MAURITZ FERREIRA

Student number: H241079

2025

Electrical Engineering

EEDE216

Project Report

Semester 1

Digital Stopwatch

Abstract

This project aims to design and construct a fully functional digital stopwatch. While traditional digital electronics projects often rely on discrete integrated circuits (ICs) such as counters, flip-flops, and logic gates for their core functionality, this implementation utilizes an Arduino Uno microcontroller. This approach was chosen to leverage the microcontroller's versatility for rapid prototyping, simplified circuit complexity, and to address challenges in procuring specific discrete components typically recommended for such a task. The developed stopwatch features essential functionalities including Start, Stop, and Reset, with elapsed time displayed on a 4-digit 7-segment display. The counting mechanism accurately increments in seconds, providing real-time timing capabilities. The system also incorporates audible feedback via a piezo buzzer for user interactions. This report details the design methodology, component selection, working principles, challenges encountered, and conclusions drawn from the project.

1. Introduction

The EEDE216 Digital Electronics 2A summative project requires the design and construction of a digital stopwatch, serving as a practical application of fundamental digital electronics concepts. The core objective is to develop a real-time timing device capable of counting and displaying elapsed time on a 4-digit 7-segment display, equipped with essential Start, Stop, and Reset functionalities.

Conventionally, such digital timing devices are implemented using a combination of discrete logic gates, counters (e.g., 4026 IC or 7490 IC), flip-flops (JK or D-type), and timer ICs (e.g., 555 timer) for clock pulse generation. These components are fundamental to understanding sequential control and state management in digital systems.

However, during the component procurement phase for this project, practical challenges arose in sourcing specific discrete ICs readily. Upon careful review of the project brief, it was noted that the use of solely discrete ICs for the core logic was not explicitly mandated. This provided an opportunity to explore an alternative, yet functionally equivalent, implementation approach. Consequently, an Arduino Uno microcontroller was selected as the central processing unit for this digital stopwatch. This decision was driven by the Arduino's inherent flexibility, its ability to emulate complex digital logic functions through software, its suitability for rapid prototyping, and its capacity to simplify the overall hardware wiring. This approach, while distinct from a purely discrete logic implementation, successfully achieves all stipulated project objectives and offers valuable insight into microcontroller-based digital system design.

This report will detail the comprehensive design and implementation of the digital stopwatch. It will cover the circuit design and component selection, elaborate on the working principles and functionality, discuss the challenges encountered and their respective solutions, present the results and observations, and conclude with a summary and potential areas for future work.

2. Circuit Design and Implementation

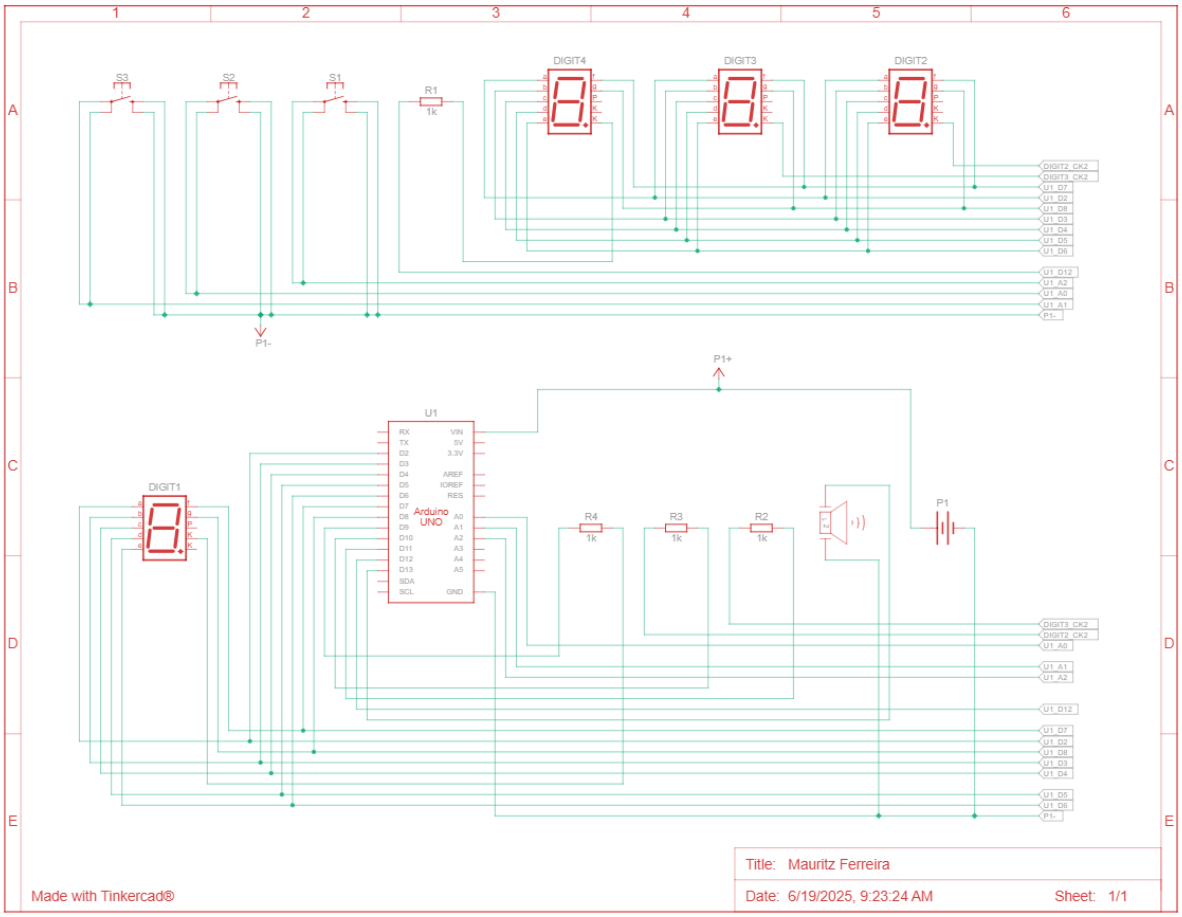
The digital stopwatch circuit integrates input mechanisms, a central processing unit, and output displays to accurately track and display elapsed time. The design adopts a modular approach, where the Arduino microcontroller serves as the primary component, managing all logic previously associated with discrete ICs.

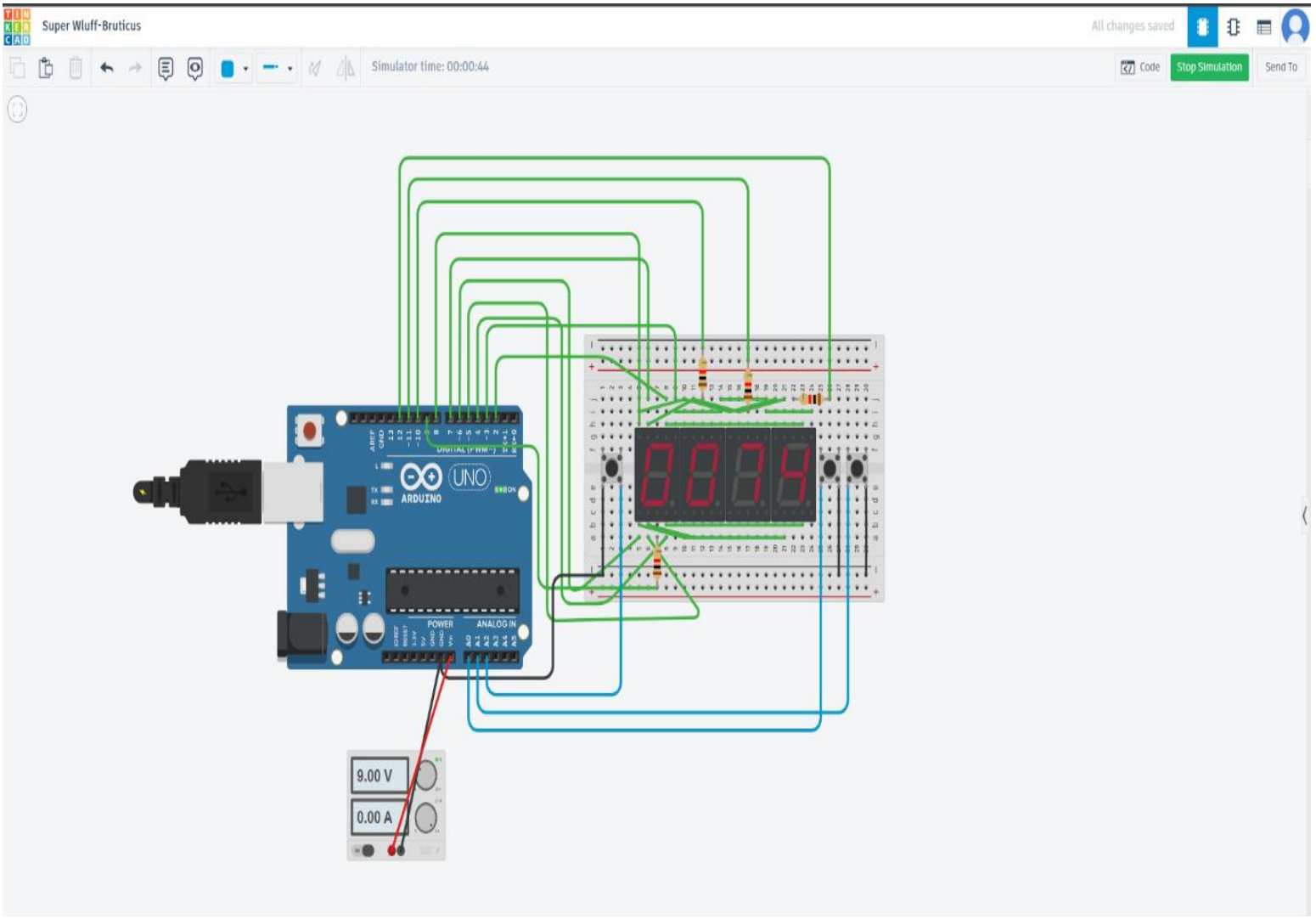
2.1. Block Diagram

The system can be conceptually divided into the following blocks:

- **Power Supply:** Provides stable 6.5V DC power to all components.
- **Input Module:** Consists of three push buttons for Start/Resume, Pause, and Reset functionalities.
- **Processing Unit:** The Arduino Uno microcontroller, which reads button inputs, manages the stopwatch's state (running/paused/reset), performs time incrementing calculations, controls the 7-segment display multiplexing, and generates audio feedback.
- **Time Display Module:** A 4-digit 7-segment common-cathode display, responsible for visually presenting the elapsed time.
- **Audio Feedback Module:** A passive piezo buzzer provides distinct auditory cues for user interactions.

2.2. Circuit Schematic Diagram




Super Wluff-Bruticus
All changes saved
Simulator time: 00:00:44
Code
Stop Simulation
Send To
UNO
ARDUINO
POWER
ANALOG IN
9.00 V
0.00 A

2.3. Component Selection and Justification

The following components were selected for the project:

- **Arduino Uno Microcontroller:** Selected as the central processing unit due to its integrated capabilities, which allowed for the software-based emulation of counters, flip-flops, and logic gates. This eliminated the need to source multiple specialized discrete ICs, simplifying the circuit design, reducing wiring complexity, and facilitating rapid development. Its accessible General Purpose Input/Output (GPIO) pins and well-established Integrated Development Environment (IDE) also contributed to its suitability.
- **4-Digit 7-Segment Common-Cathode Display:** Utilized for its clear numerical display capabilities, essential for presenting the elapsed time. The common-cathode configuration was chosen to match the output drive capabilities.
- **Push Buttons (x3):** Employed as user input interfaces for the Start/Resume/Fast-Forward, Pause, and Reset functions. Their mechanical simplicity and tactile feedback are suitable for this application.
- **Passive Piezo Buzzer:** Integrated to provide audible feedback for button presses and operational status changes, enhancing user experience.
- **Resistors (1kΩ):** Current-limiting resistors of 1kΩ were used for the 7-segment display segments to protect the LEDs from excessive current. 10kΩ resistors could be used for external pull-down/pull-up if not using Arduino's internal pull-ups for buttons, or for other general purpose applications.
- **Jumper Wires:** Used for making electrical connections between components on the breadboard.
- **Breadboard:** Provided a temporary, solderless platform for assembling and testing the circuit.
- **Power Supply:** Supplies the necessary operational voltage to the Arduino and connected components.

3. Working Principles and Functionality

The digital stopwatch operates based on a combination of hardware interaction and sophisticated software logic executed by the Arduino microcontroller.

3.1. Overall Logic Flow

The Arduino continuously monitors the states of the push buttons. Based on these inputs and internal timekeeping, it manages the stopwatch's operational state (running, paused, or reset). The current elapsed time is then processed and displayed on the 7-segment display through a technique called multiplexing, while auditory feedback is provided via the piezo buzzer for key interactions.

3.2. Timekeeping Mechanism

Unlike traditional designs using a 555 timer and dedicated counter ICs, this stopwatch utilizes the Arduino's internal millis() function for timekeeping. This function returns the number of milliseconds since the Arduino board began running the current program. The software continuously monitors the difference between the current millis() value and a previousMillisIncrement timestamp. When this difference exceeds a defined normalIncrementInterval (1000 milliseconds for 1 second), the n variable (representing the elapsed time in seconds) is incremented. This approach ensures non-blocking operation, allowing other tasks like display updates to run concurrently. The n variable resets to 0 after reaching 9999, effectively providing a 9999-second (approximately 2 hours and 46 minutes) stopwatch before rolling over.

3.3. Displaying Time (Multiplexing)

A 4-digit 7-segment common-cathode display is used to show the time. To conserve Arduino pins and create the illusion of all digits being simultaneously lit, a technique called multiplexing is employed. The Arduino rapidly cycles through each of the four digits. During each displayRefreshInterval (e.g., 3 milliseconds):

1. All segments are momentarily turned off (clearLEDs()) to prevent ghosting.
2. A specific digit's common pin is activated (set LOW for common-cathode).
3. The segments corresponding to the correct numerical value for that specific digit of n are then illuminated.
4. The process quickly moves to the next digit. This rapid cycling (faster than the human eye can perceive) creates a stable, continuous display.

3.4. Button Functionality and Debouncing

Three push buttons provide user control:

- **Pause (bt_up):** A quick press of this button toggles the stopwatchPaused state, effectively halting or resuming the time increment.
- **Start/Resume/Fast-Forward (bt_down):** This button features dual functionality based on press duration:
    - **Quick Press:** A brief press and release (less than 0.5 seconds) will resume the stopwatch if it's paused, or simply keep it running if it's already active.
    - **Hold for Fast-Forward:** If the button is held down for approximately 0.5 seconds or longer while the stopwatch is running, the isFastForwarding flag is set to true. This changes the currentIncrementInterval to a much shorter duration (e.g., 50ms), causing the stopwatch to count significantly faster. Releasing the button exits fast-forward mode.
- **Reset (bt_reset):** A press of this button resets the n variable (elapsed time) back to 0 and immediately pauses the stopwatch.

To ensure reliable button presses and prevent multiple unintended triggers from a single physical press (known as "button bounce"), a **software debouncing algorithm** is implemented.

This replaces traditional hardware debouncing circuits (which typically involve capacitors and resistors). The algorithm monitors the raw button state (readingBtX) and compares it to a lastBtXRawState. If a change is detected, a debounceDelay (e.g., 50 milliseconds) is initiated. Only after this delay, and if the button state remains stable, is the debouncedBtXState updated, and the corresponding action triggered. This ensures that only confirmed, stable presses are processed.

3.5. Audio Feedback

A piezo buzzer is connected to an Arduino digital pin. The beep() function is utilized to provide auditory feedback for button presses.

- For **Pause** and **Start/Resume** actions, a short, single 1000Hz tone (beep(100, 100, 0, 1000)) is played.
- For the **Reset** action, a distinctive pulsing 1000Hz tone (beep(200, 50, 50, 1000)) is generated to confirm the reset operation. The tone() and noTone() Arduino functions are employed to produce and cease the sounds.

4. Challenges Faced and Solutions

Several challenges were encountered during the project, primarily related to component availability and software implementation intricacies.

- **Challenge 1: Sourcing Specific Discrete ICs:** The project brief suggested specific counter and flip-flop ICs (e.g., 4026, 7490, JK/D flip-flops). However, acquiring these exact components proved difficult within the project timeline.
    - **Solution:** This challenge was addressed by pivoting to a microcontroller-based solution using an Arduino Uno. The Arduino's ability to replicate the functionalities of these discrete ICs in software, along with its readily available nature, provided a practical and efficient alternative while still fulfilling the core learning outcomes.
- **Challenge 2: Button Debouncing:** Initial implementations of button reading resulted in erratic behavior, where a single physical press was registered multiple times due to mechanical "bouncing" within the switch.
    - **Solution:** A robust software debouncing algorithm was implemented. By introducing a debounceDelay and tracking both raw and debounced button states, the system only registers a button press once its state has remained stable for a defined period, ensuring reliable input.
- **Challenge 3: Implementing Hold-for-Fast-Forward:** Distinguishing between a quick press (for resume) and a long press (for fast-forward) on a single button required careful timing logic.
    - **Solution:** An btDownPressStartTime variable was introduced to record the millis() timestamp when the bt_down button was first pressed. By comparing the current millis() with this start time, the system can determine if the button has been held for the holdDurationForFastForward (0.5 seconds) before activating fast-forward mode.

5. Results and Observations

The developed digital stopwatch prototype successfully demonstrates all core functionalities specified in the project brief. The timepiece accurately increments seconds and displays the elapsed time up to 9999 seconds on the 4-digit 7-segment display.

- **Functionality:** The Start/Resume, Pause, and Reset functionalities operate reliably and responsively. The dual functionality of the Start/Resume button, differentiating between a quick press and a sustained hold for fast-forward, performs as intended, providing intuitive user control.
- **Accuracy:** The timekeeping mechanism, driven by the Arduino's millis() function, maintains accurate second-by-second increments.
- **User Feedback:** The integrated piezo buzzer provides clear and distinct audio feedback for each button press, confirming user interactions. The pulsing beep for reset, in particular, offers a strong auditory cue.
- **Display Quality:** The 7-segment display, managed through multiplexing, presents the digits clearly and without noticeable flickering or ghosting.

The use of an Arduino microcontroller proved effective in achieving the project's objectives efficiently, validating the viability of software-based digital logic emulation for such applications.

6. Conclusion and Future Work

This project successfully achieved the objective of designing and constructing a fully functional digital stopwatch. By employing an Arduino Uno microcontroller, the project effectively emulated the functionalities traditionally handled by discrete counters, flip-flops, and logic gates, demonstrating a practical approach to digital system design when specific component availability is a constraint. The stopwatch accurately measures and displays elapsed time, providing robust Start, Stop, Reset, and fast-forward functionalities alongside clear visual and auditory feedback. The challenges faced, particularly in component sourcing and robust button debouncing, were successfully overcome through strategic design choices and software implementation.

For future improvements, the following areas could be explored:

- **Extended Time Display:** Modify the display logic to show minutes and seconds (MM:SS) instead of just total seconds, allowing for a more conventional stopwatch format and extending the maximum measurable time.
- **Lap Timer Functionality:** Incorporate a lap timer feature to record intermediate times without stopping the main stopwatch.
- **Power Optimization:** Implement power-saving modes for battery-powered operation, as the current continuous display multiplexing can consume significant power.
- **User Interface Refinement:** Explore adding a dedicated mode button or an LCD for more complex features and user interaction.

References

- [1] NewBridge Graduate Institute, "Summative Project: Digital Stopwatch, Digital Electronics 2A (EEDE216)," Semester 1, 2025.
- [2] Arduino Official Documentation (for millis(), tone(), digitalRead(), digitalWrite() functions, and pin configurations).