    luzes_mode = 5;
    command = 'x';
    break;
}

if (command == 'T'){

```

Figura – Bluetooth Arduino

```

break;
}

if (command == 'M'){
  luzes_mode += 1;

  if (luzes_mode > num_luzes_modes) {
    luzes_mode = 1;
  }
  command = 'x';
  break;
}

if (command == 'E') {

  static String hora = data_bt.substring(1, 2);
  static String minuto = data_bt.substring(3, 4);

  int int_minuto = minuto.toInt();
  int int_hora = hora.toInt();

  EEPROM.write(alarm_time_minute_address, int_minuto);
  EEPROM.write(alarm_time_hour_address, int_hora);

  Serial.println("Alarme ajustado para alarm_time_hour_address : alarm_time_minute_address");

  command = 'x';
  break;
}

if (command == 'A') {
  alarm_on = !alarm_on;
  if (alarm_on) {
    Serial.println("O Alarme está ativo");
  }
}

```

Figura – Bluetooth Arduino

Etapa 7 – Aplicação Bluetooth

Para perceber como o Android Studio cria as aplicações temos que perceber o que são as Manifests e como os Layouts funcionam.

As Manifests funcionam como o coração da aplicação é como um Admin, que dá certas permissões para a app funcionar corretamente.

O layout como o próprio nome diz é a tela que aparece, parte estética da aplicação. Mas no entanto para as funcionalidades dele programamos numa Activity que é ligada ao layout.

Na minha manifest apenas preciso de permissão para a aplicação usar Bluetooth.

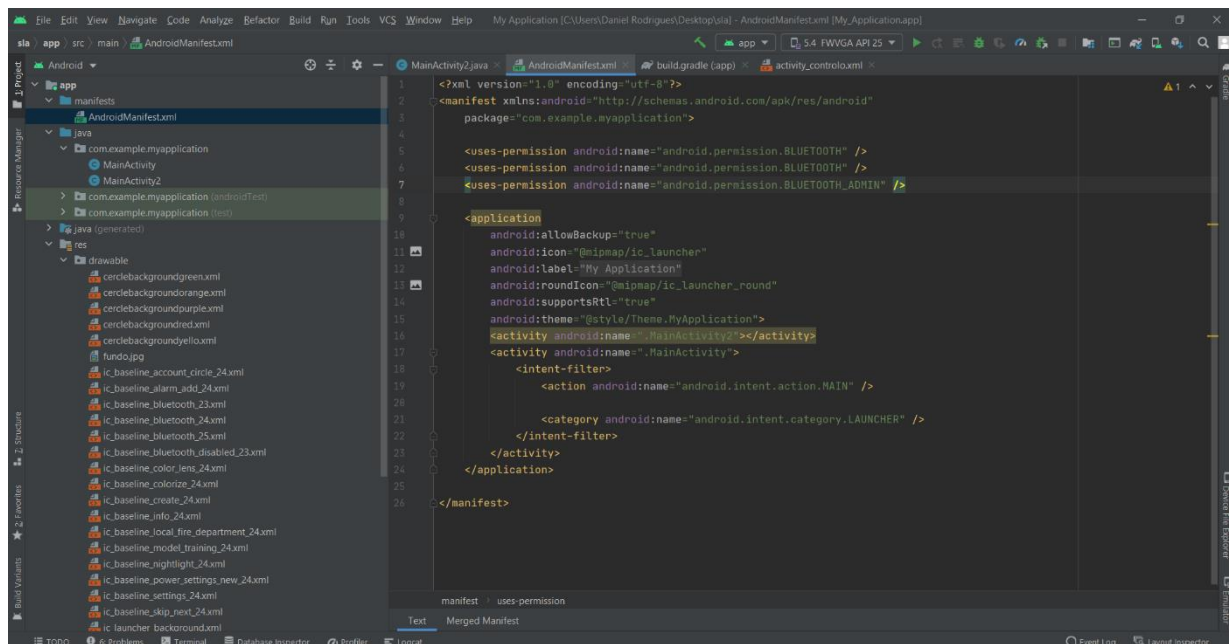


Figura – Manifest

Feito isso comecei a pensar no Layout como queria que fosse.

Neste projeto irei usar 2 layouts, um layout inicial onde a aplicação irá pedir acesso Bluetooth e uma lista de aparelhos emparelhados no telemóvel para criar uma conectividade entre o telemóvel e o arduino.

O outro layout servirá para o controlo do arduino.

O primeiro Layout terá uma TextView, um Button e uma ListView.

Quando o Button for clicado ele irá então criar a ListView, onde irá ler do telemóvel todos os dispositivos que estão registados por bluetooth numa lista.

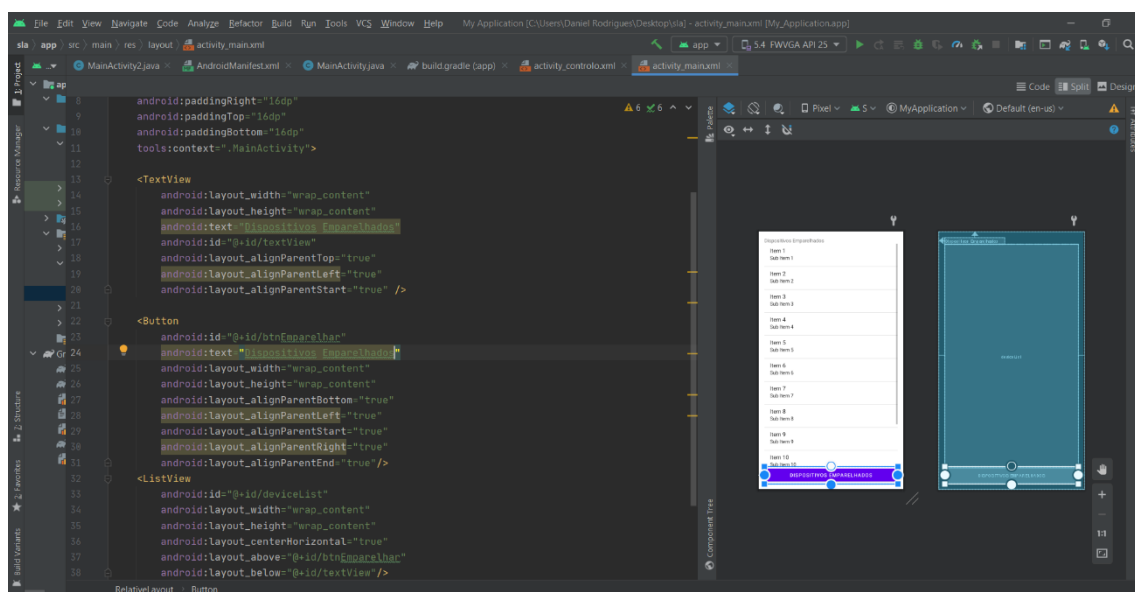


Figura – Layout1

Como já disse o layout para ter funcionalidades depende da sua Activity.

Primeiramente assim que o utilizador abrir a aplicação, a aplicação irá pedir ao dispositivo para ligar o bluetooth.

Se o dispositivo não tiver bluetooth uma caixa de texto irá aparecer e a aplicação fecha.

Quando o botão dos dispositivos emparelhados for clicado usamos a função OnClickListener para poder executar o quisermos assim que o botão for clicado, no caso irá aparecer a lista de todos os dispositivos emparelhados.

```

myBluetooth =BluetoothAdapter.getDefaultAdapter ();
if(myBluetooth == null)
{
    Toast.makeText (getApplicationContext (), text: "Dispositivo Bluetooth não disponível", Toast.LENGTH_LONG).show ();
    finish();
}
else
{
    if (myBluetooth.isEnabled())
    { }
    else
    {
        Intent turnBTon = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
        startActivityForResult(turnBTon, requestCode: 1);
    }
    btnPaired.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v){ pairedDevicesList(); }
    });
}
}

```

Figura – Activity1

Quando o botão for clicado a aplicação irá fazer uma lista de todos os dispositivos emparelhados usando “pairedDevices” e “devicesLists”, se não houver nenhum então aparecerá nenhum Dispositivo encontrado.

Depois basta só clicar no HC-06 e a aplicação irá obter o seu endereço MAC para estabelecer uma conexão e irá abrir uma nova Activity.

```

private void pairedDevicesList() {
    pairedDevices = myBluetooth.getBondedDevices ();
    ArrayList list = new ArrayList ();

    if (pairedDevices.size()>0)
    {
        for (BluetoothDevice bt :pairedDevices)
        {
            list.add (bt.getName () + "\n" + bt.getAddress ());
        }
    }
    else
    {
        Toast.makeText (getApplicationContext (), "Nenhum dispositivo Bluetooth encontrado.", Toast.LENGTH_SHORT).show ();
    }

    final ArrayAdapter adapter = new ArrayAdapter (context, this, android.R.layout.simple_list_item_1, list);
    deviceLists.setAdapter (adapter);
    deviceLists.setOnItemClickListener (myItemClickListener);

    private AdapterView.OnItemClickListener myItemClickListener = new AdapterView.OnItemClickListener()
    {
        public void onItemClick (AdapterView av, View v, int arg2, long arg3)
        {
            // obter o endereço MAC do module bluetooth
            String info = ((TextView) v).getText().toString();
            String address = info.substring(info.length() - 17);

            Intent i = new Intent(MainActivity.this, MainActivity2.class);
            i.putExtra(EXTRA_ADDRESS, address);
            startActivity(i);
        }
    }
}
    
```

Figura - Activity1

Para este novo Layout usei uma Libraby o CardView que irá dar um Aspetto de Cards, para além disso usei uma função para haver uma espécie de Scroll no Layout.

Para este Layout terei Cards, Botões e um TimePicker para o ajuste de horas e alarme. Os ícones usados nos Cards são ícones que o AndroidStudio nos disponibiliza que depois podemos editar-los como quisermos e os círculos que os circundam são feitos em código.

```

<android.support.design.widget.CardView
    android:id="@+id/power_card"
    android:layout_width="168dp"
    android:layout_height="198dp"
    android:layout_margin="18dp"
    android:clickable="true"
    android:foreground="?android:attr/selectableItemBackground">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        android:gravity="center">

        <ImageView
            android:layout_width="64dp"
            android:layout_height="64dp"
            android:background="@drawable/circlebackgroundvella"
            android:src="@drawable/ic_baseline_power_settings_new_24"
            android:padding="18dp"/>

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textStyle="bold"
            android:layout_marginTop="18dp"
            android:text="Ligar/Desligar Leds"/>

    </LinearLayout>
</android.support.design.widget.CardView>
    
```


Figura – Layout2

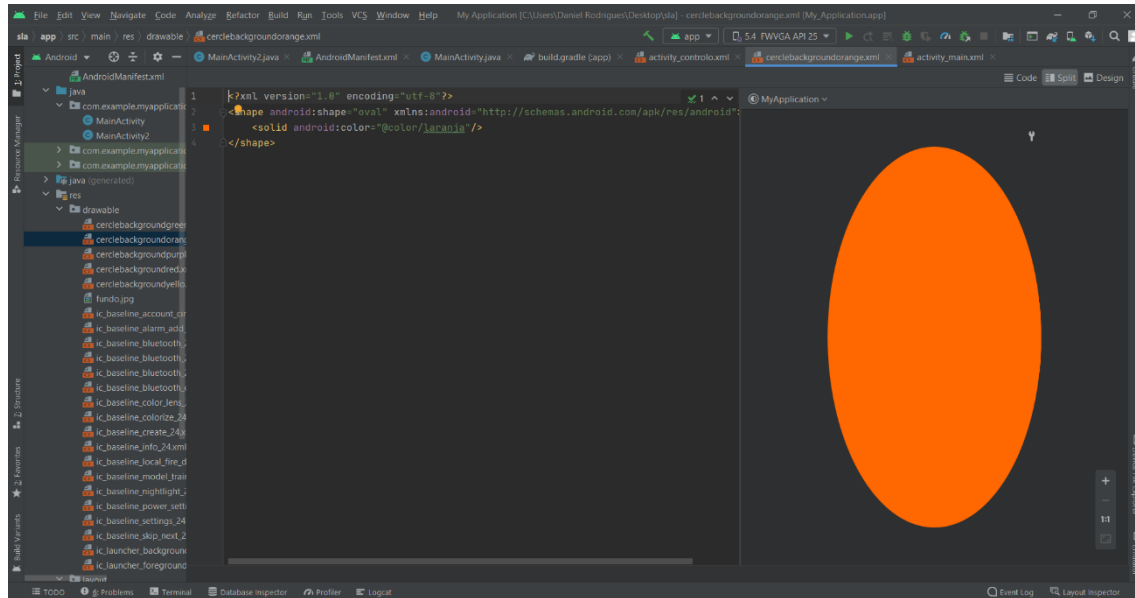


Figura – Circulos dos Ícones

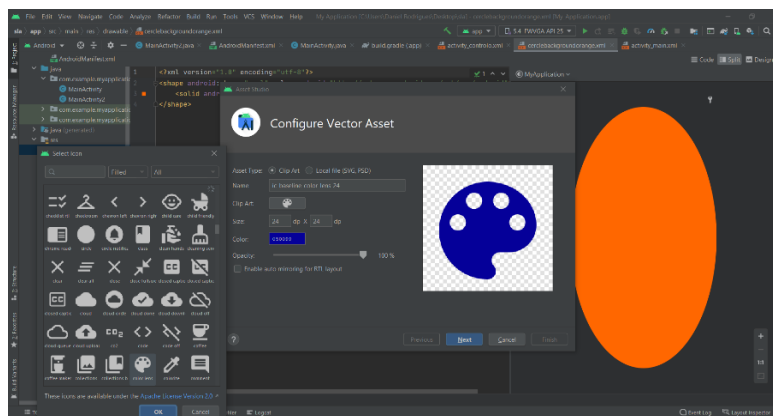


Figura – Ícones

Assim que iniciar a tela irá aparecer uma caixa de texto equanto se conecta.

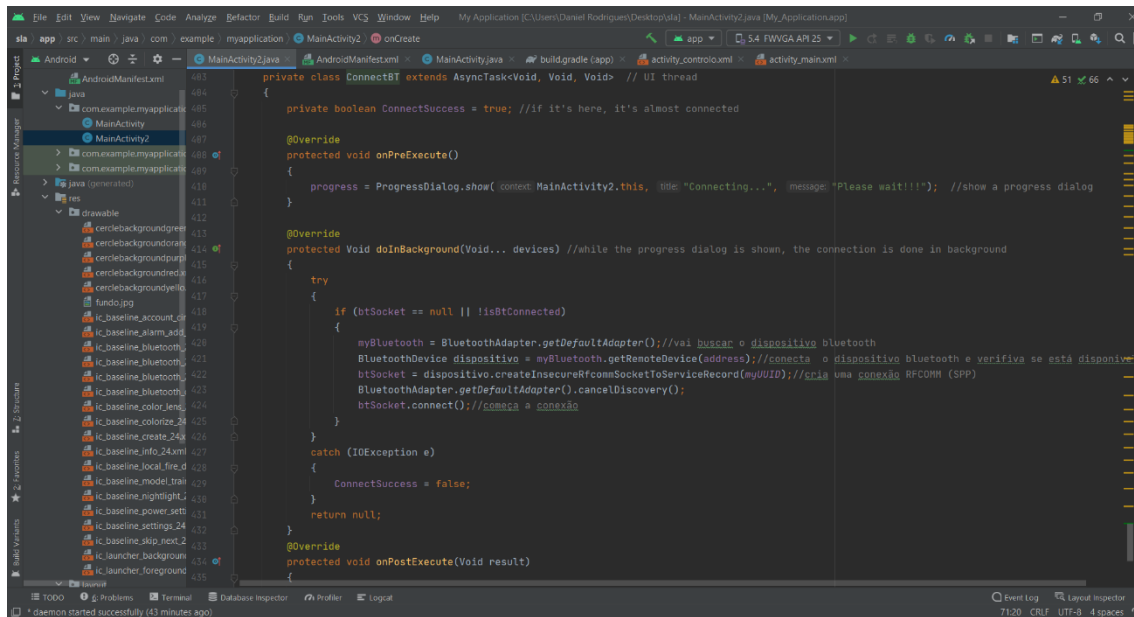


Figura – Activity1

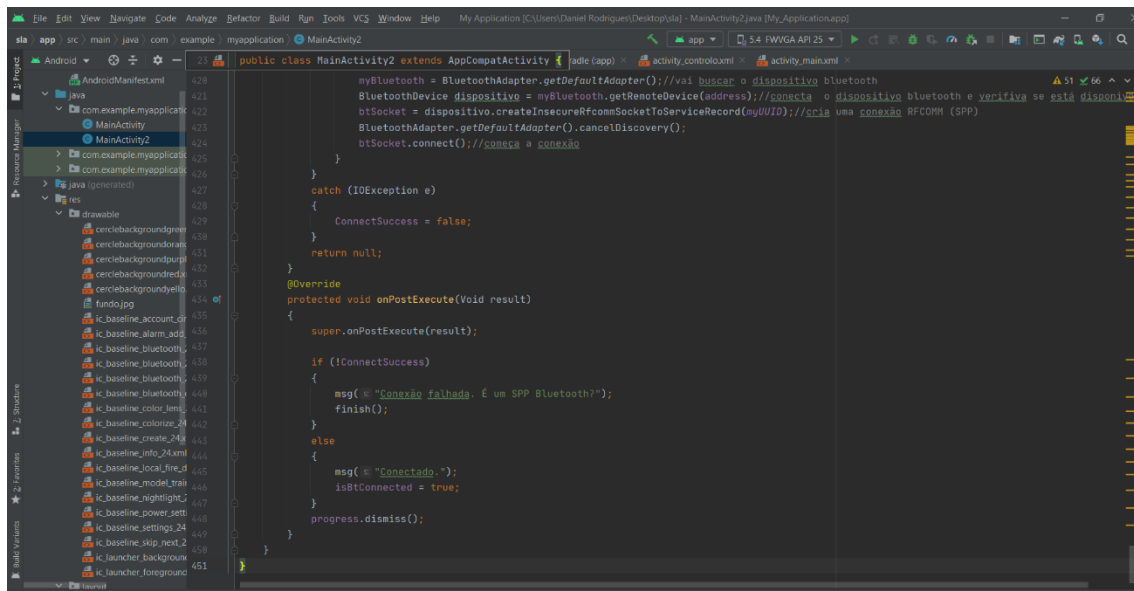


Figura – Activity1

O btSocket corresponde à conexão entre o telemóvel e o arduino.

Quando clicarmos num botão para o controlo do Arduino e como pretendemos enviar a informação então escrevemos

```
btSocket.getOutputStream().write("A".getBytes());
```

O “A” que irá corresponder ao dado recebido pelo Arduino que depois irá executar a sua função.

Prova de Aptidão Científica

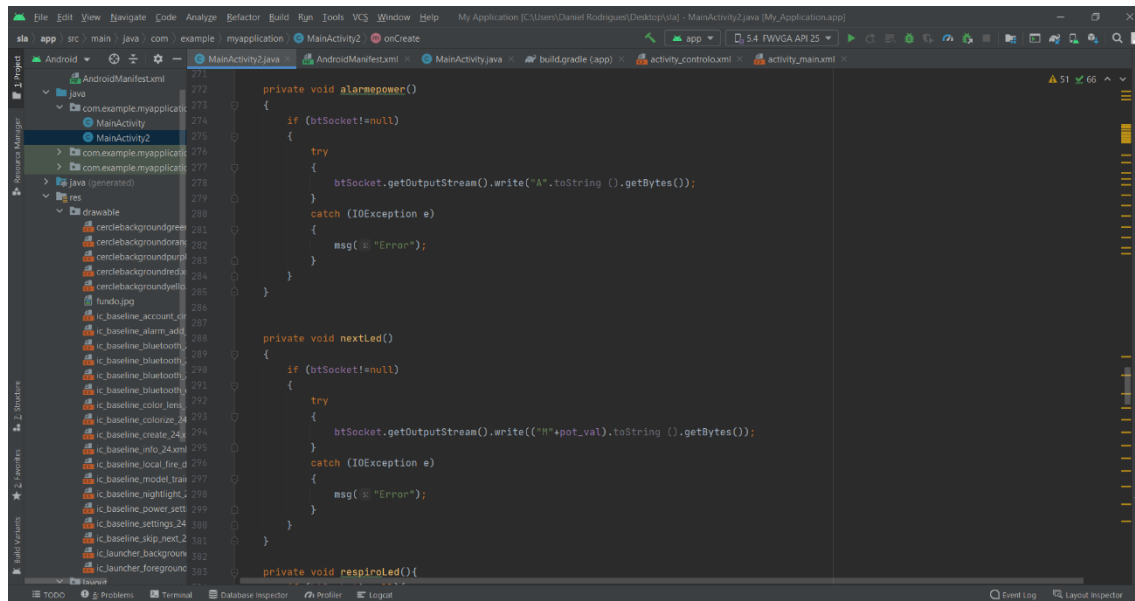


Figura – Activity2