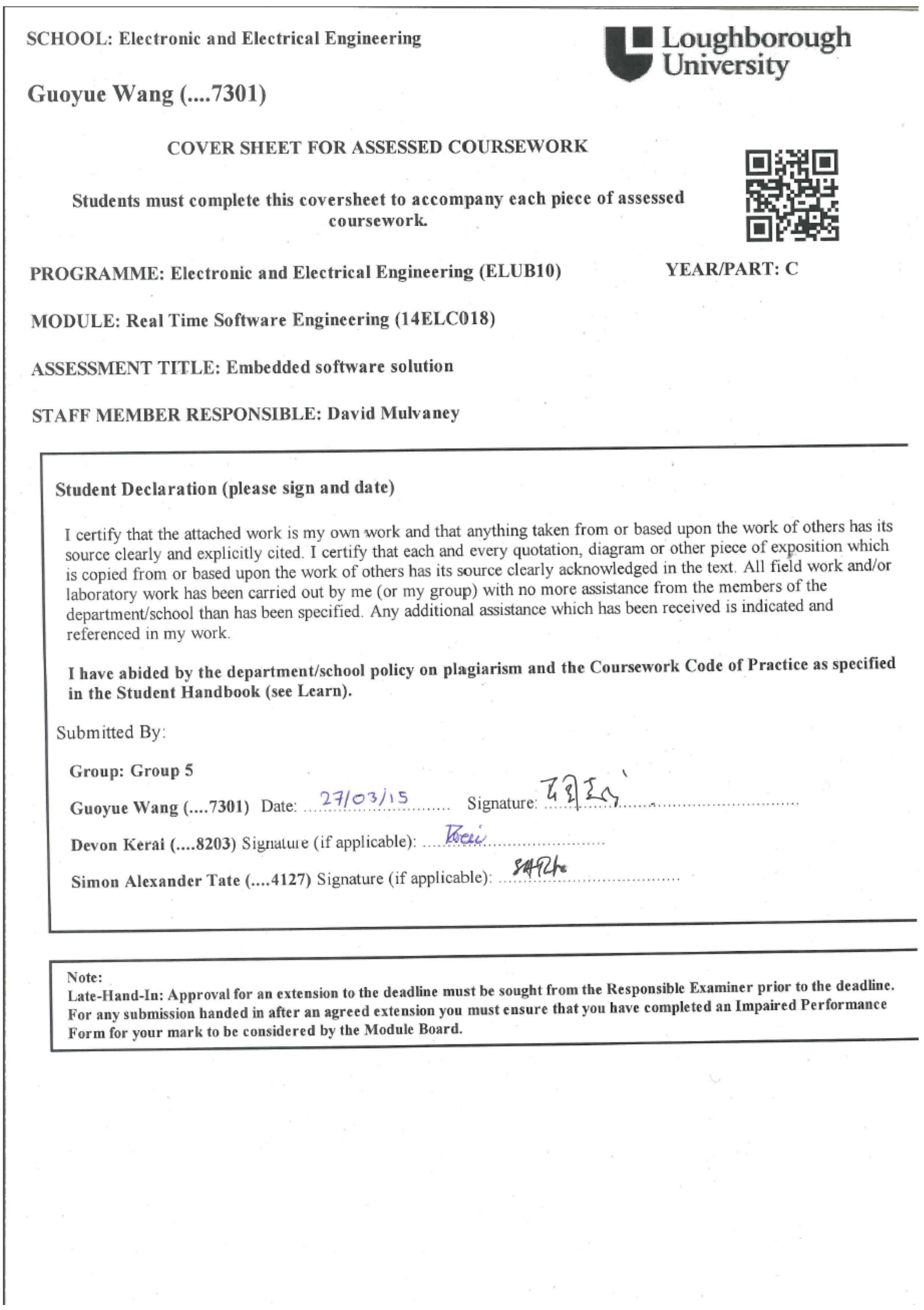


SCHOOL: Electronic and Electrical Engineering

Loughborough University

Guoyue Wang (....7301)

COVER SHEET FOR ASSESSED COURSEWORK

Students must complete this coversheet to accompany each piece of assessed coursework.

PROGRAMME: Electronic and Electrical Engineering (ELUB10)

YEAR/PART: C

MODULE: Real Time Software Engineering (14ELC018)

ASSESSMENT TITLE: Embedded software solution

STAFF MEMBER RESPONSIBLE: David Mulvaney

Student Declaration (please sign and date)

I certify that the attached work is my own work and that anything taken from or based upon the work of others has its source clearly and explicitly cited. I certify that each and every quotation, diagram or other piece of exposition which is copied from or based upon the work of others has its source clearly acknowledged in the text. All field work and/or laboratory work has been carried out by me (or my group) with no more assistance from the members of the department/school than has been specified. Any additional assistance which has been received is indicated and referenced in my work.

I have abided by the department/school policy on plagiarism and the Coursework Code of Practice as specified in the Student Handbook (see Learn).

Submitted By:

Group: Group 5

Guoyue Wang (....7301) Date: 27/03/15 Signature: [illegible]

Devon Kerai (....8203) Signature (if applicable): [illegible]

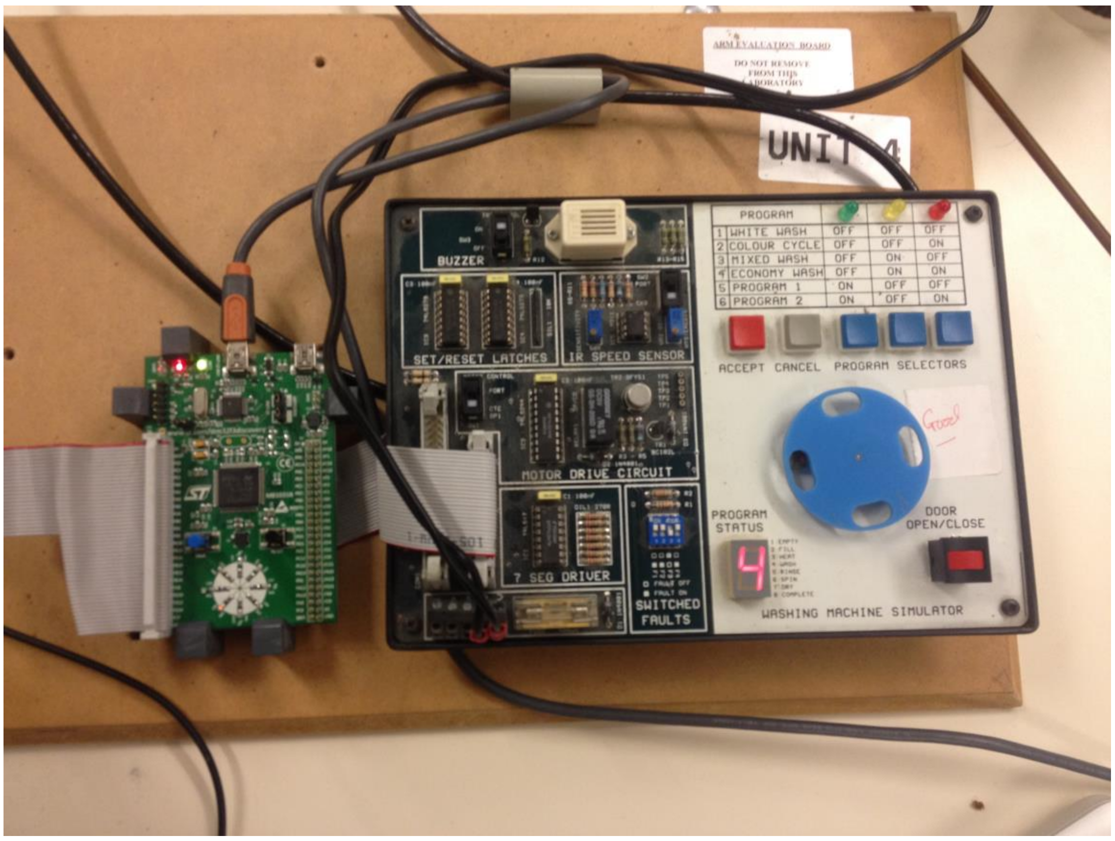
Simon Alexander Tate (....4127) Signature (if applicable): [illegible]

Note:
Late-Hand-In: Approval for an extension to the deadline must be sought from the Responsible Examiner prior to the deadline. For any submission handed in after an agreed extension you must ensure that you have completed an Impaired Performance Form for your mark to be considered by the Module Board.

Real Time Software Engineering Coursework 3

Written By:

Devon Kerai, Simon Tate, Guoyue Wang



Overview of the Programme Structure

The programme runs by using the three layers of embedded system development: target hardware layer, device layer and the application layer. These layers are made up of several files.

The target hardware layer defines the addresses of the ports. This includes *ports.h* and *ports.cpp*. "*ports.cpp*" has one class with one member function. This initialises the ports for the board. If the user wished to change washing machine and needed to change the port addresses, both of these files would have to be changed, but no other code would have to be edited.

The device layer defines all functions to be called in the main application layer. This is where the majority of the programme is written. This contains the files: *classes.h*, *ports.cpp*, *timer.cpp*, *buzzer.cpp, sevensegdisplay.cpp*, *motor.cpp*, *door.cpp*, *buttonpanel.cpp*, *washingoperations.cpp* and *processcontroller.cpp*. These C++ files use the classes which are defined within *classes.h* to control the system as a whole. These classes use member functions to carry out specific operations defined within the programme.

The application layer begins the running the main washing machine function *myController.Run().* This function is within *processcontroller.cpp* and controls the rest of the programme from there.

<u>Specification Guide</u>

<u>Specification</u>	<u>Is it satisfied?</u>	<u>Notes</u>
Program selections are indicated by LEDs	Yes	The ports are latched when you press the button and the corresponding LED is illuminated.
Pressing the accept button executes the relevant programme.	Yes	The combination of programme buttons are interpreted and the program number is returned which is then used to execute the program.
Further use of the accept button advances the programme by one stage for each press.	Yes	Pressing the accept button skips a stage by incrementing the counter by three.
The 7 segment display indicates the stage of the programme.	Yes	A washing operation is called which displays the appropriate number on the 7 segment display.
Pressing the cancel button pauses the machine.	Yes	The value of the counter running through the washing machine programme and the time left to pause, is saved to memory when the machine is paused.
Pressing the accept button while the machine is paused causes the programme to resume.	Yes	Using the saved counter and time left to pause values, the washing programme is resumed.

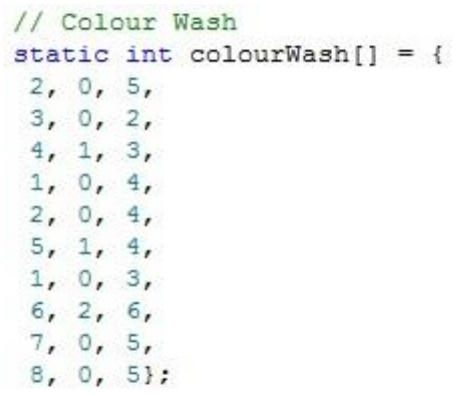
Pressing the cancel button again while the machine is paused resets the entire machine.	Yes	Pressing the cancel button twice resets all the ports, turns off the motor and resets the seven segment display.
If the accept button is pressed while the door is open, the machine will not start and a brief warning will be sounded on the buzzer.	Yes	The buzzer is sounded while the door is open and someone needs to press the accept button to start the washing programme.
If the door is opened during a wash program, the buzzer sounds briefly and the program is suspended until the door is closed again.	Yes	The buzzer briefly sounds if the door is open during the washing programme and the machine is suspended. Closing the door and pressing the accept button will resume the programme.
If the cancel button is pressed during the suspension, the machine resets.	Yes	Pressing the cancel button twice while the machine is suspended turns off the motor and resets the seven segment display and all other ports.

Figure 1 - Specification Guide Table

Adding a New Washing Programme

To add additional washing programmes, *processcontroller.cpp* must be edited, under the *Run()* function. The arrays inside of this function determine the current washing machine operations that can be carried out. If any additional programmes want to be added, the user would simply copy and paste the *static int* definition and replace the values. The user would then have to edit the switch case statement below to include their programme for the washing machine with the desired programme combination. This is simple copying and pasting and then calling the appropriate washing programme array from above. The procedures are simple and only minimal coding experience is required.

If the user wished to enter their own washing programme, they should be aware of the syntax in place for this. The first number of the sequence corresponds to the operation, e.g. 1 - empty, 2 - fill, 3 - heat, etc., the second number is the motor speed (0 = off, 1 = slow and 2 = fast) and the final number is the time to carry out this operation for (in seconds).This is then repeated until the user is satisfied with their washing programme. Figure 2 shows an example of a washing programme:



```
// Colour Wash
static int colourWash[] = {
 2, 0, 5,
 3, 0, 2,
 4, 1, 3,
 1, 0, 4,
 2, 0, 4,
 5, 1, 4,
 1, 0, 3,
 6, 2, 6,
 7, 0, 5,
 8, 0, 5};
```

Figure 2 - Colour Wash programme

UML Design Diagrams

Figure 3 is the Class Diagram for Task 1 and Figure 4 is the new Class Diagram for Task 3. Instead of initially considering the washing operations as different classes, they were created as member functions of a single class called '*WashingOperations*'. In addition, a function called '*MotorSpeed'* was created to control the spinner's speed under the *Motor* class.

Several additions had to be made to our UML diagrams to reflect the changes made in our C++ code. An example of this includes the addition of the "*mark*" and "*space*" variables within the *Motor* class. These were to control the speed of the motor using pulse width modulation. When the UML diagram for Task 1 was designed, it was not required that the motor speed needs to be controlled using PWM, so they were not included until it was discovered later on.

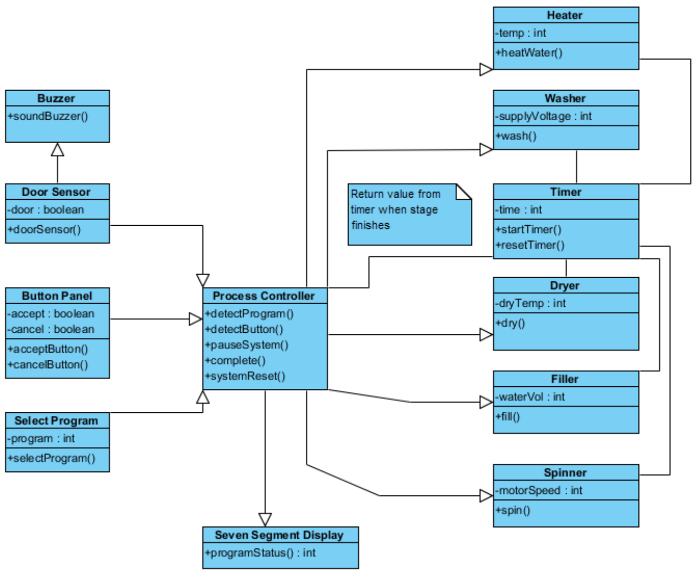


Figure 3 - Class Diagram for Task 1

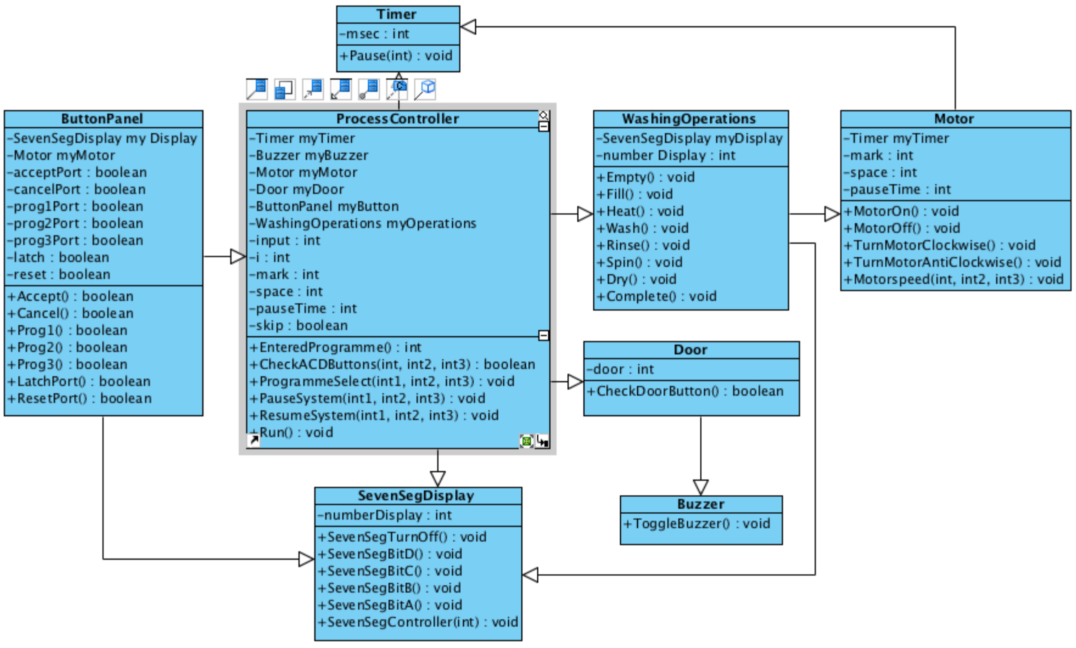


Figure 4 - Class Diagram for Task 3

The sequence diagram shown in Figure 5 is based on Figure 4. Sequences between classes were re-constructed to reflect our new function calls. In addition, minor alterations such as function names were amended to ensure that they can easily be understood.

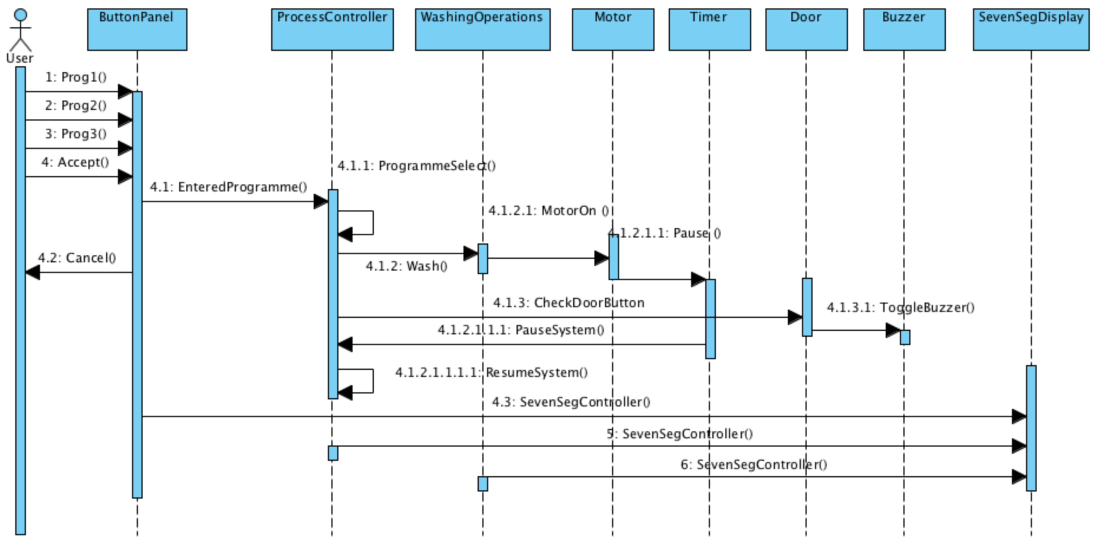


Figure 5 – Sequence Diagram for Task 3

Figure 5 is the sequence diagram for a user running a washing program, pausing the system and then pressing the accept button to resume the washing program. *CheckDoorButton()* checks the status of the door port. If the door is open, then *ToggleBuzzer()* will be called to briefly sound the buzzer.