TOSHKENT SHAHRIDAGI INHA UNIVERSITETI
INHA UNIVERSITY IN TASHKENT

Embedded Software & Design (SOC3050)

Final Project Report

Digital Alarm Clock by Team IMPERIUM

Team members (Name, ID, Group/Section):

1. Oybek Amonov	U1610176	CSE16-2/001
2. Rakhmatjon Khasanov	U1610183	CSE16-2/001
3. Bokhodir Urinboev(leader)	U1610249	CSE16-1/001

Table of Contents

Introduction

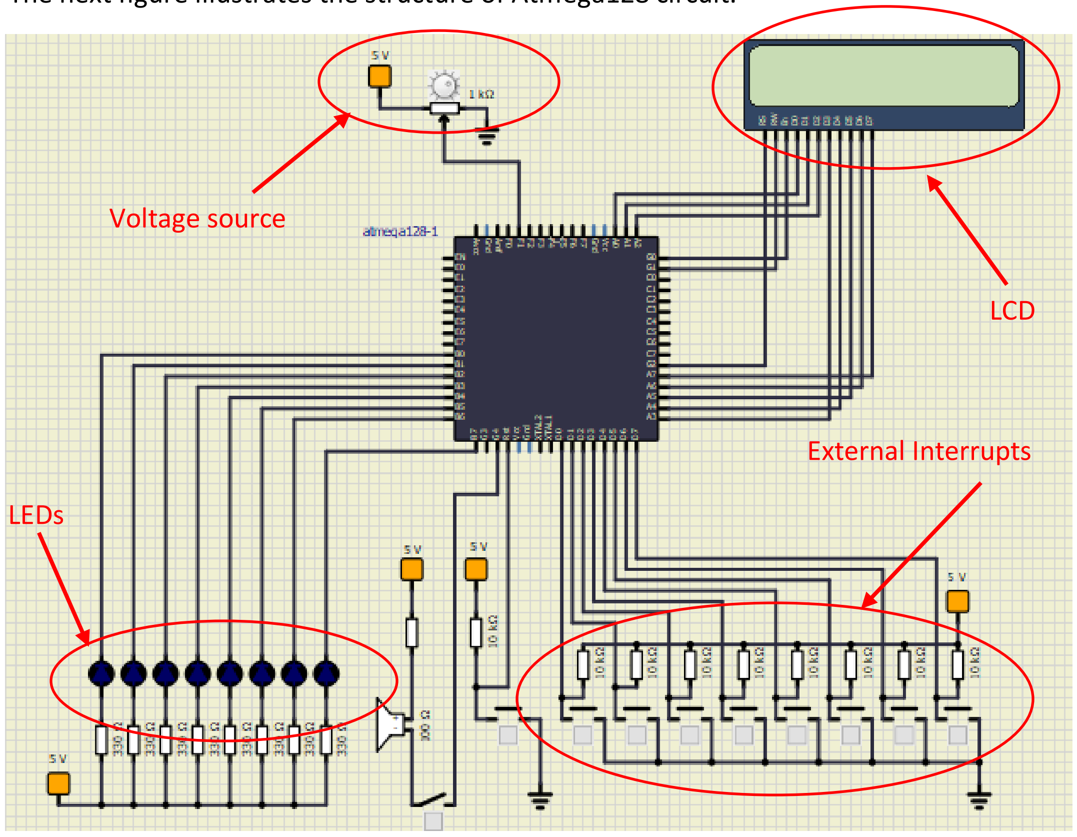
This paper is a term project report from Embedded System & Design (SOC3050) course offered by Inha University in Tashkent (IUT) during Spring semester, 2020. The main purpose of the project is to develop a digital clock using Atmega128 microcontroller. The project is implemented by Team Imperium.

Abstract

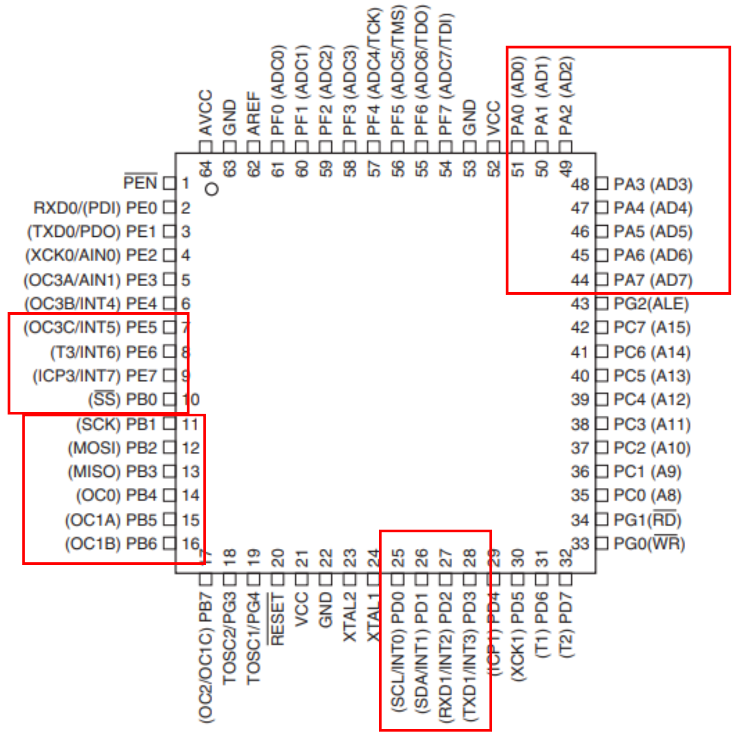
A digital clock project is developed using C language, and it has functionalities such as setting current time information, setting alarm time, displaying time mode, alarm time show mode, and stopwatch mode. All the mentioned functionalities were implemented using external interrupts (INT0~INT3) thorough Port D. And, timer is implemented using CTC mode in atmega128 microcontroller by use of 8-bit counter.

Working environment

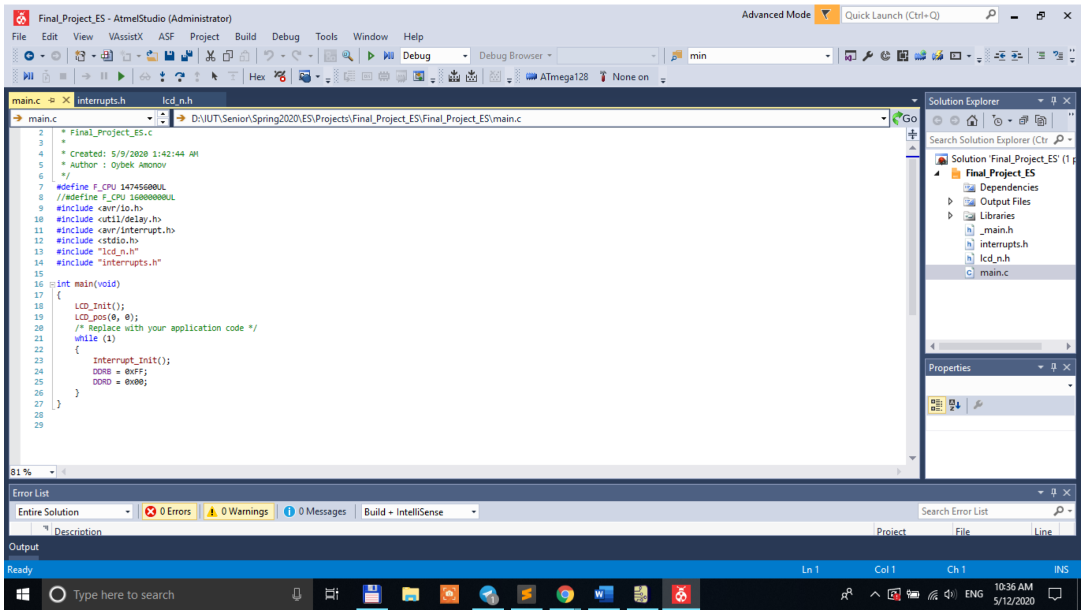
Project source code files were developed using Atmel 7.0 IDE environment. Atmel has a support for developing embedded applications in assembly and C/C++ languages. It helps to easily convert source files into executable object files (machine code, to .hex files in our case). The .hex files were loaded on SimulIDE (simulator), run, and tested for errors or bugs. There were made some small changes to the Atmega128 circuit in order to provide more convenience for the user, since the circuit is quite difficult to understand with different port connections, resistors, LCD display, LEDs, and voltage source. The next figure illustrates the structure of Atmega128 circuit.



The next figure shows the pin architecture of Atmega128 microcontroller (e.g. Port pins, power source pins, and etc.). As you can see, pin numbers 10 to 17 are assigned to Port B, which is connected to LEDs of the system. And, pins 44 to 51 are assigned to Port A in reverse order, which is LCD connection. Lastly, pins 25 to 28 are assigned to Port D for External Interrupt handling for INT0 to INT3. The rest interrupts, INT4 to INT7, are connected Port E thorough pins 6 to 9 respectively.



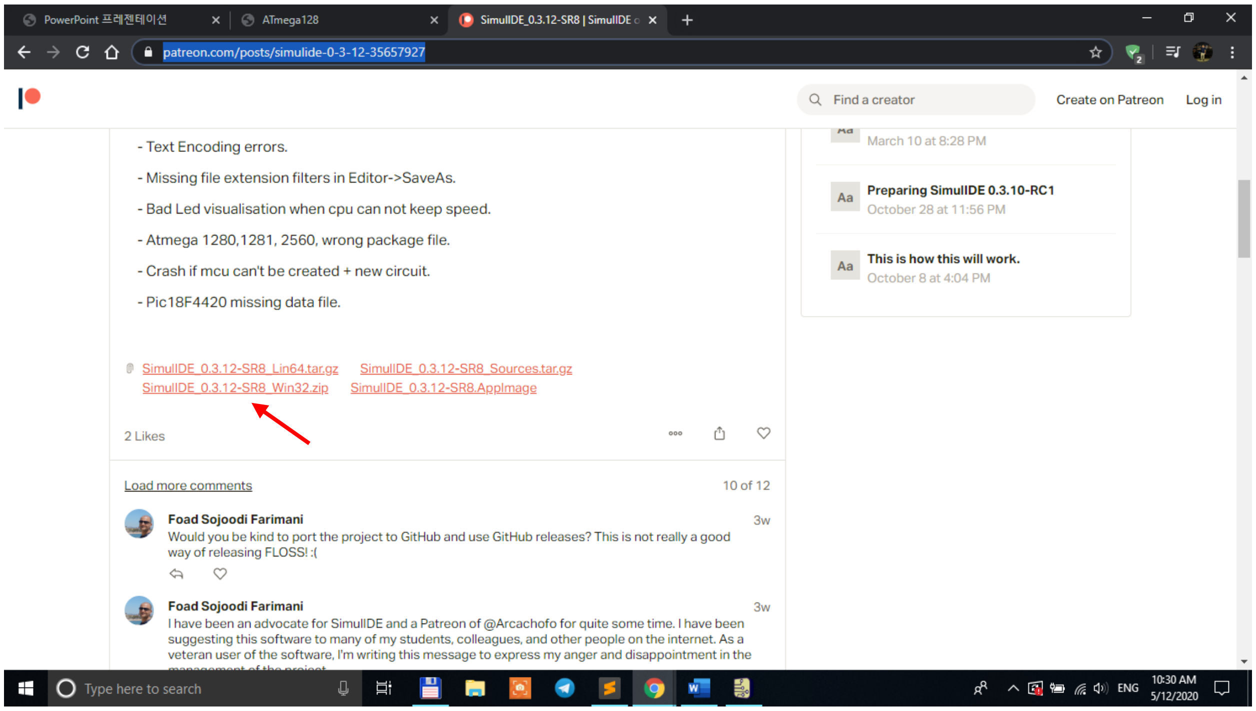
We have coded the whole clock using Atmel Studio



```
 * Final_Project_ES.c
 *
 * Created: 5/9/2020 1:42:44 AM
 * Author : Oybek Amonov
 */
#define F_CPU 14745600UL
//#define F_CPU 16000000UL
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#include <stdio.h>
#include "lcd_n.h"
#include "interrupts.h"

int main(void)
{
    LCD_Init();
    LCD_pos(0, 0);
    /* Replace with your application code */
    while (1)
    {
        Interrupt_Init();
        DDRB = 0xFF;
        DDRD = 0x00;
    }
}
```

User Manual

So, know let's see how to use our digital clock. First, firstly, download the SimulIDE from the given link and choose the appropriate zip file according to your OS system.


patreon.com/posts/simulide-0-3-12-35657927

- Text Encoding errors.
- Missing file extension filters in Editor->SaveAs.
- Bad Led visualisation when cpu can not keep speed.
- Atmega 1280,1281, 2560, wrong package file.
- Crash if mcu can't be created + new circuit.
- Pic18F4420 missing data file.

SimulIDE_0.3.12-SR8_Lin64.tar.gz SimulIDE_0.3.12-SR8_Sources.tar.gz
SimulIDE_0.3.12-SR8_Win32.zip SimulIDE_0.3.12-SR8.AppImage

2 Likes

Load more comments 10 of 12

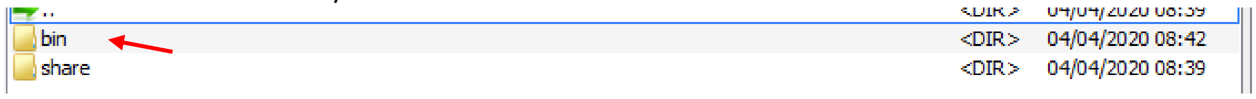
Foad Sojoodi Farimani 3w
Would you be kind to port the project to GitHub and use GitHub releases? This is not really a good way of releasing FLOSS! :(

Foad Sojoodi Farimani 3w
I have been an advocate for SimulIDE and a Patreon of @Arcachofo for quite some time. I have been suggesting this software to many of my students, colleagues, and other people on the internet. As a veteran user of the software, I'm writing this message to express my anger and disappointment in the

Preparing SimulIDE 0.3.10-RC1
October 28 at 11:56 PM

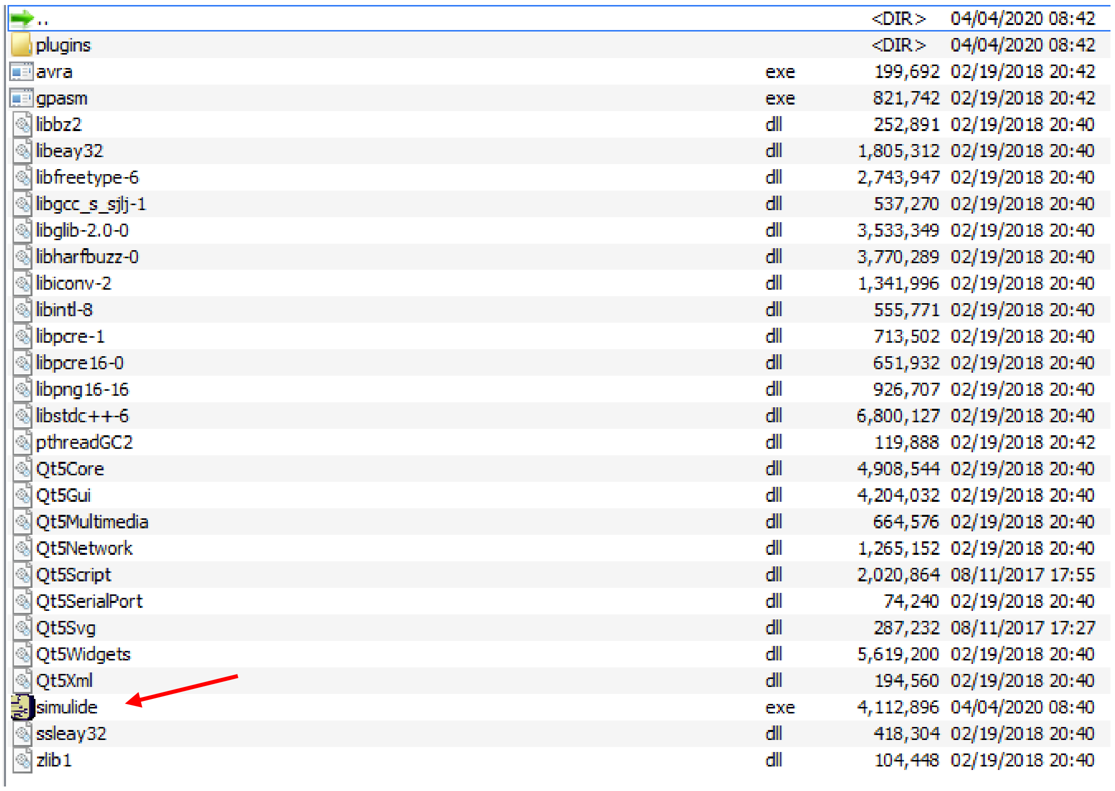
This is how this will work.
October 8 at 4:04 PM

Next step is to unzip the downloaded file to any directory you want. After unzipping, you will see two directories inside the root directory if you are using Windows OS, you might have different files for other OS. Go into the bin directory.



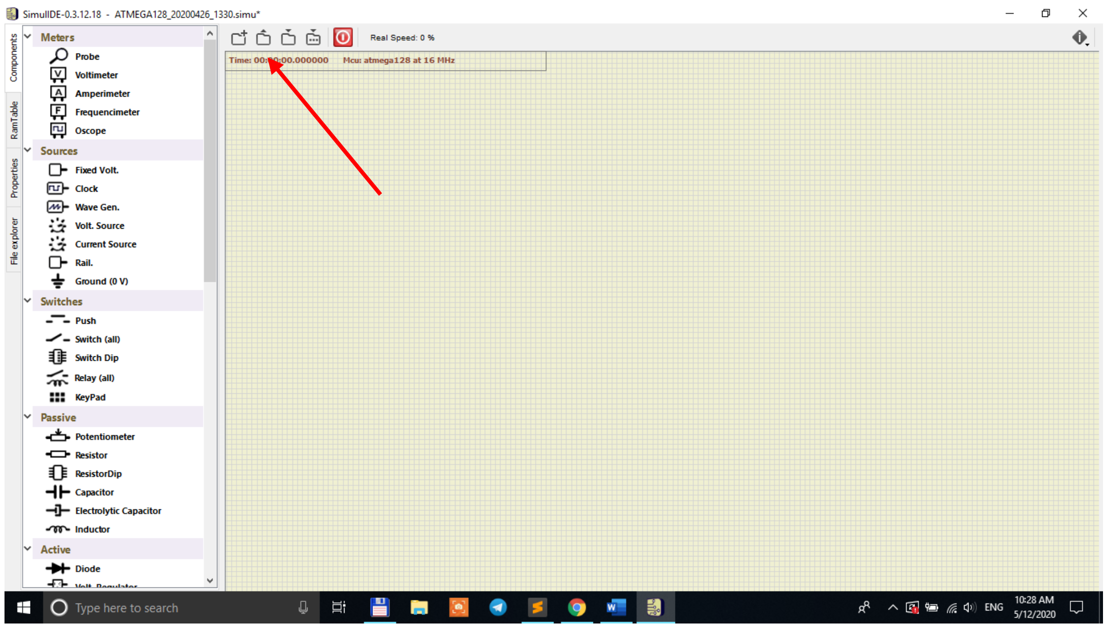
bin	<DIR>	04/04/2020 08:42
share	<DIR>	04/04/2020 08:39

Now, you can see the simulide.exe executable file, which is the actual application where we will be using to run and test our system. Open the simulide.exe file.

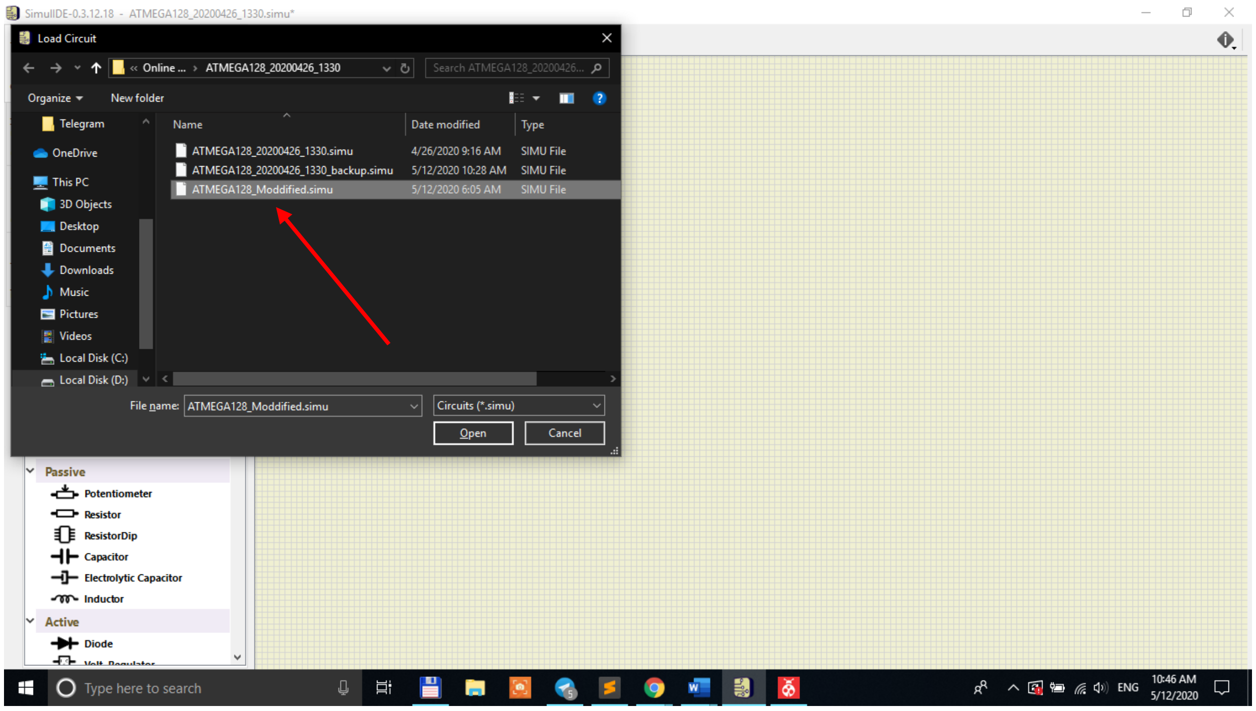


..		<DIR>	04/04/2020 08:42
plugins		<DIR>	04/04/2020 08:42
avra	exe	199,692	02/19/2018 20:42
gpasm	exe	821,742	02/19/2018 20:42
libbz2	dll	252,891	02/19/2018 20:40
libeay32	dll	1,805,312	02/19/2018 20:40
libfreetype-6	dll	2,743,947	02/19/2018 20:40
libgcc_s_sjlj-1	dll	537,270	02/19/2018 20:40
libglib-2.0-0	dll	3,533,349	02/19/2018 20:40
libharfbuzz-0	dll	3,770,289	02/19/2018 20:40
libiconv-2	dll	1,341,996	02/19/2018 20:40
libintl-8	dll	555,771	02/19/2018 20:40
libpcre-1	dll	713,502	02/19/2018 20:40
libpcre16-0	dll	651,932	02/19/2018 20:40
libpng16-16	dll	926,707	02/19/2018 20:40
libstdc++-6	dll	6,800,127	02/19/2018 20:40
pthreadGC2	dll	119,888	02/19/2018 20:42
Qt5Core	dll	4,908,544	02/19/2018 20:40
Qt5Gui	dll	4,204,032	02/19/2018 20:40
Qt5Multimedia	dll	664,576	02/19/2018 20:40
Qt5Network	dll	1,265,152	02/19/2018 20:40
Qt5Script	dll	2,020,864	08/11/2017 17:55
Qt5SerialPort	dll	74,240	02/19/2018 20:40
Qt5Svg	dll	287,232	08/11/2017 17:27
Qt5Widgets	dll	5,619,200	02/19/2018 20:40
Qt5Xml	dll	194,560	02/19/2018 20:40
simulide	exe	4,112,896	04/04/2020 08:40
ssleay32	dll	418,304	02/19/2018 20:40
zlib1	dll	104,448	02/19/2018 20:40

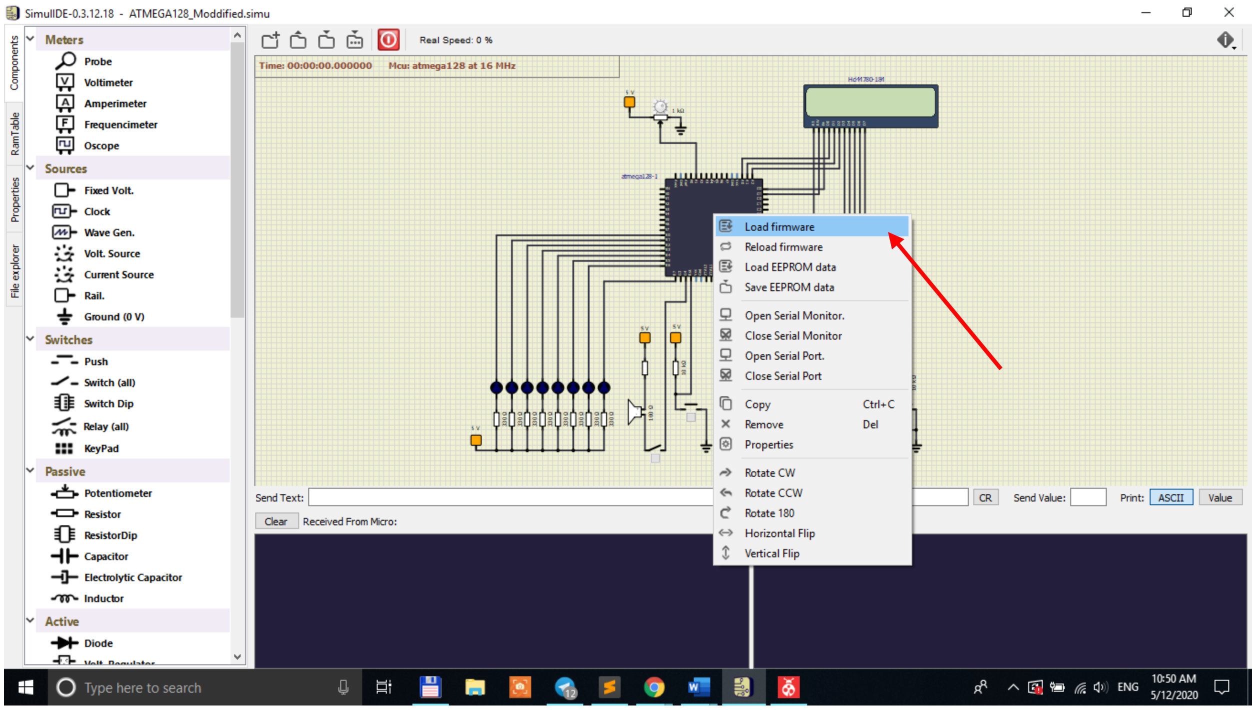
You will see the following environment to work with for the rest of the project, along with Atmel development environment too. You need to load an atmega128 circuit to simulate the project. You can download the circuit using the link. And, push the open circuit button on top panel.



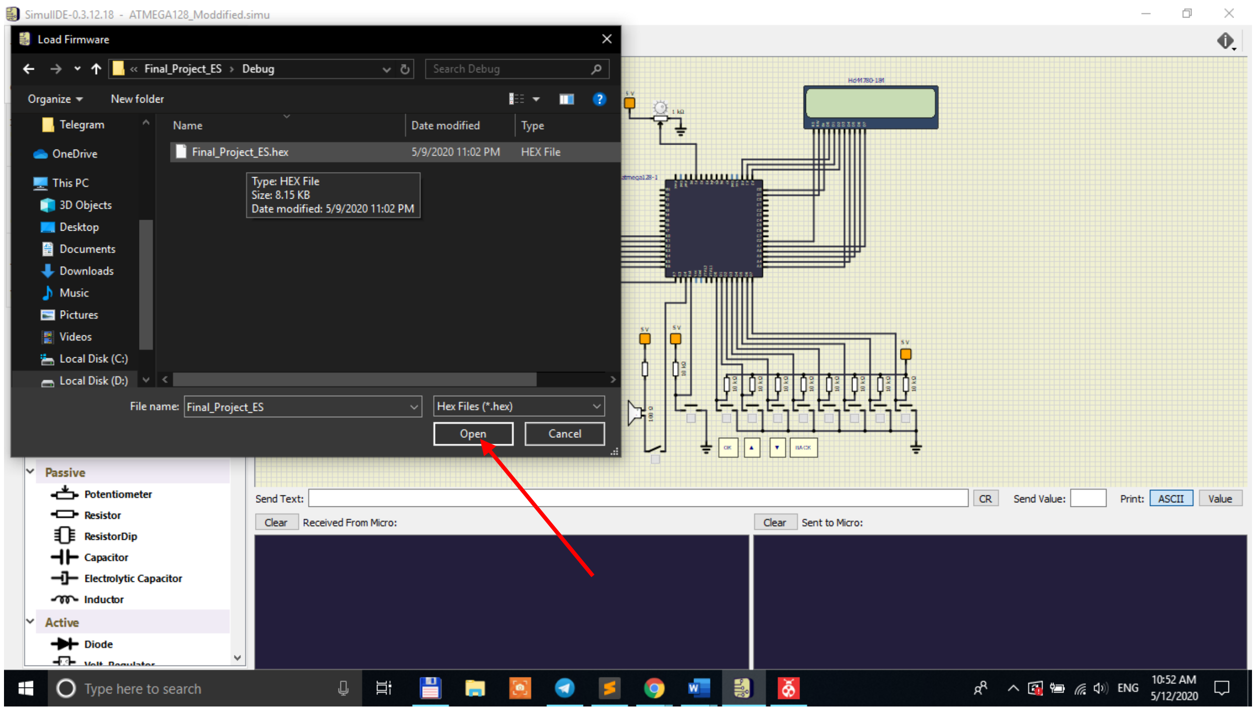
Go to the directory where you download the circuit and open it. Your downloaded file will have a name ATMEGA128_20200426_1330.simu, but we have a bit changed the circuit that you can use it in a more convenient way. You can download the modified circuit by this link.


SimulIDE-0.3.12.18 - ATMEGA128_20200426_1330.simu*
Load Circuit
« Online ... › ATMEGA128_20200426_1330
Search ATMEGA128_20200426...
Organize
New folder
Telegram
OneDrive
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Local Disk (C:)
Local Disk (D:)
Name
Date modified
Type
ATMEGA128_20200426_1330.simu 4/26/2020 9:16 AM SIMU File
ATMEGA128_20200426_1330_backup.simu 5/12/2020 10:28 AM SIMU File
ATMEGA128_Moddified.simu 5/12/2020 6:05 AM SIMU File
File name: ATMEGA128_Moddified.simu
Circuits (*.simu)
Open
Cancel
Passive
Potentiometer
Resistor
ResistorDip
Capacitor
Electrolytic Capacitor
Inductor
Active
Diode

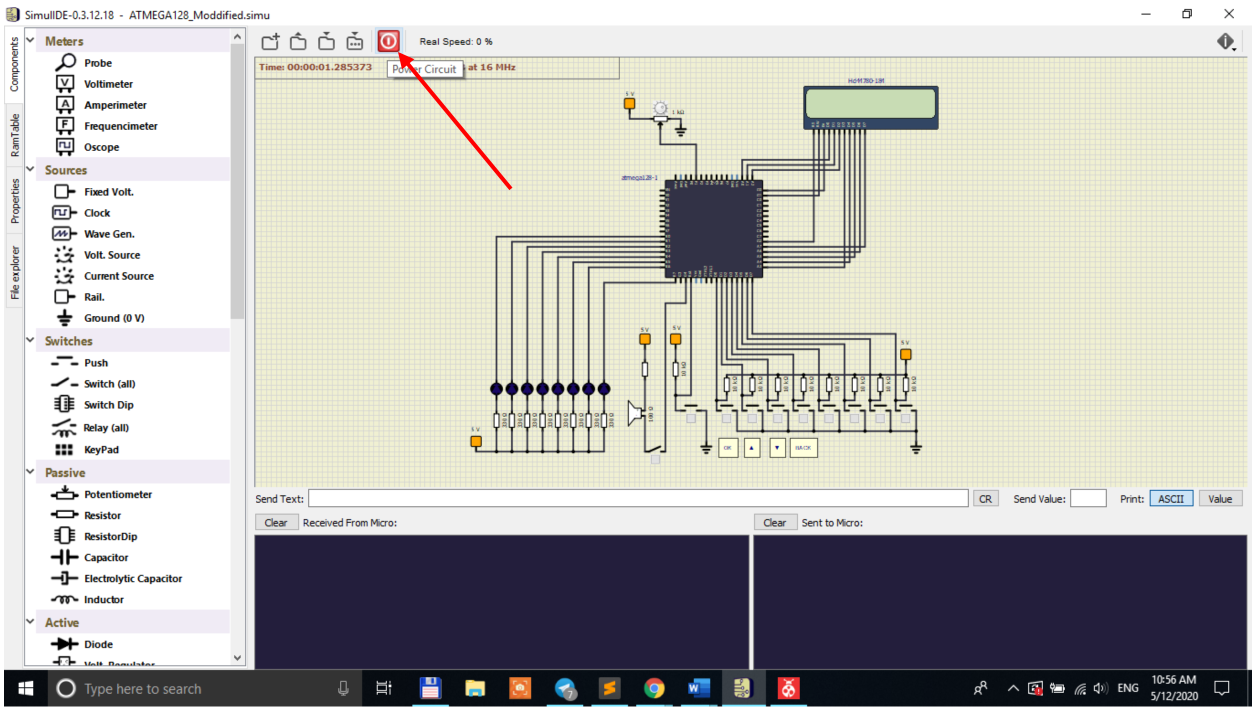
After loading the circuit, you are ready to load the machine code to run by microcontroller. Right-click on the microcontroller, and choose ‘Load firmware’ option.



Go to the project directory, and find the .hex file generated by the Atmel Studio. Load it to the circuit.



Lastly, you can press the ON button on top panel, so microcontroller starts operating. Thus, the project is successfully executed.



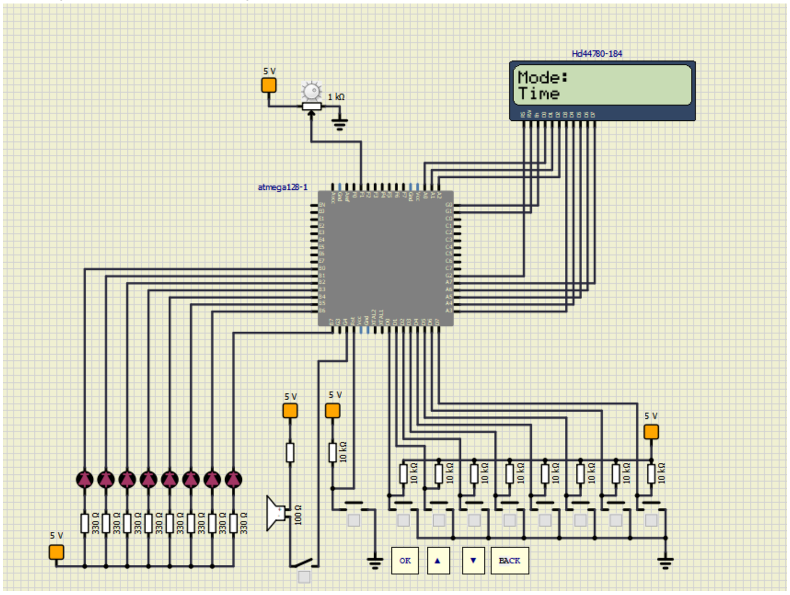
As you can see from the below screenshot, digital clock is successfully executed, and Menu is displayed. Now, you can use the following navigation rules to change the mode, and select it.

(1) OK(INT0) – is used to select chosen mode, to set(confirm) the parameters (year, month, day, hour, minutes, and seconds), to start/pause/reset stopwatch.

(2) Up(INT1) – is used to change selected mode, change parameter value (increase it).

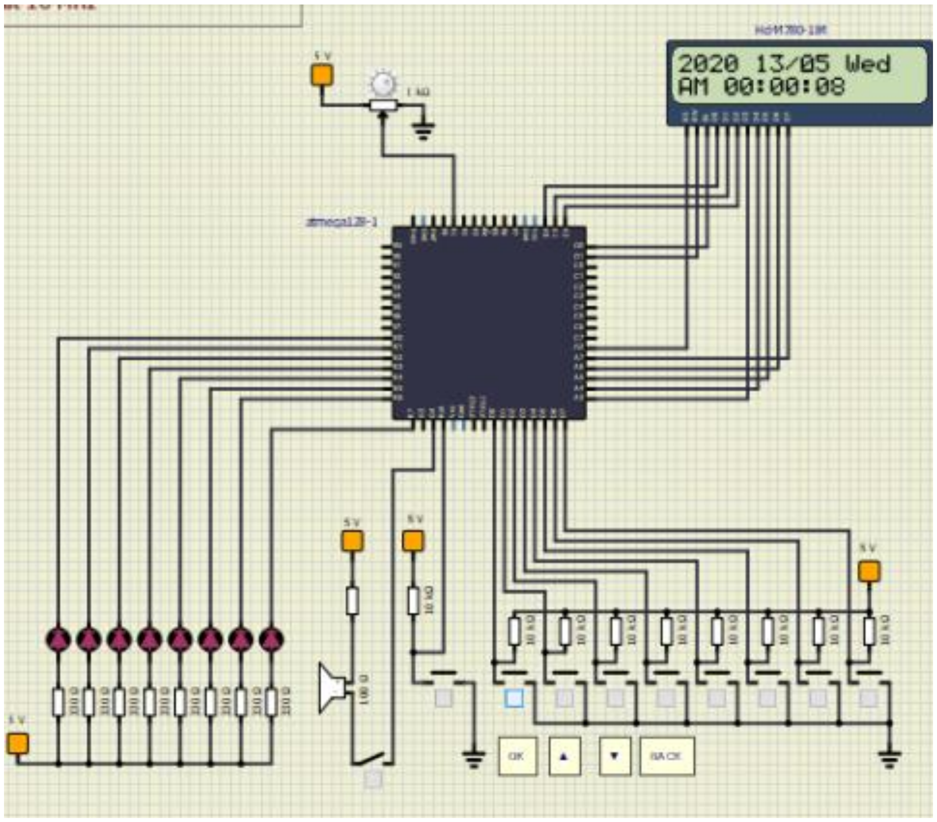
(3) Down(INT2) – is used to change selected mode, change parameter value (decrease it)

(4) Back(INT3) – is used to go back to previous mode

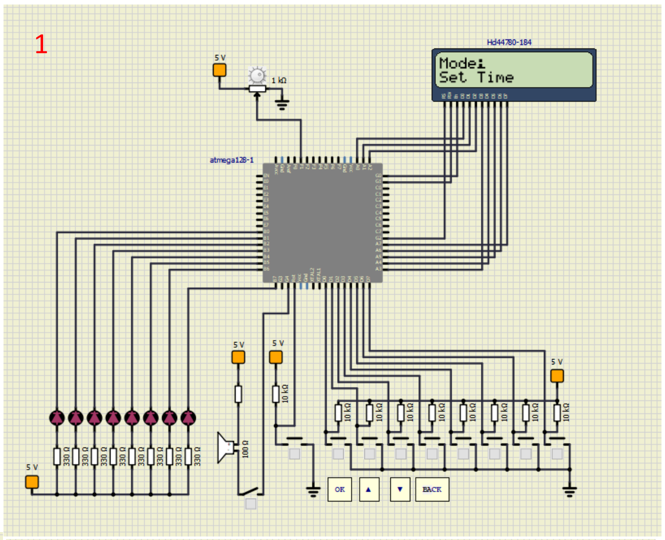


You can see that 1st mode in our clock mode is a Time (Display) Mode, where current time will be displayed. Now, you can use down button to change to which mode you want to go. First, you need to set the current time, though. Otherwise, default time given will be displayed. (Remember, the buttons are just labels, they do not give the interrupt to the microcontroller. You need to push INTn triggers, not the button itself, to give input)

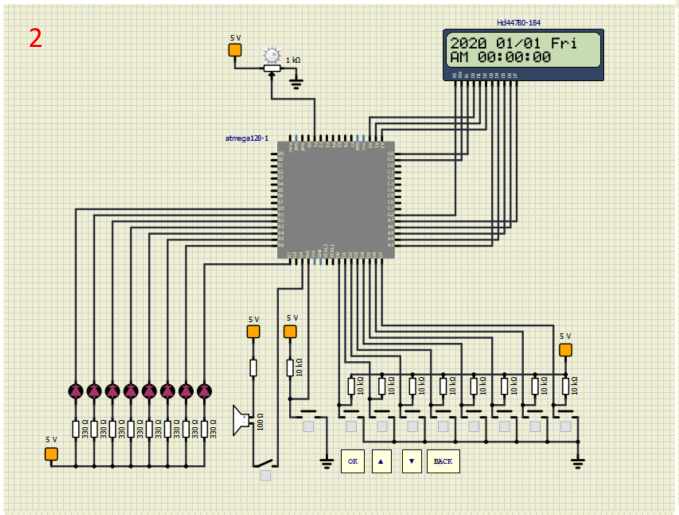
Let's assume you did not set the time manually, so the next picture shows the default time for our clock.



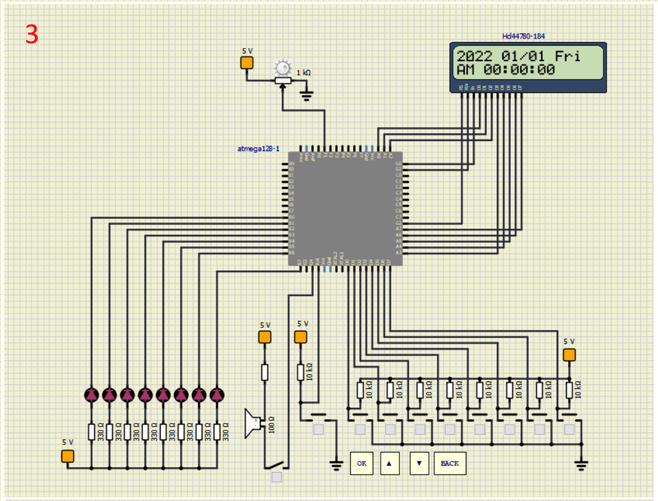
Press Back (INT3) to go back to the main menu, and go to the Set time Mode to set the current time information. You need to first input the year, then month, and day respectively. To select the chosen value, you should press OK (INT0) button. You do not have to set the day of the week, because we have implemented a procedure to automatically calculate the day. So, just set hour, minutes, and seconds lastly. It is AM or PM our code will detect automatically too. At last, once again press OK button to set the time, and system automatically goes back to the Main Menu.

1


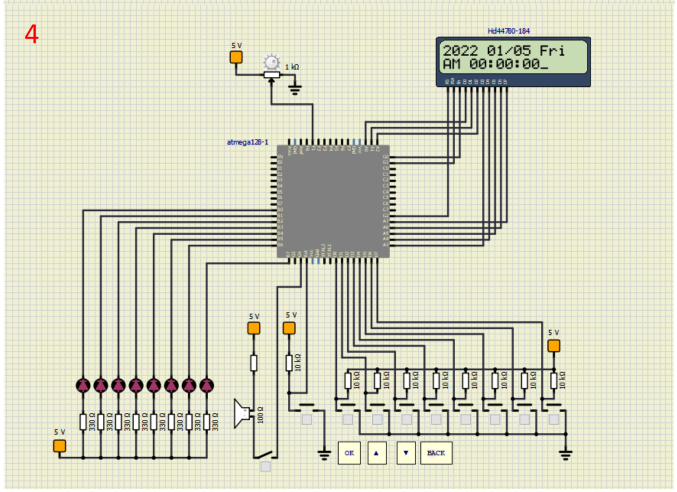
```
Mode:
Set Time
```

2


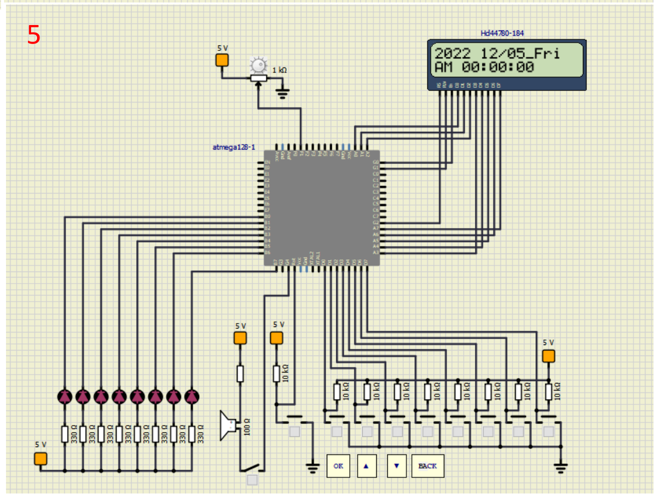
```
2020 01/01 Fri
AM 00:00:00
```

3


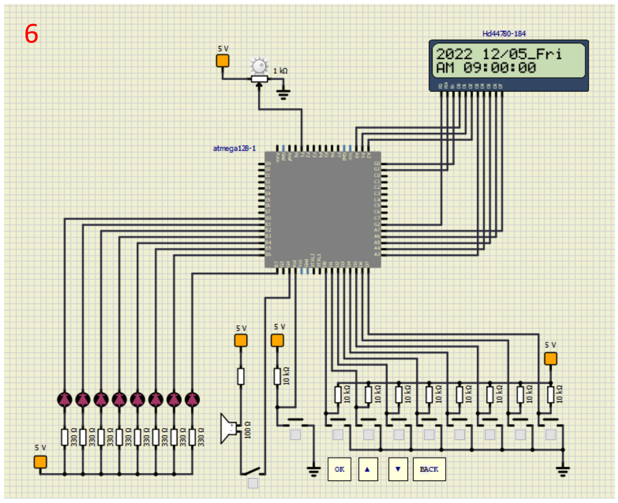
```
2022 01/01 Fri
AM 00:00:00
```

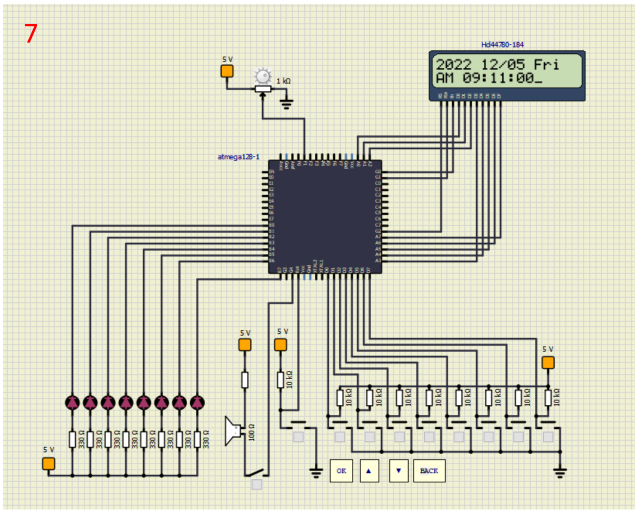
4


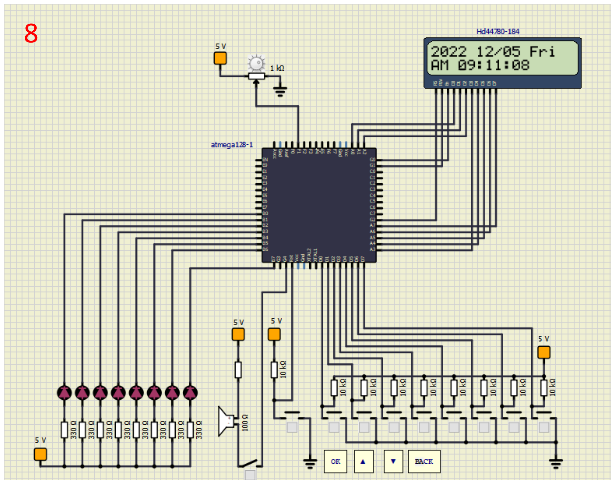
```
2022 01/05 Fri
AM 00:00:00_
```

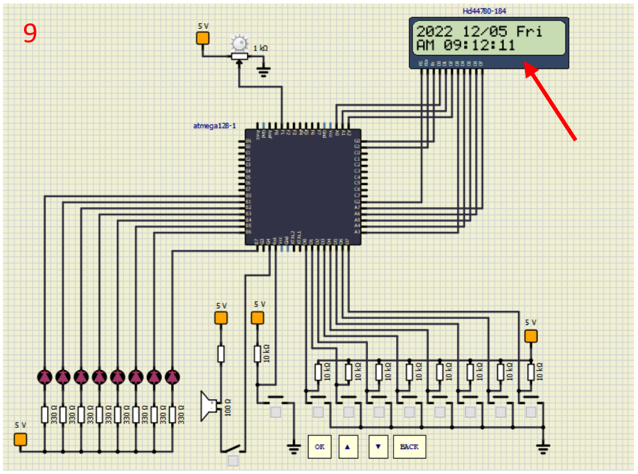
5


```
2022 12/05_Fri
AM 00:00:00
```

6


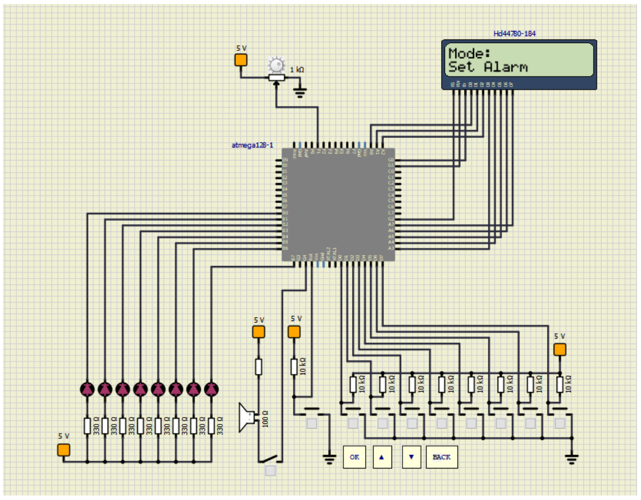
7


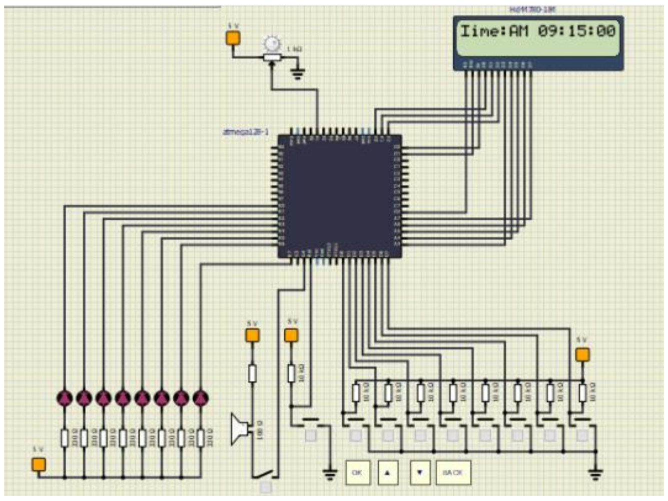
8


9


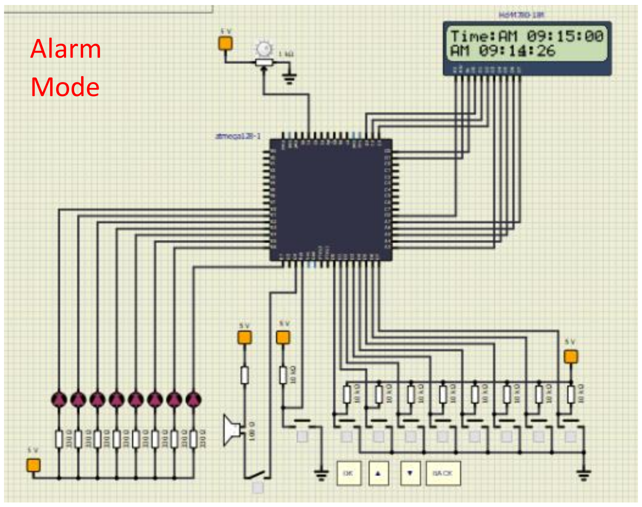
So, now we have set the time. By pressing the OK button once again, we will be directed to the main menu. Let's check the if the timer is working, by choosing Time mode and see the current time after some time. As you can see from picture number 9, we waited some time around a minute.

Now, it is time set our alarm time, let's go to the main menu(picture on the left) and get to the Set Alarm mode. When we are in set alarm mode(picture below), we just need to set hour, minutes, and seconds. Again, the assignment of AM or PM is handled by our procedure internally. You do not have to set it. We set the alarm to 09:15 a.m.So, now we just need to wait untill it is the time. When the alarm time is happening, the LEDs blink five times. You can watch the runtime process of the project on YouTube to see the effect of blinking.

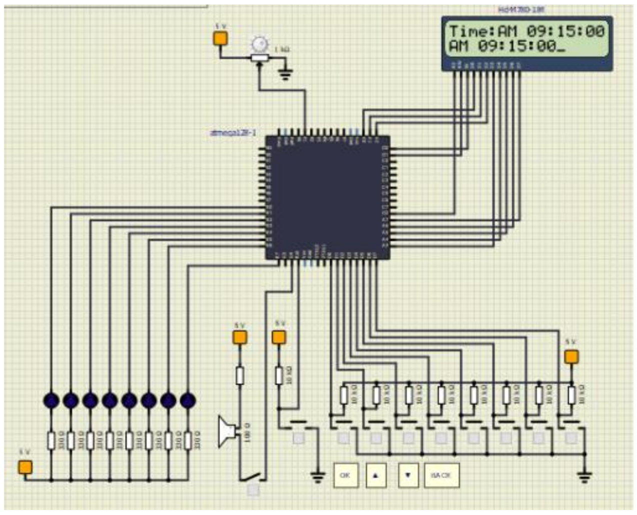




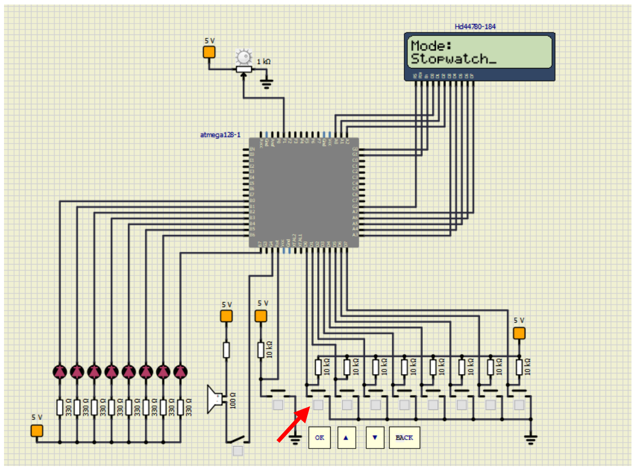
Alarm Mode

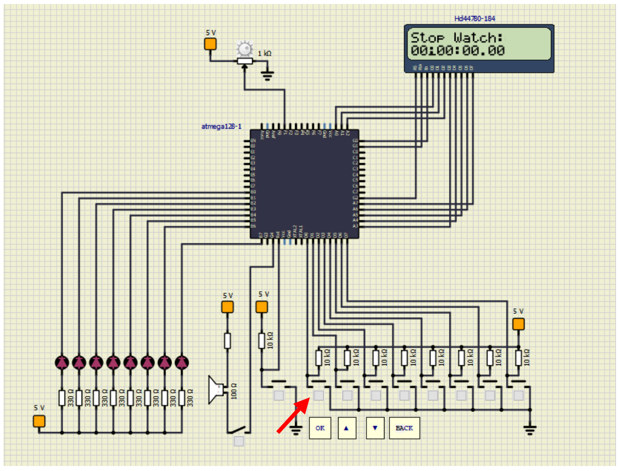


As you can see from the picture ***Alarm Mode***, it is 34 seconds left till the alarm. When the clock gets to 09:15 a.m., alarm will be activated and LEDs blink 5 times(picture on the left). You see from the picture on the left side, all LEDs were off. You should either run the .hex code on simulator or watch our simulation video on YouTube to make sure that they are blinking.

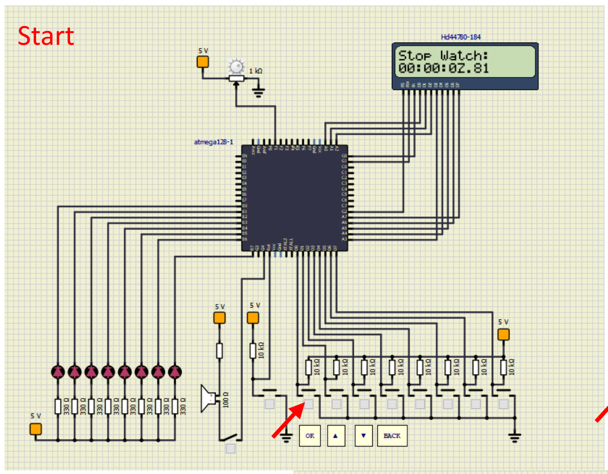


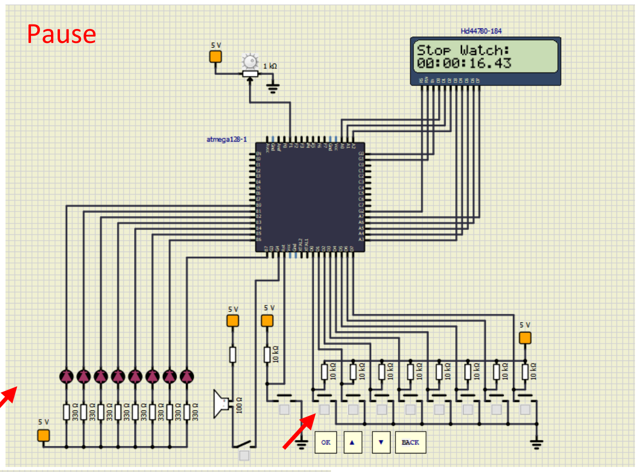
The last required mode of the digital clock must be the stopwatch mode with start/pause/stop functionalities. Let's go to the Stopwatch mode in Main Menu, and select it.

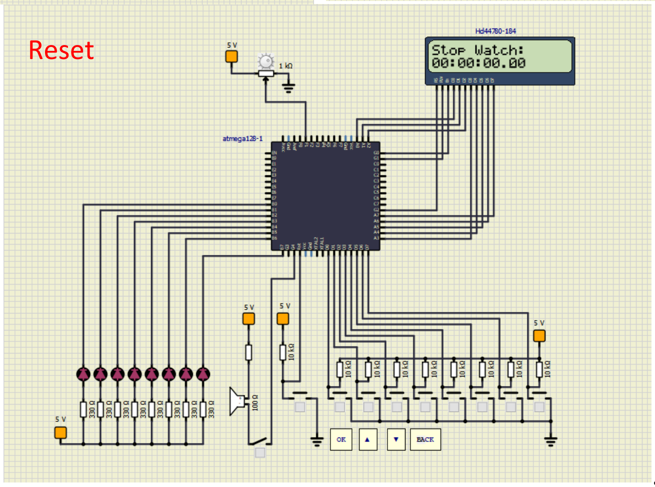




You can now start the stopwatch by pushing the OK button, and pause the stopwatch any time you want, reset it too







.

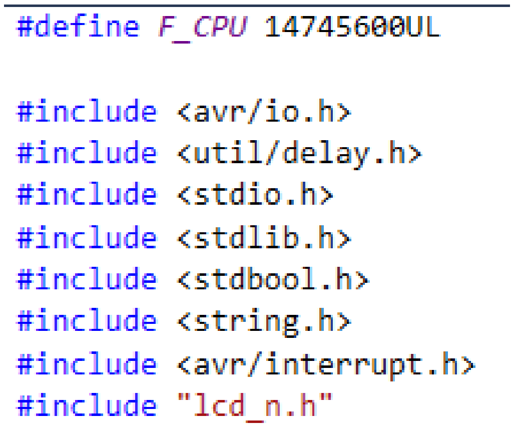
Algorithm explanation

Architecture design

Overall, the system is based on Interrupt Handling in AVR microcontroller. We tried to implement all output manipulations in main function, and ISR functions, Port initialization, Interrupt initialization are handled using other functions. Predefined LCD manipulation functions were imported from lcd_n.h header file. When the program starts, Port initialization is done first, where we assign PortA as output port (LCD). Then, LCD is initialized and cleared. Timer is also initialized, and we choose to use CTC mode in AVR Timer (8-bit timer). We used 1024 prescaling to work with each 10 ms (by assigning OCR0 = 155, meaning we get interrupt on each 156-clock cycle of AVR timer). We have designed 5 general modes (Time, Alarm, Stopwatch, Set time, and Set alarm), and modes are handled by using mode flag variables. System body is inside infinite while loop, allowing us to use modes infinite time.

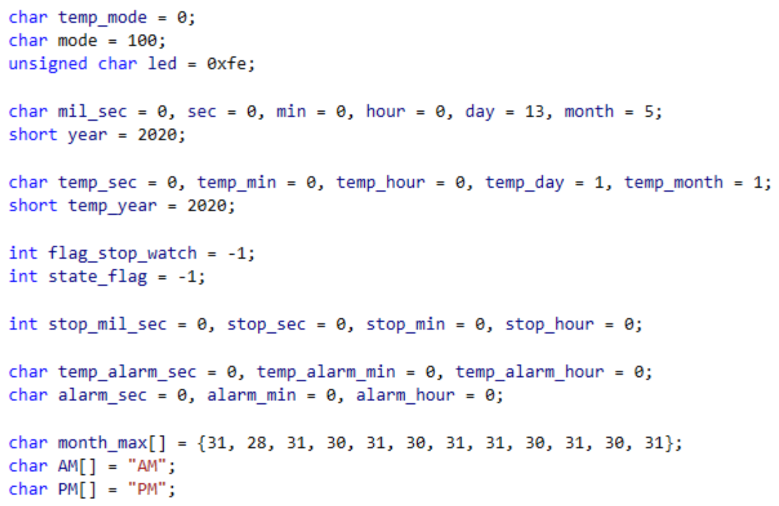
Code discussion

We need the following libraries included to our program and define frequency of cycle clock of atmega128 (however, we have used F_CPU = 16 MHz to match SimulIDE clock frequency).

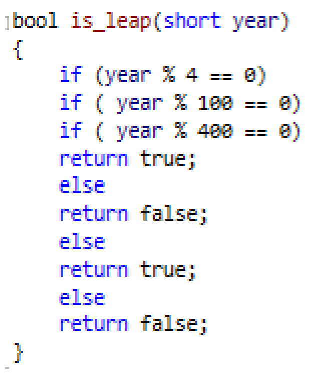


```c
#define F_CPU 14745600UL

#include <avr/io.h>
#include <util/delay.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <string.h>
#include <avr/interrupt.h>
#include "lcd_n.h"
```

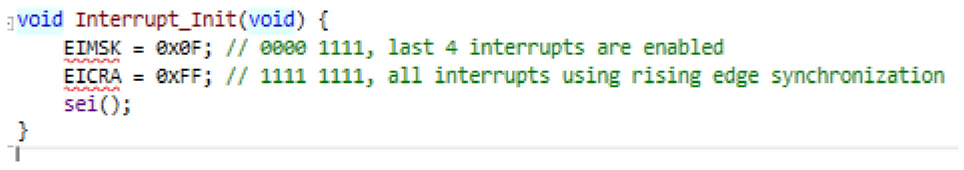
The following figure is the global variables list we need to handle the interrupts, modes, and time attributes. First variable is used to temporarily store the mode selected, when OK button is pressed then mode is assigned temp_mode, which means we go to the selected mode. Led variable is given as 0xfe(11111110) meaning only 1 led is on at first, and it is turning on/off sequentially. We have given default values for millisecond, second, minute, hour, day, month, and year variables. The same number of temporary variables used to set time. Integer flag for stop watch is used to handle start/pause/reset modes. And state flag is used to control the setting current time, alarm time variables one by one. Separate milliseconds, seconds, minutes, and hours variables are defined, and the same for alarm time. Month days are handled integer array, and time of the days using char arrays. Day of the week is handled using separate function, which will be discussed later.



```c
char temp_mode = 0;
char mode = 100;
unsigned char led = 0xfe;

char mil_sec = 0, sec = 0, min = 0, hour = 0, day = 13, month = 5;
short year = 2020;

char temp_sec = 0, temp_min = 0, temp_hour = 0, temp_day = 1, temp_month = 1;
short temp_year = 2020;

int flag_stop_watch = -1;
int state_flag = -1;

int stop_mil_sec = 0, stop_sec = 0, stop_min = 0, stop_hour = 0;

char temp_alarm_sec = 0, temp_alarm_min = 0, temp_alarm_hour = 0;
char alarm_sec = 0, alarm_min = 0, alarm_hour = 0;

char month_max[] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
char AM[] = "AM";
char PM[] = "PM";
```



```
bool is_leap(short year)
{
    if (year % 4 == 0)
    if ( year % 100 == 0)
    if ( year % 400 == 0)
    return true;
    else
    return false;
    else
    return true;
    else
    return false;
}
```



```
void Interrupt_Init(void) {
    EIMSK = 0x0F; // 0000 1111, last 4 interrupts are enabled
    EICRA = 0xFF; // 1111 1111, all interrupts using rising edge synchronization
    sei();
}
```

Since our clock automatically handles month days, we implemented is_leap function to handle 366 days per a year situation as in screenshot on the top side. We have used 4 interrupts to handle all navigation in our program, and we used Interrupt_Init() function to set INT0~INT3 interrupts using rising edge synchronization as top figure shows. Sei() is built-in function to set all system interrupts enabled.

```
char *get_weekday(short year, char day, char month){ // weekday calculation function
        int days = (year-1) * 365+day;
        if(month > 1 && is_leap(year)){
                month_max[1] = 29;
        }else  month_max[1] = 28;
        while(1)      {
                if(year >= 400){
                        year -= 400;
                        days += 97;
                }else if(year >= 100){
                        year -= 100;
                        days += 24;
                }else  {
                        days += (int)((year)/4);
                        break;
                }}
        for(int i = 0; i < month-1; i++)   days += month_max[i];
        switch(days%7){
                case 0:       return "Sun       ";        break;
                case 1:       return "Mon       ";        break;
                case 2:       return "Tue       ";        break;
                case 3:       return "Wed       ";        break;
                case 4:       return "Thurs     ";        break;
                case 5:       return "Fri       ";        break;
                case 6:       return "Sat       ";        break;
                default:      return "          ";        break; }}

void InitTimer(){
        TCCR0 = 0x0F; // CTC Mode, Prescale 104
        OCR0 = 155; // count 0 to 155, considering F_CPU = 16 MHZ
        TIMSK = 0x02; // Output compare interrupt enable
}

ISR(TIMER0_COMP_vect) {//Timer Interrupt Service Routing function to handle time

        if (flag_stop_watch % 3 == 1) { // Flag is 1,so we should start stopwatch
calculation
                stop_mil_sec++; //each 10 mil seconds to meet 1/100 resolution
                if (stop_mil_sec >= 100) {  stop_sec++;   stop_mil_sec = 0;
```

```
                    if (stop_sec >= 60)  {      stop_min++;          stop_sec = 0;
                            if (stop_min >= 60){ stop_hour++;         stop_min = 0;}}}}

        mil_sec++;  //each 10 mil seconds to handle background time, even you do not see
timer is calculation current time after setting the time
        if (mil_sec >= 100)  {
                sec++;
                mil_sec = 0;
                if (sec >= 60){
                        min++;
                        sec = 0;
                        if (min >= 60){
                                hour++;
                                min = 0;
                                if (hour >= 24)
                                hour = 0;           }}}}
ISR (INT3_vect) // MOVE BACK BUTTON using INT3 interrupt, we clear all temporary
variables to default values, and clear the LCD by writing empty string due to some errors
in LCD_Clear() function
{
        PortInit();
        LCD_Init();
        LCD_Clear();
        LCD_pos(0, 0);
        LCD_STR("                         ");
        LCD_pos(1, 0);
        LCD_STR("                         ");
        temp_sec = 0;
        temp_min = 0;
        temp_hour = 0;
        temp_day = 1;
        temp_month = 1;
        flag_stop_watch = -1;
        state_flag = -1;
        temp_alarm_sec = 0;
        temp_alarm_min = 0;
        temp_alarm_hour = 0;
        SREG |= 0x80;
        mode = 100;
}

ISR (INT2_vect) // MOVE DOWN BUTTON, here we check in which mode we are and accordingly
change the global variables to handle external INT2 interrupt
{
        if (mode == 100) { // MAIN MENU
                temp_mode++;
                if (temp_mode > 4)
                temp_mode = 0;
        }else if (mode == 3) { // SET TIME MODE
                if (state_flag == 0) { // STATE FLAG CHECK TO KNOW WHICH VARIABLE WE ARE AT
                        temp_year--;
                        if (temp_year <= 0)                 temp_year = 1;
                        if (is_leap(temp_year))     {
                                month_max[1] = 29;

                        } else {      month_max[1] = 28;   }
```

```
            } else if (state_flag == 2){
                   if (temp_day > 1 )   {
                          temp_day--;
                          } else {temp_day = month_max[temp_month - 1];    }

            } else if (state_flag == 1)        {
                   if (temp_month > 1)         {
                          temp_month--;
                          } else {
                          temp_month = 12;
                   }
            } else if (state_flag == 3){
                   if (temp_hour > 0)   {
                          temp_hour--;
                          } else {
                          temp_hour = 23;      }

            }
            else if (state_flag == 4){
                   if (temp_min > 0)    {
                          temp_min--;
                          } else {      temp_min = 59;       }

            } else if (state_flag == 5){
                   if (temp_sec > 0)    {
                          temp_sec--;
                          } else {
                          temp_sec = 59;}}
     }
     else if (mode == 4) // ALARM TIME MODE SELECTION{
            if (state_flag == 0){ // STATE FLAG CHECK TO KNOW WHICH ATTRIBUTE TO CHANGE
                   if (temp_alarm_hour > 0){
                          temp_alarm_hour--;
                          } else {      temp_alarm_hour = 23;       }

            }else if (state_flag == 1)  {
                   if (temp_alarm_min > 0){
                          temp_alarm_min--;
                          } else {      temp_alarm_min = 59;        }

            } else if (state_flag == 2){
                   if (temp_alarm_sec > 0){
                          temp_alarm_sec--;
                          } else {      temp_alarm_sec = 59;        }
            }}}

ISR (INT1_vect){} // MOVE UP BUTTON has almost the same function body structure as in ISR
(INT2_vect) function with main difference is to INCREASE the attribute values to reach
the intended value by the user. It is handled using INT1 interrupt.
```

The last ISR function we have to implement is ISR(INT0_vect) which acts like OK button for us, and its function body is implemented as following:

```
ISR (INT0_vect) // OK BUTTON, here we check in which mode we are in, and store the
temporary variables to actual variables to further calculation.
{
	mode = temp_mode;

	PortInit();
	LCD_Init();
	LCD_Clear();
	LCD_pos(0, 0);
	LCD_STR("                         ");
	LCD_pos(1, 0);
	LCD_STR("                         ");

	if (mode == 0) // TIME DISPLAY MODE, by changing mode to 0, we ensure that in
while loop case 0 is executed where we show current time
	{
		SREG |= 0x80;
		mode = 0;
	}
	else if (mode == 1) // ALARM TIME DISPLAY MODE, to show ALARM TIME on the 1st line
and current time on the 2nd line
	{

		SREG |= 0x80;
		mode = 1;
	}
	else if (mode == 2){ // STOPWATCH MODE, if modulus operation results in 0,
variables should be all 0's because we did not start the stopwatch yet. We used modulus 3
operation on flag_stop_watch, since we have 3 states START/PAUSE/RESET. On reset mode
neither below mode nor if block in ISR(TIMER0_COMP_vect) executes enabling us to freeze
the variable calculation
		flag_stop_watch++;
		if (flag_stop_watch % 3 != 2){// this block works when we choose stopwatch,
and reset the stopwatch. Its default value is -1.
			stop_sec = 0;
			stop_min = 0;
			stop_hour = 0;
			stop_mil_sec = 0;
		}
		SREG |= 0x80;
		mode = 2;
	}else if (mode == 3) { // SET TIME MODE, where we set the time attributes using
temporary attrubutes.
		if (state_flag == -1){// STATE FLAG CHECK TO HANDLE INITIALIZATION ORDER OF
ATTRIBUTES
			state_flag++;

		} else if (state_flag == 0) {
			year = temp_year;
			state_flag++;
		} else if (state_flag == 2) {
			day = temp_day;
			state_flag++;
		} else if (state_flag == 1) {
			month = temp_month;
			state_flag++;
```

```
            } else if (state_flag == 3) {
                   hour = temp_hour;
                   state_flag++;
            }else if (state_flag == 4)  {
                   min = temp_min;
                   state_flag++;
            } else if (state_flag == 5) {
                   sec = temp_sec;
                   state_flag++;
            } else if (state_flag == 6) {
                   temp_sec = 0;
                   temp_min = 0;
                   temp_hour = 0;
                   temp_day = 1;
                   temp_month = 1;
                   state_flag = -1;
                   mode = 100;
            }
     }
     else if (mode == 4){ // ALARM TIME SET MODE, the same procedure as SET TIME MODE
but for ALARM TIME attributes
            if (state_flag == -1){
                   state_flag++;

            } else if (state_flag == 0) {
                   alarm_hour = temp_alarm_hour;
                   state_flag++;
            } else if (state_flag == 1) {
                   alarm_min = temp_alarm_min;
                   state_flag++;
            } else if (state_flag == 2) {
                   alarm_sec = temp_alarm_sec;
                   state_flag++;
            } else if (state_flag == 3) {
                   temp_alarm_sec = 0;
                   temp_alarm_min = 0;
                   temp_alarm_hour = 0;
                   state_flag = -1;
                   mode = 100;          }      }      }

int main(void){ // MAIN FUNCTION STRUCTURE is simple having only one while loop with
inner switch to switch between required modes. Mode variable is HANDLED in ISR(INT0_vect)
interrupt service routine.
     while (1){ // INFINITE WHILE LOOP
            switch (mode) // MODE CHECK
            {
                   case 0:      // SHOWING CURRENT TIME HERE              break;
                   case 1:      // SHOWING ALARM TIME HERE                break;
                   case 2:      // SHOWING STOPWATCH MODE HERE            break;
                   case 3:      // SHOWING SET TIME MODE HERE             break;
                   case 4:      // SHOWING SET ALARM TIME MODE HERE       break;
                   default:     // default mode is to clear the LCD
                   LCD_Clear();
                   break;
            }
```

```
              if (hour == alarm_hour && min == alarm_min && sec == alarm_sec){
// ANY TIME THE ALARM TIME COMES LEDs ARE BLINKED AUTOMATICALLY EVEN IF YOU ARE NOT IN
SHOW ALARM TIME MODE
                     int i = 0;
                     for (; i < 5; i++){// BLINKING LED 5 TIMES
                            char LED;
                            LED = 0xff; // 1111 1111, turn off all LEDs
                            PORTB = LED;
                            _delay_ms(300);
                            LED = 0x00; //0000 0000, turn on all LEDs again
                            PORTB = LED;
                            _delay_ms(300);
                     }//for loop end      }//switch end }// while end }// end of main
```

Video URL

We have created a DEMO video to show our working system, using Bandicam screen recording application. All team members were online conferencing on Zoom desktop application, and explaining all modes implemented and the user manual. You can watch the demo video through the following link.

Team member contribution

Member List	Team Role	Final Project Contribution
Oybek Amonov (U1610176)	As a team member, always be active during the meetings. And, use theoretical knowledge to help the team to design and plan the whole cycle of development. Try to find more materials(online/offline) to present during the meeting and give some short summary about their relevance to our project. Mainly focus on the design phase, planning the whole development process phases. Creating theoretical analysis of the system, requirements and etc. Learning about hardware and software specification of the project such as getting to know the Atmega128 and SimulIDE	Designed the whole architecture theoretically, was involved during whole implementation phase to monitor the correctness of the implementation. And, helped to handle implementation phase errors. Did some coding in C language, to implement correct implementation of Interrupt and Timer Interrupt handlers. Done requirement analysis, thus wrote the Requirement Analysis document, also wrote some parts for SW Design and Report.
Rakhmatjon Khasanov (U1610183)	As a recorder, participate in every meeting on time. Take notes all the time, analyze the whole meeting, and provide brief summary related to the meeting. Record and constantly analyze the progress meeting by meeting, provide summary/suggestion to the team leader, and so on. Mainly focus on the implementation phase, learning to code in assembly/C languages.	Fully focused on implementation phase, and coding in Atmel Studio. Implemented automized weekday assignment function, and main function body. Involved in implementing ISR functions, and fixing errors and bugs. Mainly wrote the SW design document, and helped to write final report where code explanation was needed. Kept recording notes during Zoom meetings, to understand what and how to implement the system.

Bokhodir Urinboev* (1610249)	As a team leader, plan, organize meetings, supervise them. Manage the team fairly and efficiently. Coordinate the team for urgent goals and make plans for the 2 weeks ahead, and monitor its progress. Plan, organize, monitor and lead the project and its members step-by-step. Getting involved in both design and implementation phase. Doing some documentation for the project.	Organized whole cycle of development phase, and monitored the completing goals within a deadline, did division of tasks to ensure efficient development period. Involved in both design and implementation phase, doing researches on how to design the system and code in C language for AVR timer. Did coding, and prepared materials to use during the Zoom meetings. Controlled required documents writing, and mainly wrote the Final Report. Made the Demo Video.

Conclusion

By the end of the Digital Clock Project, our team has gained sufficient skills in C programming (basically, interrupt handling), and got used to the Atmel Environment and working with SimulIDE application. We learned a lot about Atmega128 microcontroller, its kit, SimulIDE, atmega128 circuit and LCD (HD44780). Programming in C for microcontroller was challenging, considering the flash memory limits, computing power limitations, so we tried to minimize both storage and computing requirement of our system.

All members got both theoretical and practical knowledge about Atmega128 microcontroller and using it.

References

[1] Atmel 8-bit Microcontroller with 128 bytes in-System Programmable Flash

[2] https://www.youtube.com/watch?v=PNlSHrDM3jo – Sample Digital Clock demo video

[3] https://www.patreon.com/posts/simulide-0-3-12-35657927 - SimulIDE

[4] https://drive.google.com/open?id=1vBCQeEhz8_Ln2E0ZIGyQyJ3Yi8BxVU9q – Atmega128 Circuit

[5] https://youtu.be/zNBbukkOazI - Demo Video on YouTube

[6] http://web.engr.oregonstate.edu/~traylor/ece473/lectures/atmega128_overview.pdf

[7] https://www.tutorialspoint.com/c_standard_library/c_function_sprintf.htm