Real Time Group

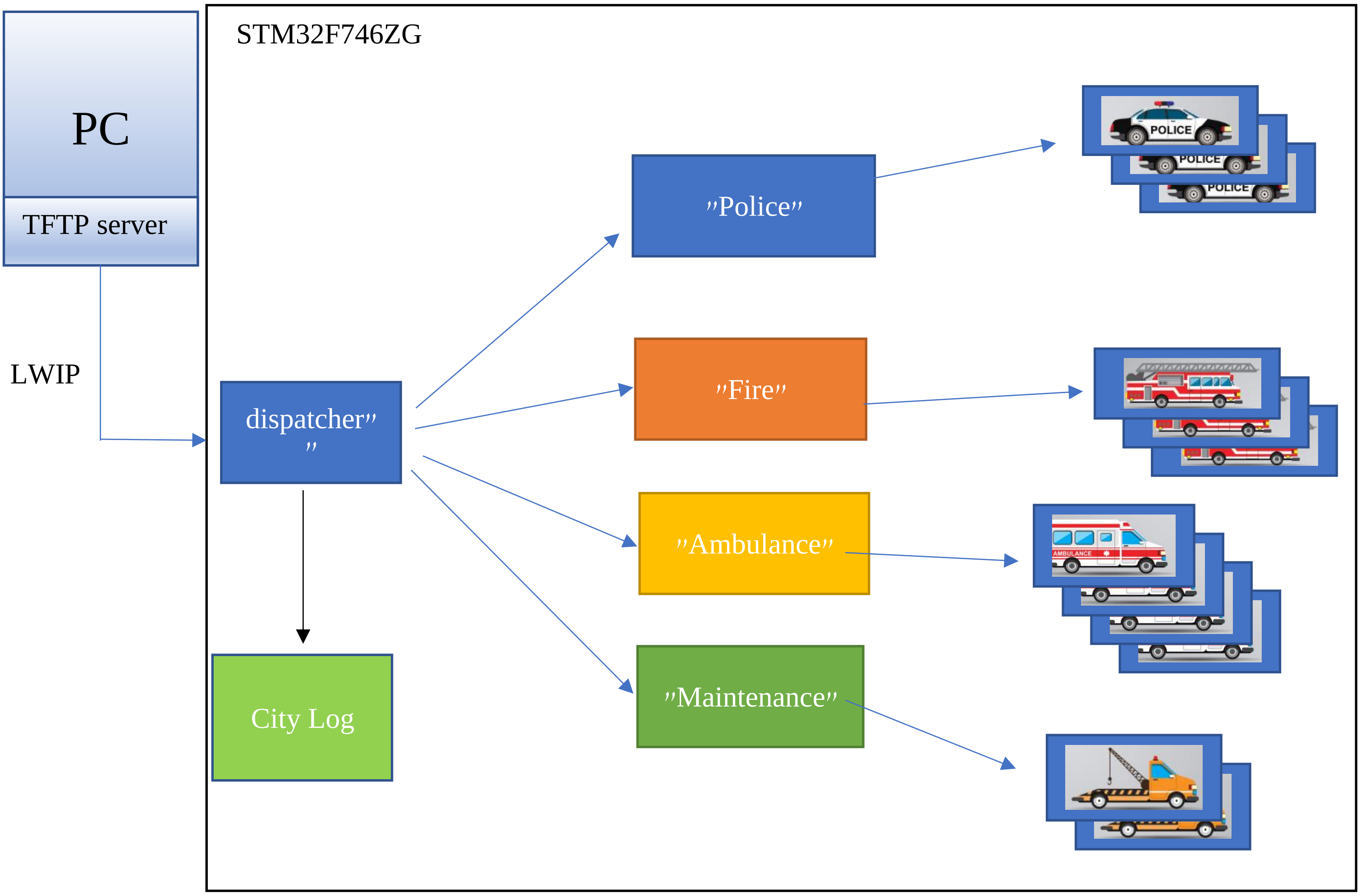
RT Embedded Linux Solutions

FreeRTOS SE Final Project

Project: City dispatch simulation

You need to simulate a "City Dispatch" unit in which:

1. A TFTP Server should hold the events as files (you are free to choose file names), you may install a TFTP Server based on Windows or Linux Machines (at the end of this project there is a tutorial for Installing TFTP Server on Linux based Machines).
2. You should implement a TFTP Client on the EVB (using the LWIP) which reads (or downloads) events as images into the centralized city "dispatcher Memory" (EVB Memory) from the TFTP Server and save them in some sort of job-queue Data Base.
3. Each event is 100 bytes in size and has the following structure:
    a. Handling department – 1 byte
    b. Event severity: – 1 byte
        i. Critical
        ii. Warning
        iii. Notice
    c. Event Address – 80 bytes
    d. Padding – 18 bytes (all 0xffff).



4. Choose the best method for saving the events in a job-queue Data Base which will allow easy access the them.
5. The events should be downloaded one by one from the TFTP server in 100 millisecond intervals.
6. Events should be sent to the appropriate department based on their "department field".
7. Severity should take precedence, events should be sent based on their "severity", for example if the in-memory job-queue holds 2 events one with a Critical severity and the other with a Warning, the event with Critical severity should be sent first.
8. Please keep track of how many events are sent (per each department) and print (write to UART) them if asked to the UART debugger.

9. The following elaborates regarding departments behavior:
    a. Each department should be running a task that will receive the incoming event into the department's message queue.
    b. Each department will have a thread pool (3 tasks are enough) to handle the events.
    c. One of the tasks from the department's thread pool will read the event from the message queue and implement it (just print it to the Debugger's UART and busy wait for 300 msec).
    d. When tasks are not running the should be put to sleep (blocking mode).
10. Create another task to indicate that your STM32 device is active by toggling any led as a heartbeat.
11. Implement test cases to insure correctness.
12. **BONUS** - Using the "+" and "-" keys you should be able to change the frequency of downloading events from TFTP, 10 millisecond for every click.

Installing and Configuring the TFTP Server on Linux

1. If you choose to use a Linux based TFTP server then read the below "How To" (You may use Windows based TFTP Server if you'd like).
2. Installing TFTP Server on a Ubuntu Machine:
    a. ***sudo apt update***
    b. ***sudo apt install tftpd-hpa***
    c. ***sudo systemctl status tftpd-hpa***

3. Configuring TFTP Server:
   sudo nano /etc/default/tftpd-hpa

   The configuration file should be opened for editing. This is the default configuration of the TFTP server.

Here, **TFTP_USERNAME** is set to tftp. It means the TFTP server will run as the user tftp.
TFTP_DIRECTORY is set to /var/lib/tftpboot. It means /var/lib/tftpboot is the directory on this server which you will be able to accessing via TFTP.
TFTP_ADDRESS is set to :69. It means TFTP will run on port 69.
TFTP_OPTIONS is set to –secure. This variable sets the TFTP options. There are many options that you can use to configure how the TFTP server will behave. I will talk about some of them later. The –secure option means change the TFTP directory to what is set on the **TFTP_DIRECTORY** variable when you connect to the TFTP server automatically. This is a security feature. If you hadn’t set the –secure option, then you would have to connect to the TFTP server and set the directory manually. Which is a lot of hassle and very insecure.

Example:

```
# /etc/default/tftpd-hpa

TFTP_USERNAME=”tftp”
TFTP_DIRECTORY=”/tftp”
TFTP_ADDRESS=”:69”
TFTP_OPTIONS=”--secure --create”
```

GOOD LUCK