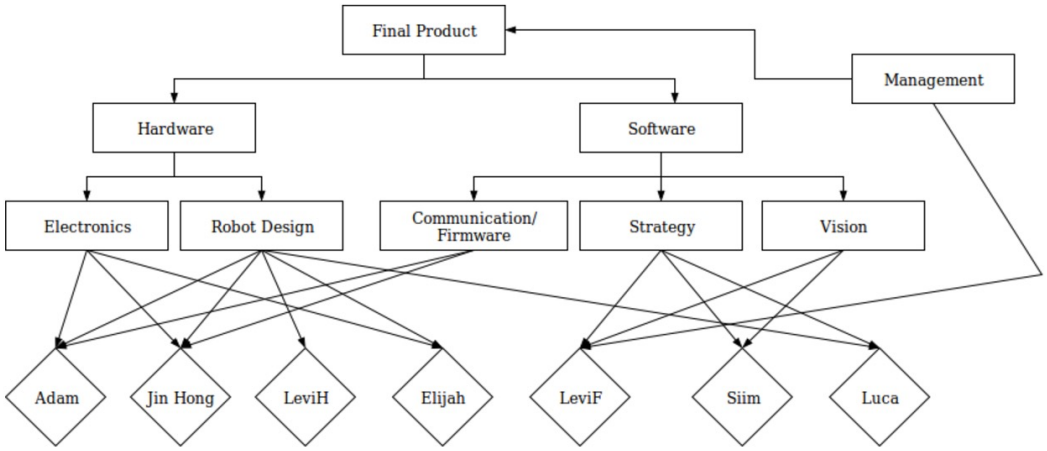
SDP Process Report

Group 12: Adam H., Elijah I., Jinhong L., Levi F., Levi H., Luca G.M., Siim S.

February 2017

Organisational Structure

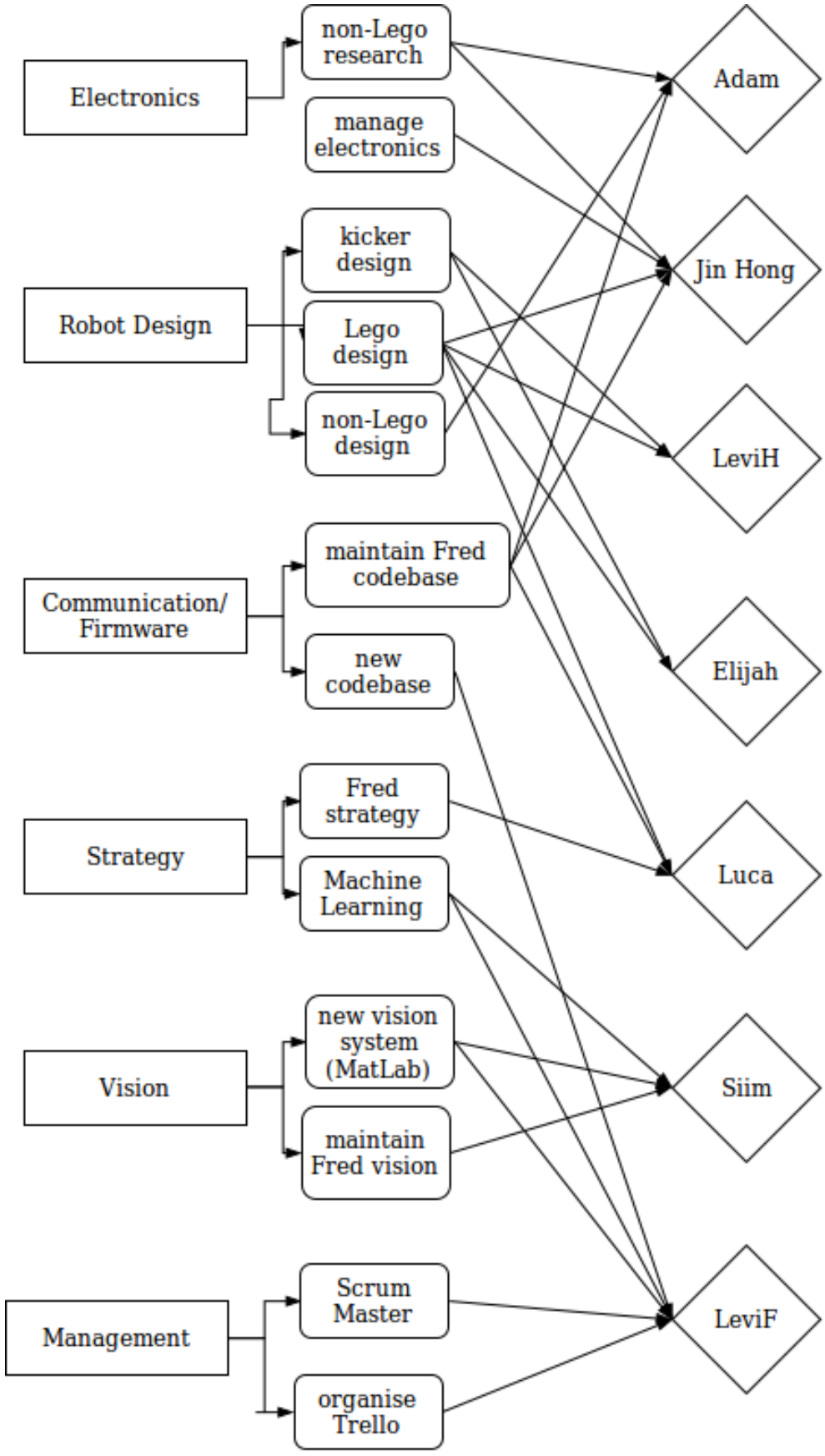
The structure of our group was largely influenced by our scrum approach to completing tasks. Scrum management largely revolves around rapid production of small-term tasks that are constantly adjusted according to the current project situation (ref ?). Completion of many small tasks as a whole is relatively counterproductive, with many ideas bottlenecking in a feud for proving one ideas merits over its alternatives. Alternatively, one approach is to constrain the decisions of each task to a smaller group of two to four members, allowing for more constructive and efficient decision-making. This division can be further exaggerated to form a hierarchy of teams, each with more finite tasks they are working to accomplish. The figure below demonstrates one possible organisation:



How subdivisions, or branching, of task allocation was decided required particular knowledge of each group member's capabilities and interests, and the global concepts that define the project we are trying to complete as a group. A clear division is to separate hardware and software, as both areas can accomplish tasks largely without the reliance of the opposite being completed. Hardware was further decomposed into electronics and robot design and software decomposed into communication/firmware, strategy, and vision. This design decision largely evolved from individual member's preferences, as well as sensible pairings (e.g. members working on electronics would most likely be working on firmware and robot design) which were discussed in the first week as follows: (NOTE: in **bold** are the assigned groups of those that members were interested in. These do not take into account the areas that members were not interested in.)

Members:	Adam	JinHong	Levi H	Elijah	Levi F	Siim	Luca
Interests:	**Electronics**, **Robot Design**	**Electronics**, Strategy, **Robot Design**, **Communication / Firmware**	Robot Design	**Robot Design**, Strategy, Vision	**Strategy, Vision**, Manager, Robot Design	**Strategy, Vision**	**Strategy**, Vision, **Robot Design**

Within each subgroup, tasks were assigned to the members to divide the labour in parallel. The format of these tasks is discussed in the "Risk Management" section in detail, but for now we emphasise the use of parallel work to have one task group working on new future features, and another task group working on a current working system (as a fallback should the new features fail). This is outlined below:


Electronics
non-Lego research
manage electronics
Robot Design
kicker design
Lego design
non-Lego design
Communication/ Firmware
maintain Fred codebase
new codebase
Strategy
Fred strategy
Machine Learning
Vision
new vision system (MatLab)
maintain Fred vision
Management
Scrum Master
organise Trello
Adam
Jin Hong
LeviH
Elijah
Luca
Siim
LeviF

Following this division-of-labour, we then employed the agile-development process of Scrum.

What is Scrum?

Scrum meetings revolve around three critical questions that each member of the group must discuss/present to the other members [1]:

- What have you accomplished since the last meeting?
- Are there any obstacles in the way of meeting your goal?
- What will you accomplish before the next meeting?

Adjustments to the overall project are made dynamically each week, allowing for a more organic development process. Because the requirements for building a robot and its accompanying software change as our team learns more about the task, it is important that we do not resist these changes but willingly adapt to the demands.
The size of our team (7 members) makes it ideal for a scrum organisation. It is rarely possible to conclude a perfect team structure by the first or second week and alterations are inevitably necessary. Our first scrum system defined sub-teams (i.e. Electronics, Robot Design, Vision, etc.) to "self-scrum" and track progress between its 2-3 members. Laziness, and a general lack of liability and communication caused this structure to inevitably collapse and become a mess of random, insensible notes with no clear direction. The key aspect of the scrum concept had been lost: collectively as a group describing goals met and goals to be met in short sprints – this makes all team members accountable for one another's progress. During our third meeting in week three, the scrum system was reorganised so that the sprints would take place during the meeting. Each member would, in-turn, discuss their answers to the three critical questions outlined above. To avoid wasting time, constraints were added to each section of the meeting. The new outline is defined below:

MEETING PLAN (as of Feb. 7th)	
What we do in this section	**How long this section will take**
1. Questions for James: any pressing questions for James concerning the game, robot parts, etc. that are not about the plan/sprint.	5-10 min
2. Outline the new meeting model: describe the scrum/sprint approach to the team if it is the first week, otherwise, each week re-clarify what the approach of the meeting is.	5 min
3. Circle Time (sprint) and discussion of 3 critical questions: incrementally, go around in the circle and give each member a chance to discuss the 3 critical (sprint) questions, as described above. At the end of this, each member should have their goals written down for next week's meeting.	30-60 min (approx. 5-10mins per member)
4. Discussion/Planning of big ideas: where does the team see itself in the future? What are the big concepts and when are we imaging these will be done? We do not discuss individual goals.	5-10 min
5. Open Discussion: end the meeting with general discussion or any essential points; only if necessary.	5 min

Communication Mechanism (Interaction between members)

When determining external channels of communication, three essential aspects were analysed for various communication systems: accessibility/Usability - "can all members use this?", features - "does the medium complement the organisation style of our group?", and accountability - "by using this, will it make all group members accountable for knowing the most recent information and dates?". An outline of the possible systems is described below based on group discussion (our decision is in red):

Communication Channel	Accessibility/ Usability	Features	Accountability	Overall Score
Email	All members have this. 10/10	Slow; Convoluted archiving of messages and media. 4/10	No accountability 0/10	14/30
Slack	Not everyone has it nor is familiar with it. Largely computer-use only. 5/10	Fast; Able to create sub-groups for different teams; Archives all data. 10/10	Less accountable for messages because it is largely a web-app. 7/10	22/30
Facebook Messenger	All members have this. 10/10	Fast; Archives all messages and media; Easy phone use. 9/10	All members are held accountable for seeing messages. 10/10	29/30
WhatsApp	All members have this. 10/10	Fast; Archives all messages and media; Easy phone use 9/10	Less accountable for seeing messages (because no record of when the member was recently online) 8/10	27/30
Text	All members have this. 10/10	Fast; No archive of group messages; Unreliable. 5/10	No accountability 0/10	15/30

After establishing that Facebook Messenger suited the requirements of an external communication system, we had to define a local area to share and distribute work in an organised manner that complemented the scrum-style of our team. For a code repository we chose Git-hub - the obvious choices were Git-hub or Bit-bucket, and a decision largely depended on which of the two was the more popular among the group members. A location to document goals and keep track of progress was required. A few options were considered, such as Google Docs, Trello, Gannt Chart, etc. and we scored their eligibility based on usability - "if the software is hard to use, lazy members won't use it", scrum-ness - "does the software complement our organisation style and goal-setting?". The outline of the possible choices is described below (our decision is in red):

Goal Tracking Software	Usability	Scrum-ness	Overall Score
Google Docs	Easy; simple; work in parallel 10/10	Not designed for scrum 5/10	15/20
Trello	Easy; simple; work in parallel 10/10	Designed for scrum 10/10	20/20
Gannt Chart	Unknown; organised visually; parallel work not as clear 7/10	Designed for scrum 10/10	17/20

Trello was the most suitable option due to its simple usability and its design pattern that complemented scrum perfectly. The purpose of the three communication mediums, including group meetings, is described below.

Facebook Messenger is used for discussing and sharing ideas with each other. The ideas we discussed would be noted down on Trello. We also used it to gather members to work and organise own meeting in the Forrest Hill.

As mentioned above, the group was divided into two sub-groups – hardware and software. Each of these was then divided again into more specific areas and each group member was assigned to one or more areas. We created a board in Trello for each area. Members of each area use their Trello board to log current goals and their deadlines and the tasks that must be undertaken to accomplish them.

At the weekly group meetings, members of each area share with the group the progress they have made since the last meeting, and whether they have met their deadlines listed on Trello. By presenting their progress to the team, it is ensured that each group remains accountable. Also, it ensures that each member gets a good idea of the overall progress the group has made.

Milestones

It is important to emphasise that the scrum-based approach to our project causes an organic, dynamic, and fluctuating environment; this allows fast adaptation. To complement this structure, milestones and future goals can be more vague the further they are (more than a week). Because decisions are made in weekly sprints, the reference points to our progress are our preparation/performance for the friendly matches. A few additional milestones may be set as the weeks progress and tighter deadlines are a requirement:

Milestone	Date	Description	Notes (On Completion)
Friendly Match #1	Feb. 1st	Build an implementation of Fred and learn about the software and hardware to have a robot driving around on match day.	Match results: 0-0, 0-0, 0-0, 0-0 Robot moved and hit the ball towards the goal periodically. We successfully implemented a Fred-esque robot and met our milestone.
Friendly Match #2	Feb. 15th	Finalise our robot design (lego or non-lego) and aim to have a new vision system built. Use the Fred codebase to build a simple strategy to go to the ball and kick it towards the goal (this should be a simple, stripped-down, Fred program). Following this match, we will begin discussion with our teammates.	N/A
Friendly Match #3	Mar. 1st	Have our new vision system working without errors for match-day use. Build new codebase for controlling robot and strategy, etc. Design with teammate-plan in mind.	N/A
Friendly Match #4	Mar.22nd	Finish the Machine Learning strategy to implement on this day (by now we will know whether it will work or not). Finalise software implementation and any enhancements to robot (no re-design).	N/A
Final Day	Apr.7th	WIN!	N/A

Risk Assessment & Contingency Plan

Before organising a contingency plan, it was important to outline the potential risks that could occur during the duration of our project plan. Our approach to tackling risks would be to have parallel work to resort to if the risks taken should not prove fruitful. For each potential feature that is a risk, we outlined the reason for the feature, its pros, how the feature could fail, and what would happen if this feature does fail. This is outlined below:

Figure 1: Gannt Chart: milestones were laid out in a gannt chart, and showed goals relative to the major milestones discussed earlier (friendly matches) so that despite the scrum-style sprints, we had a common direction to aim towards for each weekly meeting. *The yellow represents the weekly regular meeting *The pink represents the stuff that should be done by the hardware team *The grey represents each friendly match *The blue represents the development process of the vision system and strategy after each friendly match *The green represents the reports' progress

Risky Feature	Reason	Pros	How it Could Fail	If Feature Failed
Robot redesign to use non-Lego parts (motors, batteries, etc.)	The power supplied by Lego motors and the 9.6V battery is minimal. It would be a huge advantage to have powerful electronics and motors, and considering our budget is not spent we might as well use the money.	+ powerful motors + long-lasting battery + should reduce the need for a kicker if our robot uses its speed to hit the ball	- extensive research required to know which parts to buy - despite buying the parts, the new robot may not be any better than using Lego motors - new parts may not work	If the feature fails, we risk wasting a large amount of time and budget on a robot that will not work correctly. Not only will we be without a robot, but we may not have the budget to build another one
Machine Learning to curate robot strategy	Robot strategy is quite complex. Currently, the approach to strategy is a deterministic state-machine that performs a series of static actions. These actions are limited by what the programmers implement. It would be resourceful to spend time teaching the robot to play the game of football and then let it be creative, instead of hardcoding the rules into its strategy.	+ creative and original approach + once the initial learning algorithm is complete, it is easy to teach the robot new strategy without having to write more code + We can rapidly prototype new strategies in a simulated environment	- complex approach to translate the simulated world to the real world - tweaking of hyperparameters and learning algorithm can take a long time - may not be possible with our resources	If the feature fails, we risk wasting a large amount of project time that could have been spent designing a more complex strategy from the deterministic system.
Building new vision system in MatLab	Utilising MatLab's powerful resources and a good knowledge of the language, we should be able to create a new vision system that is external from the Java codebase that performs much better. The approach revolves around using attention to reduce search space and object recognition	+ Fast image manipulation with MatLab software + Pre-written code from IVR to use for object recognition + Vision system would have less parameters to tweak each time it is launched	- An important aspect of the vision system is having low latency. Running a MatLab program from Java may increase latency. - we have not tested running MatLab via Java	If this feature fails, we would have to rebuild the vision system in Java. Java's mathematical power is less than MatLab and this may result in a slow algorithm.

Handling the above risks requires a thorough contingency plan that will help recover from risky failed features. The core of our approach to handling the potential risks was to utilise our large amount of resources (i.e. number of group members) to work in parallel on risky implementations, creating a separate simpler implementation. For the "Robot redesign to use non-Lego parts", we have a second robot that is a redesigned version of Fred. We make sure all our system works on this robot first, and therefore should the new robot fail, we have a failsafe. For "Machine Learning to curate robot strategy", we have two teams: one team working on ML strategy and the other team working on simplified strategy from the Fred codebase; once again, this provides a failsafe. This parallel approach is not used for "Building new vision system in MatLab", and therefore this feature is the most risky of the three. Should this feature fail, the recovery option would be to use the vision system from Fred's codebase and restart the process of creating a new vision system in the language of the native codebase (whether it is Java or not). To allow for another recovery time, it is important that the new vision system is tested frequently and is implemented early on in the project timeline (see Gannt Chart).

Progress Tracking

Below, we write the progress for each week, following from the meeting prior. A concise order of events and goals is described:

- Week 0 (18th Jan – 24th Jan):
  - Create Facebook group for communication
  - Organise Trello for tracking progress and managing
  - Got the equipment and looked at it and tried to get electronics working
    - Set up robot communication and Arduino
    - Test Motors
    - Started design analysis
  - Research into previous RoboCup designs concerning drive systems, kickers, etc.
    - Analysis of a good approach
  - Meeting #1
    - Determine organisational structure and who will be working in which sub-teams
    - Created goal to have Fred robot and its software completed and driving for the 29th of January. This would give us 3 buffer days to test the system. Building Fred's system gives us a rough idea of how everything works together and we can start from this general perspective to break down the individual tasks and improve them.
- Week 1 (25th Jan – 31st Jan):
  - Focus was on building a robot and not on software development because Fred's codebase was functioning.
  - FRED 1.0 made on the 25th and driving the same day:
    - driving is horrible, would keep spinning and wouldn't move any dedicated point/area
    - No grabber or kicker, this is important to WIN the friendly and get a all 5 points
  - DIAGONAL 1.0 on the 26th:
    - Reason: Fred's chassis does not allow for room to place a grabber and kicker that is not a crappy propeller
    - Design is based on rotating the wheel layout by 45dgs to open up the front of the robot and add a kicker. To do this we must use omni wheels to allow for diagonal driving. This would also require editing Fred's code (which isn't hard)

    * The focus of this design was to create a modular robot that was easy to customise and edit (e.g. add new kickers, grabbers, etc.)
    * Kickers
        · Rubber band shooter: it is built following the classic idea of the bow and arrow. A motor drives pulling the rubber band backward and this provides strength to hit the ball once releases the rubber band.
        · Spinner w/ Grabber: A wheel is directly drived by a motor with gearing mechanism. A grabber would use to hold the ball in a stationary position and the wheel spinner would spin the ball to turn as fast as the wheel turns in a opposite direction. Once the grabber is up, the ball would move forward with the peak speed.
    * Testing
        · Originally the robot wasn't strong enough to move with raw motor power. We added gearing first from 1:2, then 1:3 to increase the strength, this proportionally decreased the speed. The robot was able to drive, but not fast.
        · The largest portion of the weight was in the centre of the chassis, this caused the wheels to bend outward on the robot because it lacked structural support. This was another factor that slowed down the driving and made it erratic. We attempted to fix this but instead scrapped the design.
  - 2 NEW DEISGNS on 27th/28th
    * Divided the team into two, to come up with two different designs to prove which was a better approach: a Fred-style robot with weaker grey motors or improve the diagonal robot by using the more powerful white/orange motors
    * FRED – NEW 1.1 on 27th/28th
        · The problem with Fred was that its chassis was too wide. We decided to move the motors inwards to make room in front of the wheels for a kicker/grabber
    * DIAGONAL 1.1 on 27th/28th
        · We tried to build a stronger structural support in this version and the wheels were not bending outward
  - Both new designs of Robots still kept spinning
    * We found out that this problem due to the power that sent from the server to the robot. For the original Fred, they were using a motor multiplexer board and sending power that was between [-255,255]. However, for our robot, we did not use the multiplexer board and the arduino board only accepted between [-100,100], this caused the motors drived over-power.
    * It was too late for us to figure out this problem and had no time to investigate and improve the vision system and strategy.

- Week 2 (1st Feb – 7th Feb):
  - Discuss the reason why we would fail to win a game on First Friendly Match
    * Vision System is not well detected the robot in the match
    * Strategy is out of idea
    * Spinner w/ Grabber is not powerful enough to spin the ball toward in a fast speed and hardly to grab a ball with the Fred strategy
  - Discuss the improvement of the Vision System and start to build a new one

- Week 3 (8th Feb – 14th Feb):
  - Add on

Bibliography

[1] J. Clifton, Marc; Dunlap. What is scrum? *www.CodeProject.com*, 2003.