

**LAPORAN TUGAS
SISTEM TERTANAM**

**Rancang Bangun Sistem Penampil Jam dengan Timer 24 Jam Beralarm Multi-waktu
dengan Teks dan Penyesuaikan Kecerahan Tampilan yang Selaras terhadap
Cahaya Sekitar Serta Suhu Ruangan Tempat Unit Berada**



Disusun oleh :
Muhammad Zakariya Nur Ramdhani
(07211940000016)

Dosen :
Eko Pramunanto, S.T., M.T.

DEPARTEMEN TEKNIK KOMPUTER
FAKULTAS TEKNOLOGI ELEKTRO DAN INFORMATIKA CERDAS
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2022

BAB I

PENDAHULUAN

A. Latar Belakang

Di masa sekarang ini teknologi dan ilmu pengetahuan sudah berkembang dengan pesat dan tentunya sangat membantu dalam kegiatan sehari-hari atau kehidupan manusia. Dirasakan pula bahwa teknologi dan ilmu pengetahuan berkembang dengan sangat cepat dan pesat terutama di bidang industri dan elektronika. Masyarakat cenderung menggunakan teknologi untuk menunjang kinerja juga memenuhi kebutuhan manusia. Manusia tidak pernah terlepas dari waktu, waktu adalah rangkaian saat ketika suatu peristiwa terjadi. Dalam kehidupan sehari hari manusia dibiasakan untuk mendisiplinkan diri, dimulai dari ketepatan dalam memenuhi undangan dll. Untuk mengetahui apakah datang tepat pada waktu atau tidak, maka alat ukur waktu yang digunakan adalah jam. Jam adalah suatu alat yang digunakan untuk menunjukkan waktu. Jam ini merupakan salah satu penemuan paling tua di dunia yang sangat penting kegunaannya.

Seiring dengan berkembangnya teknologi digital, saat ini banyak dibuat jam digital sebagai alat penunjuk waktu yang lebih efisien daripada jam dengan penunjukan jarum. Jam digital sendiri merupakan salah satu tipe dari jam yang menampilkan waktu secara digital yaitu dengan menampilkan angka-angka dalam sebuah layar atau display. Selain digunakan untuk menunjukkan waktu, jam digital seringkali digunakan untuk menjadi pengingat waktu (alarm) hingga penunjuk suhu ruangan. Manusia memanfaatkan alarm untuk mengingatkannya tentang kegiatan-kegiatan yang penting seperti untuk membangunkan di pagi hari, pengingat rapat, dan lain-lain. Selain itu dengan adanya penunjuk suhu pada jam digital, seseorang dapat mengetahui suhu ruangan tempat jam berada.

Berdasarkan latar belakang diatas, maka penulis ingin membuat “Sistem penampil jam dengan timer 24 jam beralarm multi-waktu dengan teks dan penyesuai kecerahan tampilan yang selaras terhadap cahaya sekitar serta suhu ruangan tempat unit berada” untuk memenuhi tugas besar dari mata kuliah Sistem tertanam.

B. Rumusan Masalah

1. Bagaimana cara merancang dan membuat sistem penampil jam dengan timer 24 jam beralarm multi-waktu dengan teks dan penyesuai kecerahan tampilan yang selaras terhadap cahaya sekitar serta suhu ruangan tempat unit berada?

C. Tujuan

1. Mengetahui cara merancang dan membuat sistem penampil jam dengan timer 24 jam beralarm multi-waktu dengan teks dan penyesuai kecerahan tampilan yang selaras terhadap cahaya sekitar serta suhu ruangan tempat unit berada.

BAB II

TINJAUAN PUSTAKA

A. Arduino Uno

Arduino Uno adalah board mikrokontroler berbasis ATmega328 (datasheet). Memiliki 14 pin input dari output digital dimana 6 pin input tersebut dapat digunakan sebagai output PWM dan 6 pin input analog, 16 MHz osilator kristal, koneksi USB, jack power, ICSP header, dan tombol reset. Untuk mendukung mikrokontroler agar dapat digunakan, cukup hanya menghubungkan Board Arduino Uno ke komputer dengan menggunakan kabel USB atau listrik dengan AC yang-ke adaptor-DC atau baterai untuk menjalankannya. Setiap 14 pin digital pada arduino uno dapat digunakan sebagai input dan output, menggunakan fungsi pinMode(), digitalWrite(), dan digitalRead(). Fungsi fungsi tersebut beroperasi di tegangan 5 volt, Setiap pin dapat memberikan atau menerima suatu arus maksimum 40 mA dan mempunyai sebuah resistor pull-up (terputus secara default) 20-50 kOhm. Arduino Uno memiliki sejumlah fasilitas untuk berkomunikasi dengan computer, Arduino Uno lain, atau mikrokontroler lain. ATMega3282 ini menyediakan UART TTL (5v) komunikasi serial, yang tersedia pada pin digital 0 (RX) dan 1 (TX).



Gambar 2.1 Arduino Uno R3

B. RTC(Real Time Clock)

RTC (*Real Time Clock*) merupakan chip IC yang mempunyai fungsi menghitung waktu yang dimulai dari detik, menit, jam, hari, tanggal, bulan, hingga tahun dengan akurat. Untuk menjaga atau menyimpan data waktu yang telah di-ON-kan pada module terdapat sumber catu daya sendiri yaitu baterai jam kancing, serta keakuratan data waktu yang ditampilkan digunakan osilator kristal eksternal. Sehingga saat perangkat mikrokontroler terhubung dengan RTC ini sebagai sumber data waktu dimatikan, data waktu yang sudah terbaca dan ditampilkan tidak akan hilang begitu saja. Dengan catatan baterai yang terhubung pada RTC tidak habis dayanya. Contoh yang dapat ditemui dalam kehidupan sehari – hari yaitu pada motherboard PC yang biasanya letaknya berdekatan dengan chip BIOS. Difungsikan guna menyimpan sumber informasi waktu terkini sehingga jam akan tetap up to date walaupun komputer tersebut dimatikan.

C. LED Dot Matrix MAX 7219

Dot Matrix ini pada umumnya terdiri dari beberapa LED atau light Emitting Diode (berbentuk “Dot”) yang disusun membentuk matriks, memiliki 8 kolom dan 8 baris (8x8) atau dengan ukuran lainnya. Sedangkan untuk MAX7219 merupakan sebuah IC Shift Register yang khusus dirancang untuk mengontrol dot matrix, 7

segment maupun independen LED. Kolom pada dot matrix berfungsi sebagai katoda (Common Cathode) dan baris sebagai anoda (Common anode) atau sebaliknya. Cara pengoperasian modul ini dengan cara multiplexing ataupun dengan cara multiplexed display. Dot Matrix dengan tipe Max7219 ini mempunyai nilai tegangan normal sebesar 5V dengan ukuran 12.8 cm x 12.8 cm x 1.3 cm. Memiliki 5 pin yaitu VCC, Ground, DIN, CS, dan Clock.

D. Push Button

Push button switch (saklar tombol tekan) adalah perangkat / saklar sederhana yang berfungsi untuk menghubungkan atau memutuskan aliran arus listrik dengan sistem kerja tekan unlock (tidak mengunci). Sistem kerja unlock disini berarti saklar akan bekerja sebagai device penghubung atau pemutus aliran arus listrik saat tombol ditekan, dan saat tombol tidak ditekan (dilepas), maka saklar akan kembali pada kondisi normal.

Sebagai device penghubung atau pemutus, push button switch hanya memiliki 2 kondisi, yaitu On dan Off (1 dan 0). Istilah On dan Off ini menjadi sangat penting karena semua perangkat listrik yang memerlukan sumber energi listrik pasti membutuhkan kondisi On dan Off. Karena sistem kerjanya yang unlock dan langsung berhubungan dengan operator, push button switch menjadi device paling utama yang biasa digunakan untuk memulai dan mengakhiri kerja mesin di industri. Secanggih apapun sebuah mesin bisa dipastikan sistem kerjanya tidak terlepas dari keberadaan sebuah saklar seperti push button switch atau perangkat lain yang sejenis yang bekerja mengatur pengkondisian On dan Off.

E. Light Dependent Resistant (LDR)

LDR (Light Dependent Resistant) merupakan suatu jenis resistor yang nilai resistansinya berubah-ubah karena adanya intensitas cahaya yang diserap. LDR dibentuk dari Cadmium Sulfide (CDS) yang mana Cadmium Sulfide dihasilkan dari serbuk keramik. Prinsip kerja LDR ini pada saat mendapatkan cahaya maka tahanannya turun, sehingga pada saat LDR mendapatkan kuat cahaya terbesar maka tegangan yang dihasilkan adalah tertinggi. Pada saat gelap atau cahaya redup, bahan dari cakram pada LDR menghasilkan elektron bebas dengan jumlah yang relatif kecil. Sehingga hanya ada sedikit elektron untuk mengangkut muatan elektrik. Artinya pada saat cahaya redup LDR menjadi penghantar arus yang kurang baik, atau bisa disebut juga LDR memiliki resistansi yang besar pada saat gelap atau cahaya redup. Pada saat cahaya terang, ada lebih banyak elektron yang lepas dari bahan semikonduktor tersebut. Sehingga akan ada lebih banyak elektron untuk mengangkut muatan elektrik. Artinya pada saat cahaya terang LDR menjadi konduktor atau bisa disebut juga LDR memiliki resistansi yang kecil pada saat cahaya terang.

LDR digunakan untuk mengubah energi cahaya menjadi energi listrik. Saklar cahaya otomatis adalah salah satu contoh alat yang menggunakan LDR. Akan tetapi karena responnya terhadap cahaya cukup lambat, LDR tidak digunakan pada situasi dimana intensitas cahaya berubah secara drastis. Contoh Aplikasi LDR sebagai sensor cahaya diantaranya: Rangkaian alarm, indikator, counter (penghitung), dan potensiometer

F. Sensor Suhu LM35

Sensor suhu LM35 adalah komponen elektronika yang memiliki fungsi untuk mengubah besaran suhu menjadi besaran listrik dalam bentuk tegangan. Sensor Suhu LM35 yang dipakai dalam penelitian ini berupa komponen elektronika yang

diproduksi oleh National Semiconductor. LM35 memiliki keakuratan tinggi dan kemudahan perancangan jika dibandingkan dengan sensor suhu yang lain, LM35 juga mempunyai keluaran impedansi yang rendah dan linieritas yang tinggi sehingga dapat dengan mudah dihubungkan dengan rangkaian kendali khusus serta tidak memerlukan penyetelan lanjutan. LM35 berfungsi untuk melakukan pendektsian terhadap suhu yang akan diukur, Sensor suhu LM35 ini mempunyai jangkauan pengukuran suhu antara 0 – 100 derajat Celcius dengan kenaikan 10 mV untuk tiap derajat Celcius yang berarti bahwa setiap kenaikan suhu (0°C) maka akan terjadi kenaikan tegangan sebesar 10 mV, dimana output dari LM35 ini yang menyatakan kondisi perubahan dari suhu lingkungan. Setiap terjadi perubahan suhu maka akan terjadi perubahan data output yang dihasilkan, dimana perubahan tersebut berupa perbedaan tegangan yang dihasilkan. Sensor Suhu LM35 ini tidak memerlukan peng-kalibrasi atau penyetelan dari luar karena ketelitiannya sampai lebih kurang seperempat derajat celcius pada temperatur ruang. Komponen ini bekerja pada arus 60 mA sampai 5 mA serta mempunyai impedansi masukan kurang dari 1.

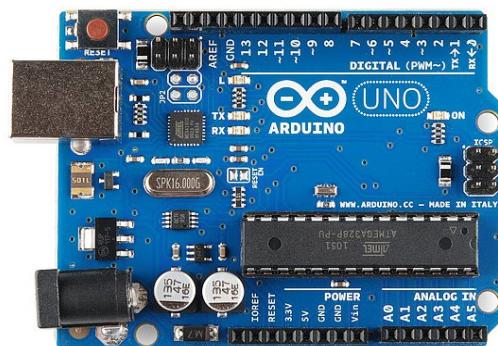
BAB III

PEMBAHASAN

A. Alat dan Bahan

Alat dan bahan yang digunakan pada percobaan ini antara lain:

1. Arduino Uno



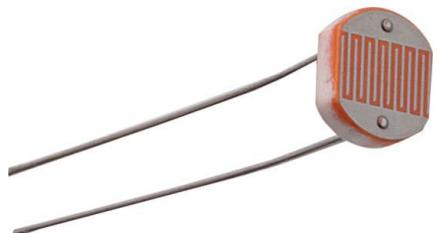
2. RTC DS1302



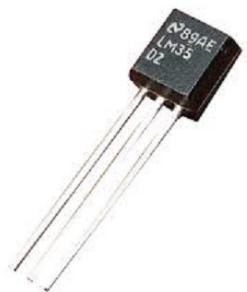
3. LED Dot Matrix MAX7219



4. LDR



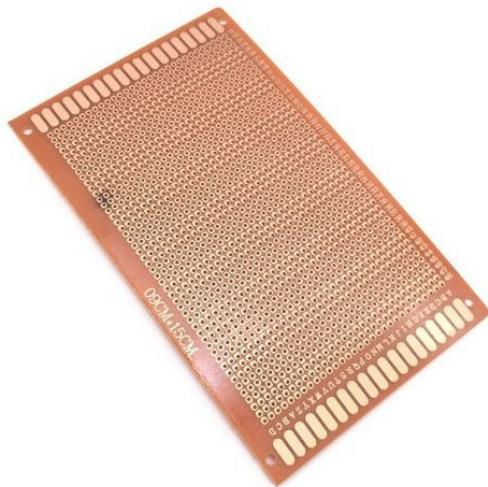
5. LM35



6. Push Button



7. PCB Dot Matrix



8. Resistor 220 Ohm dan 10K Ohm



9. Kabel Jumper



B. Deskripsi Alat

Sistem penampil jam dengan pewaktu (timer) 24 jam ber-alarm multi-waktu dengan teks (4 unit dot matrix 8x8) dan penyesuaikan kecerahan tampilan yang selaras terhadap cahaya sekitar (menggunakan sensor cahaya) serta suhu ruangan di tempat unit berada. Alarm dapat diatur waktunya sampai 5 set pengaturan waktu untuk menampilkan pesan teks tertentu selama waktu tertentu pula. Setiap set waktu pengaturan alarm dapat diaktifkan atau dinonaktifkan menurut kebutuhan.

Pengaturan dilakukan dengan tombol input untuk :

1. pengaturan waktu jam
2. pengaturan waktu alarm 1 s/d 5 bersama durasinya (atau waktu off-nya)
3. input boleh menggunakan tombol keypad (atau keyboard USB untuk masukan teks alarm 5 dst).

Tampilan :

1. 4 unit dot matrix 8X8 (boleh diperpanjang sampai 8 unit)
2. Teks yang wajib ditampilkan adalah :

Alarm 1 : NRP

Alarm 2 : Nama

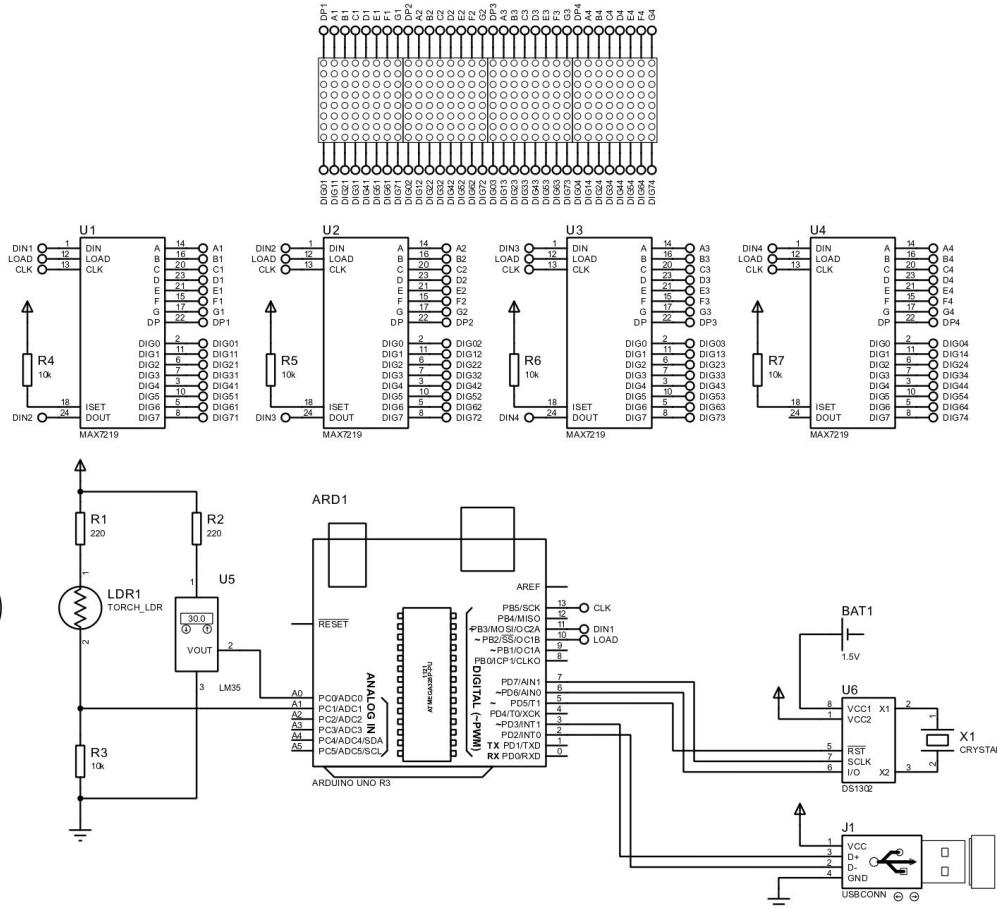
Alarm 3 : NRP dan Nama

Alarm 4 : Waktu on dan durasi (waktu off)

Alarm 5 : Alarm 5 (atau sesuai entry jika menggunakan keyboard USB)

3. Pada setiap detik ke 10 dan 40 setiap menit menampilkan suhu.
4. Kecerahan tampilan menyesuaikan kecerahan cahaya di sekitarnya.

C. Rangkaian Alat



D. Kode Program

```
#include <MD_MAX72xx.h>
#include <MD_Parola.h>
#include <virtuabotixRTC.h>
#include <PS2Keyboard.h>

#define HARDWARE_TYPE MD_MAX72XX::FC16_HW
#define MAX_DEVICES 4
#define MAX_CLK_PIN 13 // DOT MATRIX SCK
#define MAX_DATA_PIN 11 // DOT MATRIX MOSI
#define MAX_CS_PIN 10 // DOT MATRIX SS
#define RTC_CLK_PIN 7 // RTC CLK
#define RTC_DAT_PIN 6 // RTC DAT
#define RTC_RST_PIN 5 // RTC RST
#define DATA_PIN 3 // KEYBOARD DATA
#define IRQ_PIN 2 // KEYBOARD IRQ

MD_Parola P(HARDWARE_TYPE, MAX_CS_PIN, MAX_DEVICES);
virtuabotixRTC myrtc =
virtuabotixRTC(RTC_CLK_PIN, RTC_DAT_PIN, RTC_RST_PIN);
```

```
PS2Keyboard KB;

String hours;
String minutes;
float ldr;
float temperature;
float sensorValue;
unsigned long temptot = 0;

String temp_alarm_hours;
String temp_alarm_minutes;
int temp_alarm_duration = 5;
int selected_alarm;

String alarm_1_hours;
String alarm_2_hours;
String alarm_3_hours;
String alarm_4_hours;
String alarm_5_hours;
String alarm_1_minutes;
String alarm_2_minutes;
String alarm_3_minutes;
String alarm_4_minutes;
String alarm_5_minutes;

String temp_string;
char buf[99] = {" "};

int state = 1;

int current_menu = 1;
int current_alarm_menu = 1;

int current_active_alarm = 0;

int current_alarm_on_off_menu = 1;

bool is_setting_hour = false;
bool is_setting_string = false;

struct alarm {
    bool active;
    int hours;
    int minutes;
    int duration;
};

void Display(String str)
```

```

{
    Serial.println(str);
    str.toCharArray(buf, 99);
    P.displayReset();
    P.setTextBuffer(buf);
    P.displayAnimate();
}

void Clear()
{
    P.displayReset();
    P.displayClear();
}

void DisplayAnimate(String alarm, String str)
{
    Serial.println(alarm);
    P.setTextEffect(PA_SCROLL_LEFT, PA_SCROLL_LEFT );
    P.setSpeed(50);
    P.setPause(500);
    str.toCharArray(buf, 99);
    P.setTextBuffer(buf);
    if (P.displayAnimate()) {
        Clear();
    }
}

alarm alarm1 = {false, 0, 0, 0};
alarm alarm2 = {false, 0, 0, 0};
alarm alarm3 = {false, 0, 0, 0};
alarm alarm4 = {false, 0, 0, 0};
alarm alarm5 = {false, 0, 0, 0};
alarm alarms[5] = {alarm1, alarm2, alarm3, alarm4, alarm5};

unsigned long cur_time;
unsigned long prev_time;

void StateAlarm()
{
    alarms[selected_alarm - 1].hours = temp_alarm_hours.toInt();
    alarms[selected_alarm - 1].minutes = temp_alarm_minutes.toInt();
    alarms[selected_alarm - 1].duration = temp_alarm_duration;
    alarms[selected_alarm - 1].active = true;
}

void setup()
{
    Serial.begin(9600);
}

```

```

P.begin();
P.displayClear();
P.setCharSpacing(1);
attachInterrupt(digitalPinToInterruption(DATA_PIN),
interruptHandler, FALLING);
KB.begin(DATA_PIN, IRQ_PIN, PS2Keymap_US);
}

void loop()
{
    P.setTextAlignment(PA_CENTER);
    switch (state) {
        case 1:
            myrtc.updateTime();
            // cek alarm
            for (int i = 0; i < 5; i++) {
                if (alarms[i].hours == myrtc.hours && alarms[i].minutes ==
myrtc.minutes && alarms[i].active) {
                    state = 7;
                    current_active_alarm = i + 1;
                    prev_time = millis();
                }
            }
            temptot = 0;
            for(int x=0; x<500 ; x++){
                temptot = temptot + analogRead(A1);
            }
            sensorValue = temptot/500; //calculating average
            temperature = (sensorValue/1024)*500;

            ldr = analogRead(A0);
            P.setIntensity(ldr / 69);

            if (String(myrtc.hours).length() > 1) {
                hours = String(myrtc.hours);
            } else {
                hours = "0" + String(myrtc.hours);
            }
            if (String(myrtc.minutes).length() > 1) {
                minutes = String(myrtc.minutes);
            } else {
                minutes = "0" + String(myrtc.minutes);
            }
            if (10 <= myrtc.seconds && myrtc.seconds < 15 || 40 <=
myrtc.seconds && myrtc.seconds < 45 ) {
                Display((String(temperature) + "C"));
                delay(400);
                temptot = 0;
            }
    }
}

```

```

} else {
    Display((hours + " " + minutes));
    delay(500);
    Display ((hours + ":" + minutes));
    delay(500);
}
break;
case 2:
switch (current_menu) {
case 1:
    DisplayAnimate("",String("Set Time"));
    break;
case 2:
    DisplayAnimate("",String("Set Alarm"));
    break;
}
break;
case 3:
if (hours.length() == 1) {
    hours = "0" + hours;
}
if (minutes.length() == 1) {
    minutes = "0" + minutes;
}
if (is_setting_hour) {
    Display((hours + ":" + minutes));
    delay(500);
    Display(("    :" + minutes));
    delay(500);
    break;
}
else {
    Display((hours + ":" + minutes));
    delay(500);
    Display((hours + ":    "));
    delay(500);
    break;
}
}
case 4:
switch (current_alarm_menu) {
case 1:
    Display(String("1"));
    break;
case 2:
    Display(String("2"));
    break;
case 3:
    Display(String("3"));
    break;
}

```

```

    case 4:
        Display(String("4"));
        break;
    case 5:
        Display(String("5"));
        break;
    }
    break;
case 5:
    if (temp_alarm_hours.length() == 1) {
        temp_alarm_hours = "0" + temp_alarm_hours;
    }
    if (temp_alarm_minutes.length() == 1) {
        temp_alarm_minutes = "0" + temp_alarm_minutes;
    }
    if (is_setting_hour) {
        Display((temp_alarm_hours + ":" + temp_alarm_minutes));
        delay(500);
        Display(("   :" + temp_alarm_minutes));
        delay(500);
        break;
    } else {
        Display((temp_alarm_hours + ":" + temp_alarm_minutes));
        delay(500);
        Display((temp_alarm_hours + ":   "));
        delay(500);
        break;
    }
    break;
case 6:
    Display((String(temp_alarm_duration) + " s"));
    break;
case 7:
    if (millis() - prev_time > (alarms[current_active_alarm - 1].duration * 1000)) {
        alarms[current_active_alarm - 1].active = false;
        state = 1;
    } else {
        switch (current_active_alarm) {
            case 1:
                DisplayAnimate("alarm 1",String("07211940000016"));
                break;
            case 2:
                DisplayAnimate("alarm 2",String("Muhammad Zakariya Nur Ramdhhani"));
                break;
            case 3:
                DisplayAnimate("alarm 3",String("Muhammad Zakariya Nur

```

```

Ramdhani | 0721194000016"));
    break;
    case 4:
        for(int x = alarms[current_active_alarm - 1].duration; x
> 0 ;x--){
            Display((String(x) + " s"));
            delay(1000);
        }
        break;
    case 5:
        DisplayAnimate("alarm 5",String(temp_string));
        break;
    }
}
break;
case 8:
    DisplayAnimate("",String(temp_string));
    break;
case 9:
    if (current_alarm_on_off_menu == 1) {
        Display(String("On"));
    } else {
        Display(String("Off"));
    }
    break;

}
P.setTextEffect(PA_PRINT, PA_NO_EFFECT);
Serial.println(state);
}

void interruptHandler() {
    switch (state) {
        case 1:
            if (KB.available()) {
                char c = KB.read();
                if (c == PS2_ENTER) {
                    state = 2;
                }
            }
            break;
        case 2:
            switch (current_menu) {
                case 1:
                    if (KB.available()) {
                        char c = KB.read();
                        if (c == PS2_LEFTARROW || c == PS2_RIGHTARROW) {
                            current_menu = 2;
                        }
                    }
            }
    }
}

```

```

        } else if (c == PS2_ENTER) {
            state = 3;
            is_setting_hour = true;
        } else if (c == PS2_ESC) {
            state = 1;
        }
    }
case 2:
    if (KB.available()) {
        char c = KB.read();
        if (c == PS2_LEFTARROW || c == PS2_RIGHTARROW) {
            current_menu = 1;
        } else if (c == PS2_ENTER) {
            state = 4;
        } else if (c == PS2_ESC) {
            state = 1;
            current_menu = 1;
        }
    }
    break;
}
break;
case 3:
    if (KB.available()) {
        char c = KB.read();
        if (c == PS2_ESC) {
            state = 2;
        } else if (c == PS2_LEFTARROW || c == PS2_RIGHTARROW) {
            is_setting_hour = !is_setting_hour;
        } else if (c == PS2_UPARROW) {
            if (is_setting_hour) {
                if (hours.toInt() < 23 && hours.toInt() >= 0) {
                    hours = String(hours.toInt() + 1);
                } else if (hours.toInt() == 23) {
                    hours = String(0);
                }
            } else {
                if (minutes.toInt() < 59 && minutes.toInt() >= 0) {
                    minutes = String(minutes.toInt() + 1);
                } else if (minutes.toInt() == 59) {
                    minutes = String(0);
                }
            }
        }
    } else if (c == PS2_DOWNARROW) {
        if (is_setting_hour) {
            if (hours.toInt() <= 23 && hours.toInt() > 0) {
                hours = String(hours.toInt() - 1);
            } else if (hours.toInt() == 0) {

```

```

        hours = String(23);
    }
} else {
    if (minutes.toInt() <= 59 && minutes.toInt() > 0) {
        minutes = String(minutes.toInt() - 1);
    } else if (minutes.toInt() == 59) {
        minutes = String(59);
    }
}
} else if (c == PS2_ENTER) {
    myrtc.setDS1302Time(00, minutes.toInt(), hours.toInt(),
1, 4, 6, 2001);
    state = 1;
    is_setting_hour = true;
}
}
break;
case 4:
if (KB.available()) {
char c = KB.read();
if (c == PS2_LEFTARROW) {
    if (current_alarm_menu == 1) {
        current_alarm_menu = 5;
    } else {
        current_alarm_menu -= 1;
    }
} else if (c == PS2_RIGHTARROW) {
    if (current_alarm_menu == 5) {
        current_alarm_menu = 1;
    } else {
        current_alarm_menu += 1;
    }
} else if (c == PS2_ENTER) {
    state = 5;
    temp_alarm_hours = hours;
    temp_alarm_minutes = minutes;
    selected_alarm = current_alarm_menu;
} else if (c == PS2_ESC) {
    state = 3;
    current_alarm_menu = 1;
}
}
break;
case 5:
if (KB.available()) {
char c = KB.read();
if (c == PS2_ESC) {
    state = 4;
}
}
}

```

```

    } else if (c == PS2_LEFTARROW || c == PS2_RIGHTARROW) {
        is_setting_hour = !is_setting_hour;
    } else if (c == PS2_UPARROW) {
        if (is_setting_hour) {
            if (temp_alarm_hours.toInt() < 23 &&
temp_alarm_hours.toInt() >= 0) {
                temp_alarm_hours = String(temp_alarm_hours.toInt() +
1);
            } else if (temp_alarm_hours.toInt() == 23) {
                temp_alarm_hours = String(0);
            }
        } else {
            if (temp_alarm_minutes.toInt() < 59 &&
temp_alarm_minutes.toInt() >= 0) {
                temp_alarm_minutes =
String(temp_alarm_minutes.toInt() + 1);
            } else if (temp_alarm_minutes.toInt() == 59) {
                temp_alarm_minutes = String(0);
            }
        }
    } else if (c == PS2_DOWNARROW) {
        if (is_setting_hour) {
            if (temp_alarm_hours.toInt() <= 23 &&
temp_alarm_hours.toInt() > 0) {
                temp_alarm_hours = String(temp_alarm_hours.toInt() -
1);
            } else if (temp_alarm_hours.toInt() == 0) {
                temp_alarm_hours = String(23);
            }
        } else {
            if (temp_alarm_minutes.toInt() <= 59 &&
temp_alarm_minutes.toInt() > 0) {
                temp_alarm_minutes =
String(temp_alarm_minutes.toInt() - 1);
            } else if (temp_alarm_minutes.toInt() == 59) {
                temp_alarm_minutes = String(59);
            }
        }
    } else if (c == PS2_ENTER) {

        state = 6;
        is_setting_hour = true;
    }
}
break;
case 6:
if (KB.available()) {
    char c = KB.read();

```

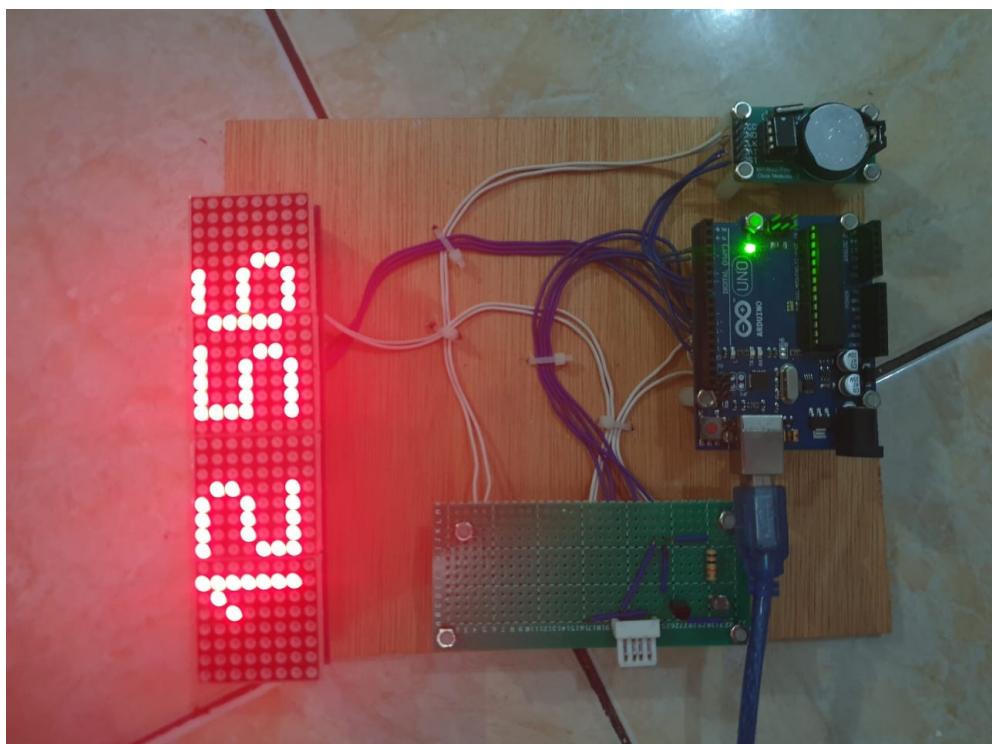
```

    if (c == PS2_UPARROW) {
        temp_alarm_duration += 1;
    } else if (c == PS2_DOWNARROW && temp_alarm_duration > 1)
    {
        temp_alarm_duration -= 1;
    } else if (c == PS2_ESC) {
        state = 5;
    } else if (c == PS2_ENTER) {
        if (selected_alarm == 5) {
            temp_string = "";
            state = 8;
        } else {
            state = 1;
            StateAlarm();
        }
    }
}
break;
case 7 :
if (KB.available()) {
    char c = KB.read();
    if (c == PS2_ENTER) {
        state = 1;
        alarms[current_active_alarm - 1].active = false;
    }
}
case 8 :
if (KB.available()) {
    char c = KB.read();
    if (c == PS2_ENTER) {
        StateAlarm();
        state = 1;
    } else {
        temp_string = temp_string + String(c);
        Clear();
    }
}
break;
case 9:
if (KB.available()) {
    char c = KB.read();
    if (c == PS2_LEFTARROW || c == PS2_RIGHTARROW) {
        if (current_alarm_on_off_menu == 1) {
            current_alarm_on_off_menu = 2;
        } else {
            current_alarm_on_off_menu = 1;
        }
    } else if (c == PS2_ENTER) {

```

```
    if (current_alarm_on_off_menu == 1) {
        state = 6;
    } else {
        state = 1;
        current_alarm_on_off_menu = 1;
        alarms[selected_alarm - 1].active = false;
    }
}
break;
}
}
```

E. Hasil Percobaan



BAB IV

KESIMPULAN

Berdasar tugas yang telah saya lakukan, dapat ditarik kesimpulan bahwa perangkat dapat bekerja dengan baik. Perangkat dapat menampilkan jam dan menit. Pada detik ke 10 dan 40 perangkat dapat menampilkan suhu sekitar. Kemudian kecerahan LED Dot Matrix dapat berubah sesuai kecerahan lingkungan sekitar. Alarm juga dapat berfungsi dengan baik dan dapat menampilkan teks sesuai dengan spesifikasi yang telah ditentukan. Pada alarm 5 juga dapat mengganti teks yang ingin ditampilkan.

DAFTAR PUSTAKA

- Mirza, Y., & Firdaus, A. (2016). LIGHT DEPENDENT RESISTANT (LDR) SEBAGAI PENDETEKSI WARNA. *JUPITER (Jurnal Penelitian Ilmu Dan Teknologi Komputer)*, 8(1), 39–45. <https://doi.org/10.5281/zenodo.3429349>
- Nataprawira, A., Rizal, A. and Wibowo, A., 2020. *Perancangan Display Led Dot Matrix Via Wi-Fi Menggunakan Aplikasi Mobile Android*. [online] Journal.unbara.ac.id. Available at: <<https://journal.unbara.ac.id/index.php/INTECH/article/view/240>> [Accessed 23 April 2022].
- Pradana, Rahman Bayu. 2017. *SISTEM KEAMANAN RUMAH DENGAN PEMBERITAHUAN MELALUI SMS BERBASIS ARDUINO*. Diploma thesis, STMIK AKAKOM Yogyakarta.
- Sutono, S. & Nursoparisa, A., 2020. Perancangan Sistem Kendali automatisasi control debit air Pada Pengisian galon menggunakan modul arduino. *Media Jurnal Informatika*, 11(1), p.33. Available at: https://www.researchgate.net/publication/343743073_Perancangan_Sistem_Kendali_Automatisasi_Control_Debit_Air_pada_Pengisian_Galon_Menggunakan_Modul_Arduino [Accessed April 26, 2022].