CAB202 Assignment:

Student Name: Jol Singh

Student Number: n10755756

TinkerCAD Link: https://www.tinkercad.com/things/gtXUAb15jSR-pomodoro-study-timer/editel?sharecode=W3UGXAv7P4ECZ6d3zq8cXp_PHQ_792QrFwXmy1QsXTE

1 Introduction

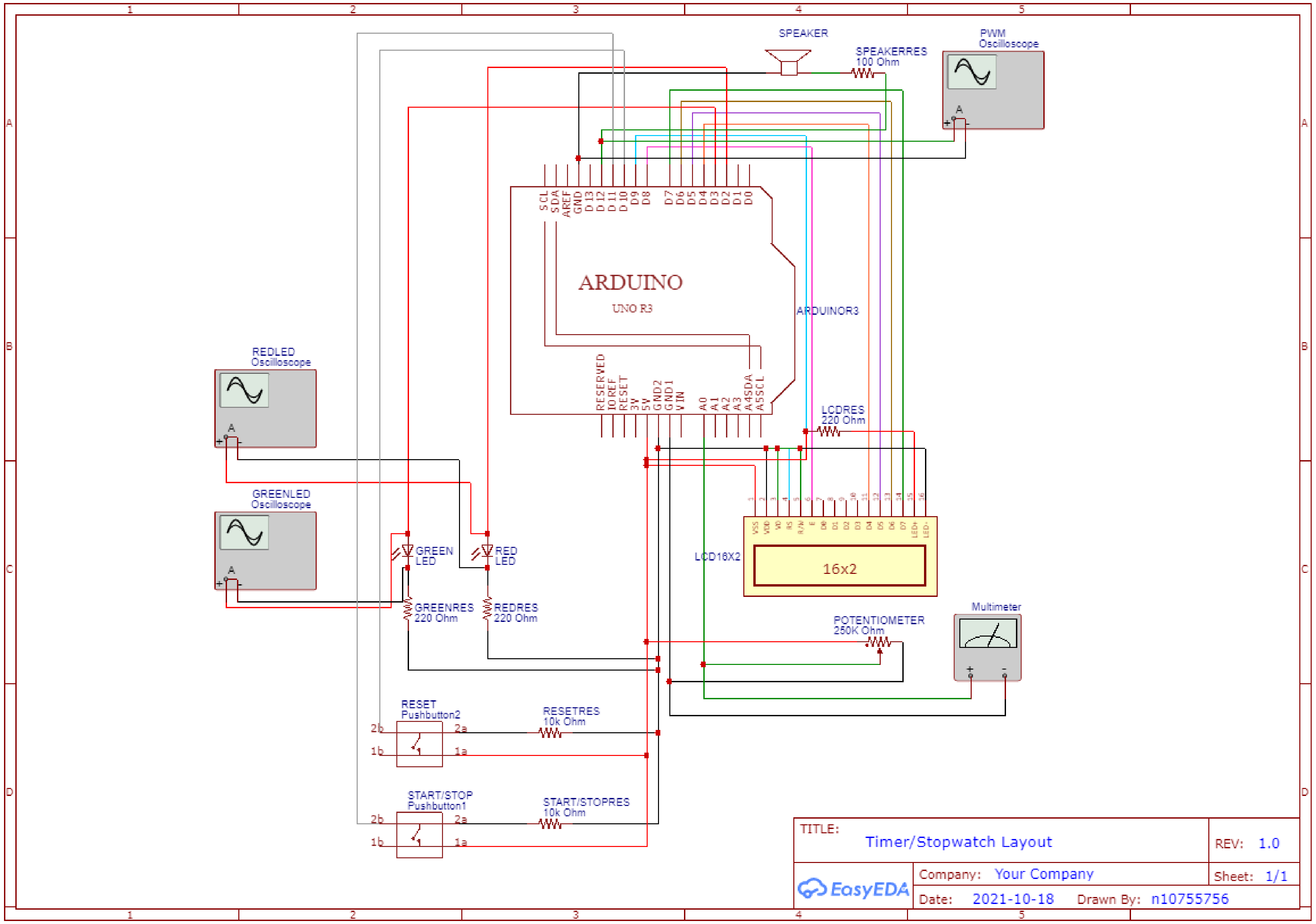
The microcontroller-based product I've created is a study timer. I've taken inspiration from the Pomodoro (tomato in Italian) technique that was developed in the late 1980s by Francesco Cirillo, who had difficulties focusing on studies, and found a tomato shaped kitchen timer to help him focus, thus, creating this technique. So, taking inspiration from this technique, it allowed me to create a similar product that's meant to be utilised in helping procrastinators improve their time management skills, organisation skills, productivity, and assisting them in focusing on more important tasks such as their studies, work, or any other important tasks they may have. The product presents users with two push buttons where each button performs a certain task, and their corresponding LED lights up, as well as, turning on the LCD screen to display the timer content. The concept is that users choose a task to complete, and once chosen, they can start the timer to measure how long it takes them to complete the task.

Furthermore, the implementation of these visual components enforces users to develop further improvement upon themselves as the time elapsed on the timer is enough to motivate them to have better time management, and increased productivity. I believe these implementations are sufficient enough in satisfying the digital input and output switches, LED, debouncing, and LCD learning outcomes. Moving on, I believe my application demonstrates proficiency with the following specific functional capabilities, each of which is directly related to CAB202 learning outcomes, and the following table provides further detail of this proficiency, as well as, detailing how each component contributes to the overall functionality of the application.

Functionality Implemented & Proficiency Related to CAB202 Learning Outcomes

Digital I/O - Switch	*Two push buttons have been implemented to allow certain tasks to be performed such as the timer being started, stopped or reset, as well as LEDs being implemented for each button that light up when their corresponding button is pressed/clicked.*
Digital I/O – Debouncing	*For each pushbuttons implemented an interrupt-based debouncing method has been integrated with them, it follows a similar structure to the Topic 9 AMS 3 code, where history, mask and debounced state are utilised to perform debouncing. The history is shifted across by one, then the current state of the pushbutton is added to history, and finally, an if code block is programmed to set the state of the button for example if debounced_state == 1 means that the button is pressed, and if debounced_state == 0 means the buttons not pressed.*
Digital I/O – LED	*Two LEDs, a green and red LED have been implemented where the green LED provides a visual representation of when the timer is running and the other when the timer is resetting. Additionally, both LEDs are integrated with a specific task so when one button is clicked it tuns on the red LED, or if the other button is clicked the green LED turns on.*
Analog Input – ADC	*A potentiometer was combined with analog to digital conversion programming to allow control over the piezo'/speakers volume level, and a multimeter was also connected to ensure voltage was going through the potentiometer. Furthermore, a larger portion of coding for ADC follows closely to the first example of topic 10, and I've also implemented the uart function, uart_putstring(), that allows ADC values to be outputted into the serial monitor to ensure ADC is working properly in my program.*
Analog Output – PWM	*A piezo/speaker was integrated PWM programming where the duty cycle has been set to a constant value equal to DC = sn/2+25. Where sn are the last two digits of my student number, 56, this means the duty cycle is DC= (24/2 + 25) % = 53%. Rounding to the nearest whole number, sets my duty cycle at 50%. It was also integrated with the reset pushbutton so that when the user resets the timer, the red LED turns on, and a tone plays' alerting the user that the timer has been reset, and the oscilloscope was placed to display the square-wave produced by the speaker. In addition, majority of the PWM code follows closely to the second example provided in the topic 10 lecture notes.*
Serial I/O – UART	*Serial Communication such as UART functions and definitions were integrated with the pushbuttons so that data can be recorded once clicked/pressed, and outputs into the serial monitor to show that the data being transmitted is actually received. Recording stops when the reset button is clicked/pressed. Moreover, the code for UART definitions and functions were provided by the first example in topic 10.*
LCD	*A 16x2 LCD Screen has been implemented to display the LCD start-up text, and then simulate the timer once the start/stop button is pressed, and when the reset button is clicked it resets the timer and updates the screen back to a clean slate. The LiquidCrystal Library was also included so that the LCD could operate, and functions such as lcd.print(), lcd.setCursor() were implemented to allow these text messages to be displayed and positioned on the LCD, as well as, the actual application itself functioning on the LCD screen. In addition, another function called lcd.being() was used to set up the LCD in 4-pin mode.*
Timers (other than debouncing or PWM)	*Two timer modules were initialised in this program, Timer 1 and Timer 0. Timer 0 was used for PWM, and other than debouncing, Timer 1 was also used to increment the integer values in my timer method. So, the overflow_count is called into a double function for the elapsed time, that then allows the timer to be incremented and displayed on the LCD screen once the start/stop pushbutton is clicked/pressed. When the reset button is clicked/pressed it resets the overflow_count back to zero. Additionally, I also included the _delay_ms() function from topic 11 example 1, to create a delay in milliseconds between certain functions.*

2 Schematic


SPEAKER
SPEAKERRES 100 Ohm
PWM Oscilloscope
ARDUINO
UNO R3
ARDUINOR3
REDLED Oscilloscope
GREENLED Oscilloscope
GREEN LED
RED LED
GREENRES 220 Ohm
REDRES 220 Ohm
LCDRES 220 Ohm
LCD16X2
16x2
POTENTIOMETER 250K Ohm
Multimeter
RESET Pushbutton2
RESETRES 10k Ohm
START/STOP Pushbutton1
START/STOPRES 10k Ohm
TITLE: Timer/Stopwatch Layout
REV: 1.0
Company: Your Company
Sheet: 1/1
EasyEDA
Date: 2021-10-18
Drawn By: n10755756

3 Wiring Instructions

Breadboard: place a breadboard anywhere; connect ground (negative) section to any ground port on arduino board; connect power (positive) section to 5V port on arduino board.

LED 1 (Green): place a green LED close to the centre of the breadboard; connect Port D Pin 3 to the anode side of the green LED; connect the cathode side of the green LED to one end of a distinct 220 Ohm resistor; connect the other end of the resistor to ground (negative) section of the breadboard.

LED 2 (Red): place a red LED on the breadboard to the left of the green LED; connect Port D Pin 2 to the anode side of the red LED; connect the cathode side of the red LED to one end of a distinct 220 Ohm resistor; connect the other end of the resistor to ground (negative) section of the breadboard.

Pushbutton 1 (Start/Stop): place pushbutton 1 a few spaces to the right of the red LED on the breadboard; make sure to place one side of the pushbutton pins (Terminal 1b & 2b) on one half of the breadboard, and place other pins (Terminal 1a & 2a) on the other half of the breadboard; connect Port B Pin 3 to Terminal 2b; connect Terminal 2a to one end of a distinct 10 kilo Ohm resistor; connect the other end of the resistor to ground (negative) section of the breadboard; connect Terminal 1a to power (positive) section of the breadboard.

Pushbutton 2 (Reset): place pushbutton 2 a few spaces to the left of pushbutton 1 on the breadboard; make sure to place one side of the pushbutton pins (Terminal 1b & 2b) on one half of the breadboard, and place other pins (Terminal 1a & 2a) on the other half of the breadboard; connect Port B Pin 2 to Terminal 2b; connect Terminal 2a to one end of a distinct 10 kilo Ohm resistor; connect the other end of the resistor to ground (negative) section of the breadboard; connect Terminal 1a to power (positive) section of the breadboard.

16x2 LCD: place LCD screen to the right of breadboard; connect ground port to any ground port on arduino board; connect power port to 5V port on arduino board; connect contrast port to any ground port on arduino board; connect register select to Port B Pin 1; connect read/write to any ground port on arduino board; connect enable to Port B Pin 0; connect DB4 to Port B Pin 4; connect DB5 to Port B Pin 5; connect DB6 to Port B Pin 6; connect DB7 to Port B Pin 7; connect LED Anode to one end of a distinct 220 Ohm resistor; connect the other end of the resistor to the 5V port on the arduino board; connect LED Cathode to any ground port on the arduino board.

Potentiometer: place potentiometer a few spaces above LCD; set to default resistance of 250 kilo ohms; connect Terminal 1 to any ground port on arduino board; connect Wiper to Port C Pin 0; connect Terminal 2 to 5V port on arduino board.

Piezo (Speaker): place piezo few spaces above arduino board; connect Positive to one end of a distinct 100 Ohm resistor; connect the other end of the resistor to Port D Pin 4; connect Negative to any ground port on arduino board.

Oscilloscope (Green LED): place oscilloscope few spaces to the left of breadboard; connect positive to anode side of green LED; connect negative to cathode side of green LED.

Oscilloscope (Red LED): place oscilloscope few spaces to the left of breadboard; connect positive to anode side of red LED; connect negative to cathode side of red LED.

Oscilloscope (Speaker): place oscilloscope few spaces above speaker; connect positive to Port B Pin 4 same port as the speaker; connect negative to any ground port on arduino board.

Multimeter (Potentiometer): place multimeter few spaces to the right of potentiometer; connect positive to Port C Pin 0 same port as potentiometer; connect negative to any ground port on arduino board.