Fire Tower

6조

201220911 최재웅

201220954 김성훈

201220896 조 현

201221022 한강희

**INDEX**

1. **All Scenarios**
2. **Before the fire occurred**
3. **After the fire occurred**
4. **Board**
5. **Arduino**
6. **Sensor**
7. **The operator principle**
8. **Web serve**

**1)Node.js**

**2)MySQL**

1. **Android(Application)**

**1)Activity**

**2)Communicating with the server**

1. **Making Tower**
2. **Problems**
3. **All Scenarios**
4. **Before the fire occurred**

**Before the fire occurred very peaceful state is maintained. The Fire Tower continues to rotate to detect if a fire has occurred. Arduino send the data detected by the sensor to Board. Looking at the components of the board. First OLED’s represents a photograph of a peaceful forest. Second text LCD represents “Guarding!”. Third the number of LED’s is determined based on the information received from the water level sensor. Fourth matrix LED show about flame and CO measured by sensor. Left of matrix LED is indicates the amount of flame detection and Right represents the amount of CO. Finally Touch LCD shows login screen which is initial screen**

1. **After the fire occurred**

**If a fire occurs, The state is in an emergency. Flame and CO sensor of the Fire Tower detecting the occurrence of fire. Arduino send this information to the board. Then board is very busy. At this time, The board sends the current time and water level information to the Web server and holds the information. Components of board operate simultaneously through the thread. Let’s look at the components from above. First text LCD represents “Fire!!”. Second OLED’s represents a photograph of a fire image. Third, Since the fire itself has a high CO or Flame measurement value. Therefore height of matrix LED will be high. Fourth, buzzer sound of a fire truck. Finally, full color LED alternates red and blue as the buzzer sounds The manager hears the sound and approaches the board. Next from the initial screen, login through the keypad to authenticate as an administrator and gain the right to control the board. The administrator can view the camera information of Fire Tower and send information to Arduino to control the rotating Tower. The administrator can control the servo motor through the c button and control the water pump by pressing the launch button.**

1. **Board**

**The code structure for implementing the above scenario is as follows.**   
**In the main, we first open the drivers for each component and handle the error.** **When the program is executed, the touch screen, etc\_periperal, and communication thread are activated.**

**In a touch screen thread, there are two parts, one representing the screen and the other reading and controlling the input value when the user touches it.** **The reason for dividing it into two parts is that if you do not touch it, the function reading from the driver file stops, so the thread itself stops and the camera is not output in real time.   
 The communication thread is responsible for communicating with Arduino and the server. Arduino receives fire, CO, and water level information and outputs it to the LED and dot matrix and sends date and water level information to the server. Also It sends the values ​​taken from the dip switches, the buttons on the touch LCD to control the Arduino. If the fire or CO value exceeds a certain value, set the Situation variable to 1 to indicate that a fire has occurred.**

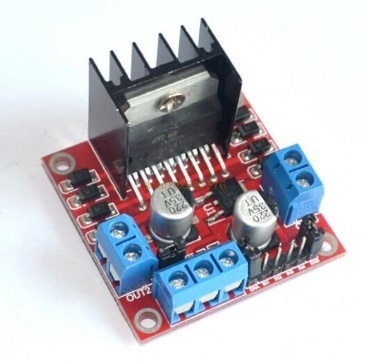
**In the etc\_periperal thread, there are two parts when the situation occurs and when it ends. When the situation occurs, the text LCD displays "fire!" And turns the OLED into a fire situation.   
In addition, in 7 segments, the time after the fire is updated every second in units of hour and minute. It can be expressed in 7 segments of 99 hours 59 minutes 59 seconds. If the situation is over, replace the text LCD and OLED initially, then turn off the 7 segment.**

1. **Arduino**
2. **Sensor**
   1. **DWP\_385(Water pump)**

****

DWP\_385 draw water from a water tank and fire it.

* 1. **L298N(Motor driver)**

****

L298 receives information from Arduino and controls DWP-385

* 1. **V2(Flame sensor)**

****

Detect the fire and let it know if the fire occurred.

* 1. **MQ-7(CO sensor)**

****

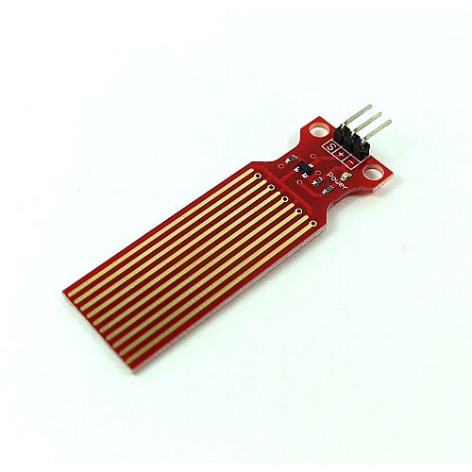
Detect CO concentration in air and send information to Arduino

* 1. **SG-5010 (Servo motor)**

****

Three are used, two are used to rotate tower 360 degrees, and the other is used to control the height of the barrel

* 1. **Water level Sensor**



Check the water level in the water tank and we can see how much water remains

1. **The operator principle**

**Two servo motors are used to allow the tower to rotate 360 degrees. (One servo motor has 180 degrees) We can know the current angle through ServoMotor.read(), so initialize the angle of the servo motors to zero to place the tower in the first position. Since the tower is set in the initial direction, turn one of servo motors 0-180 degrees. Next time Rotate the other motor to 0-180. This can be done up to 0-360. To return to the initial value, turn the servo motor that has just been turned to 180-0 and set the first servo motor to 180-0. When the tower rotations, it continuously updates the values of the flame sensor and the CO sensor, and sends this value to the board. If the board wants to control the Arduino at the beginning of loop(), control passes to board. Board send 1byte(0-255) to Arduino.**

**1 byte(receive)**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
| **Mode** | **Left,right** | **Up,down** | **Up** | **down** | **left** | **right** | **Shoot** |

**Mode=1**

**If (receive&0x80) Mode has ‘true’, so Board gain control motors of Arduino.**

**If (receive&0x30)==0x10) Raise the angle of the servo motor moving the barrel.**

**If (receive&0x28)==0x08) Down the angle of the servo motor moving the barrel.**

**If (receive&0x44)==0x04) If possible, turn the tower to the left.**

**If (receive&0x42)==0x02) If possible, turn the tower to the right.**

**If (receive&0x01) Press and hold the shoot button on the board to activate the water pump, then release it to stop the action.**

1. **Web Server**

**1)Node.js**

**The web server receives data from the board when a fire occurs or when a fire is extinguished. The data is transmitted via the POST method, and the web server parses the data to store the date, time, and the remaining amount of water in the DB. The web server also creates a SELECT query in the database when a GET request is made using an id query in the URL.** **After receiving data from DB using Query, it sends data to client.**

**2)MySQL**

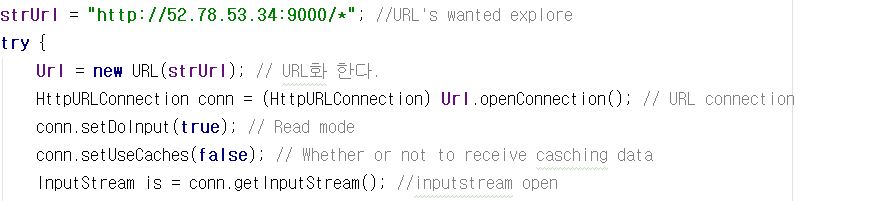
**Before running the Node server, create the user account and database specified in the JS code file. The user id used in this project is root and the database is fire. The attribute of the database used in MySQL is an Integer type Id, a Date of type string, and water of type Integer.** **Id is set as primary key and automatically incremented.**

1. **Android**
2. **Activity**

**The initial activity is login screen. Make sure the administrator correct. Click the “login” button on the screen to move to the next activity, this activity has information button. Click the button, we receives data from the server. The data includes time, water level information.**

1. **Communicating with the server**

**Android uses “HttpURLConnection” connect to web server. When the android accesses the URL(**[http://52.78.53.34:9000/\*](http://52.78.53.34:9000/*)**), the server sends android the information about MySQL. Since data is in JSON format, so it was parsed by android and displayed in activity.**



**The above code shows the part that connects to the server.**

**6. Making Tower**

**1)Design model and Factors to Consider.**

**To design the fire tower we reference ‘the missile turret’ in the ‘Starcraft’ game. The most important thing in making tower is tower to rotate 360 degree stably. And we must consider how to shoot the water from the water pump, how much is the weight would be and the risk which wires can be twistable.**

**2)Structures of the tower and Process of Making**

**The reason of the tower must rotate is to sensing the fires in any directions. And if we sensed the fire is occurred we have to change the tower’s directions to the fire. So we put the water pump, fire sensor, CO sensor, and Camera to tower’s top part and make it to rotate. Inside of the top part we using one servo motor to control the barrel of the tower. And we place second servo motor to center of the tower’s lower part and attached it to the middle connection part. Because of our servo motor just rotate 180 degree, we using two motors and rotate it separately. So attached third motor between middle connection part and tower’s lower part. Now except the water level sensor, almost everything placed vertically above the tower’s bottom part. To withstand the heavy weight, tower’s bottom part must be sturdy. So we choose materials as fomex board and cutted it 16 by 16 size squares. At the bottom part, we split it into two sections and placed Arduino, bread board to the upper section and water tank to lower section. Finally, square shaped fire tower is made and holes are drilled in each floor so that the wires of the whole system are connected from top part to the bottom part.**

1. **Problems**
2. **Difficult**

**The first is the line problem. The tower keeps turning, so the lines of Arduino can be twisted. Therefore we had difficulty in constructing the structure without turning the servo motor 360 degree. So we made tower to turning 0 to 360 degree and then back to 360-0. Eventually solved problem. The second problem is communication problem with android and web. We did a lot of searching, but communication did not work properly. After all wander, we have solved using thread. Perhaps main thread need to configure the activity, but it seems that took a long time to connect to the server, so continued to fail. Third problem is ‘Segmentation fault’ error. In testing the entire system, our module devices had to operate sequentially depending on main system’s situation. But some module devices abruptly generate the segmentation fault errors and it is really hard to know what is wrong. So, if we fix some codes and retested it then another error was generated different location. The first reason of this error what we found is deadlock situation. In operating 7-segment device with other peri devices, peri devices lock the situation mutex and waiting to 7-segment devices to end. But 7-segment devices were waiting too, to using mutex. So as a result of these endless waiting, deadlock was generated. The other difficulty was in communication process, we had to set delay to match the sync. And image size of the bitmap header was different from real data’s size so our bitmap image was printed obliquely. There were some garbage values when read the data so to remove this we divided data’s size into height and subtract width value. Now we know garbage value’s size so we subtract these values from data and finally get real data information. The last difficulty is output a bitmap image with the camera screen. At the front buffer, camera screen and the process of printing bitmap images were conflicted. So we draw bitmap image at the back buffer first, copied it to the front buffer.**

1. **Failed to implement**

**The initial idea contained more information on Android, and the administrator had control tower and directly shooting the water. However camera screen must be delivered in sequence(Board->Web Server->Android->Web Server->board->Arduino). We thought that it was difficult to handle it in real time. Also we do not have any team members who have experience with an Android and Web server, so we could not implement it.**