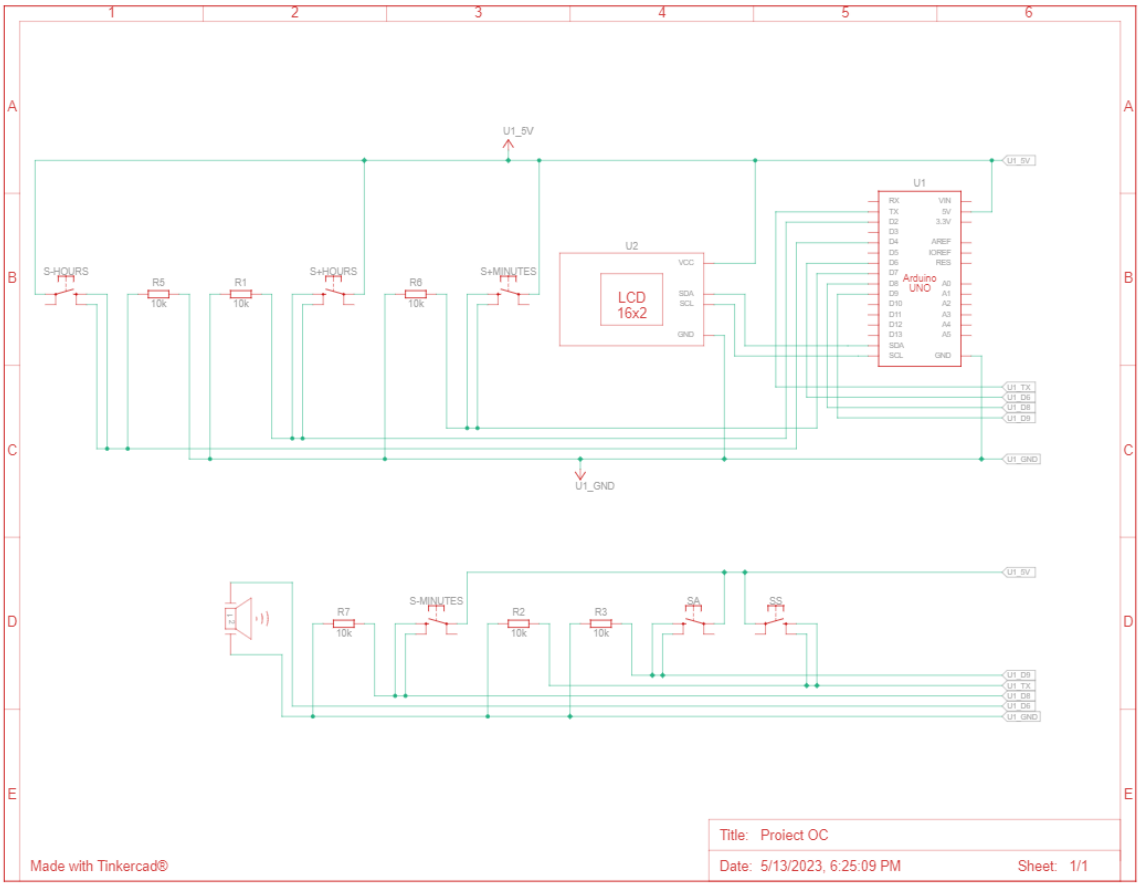
Ceas cu alarmă – Documentație

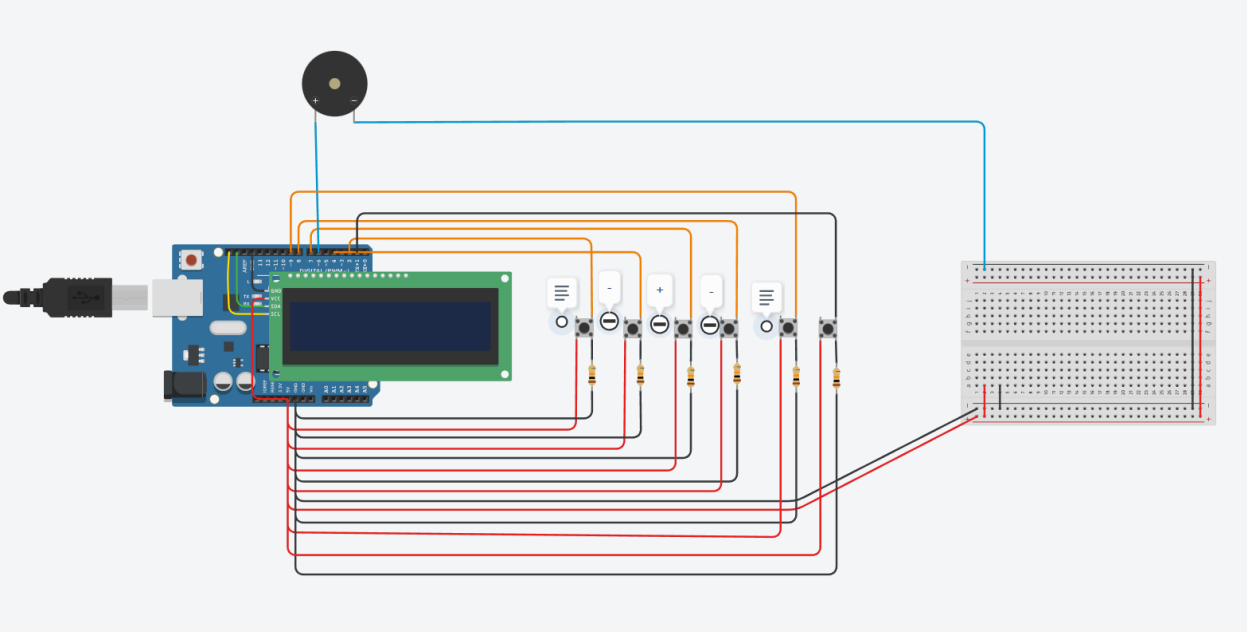
Link Tinkercad: https://www.tinkercad.com/things/iJhGjloYO4z-daring-amberis/editel?tenant=circuits

Components used:

Name	Pin	Info	Conected to the Arduino Pins	Tests
Arduino				
Breadboard			GND, 5V	
Buton	1,2,4,7,8,9	6 buttons, aiming to modify certain fields of a watch, to adjust the menu, to save or delete alarms.	GND, 5V	Written code + manual modification
LCD 16 x 2 (I2C)	GND,VCC,SCL,SDA		GND,VCC,SCL,SDA	
Piezo	+, -	Alarm that will ring when the current time coincides with the saved alarm.	6, 5V, GND	By hand, introducing a specific condition, and if it proved to be true, the alarm would start ringing

Hardware architecture





Functions used:

a) Predefined:

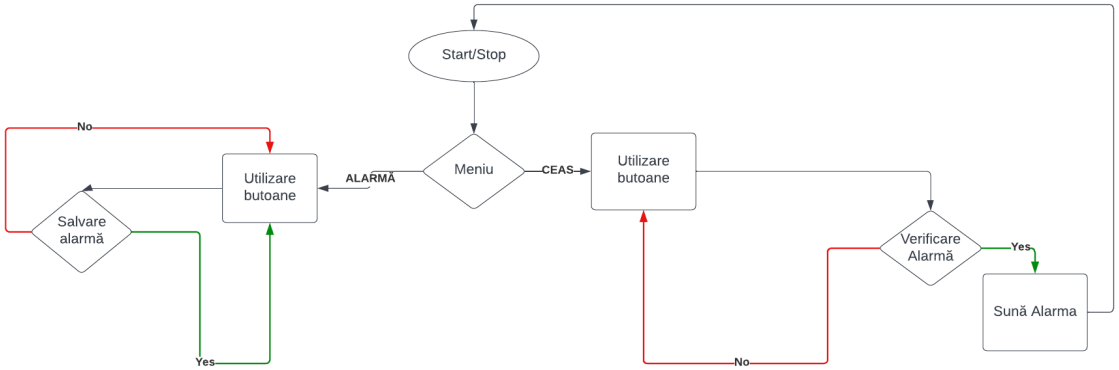
void setup(): initializes input and output pins as well as the LCD screen.
void loop(): constantly repeats the instructions inside it.
lcd.begin(): function to initialize the LCD screen.
lcd.setCursor(): writes at a specific desired position on the LCD screen.
lcd.print(): displays data on an LCD screen.
Serial.begin(): initializes serial communication between the Arduino board and another component or computer.
digitalRead(COMPONENT_PIN): reads the state of a digital PIN.
delay(value): pauses the program for a given number of milliseconds.
lcd_1.noBlink(): Disables the blinking effect.

b) Implemented by us:

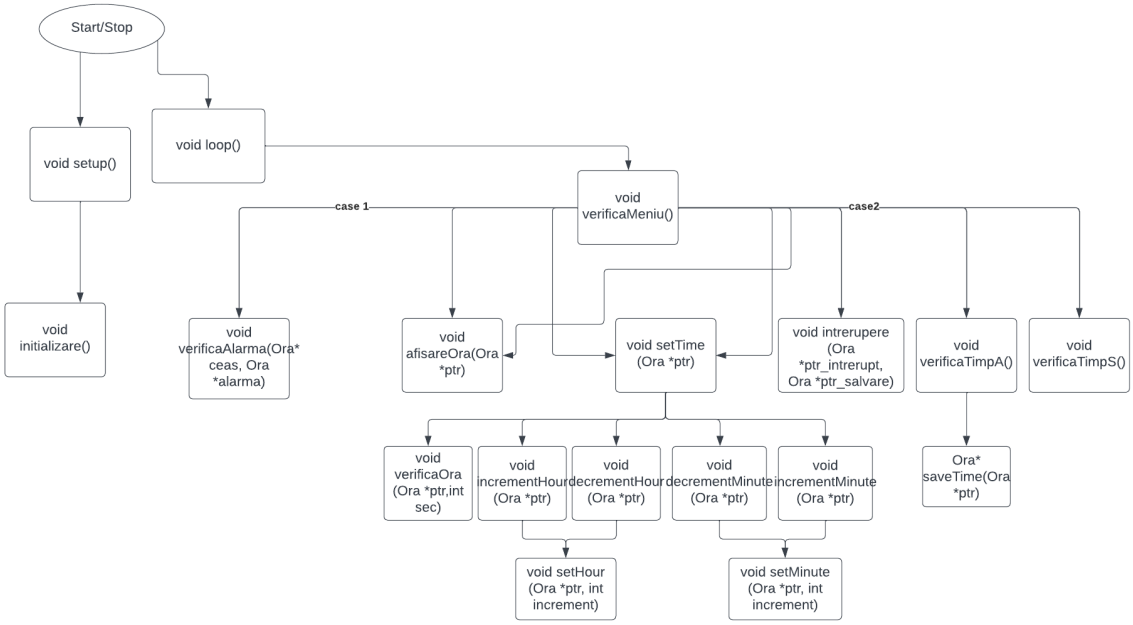
void initializare (Ora* ptr): initializes the fields of the structure with 0.
void afisareOra (Ora *ptr): displays the time.
void setTime(Ora *ptr): uses buttons to modify the time.
void verificaOra (Ora *ptr, int sec): function to check the accuracy of the time.
void setHour(Ora *ptr, int increment): modifies the hour.
void setMinute(Ora *ptr, int increment): modifies the minute.
void incrementHour(Ora *ptr): function for increasing the hour + resetting the seconds.
void decrementHour(Ora *ptr): function for decreasing the hour + resetting the seconds.
void incrementMinute(Ora *ptr): function for increasing the minutes + resetting the seconds.
void decrementMinute(Ora *ptr): function for decreasing the minutes + resetting the seconds.
void verificaMeniu (): sets the Clock/Alarm menu.
void verificaTimpA (): checks if button A has been pressed for more than 3 seconds.
void verificaTimpS (): checks how long button S has been pressed and acts accordingly to the press time
void intrerupere (Ora *ptr_interrupted, Ora *ptr_save): saves the current time to be used after switching from one menu to another.
void saveTime(Ora *ptr): saves the alarm.
void verificaAlarma (Ora* clock, Ora *alarm): checks if the current time is the same as the alarm.

Libraries used:

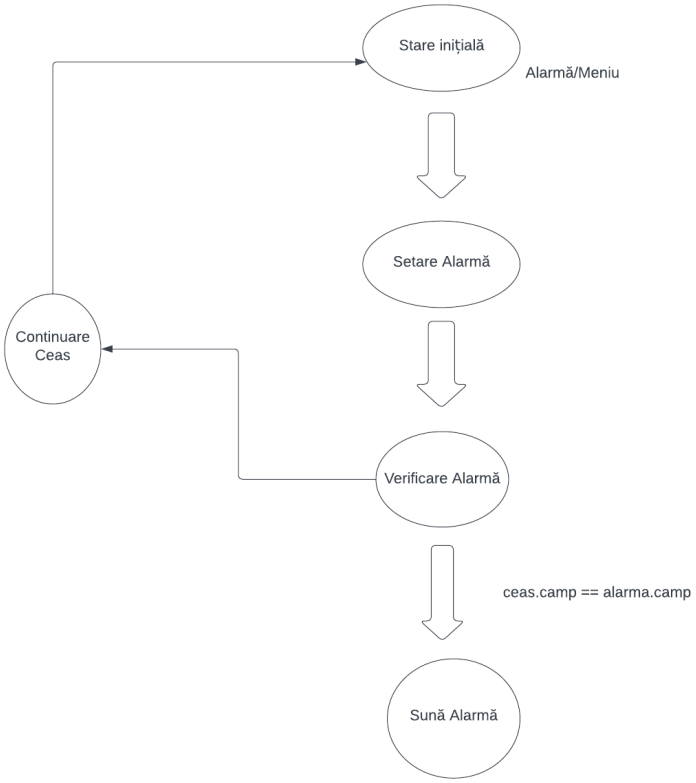
#include <LiquidCrystal.h>: library used to easily manipulate an LCD screenSchema logică



Function call schema



State diagram



Code
Initializing button variables, menu

```
//variabile
const int buttonPin1 = 2;  // the number of the pushbutton pin
int buttonState1 = 0;

const int buttonPin2 = 4;  // the number of the pushbutton pin
int buttonState2 = 0;

const int buttonPin3 = 7;  // the number of the pushbutton pin
int buttonState3 = 0;

const int buttonPin4 = 8;  // the number of the pushbutton pin
int buttonState4 = 0;

const int buttonPin5 = 9; // Pinul la care este conectat butonul
bool buttonPressed = false; // Indicator pentru a verifica dacă b
int buttonState5 = 0;

int currentState = 0;
int previousState = 0; // Variabila pentru starea anterioara a bu
unsigned long buttonStartTime = 0; // Timpul la care butonul a fo
unsigned long buttonPressDuration = 0; // Durata de apăsare a but

const int buttonPin6 = 1;  // the number of the pushbutton pin
int buttonState6 = 0;
bool buttonPressedS = false;
unsigned long buttonStartTimeS = 0;
unsigned long buttonPressDurationS = 0;

bool alarmaSetata = false;

int menu = 1; // Variabilă pentru meniu
```

Starting the program loop, checking the menu, setting the time according to the menu

```
void loop() {
   verifiicaMeniu();
}
```

```
void verificaMeniu()
{

  buttonState5 = digitalRead(buttonPin5);
  ceas = intrerupt;
  if (buttonState5 == HIGH && previousState == LOW)
  {
    delay(50);
    currentState = 1 - currentState;
    lcd_1.clear();
    if (currentState == 1)
    {
        menu = 2; // Daca s-a apasat o data, schimba la meniul "
        lcd_1.clear();
    }
    else
    {
        menu = 1; // Daca s-a apasat a doua oara, schimba la men
        lcd_1.clear();
    }
  }

 previousState = buttonState5; // Salveaza starea curenta a buto

  switch (menu)
  {
    case 1:
            lcd_1.setCursor(0, 0);
            lcd_1.print("Ceasul-");
            afisareOra(ceas);
            ceas->secunda++;
            if(alarmaSetata == true)
                verificaAlarma(intrerupt, alarma);
            setTime(ceas);
            delay(50);
    break;

    case 2:
            intrerupere(ceas,intrerupt);
            lcd_1.clear();
```

Checking the alarm save status.

```
void verificaTimpA()
{
  int buttonState = digitalRead(buttonPin5);
  if (buttonState == HIGH && !buttonPressed) {
    buttonPressed = true;
    buttonStartTime = millis();
  }


  delay(500);
  // Verifică dacă butonul a fost eliberat
  if (buttonState == LOW && buttonPressed) {
    buttonPressed = false;
    buttonPressDuration = millis() - buttonStartTime;

    // Verifică dacă durata de apăsare este de 3 secunde (3000 ms
    if (buttonPressDuration > 3000) {
        alarma = saveTime(alarma);
        menu = 1;
        lcd_1.clear();
        lcd_1.print("setat!");
    }
  }
  buttonStartTime = 0;
}
```

Checking the stop/deletion/snooze of the alarm.

```
void verificaTimpS()
{
    int buttonStateS = digitalRead(buttonPin6);
    if (buttonStateS == HIGH && !buttonPressedS) {
        buttonPressedS = true;
        buttonStartTimeS = millis();
  }

  // Verifică dacă butonul a fost eliberat
  if (buttonStateS == LOW && buttonPressedS) {
    buttonPressedS = false;
    buttonPressDurationS = millis() - buttonStartTimeS;

   if (buttonPressDurationS == 2000) {
        noTone(AlarmSND); // Oprire sunet alarma
    }

    if (buttonPressDurationS < 2000) {
        alarma->secunda += 10;
    }

    if (buttonPressDurationS > 3000 && menu == 2) {
        initializare(alarma);
        alarmaSetata = false;
    }
  }
}
```

Checking the alarm. void verificaAlarma(Ora* ceas, Ora *alarma)

```
{
 if (ceas->ora == alarma->ora && ceas->minut == alarma->minut && ceas->secunda == alarma->secunda)
  {
    tone(AlarmSND, 1000, 1000);
  }
}
```

Concluzii. Dificultăți

We consider the project complexity to be medium due to specific functions of programming with Arduino platform such as digitalRead(COMPONENT_PIN), as well as the fact that push buttons have a certain delay which complicates understanding the project's operation. Additionally, functions like checkTimeS() and checkTimeA() posed higher difficulty considering the use of the implemented millis() function, and the loop which complicates the optimal functioning of the functions.