/

**Table of Contents**

[***Problem Statement***](#bookmark=id.76l2plnsobay) 3

[***System Requirements***](#bookmark=id.55p6x84v1y1r)4

Functional Requirements4

Non-Functional Requirements [*4*](#_heading=h.3znysh7)

Assumptions 4

[***Security Risks and Mitigations***](#bookmark=id.cjyitg3ss3wi) 5

C1 Define Security Requirements5

C2 Leverage Security Frameworks and Libraries 5

C3 Secure Database Access5

C4 Encode and Escape Data6

C5 Validate All Inputs6

C6 Implement Digital Identity6

C7 Enforce Access Controls7

C8 Protect Data Everywhere8

C9 Implement Security Logging and Monitoring8

C10 Handle All Errors and Exceptions8

[***Proposed System - Design and Architecture***](#bookmark=id.bnkx587b0ni3)10

Ground Control (Telemetry Data Layer)10

Space Platform (Data Collection layer)11

Command and Control Layer 12

Clustered Microservices 14

[***System Diagrams***](#bookmark=id.6g8pqikm0mn2)14

*Database ERDs* 14

[*Class Diagram*](#bookmark=id.lgnqiapqyspd)16

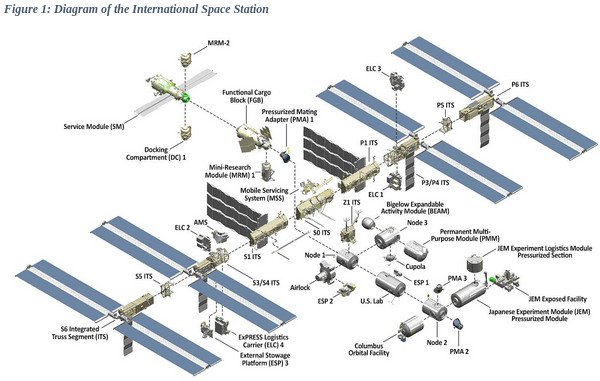
[***Tooling***](#bookmark=id.fr3frekukaum)17

[***References***](#bookmark=id.52qte4pzpa64)*18*

[**Appendices**](#bookmark=id.unybj7oyy0iw)20

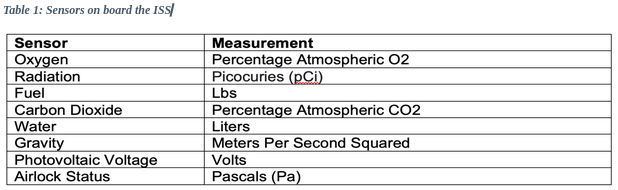




******

(2021, NASA)

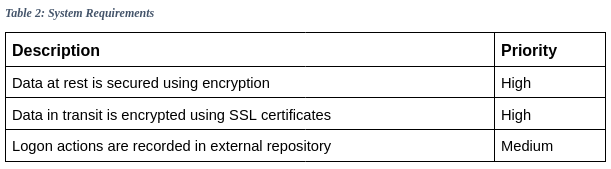
Due to limited physical access to the structure and modules aboard the ISS, data from the station's sensors need to be transmitted to mission control permitting ground support services to monitor the condition of the station. Sensors include:



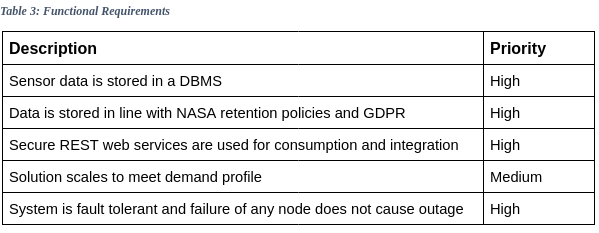
*(2018, Malesky) (2016,* Stenzel*)*

We propose to build a data-hub to collect data from the sensors which is stored locally and then transmitted to ground control in a single burst transmission as radio bandwidth is limited. [[**Appendix 1**](#bookmark=id.unybj7oyy0iw)]

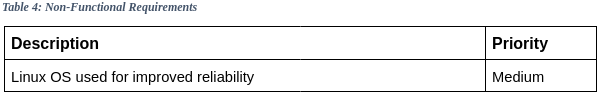
****

******



******

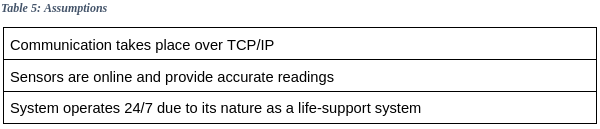


******

(Anthony, 2001) (OpenSCAP, 2022)



We have made a number of assumptions about the system:

******

****

Due to the life-supporting nature of the data, it is important that our solution addresses the security triad of confidentiality, integrity, and availability. We have used the OWASP Proactive Control methodology to identify potential risks and their mitigations (OWASP, n.d.).



“Effective security of network and information systems should be driven by organisational management and corresponding policies and practices.” (NCCS 2021)

We secure access to our system by:

1. Log Analysis — checking who is attempting to access the system and locking out unverified users.
2. Firewalls and network monitoring — blocking access to various parts of the network

****

We use Github’s Dependabot service to identify new versions of libraries we leverage, and automatically update our development branch. Any known vulnerabilities for libraries and tools we use are listed on page 17. Additionally static code analysis using Bandit and SonarQube is conducted to identify potential security issues during development.

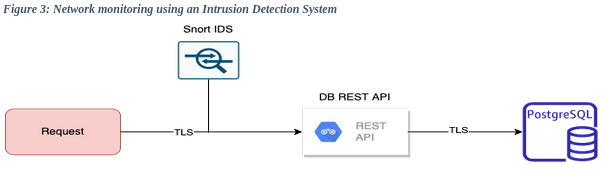
****

Data integrity is enforced through three methods:

1. **Access restricted secure API.** HTTPS requests include a custom header value X**-ACCESS-TOKEN** containing a JWT encoded token (Jones, 2015). Requests without this token are refused and flagged as suspicious.



1. **IP address restrictions.** We limit access to a pool of permitted addresses defined in the Web Application Firewall that sits between our CDN and API endpoints.
2. **Network monitoring**. Requests are completed only if the requesting IP is on our IP address white list, stored in the application. We use Snort intrusion detection system to monitor network traffic and alert us to any suspicious behaviour.



****

Client and server data is checked at each step in the system through the use of features like escaping single quotes and custom regex expressions to prevent SQL injection attacks.

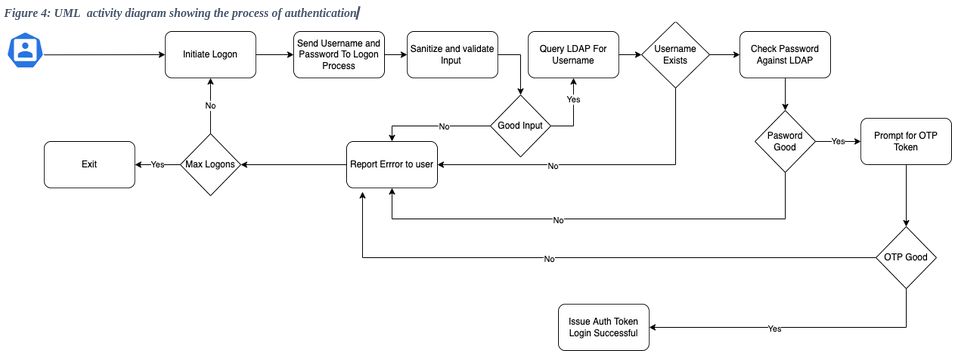
****

All user inputs are validated by the application using a series of Regex checks to ensure that the input is in the expected format.

****

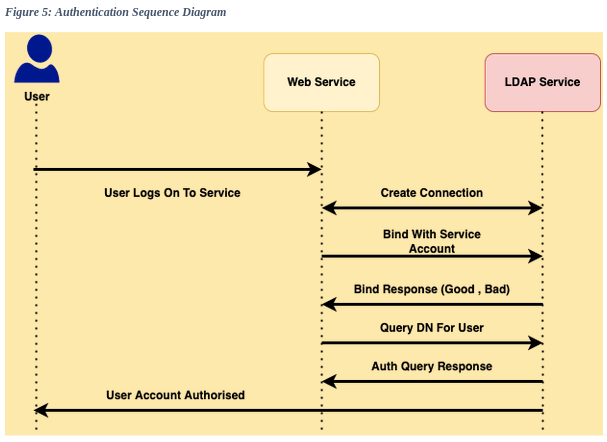
Accounts are stored in an LDAP Database. Identity is verified using passwords enforced by system group policies, ensuring passwords meet best practice in complexity and prevents users from utilising known common passwords.

Passwords are only one step in securing an account. Microsoft recently found that implementing MFA increases an account's security by 99.9% (Weinert, 2019). We use the python **PY-OTP** library to provide a OTP code, as described in the sequence diagram in Figure 4 overleaf.



****

User security is provided by separation between the application tier and the security tier. User accounts are stored in an LDAP Database with security GPOs applied defining access rights and account lockout policies. The process of authentication is shown in Figure 5. The Linux infrastructure is configured with fail2ban software to lock access if non authorised access is attempted.

****

****

Data protection Is implemented by two main tenants,

1. End-to-end encryption via SSL certificates, securing all data while in transit.
2. Encrypted-at-rest model. Data is stored in an encrypted state when not being transferred across the network.

****

Security logging is a three part process:

1. VM Hosts forward security events to a central server for alerting and analysis.
2. Network Traffic is monitored by an IDS to identify security threats.
3. FIM (File Integrity Monitoring) is enabled on key system files.

****

Complete exception handling is provided by the software, and tested by unit tests and user acceptance tests. The microservice design ensures that service is maintained by another pod taking over from any failed instance.

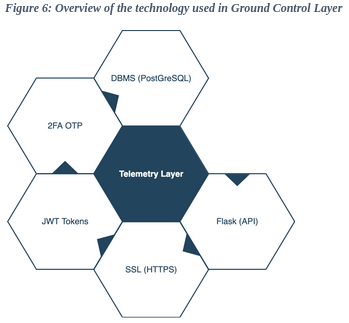
****

Our “High Availability” solution, once deployed, cannot be taken offline by the failure of a single component or service. We deliver a robust mechanism for the capturing and analysis of data, using redundant clustered microservices and fault tolerant storage.

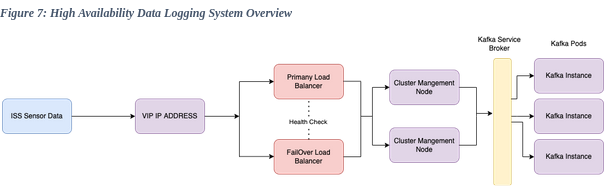
Failover technology ensures 24/7 availability. Services consist of three functional nodes permitting for the failure of any two nodes without affecting delivery.



Responsible for replication of data captured from ISS sensors and stored on ISS local storage, this layer enables the download of data to mission control, communicating with the ISS via fully authenticated and encrypted REST API’s.

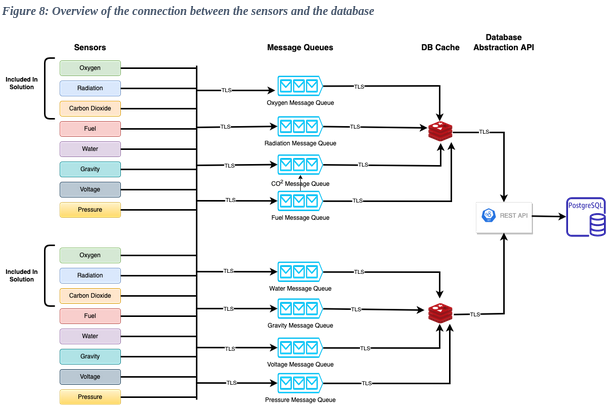
******

Data from the ISS sensors is ingested by a Kafka message queue service. The data is analysed based on a defined rule set. Should the data fall outside the expected range for the sensor type, mission control is alerted. The data is archived to a local DBMS.



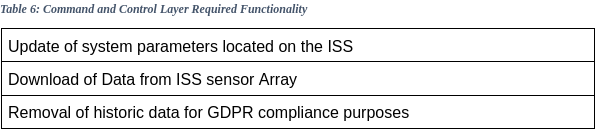


The Data Collection layer is designed using container architecture to allow for maximum horizontal scaling of the solution (Nguyen et al., 2020).

******



Permits ground control to change parameters and toggle sensors through the use of APIs. The functionality is detailed in table 6 and an overview of the system is described in Figure 8a.



******

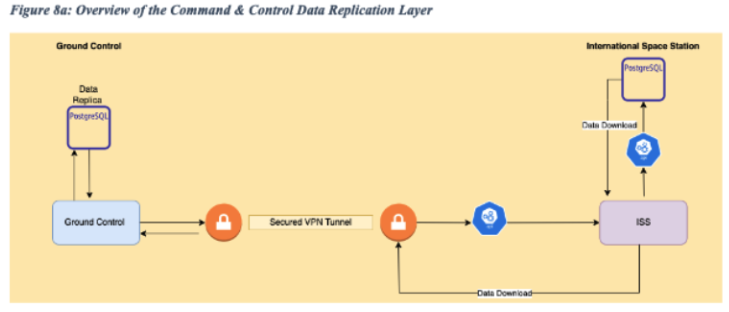
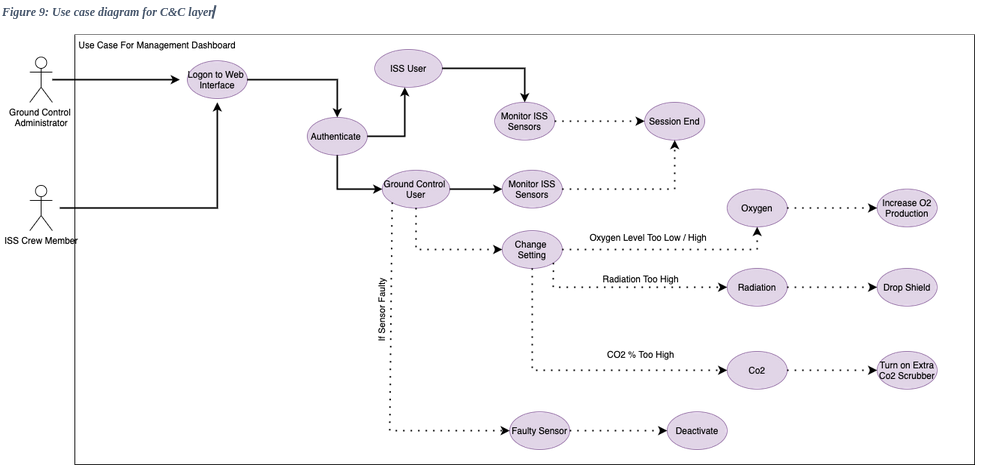
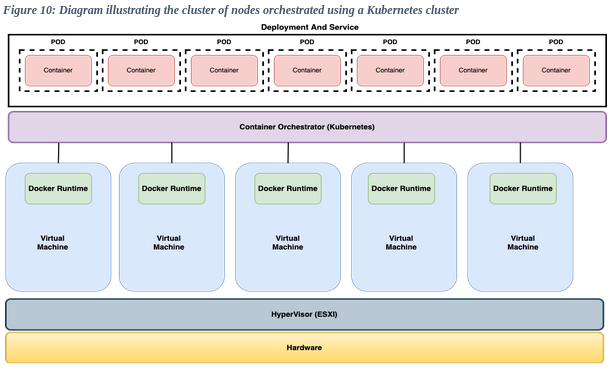
****

Figure 9 overleaf illustrates a use case diagram for the command and control layer dashboard.

******

# 

# Microservices are used to deliver the solution this provides high availability and scaling. (Nguyen et al., 2020). After investigating functionality and ease of deployment we decided to implement the core HA components using a multi-node Kubnetetes cluster, as depicted in Figure 5 (Vayghan et al., 2018).

******

****

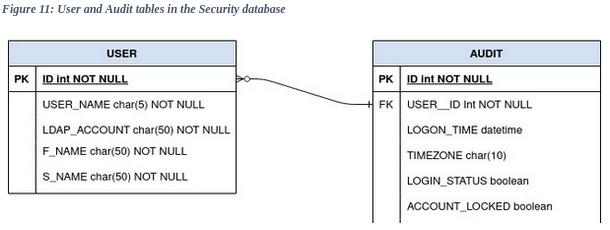
Our PostgreSQL SQL database is normalised and split into two main areas:

* Security and Authentication
* Data Logging

****

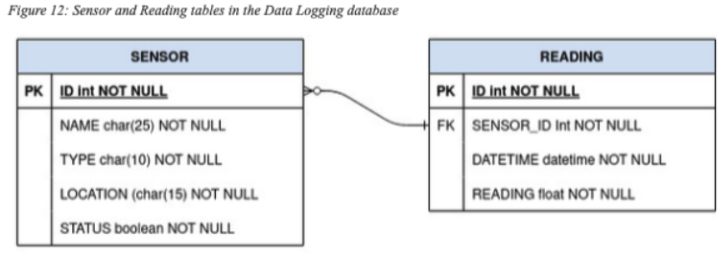
There are two tables: USER, storing account details and AUDIT, which stores logon

and logoff events.

******

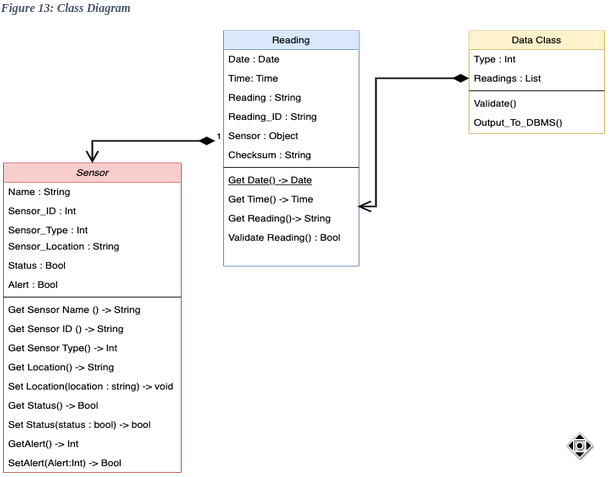


There are two tables: SENSORS, for the details of each sensor, and READINGS to store the data read from each sensor.



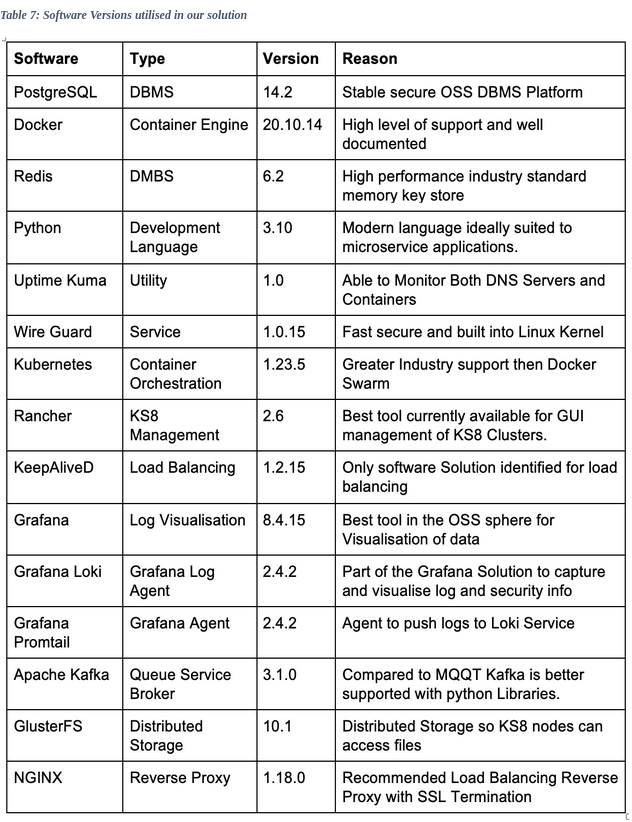
****

There are three main classes in the software, detailed in Figure 13.

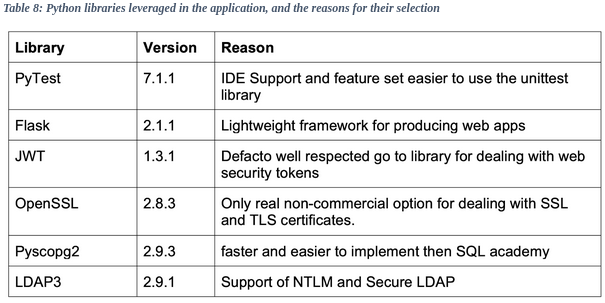
****

****

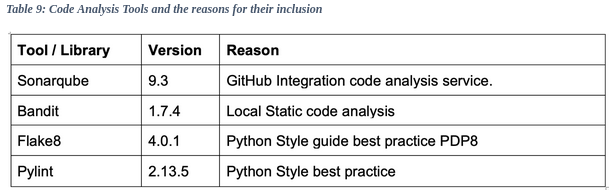
Below is a table listing the tools and versions we will use for the application, as well as the reason for each one’s choice.

******

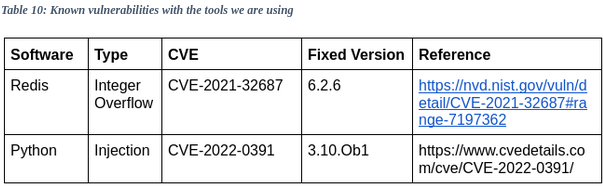
We will use the following Python libraries:

******

We will use the following static code analysis tools to ensure our code follows Python best practices, and reduce the possibility of security issues.

******

We have checked the above tools for known vulnerabilities. Any CVEs that remain unaddressed in the versions we are using are listed below:

******

****

****

Anthony**,** S.(2013) International Space Station switches from Windows to Linux, for improved reliability. Available from: <https://www.extremetech.com/extreme/155392-international-space-station-switches-from-windows-to-linux-for-improved-reliability> [Accessed 1/4/2022].

Anon (2019) MQTT Version 5.0. [Online]. Available from: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html [Accessed 12/4/22].

Anon (n.d.) MQTT QoS: Understanding Quality of Service. Available from: https://assetwolf.com/learn/mqtt-qos-understanding-quality-of-service [Accessed 25/3/2022].

Anon (2020) Kubernetes Concepts Available from: <https://kubernetes.io/docs/concepts/> [Accessed 11/3/2022].

## Anon (2022) Guide to the Secure Configuration of Ubuntu 20.04 Canonical Ubuntu 20.04 LTS Security Technical Implementation Guide (STIG) V1R1 Available from: https://static.open-scap.org/ssg-guides/ssg-ubuntu2004-guide-stig.html

Nasa (2020). NASA Records Retention Schedules. Available from: <https://nodis3.gsfc.nasa.gov/NPR_attachments/NRRS_1441.1A.pdf>[Accessed 11/3/2022].

Docker (n.d.) Docker Available from*:* <https://docs.docker.com/engine/swarm/>

Malesky M. (2018) Minimum Oxygen Concentration for Human Breathing. Available from: <https://sciencing.com/minimum-oxygen-concentration-human-breathing-15546.html> [Accessed 6/4/2022].

Nasa (2021) International Space Station facts and figures. Available from: <https://www.nasa.gov/feature/facts-and-figures> [Accessed 1/4/2022].

*NCSC (2021) NCSC* Governance. *Available from:* <https://www.ncsc.gov.uk/collection/caf/caf-principles-and-guidance/a1-governance> [Accessed 12/4/22].

NCSC (2019) Harden Ubuntu Linux. Available from: <https://www.ncsc.gov.uk/collection/device-security-guidance/platform-guides/ubuntu-lts> [Accessed 12/4/22].

OWASP (n.d.) OWASP Proactive Controls. Available from: <https://owasp.org/www-project-proactive-controls/> [Accessed 14/2/22].

So, A.(2018) CO2 on the International Space Station - Magnitude.io. Available from: <https://magnitude.io/co2-on-the-international-space-station/>[Accessed 12/3/22].

Weinert, A. (2019). Your Pa$$word Doesn’t Matter. Available from: <https://techcommunity.microsoft.com/t5/azure-active-directory-identity/your-pa-word-doesn-t-matter/ba-p/731984> [Accessed 12/3/22].



Anon (2001). Federal Information Processing Standards Publication 197 Announcing the ADVANCED ENCRYPTION STANDARD (AES). [Online]. Available at: http://csrc.nist.gov/csor**.**<http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf>

Jones, M. (2015). JSON Web Algorithms (JWT). [Online]. Available at: doi:10.17487/RFC7518.

Nguyen, T.-T., Yeom, Y.-J., Kim, T., Park, D.-H., Kim, S. (2020) ‘Horizontal Pod Autoscaling in Kubernetes for Elastic Container Orchestration’, Sensors, 20(16), 4621, available: <http://dx.doi.org/10.3390/s20164621>.

Stenzel C. (2016) Deployment of precise and robust sensors on board ISS-for scientific experiments and for operation of the station. Anal Bioanal Chem. 2016;408(24):6517-6536. doi:10.1007/s00216-016-9789-0

Vayghan, L. A., Saied, M. A., Toeroe, M. and Khendek, F., (2018) "Deploying Microservice Based Applications with Kubernetes: Experiments and Lessons Learned," 2018 IEEE 11th International Conference on Cloud Computing (CLOUD), 2018, pp. 970-973, doi: 10.1109/CLOUD.2018.00148**.**

## 

****

# ISS Speed of Radio Transmission

Rearranged to

Speed constant is calculated by or 299792458 m/s time taken to communicate with the ISS would be

Which results in a transmission time of **0.001334** seconds this assumes data is beamed directly from the ISS to a ground station which is not the case

The ISS is at an elevation of 400km the data covers a much greater distance to reach Earth. The ISS transmits the signal to a satellite positioned as high as 35,786km. Only from there can it reach ground space communication stations. This means the total distance covered by data from the ISS and the reply signal sent back is about 150,000 kilometres.

Which results in a true transmission time of **0.5003** Seconds **500.3** milliseconds