Final Report

ECSE 211 – Design Principles and Methods

Team 5

Xirui Zhang: 260656808
Xinran Li: 260774237
Cheng Chen: 260775674
Arianit Vavla: 260868601
Tony Ou: 260867785
Hongtao Xu: 260773785

November 2019

1. **Introduction**

What was the main reason(s) of doing it?
What was the project intended to achieve?

Hard Skills:

- The design process of the project consists of recognizing the constraints surrounding the problem, and controlling the design variables to find a solution meeting with the constraints in an optimal fashion.
- The whole design process is a systems perspective, where a problem composed of inputs, outputs, and a process modelling the desired input-output behavior.
- The project examines the process of design with context of electro-mechanical systems, which consists of a combination of electrical and mechanical engineering.
- To let students having understanding topics of both electrical, computer and software engineering, become familiar with differ aspects of the design process.

Soft Skills:

- Communication skills - We need to constantly communicate to know the current progress and the steps that will have to be taken.
- Team working skills (Time budget and Work allocation skills) - We allocated our work into 4 subcategories: hardware, software, testing and documentation. Each person is added to a department, depending on their major and what they are better at. We focused on our own fields and helped others' once our tasks are completed. We have a Gantt Chart to keep track of the tasks left to do and important deadlines.

2. **Team organization – the start up of the project**

How were tasks allocated?

- Tasks were allocated based on the capabilities of each member and their roles in the project. Furthermore, members were assigned to other tasks by the PM if their current tasks were completed.

How was the initial Gantt chart designed?

- The Gantt chart was designed based on the objective of the project and the goals of each week during the design process

What information was used to estimate the initial task breakdown?

- The midterms and assignments that each team member had during every week

Were any guidelines followed in developing the first version of the chart?

- To estimate the time to allocate for a task, the "expected time" equation was used. It was a really helpful tool because it made the whole project more predictable

3. **Issues encountered in the progress of the project**

Were all dependencies correctly identified at the start of the project?

- Not all dependencies were correctly identified at the start of the project. For example there were post-demo software adjustments made and it was not predicted at the start of the project that this task would be necessary, so the integration testing was postponed to a further date. Also, a lot of clarifications were made during the weekly meetings.

What dependencies contributed to the critical path of the project?

- We could not test the hardware before the final hardware design
- We could not start implementing the software without a software architecture

- We could not test the software properly without the final hardware design
- We cannot start integration testing before finishing the post-demo software adjustments and if the unit tests were not reliable

What initial ideas turned out either not to work or be based on wrong assumptions?

- Most of the hardware preliminary design ideas got eliminated along the design process. We started our hardware design with the robot implemented for Lab 5 which had a catapult style ball launcher. Throughout the testing, it did provide a stable performance as well as desired shooting power. However, the catapult style ball launcher usually has a large dimension because the longer the arm is, the larger the torque will be. Even after several iterations to reduce the size of the catapult, there was still a possibility for the robot to touch the tunnel. Therefore, we had to give up on the catapult style and changed to another completely different launching mechanism using springs.

What other issues/factors had an impact on the project?

- As stated above, the limitation on the size of the robot had an impact on the hardware design. Furthermore, there are only 4 motor ports on each EV3 brick which means we only have up to 2 motors to be used for both launching and reloading mechanisms. Also, we could not use a rotating ultrasonic sensor for obstacle detection because of this constraint.

How did these affect the project progress?

- All the factors described above delayed our project progress. For example, the change in launching mechanism required both hardware and testing team a large amount of time to build and test a new ball launcher. It also affected the software progress because the whole ball launching algorithm involved both hardware and software components.

In particular, did the project run to the plan you had initially created?

- The project did not follow the initial plan strictly. Along the design process, we modified the plan several times based on the current status of the system. For example, at first, we were thinking about implementing both the launching and reloading system on the brick. However, after finishing the ball launcher, there was no space and no time left for us to build an extra reloading system. Therefore, instead of having several shots, we decided to improve the precision and consistency of the current launcher to be able to sink the ball at the first try.

4. **The budget**

What constraints did the budget place on your team?

- We could not allocate more time to the design and correct our issues at the start of the design process because all the team members had midterms and assignments due during the design weeks, so we were limited in the time that we would come up with a solution to the problem. However, we sped up the pace in the following weeks to compensate.

How did the initial planning for available resources and budget spending affect the development of the timeline?

- The software subteam had to start implementing their code at a later date, which also affected the testing team, who had to test the software at a later date.

Did you allocate resources to all the project tasks? If not, explain why not.

- Resources were allocated to all project tasks for members to try and make up time for this project.

What would you have spent if there had been no limits on the budget and when in the process would extra budget have been useful?

- If there had been no limits on the budget, we would have spent it on modifying our software design and deal with thread problems so we could do a localization when the robot enters the tunnel and exits it, which would increase the success rate of the full course run. We would also navigate component by component to take full advantage of angle correction instead of travelling using the minimal distance algorithm, which is a lot more error prone. For the hardware, we would have spent more time on implementing a reloading system. Extra budget would have been very useful, as our robot was not working correctly at the time of the final demo. More time would have given us more opportunities to perfect the software and try out different algorithms, which would've greatly increases the chance of success.

Where were you weak in resources and what would you have done to resolve this issue if you had fewer budgetary constraints? At what point in the project could these extra resources have been brought in?

- We were weak during the middle of the project, where all team members were busy with assignments and midterms. This caused all our tasks to be pushed to later dates and ultimately resulted in the failure of the final demo. If we did not have those constraints in resources, we could have solved the issues that we faced in the final demo and have time to try new ideas. We would also have more time to change the issues in our design by approaching the solution to the problem differently.

5. **How did the process contribute to the success (or failure) of the project**

Was the process useful in achieving the goals?

- The process was very useful in achieving our goals even though we failed in the final demos. For example, we constructed our preliminary hardware and software designs at the very beginning. Afterwards, we tested these designs and found out their weaknesses/problems. We then tried to improve them as early as possible to avoid having to take steps back. This process saved us a lot of time, as we would be even more behind in our schedule if we found out the problems later on in the design process.

How would you modify the process to increase your probability of success?

- To increase our probability of success, we can integrate our hardware design and software design after clarifying the preliminary designs using components from previous labs (i.e. Odometer, Navigation, Localization). Since we had a lot of constraints in the project such as limited motor ports, it would be better to get the hardware team and the software team on the same page early in the process. In our case, we used two motors for the launcher and two motors for the wheels. During week 5, our software team wanted to use another motor port for obstacle avoidance. However, the motor ports were already occupied by the launcher and we could not launch using only 1 motor. Thus, our design could have been improved if we constructed our launcher with only one port from the beginning.

Which parts of the process were the most difficult to implement and why?

- In the project, the software part of the process was definitely the most difficult part to implement. When we were developing the software, the robot would never behave exactly how we told it to in the code. For example, if we call the turnTo method to 0 degrees in the code, the robot would turn to 10 degrees in reality instead. The problem

could have been from multiple sources, such as the Odometer class or the Navigation class. Therefore, a lot of debugging was required for every small amount of code written.

How much time was devoted to testing?

- We spent 22.9% of our time on testing. We divided our testing into three parts: sensor tests, unit tests and integration tests.

Was this at the subcomponent level or did you leave it all to the end?

- We did the testing as we progressed in the design process. As described before, we divided the testing into three parts. We first did sensor tests to make sure every member of the team had a sense of how the sensors worked and their return values. We then moved on to unit tests, where we tested localization, navigation, odometer, odometer correction, launcher and obstacle avoidance. These tests were intended for us to see if each component worked well separately. When all individual components were assessed to be working correctly, we proceeded to the next testing phase, which was integration testing.

Were the tests you designed sufficient?

- The tests we designed were sufficient, because we had a good sense of what was working and what was not working after each test. Also, after the sensor tests, we could use the test results to properly tune the threshold for when a line is detected. The failure of the demos was not surprising, as our success rate of integration testing was really low. We needed more development in the software design to fix the issues pointed out by the tests.

How much time did you estimate full prototype testing would take?

- We estimated three to four hours for the full prototype testing. We did not spend much time on the full prototype testing because we wanted to make sure that each component works well before we combine them together. The components had a lot of issues, therefore, it took a lot of time to do the unit testing. Once we started integration testing, it was too late and a lot more problems appeared during the integration of components.

How much time did it actually take? If there was a difference, why?

- We spent similar time to what we expected. Our time was really limited as we were approaching the competition day, which means we could not have spent more time than expected.

How would you change the test design process to make it more effective?

- To make our testing more effective, we could have spent more time on integration tests and less time on unit tests. When we started doing the integration tests, we encountered a lot of the same problems that we thought were fixed during unit testing. For example, there were too many threads running in one of the integration tests, which caused stack overflow and random drifts of the robot. However, that same problem was encountered during unit testing and was thought to be fixed.

What was the impact of the beta demo on your design process?

- The beta demo reminded us that we were behind schedule and that there were components that still had to be fixed/implemented. For example, our robot barely got in the tunnel because the orientation after localization was not precise. Therefore, we started right away after the beta demo to implement the angle correction component.

6. **The success of the Design in meeting the original specifications and the performance requirements**

What is your impression of how the robot performed?

- We were hopeful that the robot would at least get in the tunnel, travel to the launching point and launch the ball because our last set of integration testing proved that it could do the series of tasks. However, it was not stable, and the angle correction would sometimes correct to the wrong angle. All 4 demos failed because of that same reason: the robot localized correctly, traveled to the tunnel and before going in, corrected to the wrong angle. We were deceived, but not surprised, as our integration testing started late in the design process and we could not dig out all edge cases.

Did the robot perform as you expected?

- The perform performed as we expected because we spent a lot of time on perfecting localization, which had a 75% success rate in the final demos. The final angle after light localization was also close to 0 degrees. Furthermore, our robot could not go through the tunnel for the same reason in all 3 successful rounds: the angle correction corrects the robot to the wrong angle. This bug was found out a day before the demo, very late into the process. We did not have the time to address the issue. We could not compare the rest of the demo because it never got through the tunnel. However, we are confident that it could have traveled to the launch point while dodging obstacles and launch the ball, as we tested this sequence of steps specifically and it had an acceptable success rate.

If the robot failed, why did it fail?

- If we only talk about the performance in the demo, our robot failed because it did not correct its angle properly before going into the tunnel. However, on a broader scope, we failed because we relied too much on unit testing instead of integration testing. Also, because we were behind in the schedule, integrating testing started very late. Therefore, we justify the failure of the final demo on not being able to properly address all edge cases in the software because we did not do enough integration testing.

Can you point to the sections of the documents that describe the decisions that led to the failure (provide the references to those decisions)?

- Refer to Section 2 of TESTING DOCUMENT, unit tests of each individual software algorithms are all relatively stable. However, the full integration tests only had a success rate of 20% which is shown by the test data. As stated under conclusion, we failed to pass the tunnel because the navigation was not very accurate and we did not correct its angle properly before going into the tunnel.
- Refer to Section 3 of HARDWARE DOCUMENT, the launching mechanism went through several versions which partially led to the delay of overall project progress. In other words, we wasted some time in developing hardware which means we had less time working on the software part.

7. **Conclusions**

What did you learn from this course?

- In this course, we learned a lot from design processes. Here are some key points:
    - Work in a large team from different backgrounds specialized roles
    - How to collaborate and communicate between subteams
    - How to properly manage and document a large scale project

- How to allocate time for the right tasks when being restricted by deadlines
- How to take key decisions in our design while having restricted resources

Explain why a clear, effective and controlled process is necessary when working in a team and what it helped you achieve.

- A clear, effective and controlled process is necessary when working in a team because it makes the communication between team members and communication between subteams easier as each member knows when to allocate their time for this project and have a higher success rate to achieve our goal. It helped us predict and allocate time slots when we were supposed to work on this project and facilitate communication when we had an issue, as we knew who was responsible for which part of the design process.

Is any of it applicable to other courses you might take?
If so, what and why? (name the courses)

- Yes, this is applicable to other courses we might take
- ECSE 456 (ECSE Design Project 2) and ECSE 457 (Design Project 2)
  - Those are the subjects need ECSE 211 as pre-courses. We need to do group projects on those 2 subjects. The communication skills, time-allocation skills and teamwork skills we accumulated in ECSE 211 help us in doing those.

What would you change in what you did if you were doing it over? (important!)
For hardware, instead of building a brand-new ball launcher, we would keep the catapult style with only one motor and build the reloading system. The other motor we would use for the rotating ultrasonic sensor which would be used in obstacle avoidance. This will be specifically described in next paragraph. We would also reduce its size to fit into the tunnel and increase the launching power by using more springs. By doing so, a large amount of time would be saved because we were already familiar with the catapult through lab 5. Having a smaller robot and a reloading system would also provide more flexibility to software team.

For software, based on our final demo, we would have two main changes if we were doing it over again. First, we will not use our current way of navigating to another point. The code we implemented was using Euclidean error distance to calculate the travel distance and turning angle using odometry. However, the odometry is not always accurate which results in inaccurate turning angle and distance. This would highly affect the result of robot travelling to the front of the tunnel, and it would not pass through the tunnel. If we were doing again, we would try to use simpler way, which travels straight lines and turn 90 degree instead of travel in diagonal. During the travel, we would do odometer correction each time we pass through a black line using two lighting sensors. This might be more like a hardcoding style, but results would be more accurate. The second strategy we would like to change for software, is instead of coding obstacle avoidance based on obstacle's given length and width, we would use a rotating motor to detect the front of robot in 180 degree and decide which way it should go based on the result. This would increase the level of successful avoiding in edge cases, since the current software design we have would fail some edge cases such as two obstacles sit close to each other near the edge of the board. The last part we would like to change for software is to implement the odometer correction as a thread so that it would correct the angle every time it detects a black line. We would also start the integration tests earlier to see if each algorithm is compatible with each other.

“The undersigned members of team 05 agree that the contents of both this report and the information handed in on cd, dvd or memory key, provide an accurate representation of the work done on this course and the contributions of each team member.”

Arianit Vavla

Cheng Chen

Xirui Zhang

Xinran Li

Tony Ou

Hongtao Xu