# **Analysis of Telemetry System for a Simulated Spacecraft**

**1. CCSDS Packet Utilization Standard (PUS) for Low-Rate Telemetry**

The Consultative Committee for Space Data Systems (CCSDS) has established recommendations for packet telemetry and telecommand to standardize the transport of data between user applications on the ground and application processes onboard spacecraft.1 The Packet Utilization Standard (PUS), initially issued by the European Space Agency (ESA) in 1994 and derived from CCSDS recommendations, aims to promote standardization and reuse of onboard and ground systems by defining a standard application-level interface.1 This standard provides a set of "standard" packet-based services by prescribing the service model and the structures of associated telemetry and telecommand packets, allowing missions to choose and implement services that align with their specific needs.1

While CCSDS recommendations and the PUS address the end-to-end transfer of data, they do not dictate the internal structure or content of the packets.2 The PUS defines application processes (APs) as onboard entities capable of generating telemetry source packets and receiving telecommand packets, uniquely identified by an application process identification (APID).1 These APs can control other onboard elements, and their implementation can vary across software, firmware, or hardware, without restrictions imposed by the PUS.1 The PUS currently assumes the use of CCSDS Telemetry and Telecommand packets but could be adapted for other communication mechanisms.1

The CCSDS Packet Telemetry Service Specification establishes a common framework for spacecraft-to-ground data communication within missions that are cross-supported between agencies.4 This specification allows implementing organizations to develop compatible derived standards for both flight and ground systems. Derived agency standards may implement subsets of the optional features or incorporate features not explicitly addressed by the CCSDS recommendation.4 It is important to note that the CCSDS Recommendation for Packet Telemetry has formally removed the concept of telemetry segmentation.1 Instead, grouping flags in the telemetry source packet header can indicate membership in a group of related packets from the same APID.1 For large data blocks, the ESA PUS identified a Large Data Transfer service to provide a more secure mechanism for conveying a series of packets.1

The CCSDS Space Packet comprises a predefined header, an optional secondary header, and a payload, with the standard not defining the structure or length of the payload.3 The Packet Utilization Standard builds upon the CCSDS Space Packet by providing a standard set of application services for managing onboard parameters, reports, command execution schedules, and device management.3 Custom PUS services can also be defined for mission-specific functionalities, with service IDs higher than 127.3 Practical application of PUS and CCSDS involves connecting the communication layer to the application code, raising considerations about how the PUS layer interacts with the application and how sensor readings are requested and assembled into PUS reports.3

**2. LoRa Communication Technology for Telemetry**

LoRa (Long Range) is a wireless communication technology designed for long-range, low-power, and low-data-rate communications, making it suitable for Internet of Things (IoT) networks.6 It utilizes Chirp Spread Spectrum (CSS) modulation, which spreads the signal over a wider bandwidth, enhancing noise immunity and enabling communication over long distances with low power.7 LoRa technology typically operates in unlicensed sub-gigahertz frequency bands, including 433, 868, and 915 MHz.9

The specific characteristics of the LoRa communication in this simulated spacecraft include a carrier frequency of 433.0 MHz, a bandwidth of 125.0 kHz, a spreading factor of 12, a coding rate of 6, and a transmit power of 20 dBm. The choice of 433 MHz generally offers a longer communication range compared to higher frequencies, with potential ranges extending to 15-20 km in rural areas and 2-5 km in urban environments.11 This frequency band is commonly used for short-range and low-power communication in various regions.12

The bandwidth of 125.0 kHz is one of the standard options for LoRa, offering a balance between data rate and sensitivity.13 A narrower bandwidth generally results in a slower transfer rate but improved range.15 The spreading factor (SF) of 12 is the highest available, providing the longest possible range and the best receiver sensitivity but also the lowest data rate and the longest time on air.13 Each step increase in spreading factor doubles the time on air for the same amount of data.13 The coding rate of 6 indicates the level of forward error correction applied to the data. LoRa coding rates typically range from 5 to 8, with a higher coding rate providing more error correction at the cost of reduced data rate.15 A coding rate of 6 implies a certain level of redundancy to improve link reliability in the presence of interference.15 The transmit power of 20 dBm (equivalent to 100 mW) is relatively high for LoRa, potentially maximizing the communication range achievable with the chosen parameters.18 However, higher transmit power also increases power consumption.19

Considering these characteristics, LoRa technology appears suitable for transmitting telemetry data from a simulated spacecraft, particularly if the priority is long-range communication and low power consumption, even at the expense of a lower data rate. The specific parameters chosen (433 MHz, 125 kHz, SF12, CR6, 20 dBm) suggest an emphasis on maximizing the link budget and robustness for reliable data transmission over potentially long distances.

**3. Reed-Solomon Forward Error Correction (RS-FEC) RS(255, 223)**

Reed-Solomon (RS) codes are block-based forward error correction codes widely used in digital communications and storage to correct errors that occur during transmission or storage.20 An RS code is specified as RS(n, k) with s-bit symbols, where n is the codeword length, k is the number of data symbols, and n-k is the number of parity symbols.20 The RS(255, 223) code with 8-bit symbols (bytes) takes 223 data bytes and adds parity symbols to create a 255-byte codeword.20

The number of parity symbols determines the error correction capability of the code. For RS(255, 223), there are 255 - 223 = 32 parity bytes. A Reed-Solomon decoder can correct up to t symbol errors in a codeword, where 2t = n-k.20 In the case of RS(255, 223), 2t = 32, so t = 16. This means the decoder can correct any 16 symbol errors (byte errors) within the 255-byte codeword.20 This capability is particularly effective against burst errors, where a series of consecutive bits or bytes are received in error.20

Most RS error correction schemes are systematic, meaning the original data is included directly in the encoded codeword, with the parity symbols appended.20 In the context of the simulated spacecraft, implementing RS(255, 223) would involve taking 223 bytes of telemetry data, encoding them by adding 32 parity bytes, and then transmitting the resulting 255-byte block. At the receiver, the RS decoder would use the parity information to detect and correct up to 16 byte errors that may have occurred during transmission, thus enhancing data reliability.23 If the amount of data to be transmitted is less than 223 bytes, padding (often with zeros) can be added to reach 223 bytes before encoding, and this padding is removed after decoding.25

CCSDS specifies the use of RS(255, 223) encoding with a specific field generator polynomial (1 + X + X<sup>2</sup> + X<sup>7</sup> + X<sup>8</sup>), a code generator with the first consecutive root at 112, and a primitive element of 11.26 Implementing RS-FEC in the provided Arduino code would likely involve using a software library that supports these CCSDS-specified parameters.27 These libraries handle the complex mathematical operations required for encoding and decoding, making it feasible to incorporate robust error correction into the telemetry system.

**4. Sensor Analysis for Spacecraft Telemetry**

* **BME280 (Environmental Sensor):** The BME280 is a combined digital sensor capable of measuring temperature, humidity, and barometric pressure.28 It offers high accuracy, low power consumption, and small size, making it suitable for mobile and embedded applications.30 The sensor provides digital output through I2C and SPI interfaces.28 Key functionalities include a temperature measurement range of -40 to +85 °C with an accuracy of ±0.5 °C, a humidity measurement range of 0 to 100% RH with an accuracy of ±3%, and a pressure measurement range of 300 to 1100 hPa with an accuracy of ±1 hPa in the typical operating range.28 The BME280 finds applications in context awareness, fitness monitoring, home automation, Internet of Things (IoT) devices, GPS enhancement, indoor navigation, weather forecasting, and vertical velocity indication.28 In the simulated spacecraft, this sensor would provide valuable data about the internal environment, such as temperature and humidity, and potentially infer information about the external environment through pressure readings.
* **QMC6310 (Magnetometer):** The QMC6310 is a three-axis magnetic sensor designed for applications like e-compass, map rotation, gaming, and personal navigation in mobile and wearable devices.33 It integrates magnetic sensors and a signal conditioning ASIC, offering low noise, high accuracy, low power consumption, and temperature compensation.33 The sensor provides 16-bit digital output representing the magnetic field strength along the X, Y, and Z axes via an I2C interface.33 It enables a compass heading accuracy of 1° to 2° and has a wide magnetic field range of ±30 Gauss.33 Typical applications include compass heading determination, magnetic anomaly detection, and orientation sensing.34 In the simulated spacecraft, the QMC6310 could be used for attitude determination by measuring the Earth's magnetic field (if within range and not significantly distorted by spacecraft components) or for detecting magnetic anomalies in space.
* **QMI8658 (IMU):** The QMI8658 is a complete 6D MEMS inertial measurement unit (IMU) that incorporates a 3-axis gyroscope and a 3-axis accelerometer.36 It features low noise, low latency, and wide bandwidth for precise motion sensing.36 The IMU supports multiple host interfaces, including I3C, I2C, and SPI.36 Key functionalities include programmable data rates and filtering, a large FIFO buffer for data buffering, integrated motion detection features (pedometer, tap, any-motion, no-motion, significant-motion), and wide dynamic ranges for both the gyroscope (±16°/s to ±2048°/s) and the accelerometer (±2 g to ±16 g).36 It also includes an embedded temperature sensor and low power modes.36 Typical applications span smartphones, game controllers, robotic vacuums, e-bikes, and automotive security systems.36 For the simulated spacecraft, the QMI8658 would be crucial for attitude control, providing data on angular rates and linear accelerations, which are essential for maintaining the desired orientation and tracking any changes in motion.
* **GPS Module:** A GPS module receives signals from a constellation of satellites to determine its precise location (latitude and longitude), altitude, speed, and the current time.39 These modules are widely used in navigation, tracking, and surveying applications.40 In telemetry applications, GPS modules provide valuable location and time information along with other sensor data, enabling the tracking of mobile assets and the correlation of data with specific locations and times.40 For the simulated spacecraft, a GPS module could provide accurate positional data, which is essential for tracking the spacecraft's orbit and understanding its trajectory. It also provides a highly accurate time source that can be used to timestamp telemetry data. However, the effectiveness of a GPS module in a spacecraft depends on the altitude and mission profile, as standard GPS signals may not be available at very high altitudes or in deep space.43

**5. Spacecraft Telemetry Systems: Principles and Practices**

Spacecraft telemetry systems serve the critical purpose of establishing a communication link between a spacecraft and ground control for real-time monitoring and control.44 This allows mission teams to assess the spacecraft's health, status, and performance by receiving data from various onboard sensors and instruments.45 Telemetry is fundamental to the success of space missions, providing essential information for operational decisions, anomaly detection, fault diagnosis, and overall mission management.44 It also plays a vital role in gathering data for post-mission analysis and the improvement of future spacecraft designs.45

Common data types transmitted via spacecraft telemetry are extensive and varied.45 They broadly fall into categories such as spacecraft housekeeping data, which includes parameters like temperature, pressure, voltage, current, and the status of various subsystems.46 Orbit data, providing information about the spacecraft's position and trajectory, is also crucial.47 Payload data encompasses the scientific measurements and observations collected by onboard instruments.46 Additionally, telemetry can include data about the spacecraft's attitude (orientation), speed, acceleration, vibration, and the status of telecommand reception.45 The specific types of data transmitted are tailored to the objectives of each mission and the suite of instruments onboard.48

Standards like those developed by CCSDS are of paramount importance in spacecraft telemetry.51 These standards ensure interoperability between different space agencies and missions, facilitating cross-support and reducing the costs associated with developing unique communication protocols for each mission.51 CCSDS standards cover various aspects of space data communication, including packet telemetry protocols, RF and modulation systems, file delivery protocols, and security measures.51 They provide a common framework for data formatting, transmission, and reception, which is essential for efficient and reliable communication between spacecraft and ground stations.53 The use of CCSDS standards helps maximize the link budget, ensure data integrity through error correction and detection, and support autonomous spacecraft operations.51

In contrast to the basic telemetry concepts likely demonstrated in the simulated code, actual space missions like Voyager 1 employ highly sophisticated telemetry systems.56 Voyager 1, launched in 1977, utilizes a large high-gain antenna to communicate over vast distances with Earth via the Deep Space Network.56 Its telemetry system operates at frequencies in the S-band (around 2.3 GHz) and X-band (around 8.4 GHz) for downlink, with uplink communication in the S-band (around 2.1 GHz).56 The spacecraft has an onboard digital tape recorder to store data for later transmission when direct communication is not possible.56 The challenges encountered by Voyager 1, such as recent issues with its Flight Data System causing unreadable telemetry data and earlier problems with the attitude control system, highlight the complexity and the potential for anomalies in long-duration deep space missions.57 The immense communication time delay of over 22 hours each way for Voyager 1 also underscores the significant differences compared to a simulated system operating over much shorter distances.56 While the simulated system provides a valuable platform for understanding fundamental telemetry principles, the telemetry systems used in actual deep space missions involve far more advanced hardware, software, and operational protocols to ensure reliable communication across interplanetary distances and over decades of operation.

**6. Arduino Library Integration for Telemetry**

The RadioLib library for Arduino offers a comprehensive set of functionalities for controlling SX1262 LoRa transceivers.61 It allows for the initialization of the LoRa modem with specific parameters such as carrier frequency, bandwidth, spreading factor, coding rate, and output power.63 The setFrequency() function enables setting the carrier frequency within the range supported by the SX1262.17 The setBandwidth() function allows selection from a range of predefined bandwidth values suitable for LoRa communication.17 Similarly, the setSpreadingFactor() function permits choosing the desired spreading factor, which influences the data rate and communication range.17

RadioLib also handles packet transmission and reception through functions like transmit() and receive().61 The transmit() function can send data in various formats, including Arduino strings and byte arrays.69 The library provides status codes to indicate the success or failure of transmission attempts, including errors for invalid parameters or transmission timeouts.69 While blocking transmit and receive methods are available, RadioLib also supports interrupt-driven communication for more efficient processor utilization.61 For LoRa point-to-point communication, radio.setSpreadingFactor() can be directly used, although in LoRaWAN applications, the spreading factor might be managed by the LoRaWAN stack.72

The U8g2 library for Arduino provides extensive capabilities for displaying information on monochrome displays.73 It supports a wide variety of display controllers and communication interfaces like I2C and SPI.73 U8g2 allows drawing basic shapes, lines, and pixels, as well as rendering text using numerous fonts, with support for UTF-8 encoding in some cases.76 Functions like setFont() enable the selection of different fonts, while drawStr() and print() are used to display text at specified coordinates.77 For visualizing sensor readings and system status, U8g2 can be used to display numerical data, text labels, and simple graphical representations on a connected monochrome display. For floating-point sensor values, the sprintf() function can be used to format the output as a string before displaying it with U8g2.80

The TinyGPS++ library for Arduino plays a crucial role in parsing data from GPS modules.43 It takes the raw NMEA data stream from the GPS module and decodes it into a structured format that is easy to use in Arduino code.43 The library provides methods to extract specific information such as location (latitude and longitude) through the gps.location object, altitude via gps.altitude, date and time using gps.date and gps.time, and the number of satellites being tracked with gps.satellites.value().43 It also offers functions to check the validity and update status of the GPS data, ensuring the reliability of the information used in the telemetry system.43 By simplifying the process of working with GPS data, TinyGPS++ allows developers to easily integrate location and timing information into their Arduino projects.

**7. Conclusion and Recommendations**

The research indicates that the chosen technologies are generally suitable for a simulated spacecraft telemetry system. LoRa provides long-range, low-power communication capabilities, which are beneficial for transmitting telemetry data. The selected LoRa parameters (433 MHz, 125 kHz, SF12, CR6, 20 dBm) prioritize range and robustness, which might be desirable for a spacecraft application. Reed-Solomon RS(255, 223) FEC offers a robust error correction mechanism that can significantly enhance the reliability of the transmitted telemetry data, which is crucial in noisy communication environments. The identified sensors (BME280, QMC6310, QMI8658, and GPS module) provide a comprehensive set of data for monitoring the spacecraft's environmental conditions, attitude, motion, and position.

The integration of these technologies within an Arduino environment is facilitated by libraries like RadioLib for controlling the SX1262 LoRa transceiver, U8g2 for visualizing sensor readings and system status on a monochrome display, and TinyGPS++ for parsing data from a GPS module. These libraries abstract the complexities of the underlying hardware and protocols, making it feasible to develop a functional telemetry system.

However, there are potential challenges and considerations. Power consumption will be a critical factor for a spacecraft, and while LoRa is low power, the transmit power of 20 dBm and the continuous operation of sensors and the microcontroller need to be carefully managed. The data rate limitations of LoRa, especially with a high spreading factor like SF12, might restrict the amount and frequency of telemetry data that can be transmitted. The overhead introduced by RS-FEC (adding 32 parity bytes for every 223 data bytes) will also impact the effective data rate. Furthermore, the operational environment of a spacecraft, particularly regarding GPS availability in orbit, needs to be considered. Standard GPS signals may not be reliably accessible at high altitudes or in deep space.

For the simulated system, several recommendations can be made. When using RadioLib, it is advisable to explore non-blocking transmission methods to improve the efficiency of the Arduino. Careful consideration should be given to the trade-offs between LoRa parameters to optimize for the specific needs of the mission, balancing range, data rate, and power consumption. For visualizing sensor data with U8g2, using formatted output for numerical values will enhance readability. When working with GPS data via TinyGPS++, it is important to check the validity and update status of the data to ensure its reliability. If RS-FEC is implemented, a suitable Arduino library adhering to the CCSDS RS(255, 223) specifications should be chosen, and the impact of padding on shorter data packets should be considered.

Further research could focus on optimizing the LoRa parameters for the specific telemetry requirements of the simulated spacecraft, investigating different error correction codes and their suitability for the application, and exploring alternative methods for attitude determination and navigation if GPS is not a viable option in the intended operational environment. Investigating power management techniques for the Arduino and connected sensors would also be beneficial to ensure the longevity of the system.

**Key Tables:**

1. **LoRa Parameter Trade-offs**

| **Parameter** | **Higher Value** | **Lower Value** | **Impact on Data Rate** | **Impact on Range** | **Impact on Time on Air** |
| --- | --- | --- | --- | --- | --- |
| Bandwidth (kHz) | Higher (e.g., 500) | Lower (e.g., 125) | Higher | Lower | Lower |
| Spreading Factor | Higher (e.g., 12) | Lower (e.g., 7) | Lower | Higher | Higher |
| Coding Rate | Higher (e.g., 4/8) | Lower (e.g., 4/5) | Lower | Higher | Higher |

1. **BME280 Sensor Specifications**

| **Specification** | **Value** |
| --- | --- |
| Temperature Range | -40 to +85 °C |
| Temperature Accuracy | ±0.5 °C (within 0 to 65 °C) |
| Humidity Range | 0 to 100 %RH |
| Humidity Accuracy | ±3 % relative humidity |
| Pressure Range | 300 to 1100 hPa |
| Pressure Accuracy | ±1 hPa (within 950 to 1050 hPa, 0 to 30 °C) |
| Power Consumption (at 1 Hz) | 3.6 µA (for T, H, P) |
| Interface Options | I2C, SPI |

#### Works cited

1. arc.aiaa.org, accessed April 3, 2025, <https://arc.aiaa.org/doi/pdf/10.2514/6.2002-T5-06>
2. The ESA standard for telemetry and telecommand packet utilisation: PUS - NASA Technical Reports Server (NTRS), accessed April 3, 2025, <https://ntrs.nasa.gov/citations/19950011180>
3. What is CCSDS and ECSS PUS? - PUSopen, accessed April 3, 2025, <https://pusopen.com/ecss-pus>
4. Packet Telemetry Service Specification - CCSDS, accessed April 3, 2025, <https://public.ccsds.org/Pubs/103x0b2s.pdf>
5. Packet Telemetry - CCSDS, accessed April 3, 2025, <https://public.ccsds.org/Pubs/102x0b5s.pdf>
6. LoRaWAN Technology - MultiTech, accessed April 3, 2025, <https://multitech.com/all-products/technologies/lorawan/>
7. LoRa principle, communication technology, application scenario usage - PUSR, accessed April 3, 2025, <https://www.pusr.com/blog/LoRa-principle-communication-technology-application-scenario-usage>
8. Detailed explanation of the technical principles and characteristics of LoRa and LoRaWAN, accessed April 3, 2025, <https://ebyteiot.com/blogs/ebyte-iot-blog/detailed-explanation-of-the-technical-principles-and-characteristics-of-lora-and-lorawan>
9. What Is LoRa? What Are The Advantages And Features Of LoRa? - iotrouter, accessed April 3, 2025, <https://www.iotrouter.com/what-is-lora-what-are-the-advantages-and-features-of-lora/>
10. What are LoRa and LoRaWAN? - The Things Network, accessed April 3, 2025, <https://www.thethingsnetwork.org/docs/lorawan/what-is-lorawan/>
11. LoRa Module Buying Guide: Price,Range,Applications ... - Ebyte, accessed April 3, 2025, <https://www.cdebyte.com/news/902>
12. Difference between 433MHz and 915MHz for LoRaWAN networks - LoRa Antenna, accessed April 3, 2025, <https://www.loraantenna.com/whats-the-difference-between-433mhz-and-915mhz-for-lorawan-networks/>
13. Data Rate and Spreading Factor - Lab5e AS, accessed April 3, 2025, <https://www.lab5e.com/docs/lora/dr_sf/>
14. What is LoRa: The fundamentals - Medium, accessed April 3, 2025, <https://medium.com/@prajzler/what-is-lora-the-fundamentals-79a5bb3e6dec>
15. LoRa bitrate calculator and understanding LoRa parameters - unsigned.io, accessed April 3, 2025, <https://unsigned.io/understanding-lora-parameters/>
16. Spreading Factors | The Things Network, accessed April 3, 2025, <https://www.thethingsnetwork.org/docs/lorawan/spreading-factors/>
17. RadioLib: SX1278 Class Reference - GitHub Pages, accessed April 3, 2025, <https://jgromes.github.io/RadioLib/class_s_x1278.html>
18. Why is the LoRa module closer to the ground, the worse the signal - NiceRF, accessed April 3, 2025, <https://www.nicerf.com/news/lora-module-signal.html>
19. LoRa Module Range: Key Factors Affecting LoRa Communication\_Industry dynamics\_Blog\_ - Ebyte, accessed April 3, 2025, <https://www.cdebyte.com/news/622>
20. reed-solomon codes, accessed April 3, 2025, <https://www.cs.cmu.edu/~guyb/realworld/reedsolomon/reed_solomon_codes.html>
21. Reed–Solomon error correction - Wikipedia, accessed April 3, 2025, <https://en.wikipedia.org/wiki/Reed%E2%80%93Solomon_error_correction>
22. Reed–Solomon error correction - Simple English Wikipedia, the free ..., accessed April 3, 2025, <https://simple.wikipedia.org/wiki/Reed%E2%80%93Solomon_error_correction>
23. Forward Error Correction in 25G Fiber Optics - FS.com, accessed April 3, 2025, <https://www.fs.com/blog/enhancing-25g-fiber-optic-communication-with-advanced-fec-techniques-12881.html>
24. Understanding Forward Error Correction (FEC) in 100G Optical Networks - FS.com, accessed April 3, 2025, <https://www.fs.com/blog/forward-error-correction-fec-in-100g-data-transmission-463.html>
25. Understanding of Reed Solomon code - Signal Processing Stack Exchange, accessed April 3, 2025, <https://dsp.stackexchange.com/questions/71490/understanding-of-reed-solomon-code>
26. c# - CCSDS Reed Solomon Encoding - Stack Overflow, accessed April 3, 2025, <https://stackoverflow.com/questions/29267440/ccsds-reed-solomon-encoding>
27. Arduino-FEC/src/RS-FEC.h at master - GitHub, accessed April 3, 2025, <https://github.com/simonyipeter/Arduino-FEC/blob/master/src/RS-FEC.h>
28. www.bosch-sensortec.com, accessed April 3, 2025, <https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bme280-ds002.pdf>
29. Gravity: I2C BME280 Environmental Sensor (Temperature, Humidity, Barometer) SKU: SEN0236 - DigiKey, accessed April 3, 2025, <https://mm.digikey.com/Volume0/opasdata/d220001/medias/docus/2328/SEN0236_Web.pdf>
30. BME280 Integrated Environmental Unit - Bosch Sensortec, accessed April 3, 2025, <https://www.bosch-sensortec.com/media/boschsensortec/downloads/product_flyer/bst-bme280-fl000.pdf>
31. BME280 Bosch Sensortec - Mouser Electronics, accessed April 3, 2025, <https://www.mouser.com/ProductDetail/Bosch-Sensortec/BME280?qs=2OnyuXx6vpj2fK9HX7qb3g%3D%3D>
32. Gravity: I2C BME280 Environmental Sensor - Farnell, accessed April 3, 2025, <https://www.farnell.com/datasheets/3176138.pdf>
33. www.qstcorp.com, accessed April 3, 2025, <https://www.qstcorp.com/upload/pdf/202202/%EF%BC%88%E5%B7%B2%E4%BC%A0%EF%BC%8913-52-17%20QMC6310%20Datasheet%20Rev.C(1).pdf>
34. Examples with Suibo degzrobotics, accessed April 3, 2025, <https://wiki.degzrobotics.com/elektronik-kartlar/Kontrol-Kartlar%C4%B1/suibo-rp2040/example/>
35. PiicoDev Magnetometer QMC6310 | Buy in Australia | CE07937 - Core Electronics, accessed April 3, 2025, <https://core-electronics.com.au/piicodev-magnetometer-qmc6310.html>
36. files.waveshare.com, accessed April 3, 2025, <https://files.waveshare.com/upload/5/5f/QMI8658A_Datasheet_Rev_A.pdf>
37. QMI8658C Datasheet, accessed April 3, 2025, <https://qstcorp.com/upload/pdf/202202/QMI8658C%20datasheet%20rev%200.9.pdf>
38. Lesson 16 QMI8658C Datasheet PDF | PDF | Inertial Navigation System - Scribd, accessed April 3, 2025, <https://fr.scribd.com/document/694565649/Lesson-16-QMI8658C-datasheet-pdf>
39. Spektrum GPS Telemetry Sensor - e-Flite, accessed April 3, 2025, <https://www.e-fliterc.com/product/gps-telemetry-sensor/SPMA95871.html>
40. Technology and Development at the USDA Forest Service, Satellite/GPS Telemetry for Monitoring Lesser Prairie Chickens, accessed April 3, 2025, <https://www.fs.usda.gov/t-d/programs/im/satellite_gps_telemetry/wildlifetrackingtelementry.htm>
41. Finding Your Way With GPS : 9 Steps - Instructables, accessed April 3, 2025, <https://www.instructables.com/Finding-Your-Way-With-GPS/>
42. Telemetry Applications - Kantronics, accessed April 3, 2025, <https://kantronics.com/applications-and-support-resources/>
43. TinyGPS++ | Arduiniana, accessed April 3, 2025, <https://arduiniana.org/libraries/tinygpsplus/>
44. The History of Aerospace Telemetry | Dewesoft, accessed April 3, 2025, <https://dewesoft.com/blog/aerospace-telemetry-history>
45. Aerospace telemetry: How it works and why does it matter - Dewesoft, accessed April 3, 2025, <https://dewesoft.com/blog/how-aerospace-telemetry-works>
46. Spacecraft Telemetry, Tracking, & Command with Epsilon3, accessed April 3, 2025, <https://www.epsilon3.io/behind-the-console/spacecraft-telemetry-tracking-command>
47. Telemetry & Telecommand - European Space Agency, accessed April 3, 2025, <https://www.esa.int/Enabling_Support/Space_Engineering_Technology/Onboard_Computers_and_Data_Handling/Telemetry_Telecommand>
48. OSTM/Jason-2 Telemetry Data, 2008 to 2019, accessed April 3, 2025, <https://accession.nodc.noaa.gov/OSTM-Jason2-Telemetry>
49. What is Telemetry Data? How Does It Work, Benefits, Challenges, and Examples, accessed April 3, 2025, <https://edgedelta.com/company/blog/what-is-telemetry-data>
50. Jason-3 Telemetry data, 2015-present - Dataset - Catalog, accessed April 3, 2025, <https://catalog.data.gov/dataset/jason-3-telemetry-data-2015-present>
51. CCSDS Solutions | Southwest Research Institute, accessed April 3, 2025, <https://www.swri.org/markets/earth-space/space-research-technology/space-engineering/ccsds-solutions>
52. CCSDS: Enabling Interoperability and Efficiency in the Space Industry, accessed April 3, 2025, <https://newspaceeconomy.ca/2024/05/06/ccsds-enabling-interoperability-and-efficiency-in-the-space-industry/>
53. Traditionally, telemetry transmitted from spacecraft was formatted with a time-division multiplexing (TDM) scheme, where data it - SPIE, accessed April 3, 2025, <https://spie.org/samples/PM230.pdf>
54. CCSDS: Advancing with Technology - DLA, accessed April 3, 2025, <https://www.dsp.dla.mil/Portals/26/Documents/Publications/Journal/170101-DSPJ-05.pdf>
55. CCSDS Telemetry | Consultative Committee for Space Data Systems | NetAcquire, accessed April 3, 2025, <https://netacquire.com/ccsds/>
56. Voyager 1 - Wikipedia, accessed April 3, 2025, <https://en.wikipedia.org/wiki/Voyager_1>
57. NASA is Fixing its Link to Voyager 1 - Universe Today, accessed April 3, 2025, <https://www.universetoday.com/articles/nasa-is-fixing-its-link-to-voyager-1>
58. Voyager 1 Has Another Problem With its Computer System - Universe Today, accessed April 3, 2025, <https://www.universetoday.com/articles/voyager-1-has-another-problem-with-its-computer-system>
59. Engineers Working to Resolve Issue With Voyager 1 Computer – The Sun Spot, accessed April 3, 2025, <https://blogs.nasa.gov/sunspot/2023/12/12/engineers-working-to-resolve-issue-with-voyager-1-computer/>
60. Engineers Investigating NASA's Voyager 1 Telemetry Data - Jet Propulsion Laboratory, accessed April 3, 2025, <https://www.jpl.nasa.gov/news/engineers-investigating-nasas-voyager-1-telemetry-data/>
61. RadioLib: SX1262 Class Reference - GitHub Pages, accessed April 3, 2025, <https://jgromes.github.io/RadioLib/class_s_x1262.html>
62. jgromes/RadioLib: Universal wireless communication library for embedded devices - GitHub, accessed April 3, 2025, <https://github.com/jgromes/RadioLib>
63. Wireless Stick Lite V3 w/ RadioLib - Heltec Automation Technical Community, accessed April 3, 2025, <http://community.heltec.cn/t/wireless-stick-lite-v3-w-radiolib/14744>
64. Default configuration · jgromes/RadioLib Wiki - GitHub, accessed April 3, 2025, <https://github.com/jgromes/RadioLib/wiki/Default-configuration>
65. RadioLib: SX1268 Class Reference - GitHub Pages, accessed April 3, 2025, <https://jgromes.github.io/RadioLib/class_s_x1268.html>
66. RadioLib/examples/SX126x/SX126x\_Settings/SX126x\_Settings.ino at master · jgromes/RadioLib - GitHub, accessed April 3, 2025, <https://github.com/jgromes/RadioLib/blob/master/examples/SX126x/SX126x_Settings/SX126x_Settings.ino>
67. RadioLib: SX126x Class Reference - GitHub Pages, accessed April 3, 2025, <https://jgromes.github.io/RadioLib/class_s_x126x.html>
68. RadioLib: SX127x Class Reference - GitHub Pages, accessed April 3, 2025, <https://jgromes.github.io/RadioLib/class_s_x127x.html>
69. RadioLib/examples/SX126x/SX126x\_Transmit\_Blocking ... - GitHub, accessed April 3, 2025, <https://github.com/jgromes/RadioLib/blob/master/examples/SX126x/SX126x_Transmit_Blocking/SX126x_Transmit_Blocking.ino>
70. GP2Y1010AU0F/libs/RadioLib/examples/CC1101/CC1101\_Transmit/CC1101\_Transmit.ino · master · Tomáš Kroupa / IoT - Studenti · GitLab, accessed April 3, 2025, <https://gitlab.portabo.cz/tomas.kroupa/iot-studenti/-/blob/master/GP2Y1010AU0F/libs/RadioLib/examples/CC1101/CC1101_Transmit/CC1101_Transmit.ino>
71. Status Codes - RadioLib - GitHub Pages, accessed April 3, 2025, <https://jgromes.github.io/RadioLib/group__status__codes.html>
72. Followup: LoRaWAN: How to set spreading factor, output power, bandwidth? · jgromes RadioLib · Discussion #1172 - GitHub, accessed April 3, 2025, <https://github.com/jgromes/RadioLib/discussions/1172>
73. U8g2 - Arduino Documentation, accessed April 3, 2025, <https://docs.arduino.cc/libraries/u8g2/>
74. olikraus/u8g2: U8glib library for monochrome displays, version 2 - GitHub, accessed April 3, 2025, <https://github.com/olikraus/u8g2>
75. U8g2 for Seeeduino boards - Seeed Studio Product Catalog, accessed April 3, 2025, <https://www.seeedstudio.com/blog/2019/07/05/u8g2-for-seeeduino-boards/>
76. u8g2 - NodeMCU Documentation - Read the Docs, accessed April 3, 2025, <https://nodemcu.readthedocs.io/en/dev/modules/u8g2/>
77. u8g2reference · olikraus/u8g2 Wiki · GitHub, accessed April 3, 2025, <https://github.com/olikraus/u8g2/wiki/u8g2reference>
78. olikraus/U8g2\_for\_Adafruit\_GFX: Add U8g2 fonts to any Adafruit GFX, accessed April 3, 2025, <https://registry.platformio.org/libraries/olikraus/U8g2_for_Adafruit_GFX>
79. u8x8reference · olikraus/u8g2 Wiki · GitHub, accessed April 3, 2025, <https://github.com/olikraus/u8g2/wiki/u8x8reference>
80. ESP32 SPI - | Mbed, accessed April 3, 2025, <https://os.mbed.com/users/fpucher/code/ESP32/wiki/ESP32-SPI>
81. Formatting numbers in u8g2 · olikraus u8g2 · Discussion #2118 - GitHub, accessed April 3, 2025, <https://github.com/olikraus/u8g2/discussions/2118>
82. TinyGPSPlus | Arduino Documentation, accessed April 3, 2025, <https://docs.arduino.cc/libraries/tinygpsplus/>
83. mikalhart/TinyGPSPlus: A new, customizable Arduino NMEA parsing library - GitHub, accessed April 3, 2025, <https://github.com/mikalhart/TinyGPSPlus>
84. Decoding NMEA Data with TinyGPS++ - Lonely Binary, accessed April 3, 2025, <https://lonelybinary.com/en-nz/blogs/learn/decoding-nmea-data-with-tinygps>
85. Example Sketch: TinyGPS Serial Streaming - GPS Logger Shield Hookup Guide - SparkFun Learn, accessed April 3, 2025, <https://learn.sparkfun.com/tutorials/gps-logger-shield-hookup-guide/example-sketch-tinygps-serial-streaming>
86. TinyGPSPlus/src/TinyGPS++.h at master · mikalhart/TinyGPSPlus ..., accessed April 3, 2025, <https://github.com/mikalhart/TinyGPSPlus/blob/master/src/TinyGPS%2B%2B.h>
87. How not to read a GPS with TinyGPSplus - StuartsProjects, accessed April 3, 2025, <https://stuartsprojects.github.io/2024/09/21/How-not-to-read-a-GPS.html>