TECHNOLOGICAL UNIVERSITY OF THE PHILIPPINES

Ayala Boulevard, Ermita, Manila

CIT-ELECTRONICS DEPARTMENT

CPET11L-M – Microprocessor and Microcontroller Systems, Lab

1st Semester, SY 2-24-2025

Name: Joseph C. Arenas	**Instructor:** Prof. Tristan Mopas
Course & Section: BET-CPET- 3A	**Date Submitted:**

Activity 4

Topic 1: Beep Quarter Timer

Topic 2: LCD Timer with RTC

Topic 3: Time Alarm with LCD, TRC and Relay Module

I. OBJECTIVES

- To apply practical knowledge in using the Arduino Mega 2560
- To explain the functions and components of the related topics
- To implement the LCD output, timer circuit, RTC, and relay module components in a working circuit.
- To develop and enhance problem-solving skills related to the topics

II. EQUIPMENT AND MATERIALS

HARDWARE

- Arduino Uno R3 / Arduino Mega 2560
- Breadboard
- Jumper Wires
- Laptop
- 20x4 or 16x2 LCD

- Push button
- Buzzer
- 10KΩ Resistor
- RTC
- Relay Module
- Red and Green LEDs

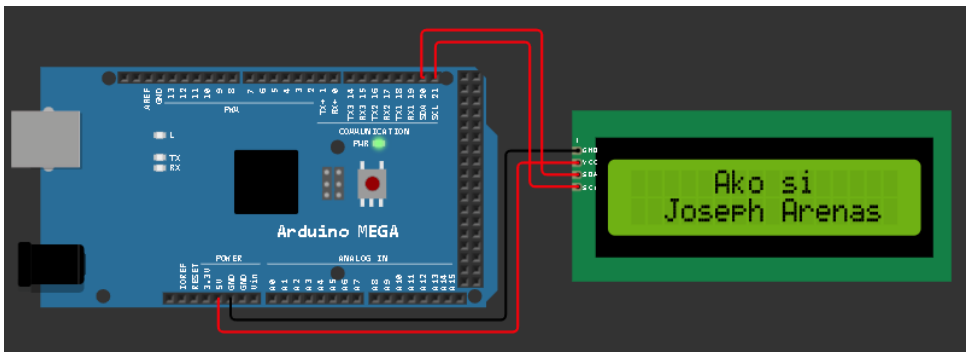
SOFTWARE

- Arduino IDE with libraries for specified components.
- MS Word
- Wokwi

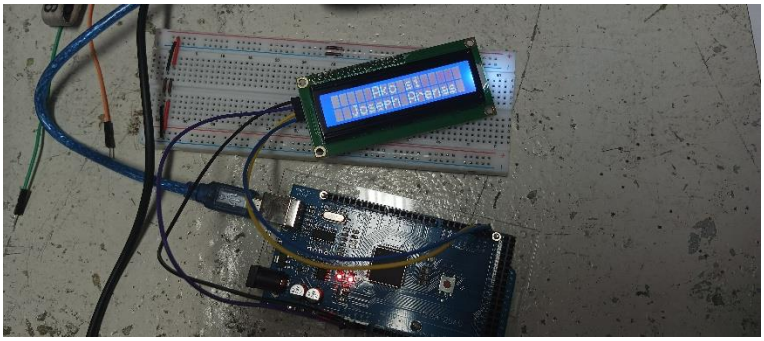
III. DIAGRAM

====== TOPIC 1: Beep Quarter Timer ======

A. Wokwi Simulation



B. Breadboard



C. Source Code

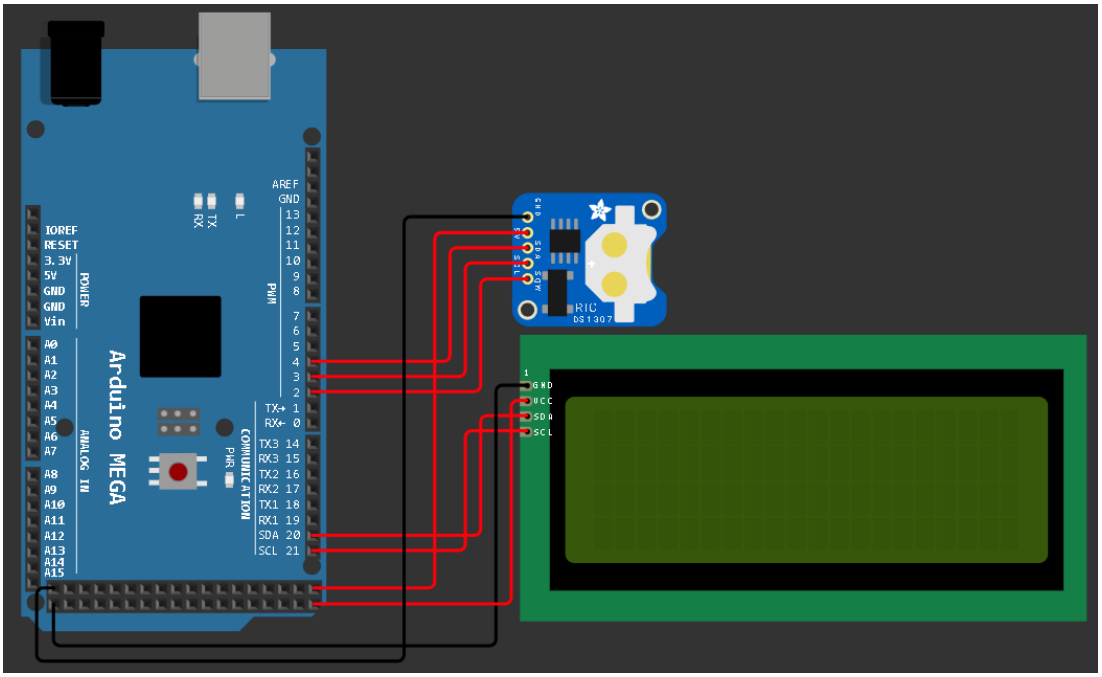
```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 20, 4);

int buzzer = 6;
int buttons[]= {2, 3, 4, 5};

int minutes = 12;
int seconds = 0;
boolean running = false;
unsigned long prevMillis = 0;

void setup() {
  lcd.init();
  lcd.backlight();
  pinMode(buzzer, OUTPUT);
  for(int i = 0 ; i < 4 ; i++){
    pinMode(buttons[i], OUTPUT);
  }
  displayTime();
}
void loop() {
  if (digitalRead(buttons[0]) == HIGH) {
    running = true;
    delay(200);
  }
```

```
if (digitalRead(buttons[1]) == HIGH) {
    running = false;
    minutes = 12;
    seconds = 0;
    digitalWrite(buzzer, LOW);
    displayTime();
    delay(200);
  }
  if (digitalRead(buttons[2]) == HIGH &&
!running) {
    if (minutes > 0 || seconds >= 10) {
      if (seconds >= 10) {
        seconds -= 10;
      } else if (minutes > 0) {
        minutes--;
        seconds = 50;
      }
    }
    displayTime();
    delay(200);
  }
  if (digitalRead(buttons[3]) == HIGH &&
!running) {
    if (seconds <= 49) {
      seconds += 10;
    } else {
      if (minutes < 99) {
        minutes++;
        seconds = 0;
      }
    }
```

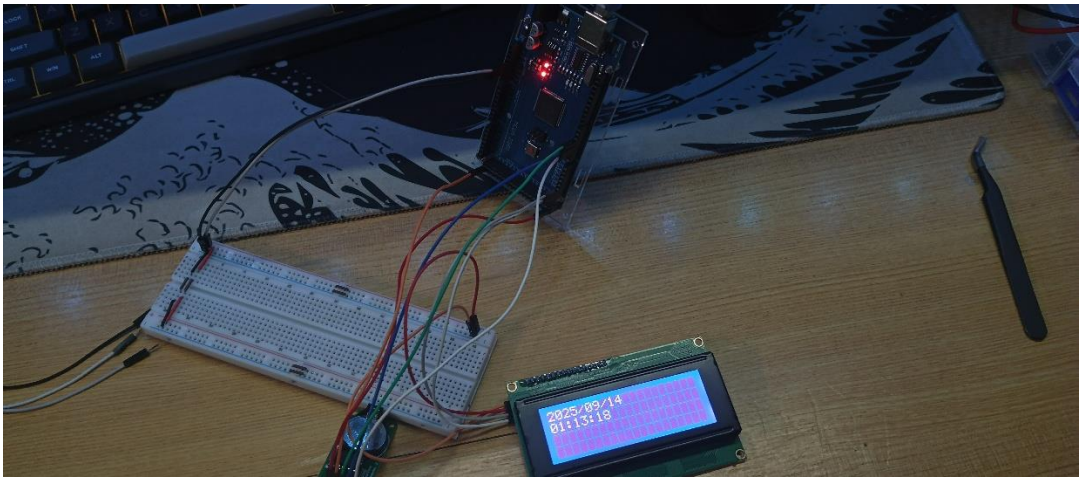
```
displayTime();
    delay(200);
  }

  if (running) {
    unsigned long currentMillis = millis();
    if (currentMillis - prevMillis >= 1000) {
      prevMillis = currentMillis;
      if (seconds == 0) {
        if (minutes == 0) {
          running = false;
          digitalWrite(buzzer, HIGH);
        } else {
          minutes--;
          seconds = 59;
        }
      } else {
        seconds--;
      }
      displayTime();
    }
  }
}
void displayTime() {
  lcd.clear();
  lcd.setCursor(4, 1);
  if (minutes < 10) lcd.print("0");
  lcd.print(minutes);
  lcd.print(":");
  if (seconds < 10) lcd.print("0");
  lcd.print(seconds);
}
```

===== Topic 2: LCD Timer with RTC=====

A. Wokwi Simulation



B. Breadboard

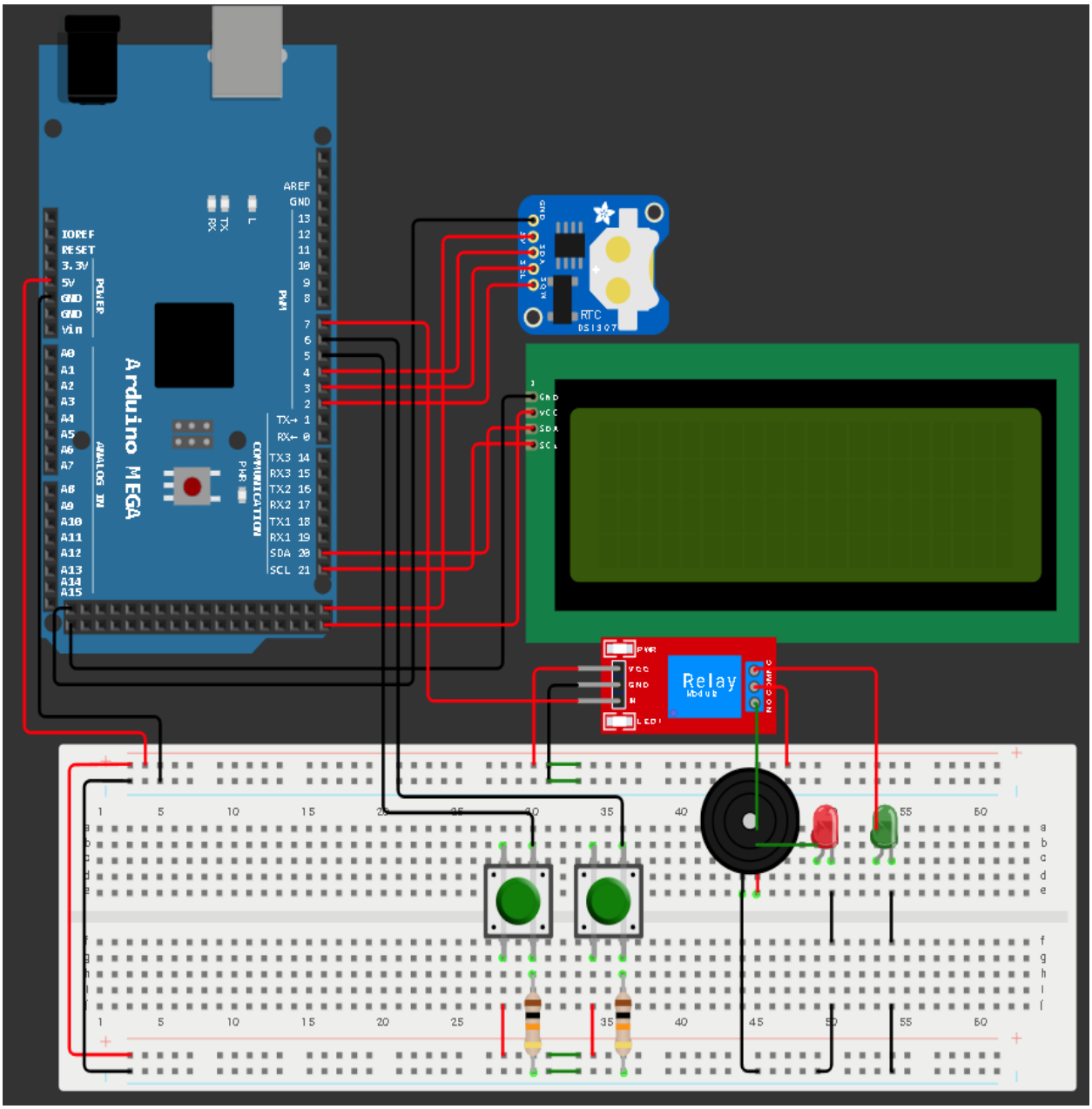


C. Source Code

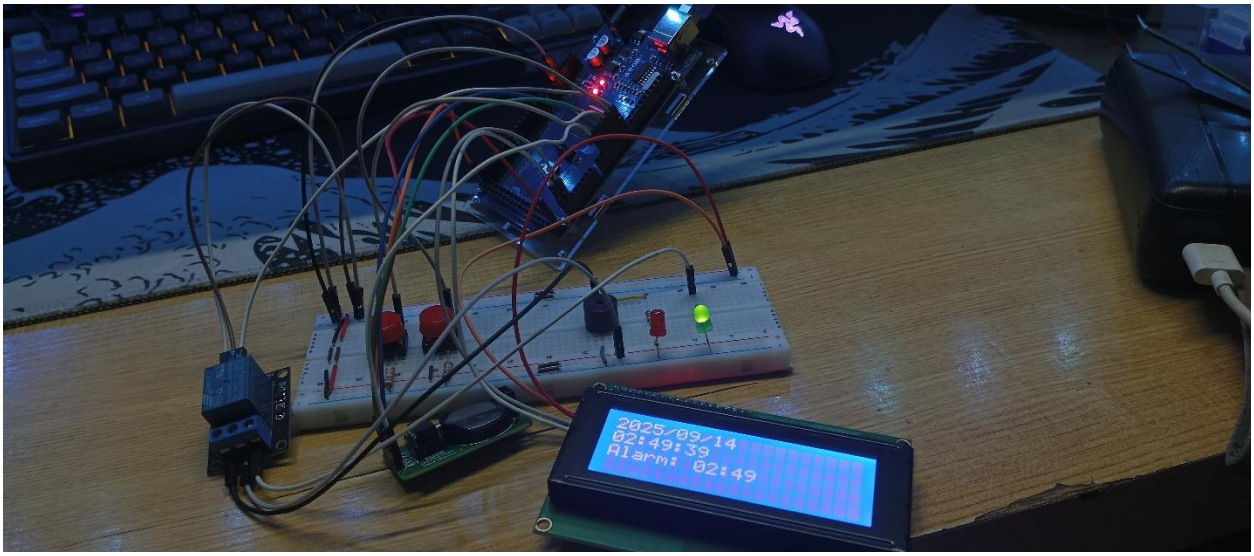
```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <DS1302.h>

LiquidCrystal_I2C lcd(0x27, 20, 4);
DS1302 rtc(2, 3, 4);

void setup() {
  lcd.init();
  lcd.backlight();

  rtc.writeProtect(false);
  rtc.halt(false);

  Time t(2025, 9, 14, 1, 00, 0,
Time::kSunday);
  rtc.time(t);
}

void loop() {
  Time t = rtc.time();
  lcd.clear();

  lcd.setCursor(0, 0);
  lcd.print(t.yr);
  lcd.print("/");
  if (t.mon < 10) lcd.print("0");
  lcd.print(t.mon);
  lcd.print("/");
  if (t.date < 10) lcd.print("0");
  lcd.print(t.date);
  lcd.setCursor(0, 1);
  if (t.hr < 10) lcd.print("0");
  lcd.print(t.hr);
  lcd.print(":");
  if (t.min < 10) lcd.print("0");
  lcd.print(t.min);
  lcd.print(":");
  if (t.sec < 10) lcd.print("0");
  lcd.print(t.sec);

  delay(1000);
}
```

=== Topic 3: Time Alarm with LCD, TRC and Relay Module ===

A. Wokwi Simulation



B. Breadboard



C. Source Code

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <DS1302.h>

LiquidCrystal_I2C lcd(0x27, 20, 4);
DS1302 rtc(2, 3, 4);

const int btnHour = 5;
const int btnMin  = 6;
const int relayPin = 7;
const int ledPin   = 8;

int alarmHour = 0;
int alarmMin = 0;
bool alarmTriggered = false;
unsigned long alarmStart = 0;
```

```
void setup() {
  lcd.init();
  lcd.backlight();
  rtc.writeProtect(false);
  rtc.halt(false);

  Time t(2025, 9, 14, 2, 46, 0,
Time::kSunday);
  rtc.time(t);

  pinMode(btnHour, INPUT);
  pinMode(btnMin, INPUT);
  pinMode(relayPin, OUTPUT);
  pinMode(ledPin, OUTPUT);

  lcd.setCursor(0, 0);
  lcd.print("Alarm Clock Ready");
  delay(2000);
}
```

```
void loop() {
  Time t = rtc.time();
  if (digitalRead(btnHour) == HIGH) {
    alarmHour = (alarmHour + 1) % 24;
    delay(200);
  }

  if (digitalRead(btnMin) == HIGH) {
    alarmMin = (alarmMin + 1) % 60;
    delay(200);
  }

  lcd.clear();

  lcd.setCursor(0, 0);
  lcd.print(t.yr);
  lcd.print("/");
  if (t.mon < 10) lcd.print("0");
  lcd.print(t.mon);
  lcd.print("/");
  if (t.date < 10) lcd.print("0");
  lcd.print(t.date);

  lcd.setCursor(0, 1);
  if (t.hr < 10) lcd.print("0");
  lcd.print(t.hr);
  lcd.print(":");
  if (t.min < 10) lcd.print("0");
  lcd.print(t.min);
  lcd.print(":");
  if (t.sec < 10) lcd.print("0");
  lcd.print(t.sec);
```

```
  lcd.setCursor(0, 2);
  lcd.print("Alarm: ");
  if (alarmHour < 10) lcd.print("0");
  lcd.print(alarmHour);
  lcd.print(":");
  if (alarmMin < 10) lcd.print("0");
  lcd.print(alarmMin);

  if (!alarmTriggered && t.hr == alarmHour
&& t.min == alarmMin && t.sec == 0) {
    alarmTriggered = true;
    alarmStart = millis();
    digitalWrite(relayPin, HIGH);
  }

  if (alarmTriggered && (millis() - alarmStart
>= 5000)) {
    digitalWrite(relayPin, LOW);
    digitalWrite(ledPin, HIGH);
  }

  delay(200);
}
```

D. PROCEDURE

A. Preparation

- Gather the required components as stated in Chapter 4.
- Prepare the Arduino IDE on your preferred device.
- Review the problem for each topic and formulate or search for a circuit diagram given the problem of each topic.
- Simulate your circuit diagram to a circuit simulator such as Wokwi.

B. Actual

- For topic 1, connect the 5v VCC and GND pin of the microcontroller to the "+" and "-" of the breadboard. Connect the VCC and GND of the LCD to VCC and ground. Connect the SDA and SCL of the LCD to the SDA and SCL pins of the microcontroller. Attach the 4 push buttons to the breadboard. Connect buttons 1-4 to digital pins 2-5 respectively then connect them all to a respective 10KΩ resistor to ground. Connect their respective adajacent pins to VCC. Attach the buzzer to the breadboard and connects its anode to digital pin 6. Connect its cathode to ground. Connect the USBVCC to the Arduino Uno R3 / Arduino Mega 2560 then verify and upload the code.
- For topic 2, connect the 5v VCC and GND pin of the microcontroller to the "+" and "-" of the breadboard. Connect the VCC and GND of the LCD to VCC and ground. Connect the SDA and SCL of the LCD to the SDA and SCL pins of the microcontroller. Connect the reset pin of the RTC to digital pin 2, its data pin to digital pin 3 and its clock pin to digital pin 4. Connect the VCC and GND pins of the RTC to 5v and ground. Connect the USBVCC to the Arduino Uno R3 / Arduino Mega 2560 then verify and upload the code.
- For topic 3, connect the 5v VCC and GND pin of the microcontroller to the "+" and "-" of the breadboard. Connect the VCC and GND of the LCD to VCC and ground. Connect the SDA and SCL of the LCD to the SDA and SCL pins of the microcontroller. Connect the reset pin of the RTC to digital pin 2, its data pin to digital pin 3 and its clock pin to digital pin 4. Connect the VCC and GND pins of the RTC to 5v and ground. Connect the VCC and GND pins of the relay module

to 5v and ground. Connect its data pin to digital pin 7. Connect the common pin of the relay module to VCC. Connect the NO pin of the relay module to the respective anodes of the red LED and buzzer with their respective cathodes leading to ground. Attach the green LED on the breadboard and connect its anode to the NC of the relay module. Connect the cathode of the green LED to ground. Attach the two push buttons to the breadboard and connect its respective pins to digital pins 5 and 6. Afterwards, direct its respective pin to ground using a 10KΩ resistor. Connect the adjacent pin of the push buttons to VCC. Connect the USBVCC to the Arduino Uno R3 / Arduino Mega 2560 then verify and upload the code.

C. Checking

- For topic 1, ensure that the LCD displays the time in proper time format. Ensure that button 1 starts the time countdown, button 2 resets the timer back to 12:00, button 3 decrements timer alarm, and button 4 increments timer alarm. Make sure that when the timer reaches zero, the buzzer goes HIGH.
- For topic 2, ensure that the LCD displays date and time and when the USBVCC is disconnected and an external power supply is utilized for the microcontroller, the time continues where it left of.
- For topic 3, ensure that the LCD displays the time and the alarm time. Make sure that button 1 increments the hour and button increments the minutes in setting up the alarm through the LCD. Ensure that when the alarm time is reached, the buzzer and red LED goes HIGH and after 5 seconds, the relay switches to NC and places the green LED to HIGH.
- Adjust values or fix errors if the output is not working as expected.

D. Uploading

- Upload the final and corrected code to the Arduino Uno R3/Arduino Mega 2560.
- Ensure that all three topics work properly.

E. CONCLUSION

The experiments demonstrated practical applications of Arduino with LCD, RTC, relay modules, and push buttons, highlighting their integration in timer and alarm systems. Each activity strengthened skills in circuit building, coding, and troubleshooting to achieve functional designs. Overall, the work enhanced technical competence and problem-solving abilities in microcontroller-based projects.