EE304 Embedded Systems

Oven Project Progress Report

Bahadır AKDEMİR

Kerem COŞKUN

03.06.2022

Design Plan

The development of technology continues exponentially, and it is important to "smart" the devices that people use nowadays. In this context, we have created a smart oven design to increase usability. In addition to ordinary ovens, this oven can be remotely controlled via Bluetooth. In addition, the oven can send notifications to the user about the cooking processes. If the oven is not used for a long time, the deep sleep mode is activated.

In order to perform these operations, two heater plates, fan, temperature sensor, flame sensor, LCD display, Bluetooth module and many other suppliers are used together with the stm32f103c6 microprocessor. The reason why we prefer this microprocessor is that it is affordable in terms of performance and contains enough pins for our needs.

We have developed a simple mobile application to send data via Bluetooth connection. Thanks to this application, users can start the oven remotely. For example, you are hungry at night and you are sitting in the living room. When you are hungry, you can start the oven without getting up and go to the kitchen to get your food when the buzzer sounds. Enjoy your meal.

Contents

DESIGN

Timers

- Check Heating Countdown
- Buzzer controller

GPIO

- Led Outputs
  is Door open
  Mode
- Setting Buttons

RTC

- Display current time

Sensors

- Temperature sensor
- Flame sensor
- HC-06

Actuators

- Resistance
- Fan
- Buzzer

Features

- Switching modes
- IoT

ADC

- Temperature
  LM35

Smart Algorithm

- If the oven is not used for a long time switch to the deep sleep mode
- Remote Access to the Oven via Bluetooth App.

Communication

- Bluetooth
- USART

Summary Table

Module/Feature	Some of the possible types	

GPIO	• ⭐Digital Output
	• ⭐Digital Input
	• ⭐Other
		• Digital Output
		  ○ Push/Pull x10
		• Digital Input
		  ○ Pull Down x3

Communication	• UART ⭐,
	• SPI ⭐⭐,
	• I2C⭐⭐,
	• CAN⭐⭐⭐, Others
	• Using multiple devices at the same communication bus ⭐⭐⭐
		• UART
		  ○ For Bluetooth communication
		  ○ Full-Duplex
		  ○ Interrupt

Watchdog timer	⭐	
Interactivity (Leds, buttons, switches, touch etc.)	⭐⭐	• Leds x3 • 16x2 Display • Relay Switch x2 • And Gates x4 • Or Gate

Using sensors	Single ⭐, few ⭐⭐, many or advanced one ⭐⭐⭐⭐	• Sensor types
		  ○ LM35 (Temp Sensor)
		  ■ Interrupt
		  ○ HC-06 (Bluetooth Sensor)
		  ■ Interrupt
		  ○ Flame Sensor

Actuators	• Motors, • ..	• Fan • Heaters • Buzzer
Timers	Systick⭐, Advanced-basic Timers ⭐⭐, RTC alarm ⭐⭐	• RTC • Timers

		○ Timer 2 (for buzzer) ■ Pwm ■ Dma ○ Timer 3 (countdown duration) ■ Interrupt
Usage of polling	❌❌❌	We have only used one polling: • LCD Display
Usage of Interrupts	No interrupt ❌❌❌, Single ⭐, few ⭐⭐, many with different priorities ⭐⭐⭐	• ADC1_read_IT (LM35) • External_It (button control) x3 • RTC_IT (Getting Real Time) • HAL_TIM_…_IT (Timer2) • Uart_Receive_IT
Error handling	No error handling ❌ , few ⭐⭐, full ⭐⭐⭐	• RTC_IT • HAL_TIM_Base_Start_IT • HAL_Uart_Receive_IT • HAL_ADC_Start_IT • HAL_TIM_STOP_IT • HAL_TIM_PWM_Init
Analog-digital Converter	ADC ⭐⭐, DAC⭐⭐	• ADC1
Advanced Things that no code is provided during the course such as DAC, CAN etc.	extra ⭐⭐⭐	
Power saving	Sleep - standby - wakeup ⭐⭐⭐	• Sleep Mode
DMA	⭐⭐⭐	• DMA ○ Used in timer 2 (buzzer)
Ethernet-internet-wi	⭐⭐⭐	

fi		
Writing own driver library for a peripheral	⭐⭐⭐⭐⭐	

bluetooth	⭐⭐⭐⭐	• Bluetooth
		  ◦ Uart
		  ◦ Used for Stm32 and app comm.

PCB	External electronics design ⭐⭐, using a different board ⭐⭐⭐, using MCU unit on your own design without the development board ⭐⭐⭐⭐⭐	
Usage of advanced tools e.g., Matlab, CubeAI etc. (Matlab code should run on MCU)	⭐⭐⭐⭐⭐	
Real time OS	⭐⭐⭐⭐⭐	
IoT	Making it work with: • Node-red, Blynk etc. • Mobile device interaction Time series database (InfluxDB)	Mobile Device Interraction via Ble app

Explanation of the table:

- GPIO
  - Outputs
    - We have used 9 GPIO outputs with Push/Pull type.
    - 6 of them are used for the LCD display. (lcd_txt.c lines 11-18)

        - 1 of them is used for stoping buzzer after 3 seconds
        - rest of them are used for turning on the LEDs. (Mode LEDs and start led)
    - Inputs
        - 3 of them are used for buttons with Pull Down mode. (Start_button, Mode_button, and Temp_button).
- Interactivity
    - Leds x3
        - Mode LEDs and Start led (LED_RED)
    - 16x2 Display
        - We have used LM016L for the LCD display. We chose LM016L due to its well-documentation.
        - Driver files lcd_txt.c and lcd_txt.h added to the project.

    - Relay Switch x2
        - Relay switches are used to control the start and stop conditions of heaters

- Sensors
    - LM35 (Temperature Sensor)
        - We used a temperature sensor to keep the internal temperature of the oven at the temperature desired by the user. When the indoor temperature is the temperature desired by the user, the heaters are turned off and when the temperature drops 10 degrees from the temperature desired by the user, the heaters are turned on again.
        - Code:

```
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc)
    if(raw_temp_value>user_temp_value){
     HAL_GPIO_WritePin(GPIOA,GPIO_PIN_9, GPIO_PIN_RESET);
     HAL_GPIO_WritePin(GPIOA,GPIO_PIN_10, GPIO_PIN_RESET);
    }
    else if(raw_temp_value-15<user_temp_value){

      GPIO_PinState bit1_value,bit2_value;
      bit1_value = (current_number & (uint8_t)1) ? GPIO_PIN_SET:GPIO_PIN_RESET;
      bit2_value = (current_number & (uint8_t)2) ? GPIO_PIN_SET:GPIO_PIN_RESET;
      bit1_GPIO_Port->BSRR = (bit1_value ? bit1_Pin : bit1_Pin << 16u) | (bit2_value ? bit2_Pin : bit2_Pin
<< 16u);
    }
```

(main.c line 102-127)

    - Flame Sensor
        - We used a flame sensor to detect any flame that occurred in the oven. When a flame is detected, the buzzer raises an alarm sound.
    - HC-06 (Bluetooth Sensor)

        - We used the Bluetooth hc-06 module to remotely control the oven. Bluetooth data comes as 8 bits and the first 3 digits represent temperature, the second 3 digits represent duration, and the next bits represent oven mode and start command, respectively.
        - Bluetooth module is connected to the uart channels of the stm32 board.

- Actuators
    - Fan
        - Ovens have modes that use the fan while they are working, apart from this, it is also necessary to use a fan to cool the oven so that the heat does not damage the system after the oven is turned off. That's why we used a fan.
    - Heaters
        - We used two heaters in our oven project. they are placed at the top and bottom. The heaters can work individually or together according to the chosen mode.
    - Buzzer
        - We used a buzzer in our project to inform the user when the food is cooked. In addition, the buzzer gives an audible warning when there is a flame in the oven.

- Timers
    - RTC
        - We used the RTC timer to show the time. The time appears always-on on the LCD.
        - We used RTC as an interrupt.
        - Code:

```
void HAL_RTCEx_RTCEventCallback(RTC_HandleTypeDef *hrtc){
    RTC_TimeTypeDef tmpTime;
    …
    … }
```

(main.c line 673-684)

    - Timer 2
        - Pwm
        - Dma
        - Timer 2 used with pwm and dma features. makes the buzzer work
    - Timer 3
        - Internal Clock
        - Set to the 1 second tick. Prescaler 124 and Counter Period 63999
        - We set timer 3 to be 1 second to know how long the food has been cooking

- Usage of Polling
  - LCD display
    - We tried to make the LCD display with I2C Communication, but it didn't work. That's why we have used polling for the LCD display.
    - Code:

```
if(isStart==1){
      lcd_puts(0,0,(int8_t*)str);
  }
.
.
. (main.c lines 205-239)
```

- Usage of Interrupts
  - ADC1_read_IT (LM35)
  - External_It (button control) x3
    - We used interrupt to check if the buttons (set_time, start and mode button) are pressed.
    - Code:

```
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin) (main.c lines 603-630)
```

  - RTC_IT
  - HAL_TIM_…_IT (Timer2)
  - Uart_Receive_IT

- Error handling
  - RTC error handling
    - We check if RTC_GetTime function works properly or not
    - Code:

```
if (HAL_OK != HAL_RTC_GetTime(hrtc,&tmpTime,RTC_FORMAT_BIN)){
    Error_Handler();
  }
```

  - HAL_TIM_Base_Start_IT
    - Code:
      -
  - HAL_Uart_Receive_IT
    - Code:

```
if (HAL_UART_Receive_IT(&huart1,(uint8_t*)buffrc,8) != HAL_OK)
{
        Error_Handler();
}
```

    - HAL_ADC_Start_IT
    - HAL_TIM_STOP_IT
    - HAL_TIM_PWM_Init

- ADC
    - ADC1 for LM35
        - 4Mhz, 55.5 cycle sampling time = 17 µs. 17 µs is requirement sampling time for temperature sensor.
        - Code:

          HAL_ADC_Start_IT(&hadc1); initiliaze ADC. (main.c)

          void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc) (main.c line 102)

- Power Saving
    - Sleep Mode
        - If no button is pressed for 4 hours, the oven will switch itself to sleep mode. CPU CLK shuts down when it goes into sleep mode. Also, the LCD screen turns off. The system wakes up when any interrupt runs.
        - Code:

```
if(isStart==0 && timer2==5){
        HAL_SuspendTick();
        HAL_PWR_EnterSLEEPMode(PWR_MAINREGULATOR_ON,
         PWR_SLEEPENTRY_WFI);
        HAL_ResumeTick();
 }
```

- DMA
    - Used in Timer 2 (Expalined at the above)

- IoT
    - Our Oven can interact with any Bluetooth device. We developed an app and using our app, users can send duration, temperature, and mode information to the oven.

Features

- The oven has 3 buttons (start button, setting temperature button and heaters mod button).
    - Temperature cannot be changed if start button is active.
- Heaters work according to the setting that is prepared before starting (if it is started, mod cannot be changed)

Code:

```
if(GPIO_Pin==GPIO_PIN_11){
if(isStart==0){
        …
        …
        bit1_value = (current_number & (uint8_t)1) ? GPIO_PIN_SET:GPIO_PIN_RESET;
        bit2_value = (current_number & (uint8_t)2) ? GPIO_PIN_SET:GPIO_PIN_RESET;
        bit1_GPIO_Port->BSRR = (bit1_value ? bit1_Pin : bit1_Pin << 16u) | (bit2_value ? bit2_Pin : bit2_Pin <<
16u);
    }
}
```

Lines: 606- 628

- If inside of the oven is being hotter than expected, heaters will stop automatically. If it is being cooler than expected, then heaters will run again until providing the temperature set. (Code is above)
- Display shows the current time in default. After clicking the mode button, it shows the setting of the temperature. After starting, it shows the duration left (currently not working for this part of the project).

Code:

```
void HAL_RTCEx_RTCEventCallback(RTC_HandleTypeDef *hrtc){
    if (HAL_OK != HAL_RTC_GetTime(hrtc,&tmpTime,RTC_FORMAT_BIN)){
        Error_Handler();
    }
    sec = tmpTime.Seconds;
    min = tmpTime.Minutes;
    hour = tmpTime.Hours;
}
```

Lines: 673 - 684

- Fan is running when cooking starts (It is handled with circuit elements. The code is the same with the mode button and start button).
- With the bluetooth connection made via UART communication, the user can operate the oven by sending 8-character information to the board with any bluetooth device (for example, a smart phone). 1-3 characters represent the temperature of the oven, 4-6 characters indicate

the time to pass. The 7th character indicates which resistor will work. Finally, if the 8th character is "s", the oven works. Otherwise, all data must be re-entered.

```
if(buffrc[7]=='s' && isStart==0){
	user_temp_value = (((int)buffrc[0])-48)*100 + (((int)buffrc[1])-48)*10 + (((int)buffrc[2])-48);
	user_duration_value = (((int)buffrc[3])-48)*100 + (((int)buffrc[4])-48)*10 + (((int)buffrc[5])-48);
	HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_12);
	timer=0;
	isStart = 1;
	if(buffrc[6]=='1'){
			current_number=1;
	}else if(buffrc[6]=='2'){
			current_number=2;
	}else{
			current_number=3;
	}

	GPIO_PinState bit1_value,bit2_value;
	bit1_value = (current_number & (uint8_t)1) ? GPIO_PIN_SET:GPIO_PIN_RESET;
	bit2_value = (current_number & (uint8_t)2) ? GPIO_PIN_SET:GPIO_PIN_RESET;
	bit1_GPIO_Port->BSRR = (bit1_value ? bit1_Pin : bit1_Pin << 16u) | (bit2_value ? bit2_Pin : bit2_Pin << 16u);
	}
	timer2=0;
```

Lines: 694 - 717

- When the cooking process is finished, the buzzer will run for 3 seconds and stop.

```
	if(isStart==1){
		if(timer==user_duration_value || timer==(user_duration_value+3)){
			if(timer==(user_duration_value+3)){
				HAL_GPIO_WritePin(GPIOB,GPIO_PIN_4, GPIO_PIN_RESET);
				isStart=0;
				isClicked=0;
				timer2=0;
			}
			else{
				HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_12);
				timer2=0;
				HAL_GPIO_WritePin(GPIOB,GPIO_PIN_4, GPIO_PIN_SET);
				HAL_TIM_PWM_Start_DMA(&htim2,TIM_CHANNEL_2, (uint32_t *)(5),(uint16_t)sizeof(5));
				}
			}
	}
	else{
		timer=0;
	}
```

Lines: 316 - 337 (stm32f1xx_it.c)

- If the output from the Flame Sensor is positive (it means there is a fire), the buzzer will sound until the negative output is received.
- With the power saving mode, if no command (such as bluetooth message or button) is given within 4 hours, the system will enter power saving mode, the screen will turn off until the next interrupt

Challenges

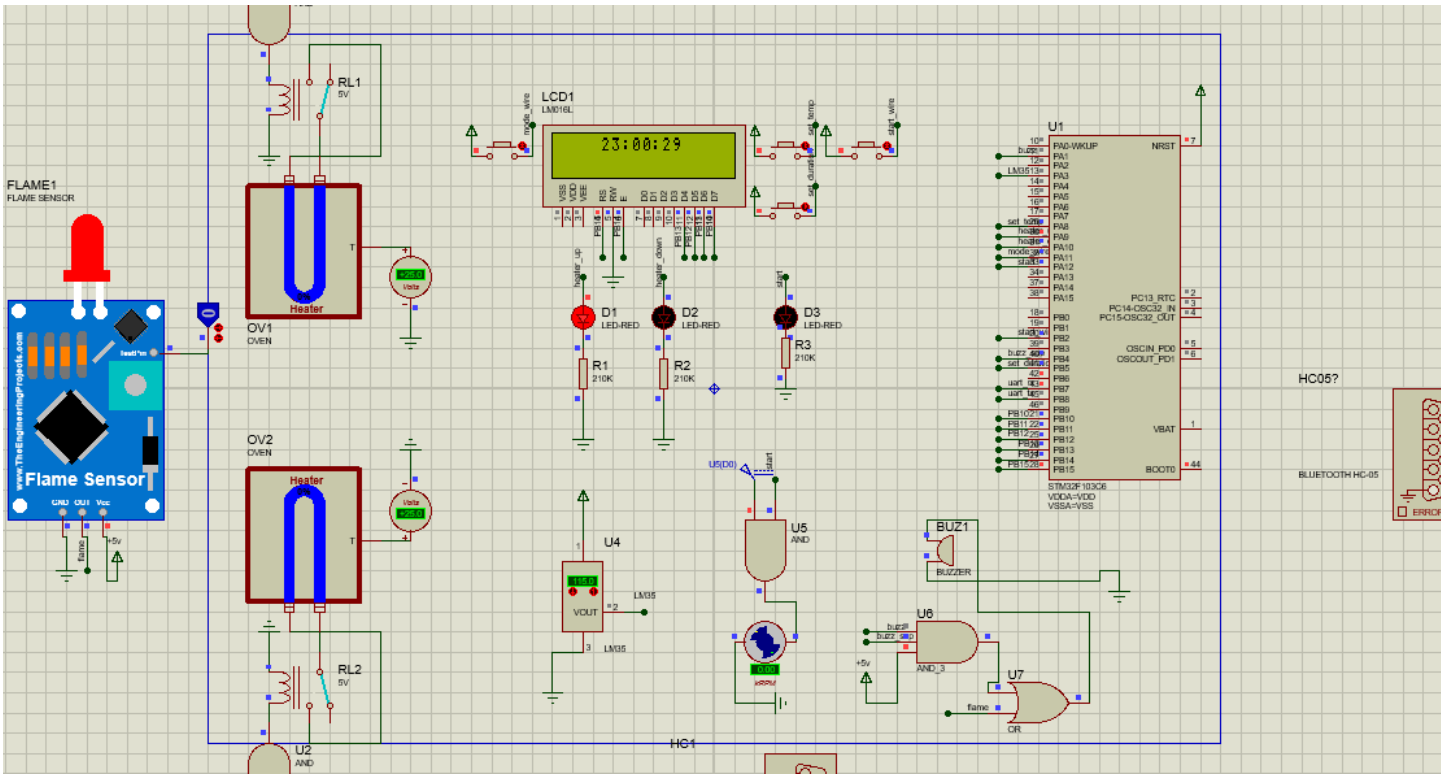
One of the most important problems that prevent our progress in the project occurred while connecting the display (lm016l). At first, when trying to connect Display with I2C, no text appeared on the screen. Later, we realized that this problem was actually caused by the pins we used. When we changed the pins, we got rid of the problem.

Another problem experienced occurred when using relay switches. We had to use a relay switch when adding the heaters and it was a circuit element that we did not know before. Thanks to various videos and blogs on the Internet, we learned for what purpose and how it was used, and we successfully added it to the circuit.

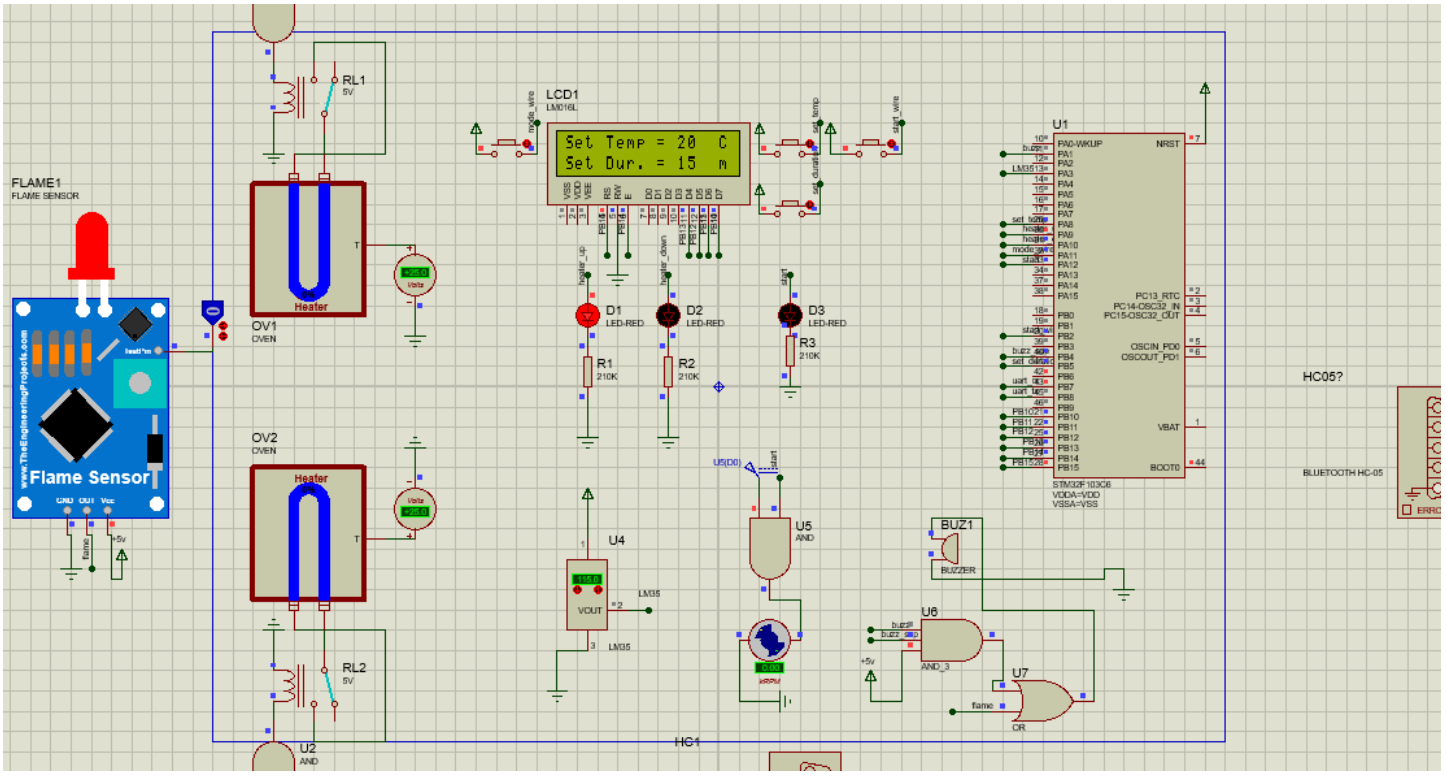
The next challenge is to add the LM35 to the circuit. We thought that we would measure the temperature from the simulation and learned that we could use lm35 to show that our algorithm is working. Then, we tried so many libraries and codes to run lm35. After spending most of our time, we realized that it can be used in the same way as the ADC potentiometer we used in the lab lectures. Afterward, we added it to the circuit comfortably.

The last but not least challenge was about bluetotth connection.When we made the Bluetooth connection, we could easily display it on the computer, but when trying to change it with a smartphone, we encountered various problems. Rather than the application we made with the Flutter framework, the application created from the platform called MIT Inventor App worked without any errors.

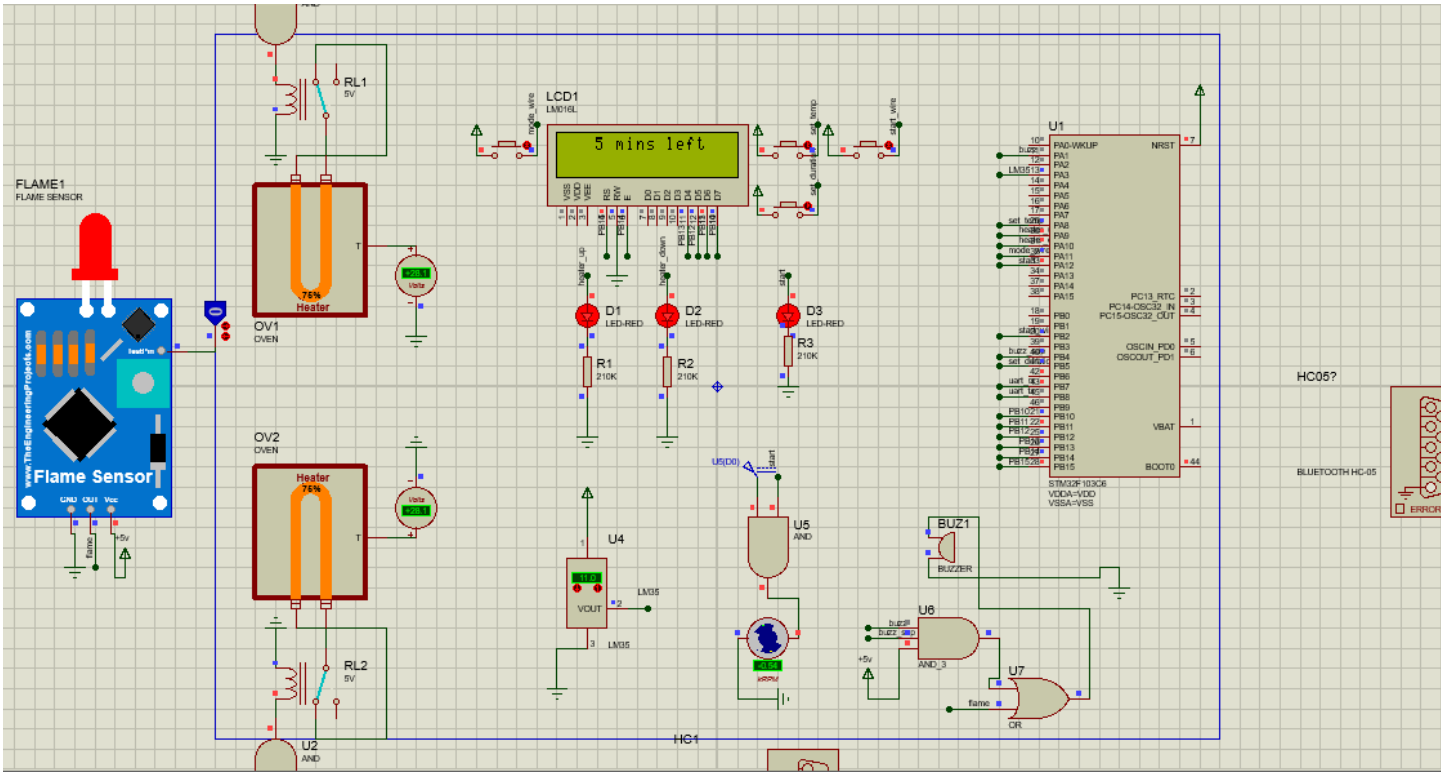
Screenshots:



(Current World Time)



(Setting Temperature and Duration)



(Heaters, Fan and Lm35 are working)

Price List

Product	Price
Lm35	**26.63₺**
LM016L	**45.61₺**
STM32F103C6	**90.24₺**
LEDs	**5.82₺**
Buttons	**21₺**
Heaters	**856,68₺**
Fan	**173.07₺**
HC-06	**69.99₺**

Flame Sensor	**10.89₺**
Buzzer	**6.60₺**
Total	**1306.53₺**