EEE 416 – Microprocessor and Embedded Systems Laboratory

Jan 2020 Level-4 Term-I Section C

Final Project Demonstration

BANGLA DIGITAL CLOCK

SUBMITTED BY – GROUP C.02

Haider Ali Shuvo
1606134

Ratul Kundu
1606147

Prasun Datta
1606148

Raisa Bentay Hossain
1606151

Department of Electrical and Electronics Engineering

Bangladesh University of Engineering and Technology

Outline

- Summary
- Background
- Methods
- Simulation
- PCB layout and 3d rendering
- Future Outlook
- Conclusion

Summary

Goals:

- ❑ A Bangla digital clock will be implemented with ARM cortex m series boards.
- ❑ LED matrix displays or LCD displays can be implemented.
- ❑ Buttons should be given to adjust hour minute and seconds.
- ❑ Bonus: Implement the entire project in a casing, show AM/PM and time zone.

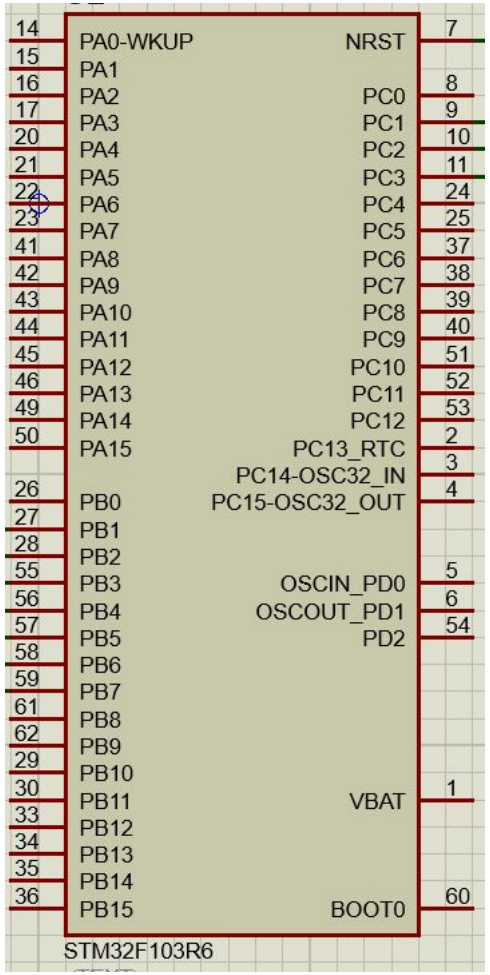
Goals Fulfilled:

- ✔ A Bangla digital clock has been implemented with STM32F 103R6.
- ✔ LCD displays is used to display the time.
- ✔ Buttons are given to adjust hour, minute and seconds.
- ✔ Bonus: AM/PM is shown.

Background

Not everyone in Bangladesh can use English clock. Besides, it is a good idea to use our native language in a tool which is used in our daily life for more comfortable experience.

Methods-1: Hierarchy



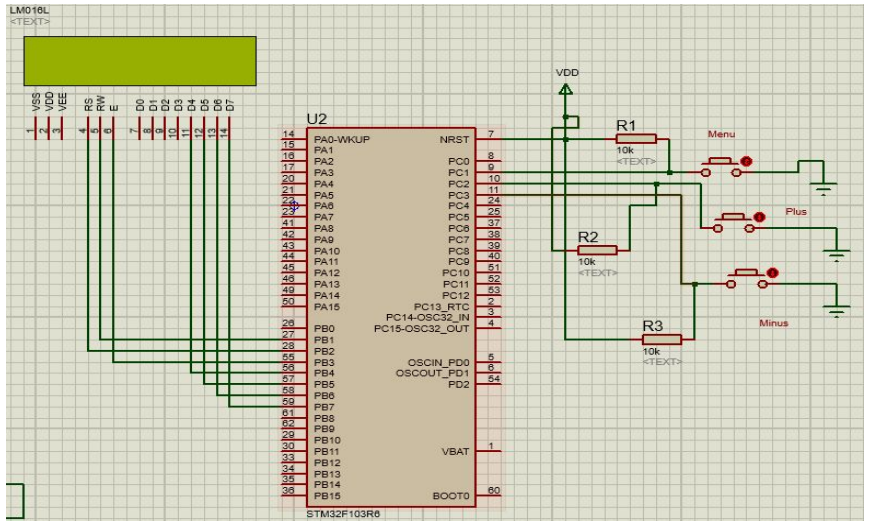
STM32F 103R6

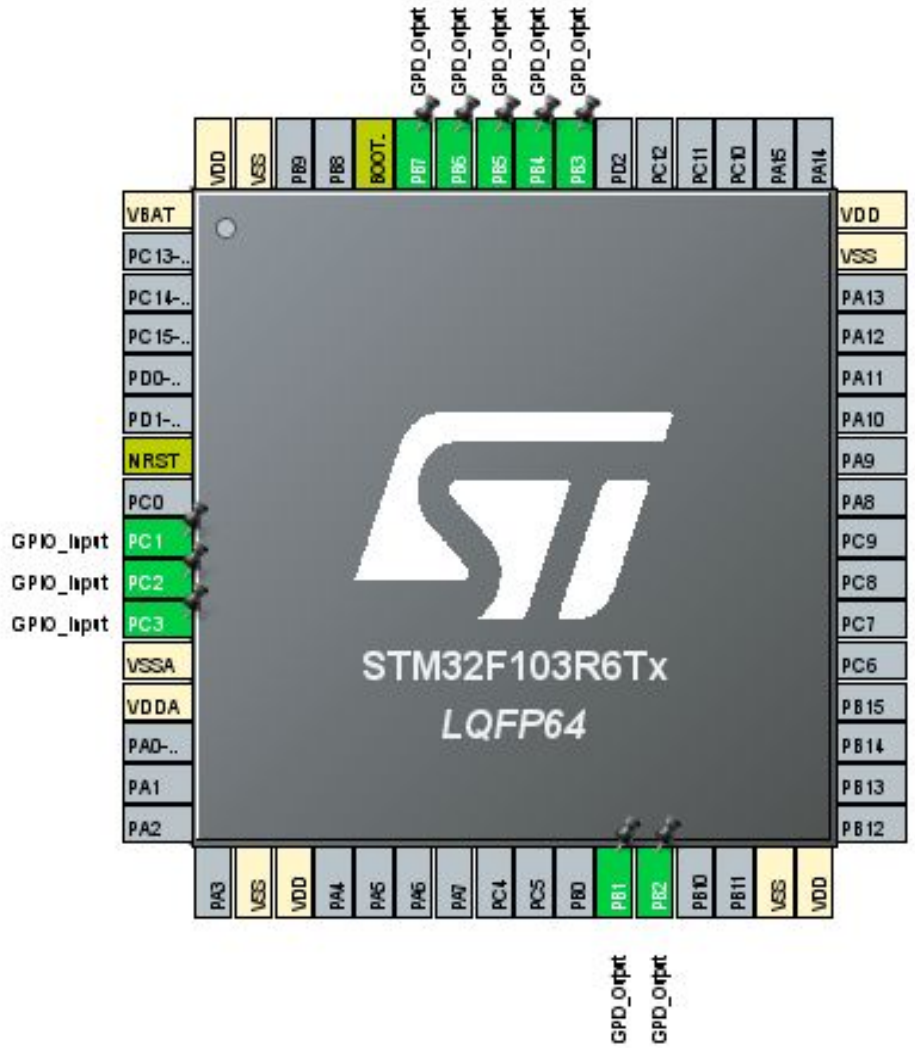
STM32 CubeMX

ARMKEIL Microcontroller Tools

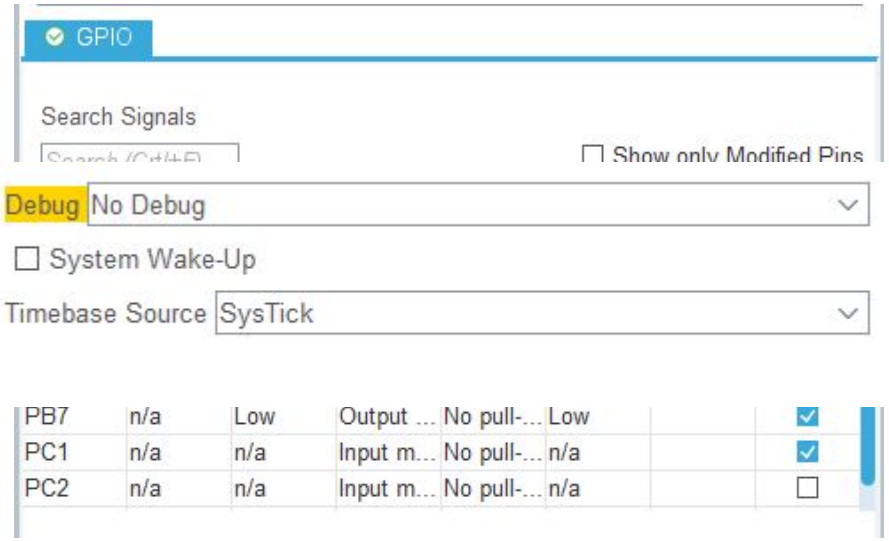
PROTEUS CAD CONNECTED

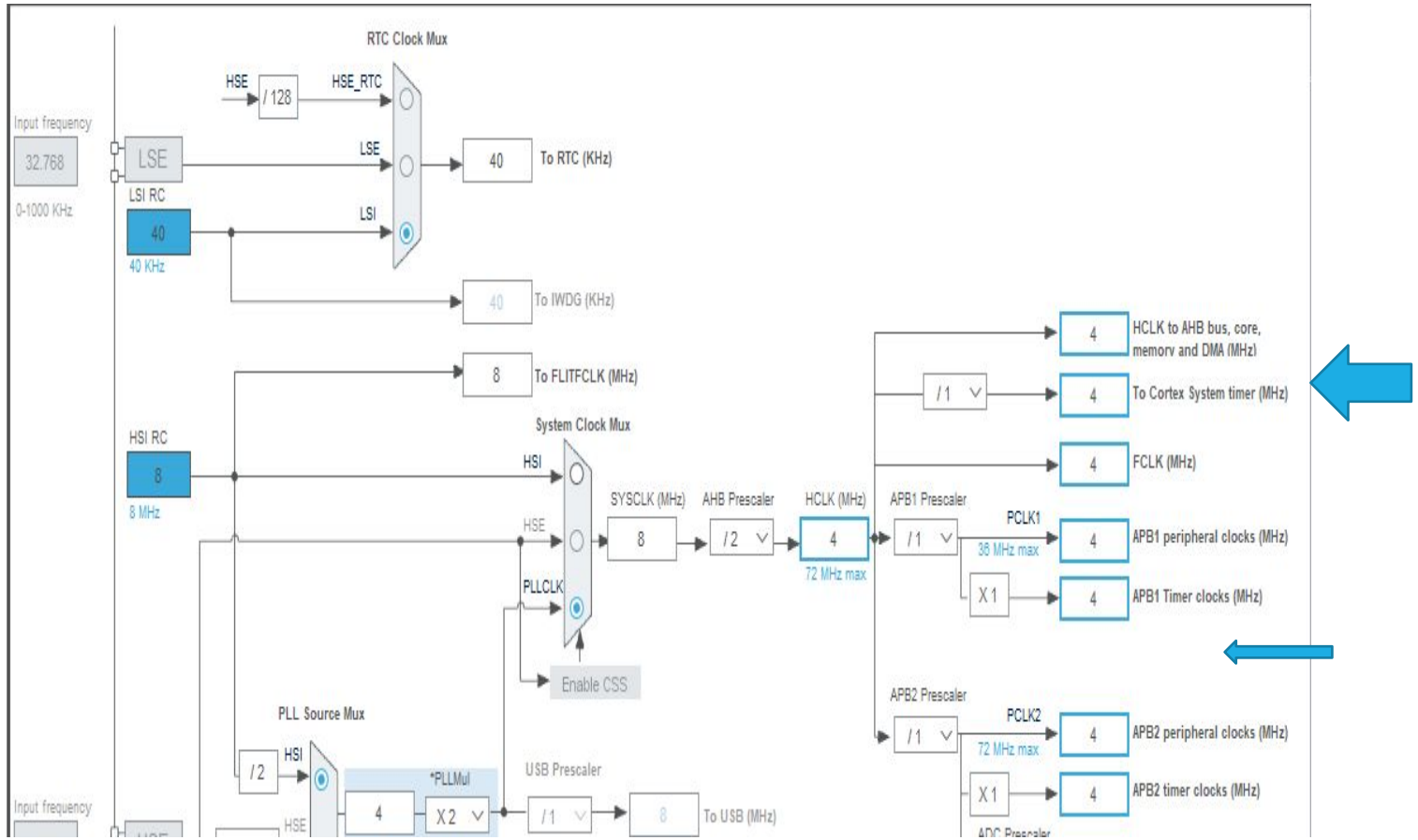
Methods-2 : Setting Up pins in STM32CubeMX





Methods-3:Generating Initial Code With other essential Setups in CubeMX

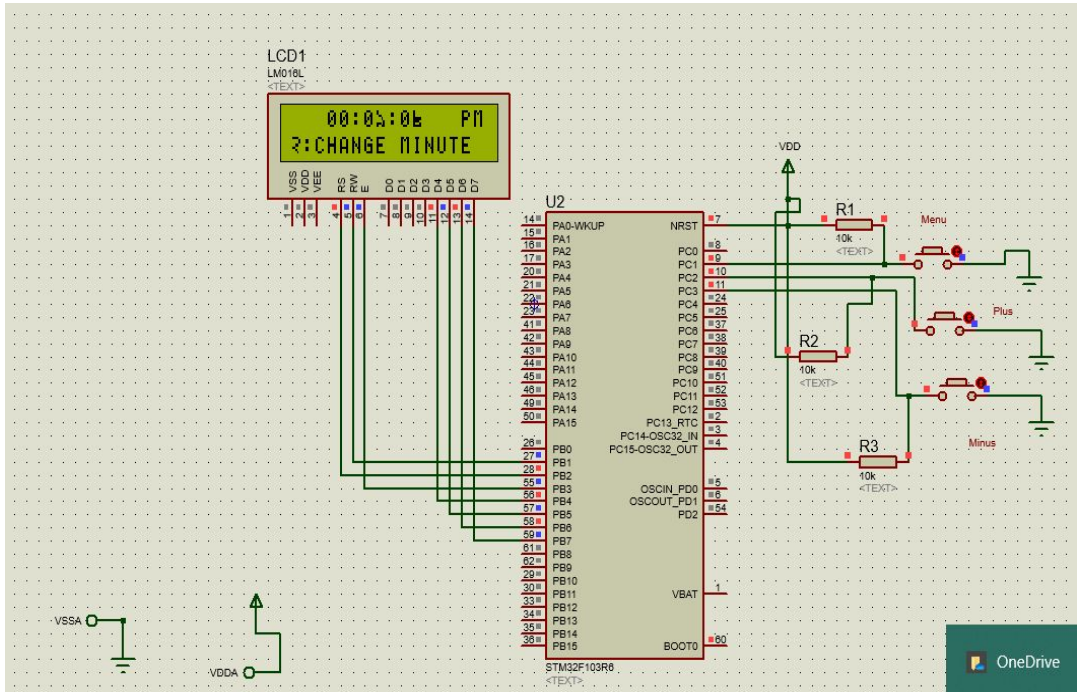




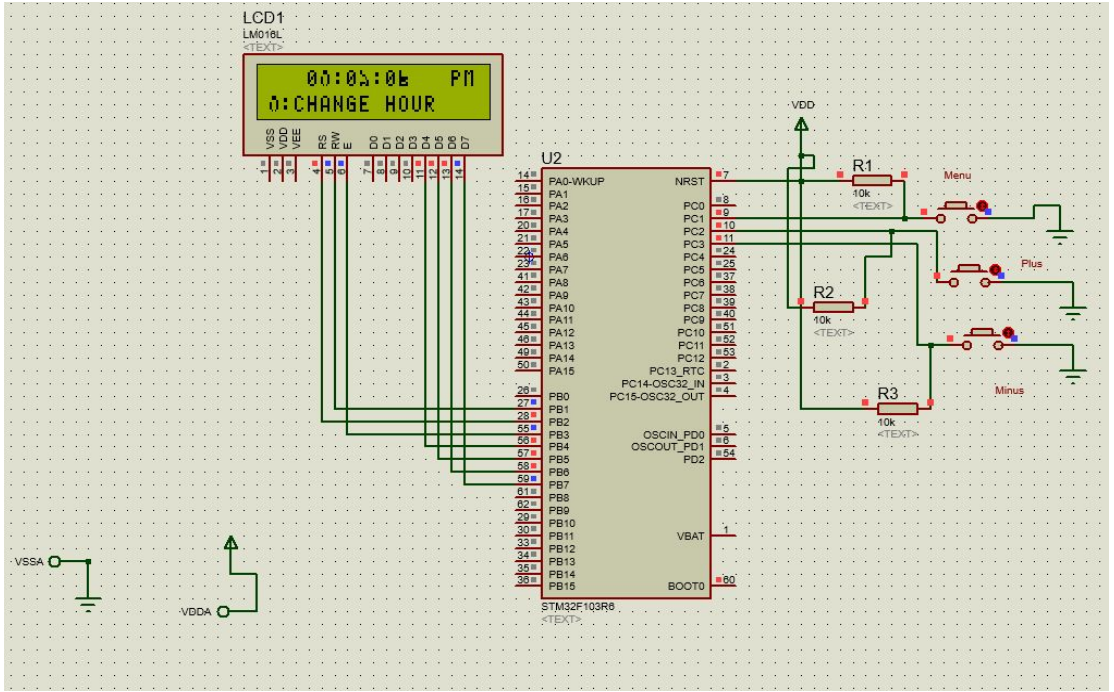
Simulation

Key features

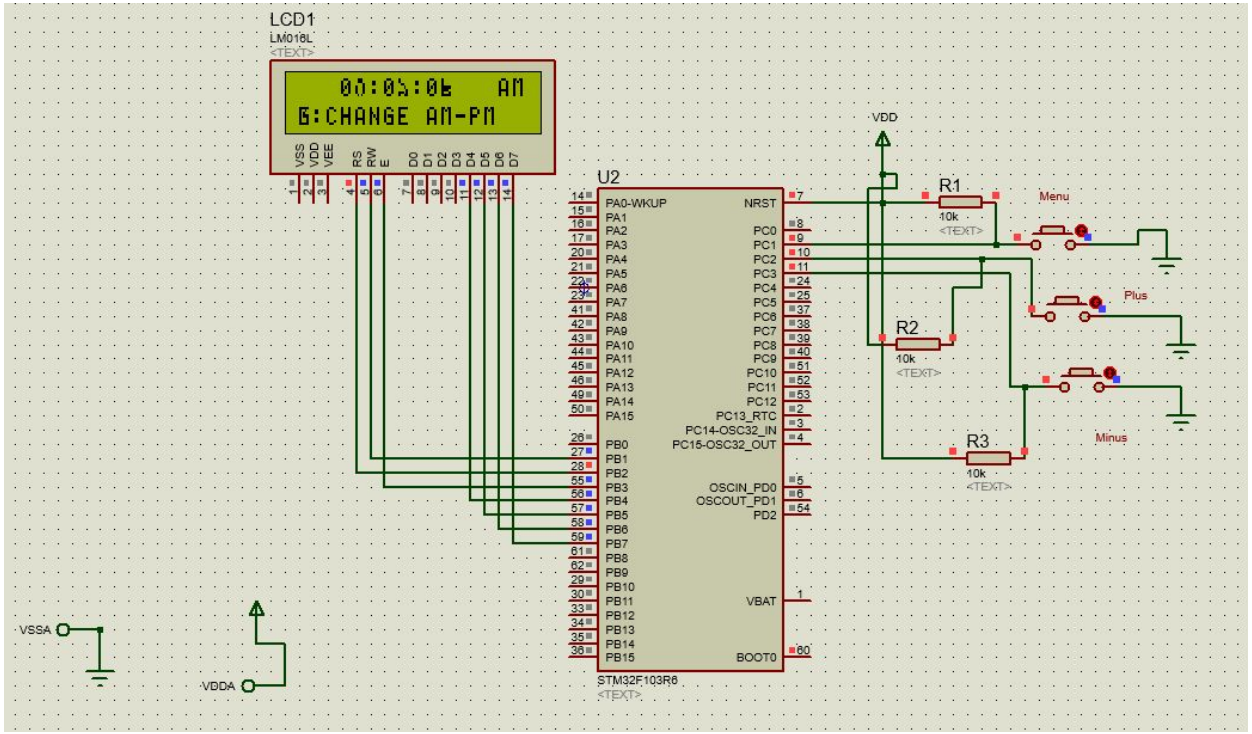
Change Minute:



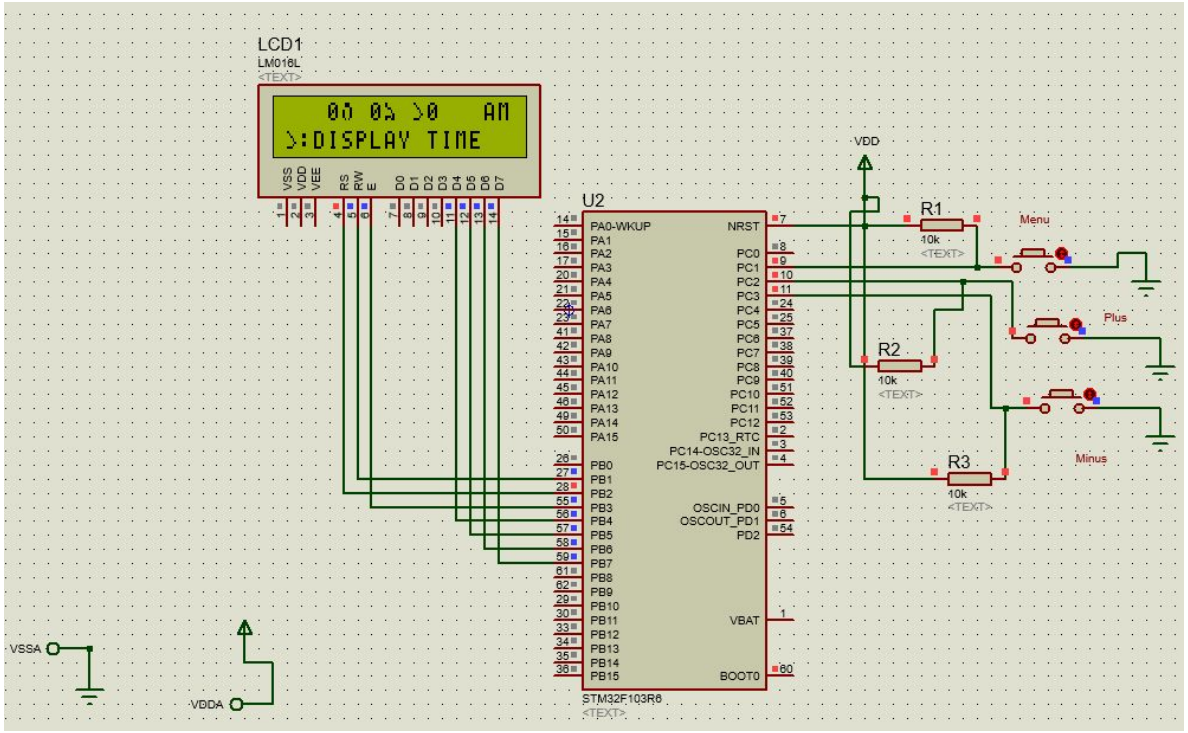
Change Hour:



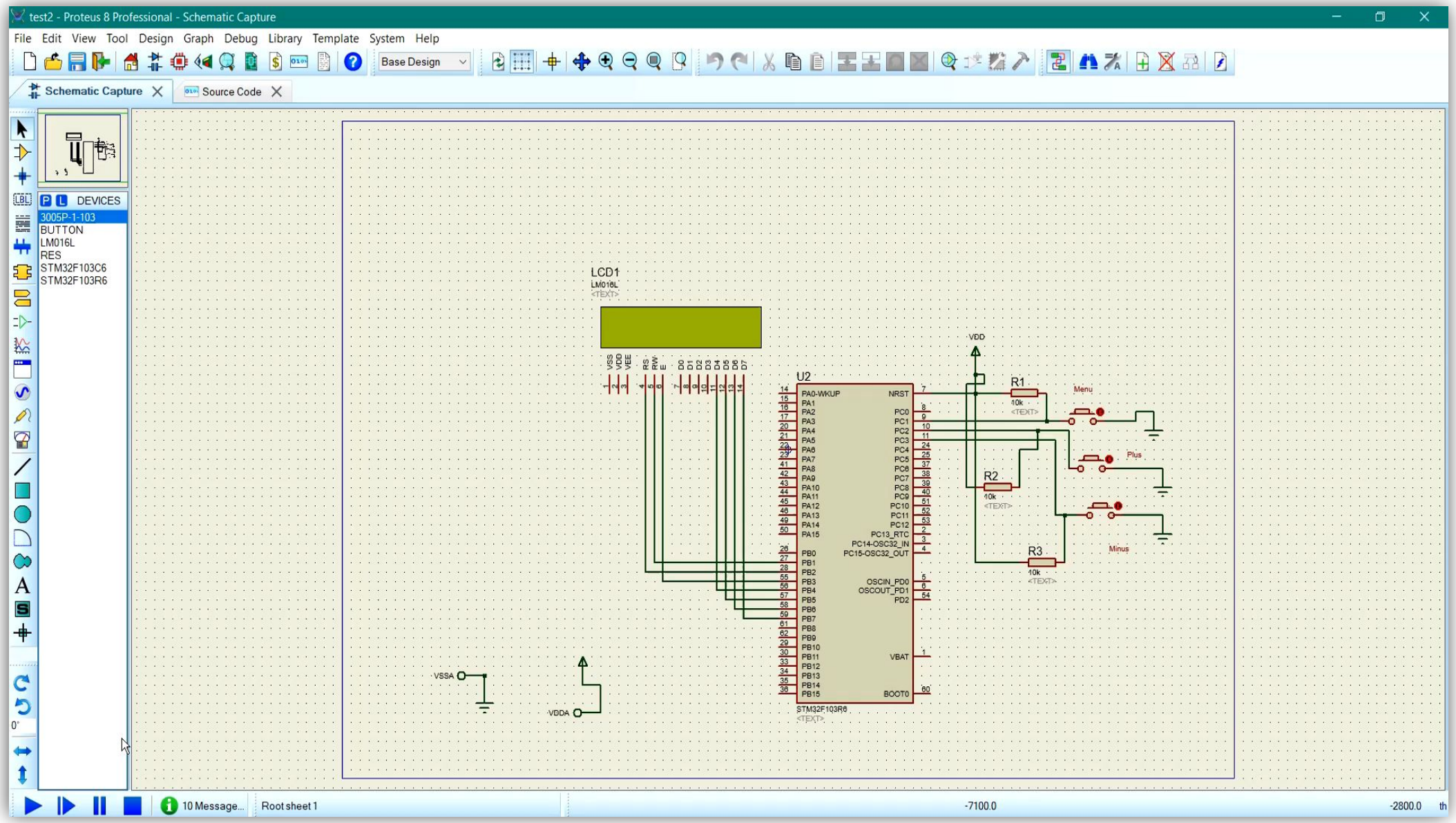
Change AM-PM



Display Current time

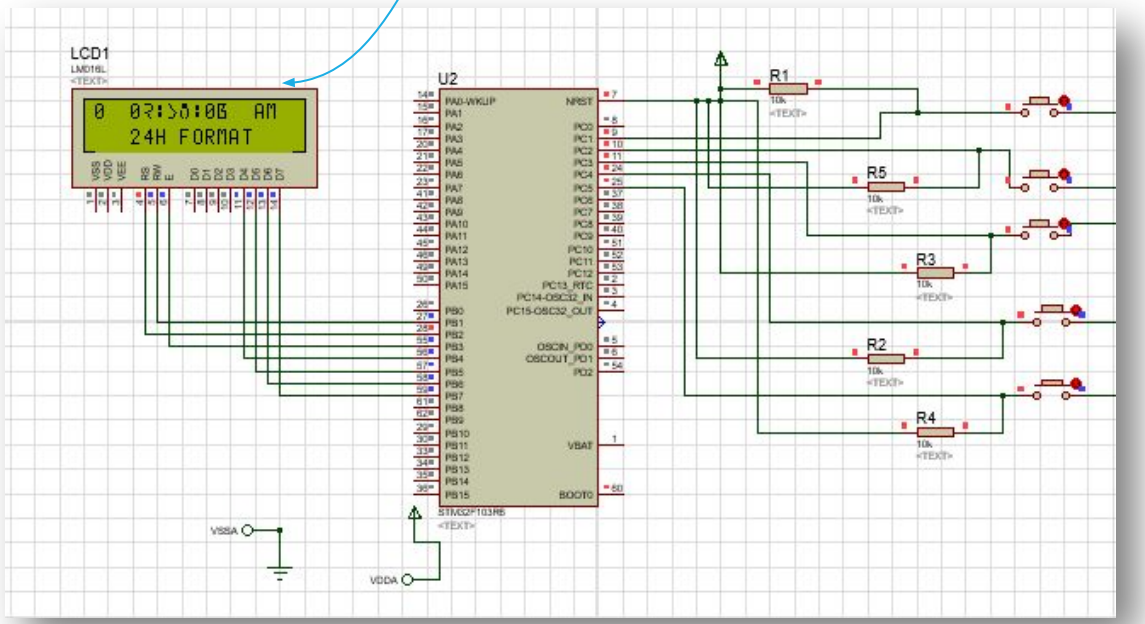


Video Demonstration

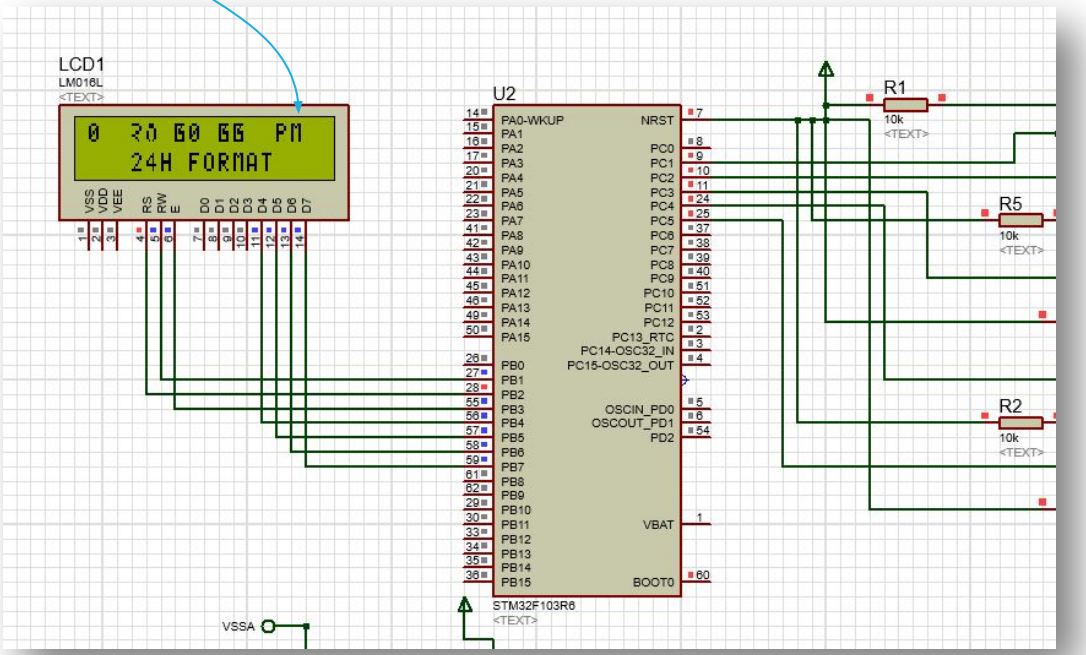


Simulation: Clock With RTC(Real Time Clock)

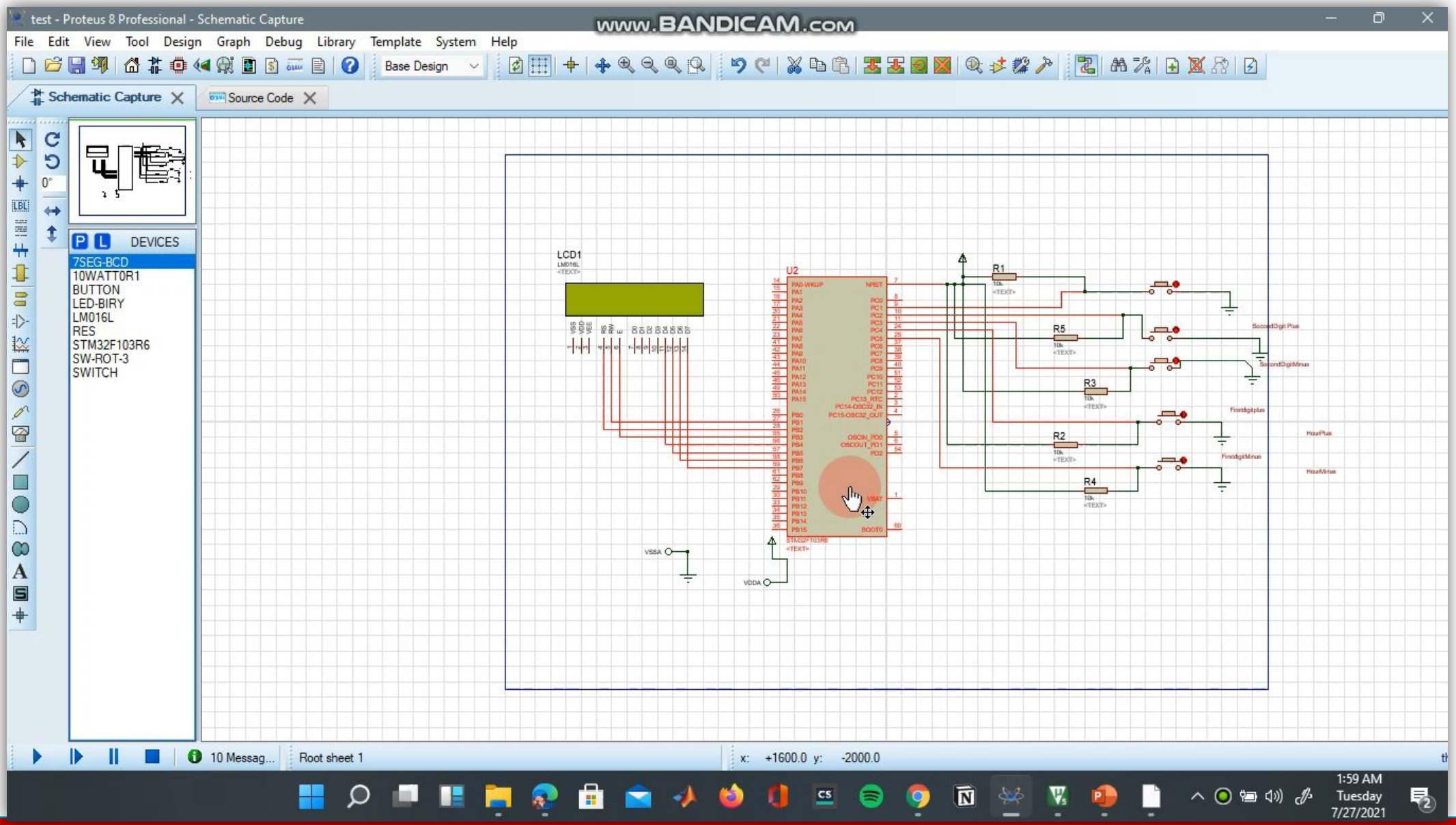
Displaying Current System Time (from AM part)



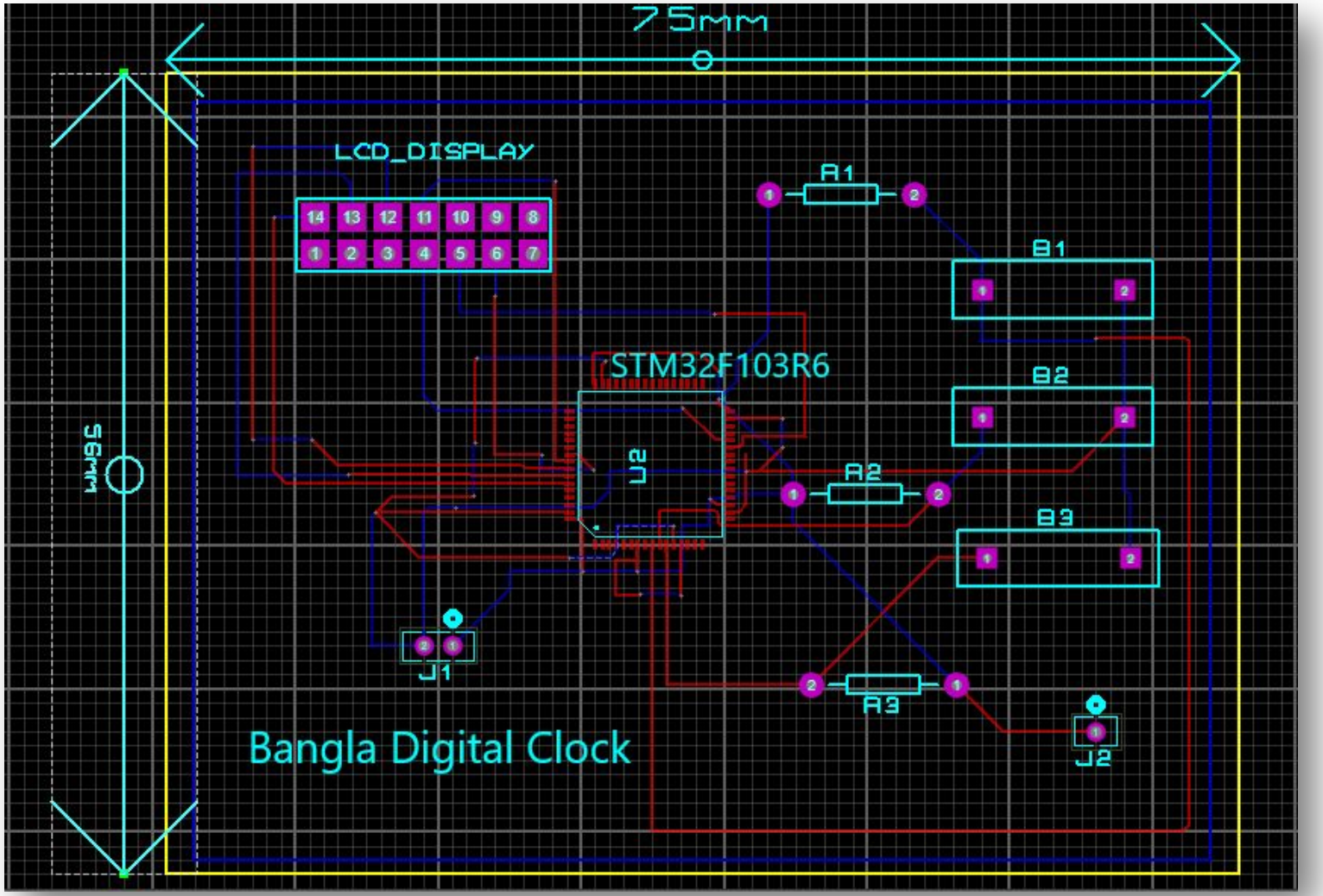
Displaying Time(from PM part)



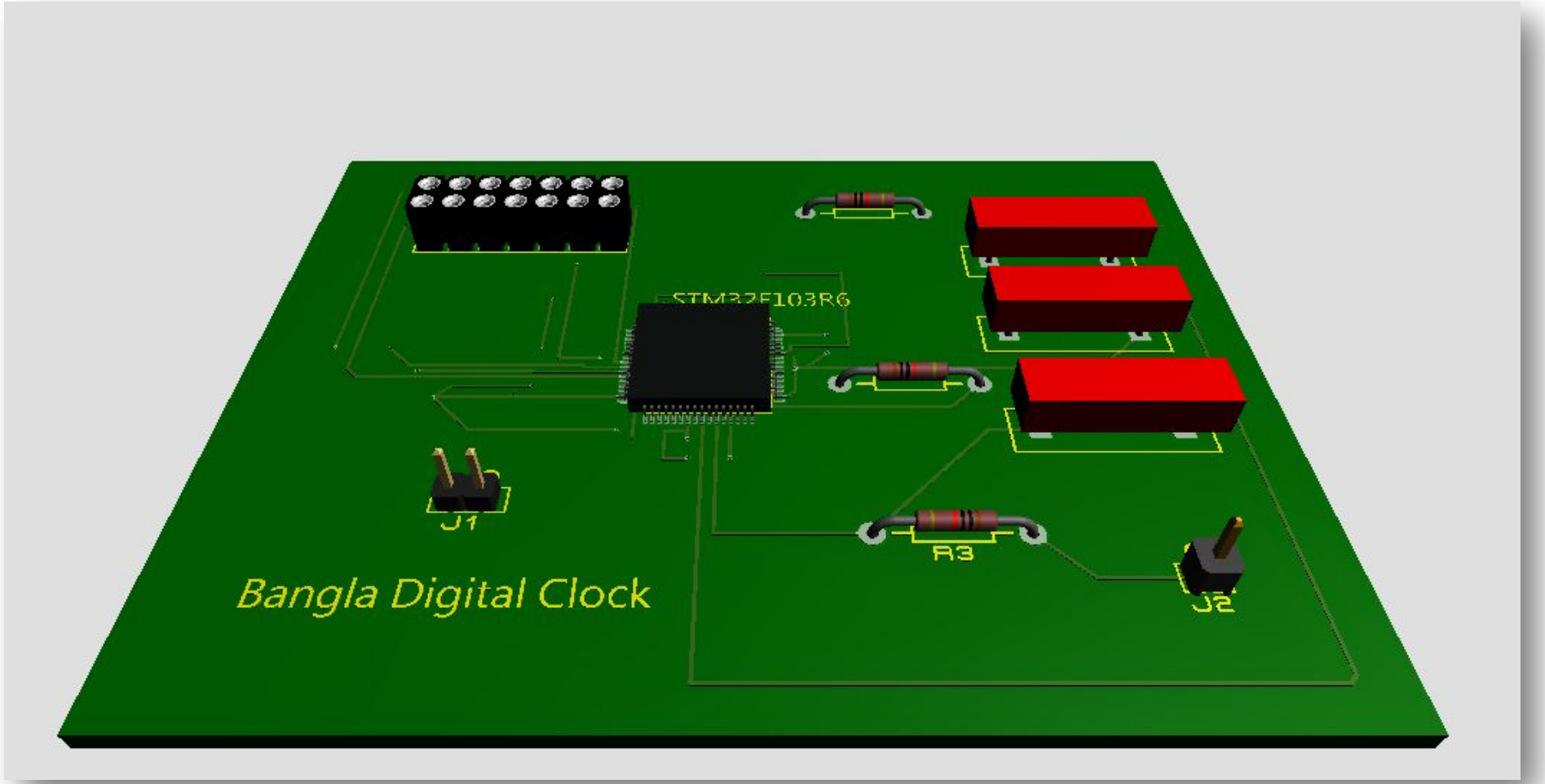
Video Demonstration



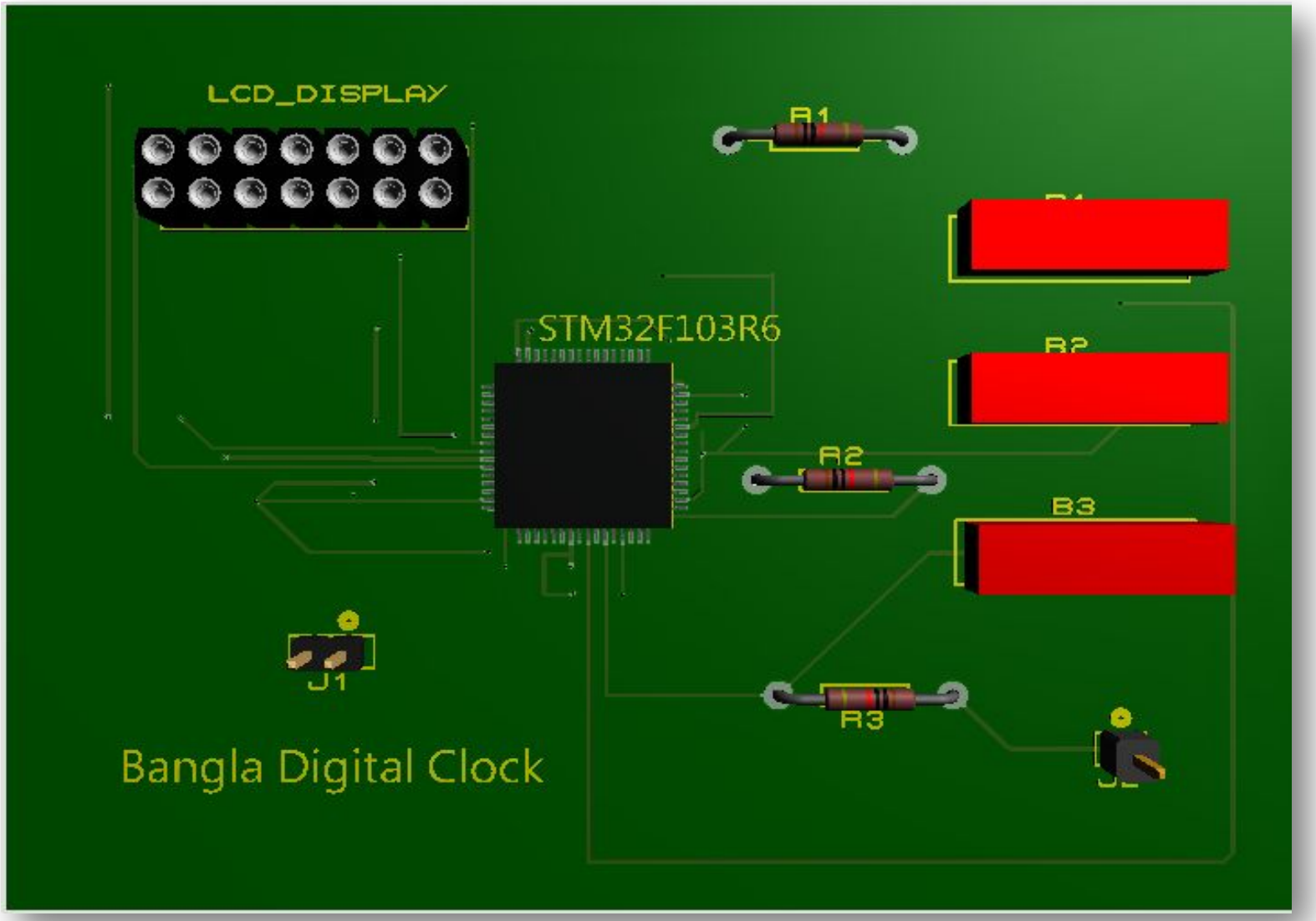
PCB Layout

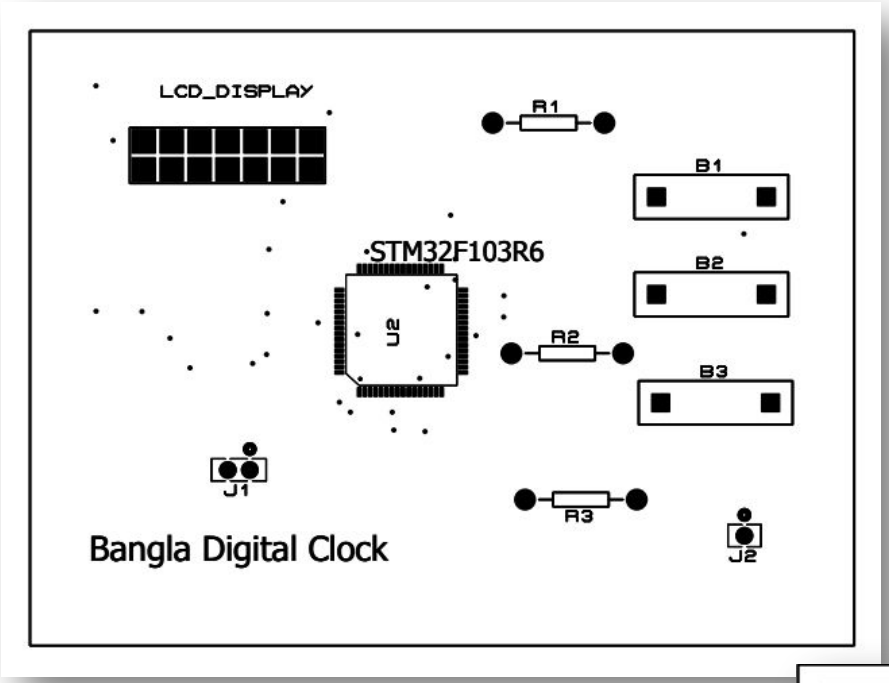

75mm
56mm
LCD_DISPLAY
14 13 12 11 10 9 8
1 2 3 4 5 6 7
A1
B1
B2
B3
A2
A3
STM32F103R6
U2
J1
J2
Bangla Digital Clock

3d rendering

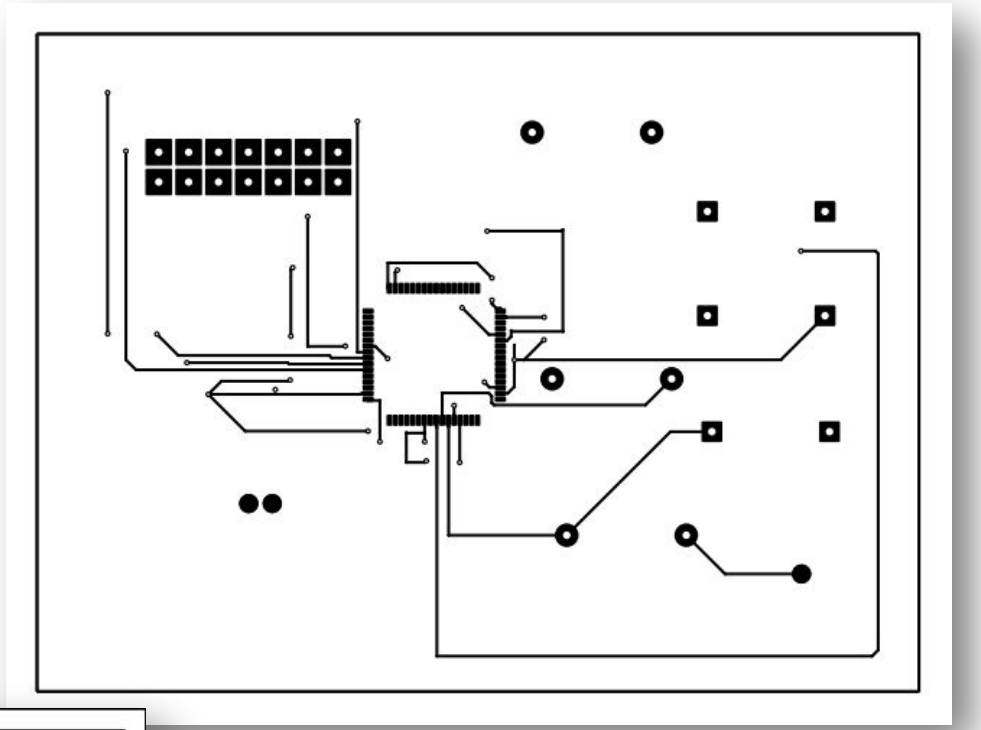


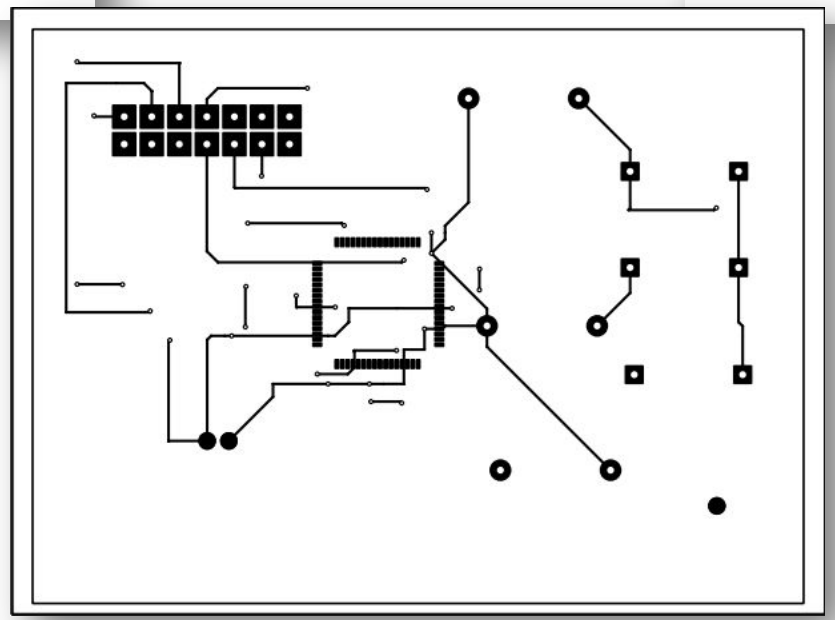
Top View of 3d Rendered PCB





Top Silk:Position of Components that will be placed on PCB





Top Copper: Traces that will be on the upper layer of PCB

Bottom Copper:
Traces that will be on the bottom side of PCB

Future Outlook

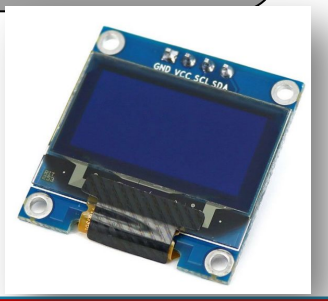
<u>This project can be extended further in the following directions:</u>

- For showing time, led matrix display can be used which will make the project more alluring.

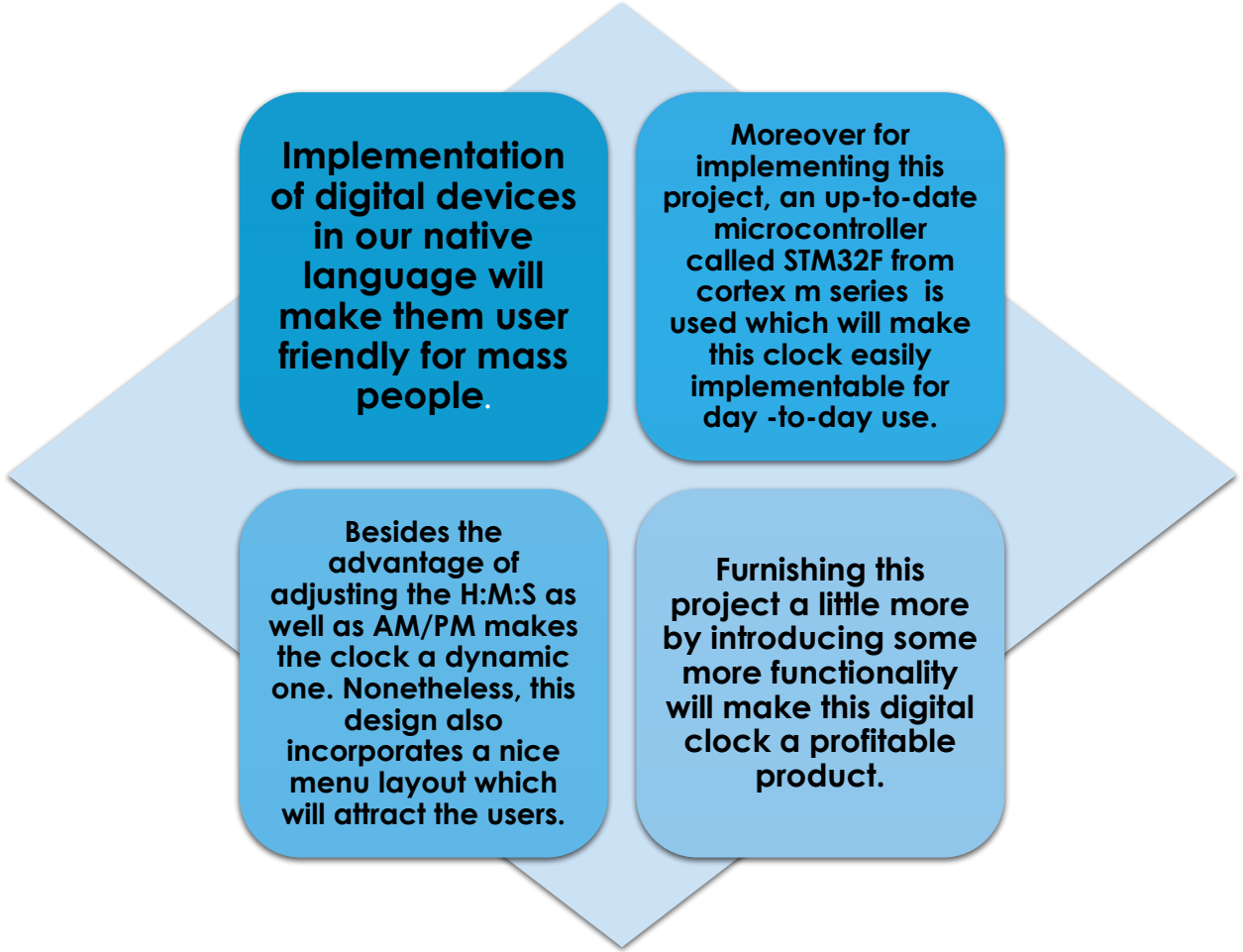
In this project, "Real Time Clock(RTC)" embedded in the STM32F 103R6 has been used to display time. To change the hour,min and sec, RTC interrupt coding has to be incorporated. This can be a promising extension of the project.

Furthermore, inclusion of time zone would make the project more dynamic as well as user friendly.

Last but not the least, integration of organic light-emitting diode (OLED) screen would make this project more realistic.



Conclusion


Implementation of digital devices in our native language will make them user friendly for mass people.
Moreover for implementing this project, an up-to-date microcontroller called STM32F from cortex m series is used which will make this clock easily implementable for day -to-day use.
Besides the advantage of adjusting the H:M:S as well as AM/PM makes the clock a dynamic one. Nonetheless, this design also incorporates a nice menu layout which will attract the users.
Furnishing this project a little more by introducing some more functionality will make this digital clock a profitable product.

ADDITIONAL SLIDES

References

Custom Character generator-

https://maxpromer.github.io/LCD-Character-Creator/

Stm32 RTC tutorial-

https://www.youtube.com/watch?v=c6D-IiWSbqA

Liquid Crystal Library for stm32-

https://github.com/SayidHosseini/STM32LiquidCrystal

Stm32f103R6 Datasheet-

https://www.st.com/resource/en/datasheet/cd00210843.pdf

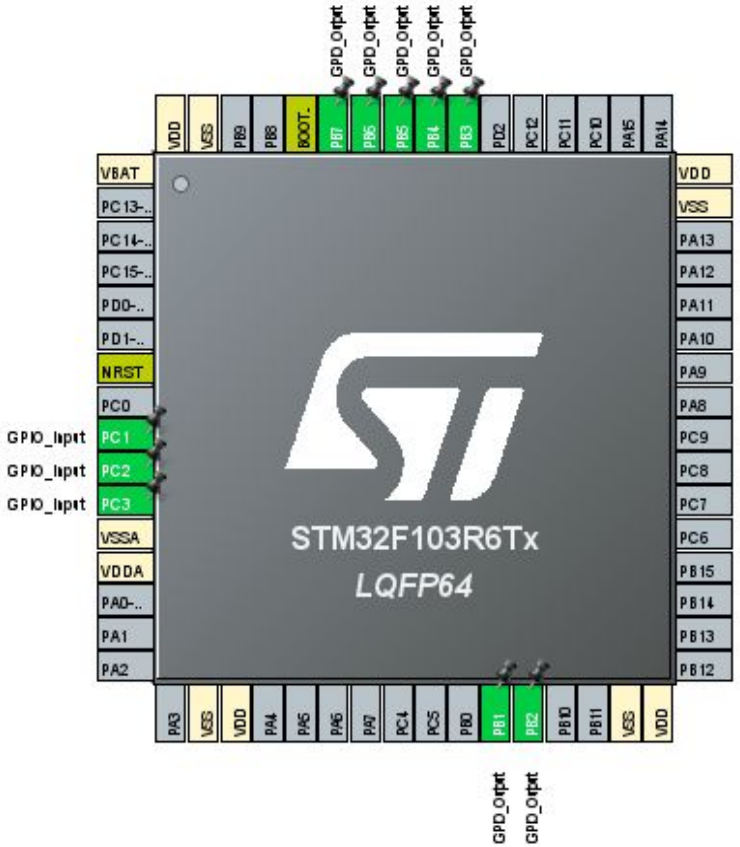
Arduino custom character function-

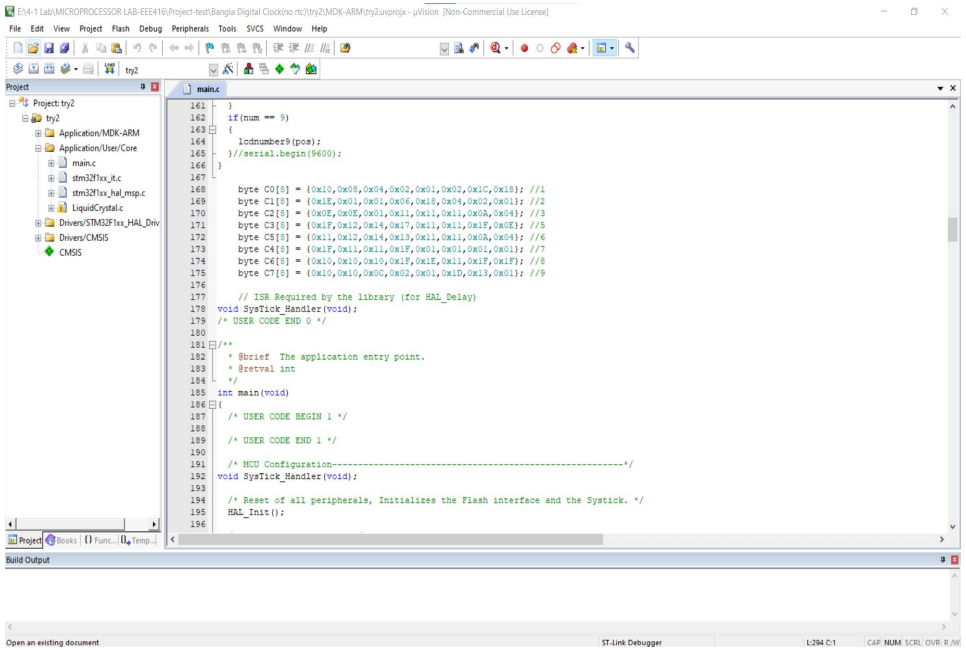
https://www.arduino.cc/en/Reference/LiquidCrystalCreateChar

Proteus PCB Design tutorial links:
https://www.youtube.com/watch?v=TwHGJlall88
https://www.youtube.com/watch?v=TwHGJlall88

Detailed Methods

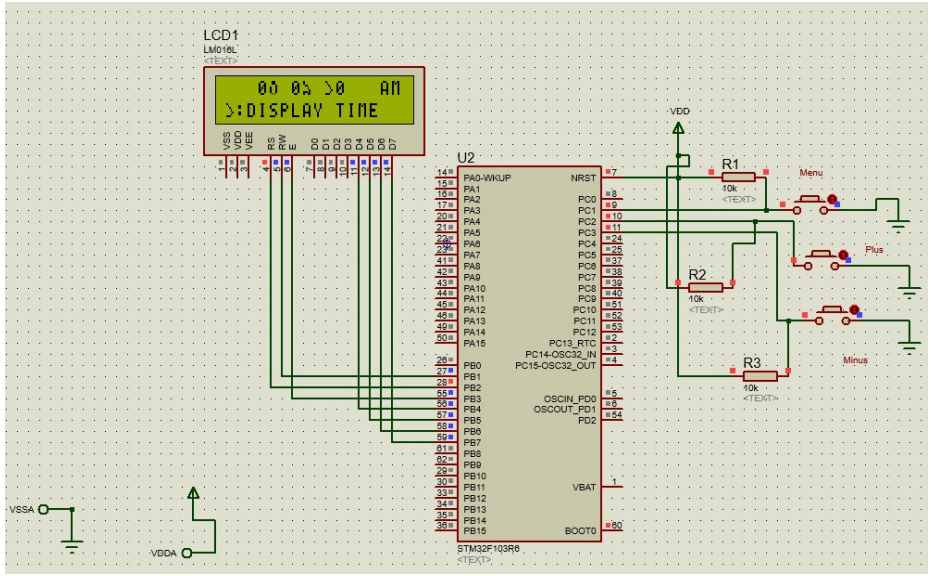
Softwares used for this project & tutorial video URLs and screen shots

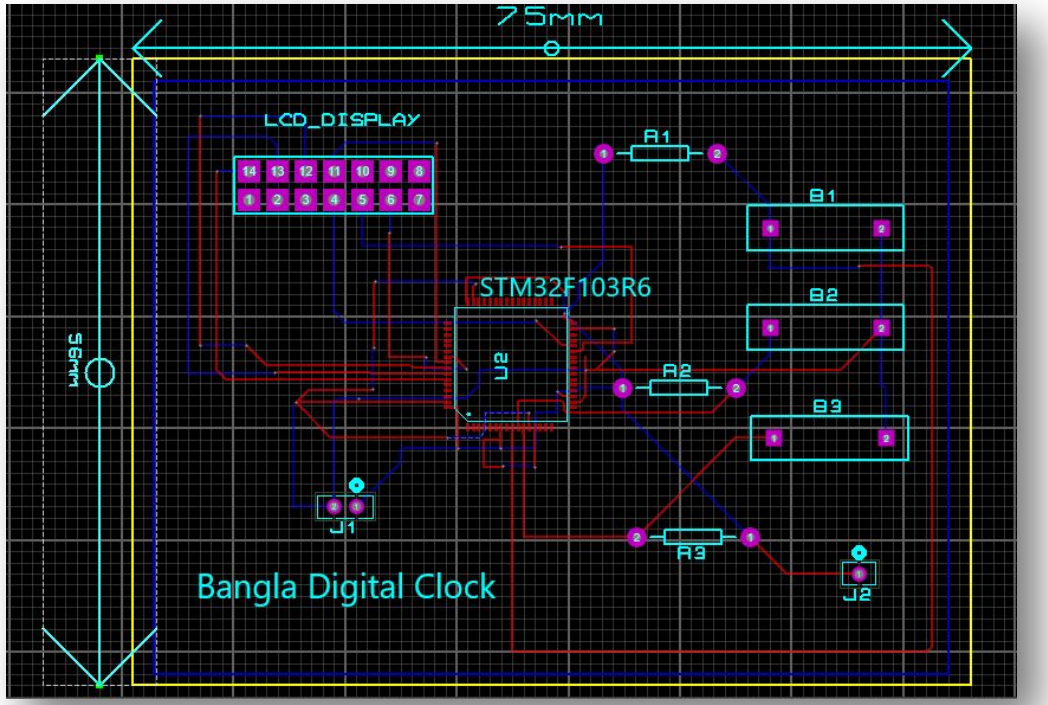




STM32CubeMX

ARM Keil uvision5





Proteus schematic capture

Proteus PCB Design tool

Explanation Of How The Software Used

STM32CubeMX:

STM32CubeMX is a graphical tool that allows a very easy configuration of STM32 microcontrollers and microprocessors, as well as the generation of the corresponding initialization C code. The step-by-step process is,

1. Selecting a STMicroelectronics STM32 microcontroller.
2. The next step was to pin-out and configuration. To configure GPIO and NVIC (which is necessary to add interrupt) and timer.
3. After that we configure the clock. For microcontroller, it is a clock-tree setting helper.
4. The last thing we did was to generate a C code, compliment with MDK-ARM.

Explanation Of How The Software Used

ARM Keil MicroVision5:

Keil MicroVision is a free software which solves many of the pain points for an embedded program developer. This software is an integrated development environment (IDE), which integrated a text editor to write programs, a compiler and it will convert your source code to hex files too.

We have used ARM Keil MicroVision5 to,

- Writing programs in C
- Compiling and Assembling Programs
- Debugging program
- Creating Hex file
- Testing our program without Available real Hardware (Simulator Mode)

Explanation Of How The Software Used

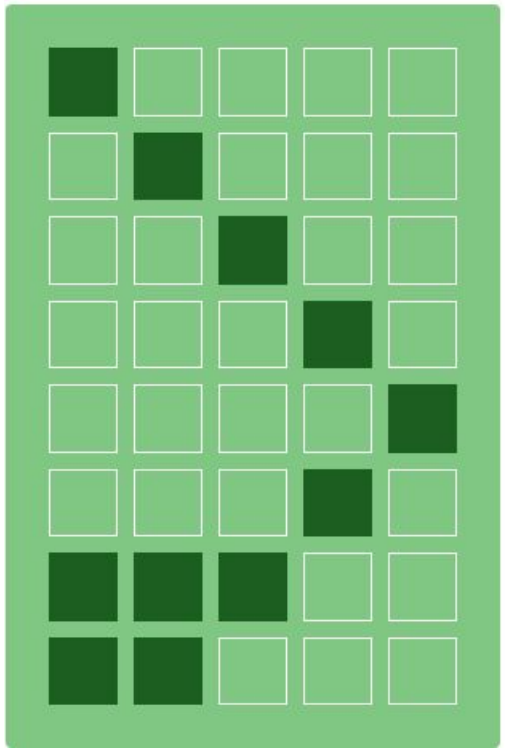
Proteus,

As we have no physical slide, we are using PROTEUS as the circuit simulator. We have used this for,

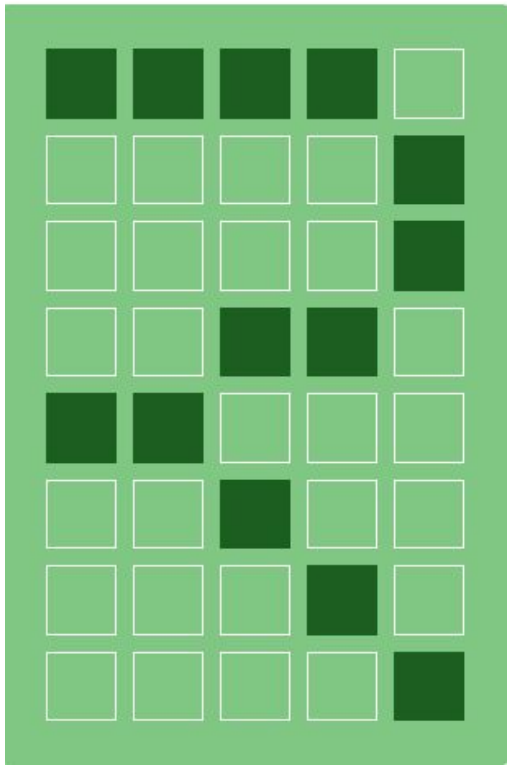
- Build the circuit
- Design the PCB layout.

Custom Character Generator-https://maxpromer.github.io/LCD-Character-Creator/

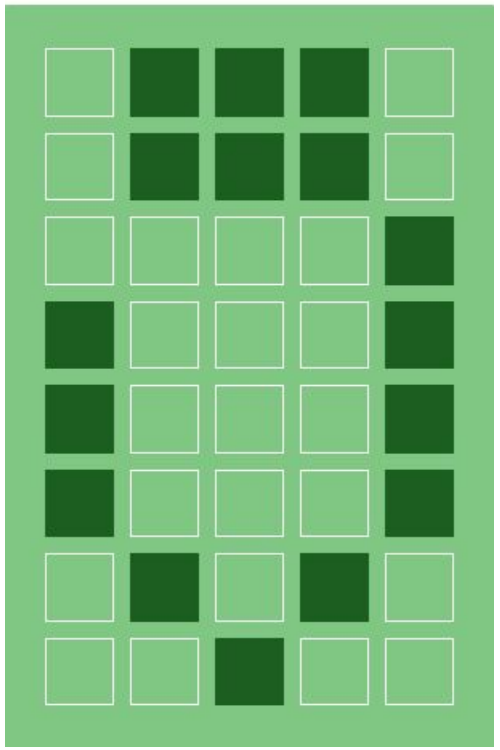
```
byte customChar[] = { 0x10, 0x08, 0x04, 0x02, 0x01, 0x02, 0x1C, 0x18 };
```



```
byte customChar[] = { 0x1E, 0x01, 0x01, 0x06, 0x18, 0x04, 0x02, 0x01 };
```

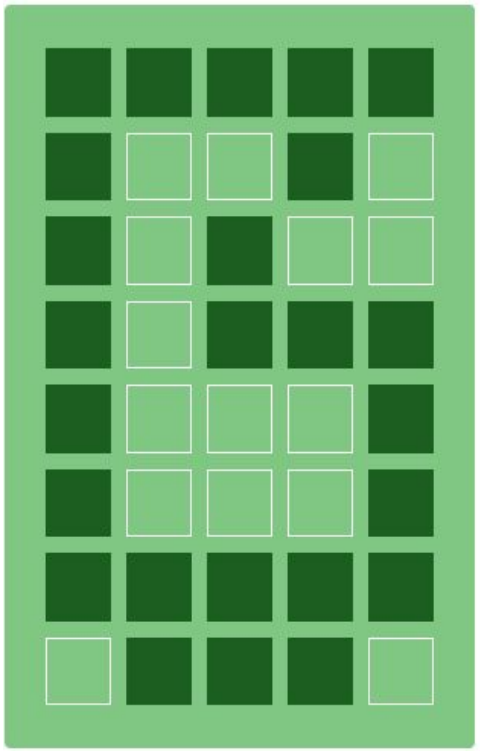


```
byte customChar[] = { 0x0E, 0x0E, 0x01, 0x11, 0x11, 0x11, 0x0A, 0x04 };
```

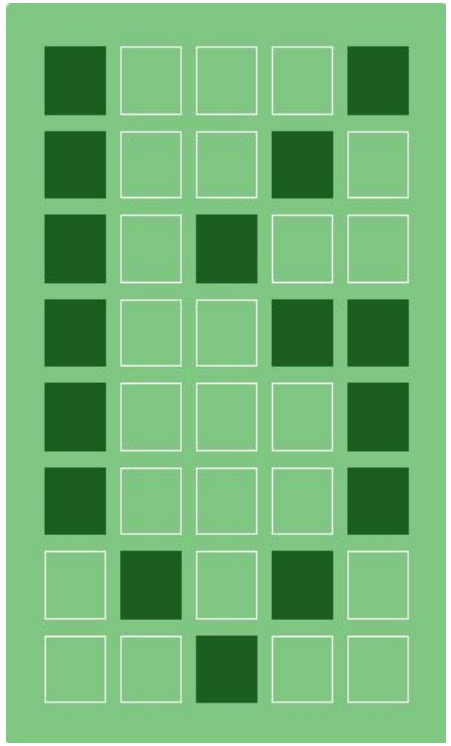


8

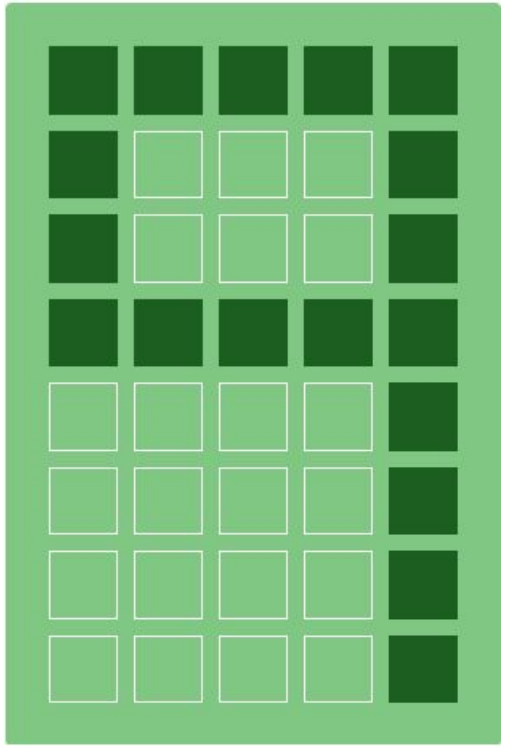
```
byte customChar[] = { 0x1F, 0x12, 0x14, 0x17, 0x11, 0x11, 0x1F, 0x0E };
```

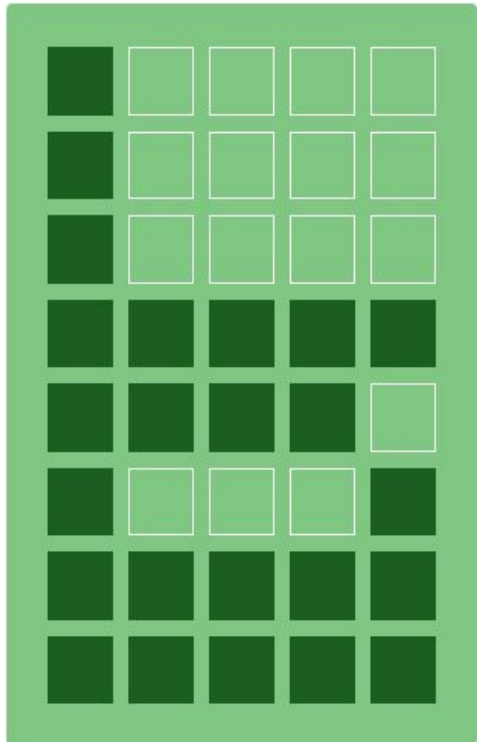


```
byte customChar[] = { 0x11, 0x12, 0x14, 0x13, 0x11, 0x11, 0x0A, 0x04
};
```



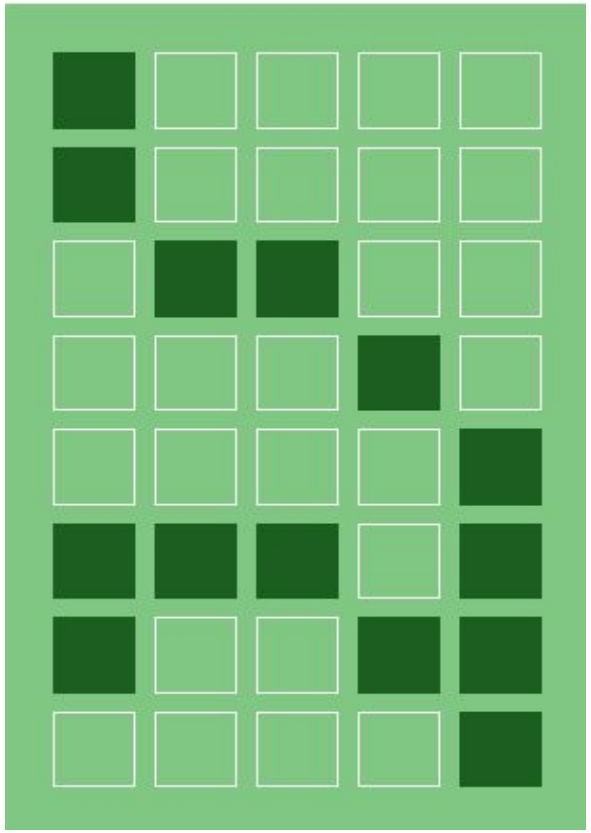
```
byte customChar[] = { 0x10, 0x10, 0x10, 0x1F, 0x1E, 0x11, 0x1F, 0x1F
};
```

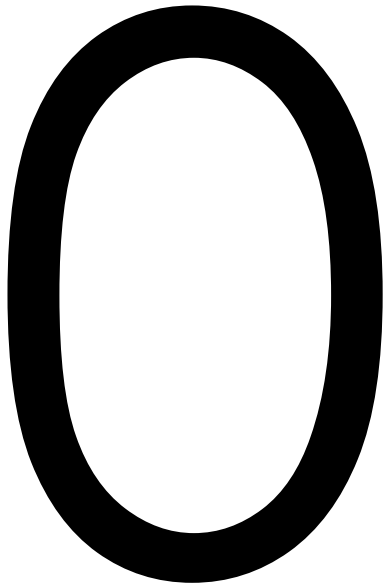




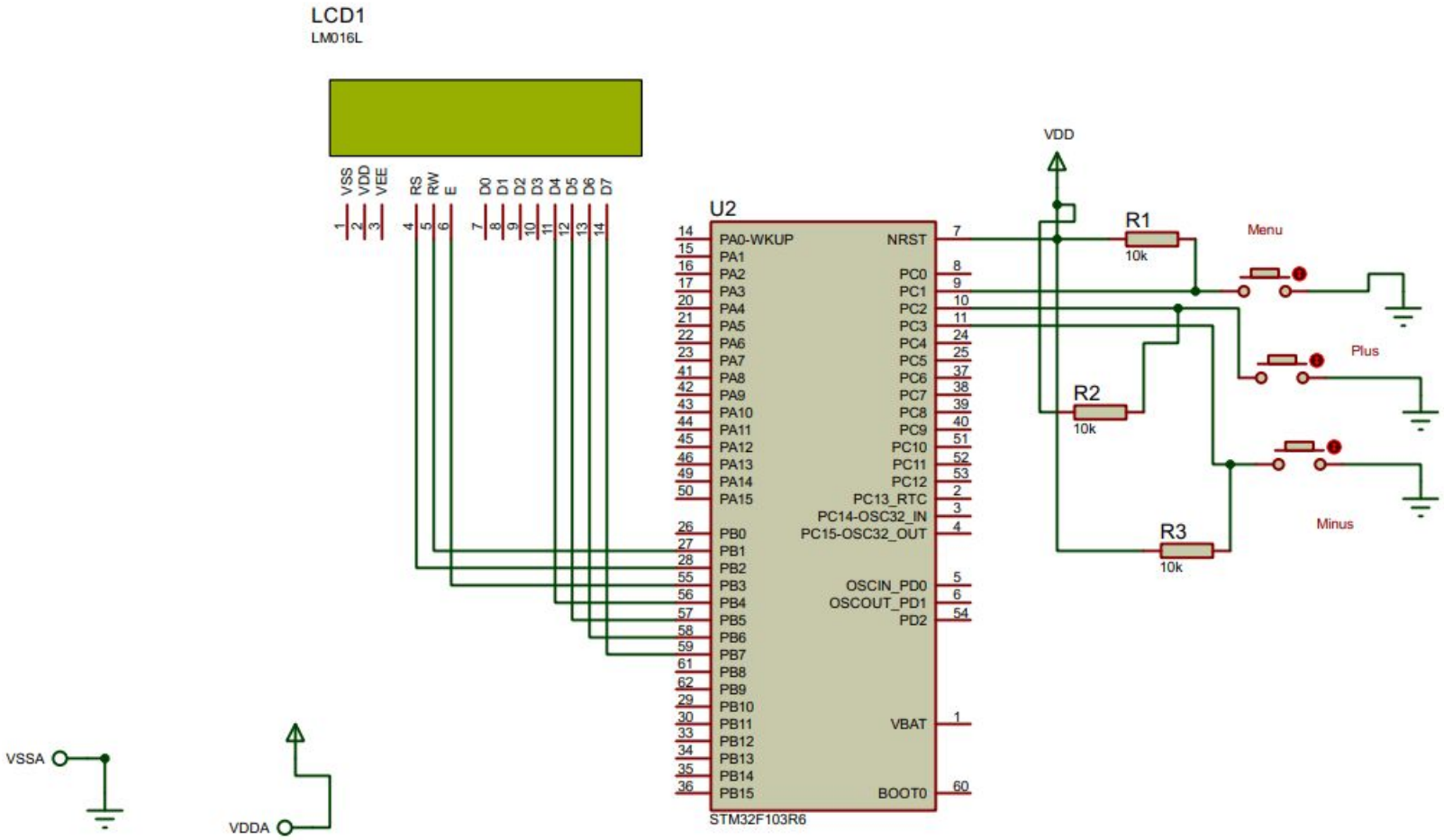
```
byte customChar[] = { 0x1F, 0x11, 0x11, 0x1F, 0x01, 0x01, 0x01,
0x01 };
```

```
byte customChar[] = { 0x10, 0x10, 0x0C, 0x02, 0x01, 0x1D, 0x13, 0x01 };
```





Circuit Diagram



Source Codes

Function Name : lcdnumber(0~9)

```
void lcdnumber0(int startposition)
{
   setCursor(startposition+0,0);
   print("0");

}
void lcdnumber1(int startposition)
{
   setCursor(startposition+0,0);
   write((0));

}
void lcdnumber2(int startposition)
{
   setCursor(startposition+0,0);
   write((1));

}
void lcdnumber3(int startposition)
{
   setCursor(startposition+0,0);
   write((2));
}
```

```
void lcdnumber4(int startposition)
{
   setCursor(startposition+0,0);
   print("8");

}
void lcdnumber5(int startposition)
{
   setCursor(startposition+0,0);
   write((3));  //5

}

void lcdnumber6(int startposition)
{
   setCursor(startposition+0,0);
   write((5));

}
```

```
void lcdnumber7(int startposition)
{
   setCursor(startposition+0,0);
   write((4));

}

void lcdnumber8(int startposition)
{
   setCursor(startposition+0,0);
   write((6));

}
void lcdnumber9(int startposition)
{
   setCursor(startposition+0,0);
   write((7));

}
```

Source Codes

Function Name : numberprinter(int num , int pos)

```
 void numberprinter(int num , int pos)
{
  if(num == 0)
  {
    lcdnumber0(pos);
  }
  if(num == 1)
  {
    lcdnumber1(pos);
  }
  if(num == 2)
  {
    lcdnumber2(pos);
  }
  if(num == 3)
  {
    lcdnumber3(pos);
  }
if(num == 4)
  {
    lcdnumber4(pos);
  }
  if(num == 5)
  {
    lcdnumber5(pos);
  }
  if(num == 6)
  {
    lcdnumber6(pos);
  }
  if(num == 7)
  {
    lcdnumber7(pos);
  }
  if(num == 8)
  {
    lcdnumber8(pos);
  }
  if(num == 9)
  {
    lcdnumber9(pos);
  }//serial.begin(9600);
}
```

Source Codes

Function : character generator

```
byte C0[8] = {0x10,0x08,0x04,0x02,0x01,0x02,0x1C,0x18}; //1
byte C1[8] = {0x1E,0x01,0x01,0x06,0x18,0x04,0x02,0x01}; //2
byte C2[8] = {0x0E,0x0E,0x01,0x11,0x11,0x11,0x0A,0x04}; //3
byte C3[8] = {0x1F,0x12,0x14,0x17,0x11,0x11,0x1F,0x0E}; //5
byte C5[8] = {0x11,0x12,0x14,0x13,0x11,0x11,0x0A,0x04}; //6
byte C4[8] = {0x1F,0x11,0x11,0x1F,0x01,0x01,0x01,0x01}; //7
byte C6[8] = {0x10,0x10,0x10,0x1F,0x1E,0x11,0x1F,0x1F}; //8
byte C7[8] = {0x10,0x10,0x0C,0x02,0x01,0x1D,0x13,0x01}; //9
```

LCD Initialization & Bangla Character Generation:

Function : LiquidCrystal (function called in main code)

```
LiquidCrystal(GPIOB, GPIO_PIN_2, GPIO_PIN_1, GPIO_PIN_3, GPIO_PIN_4, GPIO_PIN_5, GPIO_PIN_6, GPIO_PIN_7);

 // create a new character
 createChar(0, C0);  //1
 createChar(1, C1);  //2
 createChar(2, C2); //3
 createChar(3, C3);  //5
 createChar(4, C4);  //7
 createChar(5, C5);  //6
 createChar(6, C6);  //8
 createChar(7, C7);  //9
```

Menu Configuration:

```
        setCursor(0, 1);
        write(cnt);
       if(cnt==0)
                        {
                                setCursor(1, 1);
                               print(":DISPLAY TIME");
                                setCursor(14, 1);
                               print(" ");
                        }

      if(cnt==1)
                        {
                                setCursor(1, 1);
                                 print(":CHANGE MINUTE");
                        }

      if(cnt==2)
                        {
                                setCursor(1, 1);
                               print(":CHANGE HOUR");
                                setCursor(13, 1);
                               print("  ");
                        }
      if(cnt==3)
                        {
                                setCursor(1, 1);
                               print(":CHANGE AM-PM");
                                setCursor(14, 1);
                               print(" ");
                        }
```

PushButtons:

```
if(HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_1)==GPIO_PIN_RESET && state==1)
{
   cnt++;
   state=0;
          setCursor(0, 1);
   write(cnt);

   if(cnt>3){
     cnt=0;
                          setCursor(0, 1);
     write(cnt);
     }
   }
else if(HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_1)!=GPIO_PIN_RESET && state==0)

                {
   state=1;
   }

if(HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_2)==GPIO_PIN_RESET && state==1)
{
   dg=1;
   state1=0;

   }
else if(HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_2)!=GPIO_PIN_RESET && state==0)

{
   state1=1;
   }
```

```
if(HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_3)==GPIO_PIN_RESET && state==1)

{
    dg=-1;
    state2=0;
   }

         else if(HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_3)!=GPIO_PIN_RESET &&
state==0)

         {
   state2=1;
   }
```

Clock Functioning:

```
state=1;
state1=1;
state2=1;
s=0;
m=0;
h=0;
a=0;
```

```
switch(cnt)
{
   case 1:
      m=m+dg;
      dg=0;
   if(m>59)
{
    m=59;
 }
    if(m<0)
{
    m=0;
}
          break;
```

```
case 2:
      h=h+dg;
      dg=0;

      if(h>11)
{
       h=11;
}
    if(h<0)
{
    h=0;
}
   break;
```

```
case 3:
  if(dg==1){
 a=1;
 dg=0;}
  if(dg==-1){
 a=0;
 dg=0;}
 break;
   }
if(s>59){
 s=0;
 m++;

if(m>59){
 m=0;
 h++;

if(h>11){
 h=0;          //12 Hour Format
 a=!a;
}
}
}
```

Hour , Minutes, Hour Digits Printing:

```
d=h%10;
numberprinter(d, 4);
d=h/10;
numberprinter(d, 3);

d=m%10;
numberprinter(d, 7);
d=m/10;
numberprinter(d, 6);


d=s%10;
numberprinter(d, 10);
d=s/10;
numberprinter(d, 9);

setCursor(14, 0);
if(a){
    print("AM");

      }
      else{
    print("PM");

      }
```

```
if(cnt==0)
{

        s++;
        setCursor(5, 0);
        print(" ");
       setCursor(8, 0);
       print(" ");

    HAL_Delay(1000);
    setCursor(5, 0);
    print(":");
    setCursor(8, 0);
    print(":");

     }
```

GPIO Input & Output pins

```
static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_2|GPIO_PIN_1|GPIO_PIN_3|GPIO_PIN_4
                    |GPIO_PIN_5|GPIO_PIN_6|GPIO_PIN_7, GPIO_PIN_RESET);

  /*Configure GPIO pins : PC1 PC2 PC3 */
  GPIO_InitStruct.Pin = GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);


  /*Configure GPIO pins : PB2 PB3 PB4 PB5
                         PB6 PB7 */
  GPIO_InitStruct.Pin =
GPIO_PIN_2|GPIO_PIN_1|GPIO_PIN_3|GPIO_PIN_4|GPIO_PIN_5
                    |GPIO_PIN_6|GPIO_PIN_7;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_OD;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

}
```

DIFICULTIES

Some Difficulties That We Faced While Implementing This Project Are,

- ❑ There is no direct library for Liquid Crystal in ARM Keil IDE like we have in ARDUINO.
- ❑ Writing Bangla in display was another problem we had to solve.

We have solved these following issues,

- ✔ Library for LCD interfacing with STM32 (LiquidCrystal.c, LiquidCrystal.h) is collected from GitHub(1) and included in keil IDE.
- ✔ To solve the second problem we have taken ideas from 'create character function' of ARDUINO's Liquid Crystal library. The link are attached bellow (2).

References: 1. https://github.com/SayidHosseini/STM32LiquidCrystal

2. https://maxpromer.github.io/LCD-Character-Creator/