# Introduction

# System Overview

-still have to edit this a bit

# Team Collaboration

Collaboration and teamwork between members of the Software group was extremely important throughout the entirety of the project. Members were often individually responsible for different areas of the software – or, alternately, were simultaneously rewriting or contributing to different sections of the same code – so it was essential to make sure that all parts were compatible and completed on schedule. Communication between the Software group and other project groups was similarly vital, to ensure that all work positively contributed towards the project’s long-term goals. This section aims to explain how the team was able to effectively work together on the project.

## Communication

The primary time for communication and collaboration as a group was during the team’s weekly meetings. Meetings occurred at 2pm-4pm on the Monday of every week, and were generally attended by all group members. While work could be done outside the meetings – and this was expected to happen – the meetings were valuable for planning and scheduling purposes, for tackling problems and making design decisions as a group, and for solving certain issues that required face-to-face explanations. Team members were able to work together in the meetings allowing some tasks to be completed much more effectively. Importantly, at the end of each meeting, a report of the work done during the prior week and a list of tasks to do the following week was produced, giving the project a continuous, clear direction.

GitHub was used as the group’s repository for software work. GitHub is specially designed for software, and is essentially a web-based hosting service for software development projects that uses the Git revision control system. It allows all team members to contribute to the same project by working on their own local “forks”, and then “merging” their changes back into the main branch of the software. The Git system ensures that work by different team members is tracked and that other work is not accidentally overwritten or changed (obviously very important when dealing with large amounts of code). All team members were encouraged to use GitHub and it was very effective in synchronising project work. GitHub also features a notifications and issue-tracking system, which was useful for keeping track of tasks and immediately notifying team members of any changes.

Outside of meetings, email was the main form of communication – email was considered more professional and ‘permanent’ than social media. Email threads exist for all of the project’s main areas, discussing any decisions, changes and ideas, and were also used for tutorials and explanations of different tasks. Email was also used for notifications and to organise additional meetings. For less formal communication, the software group created their own IRC channel. This was essentially a chat channel that could be used to discuss any aspect of the project, to exchange ideas, and for communication about current work.

## Scheduling

At the beginning of the project, an overall schedule for the software team was created, outlining the main tasks which had to be completed and their target completion dates. While this was useful for planning purposes and creating an overall impression of the task, it became less and less relevant as the semester continued. The nature of the software team’s work meant that it was often changing from week to week; varying hardware requirements from other teams, unexpected hardware issues and some nebulous project guidelines led to frequent modifications to the schedule.

For example: use of the BeagleBone turned out to be a significant time-sink, requiring a lot of troubleshooting in certain areas since its documentation was lacking; the sensors hardware was not finalised until Week 10, making it difficult to schedule work on image processing; and use of a sophisticated login system was not mentioned until late in the project, so many resources had to be diverted to implement this. Essentially, while the software group did attempt to keep an overall schedule, this was most useful in the initial planning stages due to the changing priorities of tasks.

Far moreuseful was the weekly scheduling system. As mentioned in the “Communication” section above, a weekly task list was created at each Monday meeting, giving the team a clear direction. This was much easier to change and suited the flexibility of the software well; tasks could be shuffled and re-prioritised much more easily and split between team members. It was still very important to keep the project’s end-goal and overall deadline in mind, and the weekly task lists could be used to do this by looking separately at the main areas of the software (such as GUI design, sensors, and so on) and summarising the remaining work appropriately.

## Group Participation

The nature of software development means that it tends to be very specialised – extensive knowledge of coding is required to be effective, and while it is possible to learn, it is difficult to learn *well* in a short timeframe. The different members of the software team all had varying levels of experience and therefore could not contribute equally to all areas of the project. Some team members had done very little coding before (outside of introductory units at university) which made it difficult for them to contribute directly, while others had extensive knowledge of the networking and server setup required.

However, despite their varying skill levels in regards to coding, different team members had skills in other areas, and these skills were allocated to ensure all team members could contribute effectively. For instance, as some people worked on the base server code, other people worked on the visual elements of the GUI design, and it made sense for the people who were fastest and most efficient with coding to work on those tasks while others worked on other elements. Even though the main task *was* principally computer coding, there were many supplementary software development tasks – writing documentation, testing, safety considerations, et cetera – that were involved. Some areas of the software such as the BeagleBone interfacing were new to all team members and could therefore be worked on by everyone.

On the whole, group participation was very good. Team members regularly attended meetings and did the expected (often more-than-expected) work in between meetings, responded promptly to communications, and remained enthusiastic throughout the semester. The aforementioned splitting of tasks attempted to give important work to each team member. Of course, while all team members contributed a significant amount, some did stand out – in this case Samuel Moore and Jeremy Tan, who performed a large portion of the vital software development work. Without their input the project would not have been completed to such a high standard, and their extensive skills and dedication were vital to its success.

## Inter-Team Communication

Communication between the various project teams was also essential. The software had to be able to control and interact with most aspects of the hardware (electronics, pneumatics, sensors, case), and a lot of this communication was focused around inputs and outputs of the BeagleBone system controller which the software ran upon.

Meetings every Tuesday morning were set up specifically for inter-team communication. At these meetings, all of which were attended by the software team, information could be exchanged and discussed between the project groups. Some discussion also occurred during the Monday group meetings; the sensors team and mounting team were present at the software team’s meeting time. For the software team, most communication was with the electronics, sensors and pneumatics teams, as these three hardware areas are all directly controlled by the software. The fact that the software *can* interact with these systems is evidence that communication was relatively effective.

Many other meetings also occurred between the software group and others, outside of these times. Extensive time was spent with the electronics team, testing and setting up their BeagleBone and organising the appropriate inputs and outputs. Other meetings also occurred with the sensors team to select sensors and cameras that were compatible with the software (…though there were still some issues in this area). Practical sessions with the pneumatics, sensors and electronics teams also occurred, in which the software was tested with these aspects of the hardware, to ensure that both software and hardware were reacting correctly.

For other communication, email was used extensively. Messages were frequently sent between all members of the unit to discuss different areas, such as to ask for general input on final hardware designs, or to organise meeting times for testing. Email was often less effective that face-to-face communication (though the software team did their best to respond promptly to communication, other teams sometimes did not respond) but it was still a useful tool. In addition, an MCTX3420 DropBox was set up as a common repository for any files and documents related to the project. This was updated very often and proved to be a useful reference for work. The software team chose to keep most of their work on GitHub rather than DropBox, and the GitHub repository was publicly accessible by the other teams so that work could be shared.

# Cost Estimation

The vast majority of the cost of the software team’s contribution was in man-hours rather than hardware. The only hardware specifically purchased by software was a BeagleBone Black; all other hardware for testing was part of the electronics. Some hardware used for testing was temporarily donated by team members, and has been included here only for completeness.

|  |  |
| --- | --- |
| **Item** | **Cost** |
| BeagleBone Black | $45 |
| LinkSys router (testing only) | $50 |
| Logitech webcam (testing only) | $25 |
| Ethernet and other cabling (testing only) | $10 |
| *Total* | *$130* |

In regards to the time spent, it is difficult to get an accurate record. At least three hours per week were spent by team members in weekly meetings. By consulting the team members’ technical diaries, it is estimated that each team member spent an average of ten hours per week working on the MCTX3420 project in any aspect. Using the ten-hour number for six team members:

* Approximate time per week (individual): 10 hours
* Team size: 6 people
* Approximate time per week (team): 60 hours
* Project duration: 13 weeks
* Total time spent: 780 hours
* Hourly rate: $150/hour
* Total cost: $117,000 (+$130 for hardware)

This is a large amount at first glance, though it must be remembered that this was a complex software development project with many interfacing parts (and at one of the weekly meetings, James Trevelyan stated that software usually actually accounts for *half* the work of a project, not one-sixth). Given that there were some inefficiencies in the project work that could hopefully be avoided in the future, software cost can be rounded down to approximately $100,000.

# General Development Process

-git use

-coding style

-documentation

etc

# Individual Contributions