CAB403 Assignment 2 Report

Reuben Visser -

Patrick Daly -

Brandon Falconer -

Brent Morgan – n10215662

# Statement of Completeness

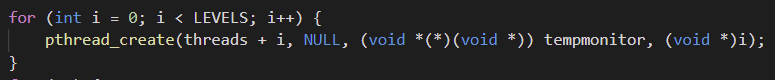
# Statement of Contribution

# Assessment of the safety-critical fire alarm system

## Analysis of the provided fire alarm

The original firealarm.c did not comply with various MISRA C rules and directives as well as other general code safety rules such as the ones in NASA’s The power of 10. This resulted in the code to be deemed unsafe and in need of a change or rewrite. To decide which is the best approach, each safety flaw needs to be analyzed to see exactly what parts of this code isn’t safely written.

**Use of Threads**Throughout the fire alarm code there is usage of thread libraries, its purpose in this code is so that each of the 5 levels in the carpark are allocated to a thread each and therefore they can then all run at the same time. However, as per MISCRA C rules and directives, no threads are allowed to be used in a safety-critical code, therefore these need to be removed/altered. A clear example of this can be seen in line 155 of the original firealarm.c where the threads for each level are created as seen below.



**Multiple Variable Declarations on the Same Line**Another important MISRA C guideline is that variables are to be declared on separate lines to achieve a safely written code. There are however parts of code in the fire alarm that conflict this rule such as the one shown below on lines 58 and 59 which declare several variables on just 2 lines.



**MISRA Directive 4.12**In the MISRA C guidelines directive 4.12 states that “Dynamic memory allocation shall not be used” (REFERENCE MISRA C), but in 5 different sections, the fire alarm code calls the malloc function which is a method of doing just that. An example of this is in line 152 where the threads pointer gets dynamically allocated memory for each level using the malloc function as seen in the snippet below.



**MISRA C Rule 11.6**Rule 11.6 of the MISRA C guidelines state that “A cast shall not be performed form pointer to void and an arithmetic type” (REFERENCE MISRA C); however, parts of the code in the fire alarm conflict this. A prime example is the one below which is a compare function (line 51) that takes an input called first in the form of a void pointer, but then uses it in the return as an integer pointer.

Text

Description automatically generated

**MISRA C Rule 17.4**Rule 17.4 states that “All exit paths from a function with non*‑void* return type shall have an explicit *return* statement with an expression” which is true for all the functions in the fire alarm except for the main. The main function should return an integer, generally 1 or 0 as an exit code so that the output can show if the code fully ran but this was not the case in this code as there is no return statements in any of the paths in main. This can be seen in figure 1 of the appendix showing the original fire alarm code.

**MISRA C Rule 18.3**Rule 18.3 States that “The +, -, += and -= operators should not be applied to an expression of pointer type” (RERENCE MISRA). This rule is not followed in several parts of the fire alarm code, an example of which is on line 155 where it tries to add the integer value of “i” onto the pointer variable threads initialised a few lines above in line 152. This code is shown below.

Text

Description automatically generated

**MISRA C Rule 21.3**Rule 21.3 of the MISRA C guidelines states that “The memory allocation and deallocation functions of <stdlib.h> shall not be used”, however in this code it is. An example can be seen below where this can be seen clearly is in line 46 inside the “deletenodes” function which calls the free function from the standard library to free the memory of the variable.

Graphical user interface, text

Description automatically generated

**MISRA C Rule 21.9**Rule 21.9 states that “The library functions *bsearch* and *qsort* of <stdlib.h> shall not be used”. This is not followed in line 87 of the fire alarm as it calls the qsort function to sort the 5 recorded temperature samples and find the median as seen in the code below.

Text

Description automatically generated

**NASA “The Power of 10” Rule 1 & MISRA Rule 15.1**Another important guide to follow to make sure your code is safety-critical is a set of 10 rules called NASA’s The power of 10. Number 1 of which states to “Avoid complex flow constructs, such as goto and recursion” (REFERENCE NASA) which is very similar to MISRA’s rule 15.1 which states “The goto statement should not be used” (REFERNCE MISRA). The fire alarm again fails this criterion by on line 159 having a “goto” call to trigger the emergency mode state of the alarm. This code is seen below.

Text

Description automatically generated

**NASA “The Power of 10” Rule 2**Number 2 of NASA’s The power of 10 rules states that “All loops must have fixed bounds. This prevents runaway code” (REFERENCE NASA). This criterion is not met in the original fire alarm as on 4 different parts of the code an infinite for loop is called to run a set of code indefinitely as seen in the code snippet below from line 157. While this is an easy way to constantly check temperatures on all the levels, it is not up to the safety standard the code needs to be.

Text

Description automatically generated

## Approach to fixing the fire alarm

Due to the large number of rule breaches when checking the safety of the original fire alarm, it was thought best to completely rewrite it. One of the main causes of this was that the provided fire alarm relied heavily on the use of threads to constantly check the temperatures on all the levels. However, as per MISCRA C guidelines the use of threads isn’t allowed, meaning this whole process would need to be redesigned. This along with the several other rule violations not just in MISCRA C but also NASA’s The Power of 10 rules meant that fixing the code to make it safety-critical would be more complicated and time consuming than just rewriting the fire alarm with the MISCRA C and NASA rules in mind.

## Potential safety-critical concerns and reservations of updated fire alarm

*safety-critical concerns and reservations of your new implementation (for example, some safety-critical software guidelines cannot be followed perfectly - list these and how you have mitigated the concerns they pose.)*