Reutlingen University

Falculty Engineering :
International Project
Engineering

EV3 Robot Tic-Tac-Toe Challenge Documentation

Aquarium Robotics (Group 7)

Date: 01.12.2020	Semester: 3
Joshua Hertel Michael Schwenkschuster Tan Jia Tian	

Executive Summary

This documentation is about the Tic-Tac-Toe game robotic challenge, which is based on an application of SCRUM for software designing.

SCURM is so far the most convenient and conventional tool which assists us to complete the software-based projects like designing a self-playing robot and developing a software which is suitable for public use. Most of our progress have been tracked weekly and this motivates our group to be always productive. Nevertheless, with the SCRUM-oriented approach we must prepare a presentation in each Sprint period, so that we can show our customers how much the current project has been expanded.

The 1st Sprint is primarily focused on basic tests for locomotion of robot and for functionality of the sensors provided in the box of Lego set. In the 1st Sprint our robot has also been rebuilt and various Tic-Tac-Toe strategies have been developed. The 2nd Sprint is mainly about the ability of robot grabbing blocks to desired position, which is one of the most important tasks that robot must execute in order to play Tic-Tac-Toe game. Besides, the code has been completed and tested in the same Sprint and the second game strategy has been in the process of development too. In the 3rd Sprint our code has been optimised and the second game strategy has been fully developed.

In the Challenge phase, our robot has successfully executed its tasks when playing Tic-Tac-Toe game and defeated the opponents, which shows that our game strategies produce effective results. Despite the fact the robot is able to play the game autonomously, it does not place the blocks accurately on every single playing field. Furthermore, the straight-line motion and 90° as well as 180° rotations are not perfect, which lead to the failure in Tic-Tac-Toe game.

Table of contents

1. Task Introduction

In order to complete our course of "Computer Science for Engineers" in the 3rd Semester of bachelor engineering programme called "International Project Engineering", a group of 3 people is required to use a robot educational set from LEGO Mindstorms EV3 integrated with the Python 3 programming language, which can be done in a virtual environment namely PyCharm Professional 2020.2.3. In order to transfer the updated codes from PyCharm to the Brickman from robot, we use too the WinSCP (Windows Secure Copy).

In the challenge, our customers, Mr. Lukas Ewecker and Mr. Markus Wachter, have assigned the team members from group 7 a task to design a self-playing robot which can play Tic-Tac-Toe game and defeat either a human or a robot as its opponent. It is expected that the robot should be automated as much as possible, to the extent that it can move via a specific pathway to a desired playing field of Tic-Tac-Toe based on the inputs given by Python coding. In order to aid our robot in playing the game a few manual sensors and many various LEGO blocks for constructing necessary body components for robot have been provided and allowed to utilise in the course of the challenge. Additionally, 5 cubical blocks have been given as a substitution tool for the robot to place its 'crosses' or 'circles' on 5 playing fields.

The robot is expected to carry out the following tasks in compliance with the rules and regulations introduced in the game:

- The robot must be placed behind the dashed line in the beginning.
- The robot must know when his new turn starts.
- The robot receives an input telling where the opponent has put its figures.
- Every input must be provided via sensors.
- If the block was placed imprecisely, the playing fields where the largest area of the block covers were counted.
- Blocks may not be touched by hands; robot should grab the blocks independently.

2. Technical and other Preconditions

At the beginning, when we first received the Tic-Tac-Toe Lego Challenge it seemed very intimidating. So, in order to lower our anxiety, we broke down the challenge into multiple subtasks. Therefore, we were able to get an overview of what was needed to be done as well, which tasks we should prioritize to have success on the project.

Working subtask by subtask also gave us a psychological advantage. Since we did not work on one giant task where we could barely notice progress over many sessions, we were able to get a motivational boost by every little assignment we completed.

We broke the challenge down in to the following steps:
First, we designed an activity diagram to get a general overview of all the interactions and technological abilities the robot must meet. All the interactions are represented in our user stories. All the points in the user story combined represent the skillset our finished robot must have for the challenge.

Afterwards we divided all the tasks into two main groups: The general hardware functionality such as moving, rotation or detecting colours. The second main part was the strategy or the algorithm. This gave us the hardest time, because we really wanted the robot to make independent and logical moves.
At last, we had minor tasks like set up a convenient board or build and design the robot properly.
To have even better visualization we sorted our tasks by priority. At our first meeting we decided to use the Kano Diagram. According to the Kano Modell we separated our tasks into Must-Haves, Performance and Delight.

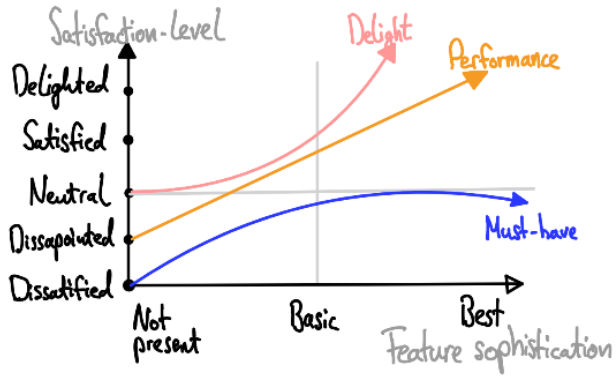

Figure 1 : Kano Diagram

Must-have tasks are a must have to participate in the challenge. They must be achieved no matter what. The challenge would not work without them; therefore, they are the most important ones.

Performance tasks are only optional for the challenge. Because the robot will be able to do the main functions without them. For example, if the robot moves properly it can still put a block on the field, even though the algorithm is flawed.

Very similar are Delight tasks. The special part of delight tasks is that these tasks have an especially bad benefit-cost ratio. Therefore, we figured that if we can fulfil them, we might get a slight advantage to other teams, because they are so time consuming and in relation to their outcome they will be not recognized as worth doing. After clarifying the categories, we sorted our slightly revised tasks into a product backlog as shown in the figure.

Product Backlog
Be able to rotate with almost perfect 90°/180° angles (Deviation of ± 3.3 %)
Be able to move to desired space/location
Be able to grab a block without releasing it when moving straight
Be able to grab a block without releasing it when rotating
Constructing the board
Be able to identify colours correctly
Use colour input to make logical moves independently
Lowering deviations by enhancing robot and board design

Figure 2: General Product Backlog for the project

We assigned six tasks into the must-have category.

Must-have tasks:

- Be able to rotate with almost perfect 90°/180° angles (Deviation of +/- 3.3 %)
- Be able to move to desired space/location
- Be able to grab a block without releasing it when moving straight
- Be able to grab a block without releasing it when rotating
- Constructing the board

Performance tasks:

- Be able to identify colours correctly
- Use colour input to make logical moves independently

Delight task:

- Lowering deviations by enhancing robot and board design

For making decisions on when a backlog is completed, we used the so-called DoDs (definitions of done). To complete a backlog, six set conditions must be met: Unit test passed, code reviewed, acceptance criteria met, functional test passed, non-functional requirements met, product owner accepts the user story.

In order to get a general overview which functions are required to succeed in the challenge we created a use-case diagram. This diagram introduces our robot (Karen) as the user in the game. In the middle of the diagram, you can see the main tasks (e.g., movements or block choosing).

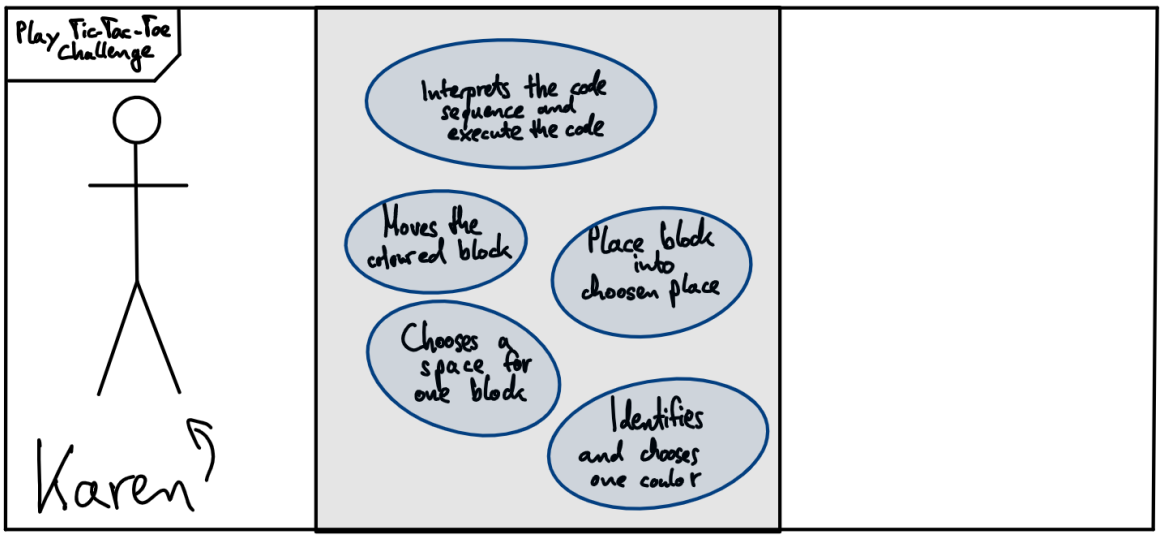

Figure 3: Use-case diagram

3. Sprint 0/ Project Setup

3.1 Analysis

Before our first sprint, our goal was to assemble the robot, get a general overview about all the sensors, functions and software(Pycharm and WinSCP). Unfortunately, our first thoughts to approach the challenge turned out to be completely wrong. We built the Lego robot from the instruction booklet instead of the Marsrover and elaborated an illegal strategy. In total we set our overall Story Points to 50 in a period of five weeks. Timewise we assumed to need 12 hours to clear 10 Story points each week. This leaves us, at a planned velocity of 0.833 story points per hour.

3.2 Design and Implementation

Let´s analyse what went wrong in our first strategy:

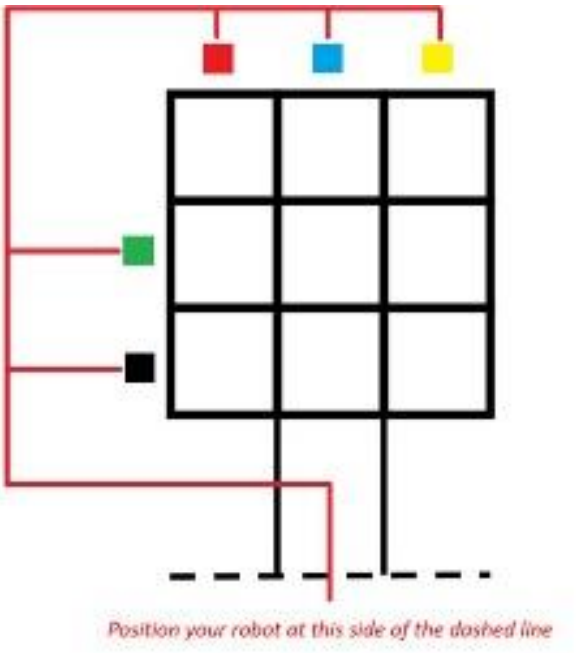

Figure 4: First elaborated strategy

Our idea was to place a block at each coloured location which you can see in the picture above. For each colour we would show via colour sensor, the robot would travel to the specific location behind one of the placed blocks. For pushing the touch sensor one, two or three times the robot would move forward one, two or three spaces.

Of course, this was forbidden because we are not allowed to tell our robot where to travel. In retrospective, we have been on the wrong horse since the beginning of the challenge unfortunately.

When we undertook the mini task “Drive square” provided in Relax we encountered severe problems with the robot's deviations on doing turns. Even though our theoretical approach was completely right, the actual result was always inaccurate. We calculated the radius as well as the distance the robot must travel and added it into the code via degrees per second. However, the deviation was too big, so we had to consult with Mr. Wachter. The solution was inserting a sleep function into the code. We felt pretty good after our first bug fixing. The last thing we did was rebuilding the robot to the Marsrover.

3.3 Discussion and Reflection

Feeling demotivated, we decided to make some changes for the upcoming sprint. First, we read carefully all the instructions in Relax. Afterwards we gave ourselves some feedback and made a list of what went wrong and what was good. Then we defined three main goals for the next sprint.

4. Sprint 1

User story: *"As an owner of Karen, I want to realise the external and internal abilities of robot when playing Tik-Tak-Toe game, so that I can confirm that the robot can execute the code perfectly and produce the desired results".*

4.1 Analysis

At our first official sprint we had a lot of things to catch up. To make things better, we defined three goals for the sprint in advance: Rebuilding the Robot, developing a new Tik-Tak-Toe strategy and to test Karens functionality in playing the game. Each group member was assigned an expert for one task. But of course, each member had an overview on how the project is developing. To organize all subtasks, we were using the Kanban board.

4.2 Design and Implementation

Rebuilding the Robot was easy due to the instructions on Relax. Also, we took in account our only Delight-Task defined earlier. Thanks to Mr. Wachter at our consultation hour we know that the robot can develop deviations when it is asymmetrical. The explanation is, that a leaning centre of mass will result in flawed movement, even though our code was correct in theory. So, we figured that spending a little more time on the robot´s design would be worth doing.

In contrast, the strategy or algorithm gave us a real headache. We realised that implementing an AI was way beyond our skill. So, we could not develop an actual working algorithm, we had decided to code for every possible move our opponent could make and added every possible countermove our robot should execute in order to win the game or at least achieve a draw. Although it was time consuming to scheme the strategy, in case we are the ones with the first turn, we achieved the goal to create a working concept which is not complicated. After informing and playing many games of Tic-Tac-Toe putting the first block at the upper left corner proved to be the best option. From that the concept is designed on setting up a trap, as well as forcing the opponent to block our moves if he can dodge the trap. This concept encloses the possible outcomes and consequently reduces the amount of coding.

For the third goal we tackled sets of criteria in the first user story: The colour sensor and the functionality of the robot to move. Which represents a must-have task as well as a performance-task.

We tested eight colours plus “no colour” (the colour which is scanned in the starting position). Even though there is a range for some colours, the colour sensor proved to be reliable. We discovered that we could close the shutter and turn on the light to provide always the same amount of light exposure for the colour sensor. Therefore, we were able to lower the deviation. Conveniently this is a Delight-Task which we solved.

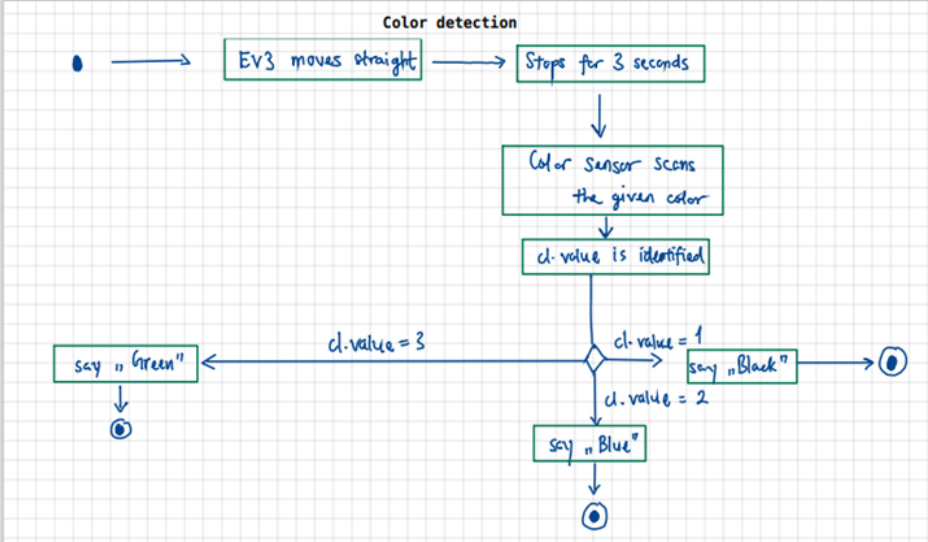

Figure 5: Activity diagram from our first sprint

In addition to that, we ensured that the wheels were dust-free for the challenge (another completed Delight-Task). Instead of a ~ ±3.3 % deviation as expected, it turned out that we have a ~ ±5% deviation. But fortunately, the colour sensor worked 100 % as planned. This time we managed to achieve 6.67 story points in 12 hours which translates to a velocity of 0.556 points per hour.

4.3 Discussion and Reflection

Our split in individual experts for the three given tasks seemed efficient. Therefore, we agreed to continue this for the next sprint. At this time our good feeling towards the strategy was still pretty good, even though it was still not completed at the end of the sprint. But we unfortunately did not feel to change it at this point of time. In conclusion for the robot´s functionality the colour sensor worked as planned. But the rotation with 90° was inaccurate. Besides, we had low team’s velocity. To solve this problem, we applied the Fish-Bone-Diagram or Root-Cause-Analysis. Our main cause of this problem or “head of the fish” was the low productivity. After outlining a few potential causes or “sub bones” our problem became more transparent. It was

due to long working sessions which were held on Fridays. Everyone was tired and psyched on the day before weekend so naturally it resulted in unproductive working. After we gained complete understanding about our problem, we had shorter sessions under the week with a set time limit of five hours.

5. Sprint 2

User story: *"As an owner of Karen, I want Karen to grab a block and release it onto one of the desired playing fields of a Tic-Tac-Toe board with the help of colour sensor, so that I could confirm that Karen is basically prepared to play Tic-Tac-Toe games."*

5.1 Analysis

Like we did in the last sprint, we set up three main goals in advance and designated an expert for each goal. It was also decided that we put our focus on the robot's functionality rather than the algorithm (code wise). However, developing a working Tic-Tac-Toe strategy when getting second turn (approach on how we want to play the game) was the most important performance task. A must have task at this sprint was to set up a board. And at last, but not least, preparing everything for the final coding basically summed up our user story.

5.2 Design and Implementation

Let´s start with the goals which we were able to complete this sprint.

Obviously without a board we would not be able to participate in the challenge at all. This might not be a task the robot has to manage but rather the team itself. Since we did not want any legal challenges afterwards with Mr. Ewecker or Mr. Wachter, we followed the board specifications on Relax as closely as possible. This was rather a legal precondition than a technical. Therefore, we decided to print the board from relax at a DinaA0 paper.

Now moving to our acceptance criteria from the second user story. We wanted to prepare everything for the final coding. We broke that down into several must-have tasks:

First must-have task:

We already had the code for making 90° or 180 ° rotations. We just needed to code several functions on how far the robot should go straight. We enhanced the angle deviation as well. A combination of these functions and the rotation functions gave us specific function to go to each of the nine different squares on the board.

Second must-have task:

We had to construct a gripper which had the right height and a high amount of grip. Which leads us to a material-based precondition. We were only allowed to use Lego parts from the box we received from Mr. Wachter. So, there is only a limited number of useful parts suited for a gripper. We decided to use some Lego parts made of rubber. In addition, we used the parts from the box in order to build a block dispenser, where our robot will drive to grab a block each round.

Third must-have task:

At first, we thought we were done, when we saw that the Robot can move straight forward while holding a block. But when the robot rotated, we were proofed wrong. In conclusion, we lowered the robots speed while rotating and we solved the issue.

Our expected results were that the robot would grab all blocks firmly. Executes all 9 paths via colour detection correctly and would be capable of moving back to the original position afterwards.

But our actual results were that the robot drops the blocks sometimes. Our solution was to lower the grabber and lower the robots speed when making turns. Also, the robot confused paths sometimes. We used coloured Origami labels to make sure that the material is consistent.

Figure 6: Origami labels to show the robot colours

This time we were very productive: 13.33 story points in 12 hours translated to a velocity of 1.111 points per hour (V3).

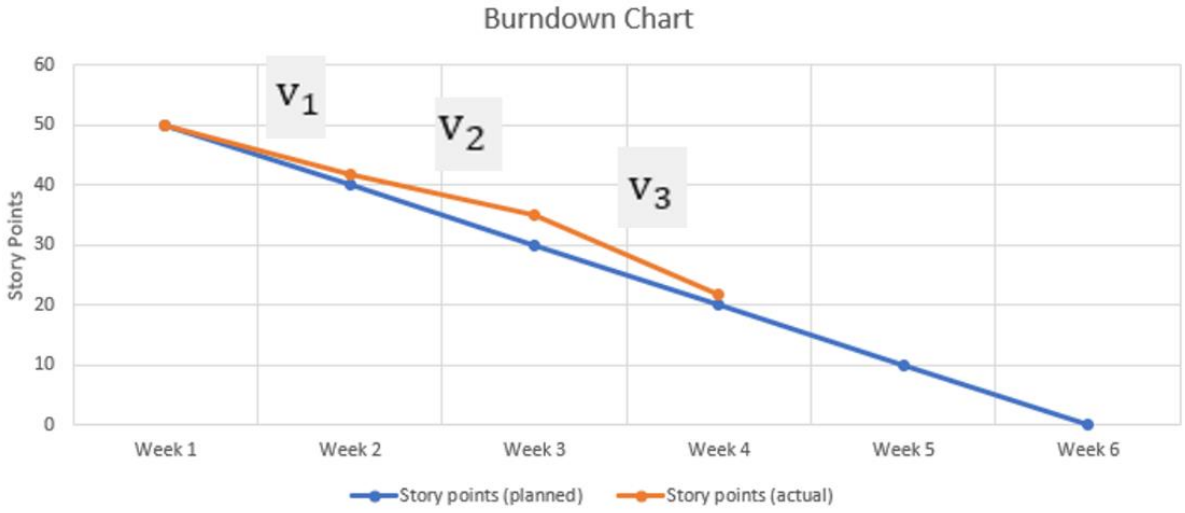

Figure 7: Burndown chart after sprint 2

5.3 Discussion and Reflection

In conclusion to the second sprint, we figured that the movement functions were almost perfect. Now we realised that our strategy was impossible to implement in the robot. To calculate every possible game outcome would require coding thousands of paths and if/else statements. At the end of our set time at the second sprint we decided to think of various agile project management methods on how we could solve our algorithm dilemma. Because now we were running out of time.

6. Sprint 3

User story: *"As an owner of Karen, I want Karen to play the Tic-Tac-Toe game independently with the help of touch/colour sensor, so that I could ensure that customers are truly satisfied in using Karen as a game tool."*

6.1 Analysis

We did not assign experts again, since the only real thing missing in the equation was a working algorithm. Our three main goals for the sprint were: Develop a Tic-Tac-Toe strategy. Complete an algorithm and setup the game. And begin to work on the documentation.

We figured to extract our problem we could use something named Dot-Voting. As we known from FPL, Dot Voting is a popular and quick method for determining priorities by voting. Having a due date on our tail, everybody sketched three to four ideas how we could develop a new algorithm or at least get our hands on one.

Long story short it turned out that many groups will use the Minimax algorithm or even copy and paste on from the internet. We also went on the internet and watched many YouTube videos on the issue.

But in the end, that was no option for us, because we wanted to learn something in this class and come up with our own solution. Also, the worst thing if we would fail the challenge is that we would not get any bonus points.
So, we decided to make an own algorithm without any help. The whole team worked on the algorithm and afterwards we finished our remaining tasks.

6.2 Design and Implementation

In order to play the game (when the opponent goes first), we need to show our robot where the opponent made his crosses. Therefore, we decided to use the colour sensor as input: Nine colours for nine spaces. Inside a while-loop which we use to start and end the game, we created an empty list which will be filled when the colour sensor receives any input. In order to allow the robot to scan colours round by round we coded a for-loop. In addition to that, we

programmed overall 325 lines of if-else statements referred on the list which cover all possible states of the game and counters with the best possible move.

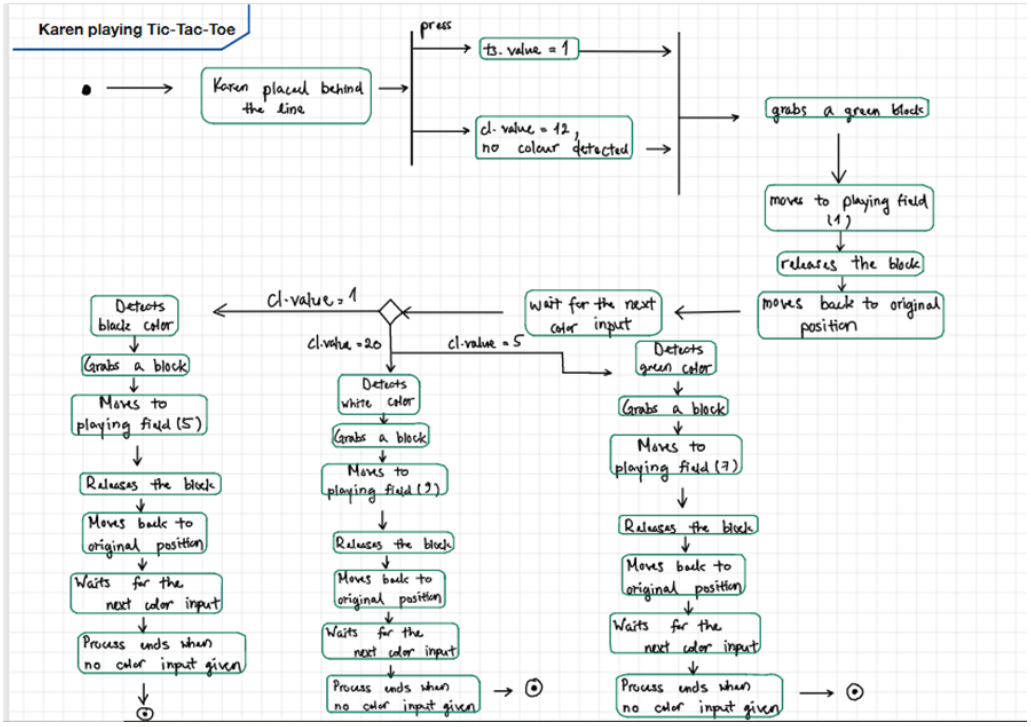

Figure 8: Activity diagram of algorithm

After completing our performance tasks (new algorithm) and finishing our must-have tasks (general movement) we tackled the delight tasks when we had some leftover time at the end. We worked on improving the board and robot design. Obstacles like dust or tape might affect the robot's movement. The user story was pretty much the same thing from last sprint.
We expected Karen to grab all blocks firmly, execute all pathways via colour sensor correctly and rotate with perfect angles.

The actual results were that the robot seemed not to go straight sometimes. This is related to the factor called internal gear slop. Our bug fixing was to reduce the robots speed to minimise the difference. Besides, the grab function did not work properly. As our solution we changed the shape of the grabber. As seen from the burndown chart our last velocity was around 1.25 points an hour. All we had to do now was to insert the code into the robot.

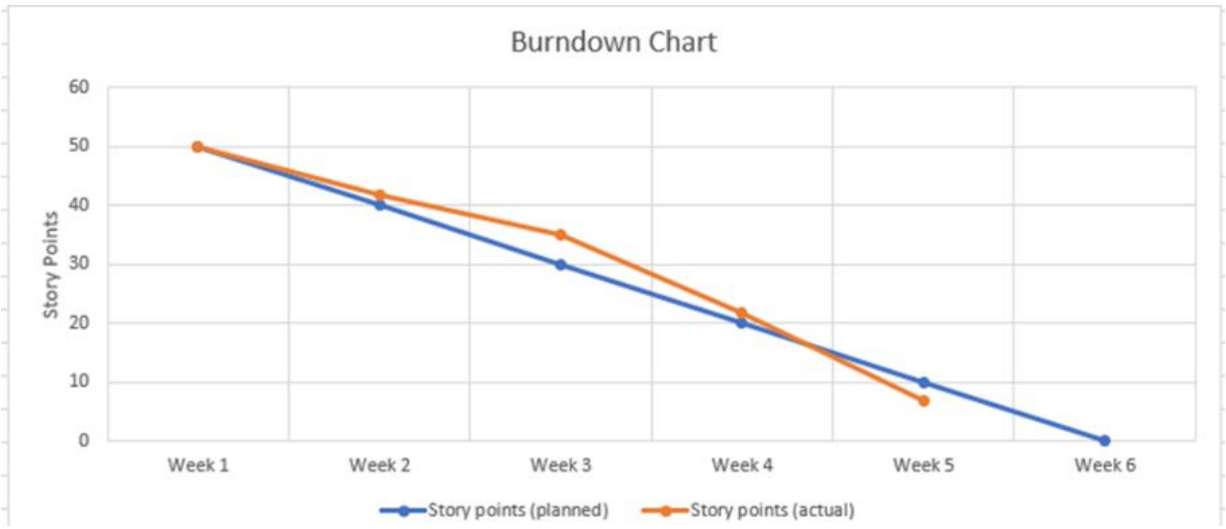

Figure 9: Burndown chart after Sprint 3

6.3 Discussion and Reflection:

In conclusion the robot is now 100% prepared to play Tic-Tac-Toe. There is the possibility that the robot fails because of deviations. However, we did everything we could think off in order to prevent any failure.

7. Challenge

7.1 Analysis

On the final week before the day of the challenge, two separately written codes (movement code and opponent detection code) have been combined into a single code. As expected, the testing of the combined code is very successful and therefore our robot is basically prepared for the challenge.

Even though with finished the code, the robot can hinder the opponent's moves in the game, particularly when the opponent is on the first turn, the combined code is only restricted for one possibility, which is when the opponent puts a cross or circle in the 5th playing field (in the middle of the Tic-Tac-Toe board). Due to this problem, we made sure to write more additional codes so that our robot can deal with the possibility that one all nine possible playing fields positions is placed firstly and randomly by the opponent.

7.2 Design and Implementation

Figure 10 illustrates one of the possibilities that robot can win against the opponent in Tic-Tac-Toe when the opponent puts cross/circles at the 2nd playing field. The game flow can be clearly shown via the following activity diagram below:

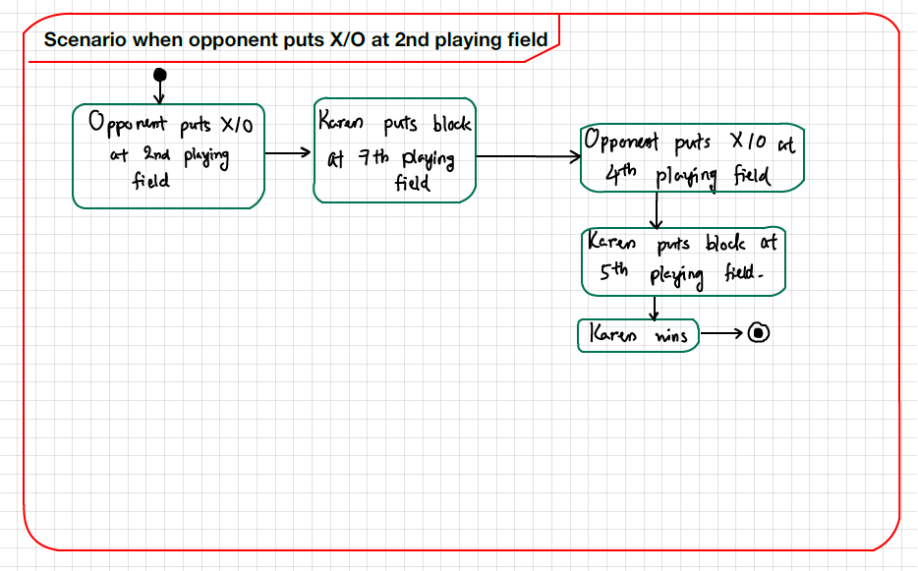

Figure 10: Karen's Possibility when Opponent starts the Game

From the activity diagram, one could interpret that the robot can obtain a winning point by placing indirectly 3 blocks diagonally. There are also more ways to ensure that the robot can win the game with 3 blocks horizontally and vertically. In order to implement all kinds of winning probabilities in the code, an empty list called ob = [ ] (opponent block) has been written in a while loop and the list can be continuously added with any one of the numbers ranging from 1 to 9. These numbers represent the position number of playing field chosen randomly by the opponent. When one provides a colour input under the colour sensor, the robot automatically detects the colour as a number position of the opponent and the ob = [ ] will turn into ob = [a] where $1 \leq a \leq 9$. This process continues until either the robot wins against the opponent or there is a tie between two players. The combined code can be seen in Figure () in the Annex.

The Challenge:

When the first round of the Challenge had begun, the robot failed to grab the very first block properly. The grabber just slipped off the sides of the block. This had to do with not enough friction being applied to the block and the 180° rotation had a significant amount of deviation by about 5.5%. It was also observed that the robot takes about 2 to 3 minutes to start moving

and executing the codes. It was because of thousands of codes, including functions and if-else statements had been written and copied securely into the file browser of robot. This led to the increase in time complexity, which also lowered the overall efficiency when all of the codes were executed. Albeit our group failed to complete the challenge, we were still content with efforts that were paid off in designing the self-playing robot.

7.3 Discussion and Reflection

The first aspect that one can discuss was about our team's velocity. It is noticeable that the team's velocity has increased weekly from Week two to Week five sharply. This is because all team members have relatively quick learning tempo about the Tic-Tac-Toe algorithm and be able to develop in a short amount of time many creative ideas of making own version of Tic-Tac-Toe algorithm according to the basic Python3 knowledge learn throughout the course "Computer Science for Engineers". The least productive Sprint throughout the project is the 1st Sprint because most of our times we were focused on rebuilding the robot and hence least number of times has was consumed in developing ideas of Tic-Tac-Toe game strategy and creating general game algorithm.

Although our challenge was not as successful as expected, it gives us a positive note that we must learn from failure so that we could have more rooms for developing more advanced and efficient programming skills as well as the ability the adapt to and overcome any technical difficulties.

It is undeniable that the skills acquired in the challenge are valuable and totally useful when we face real programming-related projects, especially the usage of SCRUM methods. Initially we found that the SCRUM methods were meant to cause all team members working in a stressed and quick-paced environment. This is because we had short meeting time per week and almost more than 50% of our last meeting time were used to prepare for Sprint weekly presentation, leaving the rest of the time for testing robot functionality, developing game strategies and coding. Nevertheless, as we were getting accustomed to the repetitive SCRUM-based tasks and workflow, SCRUM became a helpful tool for us to always complete any necessary tasks within a week by concentrating on the to-be-done tasks and neglecting the future tasks. It made our project very meaningful as in practice weekly presentations could show customers how much engagement we put as a team for the project completion.

In short, at the end of the challenge, we concluded, that the application of SCRUM is advantageous for software-design-oriented projects. Hopefully it would be put into practice more frequently in the future.

8. Annex

8.1 References

[1] Prof. Dr. Taefi, Tessa (2020), *Computer Science, Script Python and UML,* Reutlingen University, Wintersemester 2020/2021

[2] Klara Uhlig, *Tic-Tac-Toe-Roboter*, Elektrotechnik und Informationstechnik, Otto-von-Guericke-Universität Magdeburg

URL:

http://journals.ub.uni-magdeburg.de/ubjournals/index.php/LEGO/article/view/968/945

[3] How to Never Lose at Tic Tac Toe - Part 1 (Corner Game). Viewed 11. November 2020

URL : https://www.youtube.com/watch?v=5DebznNDuxU

[4] Never Lose Tic Tac Toe - Part 2 (Middle Game). Viewed 18. November 2020

URL : https://www.youtube.com/watch?v=EiShxMXwogU&t=6s

[5] Never Lose Tic Tac Toe - Part 3 (Side Game). View 18. November 2020

URL : https://www.youtube.com/watch?v=lmQrPEQtp2E&t=416s

[6] Configuring Line Separators. Viewed 11. December 2020

URL:

https://www.jetbrains.com/help/pycharm/configuring-line-endings-and-line-separators.html

[7] Stack Overflow, How to make Pycharm faster/lighter?. Viewed 11. December 2020

URL:

https://stackoverflow.com/questions/38242978/how-to-make-pycharm-faster-lighter/45775946