Title : Pomodoro timer

Team name: being Productive!

Problem Statement:

Problem Overview:

In today's fast-paced world, individuals often struggle to manage their time effectively and maintain productivity during work or study sessions. Distractions, procrastination, and a lack of structured time management can hinder performance and overall well-being. To address this challenge, we aim to develop an efficient Pomodoro Timer that combines digital logic and design to help users implement the Pomodoro Technique—a time management method that emphasizes work and break intervals.

Introduction:

To address this issue, there is a pressing need for an efficient time management solution that aligns with the brain's natural rhythms and enhances productivity. A Pomodoro countdown timer aims to provide a structured approach to work by breaking tasks into manageable intervals, typically 25 minutes of focused work followed by a short 5-minute break.

This innovative timer, utilizing the power of Arduino Nano, IC 4026, and a seven-segment display, leverages the concept of the Pomodoro Technique. It divides work into manageable intervals, typically 25 minutes of focused work (the 'work session') followed by a brief 5-minute break (the 'break session'), aligning with the brain's optimal attention span.

What sets this Pomodoro timer apart is its user-friendly interface, empowered by three buttons. These buttons allow seamless navigation between work and break sessions, enabling users to shift effortlessly between these phases with a simple press. Additionally, a stop button is incorporated, providing the convenience of resetting the timer to zero at any point.

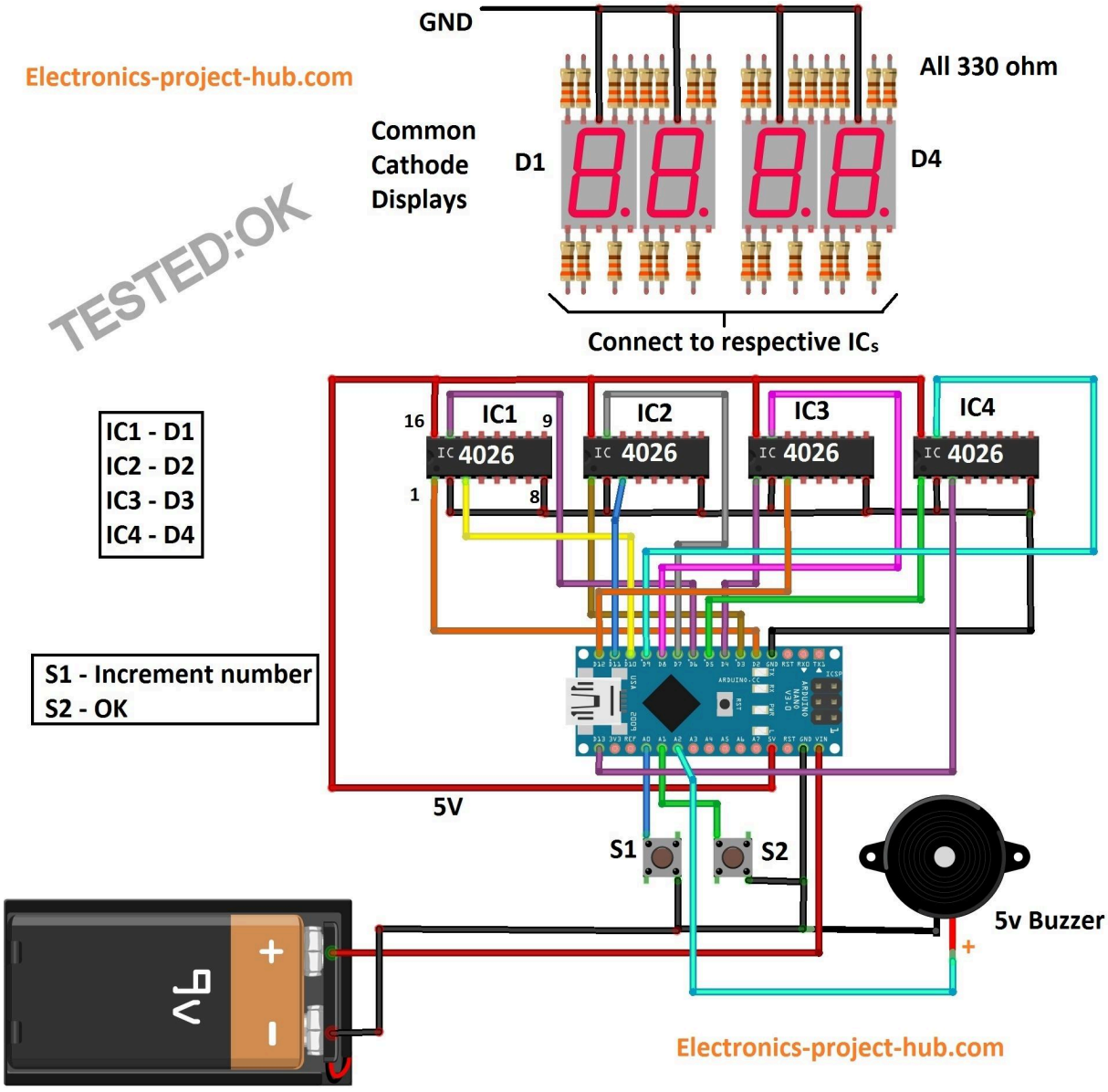
Project Objectives:

Digital Timer: Create a digital timer that facilitates the implementation of the Pomodoro Technique.
User-Friendly Interface: Develop a user-friendly interface with a seven-segment display to visualize the countdown of work and break intervals.
Timer Control: Enable users to shift between work and break session, and then reset the timer, ensuring flexibility in managing their tasks.
Audio Alerts (Optional): Implement audio alerts, such as buzzer notifications, to signal the end of work intervals and breaks.

The initial design criteria:

- **To hv a four digit output using 7-segment display**
- **To have a circuit count from 25:00 to 00:00 when work session button is clicked, and a buzzer alarm at the end of work session**
- **To have a circuit count from 05:00 to 00:00 when break session button is clicked and a buzzer alarm at the end of break session**
- **Stop or reset button that resets the display to 00:00**
- **Arduino nano microcontroller that is programmed with code for Pomodoro Technique**

BLOCK DIAGRAM OR ARCHITECTURE:



HDL Code:

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

use IEEE.STD_LOGIC_ARITH.ALL;

use IEEE.STD_LOGIC_UNSIGNED.ALL;


entity Pomodoro_Timer_Circuit_Buzzer is

    Port ( clk : in STD_LOGIC;

           reset : in STD_LOGIC;

           work_button : in STD_LOGIC;

           break_button : in STD_LOGIC;

           buzzer : out STD_LOGIC;

           seven_seg_out : out STD_LOGIC_VECTOR(6 downto 0));

end Pomodoro_Timer_Circuit_Buzzer;


architecture Behavioral of Pomodoro_Timer_Circuit_Buzzer is

    signal counter : integer range 0 to 25000000 := 25000000; -- 25-minute work session (assuming 1 Hz clock)

    signal break_counter : integer range 0 to 5000000 := 5000000; -- 5-minute break session (assuming 1 Hz clock)
```

```
    signal segment_data : STD_LOGIC_VECTOR(6 downto 0) := "0000000"; -- Initialize to zero
    signal display_enable : STD_LOGIC := '1'; -- Enable signal for the display (active low)
    signal buzzer_sound : STD_LOGIC := '0'; -- Signal to control the buzzer

    -- Counter for IC 4026
    signal ic4026_counter : integer range 0 to 9 := 0; -- Counter for IC 4026
    signal ic4026_count_enable : STD_LOGIC := '0'; -- Count enable signal for IC 4026 (active high)

begin
    process(clk, reset)
    begin
        if reset = '1' then
            counter <= 25000000; -- Reset to 25-minute work session duration
            break_counter <= 5000000; -- Reset to 5-minute break session duration
            segment_data <= "0000000"; -- Reset seven-segment display data
            display_enable <= '1'; -- Enable the display
            buzzer_sound <= '0'; -- Deactivate the buzzer
        elsif rising_edge(clk) then
            -- Decrement counter if it's not zero
            if counter > 0 then
                counter <= counter - 1;
```

```
end if;

-- Update the seven-segment display data based on the counter value
case counter is
    when 25000000 =>
        segment_data <= "0011111"; -- Display '2' on the seven-segment display
    when 0 =>
        segment_data <= "0111111"; -- Display '0' on the seven-segment display
        display_enable <= '0'; -- Disable the display at the end of the session
        buzzer_sound <= '1'; -- Activate the buzzer
    when others =>
        segment_data <= "0110000"; -- Display '5' on the seven-segment display
end case;

-- Check for button presses to switch between work and break sessions
if work_button = '1' then
    counter <= 25000000; -- Start a new 25-minute work session
    display_enable <= '1'; -- Enable the display
elsif break_button = '1' then
    break_counter <= 5000000; -- Start a new 5-minute break session
    display_enable <= '1'; -- Enable the display
```

```
        end if;

        -- Decrement break counter if it's not zero
        if break_counter > 0 then
            break_counter <= break_counter - 1;
        end if;

        -- Update the IC 4026 counter based on the break counter value
        if break_counter = 0 then
            ic4026_counter <= 5; -- Display '5' on the IC 4026 (break session)
            ic4026_count_enable <= '1'; -- Enable counting on IC 4026
        else
            ic4026_counter <= 0; -- Reset IC 4026 counter
            ic4026_count_enable <= '0'; -- Disable counting on IC 4026
        end if;
    end if;
end process;

-- Connect the seven-segment display output to the output port
seven_seg_out <= segment_data when display_enable = '1' else "1111111";
```

```
    -- Control the buzzer output
    buzzer <= buzzer_sound;

    -- Simulate IC 4026 behavior
    process(clk)
    begin
        if rising_edge(clk) then
            if ic4026_count_enable = '1' then
                -- Count up on IC 4026
                if ic4026_counter < 9 then
                    ic4026_counter <= ic4026_counter + 1;
                else
                    ic4026_counter <= 0; -- Reset the IC 4026 counter after reaching 9
                end if;
            end if;
        end if;
    end process;
end Behavioral;
```

Implementation details:

Initialization:

- Initialize the pins for the four IC 4026s, the seven-segment displays, and the buttons as inputs or outputs in the `setup()` function.

Timer Logic:

- Use a variable to store the current state of the timer (e.g., `enum { WORK, BREAK, STOP } timerState;`).
- Implement logic to handle the countdown for the work session and break session based on button presses:
    - When the "work" button is pressed, start a 25-minute countdown.
    - When the "break" button is pressed, start a 5-minute countdown.
    - Use the `millis()` function to keep track of time and update the displays accordingly.

Countdown Logic:

Work Session Countdown (25 minutes):

- Initialization: Upon pressing the "work" button, set a variable (`workTimer`) to 25 minutes (25 * 60 * 1000 milliseconds).
- Countdown: Use the `millis()` function to track time and decrement `workTimer` accordingly.
- Display Update: Continuously update the seven-segment display with the remaining time.

- Buzzer Activation: When `workTimer` reaches zero, activate the buzzer to signal the end of the work session.

Break Session Countdown (5 minutes):

- Initialization: Upon pressing the "break" button, set a variable (`breakTimer`) to 5 minutes (5 * 60 * 1000 milliseconds).
- Countdown: Use the `millis()` function to track time and decrement `breakTimer` accordingly.
- Display Update: Continuously update the seven-segment display with the remaining time.
- Buzzer Activation: When `breakTimer` reaches zero, activate the buzzer to signal the end of the break session.

Seven-Segment Display Control:

- Use the IC 4026 to control the seven-segment displays. The 4026 is a 7-segment display driver and can be used to count and display numbers.
- Connect the IC 4026 to the corresponding seven-segment display and control its count based on the time remaining in the countdown.

Buzzer Control:

- Activate the buzzer when the countdown for the work or break session ends. For example, when the countdown reaches zero, trigger the buzzer to signal the end of the session.

Button Inputs:

- Use the `digitalRead()` function to detect button presses.
- Assign functionalities to the buttons: starting the work session, starting the break session, and resetting the timer.

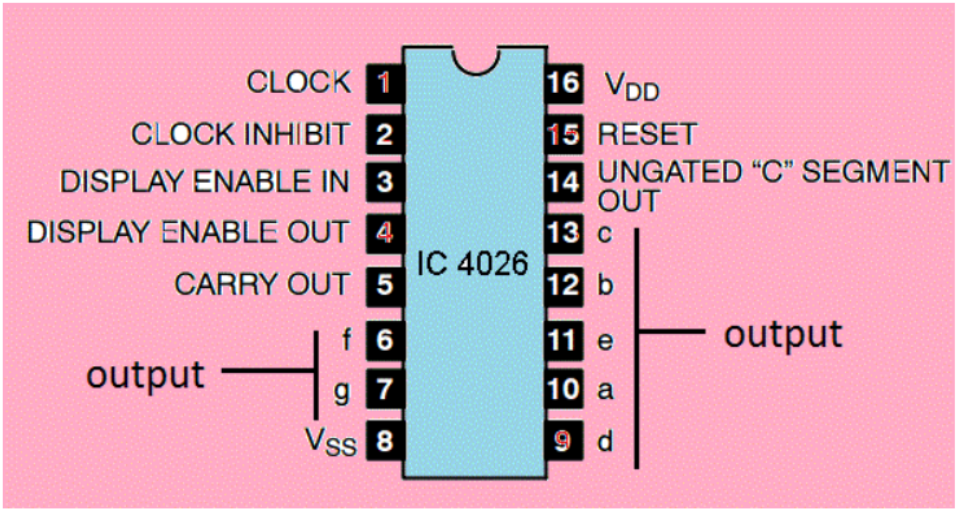
Reset Functionality:

- Implement functionality for the "stop" button to reset the timer back to zero and stop any ongoing countdown.

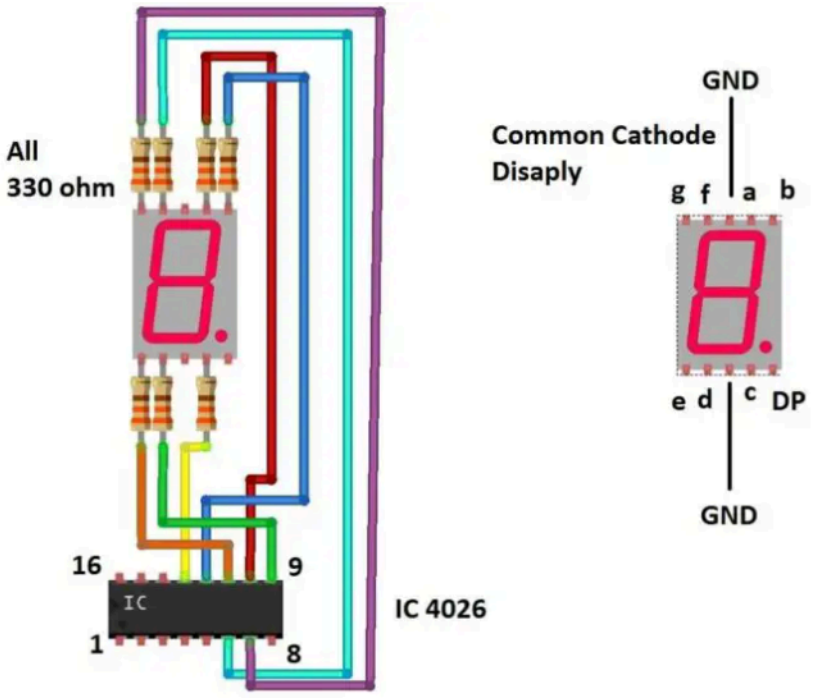
Components Required:

- Microcontroller (Arduino Nano)
- Seven-segment display - 4 NOS
- Buttons (Start, break, Reset/stop)
- Buzzer -5V
- Resistors (for current limiting)
- Breadboard or PCB for circuit assembly
- Jumper wires
- Power source (battery or power adapter) - 9V BATTERY
- **IC 4026 - 4 nos**

PIN DIAGRAM OF IC 4026:



CIRCUIT DIAGRAM:



IC 4026 to Common Cathode Display Connection

Code Explanation:

Code used in the project includes functions and logic for

- Starting - work session(25 minutes),break button - to start the break session(5 minutes) and resetting the timer
- Displaying the countdown on the seven-segment display
- Handling user input from buttons
- Implementing the pomodoro technique

Circuit Connections:

Power Supply:

- Connect the positive terminal (+) of the 9V battery to the VIN pin of the Arduino Nano.
- Connect the negative terminal (-) of the battery to the GND pin of the Arduino Nano.

IC 4026 and Seven-Segment Display Connections:

- Connect the common pin (GND) of each seven-segment display to GND on the breadboard.
- Connect VCC and GND of the 4026 IC to the +5V and GND of the battery, respectively.

Push Button Connections:

- work button connected to analog pin 0 of the Arduino Nano.
- break button connected to analog pin 1 of the Arduino Nano.
- reset/stop button connected to digital pin 3 of the Arduino Nano.

Buzzer Connection (optional):

- Connect the positive leg of the buzzer to a analog pin 2 on the Arduino Nano.
- Connect the negative leg of the buzzer to GND on the breadboard.

Pin connection Details:

```
const int clk_ML = 2;

const int clk_MR = 3;

const int clk_SL = 4;

const int clk_SR = 5;

const int rst_ML = 6;
```

```
const int rst_MR = 7;

const int rst_SL = 8;

const int rst_SR = 9;

const int DE_ML = 10;

const int DE_MR = 11;

const int DE_SL = 12;

const int DE_SR = 13;

const int work = A0;

const int break_button = A1;

const int buzzer = A2;

const int stop = A3;
```

Arduino Nano Code:

Use the code provided in the previous explanation for the Pomodoro timer with the 4026 IC, 7-segment displays, and push buttons. Ensure the connections correspond to the pins specified in the code.

Once the circuit is wired according to these instructions and the code is uploaded to the Arduino Nano, the Pomodoro timer should be ready to use with the 9V battery as its power source. Adjust the pin numbers in the code if necessary, based on the connections made in your setup.

Remember to check the connections and polarity while connecting the 9V battery and components to avoid any damage to the Arduino or components.

Result:

Digital Timer: Created a digital timer that facilitates the implementation of the Pomodoro Technique.
User-Friendly Interface: Developed a user-friendly interface with a seven-segment display to visualize the countdown of work and break intervals.
Timer Control: Enabled users to shift between work and break session, and then reset the timer, ensuring flexibility in managing their tasks.
Audio Alerts (Optional): Implemented audio alerts, such as buzzer notifications, to signal the end of work intervals and breaks.

State Machine Representation:

States:

WORK: Initial state where the 25-minute work session countdown starts upon button press.
BREAK: Transitioned to from WORK state when the work session ends; starts a 5-minute break countdown upon button press.
STOP: Can be reached from either WORK or BREAK state by pressing the stop/reset button, resetting the timer to zero.

Transitions:

- WORK to BREAK: Triggered by the "work" button press when in the WORK state. Initiates the transition from WORK to BREAK state.
- BREAK to WORK: Automatically transitioned from BREAK to WORK state when the 5-minute break session ends.
- WORK/BREAK to STOP: Transitioned to STOP state upon pressing the stop/reset button, regardless of the current state.

User Interaction:

Ease of Use:

The timer circuit was designed with user-friendliness in mind, offering intuitive button functionalities and clear display readability. The button functionalities were straightforward:

- **Work Button (Start Work Session): Initiates a 25-minute work session countdown upon a single press. The button's responsiveness ensured a quick start to the work session without any delay.**
- **Break Button (Start Break Session): Commences a 5-minute break countdown with a simple press. Its consistent functionality allowed for an easy transition between work and break sessions.**
- **Stop Button (Reset Timer): Reset the timer to zero and halted any ongoing countdowns, providing a convenient way to stop and reset the timer at any point.**

Readability of Displays: **The seven-segment displays provided clear and easily readable representations of the countdown timers.**

Conclusion:

The pomodoro timer with time displayed in a seven segment display and controlled by user input buttons.

The Pomodoro timer circuit demonstrated commendable accuracy in maintaining the prescribed 25-minute work sessions and 5-minute break sessions. Its consistent performance in accurately tracking time intervals contributed significantly to its effectiveness as a productivity tool. The users' positive experiences and the timer's reliability in adhering to designated durations affirm its suitability for aiding time management practices.

The pomodoro timer with a seven segment display offers a practical solution to a common problem - time management and productivity .

The project not only aims to improve individual productivity but also serves as an educational tool.