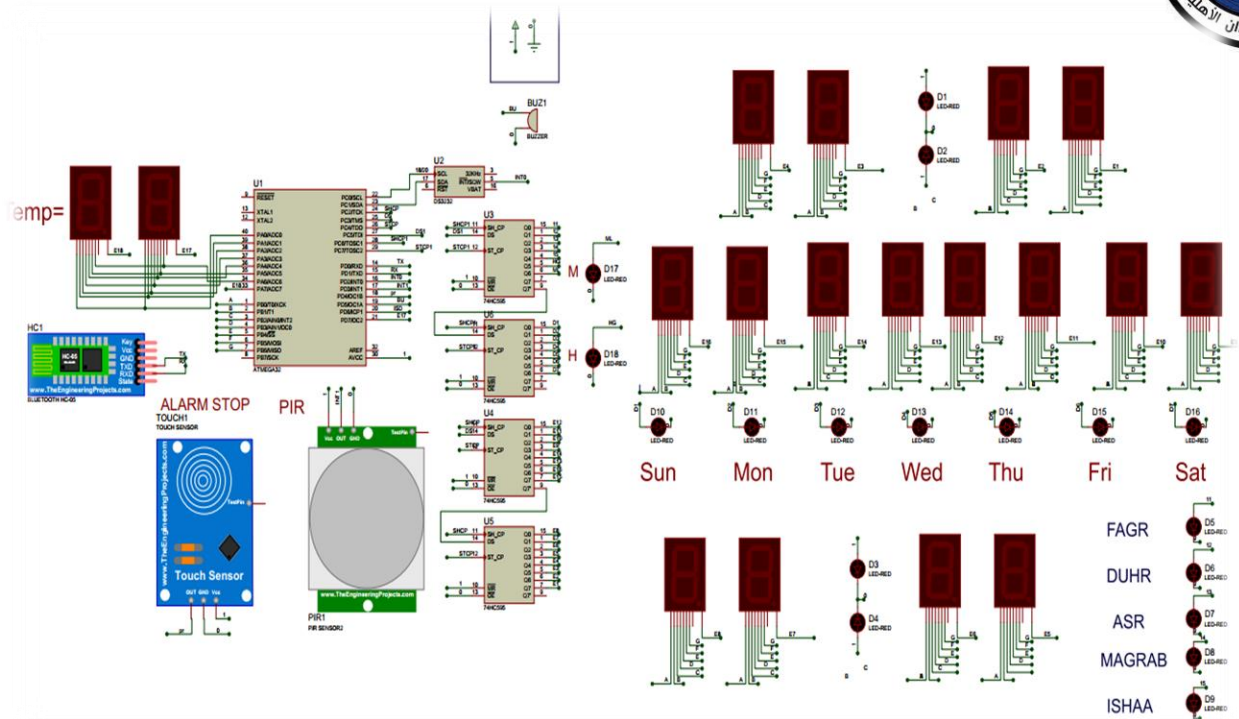


Smart Prayer Clock System



Names:

- زياد طارق أحمد.
- احمد محمد احمد فؤاد المنيلوي.
- أحمد سيف الإسلام أحمد حامد.
- أحمد ايمن سيد.
- ساره هشام حسن.

Under Supervision Of: Dr. Hossam Eldin Ali

Faculty Of Engineering
Intelligent Systems Engineering
Helwan National University
2025

Introduction: Smart Prayer Clock System

This project presents the design and implementation of a **Smart Prayer Clock** that combines both timekeeping and Islamic functionality with modern smart technologies. The system aims to provide an integrated, user-friendly solution for daily time management, religious observance, and environmental awareness.

Key Features:

1. Time and Date Display:

- Displays the **current time** (hours and minutes).
- Supports both **Gregorian and Hijri calendars** for accurate Islamic date tracking.

2. Prayer Times Display:

- Shows **daily Islamic prayer times** (Fajr, Dhuhr, Asr, Maghrib, and Isha).
- **Adhan audio** is played automatically at the time of each prayer using a built-in speaker module.

3. Temperature Monitoring:

- Displays the **current temperature from the RTC module** , offering real-time environmental information.

4. Bluetooth Connectivity:

- The system is integrated with **Bluetooth technology**, allowing remote interaction via a **mobile application**.
- Users can **set, update, and manage alarm times**, adjust prayer schedules, and configure other settings directly from the app.

5. Alarm System:

- Includes a **programmable alarm** that can be set and disabled using the mobile app.
- Also features a **capacitive touch button** to manually stop the alarm.

6. Real-Time Clock (RTC):

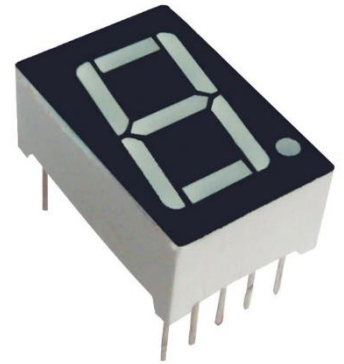
- Utilizes the **DS3231 RTC module** to maintain accurate time and date even during power outages.
- Also contain an embedded Temperature sensor .

7. Power Saving Mode with PIR Sensor:

- Equipped with a **PIR (Passive Infrared) motion sensor** that detects presence controlled through the app.
- The system enters a **low-power mode** when no motion is detected to conserve energy.

Components:

- (1) Atmega 32
- (18) 7-Segment (1inch & 0.8 inch)
- Shift registers 74HC595N
- RTC Module DS3231
- ISD Module & Speaker
- PIR Sensor
- Bluetooth Module (HC-05)
- Touch Sensor
- LEDs and Resistances
- Buzzer



Functionality:

1. ATmega32 Microcontroller

- Serves as the central processing unit of the project.
- Manages all system operations including display control, sensor input, timekeeping, and communication.

2. 7-Segment Displays (18 pieces - 1 inch & 0.8 inch)

- Used to display the current time (hours and minutes), Hijri and Gregorian dates, and temperature.

3. Shift Registers (74HC595N)

- Expands the number of output pins from the ATmega32.
- Enables control of multiple 7-segment displays with fewer microcontroller pins.

4. RTC Module (DS3231)

- Keeps accurate track of real-time clock and date.

- Retains time even when the main power is off, thanks to its backup battery.
- Also includes a temperature sensor used for environment monitoring.

5. ISD Module & Speaker

- Stores and plays the Adhan (Islamic call to prayer) sound at the correct prayer times.
- Connected to a speaker for clear audio output.

6. PIR Sensor

- Detects motion in the surrounding environment.
- disables display to save power when no one is nearby if I enabled the mode.

7. Bluetooth Module (HC-05)

- Provides wireless connectivity with a mobile application.
- Allows the user to set alarms, configure prayer times, control power modes, Time & Date ,offset of Hijri , Summer & Winter Time and more.

8. Touch Sensor

- Capacitive touch sensor used to manually stop the alarm or Adhan.

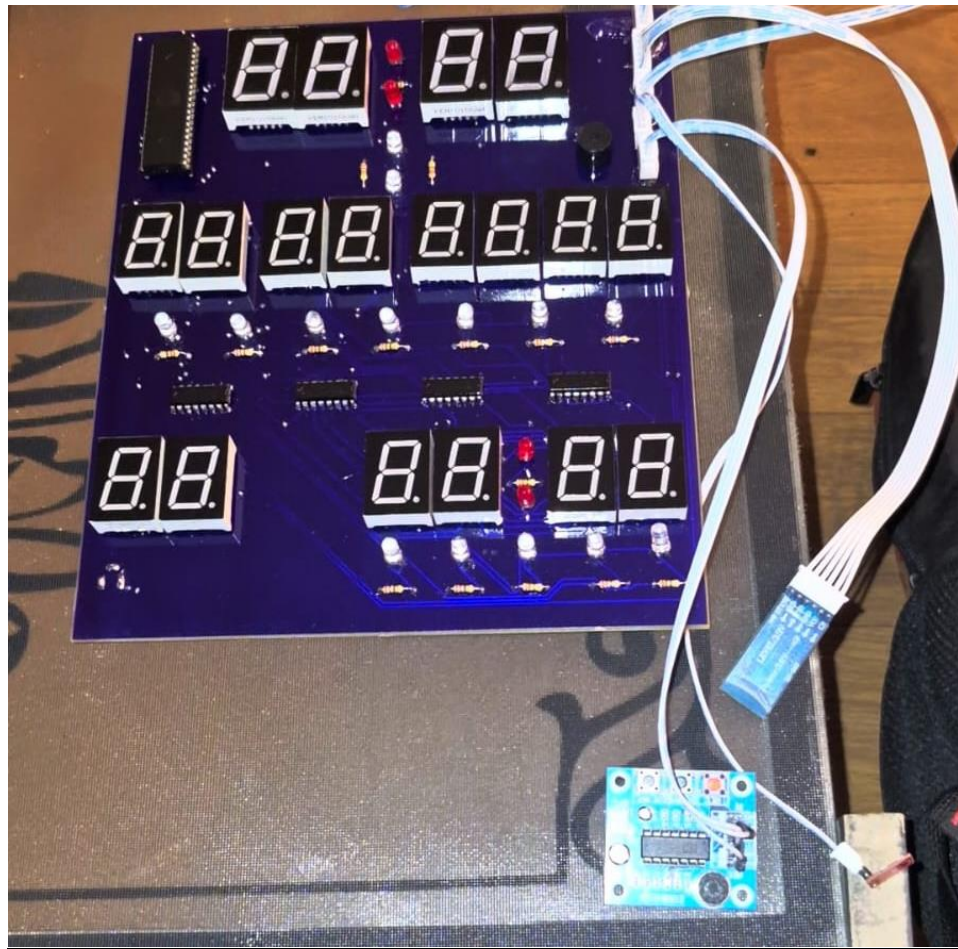
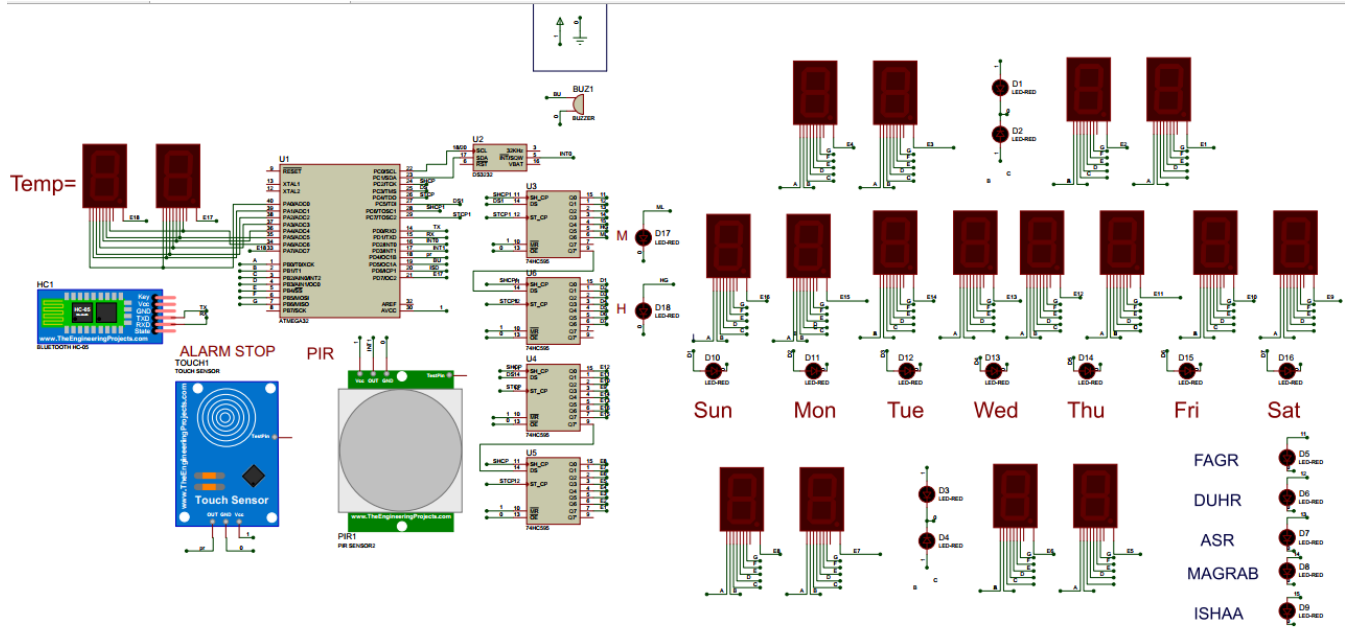
9. LEDs and Resistors

- LEDs are used to indicate status (e.g., days, hijri & miladi, prayers time).
- Resistors are used to limit current and protect components like LEDs and microcontroller pins.

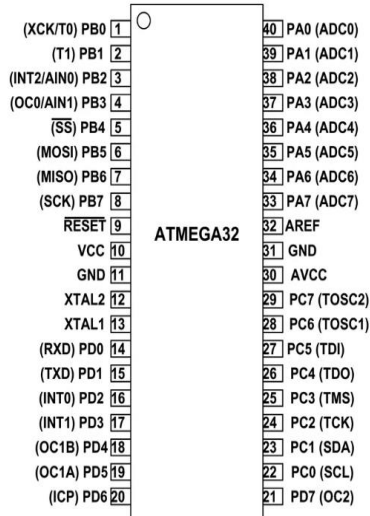
10. Buzzer

- Used to produce a simple sound alert for alarms or notifications.

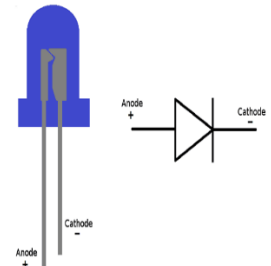
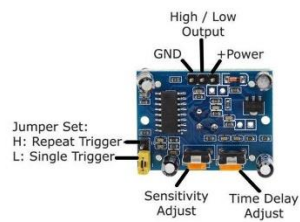
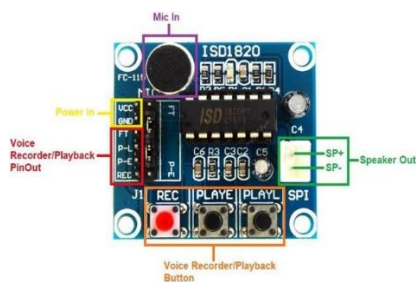
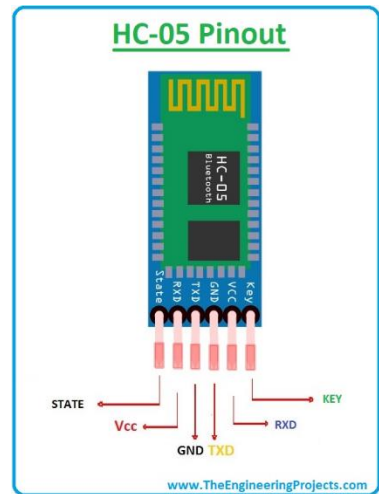
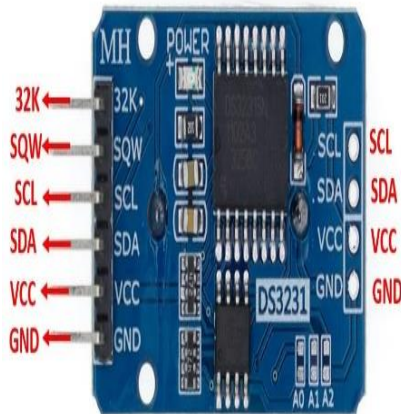
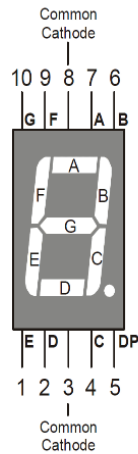
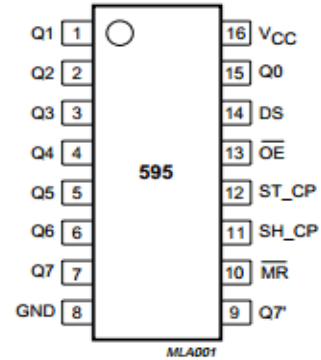
Circuit Diagram:

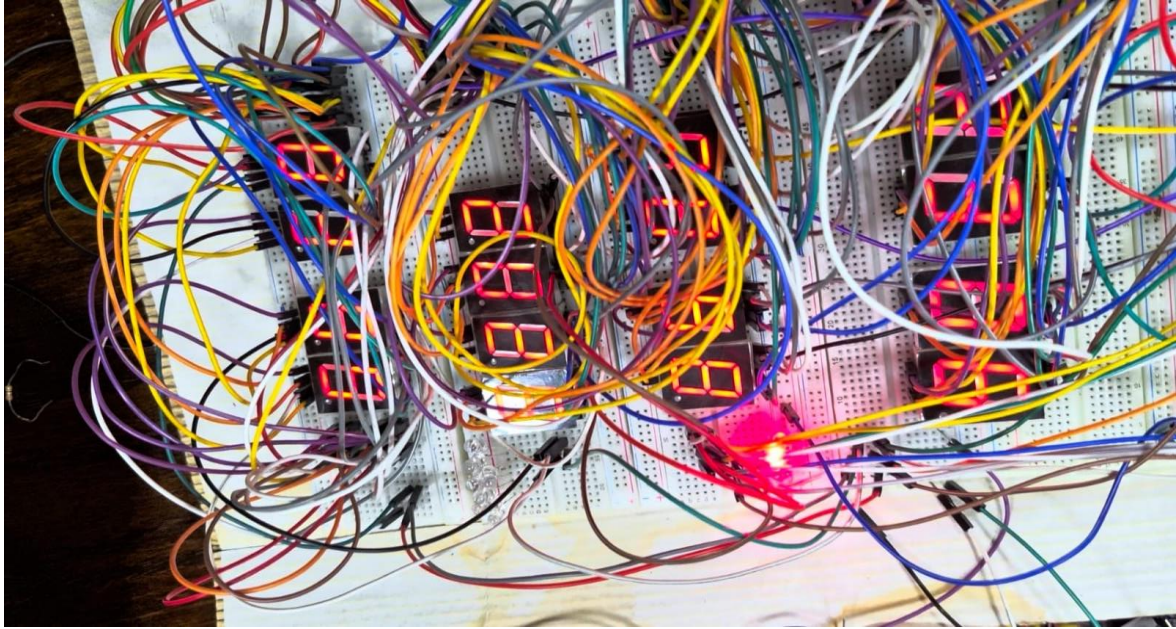


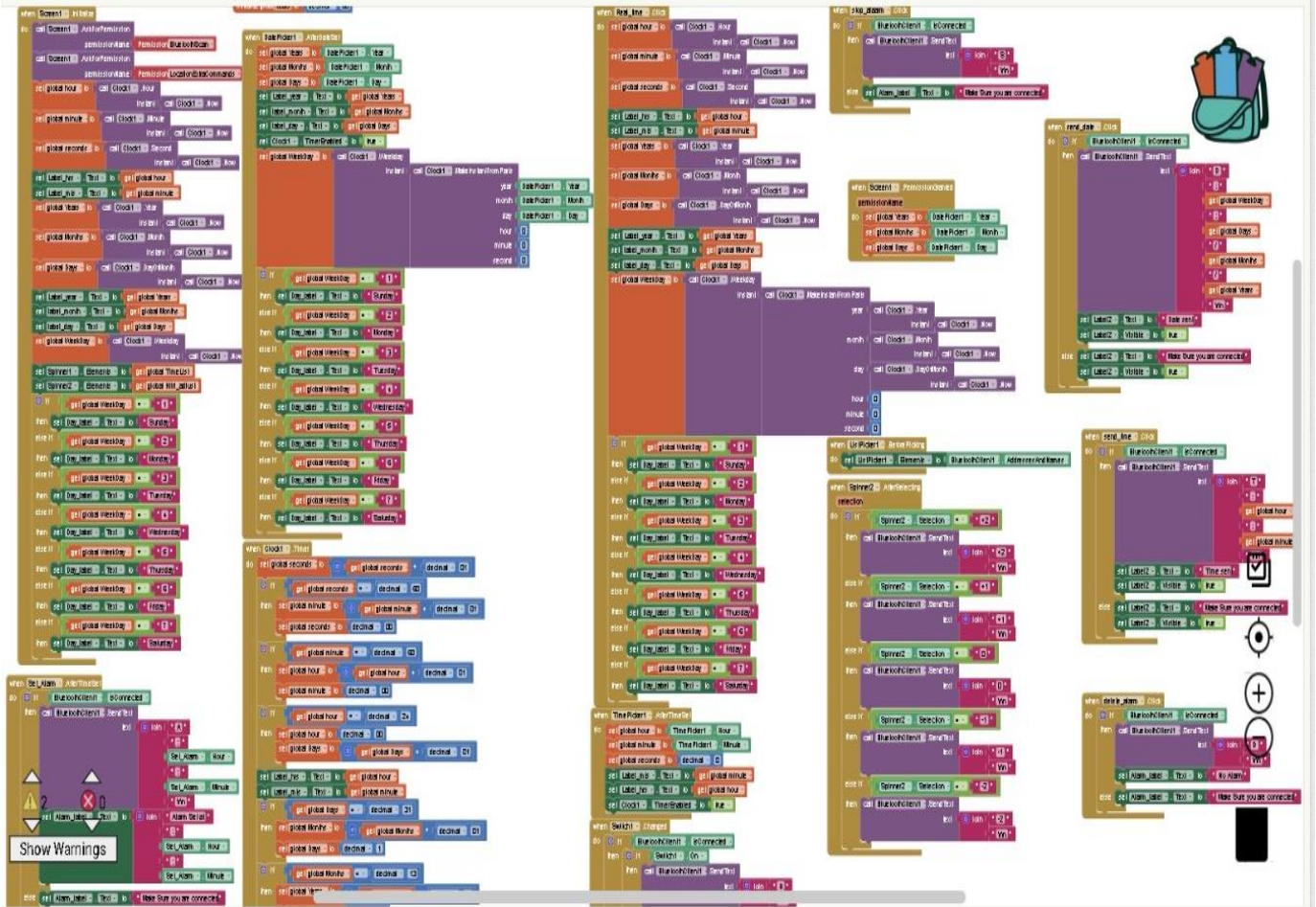
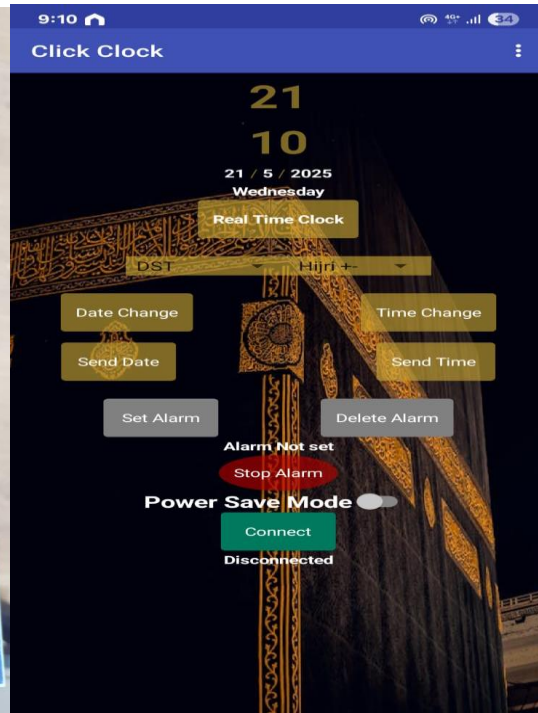
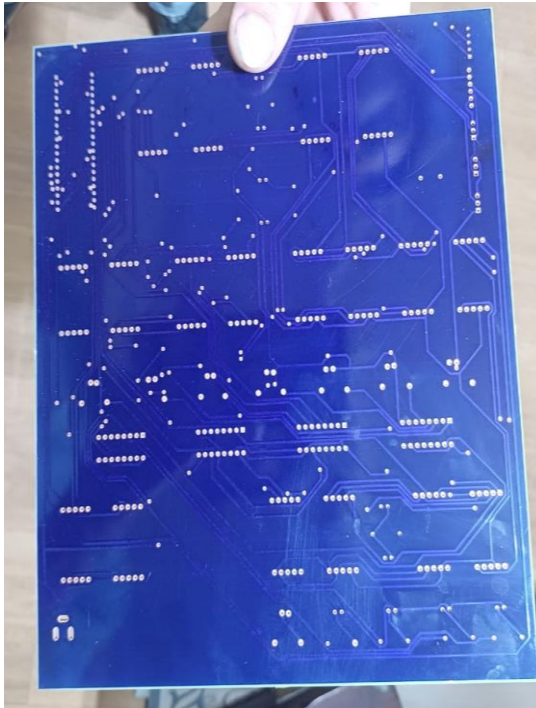
Datasheet:



74HC595







Solution 'Clock Project' (1 project)

Clock Project

Dependencies

Output Files

Libraries

DIO.c

DIO.h

DS3231.c

DS3231.h

EEPROM_driver.c

EEPROM_driver.h

HijriConverter.c

HijriConverter.h

I2C.c

I2C.h

main.c

prayerTimes.c

prayerTimes.h

seven_segment.c

seven_segment.h

SHIFTREGISTER.c

SHIFTREGISTER.h

std_macros.h

USART.c

USART.h

```

22 #include "DS3231.h"
23 #include "DIO.h"
24 #include "seven_segment.h"
25 #include "std_macros.h"
26 #include "SHIFTREGISTER.h"
27 #include "prayerTimes.h"
28 #include "USART.h"
29 #include "HijriConverter.h"
30 #include "EEPROM_driver.h"
31
32 // Hardware Configuration
33 #define I2C1230 PIN 6 // Pin connected to I2C1230 adnan prayer
34 #define LATITUDE 36.84430 // Cairo latitude
35 #define LONGITUDE 31.235712 // Cairo longitude
36
37 // EEPROM Addresses
38 #define ADDR_HIJRI_OFFSET 0x02
39 #define ADDR_TIME_OFFSET 0x01
40
41 // Type Definitions
42 #typedef struct {
43     uint8_t hour;
44     uint8_t minute;
45 } PrayerTime;
46
47 // Global Variables
48 volatile uint8_t current_adhan = -1; // Currently playing adhan (-1 means none)
49 volatile uint8_t current_digit = 0; // Currently displayed digit
50 volatile uint8_t digit_values[10]; // Values for all display digits
51 volatile uint8_t ms_counter = 0; // Millisecond counter
52 volatile uint8_t five_sec_counter = 0; // 5-second counter
53 volatile uint8_t motion_idle_counter = 0; // Motion idle time counter
54 volatile uint8_t system_makes = 1; // System makes state
55 volatile uint8_t current_prayer = 0; // Currently displayed prayer time
56 volatile uint8_t power_save_enabled = 0; // Power save mode status
57 int time_offset = -1; // Time offset for prayer times
58
59 // Time and date variables
60 uint8_t hours = 0, minutes = 0, seconds = 0;
61 uint8_t dayofweek = 0, day = 0, month = 0, year = 0;
62 uint8_t last_displayed_day = 0; // Last day prayer times were calculated
63
64 // Prayer times array
65 PrayerTime prayer_times[5];
66
67 // UART communication
68 char uart_buffer[20];
69 uint8_t uart_index = 0;
70
71 // Display control
72 int temp_digit = 0; // Temperature display digit toggle
73 uint8_t hijri_toggle = 0; // Hijri/day/gregorian display toggle
74 uint8_t hijri_day_offset_now = 0; // Hijri date adjustment
75 int temp; // Current temperature
76
77 // LED Patterns

```

```

// Digit mapping for multiplexing
const uint8_t digit_map[10] = {
    15, 14, 13, 12, 11, 10, 9, 8,
    7, 6, 5, 4, 3, 2, 1, 0
};

// Initialization Functions
void timer_init_cir_mode(void);
void hardware_init(void);

// Display Functions
void update_data_display(void);
void update_display_array(void);
void update_prayer_and_day_leds(void);
void clear_shiftregister(void);
void enable_display(void);

// Prayer Time Functions
void update_prayer_times_today(void);

// Communication Functions
void process_uart_command(void);

// Interrupt Service Routines
ISR(TIMER0_COMP_vect) {
    // Handle display multiplexing when system is awake
    if (system_makes) {
        uint8_t real_digit = digit_map[current_digit];
        uint8_t digit_mask = ~(1 << real_digit);
        shift_register_and_digit_select(digit_mask);
        seven_seg_write("B", digit_values[current_digit]);
        current_digit = (current_digit + 1) % 10;
    }

    // Update counters
    ms_counter++;
    five_sec_counter++;

    // Power save mode handling
    if (power_save_enabled) {
        if ((PINA & 0x003) & 0x003) { // Motion detected
            motion_idle_counter = 0;
            if (system_makes) {
                system_makes = 1;
                enable_display();
            }
        } else { // No motion
            if (system_makes && motion_idle_counter >= 60000) {
                system_makes = 0;
                clear_shiftregister();
            }
        }
    }

    // Stop buzzer command
    else if (strcmp(uart_buffer, "S") == 0) {
        digitalWrite(5, 0);
    }

    // Enable power save mode
    else if (strcmp(uart_buffer, "H") == 0) {
        power_save_enabled = 1;
        motion_idle_counter = 0;
    }

    // Disable power save mode
    else if (strcmp(uart_buffer, "P") == 0) {
        power_save_enabled = 0;
        system_makes = 1;
        enable_display();
    }

    // Hijri date adjustment commands
    else if (strcmp(uart_buffer, "i") == 0) {
        hijri_day_offset_now = 1;
        EEPROM_write(ADDR_HIJRI_OFFSET, (char)hijri_day_offset_now);
    } else if (strcmp(uart_buffer, "x") == 0) {
        hijri_day_offset_now = -1;
        EEPROM_write(ADDR_HIJRI_OFFSET, (char)hijri_day_offset_now);
    } else if (strcmp(uart_buffer, "2") == 0) {
        hijri_day_offset_now = -2;
        EEPROM_write(ADDR_HIJRI_OFFSET, (char)hijri_day_offset_now);
    }

    // Time offset command
    else if (strcmp(uart_buffer, "C") == 0) {
        time_offset = -1;
        EEPROM_write(ADDR_TIME_OFFSET, (char)time_offset);
        update_prayer_times_today();
    }
}

```

```

// Temperature display multiplexing
// Common cathode display handling
DIO_write("A", 7, 1); // Disable temperature tens digit
DIO_write("U", 7, 1); // Disable temperature units digit

// Alternate between temperature digits
if (temp_digit == 0) {
    seven_seg_write(temp % 10); // Display units
    DIO_write("U", 7, 0); // Enable units digit
} else {
    seven_seg_write(temp / 10); // Display tens
    DIO_write("A", 7, 0); // Enable tens digit
}
temp_digit ~ 1; // Toggle digit

// UART reception handling
if (CLOCK & 0x00000001) {
    char c = USART_receive_data();
    if (c == '\n') {
        uart_buffer[uart_index] = '\0';
        process_uart_command();
    } else if (uart_index < sizeof(uart_buffer) - 1) {
        uart_buffer[uart_index++] = c;
    }
}

ISR(INT0_vect) {
    // Trigger buzzer and wake up system if needed
    DIO_write("D", 5, 1);
    if (power_save_enabled && !system_makes) {
        system_makes = 1;
        enable_display();
    }
}

// ===== DISPLAY FUNCTIONS ===== */

void update_data_display(void) {
    uint8_t d, m;
    uint8_t full_year;

    if (hijri_toggle == 0) {
        // Gregorian display
        d = day;
        m = month;
        full_year = 2000 + year;
    } else {
        // Hijri display
        uint8_t h, hijri;
        connection_t c(2000 + year, month, day, 0, 0, 0, hijri);
    }
}

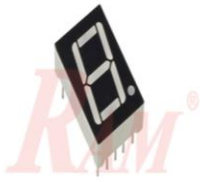
```

```

380 // Configure GPIO
381 DID_vsetPINDir('D', 3, 0); // PIR input
382 DID_vsetPINDir('D', 5, 1); // Buzzer output
383 DID_vsetPINDir('D', 2, 0); // INT0 input
384 DID_vsetPINDir('D', 7, 1); // Temperature digit control
385 DID_vsetPINDir('A', 7, 1); // Temperature digit control
386 DID_vsetPINDir('D', ISD1828_PIN, 1); // ISD1828 OUTPUT
387
388 // Configure INT0 for falling edge
389 MCHCR |= (1 << ISCRI);
390 MCHCR &= ~(1 << ISCRO);
391 GICR |= (1 << INT0);
392
393 // Enable display
394 enable_display();
395 }
396
397 /* ===== MAIN FUNCTION ===== */
398
399 int main(void) {
400     // Initialize hardware
401     hardware_init();
402     timer0_init_ctc_mode();
403
404     // Load settings from EEPROM
405     hijri_day_offset_now = (int8_t)EEPROM_read(ADDR_HIJRI_OFFSET);
406     time_offset = (int8_t)EEPROM_read(ADDR_TIME_OFFSET);
407
408     // Clear any pending alarm flags
409     RTC_ClearAlarmFlag();
410
411     // Get initial time and date
412     RTC_GetTime(&hours, &minutes, &seconds);
413     RTC_GetDate(&dayofweek, &day, &month, &year);
414     last_displayed_day = day;
415
416     // Initialize prayer times and display
417     update_prayer_times_today();
418     update_display_array();
419     update_prayer_and_day_leds();
420
421     // Enable global interrupts
422     sei();
423
424     // Main loop
425     while (1) {
426         // Update every second
427         if (ms_counter >= 1000) {
428             ms_counter = 0;
429
430             // Get current time and date
431             RTC_GetTime(&hours, &minutes, &seconds);
432
433             update_display_array();
434             update_prayer_and_day_leds();
435
436             // Check for prayer times to trigger adhan
437             for (int i = 0; i < 5; i++) {
438                 if (hours == prayer_times[i].hour && minutes == prayer_times[i].minute) {
439                     if (current_adhan != i) {
440                         current_adhan = i;
441                         DID_write('0', ISD1828_PIN, 1);
442                         delay_ms(200); // Pulse to trigger ISD1828
443                         DID_write('0', ISD1828_PIN, 0);
444                     }
445                     break;
446                 }
447             }
448
449             // Reset adhan flag if time has passed
450             if (current_adhan != -1 &&
451                 (hours != prayer_times[current_adhan].hour ||
452                  minutes != prayer_times[current_adhan].minute)) {
453                 current_adhan = -1;
454             }
455
456             // Get current temperature
457             Temp = DS3231_GetTemperature();
458
459             // Rotate displayed prayer every 5 seconds
460             if (Five_sec_counter >= 5000) {
461                 Five_sec_counter = 0;
462                 current_prayer = (current_prayer + 1) % 5;
463                 update_display_array();
464                 update_prayer_and_day_leds();
465                 hijri_toggle ^= 1; // Toggle date display
466             }
467         }
468     }
469 }

```





Size 0.8" Cathode

7 Segment 0.8" Common Cathode

8106AS-1

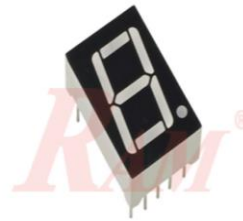
10.00 EGP

- 1 + Add to cart

♥ Add to wishlist

✎ Pick up from RAM Store
✎ Shipping: 2-3 Business Days

Internal Reference: 7SEG0.8C



f X i Size 1" Cathode

7 Segment 1.0" Common Cathode

KEM10106AH

18.00 EGP

- 1 + Add to cart

♥ Add to wishlist

✎ Pick up from RAM Store
✎ Shipping: 2-3 Business Days

Internal Reference: 7SEG1.0C



ATMEGA32AU -8-bit AVR

LE 260.00

ADD TO CART

Share Tweet Pin It



PIR Motion Sensor Module HC-SR501

PIR Motion Sensor Module

50.00 EGP

- 1 + Add to cart

♥ Add to wishlist

✎ Pick up from RAM Store
✎ Shipping: 2-3 Business Days

Internal Reference: KIT.PIR.MODULE



All Products / Kit Touch Red Switch

Kit Touch Red Switch

TTP223 Touch Switch Sensor Module Self-Locking/No-Locking Capacitive Touching Switch Button

15.00 EGP

- 1 + Add to cart

♥ Add to wishlist

✎ Pick up from RAM Store
✎ Shipping: 2-3 Business Days

Internal Reference: KIT.TOUCH.RED.1



All Products / LED 5mm Super Bright Red Color WR

LED 5mm Super Bright Red Color WR

0.50 EGP

- 1 + Add to cart

♥ Add to wishlist

✎ Pick up from RAM Store
✎ Shipping: 2-3 Business Days

Internal Reference: LED.WR



74595 (SN74HC595N)
Availability: In stock

Add to Wishlist Compare

8-bit Serial-to-Parallel Shift Register Tri-State with Output Latches (DIP-16)

11.00 EGP

1 Add to cart



Bluetooth Module HC-05 (6pin + Button)
Availability: In stock

Add to Wishlist Compare

Bluetooth Module HC-05 (6pin + Button)

210.00 EGP

1 Add to cart




Small Buzzer 5V
Availability: In stock

Add to Wishlist Compare

Small Buzzer 5V (TMB1240S)

7.00 EGP

1 Add to cart

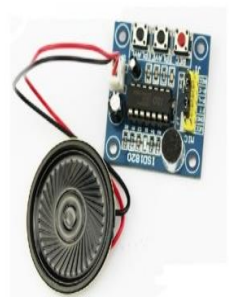


DS3231 High Precision Real Time Clock (RTC) with battery
LE 85.00

ADD TO CART

Share Tweet Pin it

Description **Reviews**



ISD1820 Voice Record Module
Availability: In stock

Add to Wishlist Compare

ISD1820 Voice Record Module

120.00 EGP

1 Add to cart

Total \approx 1000 EGP

Thank You