Ground-Based Testbed for CubeSat Communication Study

Ruth Akpalu
Department of Electrical and Computer Engineering
Old Dominion University, Norfolk, Virginia, USA
Email: rakpa001@odu.edu

***Abstract*—CubeSats are low-cost, modular satellites that remain vulnerable to cyberattacks due to limited onboard security and constrained computational resources. Many missions lack strong encryption or real-time fault recovery, making communication failures common. As demand for secure space-based communication grows, adaptive cybersecurity strategies for resource-constrained satellite platforms are increasingly necessary. This work presents a FlatSat-based testbed on a Raspberry Pi to investigate CubeSat communication security using GNU Radio and software-defined radios (SDRs). Radio-frequency monitoring experiments using an RTL-SDR receiver demonstrate the accessibility of wireless signal observation and highlight interception risks. Packet-level security mechanisms inspired by the CubeSat Space Protocol (CSP) and the Advanced Encryption Standard (AES) are implemented using Python and User Datagram Protocol (UDP), enabling encrypted data transmission between a simulated spacecraft and ground station. SparkFun environmental sensors provide real-time monitoring, with Python scripts used for data collection and analysis. GNU Radio flowgraphs are developed and evaluated across SDR platforms, including ADALM-Pluto, RTL-SDR, and USRP. Results highlight challenges in reliable data transfer, including synchronization loss, packet corruption, and framing mismatches. System performance is evaluated using latency, packet integrity, and error behavior. This paper provides a practical, low-cost framework for evaluating CubeSat communication security.**

I. Introduction

CubeSats have become increasingly popular in scientific, academic, and commercial space missions due to their low cost, modular design, and accessibility. These small satellites are widely used for applications such as Earth observation, environmental monitoring, and communication. Despite their advantages, CubeSats rely heavily on wireless communication systems to transmit telemetry and mission data between the spacecraft and ground stations. As the number of CubeSat deployments continues to grow, ensuring reliable and secure communication has become a critical concern.

Satellite communication primarily operates over radio frequency (RF) channels, which are inherently vulnerable to cyber threats such as interception, jamming, and spoofing [3]. Interception occurs when an unauthorized party listens to transmitted signals, potentially exposing sensitive data. The increasing availability of low-cost software-defined radios (SDRs) has made it easier to monitor and analyze wireless signals [16], further increasing the risk of unauthorized access [3].

Testing satellite communication systems in orbit is costly and difficult to modify once deployed [18]. As a result, ground-based platforms, commonly referred to as FlatSat systems, are used to emulate spacecraft hardware and communication pipelines in a controlled environment [17]. These testbeds allow researchers to evaluate communication performance, experiment with security mechanisms, and analyze system behavior under different conditions before deployment. However, there is a lack of accessible, low-cost platforms that integrate communication, security, and RF experimentation for CubeSat systems.

This work presents the development of a low-cost FlatSat-based communication testbed designed to investigate secure data transmission for CubeSat-inspired systems. The system utilizes a Raspberry Pi platform to simulate onboard data generation and transmission, while a ground station receives and processes the data. A layered approach is used to evaluate communication performance, beginning with RF signal monitoring using an RTL-SDR receiver to demonstrate the accessibility of wireless signal interception. A secure communication pipeline is then implemented using User Datagram Protocol (UDP) and Advanced Encryption Standard (AES-128) encryption to validate reliable packet transmission between the simulated spacecraft and ground station. In addition, software-defined radio experiments are conducted using GNU Radio with ADALM-Pluto, RTL-SDR, and USRP devices to explore radio-frequency data transmission and identify practical limitations in synchronization and decoding. Environmental sensors are also integrated to simulate telemetry data generation within the system.

The main contributions of this work include the development of a low-cost, reproducible FlatSat testbed for CubeSat communication research, the implementation and evaluation of encrypted UDP-based communication, and the experimental investigation of SDR-based transmission using multiple platforms. The results provide insight into the challenges of secure communication in resource-constrained systems and highlight important considerations for future CubeSat cybersecurity design.

The remainder of this paper is organized as follows. Section II reviews background concepts related to CubeSat communication systems and software-defined radios. Section III describes the architecture of the experimental testbed. Section IV presents the RF signal interception experiment.

Section V discusses the secure UDP communication implementation, and Section VI explores SDR-based transmission experiments. Experimental results and discussion are presented in Section VII, followed by conclusions and future work.

II. Background and Related Work

This section presents background information on CubeSat communication systems, software-defined radios, and secure communication methods. These areas are important for understanding the challenges and design considerations involved in developing a secure communication testbed.

A. CubeSat Communication Systems

The use of CubeSats in space missions has expanded rapidly due to their low cost, modularity, and accessibility. CubeSats are standardized nanosatellites that are often launched in clusters for Earth observation, technology demonstration, or scientific research [1]. Their accessibility led to their usage in space exploration, enabling universities and smaller institutions to conduct space-based experiments. In [2], the authors examine CubeSat utilization and present statistics on mission outcomes and design types, emphasizing the global involvement in CubeSat development across multiple countries.

As a result, they remain vulnerable to cyberattacks, particularly in the communication subsystem, which typically relies on unencrypted or minimally protected radio frequency (RF) signals. In [3], the authors discuss security issues related to CubeSats, focusing on both the ground station and communication links, emphasizing the importance of securing both ends of the communication chain. One major vulnerability is interception, where a third party passively listens to a satellite's downlink without authorization, potentially exposing sensitive data. Many CubeSat missions lack robust encryption protocols due to constraints on processing power, memory, and energy consumption [4].

Additional threats include jamming and spoofing. Jamming occurs when an attacker intentionally transmits interference on the same frequency to disrupt communication. In [5], the authors explore cybersecurity challenges in satellite communication systems and highlight how CubeSat limitations are vulnerable to such attacks. Similarly, [6] outlines mechanisms for detecting and mitigating jamming in modern communication systems. Spoofing represents another critical threat, where adversaries transmit forged signals to imitate legitimate commands. In [7], researchers demonstrate how spoofing attacks can succeed when authentication mechanisms are weak, and highlight how low-cost SDRs can be used to carry out such attacks. These vulnerabilities emphasize the need for secure and resilient CubeSat communication systems.

B. Software Defined Radios (SDR)

Software-defined radios enable flexible implementation of wireless communication systems by shifting physical layer functionality from hardware to software, allowing modulation, frequency selection, and signal processing to be reconfigured dynamically [14]. By implementing signal processing in software rather than hardware, SDRs allow researchers to design, modify, and test communication protocols in real time. This makes them particularly useful for CubeSat communication research, where system constraints require adaptable and efficient solutions. The use of open-source tools and SDRs is becoming more widespread for simulating and evaluating satellite communication systems. GNU Radio and SDR platforms such as the ADALM-Pluto enable researchers to model realistic RF environments, including interference and signal degradation, within controlled lab settings. Prior studies [10], [11] have demonstrated that SDR-based testbeds are effective for analyzing communication performance and system resilience in CubeSat applications. Common SDR platforms used in research include the RTL-SDR, ADALM-Pluto, and Universal Software Radio Peripheral (USRP). The RTL-SDR is a low-cost receiver widely used for signal monitoring and spectrum analysis, while the ADALM-Pluto supports both transmission and reception for full communication system development. Platforms such as the Universal Software Radio Peripheral (USRP), including the B200 model used in this work, provide wide frequency coverage and are commonly used for research and prototyping of wireless communication systems [14]. These platforms, when combined with tools such as GNU Radio, provide a flexible environment for developing and testing communication systems.

C. Secure Communication Methods

To address the vulnerabilities present in CubeSat communication systems, researchers have investigated lightweight cybersecurity protocols tailored for resource-constrained platforms. The CubeSat Space Protocol (CSP) is a modular, space-optimized network protocol stack designed to support onboard and ground communication [8]. While CSP is efficient, it does not natively support strong encryption, which limits its ability to protect sensitive data. To enhance communication security, encryption algorithms such as the Advanced Encryption Standard (AES) have been integrated into CubeSat communication systems. A study in [9] explores enhancing CSP by incorporating AES, demonstrating improved data confidentiality and integrity with minimal computational overhead. These approaches are particularly important for CubeSats, where processing power and energy consumption must be carefully managed. In addition to encryption, packet-based communication methods are used to structure transmitted data and support error detection, integrity verification, and efficient data handling. Open-source platforms such as COSMOS (OpenC3) further support secure communication testing by providing tools for telemetry monitoring, command execution, and system integration. COSMOS has been applied in academic CubeSat programs [12], [13] to support testing and simulation of communication systems. While these tools and protocols provide promising capabilities, there remains a notable gap in the availability of integrated, low-cost platforms specifically designed to simulate cyber threats and evaluate CubeSat communication security in realistic conditions. Most

existing research focuses on individual threats or high-cost implementations, highlighting the need for an accessible and reproducible testbed. This work builds upon existing research by integrating SDRs, encryption techniques, and a FlatSat platform into a unified experimental system.

III. SYSTEM ARCHITECTURE

This section describes the design and implementation of the FlatSat-based communication testbed developed to evaluate secure CubeSat communication. The system consists of a Raspberry Pi-based space node, a ground station, and a communication pipeline that supports both network-based and software-defined radio (SDR)-based transmission. The architecture integrates data generation, encryption, packet transmission, and reception to simulate a simplified CubeSat communication system in a controlled environment. The overall system architecture is illustrated in Fig. 1.

A. Space Node Hardware

The space node is implemented using the Raspberry Pi 5, which functions as a simplified onboard computer representative of a CubeSat flight system [19]. This platform provides sufficient computational capability to support real-time data acquisition, processing, and communication [20] while maintaining a low-cost and flexible development environment [19]. The Raspberry Pi 5 is equipped with a multi-core processor and enhanced I/O interfaces, enabling efficient interaction with peripheral devices and sensors.

Sensor integration is achieved through the use of I^2C-based environmental sensors, such as the SparkFun BME280 sensor [21]. These sensors measure key environmental parameters including temperature, humidity, and atmospheric pressure [21]. The use of digital sensors with standardized communication protocols allows for reliable and scalable integration, closely mirroring the modular sensor architectures used in CubeSat systems. Python libraries are utilized to interface with the sensors, enabling periodic sampling and data acquisition with minimal overhead.

Telemetry generation is handled through custom Python scripts running on the Raspberry Pi. The collected sensor data is structured into formatted packets that emulate real CubeSat telemetry frames [22]. Each packet typically includes a sequence number, timestamp, sensor readings [22], and optional integrity checks such as a cyclic redundancy check (CRC). This structured approach allows the system to simulate realistic spacecraft data flows while supporting downstream processing such as encryption, transmission, and analysis.

To improve reliability and data integrity, telemetry packets can incorporate error-detection mechanisms based on cyclic redundancy checks [23]. A common implementation uses a 32-bit CRC, computed as the remainder of a polynomial division over a binary field:

$$C(x) = x^g M(x) \bmod G(x) \tag{1}$$

where $M(x)$ represents the message polynomial, $G(x)$ is the generator polynomial, and g is its degree [23]. This formulation enables efficient detection of transmission errors by verifying that the received message is divisible by the generator polynomial.

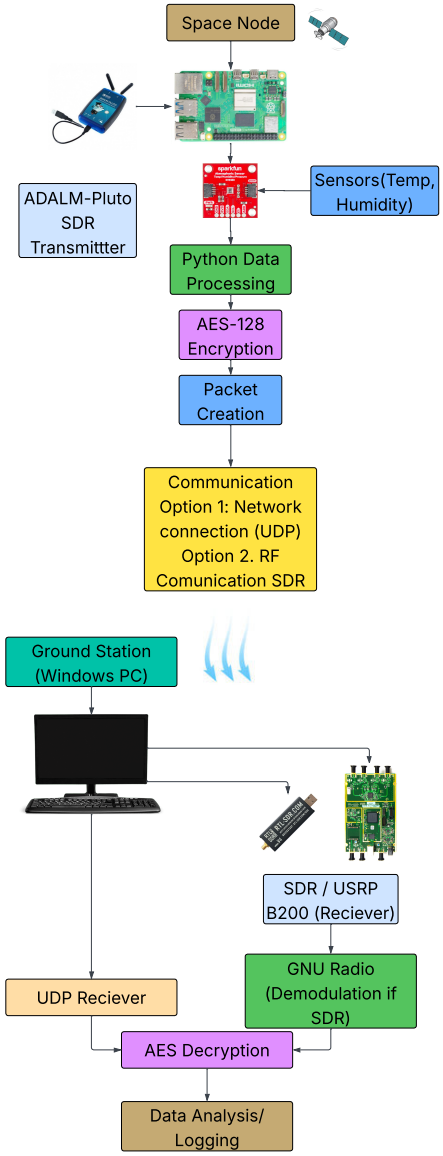

Fig. 1. system includes a Raspberry Pi–based space node that generates and encrypts telemetry data, a communication pipeline supporting both UDP-based and SDR-based transmission, and a ground station that receives, demodulates, decrypts, and analyzes the data.

B. Ground Station System

The ground station is implemented on a Windows-based computer, which acts as the receiver and processing unit for incoming data. The ground station is responsible for receiving transmitted packets, performing data decryption, and analyzing the received information. Python-based scripts are used to process incoming data streams and log system performance metrics. In SDR-based experiments, GNU Radio is used to receive and demodulate transmitted signals [24] before passing the recovered data to the processing layer for analysis.

C. Communication Pipeline

The communication pipeline is designed to simulate data transmission between a spacecraft and a ground station. Telemetry data generated on the Raspberry Pi is first processed and formatted before being encrypted using the Advanced Encryption Standard (AES-128). The encrypted data is then transmitted using User Datagram Protocol (UDP) over a network connection, allowing for reliable testing of packet-based communication. In addition to network-based communication, the system supports radio-frequency transmission using software-defined radios. In this configuration, data packets are passed to GNU Radio, where they are modulated and transmitted using SDR hardware such as the ADALM-Pluto. On the receiving side, SDR devices such as the RTL-SDR or USRP B200 are used to capture and demodulate the signal before recovering the transmitted data. This dual-mode communication design enables evaluation of both network-based and RF-based communication approaches.

D. Software Environment

The system integrates several software tools to support communication and data processing. Python is used for data generation, encryption, transmission, and logging. GNU Radio provides a flexible framework for designing and implementing SDR-based communication flowgraphs, enabling modulation, transmission, and reception of signals. GQRX [30] is used for RF signal monitoring and spectrum visualization during interception experiments. Additionally, COSMOS (OpenC3) [29] was explored as a telemetry and command platform for monitoring system behavior, although full integration was not completed. These tools collectively enable a flexible and modular test environment for evaluating communication performance.

IV. RF SIGNAL INTERCEPTION EXPERIMENT

This section presents an RF signal interception experiment using an RTL-SDR and GQRX software to observe wireless signals. The goal of this experiment is to demonstrate how easily RF transmissions can be detected and how factors such as distance and interference affect signal quality.

A. Experimental Setup

An RF signal interception experiment was conducted to evaluate the ability of low-cost software-defined radio hardware to monitor wireless transmissions. The experiment utilized an RTL-SDR receiver in conjunction with the GQRX software interface to observe radio-frequency signals. A car-based Bluetooth FM transmitter was used as the signal source, broadcasting audio from a mobile device at a frequency of 96.5 MHz. In the first test, the RTL-SDR receiver was positioned inside the vehicle to capture the transmitted signal under close-range conditions. In the second test, the receiver was placed approximately 15 feet away from the vehicle to evaluate the effect of distance on signal strength and interference. The GQRX interface was used to visualize the frequency spectrum and waterfall display, allowing for real-time observation of signal activity at the selected transmission frequency and surrounding bands.

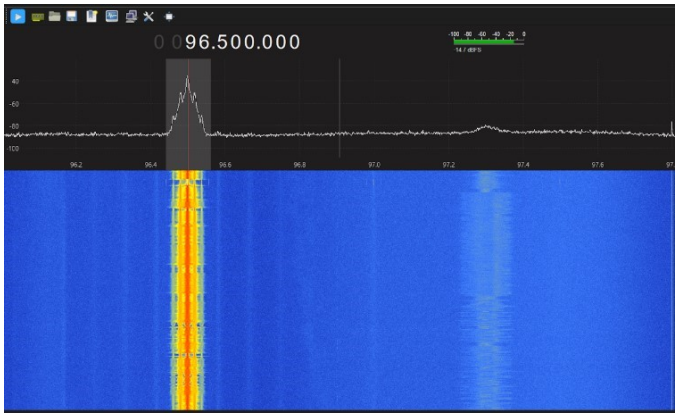

Fig. 2. A strong signal peak is observed at the transmission frequency, indicating successful interception.

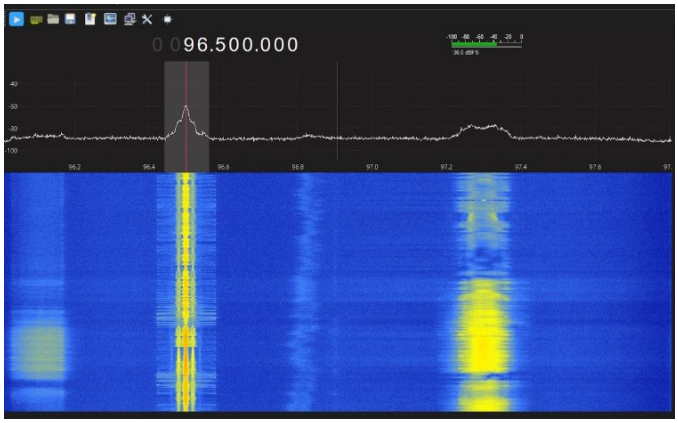

Fig. 3. The signal at 96.5 MHz is reduced, while nearby frequencies (97.2 to 97.4 MHz) show increased activity due to interference.

B. RF Interception Results

The results of the RF interception experiment are shown in Fig. 2 and Fig. 3. In the first test, shown in Fig. 2, a strong signal peak is observed at 96.5 MHz when the receiver is positioned inside the vehicle. The waterfall display confirms consistent signal activity at this frequency, indicating successful detection of the Bluetooth FM transmission.

In the second test, shown in Fig. 3, the receiver was positioned approximately 15 ft from the transmitter. The signal at 96.5 MHz remains detectable but is reduced in strength compared to the close-range measurement. Additionally, signals in nearby frequency bands, particularly in the range of 97.2–97.4 MHz, become more prominent. This suggests the presence of other RF sources and highlights increased susceptibility to interference as the receiver moves farther from the transmitter.

The observed reduction in signal strength with distance is consistent with expected RF propagation behavior, where signal power decreases as the separation between transmitter and receiver increases. This relationship can be quantified using the free space path loss (FSPL) model, which describes how signal

attenuation increases with both distance and frequency [26]:

$$\mathrm{FSPL} = \left(\frac{4\pi df}{c}\right)^2 \tag{2}$$

where d is the distance between the transmitter and receiver, f is the signal frequency, and c is the speed of light. A more detailed treatment of power budget analysis for CubeSat communication links is provided in [31].

The presence of additional signals at nearby frequencies further demonstrates the complexity of real-world RF environments. These results demonstrate that wireless transmissions can be intercepted using inexpensive SDR hardware and that signal quality is influenced by environmental conditions such as distance and interference. This highlights the importance of implementing secure communication mechanisms to protect transmitted data in CubeSat systems.

V. SECURE UDP COMMUNICATION EXPERIMENT

The objective of this experiment is to validate secure packet-based communication in a controlled network environment prior to implementing radio-frequency (RF) transmission. UDP-based communication was used to establish a reliable baseline for telemetry data transfer, enabling evaluation of data integrity, encryption, and latency without the additional complexity introduced by RF transmission

A. Network Configuration

The communication system was established between a Raspberry Pi–based space node and a Windows-based ground station using User Datagram Protocol (UDP). The Raspberry Pi acted as the transmitter, while the Windows machine functioned as the receiver. Both devices were connected over a local network, enabling direct packet transmission. UDP was selected due to its low overhead and suitability for real-time telemetry communication [31]. A dedicated port was configured for data transmission, allowing the Raspberry Pi to send packets to the ground station without requiring connection-oriented handshaking. Socket-based programming in Python was used to implement both the transmitting and receiving ends of the communication system. Additional networking tools were used during development to verify connectivity and ensure proper communication between devices. These tools assisted in confirming successful packet transmission and identifying configuration issues during system setup.

B. Telemetry Generation and UDP Communication

Telemetry data was generated using a SparkFun BME280 sensor connected to the Raspberry Pi. The sensor provided real-time measurements of temperature, humidity, and pressure, which were used to simulate typical CubeSat telemetry data. Python scripts were used to collect sensor readings and format the data into structured JSON packets [27].

Each packet included fields such as sequence number, timestamp, and sensor measurements. Sequence numbers were used to track packet order and detect packet loss, while timestamps enabled measurement of communication latency.

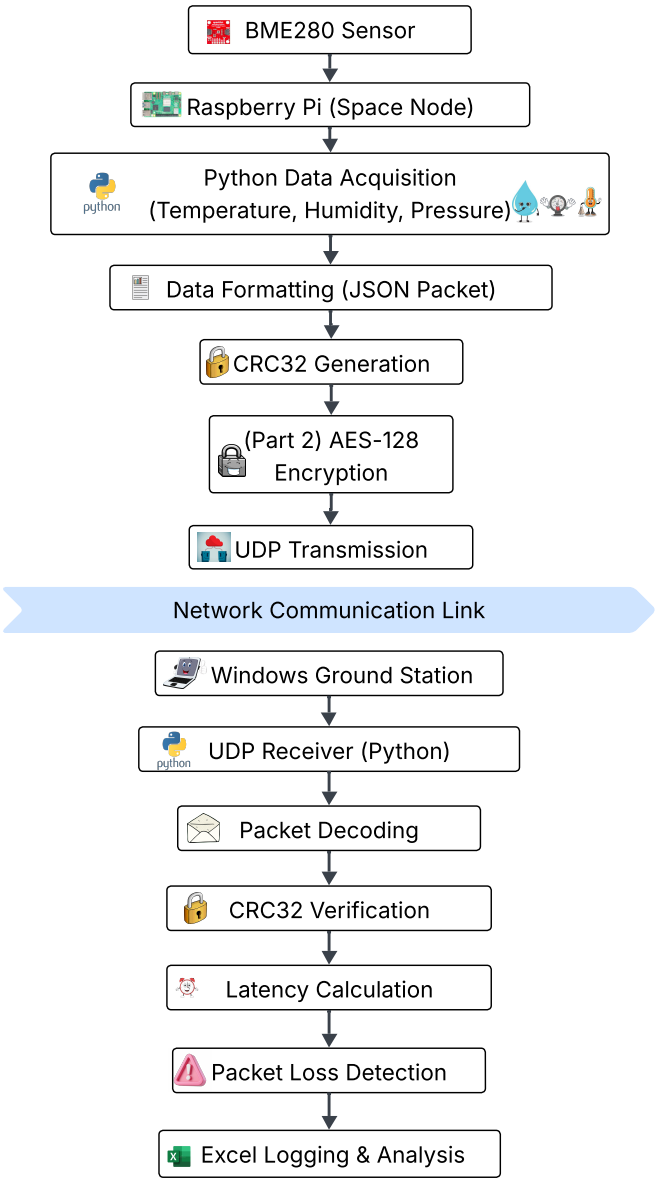

Fig. 4. Sensor data from the BME280 is processed on the Raspberry Pi, formatted into packets, and transmitted via UDP. CRC32 is used for integrity verification, and AES-128 encryption is optionally applied. At the ground station, packets are received, decoded, verified, and analyzed for latency and packet loss.

Prior to transmission, a CRC32 checksum was computed and appended to each packet to support data integrity verification.

The formatted packets were transmitted from the Raspberry Pi to the ground station using UDP sockets. At the receiver, packets were decoded and processed to extract sensor data and evaluate transmission performance. The overall communication pipeline used in this experiment is illustrated in Fig. 4.

C. AES-128 Encryption Implementation

AES-128 encryption was implemented in Galois/Counter Mode (GCM) to provide confidentiality and integrity. Packets were encrypted prior to transmission, and authentication tags

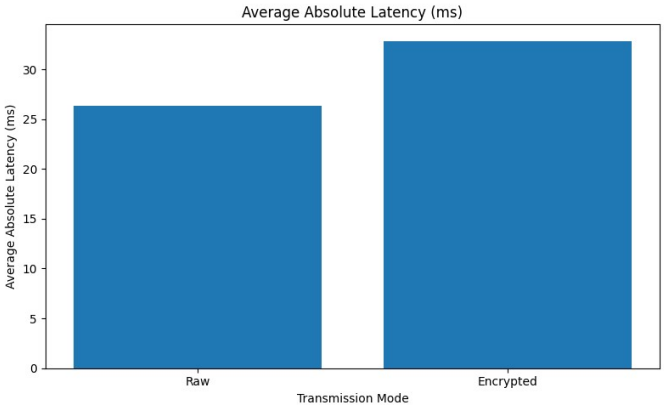

Fig. 5. Encryption introduces additional processing overhead, resulting in increased latency while maintaining reliable data transmission.

were verified at the receiver to detect tampering or corruption. In encrypted mode, the JSON packets were encrypted prior to transmission, and the resulting packet included the ciphertext, nonce, and authentication tag required for decryption. At the ground station, the received data was decrypted using the same shared key. The authentication tag was verified during decryption to ensure that the data had not been modified during transmission. If verification failed, the packet was considered invalid and discarded. This implementation ensures that transmitted data is protected from unauthorized access while also detecting any tampering or corruption that may occur during communication.

D. Latency Measurement

The performance of the communication system was evaluated by measuring the latency of transmitted packets. Latency was calculated as the difference between the packet send time and receive time, using timestamps included in each packet. Absolute latency values were recorded to account for timing discrepancies between devices.

Fig. 5 shows the average absolute latency for both raw and encrypted transmission modes. The results indicate that encrypted communication introduces additional processing overhead, resulting in increased latency compared to raw transmission. This increase is expected due to the computational cost associated with encryption and decryption operations.

In addition to latency, packet integrity and reliability were evaluated. CRC32 checksums were used to verify data integrity at the receiver, ensuring that transmitted packets were not corrupted. Sequence numbers were used to detect packet loss during transmission.

Table I presents a sample of the logged telemetry data from raw UDP communication. The results demonstrate successful packet reception with no detected data corruption or packet loss. Overall, the system provides reliable data transmission while maintaining acceptable performance, even when encryption is applied.

VI. SDR-BASED COMMUNICATION EXPERIMENTS

This section presents an SDR-based communication experiment using GNU Radio and software-defined radios to transmit telemetry data. The goal is to evaluate how the system can be extended from UDP-based communication to RF transmission.

A. Experimental Setup

To extend the communication system beyond network-based transmission, an SDR-based experiment was conducted using GNU Radio and software-defined radio hardware. The objective of this experiment was to evaluate the feasibility of transmitting telemetry data over a radio-frequency (RF) communication pipeline, building on the secure communication framework established in Section V. The experiment was implemented using the ADALM-Pluto SDR connected to a Raspberry Pi, with additional testing performed using a USRP receiver on a Windows-based system. The transmitter and receiver chains were developed using GNU Radio flowgraphs. In the primary test configuration, the system was operated in a loopback-style setup on a single device to validate end-to-end transmission and reception. Telemetry data collected from the BME280 sensor was written to a file and used as the input for the SDR transmission process. The ADALM-Pluto SDR was configured to transmit and receive signals at a carrier frequency of approximately 2.435 GHz.

B. SDR Transmission Implementation

The SDR transmission system was implemented using GNU Radio, as shown in Fig. 6 and Fig. 7. The design was based on a GNU Radio packet-based file transfer example utilizing Binary Phase Shift Keying (BPSK) modulation [15], which was adapted to support structured telemetry data generated from the experimental system. BPSK is a digital modulation scheme in which binary information is represented using two carrier phases separated by $180°$, corresponding to logical states "1" and "0" [28]. The transmitted signal can be expressed as

$$s(t) = A_c m(t) \cos(\omega_c t), \quad m(t) = \pm 1 \tag{3}$$

where $m(t)$ represents the binary data and the phase shift of π results in a sign inversion of the carrier [28]. BPSK was selected over frequency shift keying (FSK) due to its improved power efficiency, as it can achieve the same bit error rate (BER) with approximately half the transmitted power [28]. The BER performance of BPSK in an additive white Gaussian noise (AWGN) channel is given by

$$P_b = Q\left(\sqrt{\frac{2E_b}{N_0}}\right) \tag{4}$$

which illustrates that the probability of error decreases with increasing signal-to-noise ratio [28]. This modulation scheme aligns with the GNU Radio example design while providing strong performance in noisy communication environments.

In the transmitter flowgraph (Fig. 6), telemetry data was formatted into packets and processed through a protocol

TABLE I
SAMPLE RAW UDP TELEMETRY DATA WITH CRC32 INTEGRITY VERIFICATION

Seq	Latency (ms)	Abs Latency (ms)	Temp. (°C)	Humidity (%)	Pressure (hPa)	Payload (Bytes)	Recv CRC32	Calc CRC32	Integrity
1	-26.02	26.02	25.23	28.49	1024.08	155	3635950475	3635950475	TRUE
2	-26.17	26.17	25.24	28.54	1024.15	155	3225682858	3225682858	TRUE
3	-26.20	26.20	25.24	28.41	1024.12	154	476347149	476347149	TRUE
4	-26.23	26.23	25.26	28.42	1024.10	154	3538596642	3538596642	TRUE
5	-26.27	26.27	25.24	28.43	1024.06	155	4111849437	4111849437	TRUE
6	-26.27	26.27	25.24	28.37	1024.04	155	3664768411	3664768411	TRUE
7	-26.20	26.20	25.24	28.32	1024.10	153	153467876	153467876	TRUE
8	-26.36	26.36	25.24	28.27	1024.06	154	676797819	676797819	TRUE
9	-26.37	26.37	25.26	28.31	1024.04	153	1994574101	1994574101	TRUE
10	-26.42	26.42	25.26	28.34	1024.01	156	1268677370	1268677370	TRUE
11	-26.46	26.46	25.25	28.37	1024.03	156	3484950236	3484950236	TRUE
12	-26.51	26.51	25.25	28.39	1024.03	155	3003111855	3003111855	TRUE

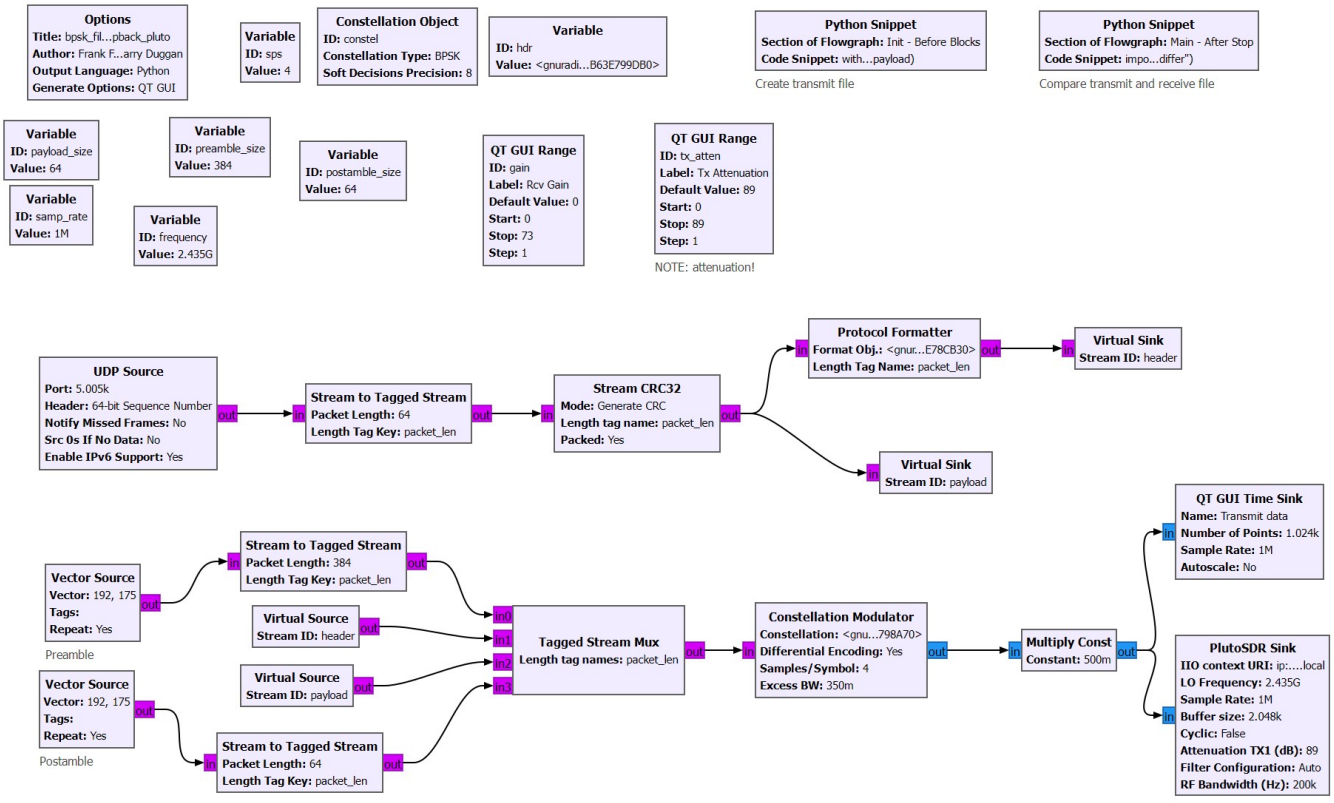

Fig. 6. Sensor data is formatted into packets, encoded, modulated using BPSK, and transmitted using the PlutoSDR hardware.

formatter, where framing and CRC32 error detection were applied. The packetized data was then modulated using BPSK and transmitted using the ADALM-Pluto SDR hardware platform.

At the receiver (Fig. 7), the incoming RF signal was processed through several signal conditioning and synchronization stages. A root raised cosine filter was applied to reduce intersymbol interference, followed by symbol synchronization and carrier recovery using a Costas loop. These steps ensured proper timing and phase alignment for accurate demodulation.

The demodulated signal was then decoded and passed through a CRC32 verification stage to detect transmission errors. The recovered packet data was converted into a readable format and written to a text file for analysis.

C. Results and Observations

The results of the SDR-based communication experiment are shown in Fig. 8, which presents the recovered telemetry data after transmission and reception. The output demonstrates that sensor data, including temperature, humidity, and pressure values, was successfully transmitted through the SDR pipeline and reconstructed into a readable format. This experiment confirms the feasibility of using SDR-based communication for transmitting structured telemetry data. The successful recovery of readable data indicates that the modulation, transmission, synchronization, and demodulation processes were functioning correctly within the controlled setup. However, the experiment was primarily conducted in a loopback configuration on a single device, which does not fully represent a real-world wireless

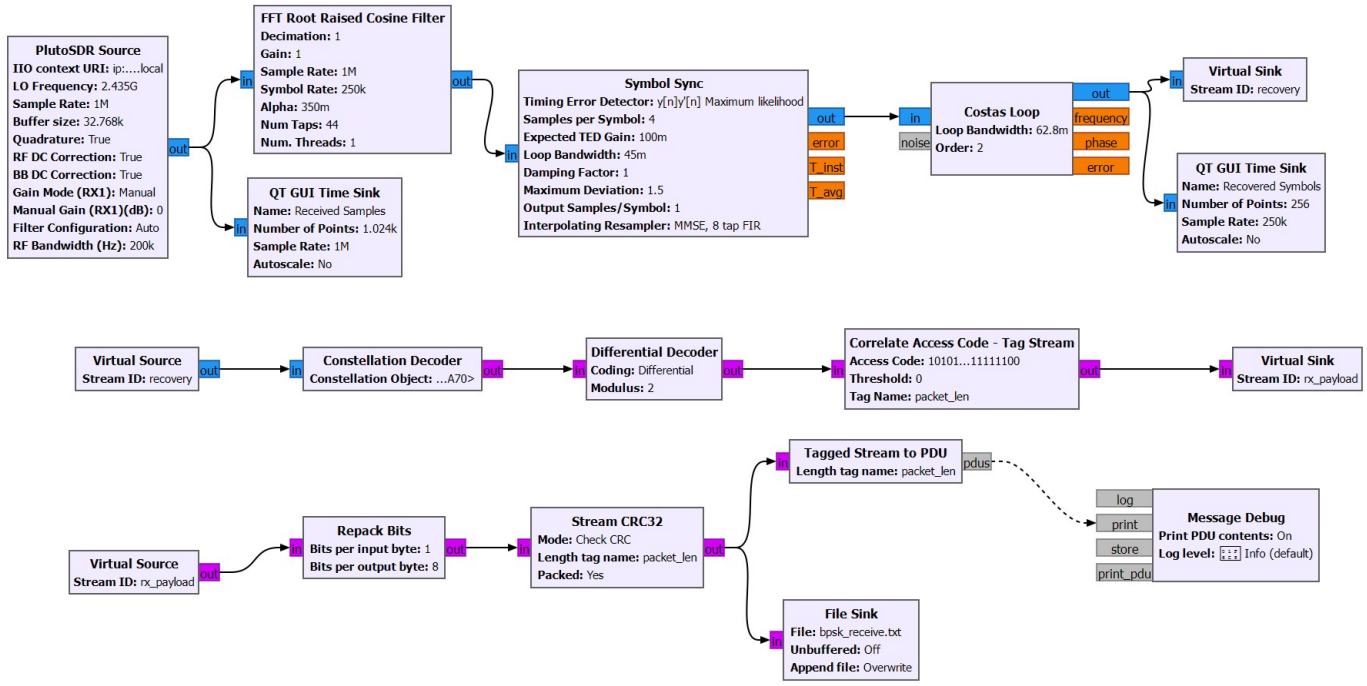

Fig. 7. The received signal is filtered, synchronized, demodulated, and decoded to reconstruct the transmitted packet data.

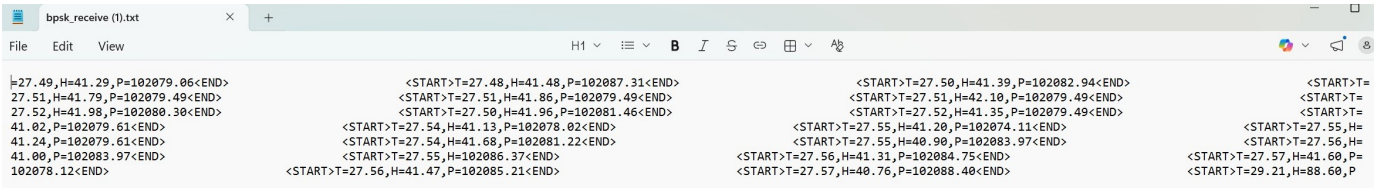

Fig. 8. The received output demonstrates successful reconstruction of transmitted sensor data, including temperature, humidity, and pressure values.

communication link. In practical RF environments, additional challenges such as noise, interference, signal attenuation, and synchronization between separate devices can significantly impact system performance. Attempts to extend the system to multi-device communication, including configurations involving the ADALM-Pluto and USRP platforms, highlighted the complexity of achieving stable real-time RF communication. Despite these limitations, this experiment was an important component of the overall research. It validated the SDR transmission pipeline and demonstrated that telemetry data can be successfully transmitted and recovered using GNU Radio and SDR hardware. This serves as a foundation for future work involving full RF communication between independent space and ground nodes, where secure communication mechanisms developed in earlier sections can be integrated into the SDR framework.

VII. RESULTS AND DISCUSSION

This section summarizes and compares the results obtained from the RF interception, UDP-based secure communication, and SDR-based transmission experiments. The goal is to evaluate the effectiveness, reliability, and limitations of each approach within the context of secure CubeSat communication systems.

A. RF Interception and Security Implications

The RF interception experiment demonstrated that wireless transmissions can be readily detected using low-cost software-defined radio hardware. Strong signal activity was observed at the transmission frequency when the receiver was positioned close to the source, and the signal remained detectable at increased distances, although with reduced strength and increased interference. These results highlight a key vulnerability in wireless communication systems: unencrypted transmissions can be passively intercepted without requiring physical access to the transmitter. This reinforces the importance of incorporating security mechanisms such as encryption and authentication to protect transmitted data, particularly in CubeSat systems where communication links are inherently exposed.

B. UDP-Based Communication Performance

The UDP-based communication system provided a reliable and controlled environment for evaluating secure data trans-

mission. Sensor data from the BME280 was successfully transmitted from the Raspberry Pi to the ground station, with all packets correctly received and verified using CRC32. Latency analysis showed that encrypted communication introduced additional processing overhead compared to raw transmission, as illustrated in Fig. 5. However, the increase in latency remained within an acceptable range for telemetry applications. The results also demonstrated consistent packet integrity, with CRC verification confirming that no data corruption occurred during transmission. Additionally, no packet loss was detected, indicating stable communication under the tested conditions. This experiment established a strong baseline for secure telemetry transmission, validating the effectiveness of AES-GCM encryption and CRC-based error detection in a controlled network environment.

C. SDR-Based Communication Observations

The SDR-based experiment demonstrated the feasibility of transmitting structured telemetry data using GNU Radio and software-defined radio hardware. The successful recovery of readable sensor data confirmed that the modulation, transmission, and demodulation processes were functioning correctly within the implemented pipeline. However, the experiment also revealed several challenges associated with RF-based communication. The system was primarily validated in a loopback configuration on a single device, and attempts to extend communication across multiple devices introduced additional complexity. Issues related to synchronization, signal stability, and hardware configuration highlighted the difficulty of achieving reliable real-time RF communication compared to network-based transmission. These observations emphasize that while SDR platforms provide flexibility and powerful capabilities for communication system design, they also require careful tuning and system-level integration to achieve robust performance in practical environments.

D. Comparative Discussion

The results from the three experimental stages demonstrate a clear progression in system development: The RF interception experiment highlighted the vulnerability of unprotected wireless communication. The UDP-based system established a reliable and secure communication framework with measurable performance metrics. The SDR-based experiment demonstrated the feasibility of extending the system to RF transmission, while also exposing practical implementation challenges. Together, these results show that secure communication mechanisms such as encryption and integrity verification are essential for protecting telemetry data, and that validating these mechanisms in a controlled environment is a critical step before deploying them in RF-based systems. Overall, the research demonstrates that a low-cost platform can support secure telemetry communication while providing a foundation for future work in SDR-based CubeSat communication systems.

VIII. FUTURE WORK

Future work will focus on extending the SDR-based communication system to support reliable transmission between separate devices. While the current implementation demonstrates end-to-end transmission in a controlled configuration, achieving stable communication across independent transmitter and receiver nodes remains a key challenge. This will require addressing issues such as synchronization, signal attenuation, and interference in real-world RF environments. Another area of future work involves integrating the secure communication mechanisms from the UDP-based system into the SDR transmission pipeline. AES-GCM encryption and CRC-based integrity verification were successfully implemented in a network environment, and incorporating these mechanisms into the RF communication chain will enable secure over-the-air transmission of telemetry data. Future work will also explore the implementation of more advanced communication protocols, such as the CubeSat Space Protocol (CSP), to support scalable and standardized communication between spacecraft and ground systems. This includes evaluating protocol performance under constrained system resources and integrating encryption at the protocol level. In addition, future improvements may include expanding the telemetry system to support additional sensors and real-time data streaming, as well as enhancing system monitoring and visualization through integration with telemetry platforms such as COSMOS (OpenC3). Finally, future experiments will investigate system performance under adverse conditions, including noise, interference, and potential attack scenarios such as jamming or spoofing. These evaluations will provide further insight into system resilience and support the development of more robust and secure communication architectures.

IX. CONCLUSION

This work presented the development of a low-cost FlatSat-based testbed for evaluating secure CubeSat communication systems. The testbed integrated a Raspberry Pi–based space node, environmental sensors, and both network-based and SDR-based communication pipelines to simulate telemetry transmission between a spacecraft and a ground station. Multiple experiments were conducted to evaluate different aspects of the communication system. An RF interception experiment demonstrated the vulnerability of unprotected wireless transmissions to passive monitoring using low-cost SDR hardware. A UDP-based communication system was then implemented to validate reliable and secure packet transmission, incorporating CRC32 for data integrity and AES-GCM encryption for confidentiality and authentication. The results showed successful transmission of sensor data with no detected packet loss or corruption, while highlighting the latency overhead introduced by encryption. Finally, SDR-based experiments using GNU Radio and ADALM-Pluto demonstrated the feasibility of transmitting telemetry data over an RF communication pipeline, while also revealing practical challenges related to synchronization and system stability.

X. Acknowledgements

I would like to thank Dr. Dimitire C. Popescu for working with me this past year as I researched and developed this project. I would also like to thank the Virginia Space Grant Consortium for creating this program that allows students to receive funding for research projects they are passionate about. Lastly, I am very thankful to the McNair staff at Old Dominion University for guiding me through the research process and providing opportunities to share my research as it progressed.

References

[1] NASA, "What are SmallSats and CubeSats?," Aug. 15, 2024. [Online]. Available: https://www.nasa.gov/what-are-smallsats-and-cubesats/

[2] T. Villela, C. A. Costa, A. M. Brandão, F. T. Bueno, and R. Leonardi, "Towards the Thousandth CubeSat: A Statistical Overview," *International Journal of Aerospace Engineering*, vol. 2019, 2019, doi: 10.1155/2019/5063145.

[3] G. Falco, A. Viswanathan, and A. Santangelo, "CubeSat Security Attack Tree Analysis," in *Proc. IEEE SMC-IT*, Pasadena, CA, USA, 2021, pp. 68–76, doi: 10.1109/SMC-IT51442.2021.00016.

[4] O. Challa, G. Bhat, and J. Mcnair, "CubeSec and GndSec: A lightweight security solution for CubeSat communications." [Online]. Available: https://digitalcommons.usu.edu/cgi/viewcontent.cgi?article=1031 &context=smallsat.

[5] A. R. K. Verma, "Cybersecurity in Satellite Communication Networks: Key Threats and Neutralization Measures," *IEEE Open Journal of the Communications Society*, vol. 6, pp. 5667–5692, 2025, doi: 10.1109/OJCOMS.2025.3585060.

[6] Y. Arjoune and S. Faruque, "Smart Jamming Attacks in 5G New Radio: A Review," in *Proc. CCWC*, Las Vegas, NV, USA, 2020, pp. 1010–1015, doi: 10.1109/CCWC47524.2020.9031175.

[7] E. Salkield *et al.*, "Satellite Spoofing from A to Z: On the Requirements of Satellite Downlink Overshadowing Attacks," 2023, doi: 10.1145/3558482.3590190.

[8] L. Grillmayer and S. Arnold, "Integrating the CubeSat Space Protocol into GSOC's Multi-Mission Environment." [Online]. Available: https://digitalcommons.usu.edu

[9] J. André *et al.*, "ISTNanosat-1 heart processing and digital communications unit," 2012. Accessed: Apr. 03, 2026. [Online]. Available: https://fenix.tecnico.ulisboa.pt/downloadFile/395144736837/MScThesis-JFerreira_vFinal.pdf

[10] N. A. Salam Bauomy and E. A. Elbeh, "Design of SDR Simulation for Wireless Communication between Ground Station and CubeSat," in *Proc. JAC-ECC*, 2020, pp. 60–63, doi: 10.1109/JAC-ECC51597.2020.9355965.

[11] R. C. Reinhart and J. P. Lux, "Space-based Reconfigurable Software Defined Radio Test Bed aboard International Space Station," 2014, doi: 10.2514/6.2014-1612.

[12] T. Joseph, "OPTASAT: An Open-Source, Flexible Software Framework for Small Satellite Operations," MIT, Feb. 2025. [Online]. Available: https://dspace.mit.edu

[13] J. Melville, A. Narvaez, and L. Jasper, "A Review of Automation in Small Satellite Operations," in *Proc. IEEE Aerospace Conf.*, 2025, doi: 10.1109/AERO63441.2025.11068397.

[14] D. C. Popescu and R. Vida, "A Primer on Software Defined Radios," *Infocommunications Journal*, vol. XIV, no. 3, pp. 16–27, Sep. 2022.

[15] GNU Radio, "File transfer using packet and BPSK," GNU Radio Wiki. [Online]. Available: https://wiki.gnuradio.org/index.php?title=File_transfer_using_Packet_and_BPSK

[16] A. M. Wyglinski, D. P. Orofino, M. N. Ettus, and T. W. Rondeau, "Revolutionizing software defined radio: Case studies in hardware, software, and education," *IEEE Communications Magazine*, vol. 54, no. 1, pp. 68–75, Jan. 2016, doi: 10.1109/MCOM.2016.7378428.

[17] R. F. III, "The Virginia Tech FlatSat: A platform for small satellite testing," VTechWorks, Nov. 14, 2023. Accessed: Apr. 03, 2026. [Online]. Available: https://vtechworks.lib.vt.edu/items/319f1e0e-a700-4868-859f-180cae7441a6

[18] NASA Office of Inspector General, "NASA's management of the International Space Station and efforts to commercialize low Earth orbit," 2021. [Online]. Available: https://oig.nasa.gov/wp-content/uploads/2024/02/IG-22-005.pdf

[19] B. Lovdahl, "Software-defined radio payload design for CubeSat and X-band communications," Naval Postgraduate School, Monterey, CA, USA, 2018. [Online]. Available: https://apps.dtic.mil/sti/trecms/pdf/AD1069657.pdf

[20] R. Hudson, C. Smith, D. Des Riviere, and T. Barnes, "Design of a low-cost CubeSat for earth monitoring and data transmission," *AIAA 2024-85697, 2024 Regional Student Conferences*, Jan. 2024. [Online]. Available: https://arc.aiaa.org/doi/pdf/10.2514/6.2024-85697

[21] Bosch Sensortec, "BME280 datasheet," 2018. [Online]. Available: https://cdn.sparkfun.com/assets/e/7/3/b/1/BME280_Datasheet.pdf

[22] Consultative Committee for Space Data Systems (CCSDS), "CCSDS historical document." Accessed: Apr. 03, 2026. [Online]. Available: https://ccsds.org/Pubs/102x0b5s.pdf

[23] Allegro MicroSystems, "Cyclic redundancy check (CRC) algorithms in sensor communications." [Online]. Available: https://www.allegromicro.com/-/media/files/application-notes/an296177-crc-algorithms-in-sensor-communication.pdf?sc_lang=en

[24] L. K. Patton, "A GNU Radio-based software-defined radar," CORE Scholar, 2016. [Online]. Available: https://corescholar.libraries.wright.edu/etd_all/91/

[25] Y. Prabowo, I. N. Y. Putro, Y. Firmansyah, N. Chasanah, A. Ruhiyat, and C. E. Santosa, "Design of IP satellite communication for real-time UAV telemetry: Case Japan–Indonesia link," in *Proc. 2021 Int. Conf. Radar, Antenna, Microwave, Electronics, and Telecommunications (ICRAMET)*, Bandung, Indonesia, 2021, pp. 153–157, doi: 10.1109/ICRAMET53537.2021.9650481.

[26] D. Cama-Pinto, M. Damas, J. A. Holgado-Terriza, F. Gómez-Mula, and A. Cama-Pinto, "Path loss determination using linear and cubic regression inside a classic tomato greenhouse," *Int. J. Environmental Research and Public Health*, vol. 16, no. 10, p. 1744, May 2019, doi: 10.3390/ijerph16101744.

[27] I. Garg, "Study on JSON, its uses and applications in engineering organizations," ResearchGate, Mar. 16, 2024. [Online]. Available: https://www.researchgate.net/publication/379001324_Study_on_JSON_its_Uses_and_Applications_in_Engineering_Organizations

[28] S. Hudson, "Binary phase shift keying (BPSK)," Oct. 19, 2004. [Online]. Available: https://eecs.wsu.edu/~ee432/Text/20-bpsk.pdf

[29] OpenC3, "Introduction," OpenC3 Docs, 2026. Accessed: Apr. 03, 2026. [Online]. Available: https://docs.openc3.com/docs

[30] A. Csete, "Gqrx SDR: Open source software defined radio," Gqrx. Accessed: Apr. 02, 2026. [Online]. Available: https://www.gqrx.dk/

[31] B. E. Bekele, K. Tokarz, N. Y. Gebeyehu, B. Pochopień, and D. Mrozek, "Performance evaluation of UDP-based data transmission with acknowledgment for various network topologies in IoT environments," *Electronics*, vol. 13, no. 18, pp. 3697–3697, Sep. 2024, doi: 10.3390/electronics13183697.

[32] O. Popescu, "Power budgets for CubeSat radios to support ground communications and inter-satellite links," *IEEE Access*, vol. 5, pp. 12618–12625, 2017, doi: 10.1109/ACCESS.2017.2721948.