Imperial College London

Department of Electrical and Electronic Engineering

ELEC50008: Engineering Design Project

Group 18: Mars Rover Project Report

Authors

Aixin Zhang
CID: 01738988
az419@ic.ac.uk

Ebby Samson
CID: 01737449
es1219@ic.ac.uk

Igor Dmytrovich Silin
CID: 01756268
ids19@ic.ac.uk

Kaling Ng
CID: 01737644
kln19@ic.ac.uk

Nur Izzah Mohd Zafer
CID: 01738670
nim19@ic.ac.uk

Xin Wang
CID: 01735253
xw2519@ic.ac.uk

June 12, 2021

Contents

1 Project Management

The project team utilised the Project Management Institute's 5 Phases of Project Management [1] as a guide to ensure all aspects of project planning and management are captured in the team's project management approach.

Project management was split into 3 areas, each covering a important section of project management.

1.1 Conception and Initiation

Project definition: Design and build a rover system that has autonomous capabilities to detect, avoid and transmit the locations of the obstacles i.e. coloured balls to a server that users can interact with.

Project requirement: The rover system is split into 5 modules, each with its own requirements:

- Command:
  - Enable bilateral communication between user and Control module
  - Enable users to nagivate the rover
  - Plot a map of the locations of the obstacles encountered by the rover
- Control:
  - Enable bilateral communication channels between Command, Drive, Energy and Vision modules
- Drive:
  - Defines the operation of the two rover motors such as:
    - Speed control
    - Direction control
    - Turning method
  - Using the optical flow sensor, measure the distance travelled by the rover
- Energy:
  - Battery charge operation: Profile design, status estimation and melt/explosion prevention
  - Battery voltage balancing and range estimation
  - Implementing PV MMPT calculation algorithm
  - Integrating and testing solar charging system
- Vision:
  - Using the on-board camera detect, avoid and record the location of obstacles encoutered by the rover

[1] PMI: https://www.smartsheet.com/blog/demystifying-5-phases-project-management

1.2 Definition and Planning

The project team had a significant amount of freedom in designing and developing the rover system to meet the project requirements. The team had identified several design themes that guided the design and implementation choices made during the developement of the rover system:

- **Modularity**:

  Having taken into account that the project team spanned four countries with different time-zones and the time constraint of the project, the team felt it was important to incorporate modular design in the development of each rover modules.

  The approach meant each subsystem only had to ensure the pre-agreed connection interfaces such as WebSocket was compatible with the required modules. This was very advantageous due to the following:

  - Each module could independently develop sections of the rover system. This made the team much more dynamic and efficient.
  - The testing strategy [2] was more methodical and could occur early in stages, gradually leading up to a full rover system test.
  - No unnecessary meetings. By reducing the number of meetings the team had, it meant less time was wasted on arranging a time suitable for three time-zones and team meetings were more productive.

- **Scalability**:

  During the first meeting, the team was not certain as to the exact features that are desirable in a rover system. Due to this reason, scalability was a critical consideration factor and gave the team a very flexible approach to the rover system.

  An example would be the MongoDB database implemented by Command. MongoDB is a type of "NoSQL" database that is not as restrictive as traditional SQL databases which allowed the team to store new types of data without having to redesign the database model.

- **Open-source**:

  Where possible, the team opted to use well-supported open-source development packages such as the FastAPI framework. This complimented modularity and scalability themes by ensuring the interfaces are industry-standard and could be easily modified to expand its capabilites.

  Being well-supported, there is ample documentation to support the development and the codebase is well-designed. This meant that the team could reduce the number of unknown bugs, decrease development time and ensure a high-quality codebase.

- **Minimalism**:

  Due to the open-ended nature of the project, the team did not want to limit the scope of their design but, at the same time, did not want to risk a bloated and inefficient rover system due to feature creep and badly integrated modules.

  The team determined any design choice would need to prioritise efficiency and scalablity. All libraries are to be lightweight and only need to support the required features.

[2] Testing was managed by the Integration module

1.3 Performance and Control

1.3.1 Project timeline

[Timeline chart image]

1.3.2 Gnatt chart

[Gnatt chart image]

1.3.3 Team communication

[Communication heirarchy chart]

Due to the remote collaborative nature required, every member was encouraged to design and implement with a standardised minimalistic approach [3] *such as concise code documentation, avoiding too many dependencies and consistent code revisions.*

[3] Minimalistic coding guidelines: https://dev.to/paulasantamaria/6-ways-minimalism-can-help-you-write-clean-code-45kp

2 Rover system design

2.1 Structural design

[Structural design diagram]

The team established the structural design of the rover system during the first week of the project timeline. The structural design is formed in three stages:

1. **Identifying the core module**:

   The Control module was designated as the core module of the rover system due the ESP32 board's numerous communication interfaces and on-board data processing capabilities. With the remote nature of the project, this also meant the Control module will take the leadership role and act as the "heartbeat" of the team - ensuring each module development was in sync with the planned project timeline and each communication interface was compatible with Control.

2. **Module connection**:

   With the role of Control established, the team could discuss how to structure the other modules that best took advantage of the ESP32's capabilities, ensured efficient operation in various environments and complimented the *scalability* design theme.

   A major design problem was the location of processing the data from sensors like the optical sensor. The team chose to do as much processing locally on the ESP32 because the connection with Command module servers naturally has a certain degree of latency and Control was best equipped module on the rover to handle the processing. The Control module can also pre-process data to minimise the amount sent to Command and reduce the end-to-end latency of the system. [**latency data**]

3. **Communication interface selection**:

   In the second stage, the Control module worked with each module to research and select the most suitable communication interface. This was critical to establish the interfaces early on to allow the modules to start development as soon as possible and asynchronously as mentioned with *modularity* design theme.

   The biggest concern was the type of connection between the Control module and Command server. The form of connection had to be energy efficient to compliment the *minimalism* design theme and be able to scale easily depending on the data generated from the rover. The final design choice was to use the WebSocket protocol since it was native to the FastAPI framework used by Command and allowed real-time updates to the user. Compared to the HTTP protocol, the WebSocket protocol has a higher performance rating at 83 ms on WebSocket and 107 ms on HTTP for an average single request [1].

2.2 Functional design

The team defined the rover system into three functional layers that each handled a different aspect of the system.

2.3 Intellectual property

3 Rover Submodules

3.1 Command

3.2 Control

3.3 Vision

3.3.1 Implementation problem

3.4 Drive

3.4.1 Movement

3.4.2 Speed Control

3.4.3 Position Control

3.4.4 Optical Sensor

3.4.5 Implementation problem

3.5 Energy

3.5.1 Battery Charge Profile Design

3.5.2 State of Charge

3.5.3 Battery Balancing Algorithms

3.5.4 PV MPPT Algorithms

3.5.5 Circuit Design

3.5.6 State of Health

3.5.7 Energy to Control interface

3.5.8 Implementation problem

4 Integration: Testing

5 References

[1] David Luecke. *HTTP vs Websockets: A performance comparison*. 2018. DOI: https://blog.feathersjs.com/http-vs-websockets-a-performance-comparison-da2533f13a77.