LAPORAN PROYEK AKHIR SISTEM EMBEDDED
DEPARTEMEN TEKNIK ELEKTRO
UNIVERSITAS INDONESIA

STUDENT CORNER e-TIMER

GROUP PA6

Muhammad Raihan Mustofa	**2306161946**
Dwigina Sitti Zahwa	**2306250724**
Tri Yoga Arsyad	**2306161920**
Muhammad Rifat Faqih	**2306250762**
Farhan Ramadhani Zakiyyandi	**2306220412**

PENGANTAR

Segala puji dan syukur ke Tuhan Yang Maha Esa atas segala rahmat dan karunia-nya sehingga Laporan Proyek Akhir dari Sistem Embedded yang berjudul “Student Corner e-Timer” bisa diselesaikan dengan baik. Penulis ucapkan terima kasih kepada Bang Evandita sebagai Pendamping Aslab yang telah membantu pengerjaan dan penulis yang telah berkontribusi dalam pengerjaan Proyek Akhir Sistem Embedded System.

Laporan ini disusun untuk melengkapi Praktikum Sistem Embedded Tahun 2025. Laporan ini berisi penjelasan lengkap mengenai Proyek “Student Corner e-Timer” sebuah solusi untuk Student Corner DTE yang selalu ramai, oleh karena itu proyek ini dibuat untuk menjawab permasalahan tersebut. Menggunakan Arduino Uno sebagai Microcontroller, DS3231 RTC sebagai sensor waktu untuk membandingkan saat mahasiswa menekan *button* sampai *time limit* dan akan menampilkan waktu ke MAX7219 dengan SPI, data atau memori *button* yang ditekan akan disimpan di EEPROM interal Arduino Uno, sebagai aktuator menggunakan LED untuk mengindikasikan apakah izin masuk diberikan, dan buzzer untuk menandakan waktu sudah habis..

Penulis menyadari kurangnya pengetahuan dan keterbatasan dalam pengerjaan dan penyusunan laporan yang harus diperbaiki. Oleh karena itu, besar harapan penulis agar bisa mendapat kritik dan saran sehingga bisa dijadikan bahan evaluasi kedepannya. Praktikkan juga memohon maaf jika ada kesalahan dan kurangnya dalam penyusunan laporan.

Depok, Mei 18, 2025

Group PA

DAFTAR ISI

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND PROBLEM

Gedung Departemen Elektro di Fakultas Teknik memiliki sebuah ruangan Student Corner yang difasilitasi untuk Mahasiswa DTE jika ingin mengerjakan tugas pada lingkungan yang hening, tersedia stop kontak, dan nyaman. Masalahnya, ruangan SC hanya memiliki 10 meja, sementara itu jumlah Mahasiswa DTE $\pm$ 756 hal ini menyebabkan tidak semua mahasiswa bisa merasakan fasilitas tersebut. Selain itu, tidak jarang beberapa mahasiswa menggunakan ruangan sampai 7 jam sehari, sehingga mahasiswa lain tidak bisa menggunakannya.

1.2 SOLUTION

Permasalahan penggunaan ruangan yang tidak pernah bisa diselesaikan ini memberikan ide kepada praktikkan untuk membuat Proyek Student Corner e-Timer, proyek ini menggunakan *assembly language* dan proteus untuk membuat desain rangkaian. Terdiri atas DS3231 RTC untuk sensor waktu yang menunjukkan *hours* - *minutes* - *seconds* dan akan ditampilkan ke MAX7219 8-Digit LED Display Drivers menggunakan Serial Communication SPI dan I2C ke Slave MAX7219, *button* sebagai *input* mahasiswa untuk akses ruangan, jika ditekan akan menghasilkan *interrupt* dan data *button* akan disimpan di EEPROM pada Arduino Uno, aktuator menggunakan LED Merah untuk menandakan jika "Akses Ditolak" yang ditampilkan ke Serial Monitor dengan USART karena mahasiswa sudah mencapai *limit* harian dan LED Hijau untuk menandakan jika "Akses Diterima", jika waktu dengan DS3231 RTC sudah menunjukkan *limit*, maka *buzzer* akan berbunyi dan hanya bisa dihentikan jika mahasiswa menekan *button* lagi.

1.3 OBJECTIVE

Tujuan dari proyek ini antara lain:

1. Memenuhi syarat kelengkapan Praktikum Sistem Embedded.
2. Mengimplementasikan *attendance system* menggunakan *button* sebagai verifikasi mahasiswa untuk mengakses ruangan.
3. Menggunakan DS3231 RTC untuk membandingkan waktu dari awal mengakses

sampai *time limit* dan disimpan di EEPROM.

4. Memberikan aktuator melalui LED untuk menandakan jika akses diberikan atau tidak dan mengirim pemberitahuan *time limit* dengan buzzer.
5. Mengaplikasikan dan mengintegrasikan Modul Assembly, Serial Port, Aritmetika, Interrupt, SPI I2C, dan Sensor Interface dalam satu proyek yang andal dan efisien.

CHAPTER 2

HARDWARE DESIGN

2.1. TOOLS

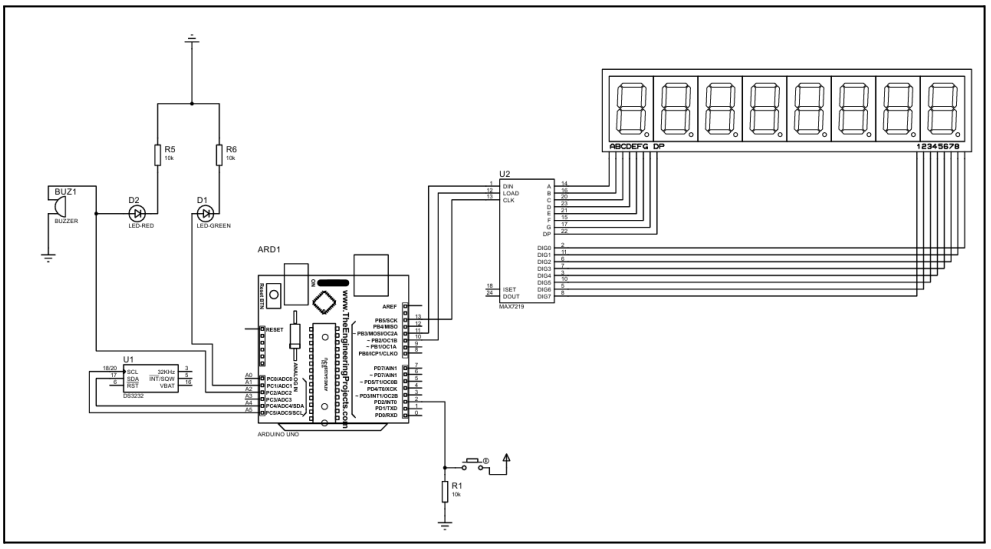
Alat yang digunakan dalam Implementasi Student Corner e-Timer antara lain:

1. Arduino Uno
2. Proteus 8 Professional
3. 3 Resistor 220 - 10k ohm
4. 1 DS3231 RTC
5. 2 Button Active 5V
6. 1 MAX7219

2.2. DESIGN IMPLEMENTATION

Perancangan desain dibuat menggunakan Proteus 8 Professional, perancangan ini didesain sebagai *prototype* dari produk proyek agar praktikkan mengetahui apakah produk bisa efisien, menggunakan apa saja, dan masalah yang ditemukan. Komponen Arduino ataupun sensor didapatkan dengan meng-*import* dari The Engineer Project.

LED Merah dan LED Hijau sebagai aktuator dihubungkan ke PIN A0 dan A1 (PORT C), katoda dihubungkan ke *ground* melalui resistor (*pull - up*) agar tidak *floating* dan anoda dihubungkan ke PIN. Aktuator lainnya untuk menandakan *time limit* menggunakan *buzzer* dengan salah satu kaki terhubung ke *ground* dan kaki lainnya terhubung ke LED Merah untuk NOT dan ke PIN A2 (PORT C). Sebagai penanda waktu dan untuk *compare* saat menekan *button* sampai waktu selesai menggunakan DS3231 RTC, sensor ini menampilkan waktu ke MAX7219 menggunakan SPI Serial Communication, PIN SCL pada sensor terhubung ke PIN A5 dan PIN SDA terhubung ke PIN A4 (PORT C). MAX7219 menghubungkan semua kaki ke MAX7219 Slave, pada slave tersebut kaki DIN dihubungkan ke PIN 11, LOAD ke PIN 10, dan CLK ke PIN 13 (PORT B). Sebagai *input* untuk memberikan akses kepada mahasiswa menggunakan *interrupt* dari *button* dan data dari *button* disimpan di EEPROM, pada salah satu kaki *button* terhubung ke PIN 7 melalui resistor untuk menghindari *floating* dengan aktifkan *pull-up resistor* dan kaki satunya ke VCC.



Gbr 1.0 Proteus Schematic Design

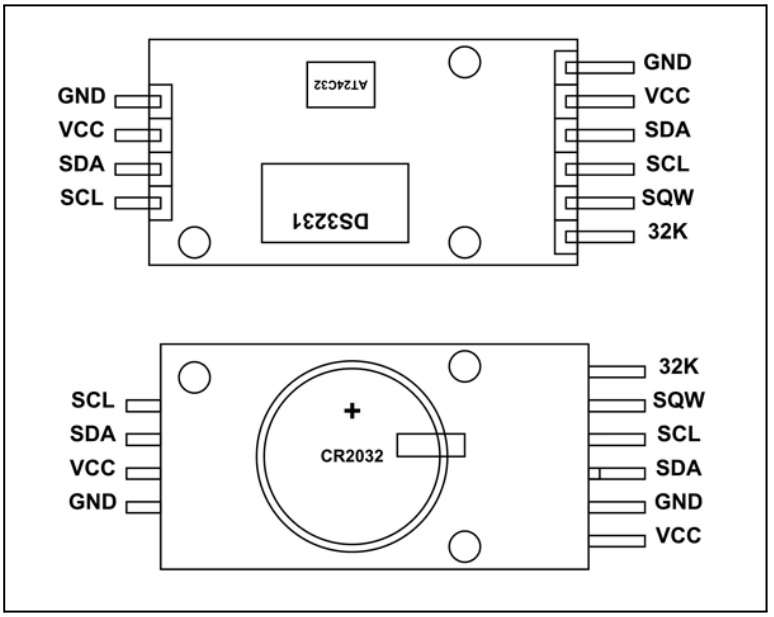
2.3. HARDWARE COMPONENT

Perancangan rangkaian menggunakan beberapa komponen agar bisa mengidentifikasi adanya *interrupt*, sensor waktu, aktuator berupa *buzzer* dan LED, serta dengan Serial Communication.

2.3.0. DS3231 RTC

Modul Real-Time Clock untuk integrasi dengan *temperature-compensated crystal oscillator (TCXO)* dan crystal, keakuratannya memiliki akurasi ±2ppm dan mempertahankan akurasi waktu ketika daya utama terputus. Modul RTC DS3231 terhubung dengan Arduino Uno dengan I2C pada pin SDA (Serial Data) untuk data I2C yang terhubung ke PC4 dan SCL (Serial Clock) untuk sinyal *clock* I2C yang terhubung ke PC5.

Data waktu disimpan dalam format BCD (Binary-Coded Decimal) pada Register DS3231 dan dikonversi ke desimal untuk ditampilkan pada display 7-segment. Modul ini menyimpan nilai waktu ke EEPROM internal ATmega328P pada alamat 0x0000 hingga 0x0002 dan terdapat fungsi untuk menyimpan nilai waktu ke alamat 0x0008 hingga 0x000A ketika *interrupt* eksternal ter-*trigger*. Protokol I2C pada modul ini dengan alamat 0x68 (*write* - 0xD0, *read* - 0xD1).

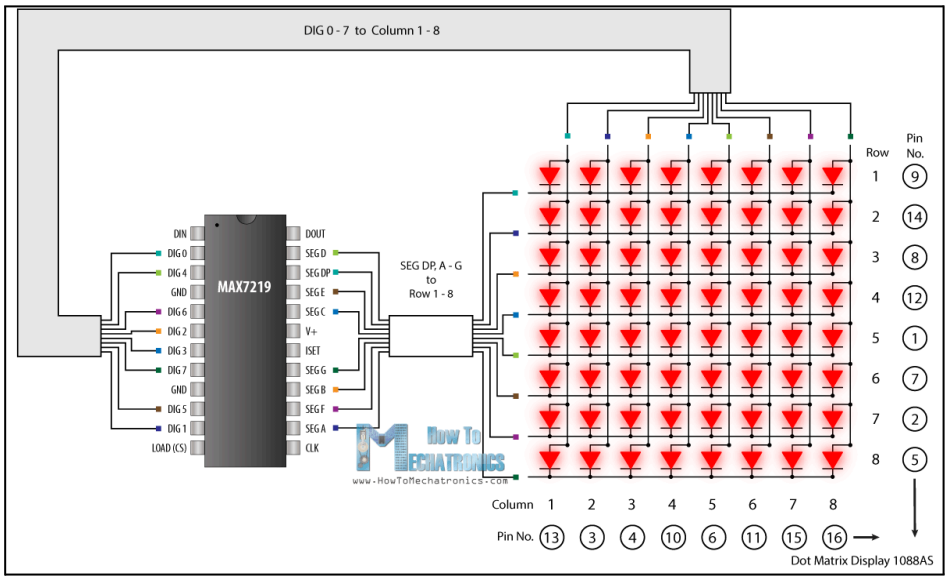


Gbr 2.0 DS3231 Schematic Design

2.3.1. **MAX7219 8-Digit LED Display Drivers**

Driver display LED 8-digit yang terintegrasi dengan *display 7-segment common-cathode*. Menyederhanakan *interface* ke display LED dan komunikasi menggunakan protokol SPI (Serial Peripheral Interface).

Tampilan dalam format waktu *hours - minutes - second* dengan tanda "-" sebagai pemisah. Program mengkonversi nilai BCD dari RTC ke format sesuai untuk MAX7219 dan mengupdate display secara kontinu. MAX7219 memiliki *interface* serial 3-wire (DIN untuk data input SPI yang terhubung ke MOSI, CLK untuk sinyal clock SPI yang terhubung ke SCK, LOAD untuk select yang terhubung ke SS) dengan frekuensi fsck. MAX7219 menawarkan mode decoding BCD untuk menampilkan digit 0-9 dan karakter khusus, selain itu data dikirim dengan aktifkan LOAD, mengirim command dan data, kemudian mematikan Slave Select tersebut.



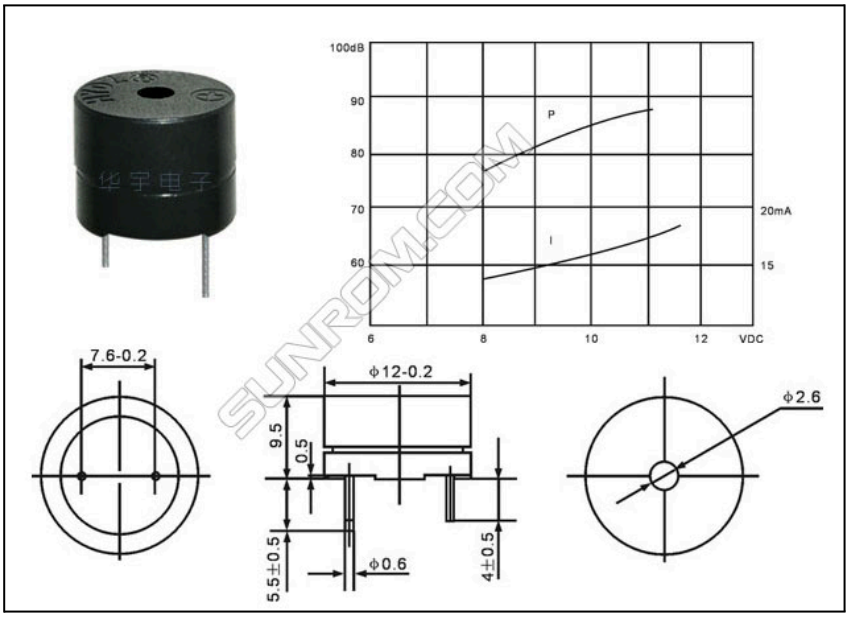
Gbr 3.0 MAX7219 Schematic Design

2.3.2. **Arduino ATmega328P**

Mikrokontroler 8-bit berbasis AVR pada platform Arduino yang memiliki 32KB Flash memory, 2KB SRAM, dan 1KB EEPROM. Arduino ini berfungsi untuk menjalankan komunikasi dengan DS3231 RTC melalui I2C, mengendalikan display LED melalui MAX7219 menggunakan SPI, menyimpan data waktu ke EEPROM internal, menangani interrupt eksternal (INT0) pada PIN PD2 untuk menyimpan waktu saat tombol ditekan, mengelola sinyal I/O dari komponen (LED, buzzer, button).

2.3.3. **Buzzer**

Komponen audio yang menghasilkan suara ketika diberi tegangan. Buzzer pada rangkaian tersambung ke ground dan pin Arduino serta LED - Merah. Buzzer yang terhubung ke transistor yang memberikan sinyal HIGH agar mengeluarkan suara, pada rangkaian buzzer akan menyala jika *compare* waktu yang terbaca di EEPROM dari RTC sudah menunjukkan waktu lebih dari 10 sekon sejak *button* ditekan, kemudian LED Merah (NOT) akan mengirim sinyal untuk membunyikan.



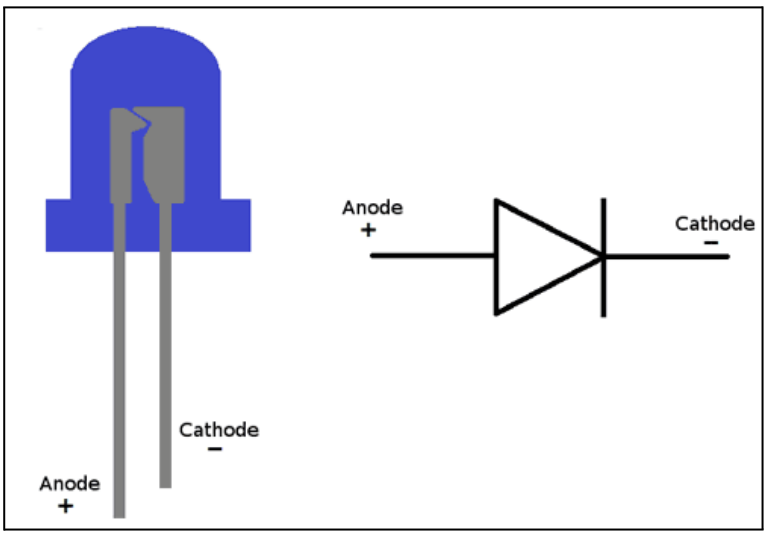
Gbr 4.0 MAX7219 Schematic Design

2.3.4. **Button Switch**

Memicu penyimpanan data waktu saat ditekan yang terhubung ke interrupt eksternal INT0 (PD2) yang memicu *interrupt*, mengonfigurasi dengan resistor pull-up internal (SBI PORTC, 2) untuk menghindari *floating*. Ketika *button* ditekan, INT0 terpicu dan mengaktifkan flag (R21) yang menandakan sistem harus menyimpan data waktu saat ini ke EEPROM pada alamat 0x0008-0x000A. Selain itu menggunakan *debounce* untuk menghindari multiple trigger akibat bouncing mekanis tombol.

2.3.5. **LED**

LED pada PIN PC2 menyala ketika selisih waktu antara nilai di alamat EEPROM 0x0002 dan 0x0008 lebih besar atau sama dengan 10 detik, sedangkan LED PIN PC1 akan menyala ketika PC2 mati dan sebaliknya. LED ini diatur dengan *compare* nilai detik yang disimpan di EEPROM. LED ini sebagai indikator visual untuk menunjukkan status atau hasil perbandingan data.



Gbr 5.0 MAX7219 Schematic Design

2.3.6. **Resistor**

Komponen pembatas arus untuk LED dan *pull-up* untuk *button*. Resistor pembatas arus untuk LED untuk melindungi dari arus berlebih dan pull-up atau pull-down untuk konfigurasi input. Mengaktifkan *pull-up* untuk menghindari adanya floating sehingga pembacaan terhindar dari noise.

CHAPTER 3

SOFTWARE DESIGN

3.1 TOOLS

Alat yang digunakan dalam Implementasi Student Corner e-Timer antara lain:

1. Visual Studio Code
2. Arduino Uno
3. Proteus 8 Professional

3.2 SOFTWARE COMPONENT

Perancangan Perangkat lunak yang diterapkan dalam proyek Student Corner e-Time. Sistem ini dituliskan menggunakan bahasa Assembly melalui Arduino IDE, yang berfungsi untuk mengelola komponen seperti RTC, tombol, EEPROM, LED, buzzer, dan tampilan simulasi proyek menggunakan Proteus 8 Professional guna memastikan kinerja sistem yang sesuai

1. Visual Studio Code (VS Code)

   Dijadikan sebagai program pengolah teks alternatif untuk menyusun kode (jika tidak menggunakan Arduino IDE secara langsung). VS Code juga dapat ditambahkan dengan ekstensi untuk mendukung pengembangan proyek Arduino dan Assembly dengan penyorotan sintaks, IntelliSense, dan VS Code dipilih karena:

   - Mendukung penulisan kode dalam Assembly dan C/C++ dengan fitur seperti penyorotan sintaks, penyelesaian otomatis (IntelliSense), serta integrasi dengan ekstensi Arduino.
   - Memfasilitasi pengaturan struktur proyek yang lebih baik dan efisien, terutama saat proyek menjadi lebih rumit.
   - Menjadi alternatif yang fleksibel bagi pengguna yang lebih akrab dengan manajemen kode melalui editor teks sederhana yang tetap kaya fitur.

2. Arduino Uno

   Arduino Uno berperan sebagai pengendali utama yang mengatur dan menyelaraskan semua komponen dalam sistem. Fungsinya lebih dari sekadar mikrokontroler biasa, melainkan juga sebagai pelaksana utama dari semua logika yang ditulis dalam bahasa Assembly, termasuk berbagai fungsi penting. Dalam proyek ini arduino uno digunakan untuk:

   - **Mengatur komunikasi antara komponen:** Arduino Uno berfungsi sebagai

pengendali utama yang mengkoordinasikan interaksi dengan RTC DS3231, EEPROM, dan LED Matrix MAX7219 melalui protokol seperti I2C dan SPI.

- **Menerima interupsi eksternal dan penyimpanan informasi:** Ketika tombol ditekan, Uno mendeteksi sinyal melalui interrupt INT0 (PD2), kemudian secara otomatis menyimpan informasi waktu dari RTC ke EEPROM.

- **Melaksanakan logika program Assembly:** Arduino Uno menjalankan kode Assembly yang meliputi pengambilan waktu, pengolahan informasi, serta penampilan hasil, dan juga mengatur buzzer sebagai sinyal atau alarm.

3. Proteus 8 Professional

Digunakan untuk mengembangkan simulasi sirkuit elektronik pada proyek. selain itu proteus juga memberi kesempatan untuk pengguna melakukan pengujian desain sirkuit dan program mikrokontroler tanpa memerlukan alat fisik dan diantara fungsi utamanya adalah mencangkup:

- Membangun rangkaian simulasi untuk keseluruhan sistem, yang meliputi elemen-elemen seperti RTC, EEPROM, MAX7219, LED, buzzer, transistor, dan tombol.
- Melaksanakan simulasi interaktif untuk melihat bagaimana sistem berfungsi secara langsung, termasuk pemantauan waktu, penyimpanan data, dan hasil tampilan.
- Memastikan bahwa kode yang ditulis di Arduino IDE dapat berfungsi sesuai dengan logika yang diharapkan tanpa menghadapi masalah logika atau kesalahan fungsional saat diterapkan pada mikrokontroler.

3.3 ASSEMBLY CODE

```
;============================================================
DS3231_RD:
;---------
    ; Konfigurasi tombol eksternal pada PC2
    CBI   DDRC, 2         ; Pin PC2 i/p (save rtc to EEPROM)
    SBI   PORTC, 2        ; Enable pull-up resistor

    ; Konfigurasi PC1 sebagai output untuk sinyal NOT dari PC2
    SBI   DDRC, 1         ; Set PC1 sebagai output
    CBI   PORTC, 1        ; Inisialisasi PC1 sebagai LOW (kebalikan d

    ; Konfigurasi INT0 (eksternal interrupt pada PD2)
    LDI   R16, (1<<ISC01) ; Trigger INT0 pada falling edge (LOW)
    STS   EICRA, R16      ; Set interrupt control register

    LDI   R16, (1<<INT0)  ; Enable INT0
    OUT   EIMSK, R16      ; Set external interrupt mask

    ; Inisialisasi flag in register instead of SRAM
    CLR   R21             ; R21 = 0, flag initially cleared

    ; Enable global interrupt
    SEI                   ; Set Global Interrupt Enable bit
```

Pada Bagian ini mengatur pin PC2 sebagai input dengan resistor pull-up yang aktif, serta mengonfigurasi pin PC1 sebagai output yang berfungsi berlawanan dengan kondisi PC2. Di samping itu, interrupt eksternal INT0 diaktifkan untuk mendeteksi perubahan sinyal jatuh pada pin PD2, yang digunakan untuk mengaktifkan proses penyimpanan data dari RTC ke EEPROM. Register R21 juga diatur sebagai bendera kontrol, dan interrupt global diaktifkan menggunakan instruksi SEI.

```
;read time
;---------
again:
    RCALL I2C_START       ;transmit START condition
    ;-------------------------------------------------------
    LDI   R27, 0b11010000 ;write address of DS3231
    RCALL I2C_write       ;send 1st byte
    ;-------------------------------------------------------
    LDI   R27, 0x00       ;set reg pointer to Seconds
    RCALL I2C_write       ;send 2nd byte
    ;-------------------------------------------------------
    RCALL I2C_STOP        ;transmit STOP condition
    ;-------------------------------------------------------
    RCALL I2C_START       ;transmit START condition
    ;-------------------------------------------------------
    LDI   R27, 0b11010001 ;read address of DS1307
    RCALL I2C_write       ;send 1st byte
    ;-------------------------------------------------------
    RCALL I2C_read        ;read seconds
    MOV   R28, R27        ;store copy in R28
    RCALL I2C_read        ;read minutes
    MOV   R30, R27        ;store copy in R30
    RCALL I2C_read_NACK   ;read hour, return NACK
    MOV   R31, R27        ;store copy in R31
    ;-------------------------------------------------------
    RCALL I2C_STOP        ;transmit STOP condition
    ;-------------------------------------------------------
    RCALL save_rtc_to_eeprom
    RCALL compare_eeprom_seconds  ; Tambahkan ini untuk cek selisih

    ; Cek flag penyimpanan yang di-set oleh interrupt
    CPI   R21, 1          ; Apakah flag = 1?
    BRNE  skip_saving     ; Jika tidak, lompat
```

```
; Cek flag penyimpanan yang di-set oleh interrupt
CPI   R21, 1          ; Apakah flag = 1?
BRNE  skip_saving     ; Jika tidak, lompat

; Flag = 1, simpan data ke EEPROM di alamat 0x0008 dan kelipatannya
PUSH  R21             ; Simpan flag
RCALL save_to_EEPROM  ; Simpan waktu ke alamat kelipatan 0x0008
POP   R21             ; Kembalikan flag

; Reset flag setelah menyimpan
CLR   R21
```

Pada Bagian ini melakukan komunikasi I2C untuk mengambil informasi waktu (detik, menit, jam) dari RTC DS3231. kemudian data yang diterima disimpan sementara di register (R28, R30, R31). Kemudian, data tersebut dibandingkan dengan yang terdapat di EEPROM untuk mengecek adanya perubahan. Apabila flag dari interrupt (R21) memiliki nilai 1, maka informasi waktu akan disimpan ke EEPROM pada alamat yang spesifik, kemudian flag tersebut di-reset agar siap untuk penyimpanan berikutnya.

```
;============================================================
;read date
;---------
    RCALL I2C_START       ;transmit START condition
    ;-----------------------------------------------------
    LDI   R27, 0b11010000 ;write address of DS3231
    RCALL I2C_write       ;send 1st byte
    ;-----------------------------------------------------
    LDI   R27, 0x04       ;set reg pointer to Date
    RCALL I2C_write       ;send 2nd byte
    ;-----------------------------------------------------
    RCALL I2C_STOP        ;transmit STOP condition
    ;-----------------------------------------------------
    RCALL I2C_START       ;transmit START condition
    ;-----------------------------------------------------
    LDI   R27, 0b11010001 ;read address of DS3231
    RCALL I2C_write       ;send 1st byte
    ;-----------------------------------------------------
    RCALL I2C_read        ;read day
    MOV   R16, R27        ;store copy in R16
    RCALL I2C_read        ;read month
    MOV   R24, R27        ;store copy in R24
    RCALL I2C_read_NACK   ;read year, return NACK
    MOV   R20, R27        ;store copy in R20
    ;-----------------------------------------------------
    RCALL I2C_STOP        ;transmit STOP condition
    ;-----------------------------------------------------
    RCALL delay_ms
```

Pada Bagian kode ini digunakan untuk mengambil informasi tanggal (hari, bulan, tahun) dari RTC DS3231 melalui I2C. Mikrokontroler mengubah pointer register ke alamat 0x04, kemudian membaca tiga angka berturut-turut: hari (R16), bulan (R24), dan tahun (R20), ditutup dengan sinyal NACK dan STOP.

```
;============================================================
    ;-----------------------------------------------------
    ;display seconds on MAX7219
    ;-------------------------
    MOV   R29, R28
    ANDI  R28, 0x0F
    LDI   R17, 01
    RCALL binary2decimal  ;convert & display LSD of seconds
    MOV   R28, R29
    ANDI  R28, 0xF0
    SWAP  R28
    LDI   R17, 02
    RCALL binary2decimal  ;convert & display MSD of seconds
```

Fungsi dari kode ini adalah guna untuk menunjukkan nilai detik pada MAX7219. Digit satuan diambil dari R28 dengan ANDI 0x0F dan ditampilkan di digit 1, sedangkan digit puluhan diperoleh dengan ANDI 0xF0, kemudian ditukar dan ditampilkan di digit 2.

```
    ;-----------------------------------------------------
    ;display minutes on MAX7219
    ;-------------------------
    MOV   R28, R30
    ANDI  R28, 0x0F
    LDI   R17, 04
    RCALL binary2decimal  ;convert & display LSD of minutes
    MOV   R28, R30
    ANDI  R28, 0xF0
    SWAP  R28
    LDI   R17, 05
    RCALL binary2decimal  ;convert & display MSD of minutes
```

Fungsi dari bagian kode ini adalah untuk menunjukkan angka menit pada MAX7219. Angka satuan diambil dari R30 dengan menggunakan ANDI 0x0F dan ditampilkan pada digit 4, sementara angka puluhan diambil menggunakan ANDI 0xF0, kemudian dibalik dan ditampilkan di digit 5.

```
    ;-----------------------------------------------------------
    ;display hours on MAX7219
    ;------------------------
    MOV   R28, R31
    ANDI  R28, 0x0F
    LDI   R17, 07
    RCALL binary2decimal  ;convert & display LSD of hours
    MOV   R28, R31
    ANDI  R28, 0x30
    SWAP  R28
    LDI   R17, 8
    RCALL binary2decimal  ;convert & display MSD of hours
;-----------------------------------------------------------
    RCALL delay_ms
ext:RJMP  again
```

Fungsi dari kode ini adalah untuk menunjukkan nilai jam pada MAX7219. Angka satuan diambil dari R31 dengan melakukan operasi ANDI 0x0F dan ditampilkan pada digit 7, sementara angka puluhan diambil dengan ANDI 0x30, kemudian diswap dan ditampilkan di digit 8.

```
;============================================================
; External Interrupt 0 Vector (INT0)
;============================================================
.global INT0_vect
INT0_vect:
    PUSH  R16
    IN    R16, SREG        ; Simpan Status Register
    PUSH  R16

    ; Set flag untuk menyimpan data ke EEPROM
    LDI   R16, 1
    MOV   R21, R16         ; Set flag langsung ke R21

    ; Debounce delay sederhana
    PUSH  R17
    LDI   R17, 0xFF
debounce_loop:
    DEC   R17
    BRNE  debounce_loop
    POP   R17

    POP   R16
    OUT   SREG, R16        ; Kembalikan Status Register
    POP   R16
    RETI                   ; Return from interrupt
```

Tujuan dari kode ini adalah untuk mengelola interupsi eksternal INT0. Ketika interupsi berlangsung, kode akan menyimpan kondisi register, men-set flag pada R21 sebagai indikasi bahwa data akan disimpan ke EEPROM, setelah itu memberikan waktu jeda sederhana untuk menghindari kesalahan sebelum mengembalikan kondisi dan keluar dari interupsi.

```
;============================================================
;I2C subroutines
;============================================================
I2C_init:
;--------
    LDI   R21, 0
    STS   TWSR, R21       ;prescaler = 0
    LDI   R21, 12         ;division factor = 12
    STS   TWBR, R21       ;SCK freq = 400kHz
    LDI   R21, (1<<TWEN)
    STS   TWCR, R21       ;enable TWI
    RET
;============================================================
I2C_START:
    LDI   R21, (1<<TWINT)|(1<<TWSTA)|(1<<TWEN)
    STS   TWCR, R21       ;transmit START condition
    ;-----------------------------------------------------------
wt1:LDS   R21, TWCR
    SBRS  R21, TWINT      ;TWI interrupt = 1?
    RJMP  wt1             ;no, wait for end of transmission
    ;-----------------------------------------------------------
    RET
;============================================================
I2C_write:
    STS   TWDR, R27       ;copy SLA+W into data register
    LDI   R21, (1<<TWINT)|(1<<TWEN)
    STS   TWCR, R21       ;transmit SLA+W
    ;-----------------------------------------------------------
wt2:LDS   R21, TWCR
    SBRS  R21, TWINT
    RJMP  wt2             ;wait for end of transmission
    ;-----------------------------------------------------------
    RET
```

```
;============================================================
I2C_STOP:
    LDI   R21, (1<<TWINT)|(1<<TWSTO)|(1<<TWEN)
    STS   TWCR, R21       ;transmit STOP condition
    RET
;============================================================
I2C_read:
    LDI   R21, (1<<TWINT)|(1<<TWEA)|(1<<TWEN)
    STS   TWCR, R21         ;enable TWI & ACK
    ;-----------------------------------------------------------
wt3:LDS   R21, TWCR
    SBRS  R21, TWINT
    RJMP  wt3               ;wait for data byte to be read
    ;-----------------------------------------------------------
    LDS   R27, TWDR         ;store received byte
    RET
;============================================================
I2C_read_NACK:
    LDI   R21, (1<<TWINT)|(1<<TWEN)
    STS   TWCR, R21         ;enable TWI & ACK
    ;-----------------------------------------------------------
wt4:LDS   R21, TWCR
    SBRS  R21, TWINT
    RJMP  wt4               ;wait for data byte to be read
    ;-----------------------------------------------------------
    LDS   R27, TWDR         ;store received byte
    RET
```

Pada kode I2C ini berfungsi untuk berkomunikasi antara mikrokontroler dan perangkat luar. I2C_START mengeluarkan sinyal START untuk memulai proses komunikasi, I2C_write mentransfer data atau alamat, dan I2C_STOP menandai akhir dari komunikasi. I2C_read digunakan untuk mengambil data dengan ACK, sementara I2C_read_NACK mengambil data terakhir tanpa ACK. Semua subrutin ini menciptakan alur komunikasi I2C yang efektif.

```
;============================================================
;MAX7219 subroutines
;============================================================
SPI_MAX7219_init:
;----------------
.equ  SCK, 5
.equ  MOSI, 3
.equ  SS, 2
;-------------------------------------------------------------
      LDI   R17, (1<<MOSI)|(1<<SCK)|(1<<SS)
      OUT   DDRB, R17       ;set MOSI, SCK, SS as o/p
      ;-----------------------------------------------------
      LDI   R17, (1<<SPE)|(1<<MSTR)|(1<<SPR0)
      OUT   SPCR, R17       ;enable SPI as master, fsck=fosc/16
      ;-----------------------------------------------------
      LDI   R17, 0x0A       ;set segment intensity (0 to 15)
      LDI   R18, 8          ;intensity level = 8
      RCALL send_bytes      ;send command & data to MAX7219
      ;-----------------------------------------------------
      LDI   R17, 0x09       ;set decoding mode command
      LDI   R18, 0b11011011 ;decoding byte
      RCALL send_bytes      ;send command & data to MAX7219
      ;-----------------------------------------------------
      LDI   R17, 0x0B       ;set scan limit command
      LDI   R18, 0x07       ;8 digits connected to MAX7219
      RCALL send_bytes      ;send command & data to MAX7219
      ;-----------------------------------------------------
      LDI   R17, 0x0C       ;set turn ON/OFF command
      LDI   R18, 0x01       ;turn ON MAX7219
      RCALL send_bytes      ;send command & data to MAX7219
      ;-----------------------------------------------------
      RET
```

```
;============================================================
send_bytes:
      CBI   PORTB, SS       ;enable slave device MAX7219
      OUT   SPDR, R17       ;transmit command
      ;-----------------------------------------------------
l12:  IN    R19, SPSR
      SBRS  R19, SPIF       ;wait for byte transmission
      RJMP  l12             ;to complete
      ;-----------------------------------------------------
      OUT   SPDR, R18       ;transmit data
      ;-----------------------------------------------------
l13:  IN    R19, SPSR
      SBRS  R19, SPIF       ;wait for byte transmission
      RJMP  l13             ;to complete
      ;-----------------------------------------------------
      SBI   PORTB, SS       ;disable slave device MAX7219
      RET
;============================================================
MAX7219_disp_text:
;-----------------
      LDI   R17, 0x03       ;select digit 2
      LDI   R18, 0x01       ;data = dash
      RCALL send_bytes      ;send command & data to MAX7219
      ;-----------------------------------------------------
      LDI   R17, 0x06       ;select digit 5
      LDI   R18, 0x01       ;data = dash
      RCALL send_bytes      ;send command & data to MAX7219
      RET
```

Pada bagian kode ini mencakup subrutin untuk mengendalikan tampilan 7-segmen yang menggunakan MAX7219 melalui SPI. Fungsi SPI_MAX7219_init yakni akan mengonfigurasi pin SPI, mengaktifkan mode master pada SPI, dan mengirimkan pengaturan awal kepada MAX7219. Subrutin send_bytes bertugas untuk mengirimkan kombinasi perintah dan data. MAX7219_disp_text menampilkan simbol "-" pada digit kedua dan kelima.

binary2decimal mengubah nilai biner yang berada di R28 menjadi format desimal (ratusan, puluhan, satuan) dan mengirimkan hasilnya ke tampilan.

```
;============================================================
; Fungsi untuk selalu menyimpan waktu RTC ke EEPROM alamat 0x0000
;============================================================
save_rtc_to_eeprom:
    PUSH  R16
    PUSH  R17
    PUSH  R18

    ; Save hours (R31) to EEPROM address 0x0000
    ; Tunggu hingga EEPROM siap
rtc_wait1:
    SBIC  EECR, 1           ; Skip if EEWE bit is clear (EEPROM ready)
    RJMP  rtc_wait1         ; Wait until EEPROM ready

    ; Setup alamat dan data
    LDI   R18, 0x00         ; High byte of EEPROM address
    LDI   R17, 0x00         ; Low byte of EEPROM address (0x0000)
    OUT   EEARH, R18        ; Set EEPROM address high byte
    OUT   EEARL, R17        ; Set EEPROM address low byte
    MOV   R16, R31          ; Data to write (hours)
    OUT   EEDR, R16         ; Set EEPROM data register
    SBI   EECR, 2           ; Set EEMWE bit (Master Write Enable)
    SBI   EECR, 1           ; Set EEWE bit (Write Enable)
```

```
    ; Save minutes (R30) to EEPROM address 0x0001
rtc_wait2:
    SBIC  EECR, 1
    RJMP  rtc_wait2

    LDI   R18, 0x00
    LDI   R17, 0x01         ; Address 0x0001
    OUT   EEARH, R18
    OUT   EEARL, R17
    MOV   R16, R30          ; Minutes
    OUT   EEDR, R16
    SBI   EECR, 2
    SBI   EECR, 1

    ; Save seconds (R28) to EEPROM address 0x0002
rtc_wait3:
    SBIC  EECR, 1
    RJMP  rtc_wait3

    LDI   R18, 0x00
    LDI   R17, 0x02         ; Address 0x0002
    OUT   EEARH, R18
    OUT   EEARL, R17
    MOV   R16, R28          ; Seconds
    OUT   EEDR, R16
    SBI   EECR, 2
    SBI   EECR, 1

    POP   R18
    POP   R17
    POP   R16
    RET
```

Fungsi **save_rtc_to_eeprom** berperan guna untuk menyimpan informasi waktu dari RTC ke EEPROM secara bertahap. Jam (R31) disimpan di alamat EEPROM 0x0000, menit (R30) di 0x0001, dan detik (R28) di 0x0002. Sebelum melakukan setiap penulisan, sistem memverifikasi kesiapan EEPROM dengan memeriksa bit EEWE. Setelah itu EEPROM siap, alamat diatur, data dimasukkan ke dalam register EEDR, dan penulisan diaktifkan dengan mengatur bit EEMWE terlebih dahulu kemudian EEWE. Subrutin ini memastikan bahwa waktu yang tercatat di RTC tetap terjaga meskipun sistem dimatikan.

```
;============================================================
; Fungsi untuk menyimpan waktu RTC ke alamat 0x0008 saja
; saat interrupt INT0 terpicu
;============================================================
save_to_EEPROM:
    PUSH  R16
    PUSH  R17
    PUSH  R18

    ; Simpan detik (R28) ke alamat tetap 0x0008
    SBIC  EECR, 1
    RJMP  .-4

    LDI   R18, 0x00       ; High byte of EEPROM address
    LDI   R17, 0x08       ; Low byte of EEPROM address (0x0008)
    OUT   EEARH, R18
    OUT   EEARL, R17
    MOV   R16, R28        ; Data to write (seconds)
    OUT   EEDR, R16
    SBI   EECR, 2
    SBI   EECR, 1

    ; Simpan menit (R30) ke alamat tetap 0x0009
    SBIC  EECR, 1
    RJMP  .-4

    LDI   R18, 0x00
    LDI   R17, 0x09       ; Alamat tetap 0x0009
    OUT   EEARH, R18
    OUT   EEARL, R17
    MOV   R16, R30        ; Data to write (minutes)
    OUT   EEDR, R16
    SBI   EECR, 2
    SBI   EECR, 1
```

```
; Simpan jam (R31) ke alamat tetap 0x000A
SBIC  EECR, 1
RJMP  .-4

LDI   R18, 0x00
LDI   R17, 0x0A       ; Alamat tetap 0x000A
OUT   EEARH, R18
OUT   EEARL, R17
MOV   R16, R31        ; Data to write (hours)
OUT   EEDR, R16
SBI   EECR, 2
SBI   EECR, 1

; Tampilkan indikator bahwa data telah disimpan
RCALL blink_display

POP   R18
POP   R17
POP   R16
RET
```

Fungsi **save_to_EEPROM** bertugas untuk merekam informasi waktu dari RTC ke alamat EEPROM permanen ketika interrupt eksternal INT0 diaktifkan. Nilai detik (R28) disimpan di alamat 0x0008, menit (R30) disimpan di 0x0009, dan jam (R31) ditempatkan di 0x000A. Dan setiap kali data ditulis, proses ini menunggu hingga EEPROM siap, kemudian mengatur alamat serta data sebelum memulai kegiatan penulisan. Setelah seluruh data disimpan, fungsi blink_display dipanggil sebagai tanda bahwa penyimpanan telah selesai.

```
;================================================================
; Helper subroutine to wait until EEPROM is ready for writing
;================================================================
wait_eeprom_ready:
    SBIC  EECR, 1          ; Skip if EEWE bit is cleared (EEPROM not busy)
    RJMP  wait_eeprom_ready ; EEPROM is busy, wait
    RET
```

Subrutin **wait_eeprom_ready** berfungsi untuk menunggu sampai EEPROM dapat melakukan operasi penulisan. Subrutin ini akan terus memantau bit EEWE dalam register EECR, dan hanya akan keluar dari loop ketika bit tersebut sudah dalam keadaan clear (yang menunjukkan bahwa EEPROM tidak sedang sibuk). Setelah EEPROM siap, subrutin akan melakukan pengembalian (RET). Fungsinya adalah untuk mencegah penulisan ketika EEPROM masih dalam proses menangani data yang ada sebelumnya.

```
;============================================================
; Fungsi untuk membandingkan seconds di alamat 0x0002 dengan 0x0008
; dan mengaktifkan PC2 HANYA jika 0x0002 lebih besar minimal 10 dari 0x0008
;============================================================
compare_eeprom_seconds:
    PUSH  R16
    PUSH  R17
    PUSH  R18
    PUSH  R19

    ; Set PC2 sebagai output terlebih dahulu
    SBI   DDRC, 2         ; Set PC2 sebagai output

    ; Baca detik saat ini dari alamat 0x0002
    RCALL wait_eeprom_ready

    LDI   R18, 0x00
    LDI   R17, 0x02       ; Alamat 0x0002
    OUT   EEARH, R18
    OUT   EEARL, R17
    SBI   EECR, 0         ; Baca EEPROM
    IN    R16, EEDR       ; R16 = detik saat ini (0x0002)

    ; Baca detik dari alamat 0x0008
    RCALL wait_eeprom_ready

    LDI   R18, 0x00
    LDI   R17, 0x08       ; Alamat 0x0008
    OUT   EEARH, R18
    OUT   EEARL, R17
    SBI   EECR, 0         ; Baca EEPROM
    IN    R17, EEDR       ; R17 = detik referensi (0x0008)
```

```
; Konversi nilai BCD ke desimal untuk perbandingan yang akurat

; Konversi R16 (detik saat ini dari 0x0002) ke desimal
MOV   R18, R16        ; Backup nilai BCD dari 0x0002
ANDI  R18, 0xF0       ; Isolasi digit puluhan
SWAP  R18             ; Geser ke posisi satuan
LDI   R19, 10
MUL   R18, R19        ; R18 * 10
MOV   R18, R0         ; R18 = puluhan * 10
MOV   R19, R16        ; Gunakan nilai asli
ANDI  R19, 0x0F       ; Isolasi digit satuan
ADD   R18, R19        ; R18 = nilai desimal dari 0x0002

; Konversi R17 (detik referensi dari 0x0008) ke desimal
MOV   R19, R17        ; Backup nilai BCD dari 0x0008
ANDI  R19, 0xF0       ; Isolasi digit puluhan
SWAP  R19             ; Geser ke posisi satuan
LDI   R16, 10
MUL   R19, R16        ; R19 * 10
MOV   R19, R0         ; R19 = puluhan * 10
MOV   R16, R17        ; Gunakan nilai asli
ANDI  R16, 0x0F       ; Isolasi digit satuan
ADD   R19, R16        ; R19 = nilai desimal dari 0x0008

; Sekarang R18 = nilai desimal 0x0002, R19 = nilai desimal 0x0008

; KASUS KHUSUS: Penanganan jika salah satu nilai tidak valid
CPI   R17, 0xFF       ; Cek jika 0x0008 tidak valid (0xFF)
BREQ  turn_off        ; Jika tidak valid, matikan LED
```

```
; Sekarang R18 = nilai desimal 0x0002, R19 = nilai desimal 0x0008

; KASUS KHUSUS: Penanganan jika salah satu nilai tidak valid
CPI   R17, 0xFF       ; Cek jika 0x0008 tidak valid (0xFF)
BREQ  turn_off        ; Jika tidak valid, matikan LED

; Hanya nyalakan lampu jika 0x0002 > 0x0008 + 10
; 1. Periksa apakah 0x0002 > 0x0008
CP    R19, R18        ; Bandingkan R19 (0x0008) dengan R18 (0x0002)
BRSH  turn_off        ; Jika 0x0008 >= 0x0002, matikan LED (tidak memenuhi syarat)

; 2. 0x0002 > 0x0008, sekarang cek apakah selisihnya >= 10
MOV   R16, R18        ; R16 = nilai 0x0002
SUB   R16, R19        ; R16 = 0x0002 - 0x0008 (selisih)

; Kasus khusus: jika hasil pengurangan negatif (overflow detik)
BRCS  handle_overflow

; Perbandingan normal
CPI   R16, 10         ; Bandingkan selisih dengan 10
BRSH  turn_on         ; Jika selisih >= 10, nyalakan LED
RJMP  turn_off        ; Jika selisih < 10, matikan LED
```

```
handle_overflow:
    ; Jika terjadi overflow (mis. 0x0002=5, 0x0008=55)
    ; Hitung selisih dengan mempertimbangkan lingkaran 60 detik
    LDI   R17, 60
    ADD   R16, R17        ; R16 = (0x0002 - 0x0008) + 60
    CPI   R16, 10         ; Bandingkan dengan 10
    BRSH  turn_on         ; Jika selisih >= 10, nyalakan LED
    RJMP  turn_off        ; Jika tidak, matikan LED

turn_on:
    ; Selisih >= 10 dan 0x0002 > 0x0008, nyalakan LED
    SBI   PORTC, 2        ; Set PC2 high (LED ON)
    CBI   PORTC, 1        ; Set PC1 low (NOT dari PC2)
    RJMP  exit_compare

turn_off:
    ; Matikan LED jika kondisi tidak terpenuhi
    CBI   PORTC, 2        ; Set PC2 low (LED OFF)
    SBI   PORTC, 1        ; Set PC1 high (NOT dari PC2)

exit_compare:
    POP   R19
    POP   R18
    POP   R17
    POP   R16
    RET
```

Fungsi **compare_eeprom_seconds** berfungsi untuk membandingkan nilai detik pada EEPROM yang terletak di alamat 0x0002 dengan yang ada di 0x0008. Dan nantinya apabila nilai pada 0x0002 lebih besar setidaknya 10 detik, maka pin PC2 akan diaktifkan (LED menyala), sementara PC1 akan dimatikan. Sebaliknya, jika kondisi itu tidak terpenuhi, PC2 akan dimatikan dan PC1 dinyalakan. Nilai BCD dari kedua alamat tersebut dikonversi menjadi desimal untuk perbandingan yang lebih tepat, termasuk penanganan saat detik beralih dari 59 ke 0. Fungsi ini memungkinkan sistem untuk membaca waktu dan menyalakan LED apabila selisih waktu cukup signifikan

```
;============================================================
; Blink display to indicate data saved
;============================================================
blink_display:
    ; Save used registers
    PUSH  R17
    PUSH  R18

    ; Turn OFF display
    LDI   R17, 0x0C       ; Set shutdown register
    LDI   R18, 0x00       ; Turn OFF (shutdown mode)
    RCALL send_bytes

    ; Short delay
    RCALL delay_ms

    ; Turn ON display
    LDI   R17, 0x0C       ; Set shutdown register
    LDI   R18, 0x01       ; Turn ON (normal operation)
    RCALL send_bytes

    ; Short delay
    RCALL delay_ms

    ; Turn OFF again
    LDI   R17, 0x0C
    LDI   R18, 0x00
    RCALL send_bytes
```

```
    ; Short delay
    RCALL delay_ms

    ; Turn OFF again
    LDI   R17, 0x0C
    LDI   R18, 0x00
    RCALL send_bytes

    ; Short delay
    RCALL delay_ms

    ; Turn ON again
    LDI   R17, 0x0C
    LDI   R18, 0x01
    RCALL send_bytes

    POP   R18
    POP   R17
    RET
```

Fungsi **blink_display** berfungsi untuk memberikan tanda visual bahwa data sudah berhasil disimpan, dengan cara membuat layar berkedip. Fungsi ini menyimpan register yang digunakan, lalu mengatur tampilan ke mode mati dan hidup kembali secara bergantian dua kali. Setiap perubahan status diikuti dengan jeda singkat menggunakan delay_ms, sehingga tampilan tampak berkedip. Ini memberikan sinyal visual kepada pengguna bahwa proses penyimpanan telah tuntas.

```
;============================================================
;delay subroutines
;============================================================
delay_ms:
    LDI   R22, 0xFF
a1: DEC   R22
    NOP
    BRNE  a1
    RET
;============================================================
delay_3s:
     LDI   R21, 255
16:  LDI   R22, 255
17:  LDI   R23, 255
18:  DEC   R23
     BRNE  18
     DEC   R22
     BRNE  17
     DEC   R21
     BRNE  16
     RET
```

Pada bagian kode ini menjelaskan dua subrutin penundaan yang digunakan untuk memberikan waktu jeda dalam pelaksanaan program. **Subrutin delay_ms** menciptakan penundaan singkat sekitar 1 milidetik (tergantung pada kecepatan jam) dengan cara mengurangi nilai pada register R22 dari 0xFF hingga 0. Di sisi lain, delay_3s dirancang untuk menciptakan penundaan sekitar 3 detik melalui tiga loop bertingkat yang secara bertahap mengurangi nilai R23, R22, dan R21, dari 255 hingga 0. Kombinasi dari loop ini menghasilkan jeda yang cukup lama sebelum program melanjutkan ke bagian berikutnya.

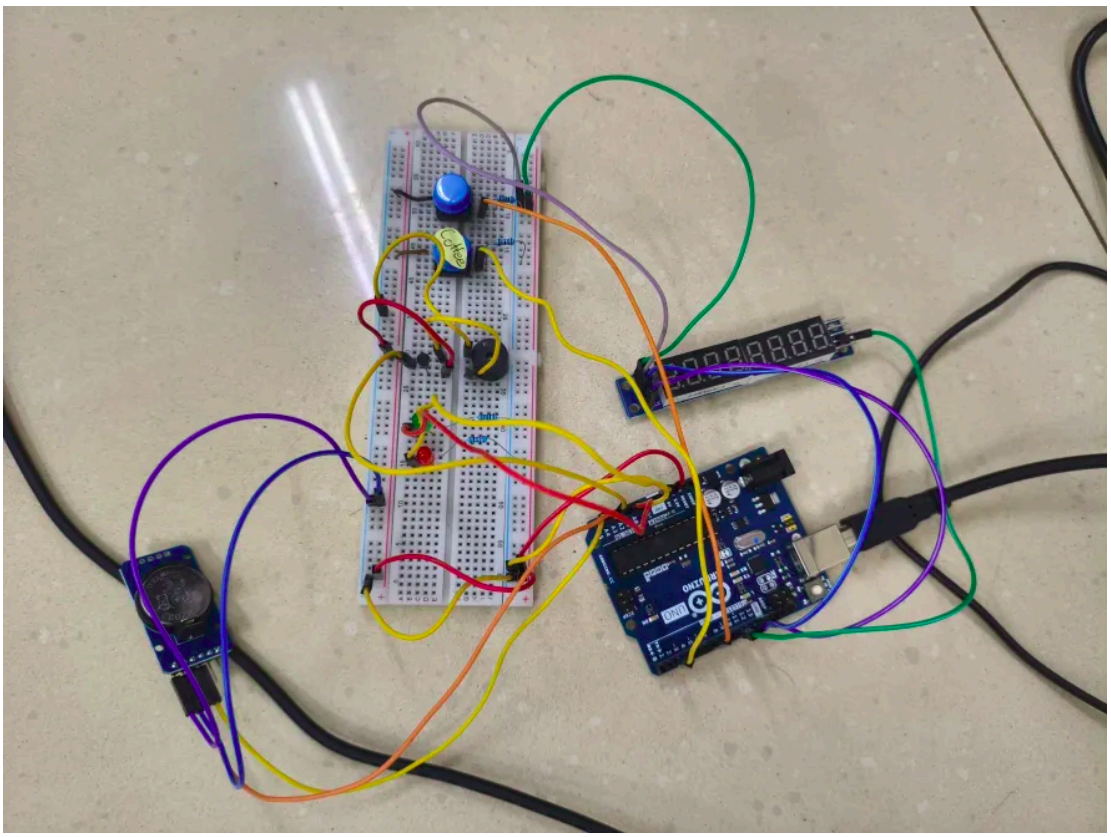
CHAPTER 4

TEST RESULT AND PERFORMANCE

4.1 TEST RESULTS

Pada tahap pengujian ini, rangkaian elektronika Student Corner e-Timer telah dibangun secara fisik berdasarkan rancangan yang sebelumnya dibuat menggunakan software simulasi Proteus. Rangkaian menggunakan Arduino Uno sebagai mikrokontroler utama, modul RTC DS3231 untuk pembacaan waktu nyata, MAX7219 sebagai driver display seven-segment matrix, serta buzzer dan tombol sebagai antarmuka pengguna. Kode program ditulis dalam bahasa Assembly (melalui environment Arduino IDE) dan diunggah ke Arduino Uno untuk mengendalikan alur kerja perangkat. Pengujian dilakukan untuk memastikan bahwa setiap komponen dapat bekerja sesuai spesifikasi yang direncanakan. Fokus utama pengujian adalah pada fungsi dasar perangkat, yaitu membaca waktu, menampilkannya, menyimpan nilai waktu ke EEPROM ketika tombol ditekan, serta mengaktifkan buzzer selama 10 detik sebelum kembali nonaktif. Dalam pengujian ini, timer sengaja diatur ke 10 detik agar memudahkan pengamatan terhadap respons sistem secara langsung.

Rangkaian ini dibangun dengan komponen-komponen hardware yang di atas, dan kode assembly yang dibuat untuk rangkaian elektronika ini dimasukkan ke dalam Arduino Uno dengan menghubungkan Arduino Uno ke sebuah alat elektronik yang menjalankan program Arduino IDE. Tujuan dari melakukan hal ini adalah untuk menunjukkan cara kerja alat elektronik ini dalam dunia nyata.

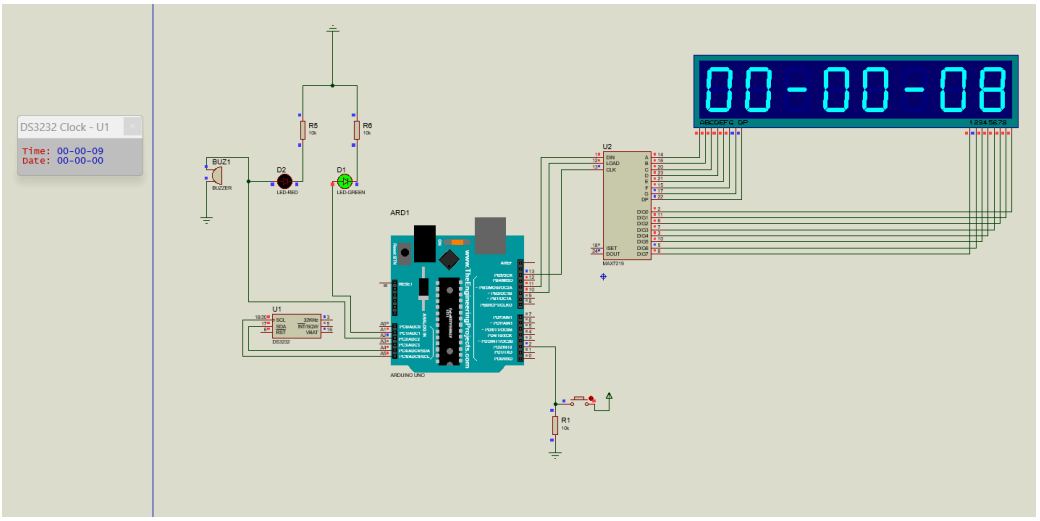


Sekarang, kita bisa mulai menguji kerja Student Corner e-Timer di dunia nyata.

Timer ini dikendalikan dengan menekan button, yang mengatur seberapa lama sebelum buzzer mengeluarkan suara, dan juga mematikan buzzer ketika buzzer itu bersuara. Percobaan dengan mengatur timer diatur dengan 10 detik sebagai simulasi proyek bekerja sesuai implementasi.

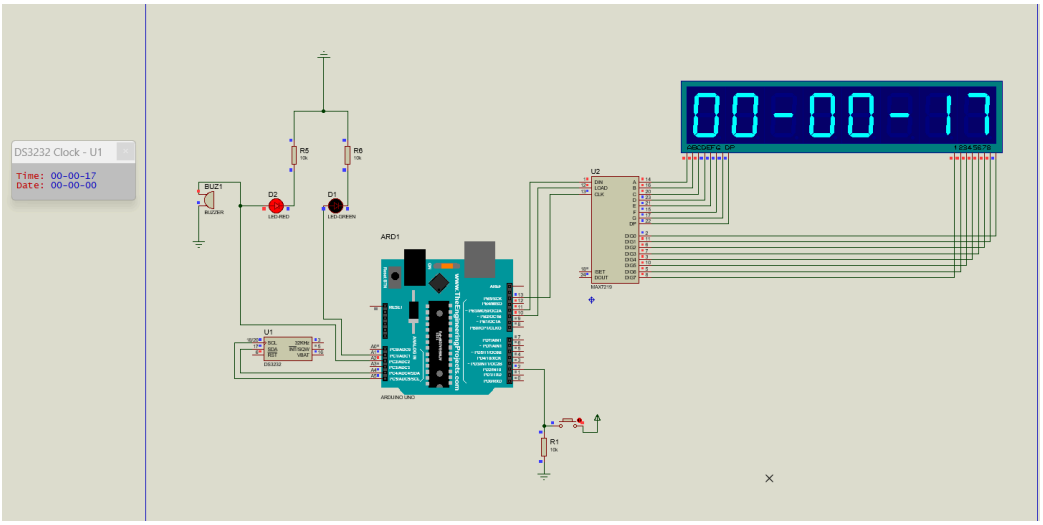
Percobaan dilakukan dalam proteus untuk mencoba saat menekan *button* yang akan dibandingkan waktu dari awal menekan sampai *limit* (10 Sekon) dengan Modul DS3231 RTC, antara lain:

a. Akses Diberikan



Pada rangkaian tersebut mahasiswa belum memiliki limit waktu penggunaan yang ditandai dengan LED Green menyala sebagai indikator pemberian akses, kemudian mahasiswa menekannya dan mengirim data ke EEPROM untuk waktu.

b. Akses Limit



Setelah *compare* waktu sejak ditekan telah melewati 10 sekon maka LED Merah menyala yang mengindikasikan waktu sudah *limit* dan *buzzer* menyala, hal ini terlihat dari MAX7219 pada detik 17 karena menekan pertama pada detik 7.

4.2 PERFORMANCE

Berdasarkan pengujian yang dilakukan, proyek ini menunjukkan performa yang baik d. Berdasarkan akurasi waktu, modul RTC DS3231 bisa menjaga ketepatan waktu dengan sangat baik, deviasi yang hampir tidak terdeteksi selama pengujian. Sistem penyimpanan data ke EEPROM bekerja dengan konsisten, baik dalam mode penyimpanan kontinu maupun penyimpanan berbasis *interrupt*, meskipun terdapat sedikit delay sekitar 3.4ms per operasi tulis yang merupakan karakteristik bawaan EEPROM ATmega328P.

Selain itu, modul MAX7219 memberikan output yang stabil, tetapi terdapat sedikit flicker saat simulasi karena timing komunikasi SPI yang perlu disesuaikan. Sistem interrupt berfungsi sesuai ekspektasi dengan latency respon sekitar 25 clock cycle. Buzzer memberikan feedback audio yang jelas dengan durasi aktif yang tepat sesuai logika program.

Penggunaan memori program yang hanya 3% dari kapasitas total, penggunaan RAM yang minimal (sekitar 30 byte) juga memastikan tersedianya cukup memori untuk pengembangan fitur tambahan di masa depan. Secara keseluruhan, proyek ini telah memenuhi persyaratan.

Sistem interupsi menggunakan INT0 juga berfungsi stabil, saat tombol ditekan maka data waktu langsung tersimpan dan jika buzzer aktif, maka suara buzzer langsung berhenti. Latensi antara pemicuan interupsi dan eksekusi penyimpanan ke EEPROM berkisar antara 5–10 ms, dengan delay tambahan untuk mencegah bouncing pada tombol fisik. Meskipun mengganti *keypad* menjadi *button* dan penggantian media tampilan dari LCD ke MAX7219, proyek tetap bisa bekerja dengan baik.

CHAPTER 5

CONCLUSION AND DEVELOPMENT

Perancangan Student Corner e-Timer telah berhasil diimplementasikan menggunakan *button* yang akan *interrupt* jika mahasiswa menekan *button* tersebut (INT0) maka data waktu disimpan ke EEPROM pada *address* kelipatan 0x0008, Modul RTC akan membandingkan selisih waktu antara data terakhir dengan saat ini, jika selisih lebih dari 10 detik, maka program akan mengaktifkan sinyal pada PC2, yaitu LED Merah dan Buzzer yang menyala sebagai indikasi waktu sudah *limit*, tetapi jika LED Hijau menyala maka akan memberikan akses kepada mahasiswa untuk menekan *button*. RTC DS3231 untuk membaca waktu yang secara otomatis disimpan ke EEPROM dan tampilkan di MAX7219.

Sebagai pengembangan pada masa depan, praktikan mengharapkan bisa menggunakan Keypad 4x4 ataupun RFID yang bisa membaca identitas mahasiswa dan menyimpannya di EEPROM secara efisien. Praktikkan juga mengharapkan kedepannya program bisa diperbaiki agar lebih baik.

REFERENCES

[1] ATMEGA328P, https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf (accessed May 18, 2025).

[2] Analog, https://www.analog.com/media/en/technical-documentation/data-sheets/ds3231.pdf (accessed May 18, 2025).

[3] MAX7219/max7221 - serially interfaced, 8-digit led ..., https://www.analog.com/media/en/technical-documentation/data-sheets/max7219-max7221.pdf (accessed May 18, 2025).

[4] "Assembly via Arduino - Programming EEPROM," *Blogspot.com*, 2021. https://akuzechie.blogspot.com/2021/11/assembly-via-arduino-programming-eeprom.html (accessed May 18, 2025).

[5] "Assembly via Arduino - Programming DS3231 RTC," *Blogspot.com*, 2022. https://akuzechie.blogspot.com/2022/01/assembly-via-arduino-programming-ds3231.html (accessed May 18, 2025).

[6] "Assembly via Arduino - Interfacing Keypads," *Blogspot.com*, 2021. https://akuzechie.blogspot.com/2021/11/assembly-via-arduino-interfacing-keypads.html (accessed May 18, 2025).