Arduino Timer

Miguel Tamayo

May 5th, 2022

Contents

List of Figures

List of Tables

1 Project Overview

The objective of this project is to introduce the Arduino microcontroller its ability to read inputs and turn them into useful outputs. The input data will come from three buttons which will manipulate a timer. The output is then displayed through a 7-segment 4 digit display by using bit shifting to turn on the different segments and digits in a pattern.

The outcome of the project is to demonstrate a working timer that allows the user to set the preferred time by using two buttons and start a count-down sequence once the start button pressed.

The components needed for this project are:

- Breadboard
- Arduino Nano
- Button (x3)
- Resistor: $10K\Omega$ (x3)
- 7-Segment 4 Digit Display
- Wires

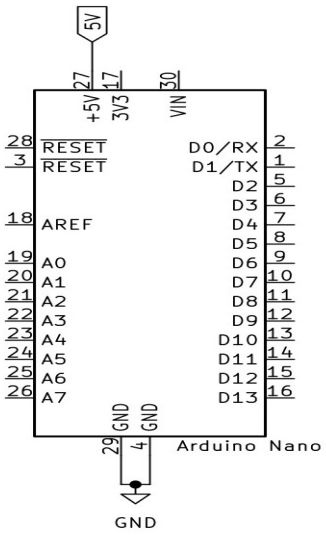
The software products used in the project are:

- Arduino IDE
- KiCad

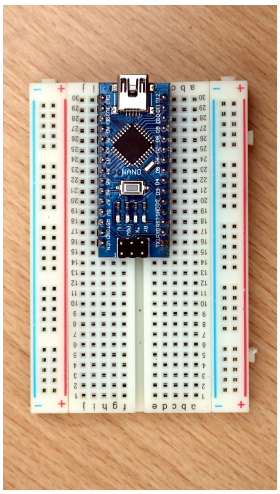
Note: The 7-Segment 4 Digit display used is active low. If an active high display is used, the code must be changed in order to successfully turn on the segments in the display. This is covered in the software section.

2 Hardware Design

2.1 Microcontroller (Arduino Nano)



(a) Arduino Nano Schematic



(b) Arduino Nano Size

Figure 1: Arduino Nano Characteristics

A microcontroller is a small computer designed to do specific tasks of embedded systems such as controlling motors, LEDs, receiving, and sending signals. Arduino has a variety of microcontrollers but the Nano was chosen due to its compact shape which allows the project to be built on a single breadboard (Figure 1).

The Arduino Nano acts as the main computer of the timer. It monitors the signals coming from the buttons and displays the correct number in seconds on the display. The microcontroller also manages the state machine (see Software Design) and changes states according to the signals from buttons and an internal timer.

2.2 Buttons

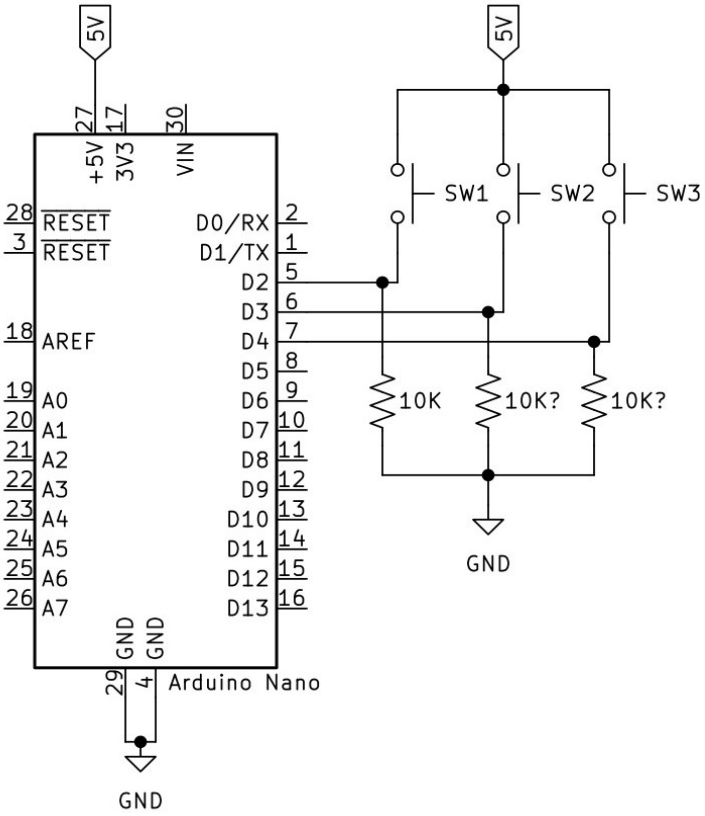


Figure 2: Push Buttons Schematic

A pushbutton acts as a mechanical switch to toggle a circuit by closing it or opening it. When working with microcontrollers, it is important to attach a resistor along with the pushbutton in order to stabilize the signal. When the pushbutton is not pressed, the voltage being read in the input pin can vary between $0V$ and the power voltage (usually $5V$) therefore giving a mixed signal between 0 and 1 to the microcontroller which can falsely trigger the input.

To fix this stability issue, a resistor connects the microcontroller pin to ground $0V$ as shown in Figure 2. This connection ensures that the digital signal coming into the microcontroller is stabilized at 0 whenever the pushbutton is no pressed.

This project uses 3 buttons to control the timer. Two buttons are used to increment and decrement the timer and the last button is used to start the count-down sequence. The buttons are connected to the pins [324] respectively. The connection is shown in Figure 2 above.

2.3 7-Segment 4-Digit LED Display

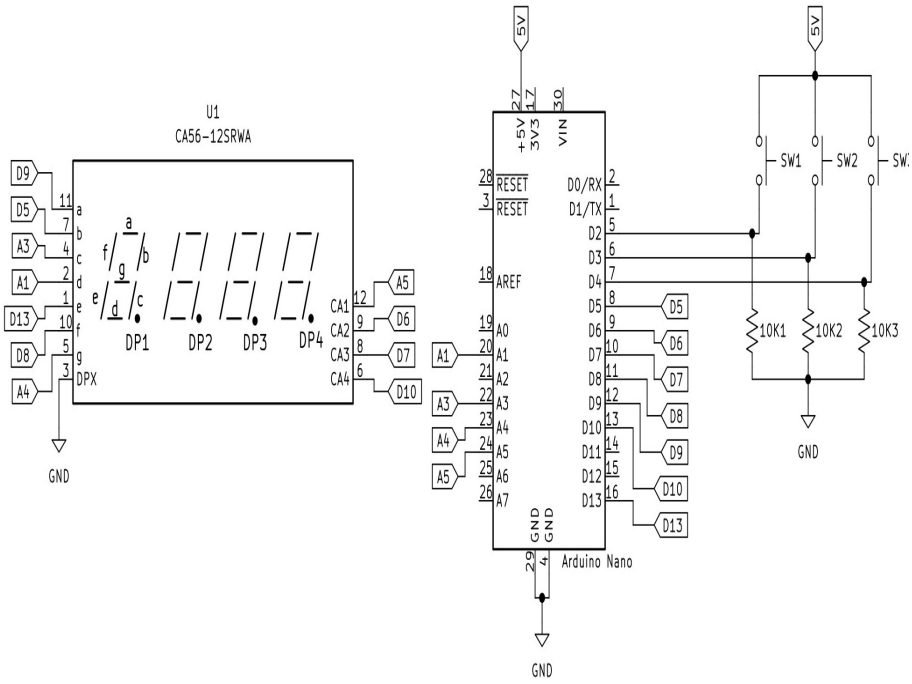


Figure 3: 7-Segment 4-Digit LED Display Schematic

A 7-segment display consists of 7 LEDs arranged in a way that they can represent numbers in the range 0-9. These segments can be common anode or cathode. If they are a common anode, it means they are connected to a common positive voltage supply and are turned on by driving their respective pin to ground. Common cathode is the opposite by having the segments connected to a common ground and turning them on through a positive voltage on the designated pin.

A 4-digit display consists of 4 7-segments next to each other. In this project, the 4-digit module has common anode segments. To save space, the 4 digit module only contains 7 pins for the 7 LEDs that make up the segments. These are connected to pins [$95A3A1138A4$]. The Software Design section will explain in detail how the pins are driven in order to display a different number in each segment while only being able to control 7 LEDs and not the 28 total individual LEDs on their own. In addition to the software design, there are 4 signals that control the individual segments D1 - D4. These are connected to pins [$A56710$] respectively. The connection is shown in Figure 3 above.

3 Software Design

3.1 State Machine

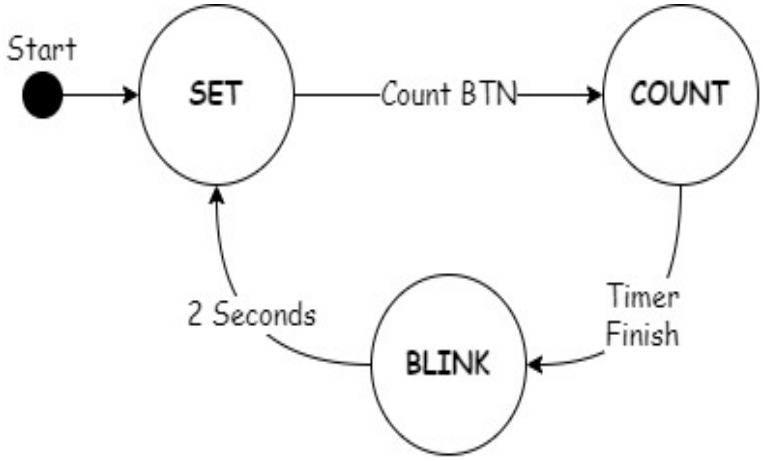


Figure 4: State Machine

The state machine (Figure 4) consists of the following 3 states:

1. SET
2. COUNT
3. BLINK

The SET state allows the user to use two of the buttons to input the amount of time for the timer. Once the third button is pressed, the state machine advances to the COUNT state where the timer begins to count down. Once all the segments reach 0 (meaning the time is done), it transitions to the BLINK state. In this state, the segments blink for 2 seconds to indicate the time being done. Finally, the state machine loops back to the start.

3.2 Digit Tracking

Since each 7-segment can only hold a single digit number, each decimal place of time is assigned to each digit display with the rightmost display being the ones place and so on. These numbers are tracked in software through an array with the initial state [0000]. Instead of having a single variable increase, each number in the array increases to 9 and resets on the next increment. For example, if the counter is currently at [0009], on the next increment it will look like [0010]. This allows us to display the number 10 correctly on the 7-segment displays.

3.3 Segment Display

The 4-digit module contains 8 connections to control the LEDs in the 7-segment displays. 7 of those control the number to be displayed, and the 8th one controls the DP point (not used in this project). The segments are labeled with letters $A \rightarrow G$ as shown in Figure 5.

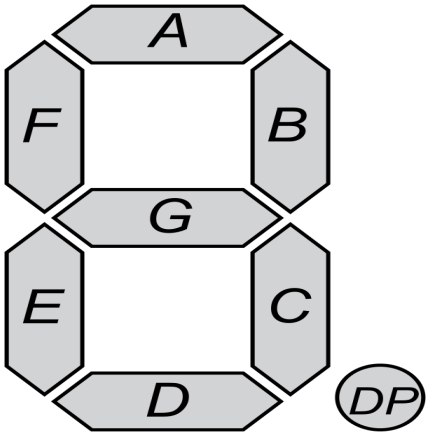


Figure 5: 7-Segment Display Labels

To turn on these segments in the right sequence, we can think of the number we want to display as a sequence of bits representing high and low states. Since the least significant is the rightmost, this represents our A letter and the leftmost bit represents the letter G. In order to represent the number 0 in the segments, the binary sequence looks as follows: $0b00111111$. What this sequence says is that letters $A \rightarrow F$ are turned on and letter G is off. There is an extra 0 at the end which represents the DP point. A table with all the sequences of numbers from 0 to 9 is shown below.

Number	Bit Code	DP	G	F	E	D	C	B	A
0	0b00111111	0	0	0	1	1	1	1	1
1	0b00000110	0	0	0	0	0	1	1	0
2	0b01011011	0	1	0	1	1	0	1	1
3	0b01001111	0	1	0	0	1	1	1	1
4	0b01100110	0	1	1	0	0	1	1	0
5	0b01101101	0	1	1	0	1	1	0	1
6	0b01111101	0	1	1	1	1	1	0	1
7	0b00000111	0	0	0	0	0	1	1	1
8	0b01111111	0	1	1	1	1	1	1	1
9	0b01101111	0	1	1	0	1	1	1	1

Table 1: Sequence For Numbers on 7-Segment Display

To actually use this sequence and light up LEDs, a for loop is used to iterate through the 8 bits and LEDs. The LED pins are contained in an array in range $A \rightarrow G$ and the different numbers are also in an array of bit sequences with numbers $0 \rightarrow 9$. The for loop iterates through the array and at the same time uses bit shifting to check if the bit on

the sequence is high or low. If the bit is high, we set the array at the LED to low since the segments are active low. This iteration creates the different numbers.

3.4 Digit Display

Now that we know how to display different numbers on the segments, we can turn on the displays to show the number we want. If you turn on all the displays at once, you might notice that they all show the same number. This is because the 7 pins control the 7 segments for all the different digits. To take care of this, we need to iterate through each different display with a for loop and show the corresponding number to that display and so on. You can do this slowly at first to see how each number changes depending on the display. However, if the delay between each digit is very low, our eyes will not be able to see the iteration because of how fast it is going. This will give the effect that all the numbers are turned on at the same time.

4 Results

The final build of the Arduino timer is shown in Figure 6 below:

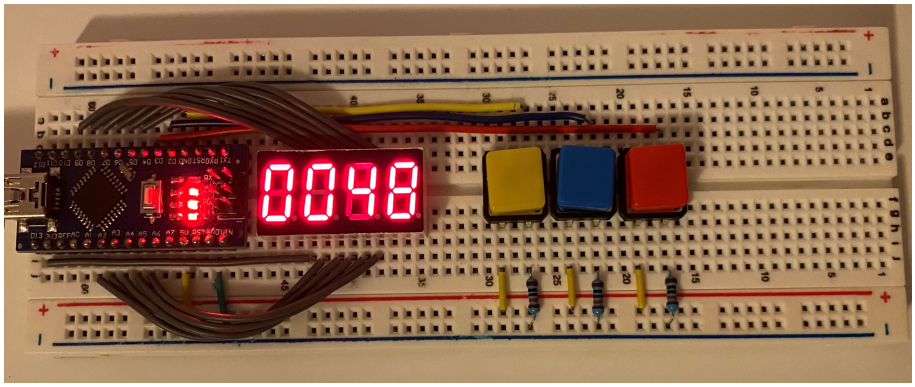


Figure 6: Finished Arduino Timer

A video showing how this timer works can be found in the YouTube video linked here. Please use the link below if the hyperlink above does not work.

- https://www.youtube.com/watch?v=21du34BC54U

4.1 Discussion

4.1.1 Assumptions and Errors

In the project, it was assumed that the voltage being applied to the segments was not high enough to burn them therefore resistors were not included in the build. This decision was also taken in order to save space due to the small area of work. This assumption was valid since the prototype showed that the segments worked fine without the resistors. Even though the lack of resistors for the segments proved that the timer worked without them, it was observed that the 4-digit module's temperature rose, and the heat could be felt by touch.

When testing the design and increasing the speed at which the code iterated through the different digits, there was leak current that turned on segments from different digits when they should not be turned on. This error is shown in Figure 7 with the unwanted LEDs being circled in blue. Leak current was concluded to be the cause due to the lower brightness of the segment compared to those who were meant to be on. With such low brightness, it must mean that the segment is no having enough current pass through it in order to fully turn it on.

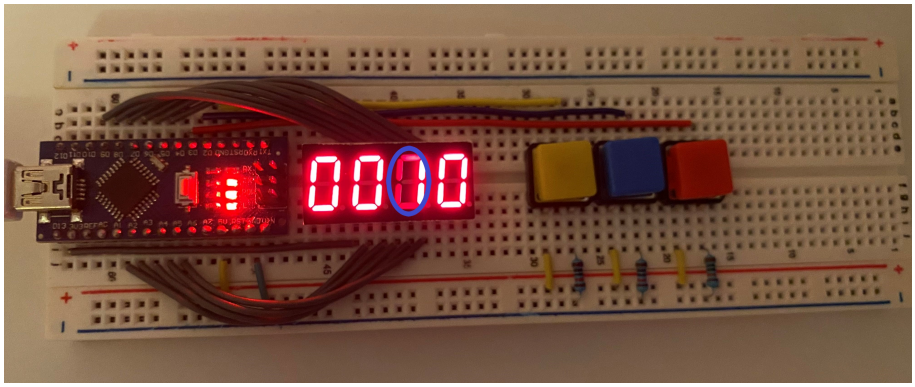


Figure 7: 7-Segment Display Labels

4.1.2 Future Improvements

The signal sent from the microcontroller is $5V$ and cannot be changed. Therefore, in order to mitigate the heating problem in the 4-digit module, resistors can be used to limit the voltage applied to the module.

One solution to the leak current can be provided through the use of transistors. Since these devices act as a switch depending on the base voltage, we can apply the voltage from the microcontroller. By controlling the voltage, we can limit the current provided to the digits and prevent leak current from flowing through.

This first design lets the user know when the timer is finished by flashing the 4-digit module for 2 seconds. However, if the user needs to step away there is no way of know whether the timer is finished or not without having to come back and check on it. With a larger workspace, a buzzer can be added in order to make an alarm that notifies the user even after stepping away.

5 Conclusion

The objective of this project is to introduce the Arduino microcontroller its ability to read inputs and turn them into useful outputs. This was accomplished by creating a timer using the Arduino Nano, 4-digit 7 segment display module, and 3 buttons. The final product of this project is a working timer which uses 2 buttons to increase and lower the time and a final button to start the count-down.

The future design can improve heating issues by adding resistors to limit the voltage sent by the microcontroller since it cannot be changed through software. Additionally, using transistors to control the digits will prevent from leakage current to flow and turn on unwanted segments while displaying numbers. Lastly, a buzzer can be added to notify when the timer has finished without needing to look at it.

6 References

1. What is a Microcontroller
   https://www.tutorialspoint.com/microprocessor/microcontrollers_overview.htm

2. 7 Segment Labels Image
   https://www.geeksforgeeks.org/maximum-number-that-can-be-display-on-seven-segment-display-using-n-segments/