**LAPORAN AKHIR**

**FINAL PROJECT SISTEM TERTANAM**

**Icon

Description automatically generated**

**Disusun oleh:**

**Iqbal Muchlis 5024201073**

**Dosen Pembimbing:**

**Eko Pramunanto, S.T., M.T.**

**FAKULTAS TEKNOLOGI ELEKTRO DAN INFORMATIKA CERDAS**

**TEKNIK KOMPUTER**

**INSTITUT TEKNOLOGI SEPULUH NOPEMBER**

**SURABAYA**

**2023**

1. **PENJELASAN TUGAS**

Rancanglah sistem untuk menampilkan jam, kalender, suhu dan alarm dengan karakter pada dot matrix LED 8 x 32 (4 buah matrix LED 8x8), dengan input keyboard USB. Suhu diukur dengan menggunakan sensor suhu analog (seperti LM35 atau yang sejenisnya).

Spesifikasi fitur:

a. mode : run, set jam, set tanggal, set alarm.

b. select set : jam, menit, detik / tgl, bln, thn,

c. 3 waktu alarm dengan text (wajib) + buzzer (optional nilai plus):

- alarm 1: Display NRP

- alarm 2: Display NRP + Nama

- alarm 3: Display text yang diinputkan

Waktu aktif untuk alarm 1, 2 dan 3 bisa diset dengan tanggal, jam, menit dan durasinya dalam detik.

d. Tampilan:

- Kecerahan diatur sesuai kecerahan lingkungan dengan sensor cahaya.

- Jam, menit, detik

- Pada setiap detik ke 10 dan 40 tampilkan tgl-bln-thn selama 3 detik

- Pada setiap detik ke 13 dan 43 tampilkan suhu dengan keterangan ͦ C (derajad Celcius)

- Pada saat waktu tepat sama dengan waktu seting alarm tampilkan text alarm sesuai durasi setingnya.

- Kecepatan geser tampilan text panjang (running text) adalah 0.5 detik per kolom matrix LED.

- Sebagai tambahan nilai : tampilan berkedip saat berada pada mode seting (input dari keyboard).

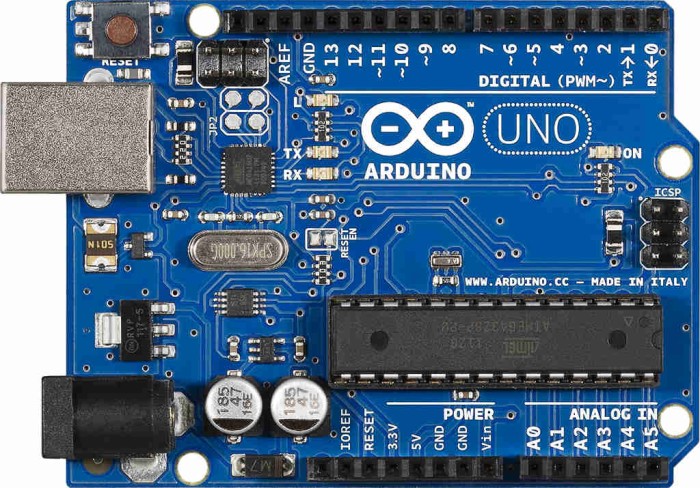
Output:

- 8x32 dot matrix LED

1. **ALAT DAN BAHAN**

Alat dan bahan yang saya pakai untuk final project ini adalah:

* Arduino UNO



* Keyboard PS/2 Protocol



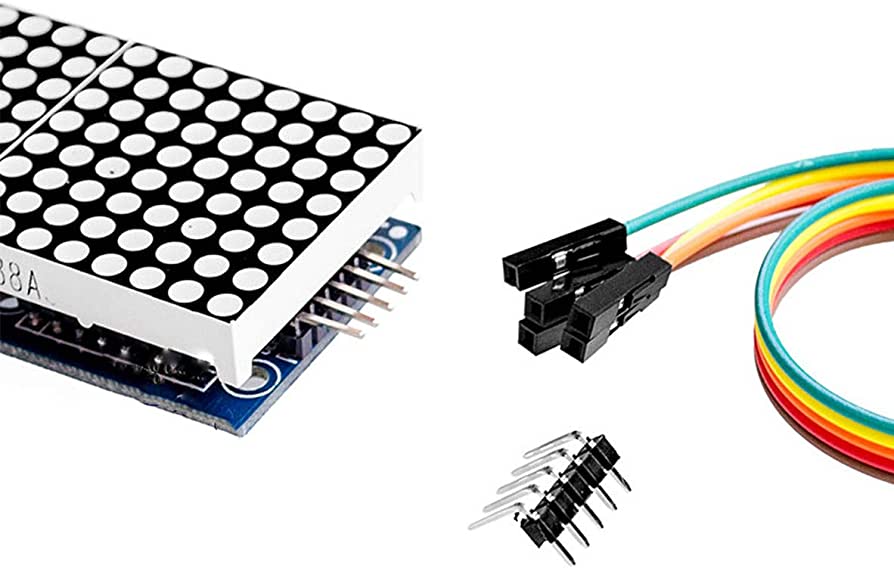
* Female PS/2 Socket



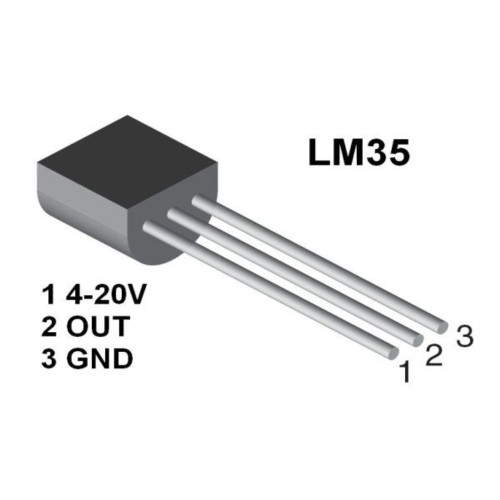
* Active Buzzer



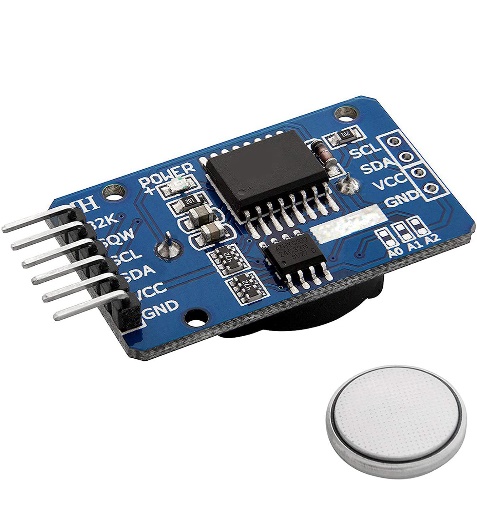
* MAX7219 LED Dot Matrix Module 4-IN-1 32x8



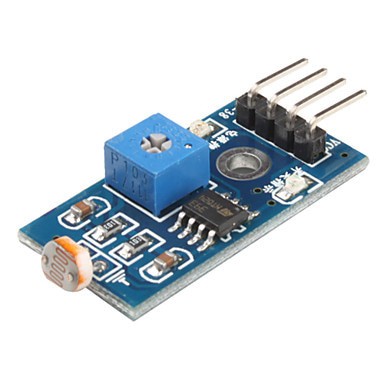
* LM35



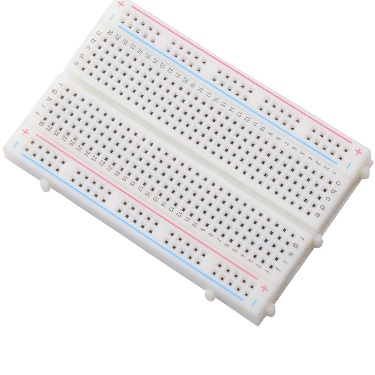
* Real time clock (DS3232)



* LDR (Light Sensor Module)



* Breadboard (Untuk bereksperimen)



* PCB

A group of green circuit boards

Description automatically generated with low confidence

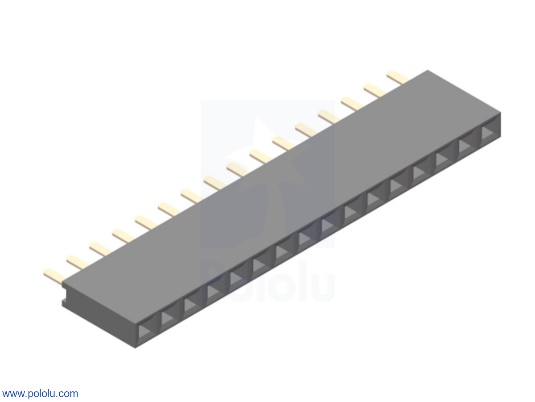
* Solder dan timahnya



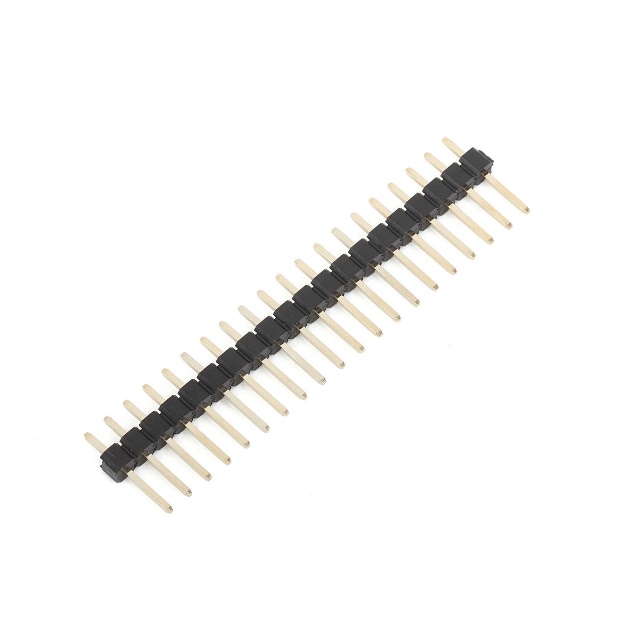
* Kaki PCB



* Female to Male Pin Header



* Male to Male Pin Header

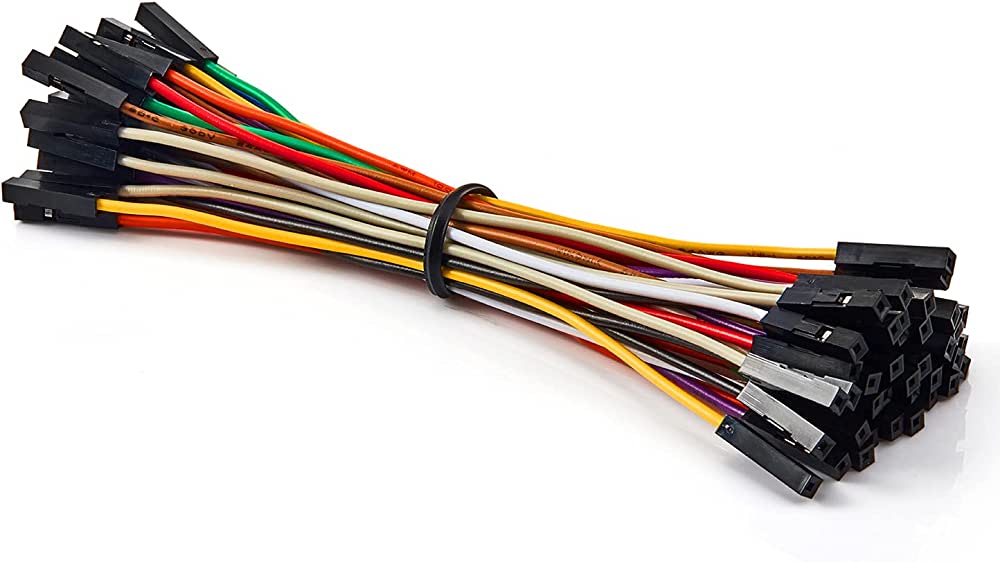


* Kabel

- Male to female



- Female to female



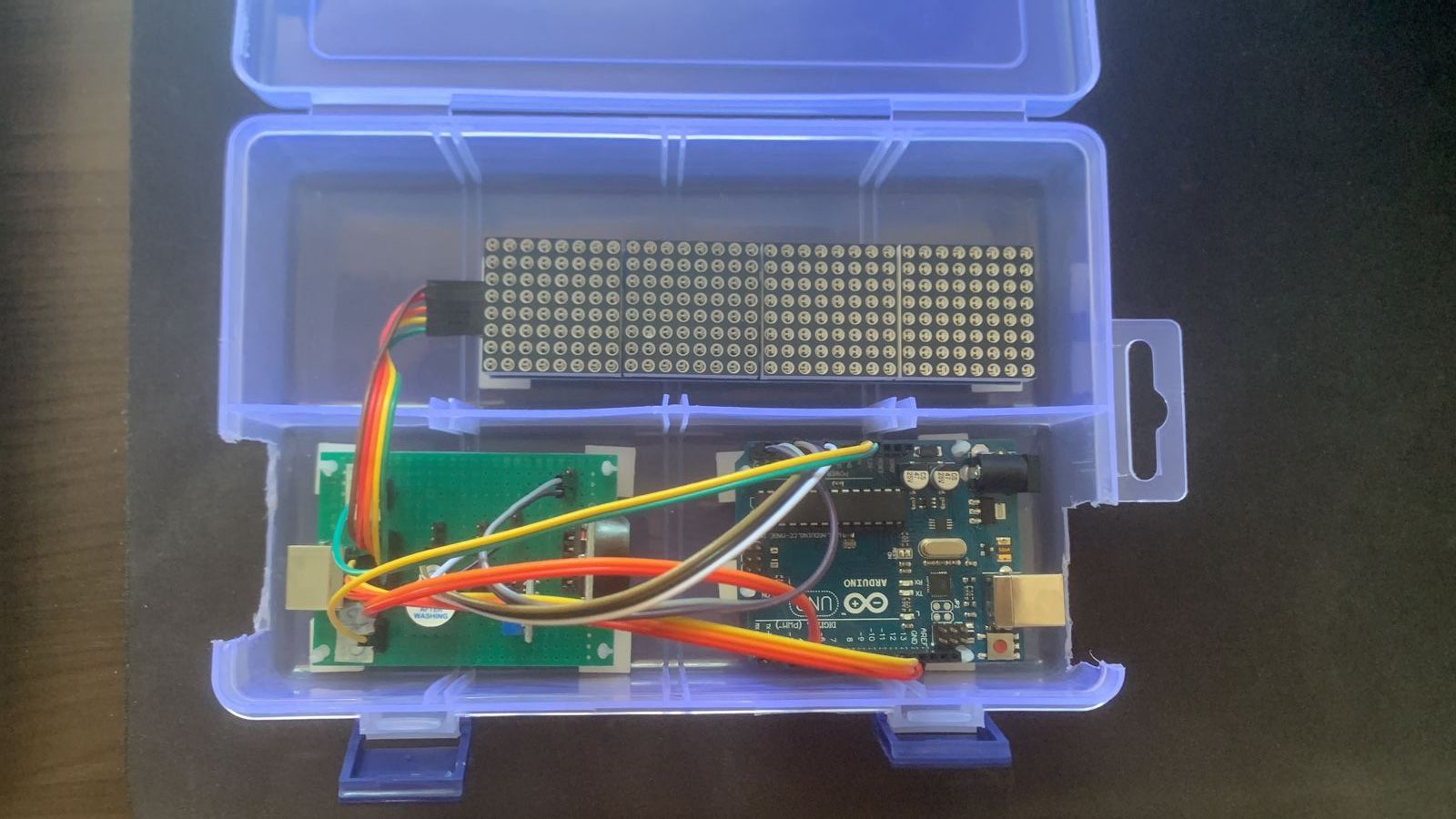
- Male to male (untuk bereksperimen)



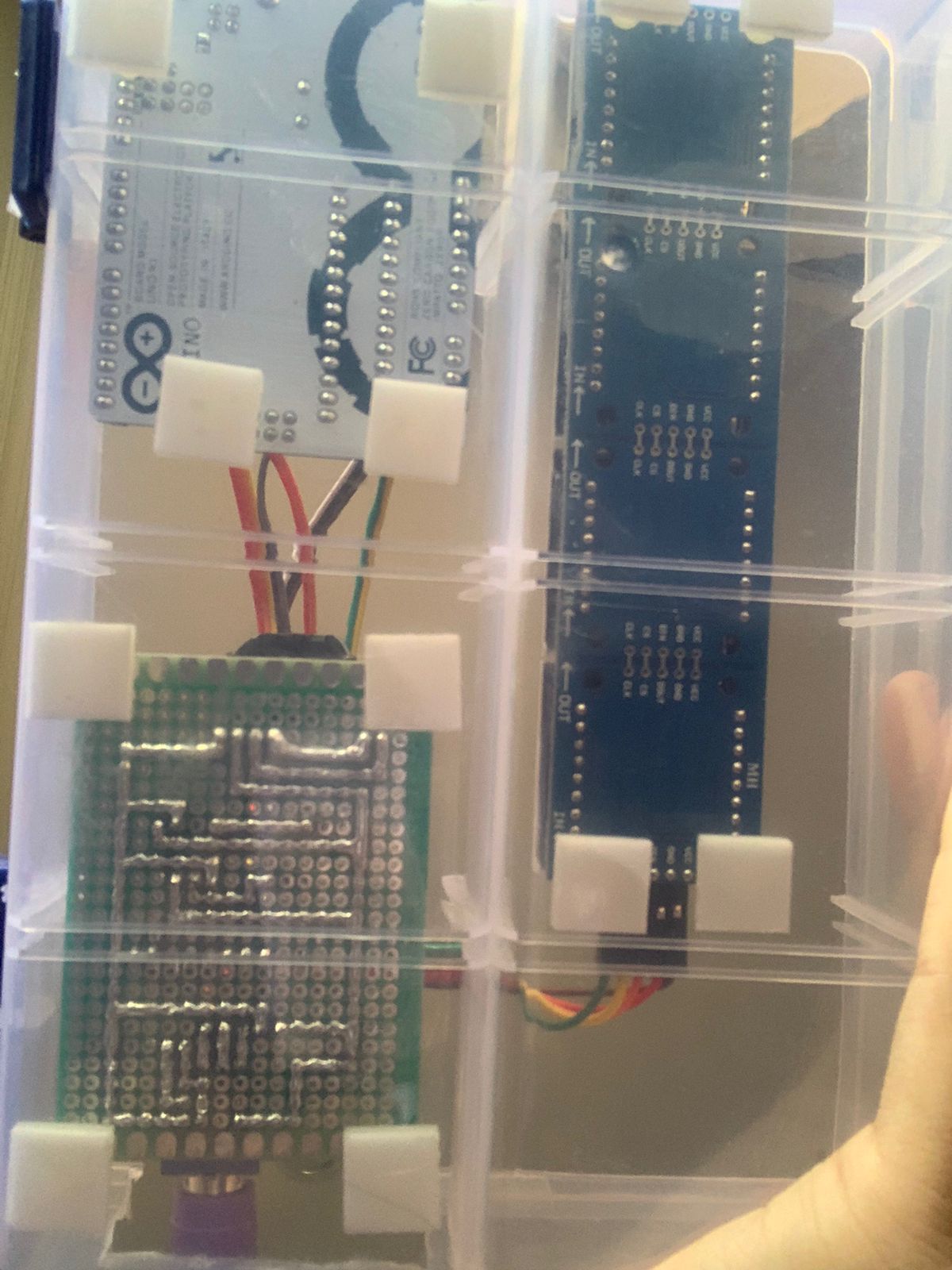
1. **RANGKAIAN**

Berikut merupakan rangkaian yang telah saya rancang:

- Tampak atas



- Tampak bawah



Detail rangkaian:

* Dot matrix

- VCC disambungkan pada 5v di arduino

- GND disambungkan pada GND di arduino

- DIN disambungkan pada pin 11 di arduino

- CS disambungkan pada pin 10 di arduino

- CLK disambungkan pada pin 13 di arduino

* LDR

- VCC disambungkan pada 5v di arduino

- GND disambungkan pada GND di arduino

- A0 disambungkan pada A1 di arduino

* Active Buzzer

- Pin 1 disambungkan pada GND di arduino

- Pin 2 disambungkan pada 8 di arduino

* RTC

- VCC disambungkan pada 5v di arduino

- GND disambungkan pada GND di arduino

- SDA disambungkan pada A4 di arduino

- SCL disambungkan pada A5 di arduino

* LM35

- 4-20V disambungkan pada 5v di arduino

- GND disambungkan pada GND di arduino

- OUT disambungkan pada A0 di arduino

* Female PS/2 Socket

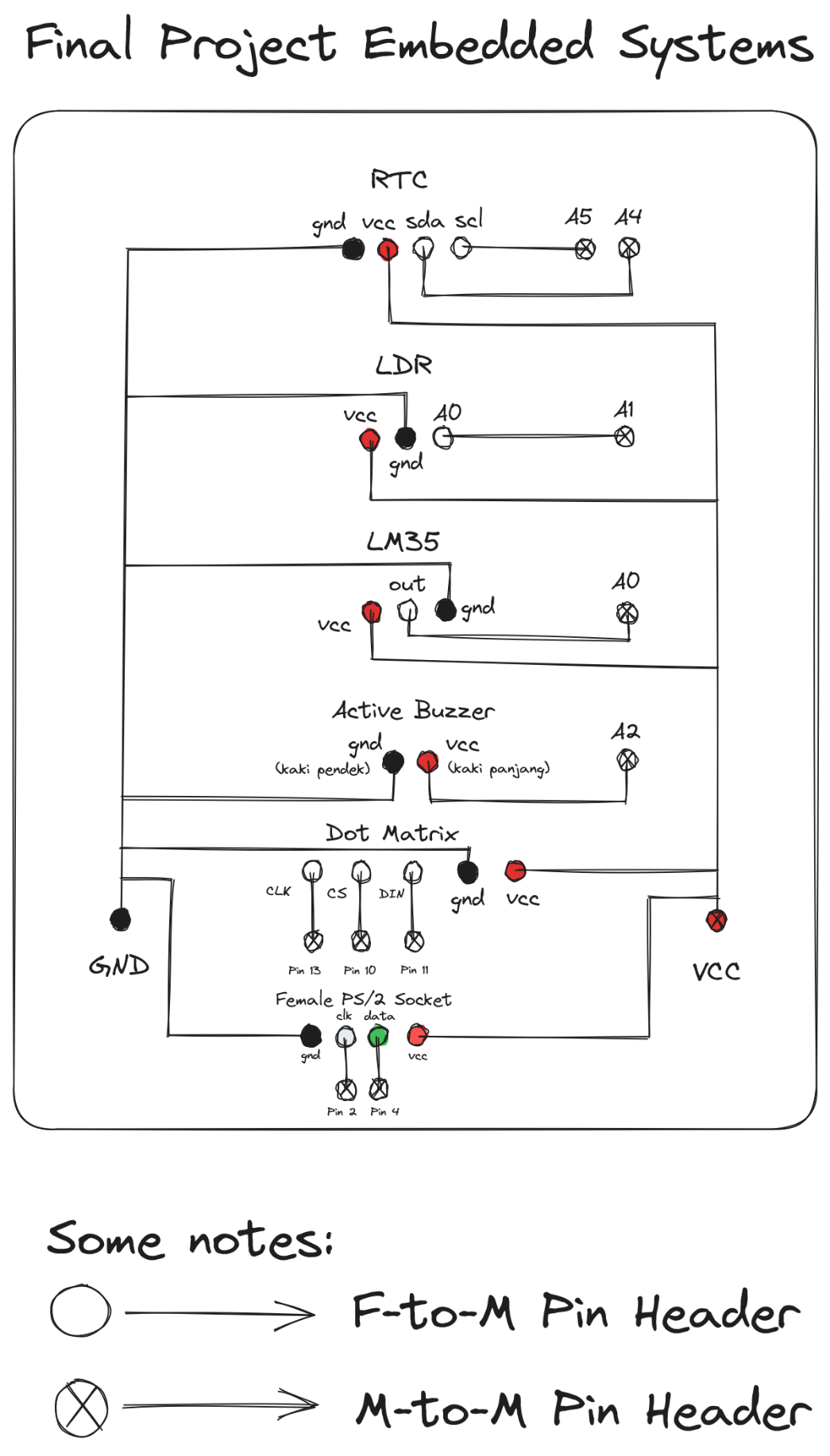
- VCC disambungkan pada VCC di arduino

- GND disambungkan pada GND di arduino

- CLK disambungkan pada Pin 2 di arduino

- DATA disambungkan pada Pin 4 di arduino

Rancangan pada PCB:



1. **HASIL**

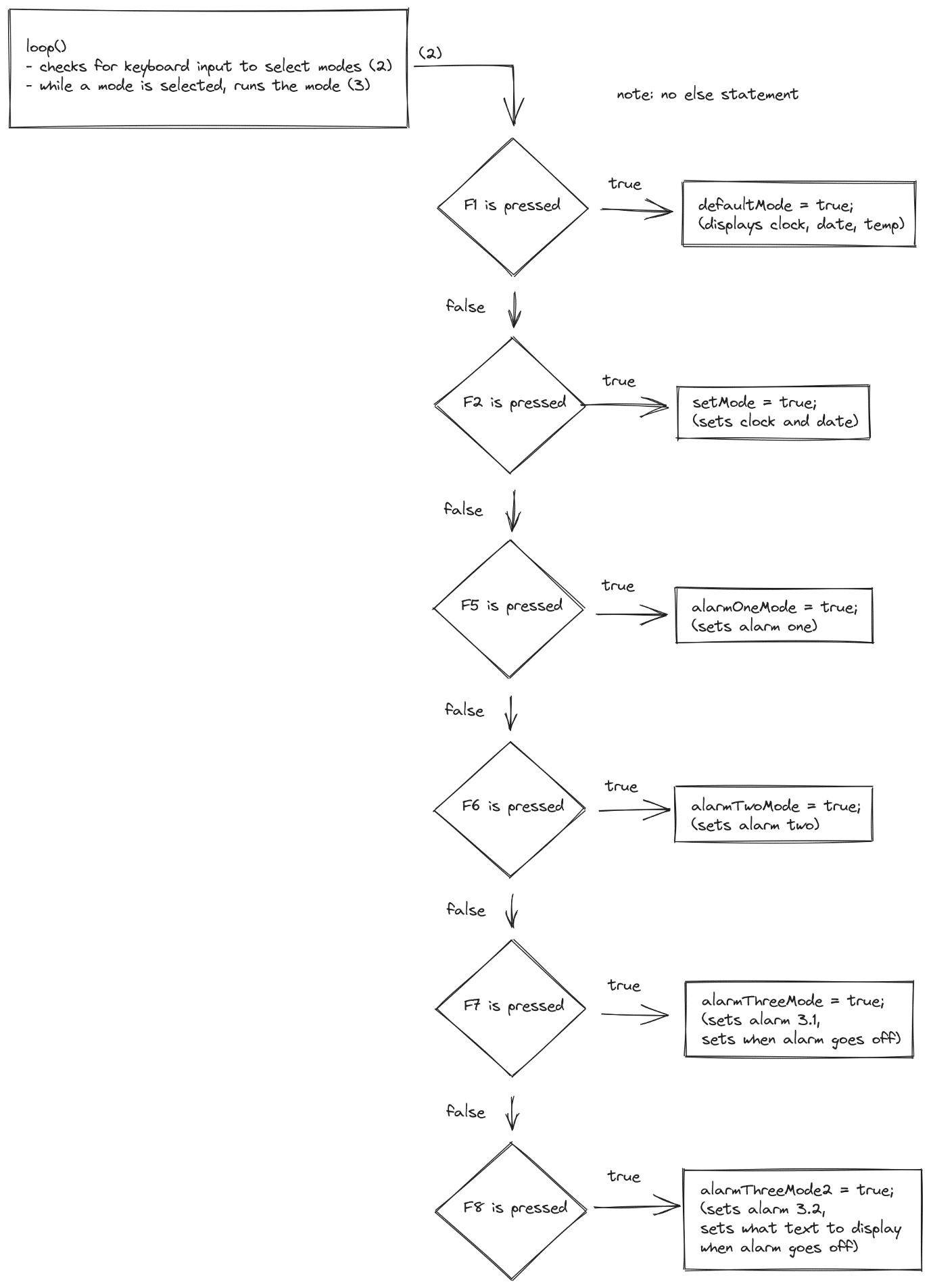
- Diagram blok:

- Part 1:

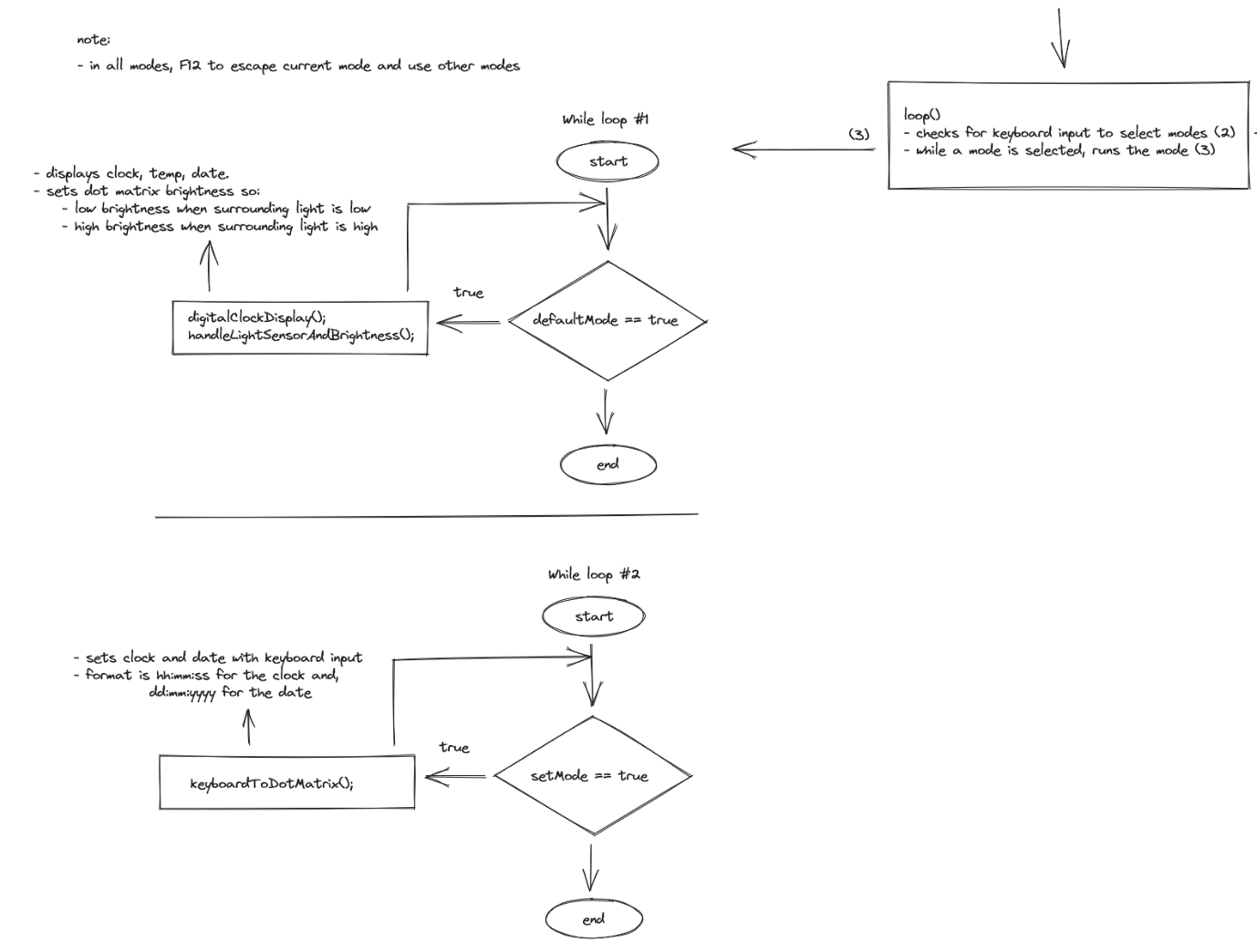
A picture containing text, diagram, sketch, drawing

Description automatically generated

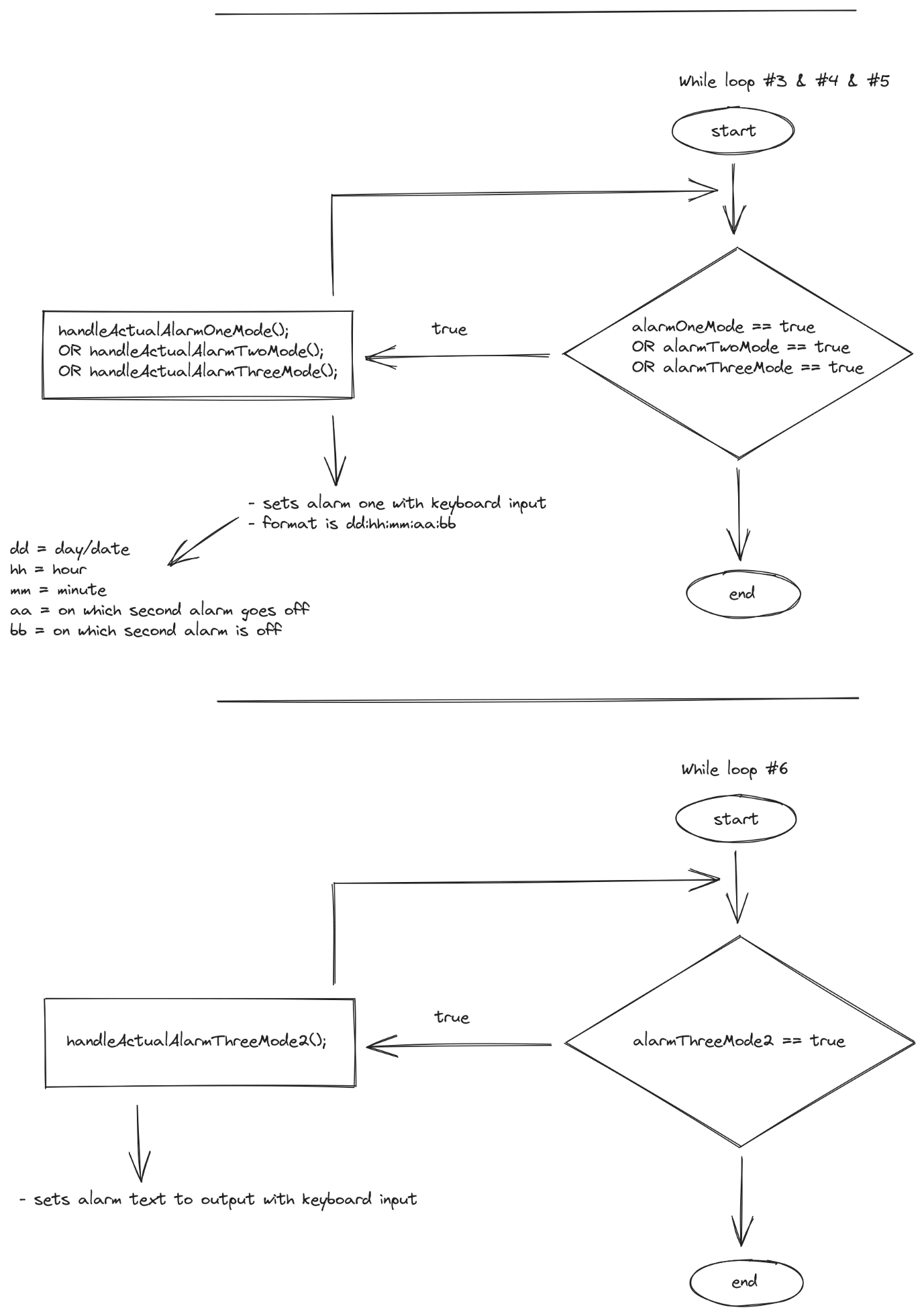
- Part 2:



- Part 3



- Part 4



- Dengan menggunakan library:

* Adafruit\_GFX.h (<https://github.com/adafruit/Adafruit-GFX-Library>)

- untuk mengatur display dot matrix.

* Fonts/Picopixel.h

- untuk mengatur font pada dot matrix

* Max72xxPanel.h (<https://github.com/markruys/arduino-Max72xxPanel>)

- helper library untuk menjambati agar Adafruit\_GFX.h dapat dipakai untuk mengatur display dot matrix.

* DS3232RTC (<https://github.com/JChristensen/DS3232RTC>)

- untuk mengatur real time clock

* LM35 (<https://github.com/wilmouths/LM35>)

- untuk mengatur sensor temperature agar lebih mudah

* PS2Keyboard.h (<https://github.com/PaulStoffregen/PS2Keyboard>)

- untuk memproses input keyboard.

* SPI.h

- untuk fungsionalitas library yang lain.

- Hasil:

* dapat melakukan semua yang diminta dari spesifikasi fitur pada detail tugas kecuali:

- berkedip saat mode input

- Code:

// how to operate:

// there are 5 different modes,

//

// - (press F1) displays clock, temp, and date mode

// - (press F2) set clock, date mode

//   - here, you can:

//     - ESC to remove/reset output

//     - BACKSPACE to remove output one by one

//     - ENTER to set clock IF format is correct (6 numbers, hh:mm:ss)

//     - ']' to set date IF format is correct (6 numbers, d:m:yyyy)

// - (press F5) set and displpay alarm one mode

// - (press F6) set and displpay alarm two mode

// - (press F7) set and displpay alarm 3.1 mode

// - (press F8) set and displpay alarm 3.2 mode

//     - ']' to set alarm text to be displayed

// - for F5, F6, and F7

//     - ENTER to set when alarm happens IF format is correct (10 numbers, dd:hh:mm:aa:bb, aa = on which second alarm goes off; bb = on which second alarm is off)

// note:

// - in all modes, F12 to escape current mode and use other modes

//

// bugs:

// - when setting clock/date, the length of output permissible shouldn't be more than 6 (since the format have 6 digits (see -> hh:mm:ss))

// - etc

#include <SPI.h>

#include <Adafruit\_GFX.h>

#include <Max72xxPanel.h>

#include <PS2Keyboard.h>

#include <Fonts/Picopixel.h>

#include <LM35.h>

#include <DS3232RTC.h>

// temp sensor

LM35 lm35(A0);

// RTC

DS3232RTC rtc;

// light sensor

#define lightSensor A1

// buzzer

#define buzzer A2

// keyboard

#define DataPin 4

#define IRQpin 2

PS2Keyboard keyboard;

// dot matrix

// Attach CS to this pin, DIN to MOSI and CLK to SCK (cf http://arduino.cc/en/Reference/SPI )

#define pinCS 10

#define numberOfHorizontalDisplays 4

#define numberOfVerticalDisplays 1

Max72xxPanel matrix = Max72xxPanel(pinCS, numberOfHorizontalDisplays, numberOfVerticalDisplays);

// str for displaying keyboard input to dot matrix

char tempStr[50] = "";

char str[50] = "";

// for changing between modes

bool defaultMode;

bool setMode;

bool alarmOneMode;

bool alarmTwoMode;

bool alarmThreeMode;

bool alarmThreeMode2;

// for setMode and first alarm

char enteredValueEnter[20] = "";

char enteredValueBracket[20] = "";

bool checkEnteredValueBracket;

bool checkEnteredValueEnter;

// for first alarm

String hourFromKeyboard;

String minuteFromKeyboard;

String secondFromKeyboard;

String dayFromKeyboard;

String monthFromKeyboard;

String yearFromKeyboard;

// for second alarm

String hourFromKeyboard2;

String minuteFromKeyboard2;

String secondFromKeyboard2;

String dayFromKeyboard2;

String monthFromKeyboard2;

String yearFromKeyboard2;

// for third alarm

String hourFromKeyboard3;

String minuteFromKeyboard3;

String secondFromKeyboard3;

String dayFromKeyboard3;

String monthFromKeyboard3;

String yearFromKeyboard3;

// for set alarm one and two and three

char alarmOne[30] = "1. 5024201073";

char alarmTwo[30] = "2. 5024201073 Iqbal Muchlis";

char alarmThree[50] = "";

char enteredValueEnterAlarm[20] = "";

bool checkAlarmOneStart;

bool checkAlarmTwoStart;

bool checkAlarmThreeStart;

bool checkEnteredValueEnterAlarmOne;

bool checkEnteredValueEnterAlarmTwo;

bool checkEnteredValueEnterAlarmThree;

String firstDurationFromKeyboard;  // alarm one

String secondDurationFromKeyboard;

String firstDurationFromKeyboard2;  // alarm two

String secondDurationFromKeyboard2;

String firstDurationFromKeyboard3;  // alarm two

String secondDurationFromKeyboard3;

void setup() {

  Serial.begin(9600);

  pinMode(buzzer, OUTPUT);

  keyboard.begin(DataPin, IRQpin);

  rtc.begin();

  // fixes dot matrix display orientation

  setMatrixDisplayOrientation();

  // set dot matrix brightness, font, size

  // matrix.setIntensity(0);

  matrix.setFont(&Picopixel);

  matrix.setTextColor(1);

  matrix.setTextSize(1);

  // matrix.setTextWrap(boolean w);

  // matrix.setTextColor(uint16\_t color, uint16\_t backgroundcolor);

  setSyncProvider(rtc.get);  // the function to get the time from the RTC

  if (timeStatus() != timeSet) {

    Serial.println("Unable to sync with the RTC");

  } else {

    Serial.println("RTC has set the system time");

  }

}

void loop() {

  // check for keyboard input

  if (!keyboard.available()) {

    return;

  }

  char c = keyboard.read();

  if (c == PS2\_F1) {  // default mode (clock, temp, date)

    strcpy(str, "");

    matrix.fillScreen(0);

    defaultMode = true;

    setMode = false;

    alarmOneMode = false;

    alarmTwoMode = false;

    alarmThreeMode = false;

  } else if (c == PS2\_F2) {  // set clock and date mode

    strcpy(str, "");

    matrix.fillScreen(0);

    defaultMode = false;

    setMode = true;

    alarmOneMode = false;

    alarmTwoMode = false;

    alarmThreeMode = false;

  } else if (c == PS2\_F5) {  // set alarm 1

    strcpy(str, "");

    matrix.fillScreen(0);

    defaultMode = false;

    setMode = false;

    alarmOneMode = true;

    alarmTwoMode = false;

    alarmThreeMode = false;

  } else if (c == PS2\_F6) {  // set alarm 2

    strcpy(str, "");

    matrix.fillScreen(0);

    defaultMode = false;

    setMode = false;

    alarmOneMode = false;

    alarmTwoMode = true;

    alarmThreeMode = false;

  } else if (c == PS2\_F7) {  // set alarm 3

    strcpy(str, "");

    matrix.fillScreen(0);

    defaultMode = false;

    setMode = false;

    alarmOneMode = false;

    alarmTwoMode = false;

    alarmThreeMode = true;

  } else if (c == PS2\_F8) {  // set alarm 3.2

    strcpy(str, "");

    matrix.fillScreen(0);

    defaultMode = false;

    setMode = false;

    alarmOneMode = false;

    alarmTwoMode = false;

    alarmThreeMode = false;

    alarmThreeMode2 = true;

  }

  while (defaultMode == true) {

    digitalClockDisplay();

    handleLightSensorAndBrightness();

  }

  while (setMode == true) {

    keyboardToDotMatrix();

  }

  while (alarmOneMode == true) {

    handleActualAlarmOneMode();

  }

  while (alarmTwoMode == true) {

    handleActualAlarmTwoMode();

  }

  while (alarmThreeMode == true) {

    handleActualAlarmThreeMode();

  }

  while (alarmThreeMode2 == true) {

    handleActualAlarmThreeMode2();

  }

}

void handleActualAlarmOneMode() {

  if (!keyboard.available()) {

    return;

  }

  char c = keyboard.read();

  // normal text

  matrix.setCursor(0, 6);

  if (strlen(tempStr) > 15) {

    strcpy(tempStr, "");

    matrix.fillScreen(0);

  } else if (c == PS2\_F12) {

    strcpy(str, "");

    matrix.fillScreen(0);

    alarmOneMode = false;

  } else if (c == PS2\_ESC) {

    // resets dotmatrix screen

    strcpy(tempStr, "");

    matrix.fillScreen(0);

  } else if (c == PS2\_DELETE) {

    // for backspace

    int len = strlen(tempStr);

    if (len > 0) {

      tempStr[len - 1] = '\0';

    }

    matrix.fillScreen(0);

  } else if (c == PS2\_ENTER) {

    checkAlarmOneStart = true;

    handleSetClockOrDate("enteredValueEnterAlarm");

  } else if (c >= '0' && c <= '9') {

    // updates display based on keyboard input

    strncat(tempStr, &c, 1);

  } else {

    // else if other characters are pressed

    // do nothing

  }

  strcpy(str, "1:");  // Copy the first part of the string into 'str'

  strcat(str, tempStr);

  // matrix.print("°");

  // Serial.println(str);

  matrix.print(str);

  matrix.write();

}

void handleActualAlarmTwoMode() {

  if (!keyboard.available()) {

    return;

  }

  char c = keyboard.read();

  // normal text

  matrix.setCursor(0, 6);

  if (strlen(tempStr) > 15) {

    strcpy(tempStr, "");

    matrix.fillScreen(0);

  } else if (c == PS2\_F12) {

    strcpy(str, "");

    matrix.fillScreen(0);

    alarmTwoMode = false;

  } else if (c == PS2\_ESC) {

    // resets dotmatrix screen

    strcpy(tempStr, "");

    matrix.fillScreen(0);

  } else if (c == PS2\_DELETE) {

    // for backspace

    int len = strlen(tempStr);

    if (len > 0) {

      tempStr[len - 1] = '\0';

    }

    matrix.fillScreen(0);

  } else if (c == PS2\_ENTER) {

    checkAlarmTwoStart = true;

    handleSetClockOrDate("enteredValueEnterAlarm");

  } else if (c >= '0' && c <= '9') {

    // updates display based on keyboard input

    strncat(tempStr, &c, 1);

  } else {

    // else if other characters are pressed

    // do nothing

  }

  strcpy(str, "2:");  // Copy the first part of the string into 'str'

  strcat(str, tempStr);

  // matrix.print("°");

  // Serial.println(str);

  matrix.print(str);

  matrix.write();

}

void handleActualAlarmThreeMode() {

  if (!keyboard.available()) {

    return;

  }

  char c = keyboard.read();

  // normal text

  matrix.setCursor(0, 6);

  if (strlen(tempStr) > 15) {

    strcpy(tempStr, "");

    matrix.fillScreen(0);

  } else if (c == PS2\_F12) {

    strcpy(str, "");

    matrix.fillScreen(0);

    alarmThreeMode = false;

  } else if (c == PS2\_ESC) {

    // resets dotmatrix screen

    strcpy(tempStr, "");

    matrix.fillScreen(0);

  } else if (c == PS2\_DELETE) {

    // for backspace

    int len = strlen(tempStr);

    if (len > 0) {

      tempStr[len - 1] = '\0';

    }

    matrix.fillScreen(0);

  } else if (c == PS2\_ENTER) {

    checkAlarmThreeStart = true;

    handleSetClockOrDate("enteredValueEnterAlarm");

  } else if (c == ']') {

    handleSetClockOrDate("enteredValueBracketAlarm3");

  } else if (c >= '0' && c <= '9') {

    // updates display based on keyboard input

    strncat(tempStr, &c, 1);

  } else {

    // else if other characters are pressed

    // do nothing

  }

  strcpy(str, "3.1:");  // Copy the first part of the string into 'str'

  strcat(str, tempStr);

  // matrix.print("°");

  // Serial.println(str);

  matrix.print(str);

  matrix.write();

}

void handleActualAlarmThreeMode2() {

  if (!keyboard.available()) {

    return;

  }

  char c = keyboard.read();

  // normal text

  matrix.setCursor(0, 6);

  if (strlen(tempStr) > 15) {

    strcpy(tempStr, "");

    matrix.fillScreen(0);

  } else if (c == PS2\_F12) {

    strcpy(str, "");

    matrix.fillScreen(0);

    alarmThreeMode2 = false;

  } else if (c == PS2\_ESC) {

    // resets dotmatrix screen

    strcpy(tempStr, "");

    matrix.fillScreen(0);

  } else if (c == PS2\_DELETE) {

    // for backspace

    int len = strlen(tempStr);

    if (len > 0) {

      tempStr[len - 1] = '\0';

    }

    matrix.fillScreen(0);

  } else if (c == ']') {

    handleSetClockOrDate("enteredValueBracketAlarm3");

  } else {

    // eo nothing

    strncat(tempStr, &c, 1);

  }

  strcpy(str, "3.2:");  // Copy the first part of the string into 'str'

  strcat(str, tempStr);

  // matrix.print("°");

  // Serial.println(str);

  matrix.print(str);

  matrix.write();

}

void handleAlarmThreeMode() {

  // char c = keyboard.read();

  // matrix.setCursor(0, 6);

  // if (c == PS2\_F12) {

  //   strcpy(str, "");

  //   matrix.fillScreen(0);

  //   alarmThreeMode = false;

  // }

  // char alarmThreeTextOutput[30];

  // // String alarmThreeText = "i'm alarm 3";

  // // sprintf(alarmThreeTextOutput, "3. %s", alarmThreeText);

  // strcpy(str, "3. input from keyboard");

  // matrix.print(str);

  // matrix.write();

  char c = keyboard.read();

  if (c == PS2\_F12) {

    strcpy(str, "");

    matrix.fillScreen(0);

    alarmThreeMode = false;

  }

  strcpy(str, "3. input from keyboard");

  // running text right to left

  for (int i = 50; i >= -45; i--) {

    matrix.fillScreen(0);

    matrix.setCursor(i, 6);

    matrix.print(str);

    matrix.write();

    delay(55);

  }

}

void keyboardToDotMatrix() {

  if (!keyboard.available()) {

    return;

  }

  char c = keyboard.read();

  // normal text

  matrix.setCursor(0, 6);

  if (strlen(tempStr) > 15) {

    strcpy(tempStr, "");

    matrix.fillScreen(0);

  } else if (c == PS2\_F12) {

    strcpy(str, "");

    matrix.fillScreen(0);

    setMode = false;

  } else if (c == PS2\_ESC) {

    // resets dotmatrix screen

    strcpy(tempStr, "");

    matrix.fillScreen(0);

  } else if (c == PS2\_DELETE) {

    // for backspace

    int len = strlen(tempStr);

    if (len > 0) {

      tempStr[len - 1] = '\0';

    }

    matrix.fillScreen(0);

  } else if (c == PS2\_ENTER) {

    handleSetClockOrDate("enteredValueEnter");

  } else if (c == ']') {

    handleSetClockOrDate("enteredValueBracket");

  } else if (c >= '0' && c <= '9') {

    // updates display based on keyboard input

    strncat(tempStr, &c, 1);

  } else {

    // else if other characters are pressed

    // do nothing

  }

  if (checkEnteredValueEnter == true && checkEnteredValueBracket == true) {

    setTime(hourFromKeyboard.toInt(), minuteFromKeyboard.toInt(), secondFromKeyboard.toInt(), dayFromKeyboard.toInt(), monthFromKeyboard.toInt(), yearFromKeyboard.toInt());

    rtc.set(now());

    Serial.println("time set");

    hourFromKeyboard = "";

    minuteFromKeyboard = "";

    secondFromKeyboard = "";

    dayFromKeyboard = "";

    monthFromKeyboard = "";

    yearFromKeyboard = "";

    checkEnteredValueEnter = false;

    checkEnteredValueBracket = false;

  }

  strcpy(str, "s:");  // Copy the first part of the string into 'str'

  strcat(str, tempStr);

  // matrix.print("°");

  // Serial.println(str);

  matrix.print(str);

  matrix.write();

}

void digitalClockDisplay() {

  uint16\_t celciusTemp = lm35.getTemp(CELCIUS);

  char c = keyboard.read();

  if (c == PS2\_F12) {

    strcpy(str, "");

    matrix.fillScreen(0);

    defaultMode = false;

  }

  // digital clock display of the time

  Serial.print(hour());

  printDigits(minute());

  printDigits(second());

  Serial.print(' ');

  Serial.print(day());

  Serial.print('-');

  Serial.print(month());

  Serial.print('-');

  Serial.print(year());

  Serial.println();

  // minute < 10 and second < 10 handler

  int hourNow = hour();

  int minuteNow = minute();

  int secondNow = second();

  // Format minutes and seconds with leading zeros

  char minuteString[3];

  char secondString[3];

  sprintf(minuteString, "%02d", minuteNow);

  sprintf(secondString, "%02d", secondNow);

  // day < 10 and month < 10 handler

  int dayNow = day();

  int monthNow = month();

  int yearNow = year() - 2000;

  // Serial.println(yearNow);

  // Format minutes and seconds with leading zeros

  char dayString[3];

  char monthString[3];

  sprintf(dayString, "%02d", dayNow);

  sprintf(monthString, "%02d", monthNow);

  //

  digitalWrite(buzzer, LOW);

  if (checkEnteredValueEnterAlarmOne == true && dayNow == dayFromKeyboard.toInt() && hourNow == hourFromKeyboard.toInt() && minuteNow == minuteFromKeyboard.toInt() && secondNow >= firstDurationFromKeyboard.toInt() && secondNow <= secondDurationFromKeyboard.toInt()) {

    digitalWrite(buzzer, HIGH);

    Serial.println("alarm one playing111111111111");

    matrix.setCursor(0, 6);

    sprintf(str, "%s", alarmOne);

    if (secondNow > secondDurationFromKeyboard.toInt()) {

      Serial.println("alarm stopped");

      dayFromKeyboard = "";

      hourFromKeyboard = "";

      minuteFromKeyboard = "";

      firstDurationFromKeyboard = "";

      secondDurationFromKeyboard = "";

      checkEnteredValueEnterAlarmOne = false;

    }

  } else if (checkEnteredValueEnterAlarmTwo == true && dayNow == dayFromKeyboard2.toInt() && hourNow == hourFromKeyboard2.toInt() && minuteNow == minuteFromKeyboard2.toInt() && secondNow >= firstDurationFromKeyboard2.toInt() && secondNow <= secondDurationFromKeyboard2.toInt()) {

    digitalWrite(buzzer, HIGH);

    Serial.println("alarm two playing222222222222");

    matrix.setCursor(0, 6);

    sprintf(str, "%s", alarmTwo);

    if (secondNow > secondDurationFromKeyboard2.toInt()) {

      Serial.println("alarm stopped");

      dayFromKeyboard2 = "";

      hourFromKeyboard2 = "";

      minuteFromKeyboard2 = "";

      firstDurationFromKeyboard2 = "";

      secondDurationFromKeyboard2 = "";

      checkEnteredValueEnterAlarmTwo = false;

    }

  } else if (strlen(alarmThree) > 0 && checkEnteredValueEnterAlarmThree == true && dayNow == dayFromKeyboard3.toInt() && hourNow == hourFromKeyboard3.toInt() && minuteNow == minuteFromKeyboard3.toInt() && secondNow >= firstDurationFromKeyboard3.toInt() && secondNow <= secondDurationFromKeyboard3.toInt()) {

    digitalWrite(buzzer, HIGH);

    Serial.println("alarm three playing33333333333333333");

    matrix.setCursor(0, 6);

    sprintf(str, "%s", alarmThree);

    if (secondNow > secondDurationFromKeyboard3.toInt()) {

      Serial.println("alarm stopped");

      strcpy(alarmThree, "");

      dayFromKeyboard3 = "";

      hourFromKeyboard3 = "";

      minuteFromKeyboard3 = "";

      firstDurationFromKeyboard3 = "";

      secondDurationFromKeyboard3 = "";

      checkEnteredValueEnterAlarmThree = false;

    }

  } else if (secondNow >= 10 && secondNow <= 13 || secondNow >= 40 && secondNow <= 43) {

    matrix.setCursor(0, 6);

    // sprintf(str, "%02d-%s-%s", yearNow, dayString, monthString);

    sprintf(str, "%s-%s-%02d", dayString, monthString, yearNow);

  } else if (secondNow >= 13 && secondNow <= 16 || secondNow >= 43 && secondNow <= 46) {

    // get temp, centers font, delay so temp is stable

    matrix.setCursor(10, 6);

    sprintf(str, "%d °C", celciusTemp);

    delay(500);

  } else {

    // Create the formatted time string

    matrix.setCursor(3, 6);

    sprintf(str, "%02d:%s:%s", hourNow, minuteString, secondString);

  }

  if (checkEnteredValueEnterAlarmOne == true && dayNow == dayFromKeyboard.toInt() && hourNow == hourFromKeyboard.toInt() && minuteNow == minuteFromKeyboard.toInt() && secondNow >= firstDurationFromKeyboard.toInt() && secondNow <= secondDurationFromKeyboard.toInt()) {

    // running text right to left

    for (int i = 50; i >= -45; i--) {

      matrix.fillScreen(0);

      matrix.setCursor(i, 6);

      matrix.print(str);

      matrix.write();

      delay(55);

    }

  } else if (checkEnteredValueEnterAlarmTwo == true && dayNow == dayFromKeyboard2.toInt() && hourNow == hourFromKeyboard2.toInt() && minuteNow == minuteFromKeyboard2.toInt() && secondNow >= firstDurationFromKeyboard2.toInt() && secondNow <= secondDurationFromKeyboard2.toInt()) {

    // running text right to left

    for (int i = 50; i >= -100; i--) {

      matrix.fillScreen(0);

      matrix.setCursor(i, 6);

      matrix.print(str);

      matrix.write();

      delay(55);

    }

  } else if (checkEnteredValueEnterAlarmThree == true && dayNow == dayFromKeyboard3.toInt() && hourNow == hourFromKeyboard3.toInt() && minuteNow == minuteFromKeyboard3.toInt() && secondNow >= firstDurationFromKeyboard3.toInt() && secondNow <= secondDurationFromKeyboard3.toInt()) {

    // running text right to left

    for (int i = 50; i >= -50; i--) {

      matrix.fillScreen(0);

      matrix.setCursor(i, 6);

      matrix.print(str);

      matrix.write();

      delay(55);

    }

  } else {

    matrix.print(str);

    matrix.write();

    matrix.fillScreen(0);

  }

}

void handleLightSensorAndBrightness() {

  // changes brightness according to light sensor

  if (analogRead(lightSensor) < 100) {

    // terang banget

    matrix.setIntensity(10);

  } else if (analogRead(lightSensor) < 200) {

    // terang

    matrix.setIntensity(8);

  } else if (analogRead(lightSensor) < 500) {

    // normal

    matrix.setIntensity(5);

  } else if (analogRead(lightSensor) < 800) {

    // gelap

    matrix.setIntensity(3);

  } else {

    // gelap banget

    matrix.setIntensity(0);

  }

}

void setMatrixDisplayOrientation() {

  // Adjust to your own needs

  //  matrix.setPosition(0, 0, 0); // The first display is at <0, 0>

  //  matrix.setPosition(1, 1, 0); // The second display is at <1, 0>

  //  matrix.setPosition(2, 2, 0); // The third display is at <2, 0>

  //  matrix.setPosition(3, 3, 0); // And the last display is at <3, 0>

  //  ...

  //  matrix.setRotation(0, 2);    // The first display is position upside down

  //  matrix.setRotation(3, 2);    // The same hold for the last display

  matrix.setRotation(0, 1);

  matrix.setRotation(1, 1);

  matrix.setRotation(2, 1);

  matrix.setRotation(3, 1);

}

void printDigits(int digits) {

  // utility function for digital clock display: prints preceding colon and leading 0

  Serial.print(':');

  if (digits < 10)

    Serial.print('0');

  Serial.print(digits);

}

void handleSetClockOrDate(String enteredValue) {

  if (enteredValue == "enteredValueEnter") {

    strcat(enteredValueEnter, tempStr);

    Serial.println(enteredValueEnter);

    // grab hour,minute,second from enteredValueEnter

    hourFromKeyboard = String(enteredValueEnter).substring(0, 2);

    minuteFromKeyboard = String(enteredValueEnter).substring(2, 4);

    secondFromKeyboard = String(enteredValueEnter).substring(4, 7);

    Serial.println(hourFromKeyboard);

    Serial.println(minuteFromKeyboard);

    Serial.println(secondFromKeyboard);

    if (!(hourFromKeyboard.length() == 0) && !(minuteFromKeyboard.length() == 0) && !(secondFromKeyboard.length() == 0)) {

      checkEnteredValueEnter = true;

      Serial.println("defined");

    } else {

      Serial.println("hour is undefined");

    }

    // end of input

    strcpy(tempStr, "");

    strcpy(enteredValueEnter, "");

    matrix.fillScreen(0);

  } else if (enteredValue == "enteredValueBracket") {

    strcat(enteredValueBracket, tempStr);

    Serial.println(enteredValueBracket);

    // grab day,month,year from enteredValueBracket

    dayFromKeyboard = String(enteredValueBracket).substring(0, 2);

    monthFromKeyboard = String(enteredValueBracket).substring(2, 4);

    yearFromKeyboard = String(enteredValueBracket).substring(4, 8);

    Serial.println(dayFromKeyboard);

    Serial.println(monthFromKeyboard);

    Serial.println(yearFromKeyboard);

    if (!(dayFromKeyboard.length() == 0) && !(monthFromKeyboard.length() == 0) && !(yearFromKeyboard.length() == 0)) {

      checkEnteredValueBracket = true;

      Serial.println("defined");

    } else {

      Serial.println("hour is undefined");

    }

    // end of input

    strcpy(tempStr, "");

    strcpy(enteredValueBracket, "");

    matrix.fillScreen(0);

  } else if (enteredValue == "enteredValueEnterAlarm") {

    strcat(enteredValueEnterAlarm, tempStr);

    Serial.println(enteredValueEnterAlarm);

    if (checkAlarmOneStart == true) {

      // grab day,hour,minute,firstDuration,secondDuration from enteredValueEnterAlarm

      dayFromKeyboard = String(enteredValueEnterAlarm).substring(0, 2);

      hourFromKeyboard = String(enteredValueEnterAlarm).substring(2, 4);

      minuteFromKeyboard = String(enteredValueEnterAlarm).substring(4, 6);

      firstDurationFromKeyboard = String(enteredValueEnterAlarm).substring(6, 8);

      secondDurationFromKeyboard = String(enteredValueEnterAlarm).substring(8, 10);

      Serial.println(dayFromKeyboard);

      Serial.println(hourFromKeyboard);

      Serial.println(minuteFromKeyboard);

      Serial.println(firstDurationFromKeyboard);

      Serial.println(secondDurationFromKeyboard);

      // validate if new variable (enteredValueEnterAlarm) is empty or not

      if (!(dayFromKeyboard.length() == 0) && !(hourFromKeyboard.length() == 0) && !(minuteFromKeyboard.length() == 0) && !(firstDurationFromKeyboard.length() == 0) && !(secondDurationFromKeyboard.length() == 0)) {

        Serial.println("heloooo11111111111");

        checkEnteredValueEnterAlarmOne = true;

        Serial.println("defined");

      } else {

        Serial.println("undefined");

      }

      checkAlarmOneStart = false;

    } else if (checkAlarmTwoStart == true) {

      // grab day,hour,minute,firstDuration,secondDuration from enteredValueEnterAlarm

      dayFromKeyboard2 = String(enteredValueEnterAlarm).substring(0, 2);

      hourFromKeyboard2 = String(enteredValueEnterAlarm).substring(2, 4);

      minuteFromKeyboard2 = String(enteredValueEnterAlarm).substring(4, 6);

      firstDurationFromKeyboard2 = String(enteredValueEnterAlarm).substring(6, 8);

      secondDurationFromKeyboard2 = String(enteredValueEnterAlarm).substring(8, 10);

      Serial.println(dayFromKeyboard2);

      Serial.println(hourFromKeyboard2);

      Serial.println(minuteFromKeyboard2);

      Serial.println(firstDurationFromKeyboard2);

      Serial.println(secondDurationFromKeyboard2);

      // validate if new variable (enteredValueEnterAlarm) is empty or not

      if (!(dayFromKeyboard2.length() == 0) && !(hourFromKeyboard2.length() == 0) && !(minuteFromKeyboard2.length() == 0) && !(firstDurationFromKeyboard2.length() == 0) && !(secondDurationFromKeyboard2.length() == 0)) {

        Serial.println("heloooo2222222222");

        checkEnteredValueEnterAlarmTwo = true;

        Serial.println("defined");

      } else {

        Serial.println("undefined");

      }

      checkAlarmTwoStart = false;

    } else if (checkAlarmThreeStart == true) {

      // grab day,hour,minute,firstDuration,secondDuration from enteredValueEnterAlarm

      dayFromKeyboard3 = String(enteredValueEnterAlarm).substring(0, 2);

      hourFromKeyboard3 = String(enteredValueEnterAlarm).substring(2, 4);

      minuteFromKeyboard3 = String(enteredValueEnterAlarm).substring(4, 6);

      firstDurationFromKeyboard3 = String(enteredValueEnterAlarm).substring(6, 8);

      secondDurationFromKeyboard3 = String(enteredValueEnterAlarm).substring(8, 10);

      Serial.println(dayFromKeyboard3);

      Serial.println(hourFromKeyboard3);

      Serial.println(minuteFromKeyboard3);

      Serial.println(firstDurationFromKeyboard3);

      Serial.println(secondDurationFromKeyboard3);

      // validate if new variable (enteredValueEnterAlarm) is empty or not

      if (!(dayFromKeyboard3.length() == 0) && !(hourFromKeyboard3.length() == 0) && !(minuteFromKeyboard3.length() == 0) && !(firstDurationFromKeyboard3.length() == 0) && !(secondDurationFromKeyboard3.length() == 0)) {

        Serial.println("heloooo333333333333333333333");

        checkEnteredValueEnterAlarmThree = true;

        Serial.println("defined");

      } else {

        Serial.println("undefined");

      }

      checkAlarmThreeStart = false;

    }

    // end of input

    strcpy(tempStr, "");

    strcpy(enteredValueEnterAlarm, "");

    matrix.fillScreen(0);

  } else if (enteredValue == "enteredValueBracketAlarm3") {

    strcat(enteredValueEnterAlarm, tempStr);

    strcpy(alarmThree, enteredValueEnterAlarm);

    Serial.println(alarmThree);

    // end of input

    strcpy(tempStr, "");

    strcpy(enteredValueEnterAlarm, "");

    matrix.fillScreen(0);

  }

}

- Hasil display

- clock, yang dimana pada saat detik 10, 13 dan 40, 43 akan tampil kalender dan suhu



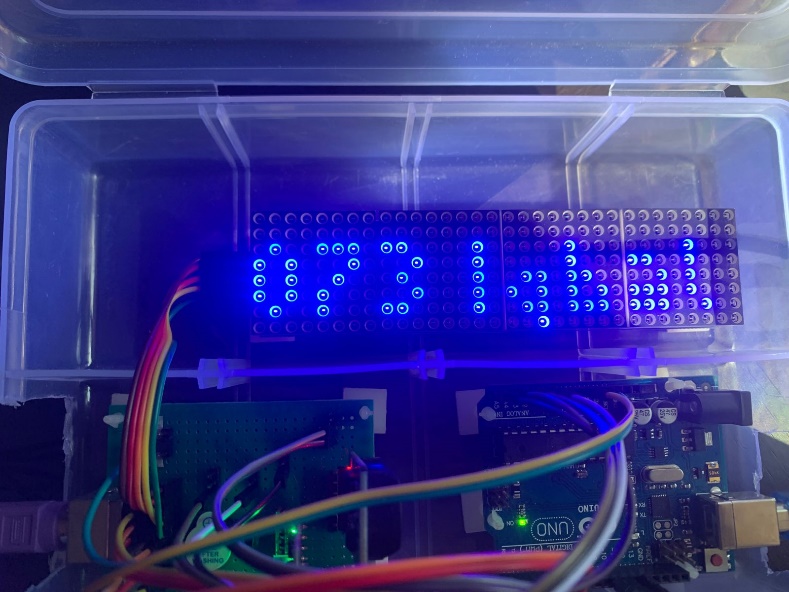




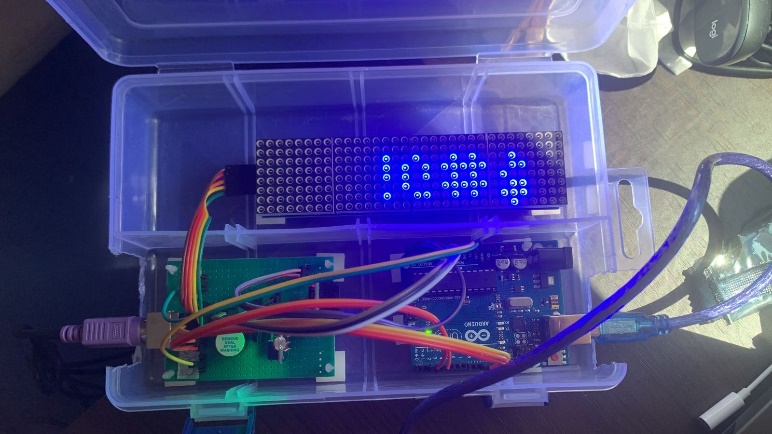
- display pada saat alarm 1 bernyala



- display pada saat alarm 2 bernyala



- display pada saat alarm 3 bernyala (menampilkan simbol !@#$)



1. **KESIMPULAN**

Walaupun fitur utama sudah saya kerjakan, hasil akhirnya masih memiliki beberapa kekuranga, seperti:

- Durasi alarm yang diset tidak sepenuhnya akurat karna display menunggu scrolling sepenuhnya sblm mematikan alarm tersebut.

- Pada saat mode set alarm 3.2 apabila melakukan enter empty string “” akan tercacat menjadi string yang akan dioutput pada alarm ketiga nanti.

- Saat setting clock/date, seharusnya input tidak boleh lebih dari 6 digit, tetapi user sekarang dapat melakukan lebih dari 6 digit.

- LM35 tidak stabil.

- Saat mode input, display tidak kedap kedip.

- Pada saat modeSet untuk set semua alarm dan set clock/date biasa, apabila input dari keyboard melebihi batas dari display dotmatrix, input tidak kelihatan, menyulitkan user untuk melakukan penyetelan.

1. **CARA NAVIGASI RANGKAIAN**

Terdapat 6 mode:

- “F1” untuk mode tampilan clock, temp, dan date

- “F2” untuk mode set clock dan date

a. pada mode ini dot matrix menampilkan “s:”, melambangkan set clock/date mode

b. dapat melakukan set clock dan date dengan cara,

- untuk clock, masukkan angka dengan format hhmmss (hour, minute, second, tanpa spasi) lalu tekan “enter”.

- untuk date, masukkan angka dengan format ddmmyyyy (date, month, year, tanpa spasi) lalu tekan “]”.

- “F5” untuk mode set alarm 1

a. dot matrix menampilkan “1:”, melambangkan set mode alarm 1

- “F6” untuk mode set alarm 2

a. dot matrix menampilkan “2:”, melambangkan set mode alarm 2

- “F7” untuk mode set alarm 3.1

a. dot matrix menampilkan “3.1:”, melambangkan set mode alarm 3.1

- “F8” untuk mode set alarm 3.2

a. dot matrix menampilkan “3.2:”, melambangkan set mode alarm 3.2

Untuk keluar dari sebuah mode, bisa tekan “F12”, tekan “F12” juga apabila display dot matrix tampilannya aneh/rusak.

Pada set mode clock/date, set alarm 1, set alarm 2, set alarm 3.1, dan set alarm 3.2:

- tekan “BACKSPACE” untuk menghilangkan angka satu persatu.

- tekan “ESC” untuk menghilangkan seluruh angka.

Pada set alarm 1, set alarm 2, dan set alarm 3.1:

- untuk set alarm dapat pencet “ENTER” dengan format ddhhmmaabb (date, hour, minute, when alarm starts, when alarm ends, tanpa spasi, disini hanya dapat input angka).

Pada set alarm 3.2:

- untuk set alarm dapat pencet “]” dengan format bebas.

**Note**:

- pada setiap mode setting, karena format input yang ada terkadang lebih dari panjangnya tampilan dot matrix, user terkadang tidak dapat melihat apa yang diinputkan.

contoh: pada set mode clock/date, format set date adalah ddmmyyyy user tidak bisa melihat yyyy pada tampilan dot matrix, tapi tetap terbaca dan tetap bisa melakukan “backspace” dan “esc”.