UNIVERSITYOF HERTFORDSHIRE
School of Engineering and Computer Science

COURSEWORK ASSIGNMENT

Module Title: Embedded Systems Development	**Module Code:** 6COM1043
Assignment Title: CW2 Extended Autonomous Reliable Car	**Individual Assignment:** report individual, code in groups possible
Tutor: Raimund Kirner / Raymond Hu	**Internal Moderator**: Cherry Xianhui

Student ID Number **ONLY**:	Year Code:
16049767	0901-2019 (19/20)
16082770	0901-2019 (19/20)

Marks Awarded %:	Marks Awarded after Lateness Penalty applied %:

Penalties for Late Submissions

- Late submission of any item of coursework for each day or part thereof (or for hard copy submission only, working day or part thereof) for up to five days after the published deadline, coursework relating to modules at Levels 0, 4, 5, 6 submitted late (including deferred coursework, but with the exception of referred coursework), will have the numeric grade reduced by 10 grade points until or unless the numeric grade reaches or is 40. Where the numeric grade awarded for the assessment is less than 40, no lateness penalty will be applied.
- Late submission of referred coursework will automatically be awarded a grade of zero (0).
- Coursework (including deferred coursework) submitted later than five days (five working days in the case of hard copy submission) after the published deadline will be awarded a grade of zero (0).
- Where genuine serious adverse circumstances apply, you may apply for an extension to the hand-in date, provided the extension is requested a reasonable period in advance of the deadline.

Please refer to your student handbook for details about the grading schemes used by the School when assessing your work. Guidance on assessment will also be given in the Module Guide.

Guidance on avoiding academic assessment offences such as plagiarism and collusion is given at this URL: http://www.studynet.herts.ac.uk/ptl/common/LIS.nsf/lis/citing_menu

UNIVERSITYOF HERTFORDSHIRE
School of Engineering and Computer Science

ASSIGNMENT BRIEF

Students, you should delete this section before submitting your work.

This Assignment assesses the following module Learning Outcomes (Take these from the module DMD):

a. **Knowledge and Understanding:**
Successful students will typically have a knowledge and understanding of:
- LO1 - requirements and design techniques for dependable real-time computing
- LO2 - techniques to analyse resource-requirements of embedded systems

b. **Skills and Attributes:**
Successful students will typically be able:
- LO3 - to develop software for an embedded platform
- LO4 - to specify design solutions to ensure fulfilling the resource and dependability requirements of an embedded system

Assignment Brief:

See separate CW description.

Submission Requirements:

The assignment report of each group member has to be submitted individually on StudyNet by 3pm on the day of the submission deadline.

Each group member has to write and individual report in their own words, however the attached corresponding software can be product of a group collaboration, with other group members named on front page. An electronic copy of your program files has to be included in the submission as a .zip file with name `ESD-<GRP-Name>-<Report-Author>-SW.zip`.

This assignment is worth **70%** of the overall assessment for this module.

Marks awarded for:

This coursework consists of a prototype implementation which can be done in group work, an individual report for each group member, and also a practical demonstration, with questions related to the understanding of the report and prototype and the basic concepts from the lecture slides.

A. Report and Prototype

The report and prototype is worth 70% of the overall assessment for this module.

Out of this 70%, 60% go to the technical realisation of six individual topics and its correct description in the report, as detailed in the corresponding task description. The remaining 10% go to the clearness and presentation of your results.

B. Demonstration

All group members in case of group work have to understand the whole solution:
The coursework documents are submitted and graded as group work (report must be written individually, but the developed code can be based on group work). However, during the demo sessions questions are asked to individuals to verify the understanding the content of the submitted coursework:
- For CW2 the questions include a list of published questions covering important topics of the lecture slides.
  (CW2 is linked to understanding of methods and concepts provided in the lecture. To make sure that each student understands them, questions related to the lecture will be asked. The list of possible questions will be published on Studynet, so you can prepare for them well in advance.)

The total score for this module is calculated as follows:

total_grade_CW2 = grade_report_CW2 * (scale_demo_CW2 / 5)

From this formulae it is important to understand, that delivering the reports alone will not give you a good grade, you have to show your understanding based on the criteria explained above.

The value of **grade_report_CW2** is between 0 and 70.

The value of **scale_demo_CW2** is between 2 and 5. The value is determined via questions during the demo. While multiple questions are asked to determine a fine-grained value of **scale_demo_CW2**, the following discrete values are for your orientation:

2	3	4	5
Did not attend or answered all questions poorly	Answers show lack of basic understanding for most parts	Answers show basic understanding for most parts	Answers show clear understanding of all parts

This grading scheme allows you to produce your code and report without exam stress, but requires you to show understanding of your solution by means of a few questions during the demo sessions. The demo questions focus on fundamental concepts only, as explained above.

A note to the Students:

1. For undergraduate modules, a score above 40% represent a pass performance at honours level.
2. For postgraduate modules, a score of 50% or above represents a pass mark.
3. Modules may have several components of assessment and may require a pass in all elements. For further details, please consult the relevant Module Guide or ask the Module Leader. (this module requires overall pass, i.e., 40%)

Typical (hours) required by the student(s) to complete the assignment: **50** hours

Date Work handed out: 18.11.2019	**Date Work to be handed in:** 10.01.2020	**Target Date for the return of the marked assignment:** 24.01.2020

Type of Feedback to be given for this assignment:

Scores (overall and for each phase) will be announced via StudyNet. Detailed technical feedback will be provided during scheduled class and tutorial sessions, via email and by appointment out of class.

There will be the opportunity to submit draft reports to tutors for early feedback prior to final submission.

CW2: Blayne Cullen, 16049767, Group: ESD55

Section 1: Overall Design and Application Behaviour

The intended behaviour of the robot car is that it must react to the position of a blue dot. By this I mean, the robot car has been programmed to keep a certain distance (as defined in the program). Therefore, if the blue dot was to move towards the car, the car will reverse to keep the distance. The same will happen when the blue do move further away from the car, the car will drive forward to keep the distance. We also designed the code to make the robot keep aligned with the centre of the blue dot. Therefore, when the blue dot moves left or right, the car will move in that direction too and then realign itself with the centre of the dot. Finally, when no blue dot can be detected, an on-board blue LED will light up (telling us that it cannot find a blue dot) then the robot car will stay in one spot and spin right until a blue dot is found.

Section 2: Resource Awareness

There are a lot of factors that affect the time of which code and functions are executed on the robot car. One of the biggest factors of this must be the onboard battery. There are numerous problems where depending on the battery charge percentage, certain features become slower (like the movement of the car) and at certain time will cause hardware failure such as the camera. We conducted the measurement of reaction time by eye when testing the responses of the robot car. Based on this, we can say that the car has (in human terms) fast reactions as it takes less then 1 second for the car to react to sensors. Electronic timing has not been completed, so these results are unknown. The main factors in the insert_blob() function that affect the execution time is the while loop within it. As there is only 1 loop and there is no other loop nested inside it, it gives the function O(n) where n is the number if elements in the b1_start double-linked list. This means that the complexity is linear. With respect to memory allocation, the complexity of this function is O(2) as there are 2 variables that are not temporary being used in the function. These variables are: “b1” and “b3”. The function does use other variables, but unlike b1 and b3, they are not declared in the function, therefore, no additional memory allocation is necessary.

Section 3: Real-time Assurance

Priority	Service
1	Obstacle avoidance
2	Blob Search
2	Blob detection
3	Distance keeping
4	Blob alignment

In all things, safety is the most important thing. Hence, why it is the top priority in the service rank. Safety makes sure that the car and/or other objects are not in danger if a fault was to occur.

This design is perfect for the solution as the Camera and the blob analysis must be concurrent. This is because the Camera sends information to the blob analysis thread for the car to react to what the camera sees. If they weren’t concurrent, then there would be a large delay between the camera getting a picture and the other thread analysing it, which means that there is less time for the car to react, which could be dangerous (e.g. if there is an obstruction). A possible improvement to the program would be to allow it to run on more than two threads, allowing for different services to run on their own thread making the car more efficient. But two threads are a definite minimum amount due to the process of analysing camera data.

Section 4: Resilience

There are 3 types of sensors used in this project; Infrared, ultrasound and a camera.

The Infrared Sensors are mounted on the front near the bottom and are used to detect objects that may cause interference with the robot car, making it safe. The likelihood of one of these sensors to fail is quite minimal and is predicted in the fault tree, but there are two of these sensors (one on each side) that perform the same tasks. The probability of both sensors failing is doubled the original probability. So even if one sensor fails, the robot car will still be able to be operational with the other sensor. In the unlikely event that both sensors fail, there is an ultrasound sensor that can reduce the risk of an accident. The problems with the IR sensors are that they are not detecting evenly, for example, one sensor can detect things that are farther away than the other IR sensor. But no matter what IR sensor fails (if one failed) the robot car will still stop due to the received signal from the other working sensor, the only difference is the distance away from the object the car will stop at.

The Ultrasound Sensor is mounted at the front near the top of the car and is used to detect the distance between the car and an object in front of it. If this sensor was to fail, the program should crash as it no longer receives data from the sensor, causing the car to come to a complete stop. If this was to not happen, then the car will remain safe as the IR sensors would stop the car if a hazard was detected. The Ultrasound sensor sends distances to the program to determine what the car should do in order to keep a certain distance away from the object (a blob). As nothing is perfect, the sensor could return inaccurate data causing the car to possibly not be within the determined distance (but make the car think that it is). This is not ideal, but the car will remain safe as it responds to the distances and as a backup, the IR sensors. The fault tolerance is quite good for this sensor as the programmed minimum and maximum distances are quite far apart (but still suitable) allowing the ultrasound sensor to return inaccurate data and still allow the car to perform well.

The Camera is mounted at the front of the car just below the Ultrasound sensors and is used to detect the blob. The camera does this by comparing colours and checking if the read colour is within a predetermined range of accepted colours for it to tell the car to follow. If the camera was to fail, then the program will definitely crash as it can no longer receive data. As this sensor data manipulation runs on another simultaneous thread, the response would be quite quick. The crashing of the program will cause the robot car to come to a stop. As the camera works by picking up colours, it can be inaccurate if it detects a different object whose colour is within the approved range. To limit this, the camera has been written to follow a blue blob rather than a red blob. This is because the environment in which the robot car will be operating has a lot of red colour objects but very minimal number of blue objects, making the chance of this inaccuracy happening, minimal. The fault tolerance for detecting a blob on this sensor is quite good as the range of accepted colours is vast, but still accurate, allowing for the camera to have different contrasts, brightness, etc and still have it be able to detect a blue blob.

Section 5: Testing

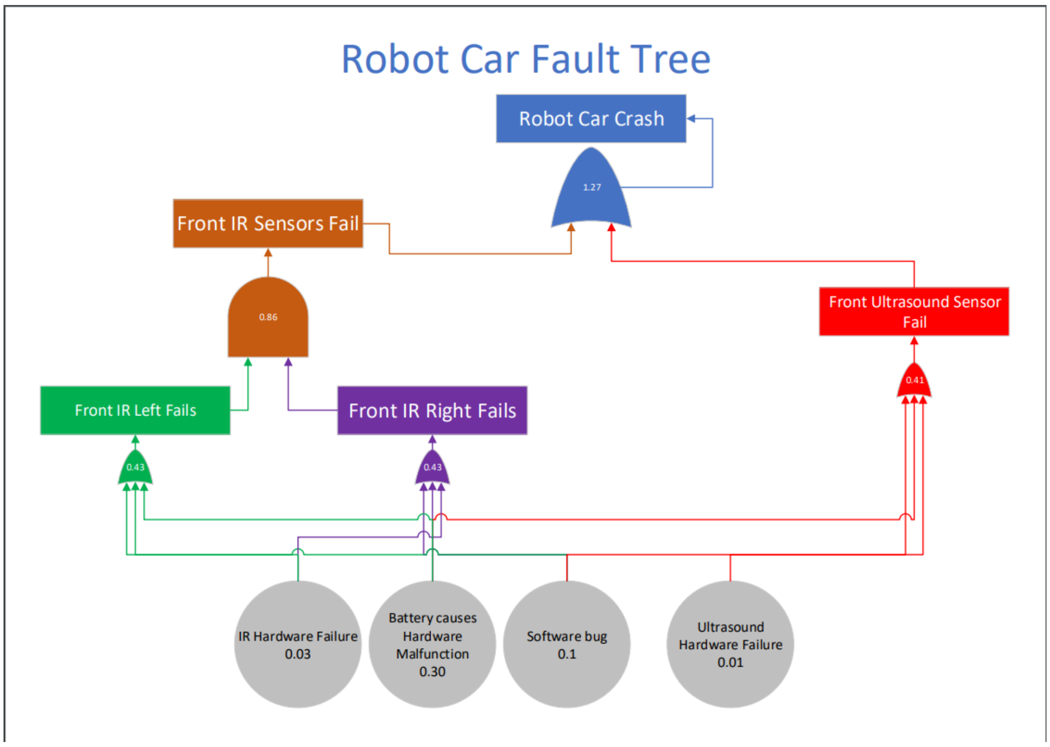
Test	Scenario	Expected Results	Actual Results	Success?
1	Left Infrared Sensor Detect Object	The robot car will come to a stop and remain stationary until the object is removed.	The robot car follows the expected result.	✓

2	Right Infrared Sensor Detect Object	The robot car will come to a stop and remain stationary until the object is removed.	The robot car follows the expected result.	✓
3	Both Infrared Sensors Detect Object	The robot car will come to a stop and remain stationary until the object is removed.	The robot car follows the expected result.	✓
4	Ultrasound Sensor – Object too close	The robot car will stop (if in forward motion), then drive in reverse until the ultrasound sensor detects that the car is within the approved distance.	The robot car follows the expected result.	✓
5	Ultrasound Sensor – Object too far	The robot will stop (if in reverse motion), then drive forward until the ultrasound sensor detects that the car is within the approved distance.	The robot car follows the expected result.	✓
6	Ultrasound Sensor – Object at correct distance	The robot car will stop (if in any motion) or will remain still until.	The robot car follows the expected result.	✓
7	Camera – No blob detected	The robot car will stop, then spin around in a clockwise direction until the camera picks up the blob.	The robot car follows the expected results, but there is a problem with the battery where if it's below a certain level, then the camera will eventually crash causing the car program to stop and therefore the car will stop.	/
8	Camera – Partial blob detected on right	The robot car will stop (if in any motion), then will turn right, fractions at a time, until the blob is within aligned position. Once done, the car will stop movement.	The robot car follows the expected result.	✓
9	Camera – Partial blob detected on left	The robot car will stop (if in any motion), then will turn left, fractions at a time, until the blob is within aligned position. Once done, the car will stop movement.	The robot car follows the expected result.	✓
10	Camera – Full blob detected	The car will remain stationary.	The robot car follows the expected result.	✓

Section 6: System Analysis

- Battery level too low
    - As we experienced during the testing stages, the robot car has a flaw where a lowered battery level causes malfunctions with the sensors and motors. One problem that persists is when the batter reaches a certain level, the camera stops working. As a result, the program crashes and the car come to a halt. Which is not ideal but considering that the car automatically stops as a result, it is still considered relatively safe.
- Sensors fail
    - Depending on what sensors fail, determines the severity of the risk. If the Ultrasound sensor was to fail, the car would not be able to track distances but would still be safe as the Infrared Sensors would pick up immediate hazards and would stop the car accordingly. However, if the IR sensors were to fail, the car would still be able to keep a distance from the target in front, but any immediate hazard would not be detected and therefore the car would crash.
- Motors fail
    - The motors are powered by DC from the battery. If the motor was to die, it would simply stop working. However, the other motors would continue to work. This means, depending on which wheel, the car will move in an unwanted direction. Despite this being undesirable, it is still safe as the US and IR sensors are operational and will stop the working motors when a hazard is detected.
- On board hardware fail
    - This is very unlikely, but still causes a threat. If the motherboard (Raspberry Pi) were to have a hardware failure, it would cause the whole robot to fail. Based on what we witnessed with the camera, when the program cannot communicate with certain dependent hardware, it crashes bringing the robot car to a halt.
- Environmental
    - The robot car is designed for indoor use only, this limits the effects of the environment. If it was outside, the rain could cause problems to the circuitry as it is not waterproof. Also, winds could affect the cards movement or sensors as the car is very lightweight and could be blown easily.

CW2: Blayne Cullen, 16049767, Group: ESD55


Robot Car Fault Tree
Robot Car Crash
1.27
Front IR Sensors Fail
0.86
Front Ultrasound Sensor Fail
0.41
Front IR Left Fails
Front IR Right Fails
0.43
0.43
IR Hardware Failure 0.03
Battery causes Hardware Malfunction 0.30
Software bug 0.1
Ultrasound Hardware Failure 0.01