Yet Another Alarm Clock

Done by: Beibarys Mukhammedyarov,
Baiakhmet Zhekeyev,
Abylay Tastanbek,
Maxim Zhabinets
and Zhanibek Darimbekov

1. Introduction

The purpose of our project was to design and implement an alarm clock prototype with additional functions. The experimental process involved programming the Arduino microcontrollers and working on setting up equipment, connecting wires and constructing a whole working alarm clock. While working on the project we gained a hands-on experience on Arduino programming, implemented and tested ideas and theoretical knowledge we obtained during Microcontrollers lectures and lab sessions.

2. Materials and Methods

1. Arduino Mega board
2. Speaker
3. Four 7segment digit LED displays
4. Two small breadboards
5. Three push buttons
6. Seven resistors: three 10k ohm for push buttons and four ~270 ohm for 7-segment digit LED displays
7. ~36 connection wires

Our project was based and implemented on Arduino Mega. The source code was written in Arduino IDE and tested by uploading to the microcontroller. Then we connected four 7-segment displays, showing time in minutes and seconds, or arithmetic expressions depending on the clock`s mode. The settings of the clock are changed by three buttons. One speaker is used to play a melody when timer is over. The displays and buttons were located in two separate breadboards. Finally, all electronic equipment were wrapped up in a small box to hide all wires and left only buttons, display and speaker for usability.

Design:

At first we were planning to use decoders to provide output for 7-segment digit displays. This combination would allow to significantly decrease the number of used pins on Arduino board (from 32 down to 16) and use smaller and cheaper Arduino Uno for this device. After the analysis this design was rejected for the following reasons:

1) Due to the size of decoders from lab sets and restrictions on their position on the breadboard (right in the middle) four decoders should be placed on two additional small breadboards or on one big breadboard.

2) 7-segment digit LED displays that we were given use inverse logic. This means that we would need additional inverter for each output from a decoder. This would require us to add 28 inverters to our device and double the amount of connection wires.

As this design would lead to construction of a device of enormous size and as we simply did not have so many wires as well as a room for them it was decided to use output from Arduino Mega pins directly for the input of four seven segment displays.

Code:

As the hardware system implementation is fairly simple, the complexity of the device is lies within the code. Microcontroller on Arduino Mega does not support multiple threads. As any timer our device should constantly check the remaining time and update the digits on seven segment displays accordingly. This does not allow us to use any long operations in the code including large delay()(s), playing long sounds and using separate interrupt functions outside of the main loop. The loop iteration time should be as small as possible in order to update digits precisely and give a fast response to the pressing of the buttons. The countdown of time is implemented using millis() function that returns the time in milliseconds since the last reset of the Arduino board. Remaining time is calculated by taking difference between current time and countdown start time.

```
if(mode == 2){
  time1 = currentTime - startTime;
  rtime = alltime - time1/1000;
  minutes = rtime/60;
  seconds = rtime%60;
  d4=seconds%10;    //d4 – digit N4
  d3=seconds/10;
  d1=minutes/10;
  d2=minutes%10;
}
```

Figure 1: Section of code for calculating the remaining time.

For implementation of the sound signal at the end of timer’s countdown we could not use long predetermined sound because while paying sound signal the device should be responsive for buttons presses. To bypass this limitation we use tone() function only once in the loop and change its parameters afterwards. This way we get different sound for each loop iteration.

```
if(mode == 3 || mode == 4 || mode == 5){
  t+=35;
  if(t>2000){
    t = 100;
  }
  tone(5, t, 2000);
 }
```

Figure 2: Section of code for producing sound.

Setting a pattern on a seven segment LED to the pattern of one of 10 digits requires 7 digital writes. For making the code shorter and more readable a new statement was defined for every digit (40 defines in total).

```
#define sd1t1 \
digitalWrite(25,1);\
digitalWrite(28,1);\
digitalWrite(22,0);\
digitalWrite(26,0);\
digitalWrite(23,0);\
digitalWrite(24,0);\
digitalWrite(27,0)
```

Figure 3: Section of code with define statement

3. Results

The finished device turned out to be quite large but not too heavy. The device is fully functional and works without failures. Due to the debouncing of buttons and overall time of loop iteration buttons are not able to register extremely fast presses, however they still able to register fast presses and should not irritate the user.

When time is over, the speaker starts playing the melody until the user enters correct answer to the arithmetic expression shown on the displays. Because we had only four 7-segment displays, they were designed to show minutes and seconds only. Initially, we wanted our clock to show real time, but for simplicity of our code we chose to design a timer, which sets a countdown in minutes and seconds. The melody of the alarm was hardcoded and is the only one. The arithmetic problems are displayed in three displays, so they are very simple: addition and subtraction of two one-digit numbers. Subtraction operator is displayed as a “-“ in the central segment of second display and addition operator is shown as “P” on the same display. The answer is entered in the fourth display. After the user picks the right answer and enters the second button, the alarm stops.

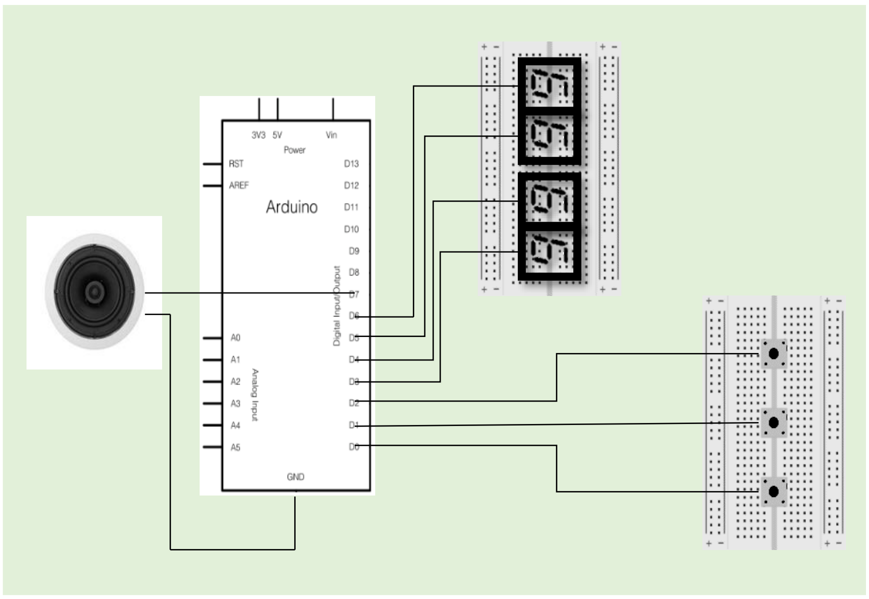
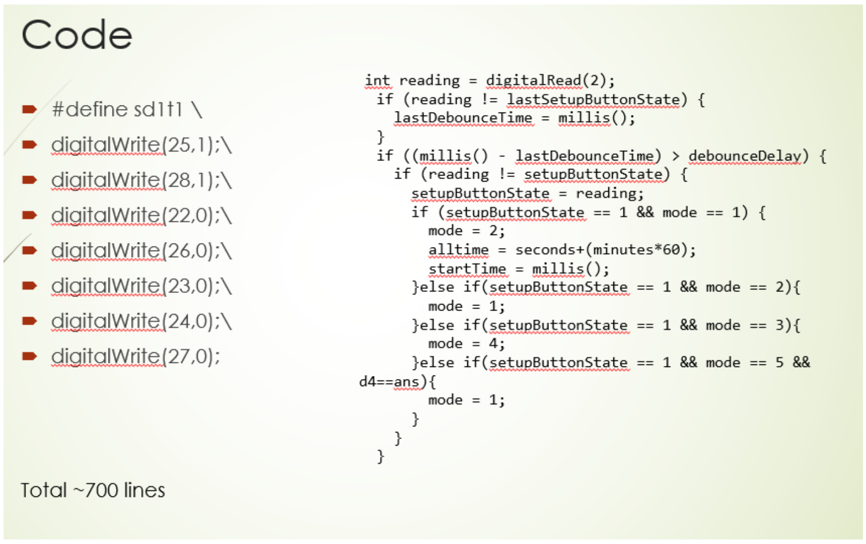


Figure 3. Project Schema: Arduino board, speaker, clock display and buttons on a separate breadboards.



Code

- #define sd1t1 \
- digitalWrite(25,1);\
- digitalWrite(28,1);\
- digitalWrite(22,0);\
- digitalWrite(26,0);\
- digitalWrite(23,0);\
- digitalWrite(24,0);\
- digitalWrite(27,0);

```
 int reading = digitalRead(2);
  if (reading != lastSetupButtonState) {
    lastDebounceTime = millis();
  }
  if ((millis() - lastDebounceTime) > debounceDelay) {
    if (reading != setupButtonState) {
      setupButtonState = reading;
      if (setupButtonState == 1 && mode == 1) {
        mode = 2;
        alltime = seconds+(minutes*60);
        startTime = millis();
      }else if(setupButtonState == 1 && mode == 2){
        mode = 1;
      }else if(setupButtonState == 1 && mode == 3){
        mode = 4;
      }else if(setupButtonState == 1 && mode == 5 &&
d4==ans){
        mode = 1;
      }
    }
  }
```

Total ~700 lines

Figure 4. Code snippet: define statement fragment at the first line, setup statements below and a piece of code lines from the main body.

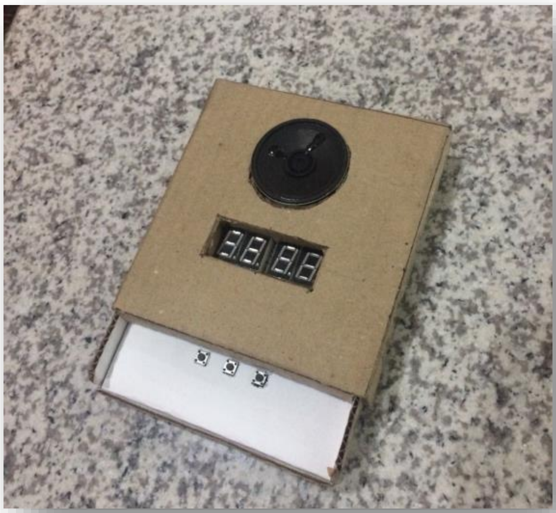


Figure 5. The final working version of the clock.

4. Conclusion

The final design and logical performance of the clock was tested and showed results as it as expected at the beginning of the project. The digits on the time display setting buttons work the same way as popular clock or timer models. The main difficulty of this project was to construct stably working device while bypassing the single-threadedness of the microcontroller.

Student Contributions:

As a one team every member equally contributed to the project but the inner tasks were fairly divided among us depending on the abilities of the individuals , Maxim wrote code, Beibarys and Abylai worked on design and performance of the project, Zhanibek and Baiakhmet wrote the report.