

# Assessment 2 component: Fire alarm call-point controller

## Summary

---

Each fire alarm call-point controller is responsible for a single wall-mounted manual call-point, which is a little button on the wall that users can press during an emergency to activate the alarm system.

When the button is pressed, the controller will send a UDP datagram to the fire alarm unit telling it to signal the alarm. It will then continue sending these datagrams in a loop perpetually to ensure that they are received.

## Safety-critical component

---

As the call points are used to manually signal a fire emergency, the call-point controller is considered a **safety-critical component** and therefore must be implemented following appropriate safety-critical software standards. Make sure to include a **block comment** near the top of your program's source code documenting and justifying any deviations or exceptions.

## Program name

---

callpoint

## Command-line arguments

---

{resend delay (in microseconds)} {shared memory path} {shared memory offset} {fire alarm unit address:port}

## Shared memory structure


---

```
struct {
    char status; // '-' for inactive, '*' for active
    pthread_mutex_t mutex;
    pthread_cond_t cond;
};
```

## Normal operation

---

The component will perform the following loop:

1. Lock the mutex
2. Check 'status'- if it is  do the following:
  - A. Send a fire emergency datagram (see the 'Datagram format' section for more detail) to the fire alarm unit
  - B. Sleep for {resend delay} microseconds
  - C. Loop back to A
3. Otherwise, wait on 'cond'
4. Loop back to 2

## Datagram format

---

The following UDP datagram format is used to notify the fire alarm unit in the event of an emergency.

## Fire emergency datagram

This datagram has the following format:

```
struct {  
    char header[4]; // {'F', 'I', 'R', 'E'}  
};
```

## Example operation

---

The program might be executed from the command-line with the following:

```
./callpoint 100000 /shm 50 127.0.0.1:5000
```

The program will `shm_open` shared memory segment at `/shm` with an offset of 50 and the size of the struct defined above, and `mmap` it into memory.

It will then lock the mutex and check the status character. If the character is anything other than `*` it will wait on the condition variable, checking the status character again when it wakes up.

If it wakes up and the status character is `*`, it will then send a 4 byte UDP datagram consisting of FIRE to 127.0.0.1 on port 5000 and then sleep for 100000 microseconds. It will then resend the FIRE datagram, sleep for another 100000 microseconds and so on.

TEQSA PRV12079 | CRICOS 00213J | ABN 83 791 724 622