16th International Conference on Space Operations, Cape Town, South Africa – 3 - 5 May 2021.

SpaceOps-2021,4,x1364

Development of the Ground Segment Communication System for the EIRSAT-1 CubeSat

Fergal Marshall[A,B], David Murphy[B], Lána Salmon[B], Derek O'Callaghan[B], Maeve Doyle[B], Jack Reilly[B], Rachel Dunwoody[B], Jessica Erkal[B], Gabriel Finneran[B], Gianluca Fontanesi[D], Jack Kyle[B], Joseph Mangan[B], Joseph Thompson[C], Sarah Walsh[B], Daithí de Faoite[C], Lorraine Hanlon[B], David McKeown[C] William O'Connor[C], Ronan Wall[B], Sheila McBreen[B], Derek Greene[A].

[A]*School of Computer Science, University College Dublin, Belfield, Dublin 4, Ireland*
[B]*School of Physics, University College Dublin, Belfield, Dublin 4, Ireland*
[C]*School of Mechanical and Materials Engineering, University College Dublin, Belfield, Dublin 4, Ireland*
[D]*School of Electrical And Electronic Engineering, University College Dublin, Belfield, Dublin 4, Ireland*

Abstract

The Educational Irish Research Satellite (EIRSAT-1) is a student-led project to design, build and test Ireland's first satellite. As part of the development, a ground segment (GS) has also been designed alongside the spacecraft. The ground segment will support two-way communications with the spacecraft throughout the mission. Communication with the satellite will occur in the very high frequency (VHF) and the ultra high frequency (UHF) bands for the uplink and downlink respectively. Different modulation schemes have been implemented for both uplink and downlink as part of the GS system. Uplink incorporates an Audio Frequency Shift-Keying (AFSK) scheme, while downlink incorporates a Gaussian Minimum Shift-Keying (GMSK) scheme. In order for the spacecraft to successfully receive a telecommand (TC) transmitted from the ground station, a framing protocol is required. AX.25 was selected as the data link layer protocol. A hardware terminal node controller (TNC) executes both the AX.25 framing and the AFSK modulation. Keep It Simple Stupid (KISS) framing software was developed to allow data to be accepted by the TNC. A software defined radio (SDR) approach has been chosen for the downlink. GNURadio is software that allows flowcharts to be built to undertake the required signal processing of the received signal, the demodulation of the signal and the decoding of data. This paper provides a detailed account of the software developed for the ground segment communication system. A review of the AX.25 and KISS framing protocols is presented. The GNURadio flowcharts that handle the signal processing and data decoding are broken down and each constituent is explained. To ensure the reliability and robustness of the system, a suite of tests was undertaken, the results of which are also presented.

I. Introduction

The Educational Irish Research Satellite (EIRSAT-1) is a 2U CubeSat being developed and built by students at University College Dublin (UCD) as part of the European Space Agency (ESA) Fly Your Satellite! programme [1]. The satellite will host three payloads that will fulfil the scientific and educational goals of the mission. The primary payload, the Gamma-ray Module (GMOD), is a novel gamma-ray detector that was designed and constructed in-house [2], [3]. The second payload is the ENBIO Module (EMOD) which aims to test the performance of the SolarBlack and SolarWhite thermal surface coatings, designed by ENBIO Ltd., with the thermal coatings initially designed for the ESA Solar Orbiter mission [4]. The third on-board payload, Wave Based Control (WBC), is a novel attitude control system that has been developed in-house. WBC can take over from the Commercial Off The Shelf (COTS) Attitude Determination and Control System (ADCS) and control EIRSAT-1's attitude [5], [6]. The spacecraft's Antenna Deployment Module (ADM) subsystem has also been designed and developed by the EIRSAT-1 project [7]. The remaining subsystems of the spacecraft consist of COTS components procured from AAC Clyde Space [8].

In addition to the space segment, the ground segment is also being developed. This paper provides a high-level overview of the EIRSAT-1 ground segment design, detailing both its hardware and software components in §II. We then give an in-depth description of the communication system of the ground segment. In §III the uplink component is discussed, together with the relevant protocols and the software that was developed to implement these protocols. Similarly, for the downlink component, the demodulation and decoding techniques that are employed to receive data from the satellite are detailed in §IV. Finally, the results of the testing that was undertaken to show the reliability of the

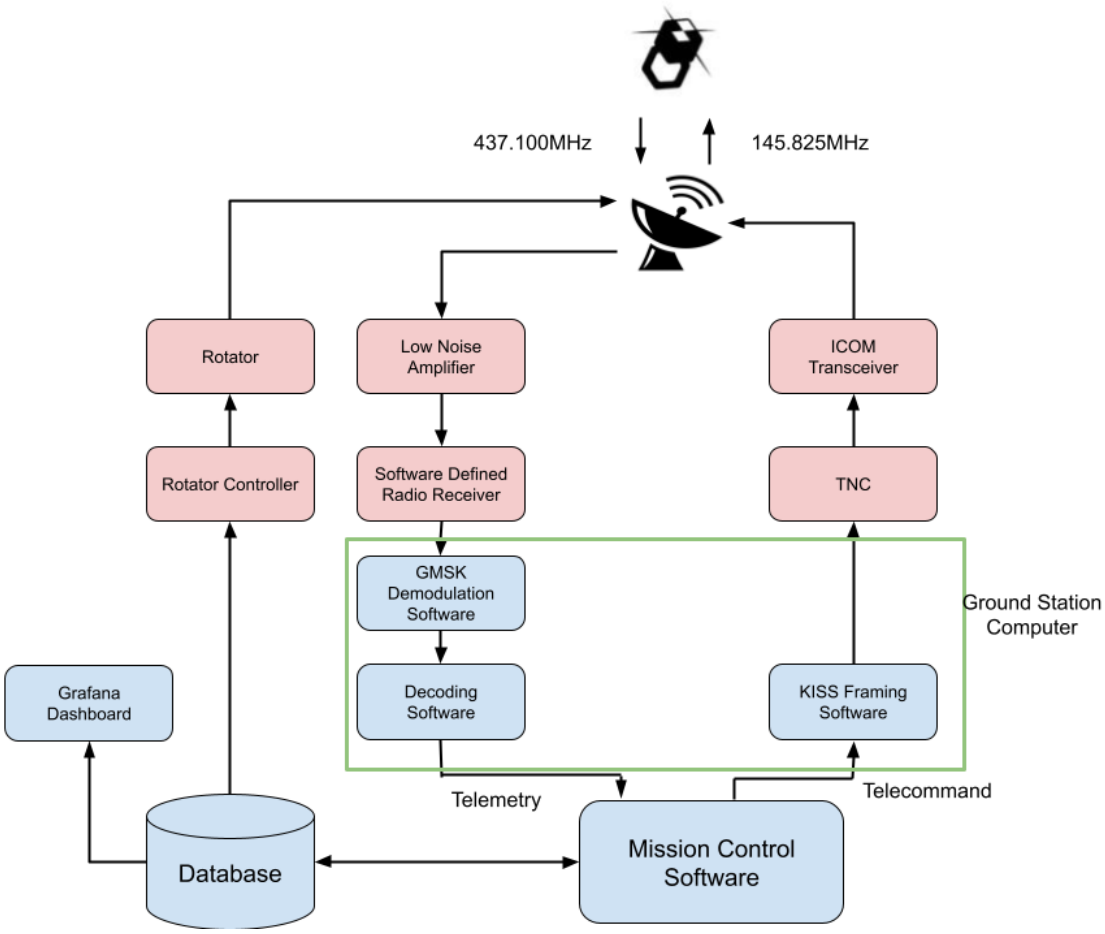

Fig. 1 A high-level overview of the EIRSAT-1 ground segment. The components highlighted in red are hardware components and the blue components are all software components. The components enclosed in the green box signify they run on the ground station computer and are presented in this paper.

entire system are presented in §V.

II. EIRSAT-1 Ground Segment

The ground segment consists of the hardware and software that will make up the ground station, the Mission Control Software (MCS) that will be used to interact with the spacecraft, the database that will be used to store the telecommands (TC) and telemetry (TM) sent and received from EIRSAT-1, and finally the dashboard that will allow operators to view the telemetry in a graphical representation. Figure 1 shows a high-level overview of the design and the flow of data through the ground segment.

A. Ground Station

The ground station will consist of the hardware and software that will be directly involved in the communications with the spacecraft. Communication between EIRSAT-1 and the ground station will occur in the Very High Frequency (VHF) and Ultra High Frequency (UHF) bands, for uplink (f = 145.82MHz) and downlink (437.1MHz) respectively. The ground station will be located at the UCD campus in Dublin, Ireland, with coordinates of 53.3083°N, 6.2237°W. This location will provide communication windows between 4 to 7 minutes in duration. The antennas that will carry out the Radio Frequency (RF) communication will be a 2×10 element 2m Yagi antenna for the uplink and a 2×18 element 70cm Yagi antenna for the downlink. A Yaesu G-5500 rotator will be connected directly to the antenna mast to track the

spacecraft. The uplink will employ the AX.25 framing protocol and 1200bps AFSK modulation technique. To carry out both of these requirements, a Terminal Node Controller is used. The SCS Tracker DSP TNC has been selected. This component will connect directly to the transceiver which will be used for the uplink. The selected transceiver is the ICOM-9100. For the downlink a 9600bps GMSK modulation scheme is used. We have implemented a Software Defined Radio (SDR) approach to handle the demodulation and decoding of the downlinked data. The SDR will receive data from a Low Noise Amplifier (LNA) connected to the antenna and will route the data into the demodulation software, which will run on the ground station computer (a Dell OptiPlex 7040).

B. Mission Control Software

The Mission Control Software (MCS) was developed as part of the ground segment. This system is responsible for command generation for TCs to be transmitted to the spacecraft, as well as receiving and parsing the TM data, and then storing this data in a database. The received TM will be displayed within the MCS in a tabular format for quick inspection of parameters by the operator. Both scheduled and real-time operations will take place through the MCS. A scheduled operation is a script that generates TCs to interact with the spacecraft to achieve a predefined set of tasks. In contrast, the real-time case requires an operator to directly interact with the spacecraft during a communication window. The MCS was implemented in Python 3.7, using the Django Web Application framework [1], with Celery [2] and RabbitMQ [3] handling the scheduling of tasks. The system will be hosted on a remote server and will connect to the ground station via a TCP connection. When an operator will interact with the MCS, an operations manual has been developed by a member of the EIRSAT-1 team to ensure the correct procedures are followed navigating through the MCS and transmitting TCs to the spacecraft [9].

C. Visualisation

To allow for the operator to effectively inspect the data, a Grafana[4]-based dashboard is being developed by a member of the ground segment team. The aim is to have both a private dashboard, solely accessible by EIRSAT-1 operators, and a public dashboard. The latter will allow members of the general public to view and keep updated with specific telemetry information received from EIRSAT-1. In each case, the dashboard will interact directly with the database to extract TM and visualise the data. Predefined conversions will be required and implemented automatically to ensure the extracted parameters are displayed correctly. The primary benefits of using Grafana are ease of development and its flexibility to connect to many different types of database.

III. EIRSAT-1 Uplink

The design for the ground segment communication system was driven by the COTS on-board communication subsystem and the design of the on-board software. The acquired communication subsystem is the Cape Peninsula University of Technology (CPUT) CMC board procured through AAC Clyde Space. The CMC board has flown on a number of CubeSats to date. The CMC board supports a VHF receiver and a UHF transmitter and allowing full-duplex communications to occur. Both a 1200bps Audio Frequency Shift Keying (AFSK) and a 9600bps Gaussian Minimum Shift Keying (GMSK) modulation schemes are available for use with both schemes configurable for either the uplink or downlink. The CMC board incorporates the AX.25 (Amateur X.25) framing protocol. An important feature is when transmitting, the CMC board can operate in a transparent mode where there are no protocols implemented by the board itself but rather allows the on-board computer (OBC) to implement the desired protocols. The on-board software was developed using Bright Ascension's GenerationOne Flight Software Development Kit (FSDK) [10]. The FSDK has native support for a number of space industry standard communication protocols including the CCSDS Space Data Link protocol and the CCSDS TM Synchronisation and Channel Coding protocol. This allows for the OBC to implement the downlink framing and Forward Error Correction (FEC) coding scheme for the CMC board to transmit. A convolutional encoder can be activated to add a final layer of FEC when the CMC is operating in transparent mode. These constraints have led to the decision for the EIRSAT-1 ground segment communication system to implement a 1k2bps AFSK uplink incorporating the AX.25 protocol. The downlink, described in detail in §IV, implements a 9k6 GMSK downlink.

The implementation of the AX.25 framing protocol and the AFSK modulation is carried out by the selected TNC.

[1] https://www.djangoproject.com/
[2] https://docs.celeryproject.org/en/stable/getting-started/introduction.html
[3] https://www.rabbitmq.com/
[4] https://grafana.com/

Field	Flag	Destination Address	Source Address	Digipeater Address	Control Field	Protocol ID	Information Field	Frame Check Sequence (FCS)	Flag
Bytes	1	7	7	0 - 56	1	1	0 - 256	2	1

Table 1 The structure of a UI AX.25 frame.

The TNC is connected to the ground station computer over serial. The software that we have developed for the uplink is designed to interface with the TNC, since the TNC carries out the AX.25 framing and the AFSK modulation. A serial interface protocol called Keep It Simple, Stupid (KISS) is used to send the uplink data from the host machine to the TNC.

A. AX.25 Frame

A full in-depth review of the AX.25 protocol can be found at [11] as the following information is only relevant for the EIRSAT-1 mission. The AX.25 protocol has three main frame types: Information frames, Supervisor frames, and Unnumbered frames. The mission will also use a frame known as an Unnumbered Information (UI) frame, which can work outside of the normal flow control protocols with frames being transmitted without prior consent of nodes. The structure of the UI AX.25 frame that is employed can be seen in Table 1.

- **Flags:** The beginning and end of a frame is denoted with a flag that is one byte long. This byte has a value of `0x7E`. Multiple flags can be sent once the frame has been transmitted to ensure that the frame has ended correctly.
- **Callsigns:** The Destination and Source callsigns are seven bytes long each, with 6 bytes being uppercase alphanumeric characters and a Secondary Station IDentifier (SSID) byte. The first three bits of the SSID byte denote whether either node is sending or receiving a command. The following four bits of the byte denote the SSID value, in the range of 0-15. The final bit states whether the current callsign is the final callsign in the frame. If there are digi-peaters to follow, the final bit is given the value 0, but if the callsign is the final callsign in the frame, a value of 1 is given. Digi-peaters can be used to amplify a decaying signal by routing the signal through a station to amplify it. There is a possibility to include up to seven digi-peater callsigns. This can greatly increase the range of transmission, while also allowing the receiver to follow the transmitted route.
- **Control:** The control field defines the type of frame that is being transmitted, whether it is an Information, Supervisory or Unnumbered frame. This field is one byte long and is given the value of `0x03` which signifies a UI frame.
- **Protocol Identifier:** As AX.25 frames operate in Layer 2 of the OSI model, the Protocol ID defines what Layer 3 protocol is used. In the case of EIRSAT-1, there are no network protocols used and this is given the value of `0x03`.
- **Information:** This is the field where the desired data is transmitted. Up to 256 bytes of data can be transmitted in an UI AX.25 frame. There is only one check of the information that occurs. As the start and end flag are denoted by `0x7E` = 1111110, within the AX.25 frame, no more than five consecutive 1 bits can appear. A process known as bit-stuffing, where non-information 0s are inserted, if more than five consecutive 1s are present.
- **Cyclical Redundency Check:** The final field in the frame is a two byte checksum. This checksum is present to allow the receiver to check whether the frame has been transmitted correctly. The 16 bit Cyclical Redundancy Check (CRC) is calculated by the sender using the (CRC)-16-CCITT approach. Equation 1 is used for this method. The receiver then does the same calculation and if the two CRCs match then it is known that the transmission is successful. There is no Forward Error Correction (FEC) with the CRC, but its purpose is just to indicate whether an error has occurred.

$$CRC = x^{16} + x^{12} + x^5 + 1 \tag{1}$$

The generated AX.25 frame is AFSK-modulated on the built-in modem of the TNC and is then sent to the ICOM for transmission to the spacecraft. The AX.25 frame incorporates parts of the High-Level Data Link Control (HDLC) protocol. The HDLC protocol utilises Non Return Zero Inverted (NRZ-I) encoding. This encoding scheme is what allows for differentiation between the logical symbols. A change in state is denoted as a binary 1, with 0 indicating a non-transition between states. The AFSK modulation technique represents these symbols as 1200Hz and 2200Hz tones.

Description	Acronym	Hex Value	Replaced With
Frame End	FEND	0xC0	FESC TFEND
Frame Escape	FESC	0xDB	FESC TFESC
Transpose Frame End	TFEND	0xDC	
Transpose Frame Escape	TFESC	0xDD	

Table 2 A list of the special characters reserved in the KISS protocol.

B. Keep It Simple, Stupid (KISS) Framing

The Keep It Simple, Stupid (KISS) serial protocol was developed to allow for a serial interface between the host machine and the TNC [12]. The protocol gives the user more flexibility over the AX.25 frames by taking the bulk of the framing process away from the TNC and moving it to the host machine. The generated KISS frames are then sent to the TNC for the final part of AX.25 framing. Similar to an AX.25 frame, a KISS frame has beginning and end flags, which are given the value `0xC0` and named FEND. The following byte defines the type of KISS frame. As KISS was developed to interface with a TNC, there are KISS frames sent to the TNC to change low level functionality of the TNC. The functionality that can be controlled with KISS frames are to do with different transmission parameters such as the TX delay, the Slot Time and the persistence. In total there are 6 transmission properties that can be changed. A pure data frame can also be generated. To signify a data frame, the byte preceding the initial FEND byte has the value of `0x00`. The remainder of the KISS frame is the full AX.25 frame without the `0x7E` flags and the 2 byte CRC.

Within the KISS protocol there are specific bytes that cannot appear in the middle of a frame. One of these special bytes is the FEND byte. The four special KISS bytes can be seen in table 2. If a FEND character appears within a KISS frame, the FEND byte is replaced with the FESC, TFEND characters. A similar process occurs if a FESC character appears in the frame, it is replaced with the FESC, TFESC characters.

Overall, the TeleCommand (TC) that the KISS framing software receives from the MCS applies the following steps:

1) Format the destination and source callsigns.
2) Adds the control and PID bytes to proceed the callsigns and then adds the TC.
3) Defines the KISS frame type.
4) Scans through the current frame for any of the special reserved bytes that may be present and replaces the bytes, if necessary.
5) Adds the FEND bytes to the beginning and end of the frame.
6) Writes the current KISS frame over serial to the TNC, where the FEND characters are removed and replaced with the AX.25 flags, CRC is calculated and the AX.25 frame is 1k2 AFSK modulated for transmission.

The KISS framing software that was developed for the EIRSAT-1 mission was written in Python 3.7 using standard Python libraries.

IV. EIRSAT-1 Downlink

Ground stations for satellite communications have typically been built using a pipeline of hardware components for the transmission and reception of data to and from a satellite. The standard hardware components for a receiving chain include an antenna, an amplifier and filter, and a modem. The software that was incorporated into a ground station would control the functionality of the hardware, rather than perform any of the necessary tasks needed to recover the transmitted data. With recent advances in computing power, now some legacy hardware components can be replaced by software components. In particular, Software Defined Radios (SDR) are becoming more prominent throughout ground station designs. The main reason for this is the ease of reconfigurability if an error occurs within the ground station solution [13], [14]. Software updates and code patches are preferable over ordering new hardware components due to the reduced cost and lead time. For this reason, CubeSat missions with small budgets and the need for rapid development time are increasingly incorporating SDRs into their ground station design. As one of those teams, the EIRSAT-1 mission has opted to use an SDR approach.

We use GNURadio [15] as the development toolkit to build the software solution for the receiver chain in the communication system for the ground station. GNURadio is an open source platform that interfaces with SDR receivers, handles signal processing, and includes options for communication through many network protocols. Extensions can be readily added to carry out the decoding of encoded data. GNURadio provides a graphical tool, GNURadio Companion,

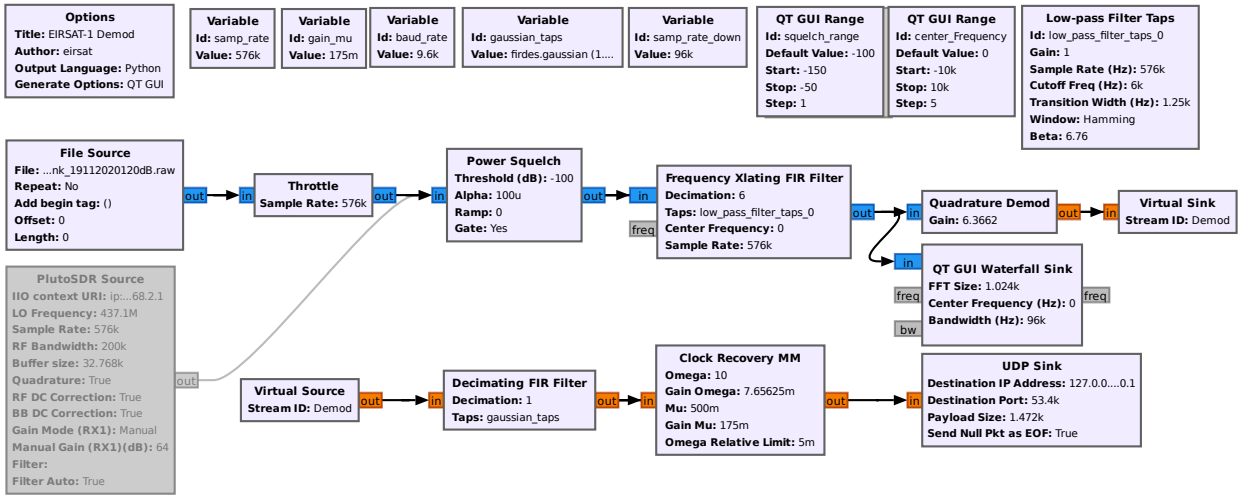

Fig. 2 The GNURadio flowchart that was built to accept the received signal from the SDR receiver, carry out the required signal processing and handle the demodulation of the signal. The flowchart can currently read from a file source of recorded IQ samples or from the Analog Devices Active Learning Module (ADALM) Pluto SDR receiver. The output is fed to a UDP sink to be routed to the decoding flowchart.

which is built around a graphical coding scheme where a user connects blocks of varying functionality to generate a flowchart, representing the communications pipeline. The GNURadio blocks can be implemented in both C++ and Python, with the computationally intensive blocks being written in C++ for performance reasons and the higher-level aspects being written in Python.

A. Receiving and Demodulation

The EIRSAT-1 mission implements a Gaussian Minimum Shift Keying (GMSK) modulation scheme that is transmitted at a baudrate of 9600bps for the downlink. The downlink adheres to the CCSDS Space Data Link protocol [16], ensuring that the transmitted frames are of constant length throughout the entirety of the mission. Figure 2 shows the GNURadio flowchart that was built to receive the downlinked data and apply the necessary signal processing steps to allow for a successful demodulation of the GMSK signal. The flowchart is configured to allow for the data to be either read directly from the ADALM Pluto SDR receiver or, if the data has been prerecorded, to be read from a file. It is important to note that a throttle is required if reading from a file to fix the sample rate and to limit CPU usage at runtime. When the data is routed from the SDR receiver, the listening frequency (437.1MHz), the sample rate, and the bandwidth are all configured in the PLUTO SDR source block. There is no native support for the PLUTO SDR in GNURadio, so the blocks themselves must be installed by the user. The power squelch block sets the lower power threshold and removes all signals below the threshold. This threshold can be varied during runtime by the operator.

Filtering pre-demodulation is important to remove high-frequency noise that is embedded in the received signal. The frequency translating Finite Impulse Response (FIR) filter block is configured to act as a low-pass filter. The relevant parameters are defined in the low pass filter taps definition block. Here the higher limit is defined, as well as the sharpness of the filter's edge. An option to introduce a gain to the signal is also available. The centre frequency parameter in the FIR filter block allows a frequency offset to be introduced which can be used to correct for Doppler shift when the spacecraft is in flight. The sample rate is also reduced here by a factor defined in the decimation field. A post-filtering waterfall plot allows the user to see when a signal has been received. The filtered signal is then fed into a quadrature demodulation block to carry out the demodulation of the GMSK signal. Here a type conversion occurs, denoted by the change in colour of the connection tabs of the blocks. The blue tab denotes an IQ sample with the orange tab output signifying the data being floating point numbers. The signal can be increased by the defined gain parameter.

A final filter is introduced before synchronisation between the timing of the transmitter and the receiver is established. The decimating FIR filter is configured to be a Gaussian filter that shapes the signal to a Gaussian profile. Here the gain of the filter is defined as well as the symbol rate, the bandwidth to baudrate ratio and the number of taps. The number of taps defines the amount of memory usage that is required when carrying out the signal filtering and also the number of

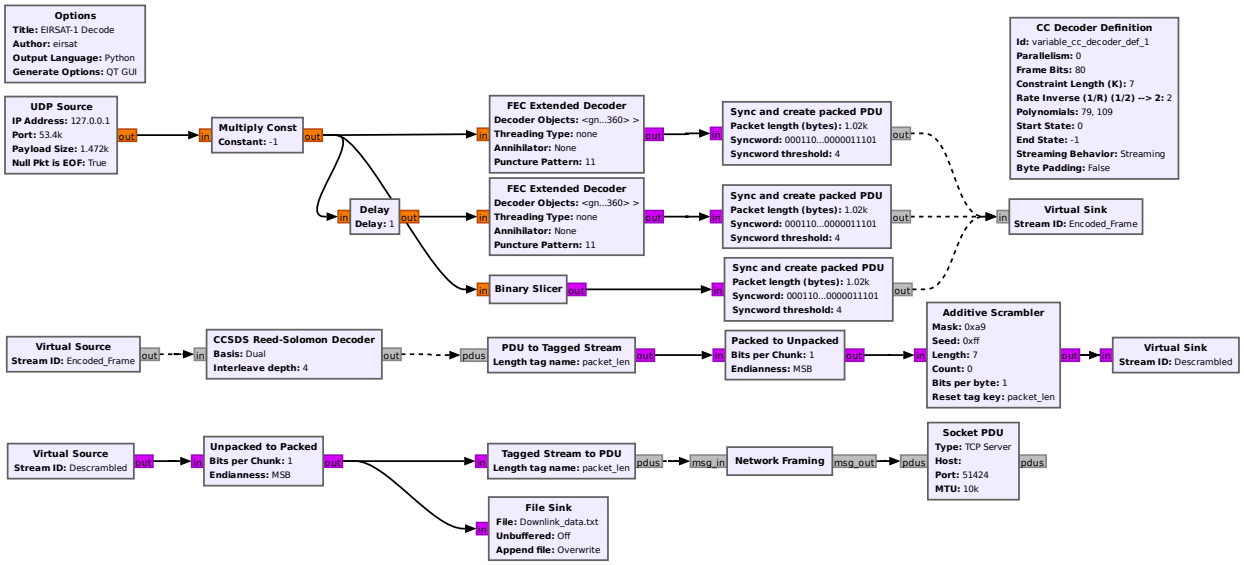

Fig. 3 The GNURadio flowchart that was developed to decode the EIRSAT-1 downlinked data. The flowchart receives the output data from figure 2 ovr UDP. The decoding flowchart implements a convolutional decoder, a Reed-Solomon decoder, and uses an additive descrambler to descramble the transmitted frames. The data then passes through network framing software to allow for the frames to be accepted by the MCS. The frames are sent from GNURadio to the MCS over a TCP connection.

calculations that occur during the filtering. The final stage before the decoding can occur is the synchronisation. The Clock Recovery MM block implements a Muller and Mueller discrete-time error-tracking synchroniser. The initial parameter, omega, is an estimate of the current number of samples per symbol. At this point, the sample rate is at 96kHz with the baudrate being 9.6kbps resulting in 10 samples per symbol. The block finds the centre point of each symbol and outputs one sample per symbol. The output are soft bits which are a representation of binary bits with any positive integer being interpreted as a binary one and a negative integer being interpreted as a binary 0. The output of the block is fed to a User Datagram Protocol (UDP) sink to be routed to the decoding stage of the EIRSAT-1 downlink chain.

B. Decoding

The EIRSAT-1 mission has opted to apply FEC codes to the downlinked frames, with the encodings varying depending on the state of the spacecraft. When the satellite is in nominal mode, which is the case for the majority of the mission, a Reed-Solomon (RS) scheme is used. If the satellite enters safe mode, then a concatenated encoding scheme will be implemented. Here the outer encoding scheme will be a Reed-Solomon code but an inner convolutional encoding scheme. The flowchart shown in Figure 3 handles both cases without the need for any reconfiguration. The Reed-Solomon encoding scheme adheres to the CCSDS TM Synchronisation and Channel Coding standards [17], with the concatenated FEC codes using non-standard polynomials for the convolutional encoding. GNURadio source tree modules don't support CCSDS FEC decoding. So to build the decoding flowchart, gr-satellites, the open source project developed by Daniel Estevez, was installed as an extension to GNURadio to provide extra blocks to handle the required functionality [18].

The decoding flowchart begins by receiving the data from the demodulating flowchart over a UDP connection. The data is fed into a multiply constant block and is multiplied by -1 to invert the data. It was found during the development stage that there was an inversion occurring at a point in the receiver chain causing all transmitted binary 0s to be interpreted as a binary 1 and visa-versa. As the current data is soft bits, the -1 inverts the data passing through the block to be recovered correctly.

The output of the inverting block is fed into two streams, the convolutional decoding stream, for when the satellite is in safe-mode and convolutional encoding is enabled and one which is a binary slicer for when only RS codes are enabled. These streams run in parallel during runtime. If no convolutional encoding is used, the output from the convolutional decoding stream will be incorrectly RS decoded and rejected. Similarly, if the concatenated scheme is used, the solely

RS decoding stream will fail the decoding and this data will be rejected. By having both streams running in parallel allows the flowchart to successfully recover the received data regardless of if a single FEC scheme or concatenated FEC scheme is used.

To decode the convolutional encoding, a Viterbi decoding is applied. The decoder is defined in the CC Decoder Definition with the decoding occurring at the FEC extended decoders. The convolutional encoding scheme that has been implemented for the EIRSAT-1 mission is a K=7, r=1/2. K is the constraint length and is equal to the length of the shift register. The shift register has k-1 storage elements with each element able to store 1 bit. The constant r denotes the rate of the convolutional encoding scheme. It is a ratio to the number of input bits to the number of output bits. For every 1 bit that enters the convolutional encoder, there are 2 output bits. The inverse is true at the decoder, with every 2 bits that enter the decoder, a single bit is output. The encoder and decoder use two polynomials to encode and decode the data. The implemented polynomials are given in equation 2 and 3 below:

$$G1 = 1 + x_1 + x_2 + x_3 + x_5 + x_6 = 111101 = 171(\text{octal}) \tag{2}$$

$$G2 = 1 + x_2 + x_3 + x_5 + x_6 = 1011011 = 133(\text{octal}) \tag{3}$$

GNURadio requires that the equations be input with the binary representation being reversed, resulting in octal values of 155 and 117 due to the least significant bit (LSB) denoting the coefficient of exponent zero of the polynomials. Finally these values are converted to decimal given the equations for input as 109 and 79. The decoder definition requires that the input of the polynomials are (G2, G1) resulting in an input of (79, 109). The flowchart incorporates two FEC extended decoders, with one running at a 1-bit delay. This is due to a potential phase ambiguity of 90°and the input bits may not be synchronised to either of the polynomials. Having one running at a 1 bit delay ensures synchronisation of the decoder throughout.

In the case when EIRSAT-1 is in nominal mode and no convolutional encoding is applied to the downlinked frames, the soft bits are converted into hard binary bits using the binary slicer. All three streams are then fed into the Sync and Create Packed PDU blocks. This is the stage where the original frames are recovered and packed. Pre-appended to the downlinked frames is an Attached Sync Marker (ASM) that denotes the start of the frame. In the case of EIRSAT-1 the ASM is `0x1ACFFC1D` which is a CCSDS standard. The block searches for the ASM and once found, the following bits are packed into bytes and a frame of 1020 bytes is created. An error threshold is defined which allows the ASM to be recognised, even with a certain number of errors.

The Reed-Solomon code that is employed for communications is a (255,223) scheme with an interleaving depth of 4. A (255, 223) scheme is where there are 255 bytes transmitted, with 223 of those bytes being data bytes and 32 bytes are parity bytes to allow for detection and correction of errors that can occur during transmission. An interleaving depth of 4 defines the size and structure of the downlinked frame. The resulting frame is 1020 bytes long (255×4). Within the frame, there are 4 223 byte data chunks followed by four 32 byte RS codes. The CCSDS TM Synchronisation and Channel Coding standards recommend a Dual basis RS codes, which is utilised by the EIRSAT-1 mission. At times it is possible for the location and the errors to be known pre-decoding. These known errors are called erasures. The RS decoder can correct up to 32 erasures within the 223 bytes, but if the errors are unknown prior to the decoding then only up to 16 bytes can be corrected. The output of the CCSDS Reed-Solomon Decoder is an 892 byte frame (1020 - (32*4) = 892). The bytes within the frame are then unpacked to allow for descrambling of the data to occur.

Prior to transmission, the frames are passed through a psuedorandomiser. If a constant stream of repeating symbols is transmitted, the receiver can lose synchronisation with the data and which leads to a loss of data. To ensure that this does not occur, the data is passed through a pseudorandomising equation. An additive (synchronised) descrambler is used to recover the pre-scrambled frame. The equation that is used to scramble the EIRSAT-1 data is given in equation 4. The hex representation of the polynomial is entered into the mask parameter of the descrambler block. The synchronisation of the descrambler is done using the seed value, in the case of EIRSAT-1 it is 11111111 = `0xFF`. The length defines the length of the shift register in bits with the shift register being cleared after each frame.

$$p = x^8 + x^7 + x^5 + x^3 + 1 = \texttt{0xA9}(\text{hex}) \tag{4}$$

The output of the block is an unpacked frame of bits that gets fed into a block that repacks the bits into bytes in MSB order.

The final stage before the data can be sent to the MCS requires network framing to be applied to the frame. The MCS uses an 8 byte header and a 4 byte checksum appended to the tail of the frame. This block was written specifically for the EIRSAT-1 flowchart using the Embedded Python Block. This is an empty block that allows the user to include

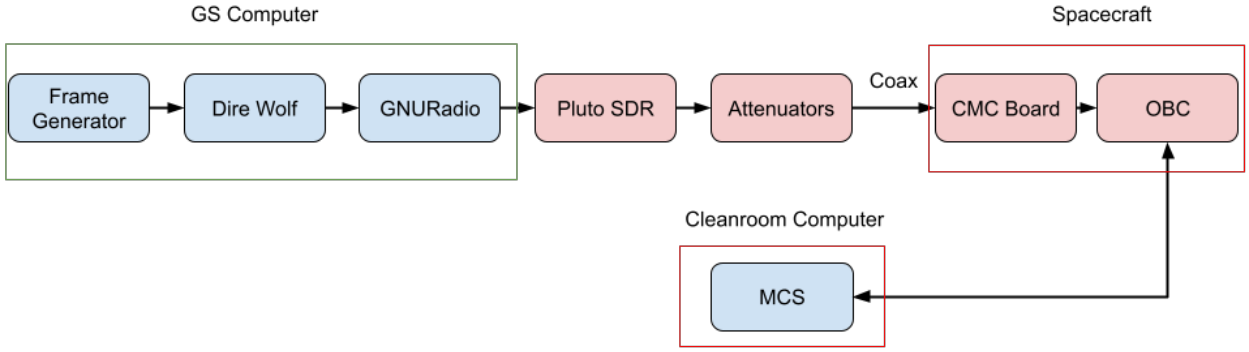

Fig. 4 The setup used for the BER test for the EIRSAT-1 uplink. The boxes in blue denote software components, with the red boxes denoting hardware components. A coaxial cable was used to connect the attenuators to the CMC board of the spacecraft.

Python code to carry out custom functionality. The required network framing header is 8 bytes long, with the initial 4 bytes being a predefined signature, and the second 4 bytes being the length of the frame. The 4 byte checksum is calculated by summing the entire frame and spreading this sum across 4 bytes. Once the framing has been applied, the data is fed to a Socket PDU to be sent over TCP to the MCS.

V. Testing

A suite of tests was developed and executed to assess the robustness of the communication system under different conditions. The idea is to send varying amounts of information through the system in different time intervals to determine whether both the uplink and downlink can perform correctly, regardless of the data amount and time frame. A Bit Error Rate (BER) test was carried out on both the uplink and downlink systems to identify the lower limit of the signal strength that is necessary for the receiver to recover the data correctly. The transmitted signal was attenuated to a range of receiving power levels that were based on prior link budget calculations. The data was transmitted over coax to ensure better control over the received signal strength than if the test was carried out over RF. In addition, to test the downlink a long period beacon test was carried out by leaving both the spacecraft and ground system running for 72 hours and have the ground station receive and log all the beacons that were transmitted over that time period.

A. Uplink BER Test

The aim of the uplink BER test was to find the number of erroneous transmitted bits per million bits transmitted. The signal from the ground station setup passed through a series of attenuators to reduce the signal strength to simulate attenuation levels to in flight conditions. The number of transmitted bits that were in error was calculated by finding the number of frames that were rejected by the spacecraft, either by an incorrect header or a mismatch between the CRC calculation carried out on board compared to that which was contained within the AX.25 frame. This results in a lower limit of the BER as it assumes that there is a single erroneous bit that results in the frame being rejected. Thus, this assumption assumes that the frame error rate is equal to the BER. The setup for this test is shown in Figure 4. As can be seen from this diagram, the actual hardware solution for the uplink deployment was replaced with a software solution. The software solution has the ability to control the transmit power of the PlutoSDR. An attenuation field is present in the PlutoSDR Sink block in GNURadio that allows for software to act as hardware attenuators to reduce the transmitted signal strength to the desired levels. The SCS Tracker TNC was replaced with the software TNC Dire Wolf [19], with the functionality of the ICOM transceiver being replaced with a GNURadio flowchart, which can be seen in Figure 5, and the Pluto SDR.

The KISS frame generator software was used extensively throughout this test. The KISS frames that were generated had no functional commands for the satellite, but rather the information field was filled with random alphanumeric characters. An AX.25 frame that has a filled 256 byte information field with no digi-peaters is of length 274 bytes or

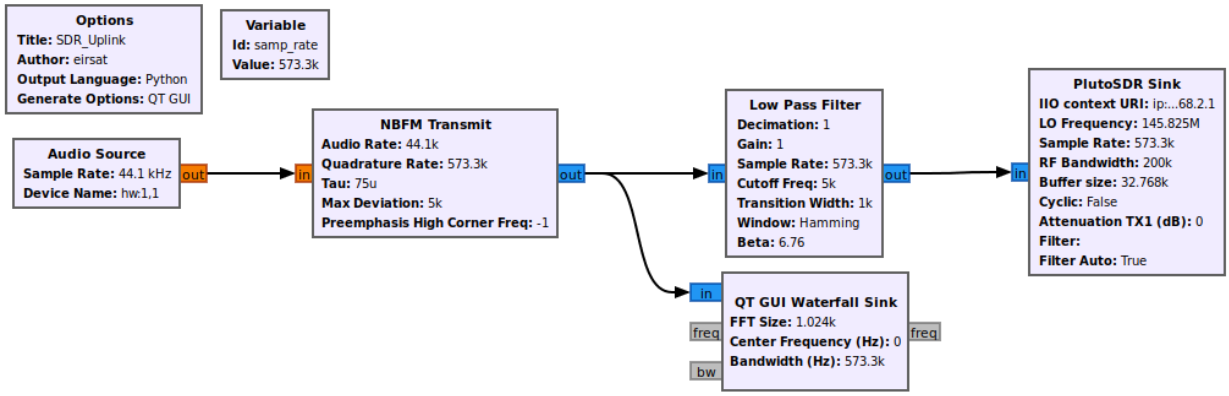

Fig. 5 The GNURadio flowchart that was developed to carry out the transmission of the audio signal received from the software TNC Dire Wolf.

2192 bits. Therefore, to send 1 million bits a total of 453 frames are required. The KISS frames that were generated were sent to Dire Wolf over TCP where the AX.25 framing and AFSK modulation took place. Dire Wolf was configured to output the AFSK audio samples to the GNURadio flowchart using an audio loopback interface. The Narrowband FM transmitter block converts the floating point audio samples to a modulated signal. This signal passed through a low pass filter and was routed to the PlutoSDR using the PlutoSDR sink where the uplink transmitting frequency was set at f = 145.82MHz. The PlutoSDR sink has the added functionality of adding a software attenuation to the signal in dB. Finally, the 145.82MHz radio signal passes through the hardware attenuators where the signal strength is reduced to the desired level. The PlutoSDR has a transmitting power of 1mW or 0dBm. The range of the receiving signal strength was decided to be from -70dBm to -110dBm with the attenuation increasing in steps of 5dB.

The number of erroneous bits transmitted was given by counting the number of frames rejected by the spacecraft due to an incorrect header or due to an incorrect CRC. The MCS ran on a computer that was connected directly to the spacecraft over a serial connection. In this way it was possible to track the number of frames received by the spacecraft. If the number of frames does not match the number of transmitted frames, this indicates that errors occurred in the header of the AX.25 frame. A frame accepted by the CMC board goes through the CRC check prior to being accepted by the OBC. The number of frames rejected to an incorrect CRC was also computed.

The results for the BER test for the uplink are given in Figure 6 and 7. The total number of frames received by the spacecraft is shown in Figure 6. To reduce variation in the results, a total of 3 million bits were transmitted at each power level, rather than 1 million bits. These results were then averaged to show the error rate for a transmission of 1 million bits. The receiving signal strength of -90dB and -95dB were the most stable with no errors occurring at the -90dB level and a single frame dropped at -95dB. Once the received signal strength reaches -105dB, the number of frames received drops exponentially with almost half of the transmitted frames containing errors. Once the signal strength reaches -110dB, a single frame was received. This shows that when the spacecraft is in flight, the required signal strength for a stable link between the spacecraft and the ground should remain above -100dB.

Figures 7 provide a breakdown of the causes for frames to be rejected. It was expected prior to conducting the experiments that the majority of frames would be rejected due to a CRC error, which at most received signal strengths is the case. Between -70dB and -100B, nearly all frames rejected were by a CRC error. Once the received strength reaches -105dB, the number of frames rejected by a CRC error versus an erroneous header are almost the same (an average of 117 being rejected due to a CRC error and 112 due to an incorrect header). The exponential increase in the number of frames rejected by CRC errors is lost, with almost half the number of CRC errors occurring at this power level than that of -105dB. A total of 56 CRC errors occurred at -110dB. This decrease is due to a massive increase in the number of erroneous headers. An average of 397 frames were rejected to an incorrect header.

B. Downlink BER Test

A similar process to the uplink was carried out by sending large amounts of data through the link and finding the number of errors that occurred during transmission. Unlike the BER uplink test, the results will present the total number of erroneous bytes rather than erroneous frames. The hardware and software setup used for this test can be

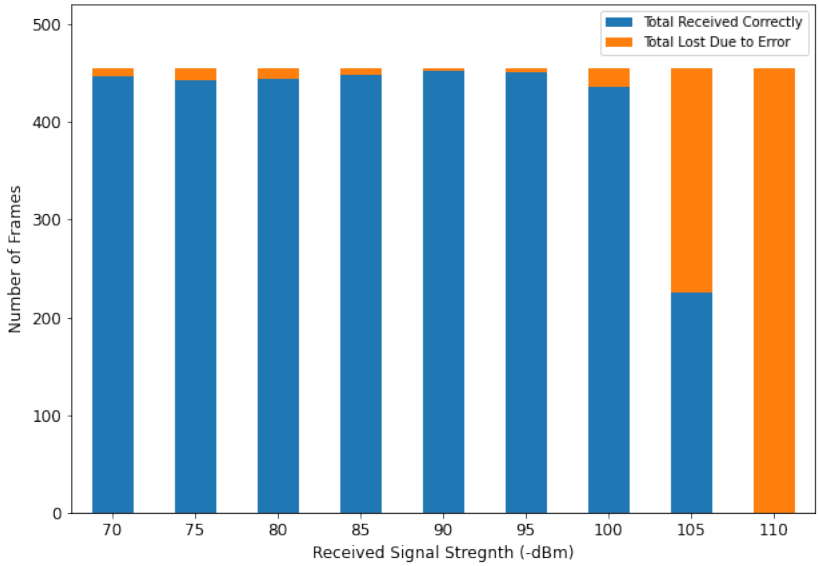

Fig. 6 Results for the BER test for the EIRSAT-1 uplink. The blue bars denote the total number of frames that were successfully received by the spacecraft during the test with the orange representing the total number of frames that were lost due to an error. The type of errors that occurred can be seen in Figure 7.

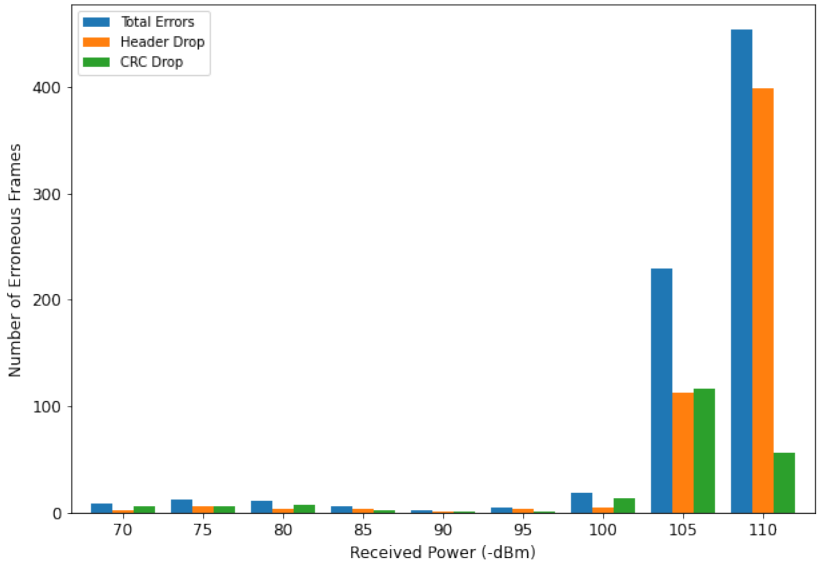

Fig. 7 The breakdown of the number of errors that occurred during the BER uplink test. The blue bars denote the total number of errors that occurred, the orange denotes that a frame was rejected due to an error in the header and the blue signifies an incorrect CRC.

seen in Figure 8. The spacecraft is connected to the PlutoSDR over a coaxial connection with 25dB attenuation of the spacecraft side of the coax and a varying amount of attenuators present on the PlutoSDR side. The data received by the PlutoSDR is connected to the ground station computer over USB and is routed to the receiving and demodulating flowchart displayed in Figure 2. During the test, the spacecraft was transmitting with a power of 0.5W or 27dB.

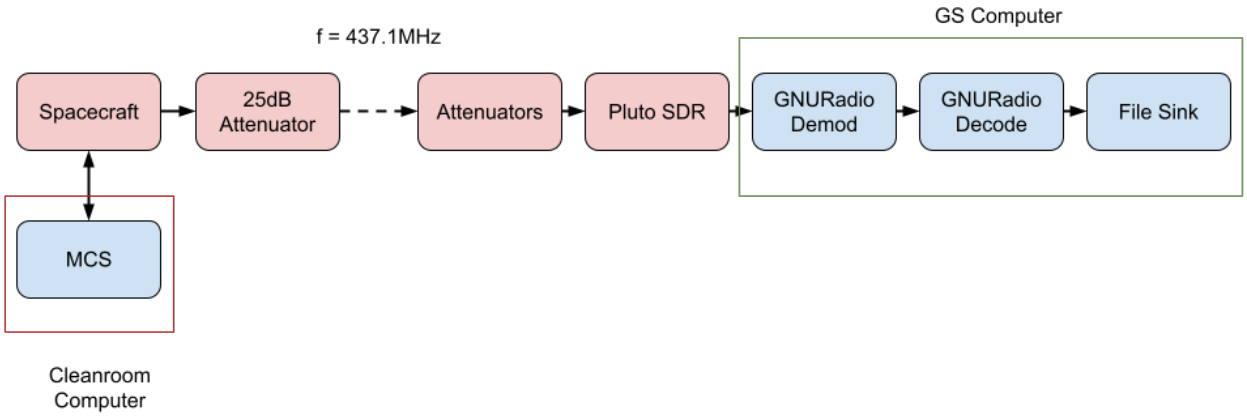

Fig. 8 The setup used to carry out the BER test for the EIRSAT-1 downlink. The MCS was run on the cleanroom computer, which connects to the spacecraft over serial, to interface with the spacecraft. The downlinked data is sent both over serial to the MCS and transmitted over RF to the ground station communications system.The red boxes in denote hardware in the configuration while the blue denotes software.

Data is stored on board in channels, with each channel storing the data in rows. Each row has a predefined structure and therefore a predefined length. The spacecraft was queried to find a channel that has enough rows stored to transmit a million bits. A channel with a row length of 125 bytes was selected as there were 1299 rows present in the channel. This operation was carried out using the MCS which was running on the computer connected to the spacecraft over a serial connection. As the communications adhere to the CCSDS Space Data Link protocol, a constant downlink frame length of 1020 bytes is employed. These 1299 rows were spread across 221 frames, resulting in a total of 1,803,360 bits transmitted at each attenuation level. The number of erroneous bits that are present are found by recording the output of the Reed-Solomon decoding stage of the decoding flowchart in Figure 3. As RS codes allow for correction of a number of errors, the number of bytes that were corrected for each frame were found. The results for the test can be viewed in the following subsection.

The results for the BER test for the downlink can be seen in Figure 9. The results show the number of bytes that were corrected for all 221 frames between a received signal strength of -64dB to -120dB. The number of RS bytes corrected for each frame is printed to the message debug in GNURadio. The message debug was captured for the entire test and the number of bytes corrected for each frame were summed.

It can be seen in Figure 9, that there is a single outlier point at a received signal strength of -100dB. For the 221 frames that were transmitted, a total of 57 bytes were corrected. These 57 corrections came from a single frame that was transmitted. As RS codes with an interleaving depth of 4 are implemented, it was possible to find the number of erroneous bytes in each of the data blocks. A total of 15 bytes were corrected in the first of the RS code blocks, with each of the proceeding three code blocks containing 14 erroneous bytes. It is unknown as to why this single frame contained such a large number of incorrect bytes compared to all other frames. It can be seen that at a receiving signal strength of -68dB, within the 221 frames, just a single byte was corrected and at -74dB only 4 bytes were corrected. At all other receiving power levels not a single byte was transmitted incorrectly and all bytes were recovered by the GNURadio flowcharts correctly. Even at the -100dB power level with the single frame being transmitted with 57 errors, there is still a 99.975% accuracy in transmission. Throughout the full test, a total of 2,705,040 bytes were transmitted with a total of 62 bytes were recovered incorrectly.

C. Long Period Beacon Test

A long period beacon test was designed to verify that the downlink solution is stable when running over an extended period of time. Even though the communication passes are between 4 and 7 minutes in length, it is important to show that the solution is reliable at correctly receiving data for periods of time longer than this. The test was carried out from a Friday afternoon, with the satellite being left on transmitting the beacon once every minute, to a Monday morning where the downlink flowcharts were stopped. The frames that were received by the ground station setup were saved to a file for analysis. The setup for the test can be seen in Figure 10.

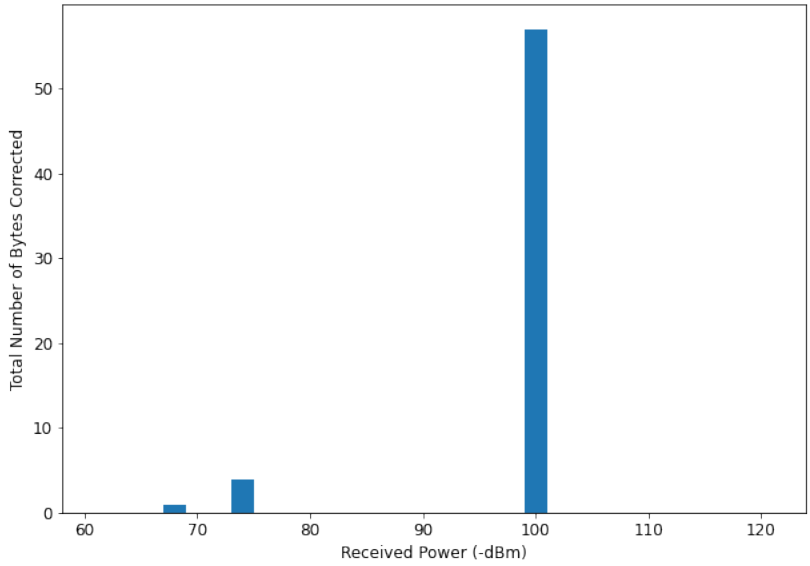

Fig. 9 Results for the BER test for EIRSAT-1's downlink. The received power level ranged between -62dB and -120dB. The plot displays the total number of bytes corrected for 221 frames at the RS decoder in GNURadio at each power level.

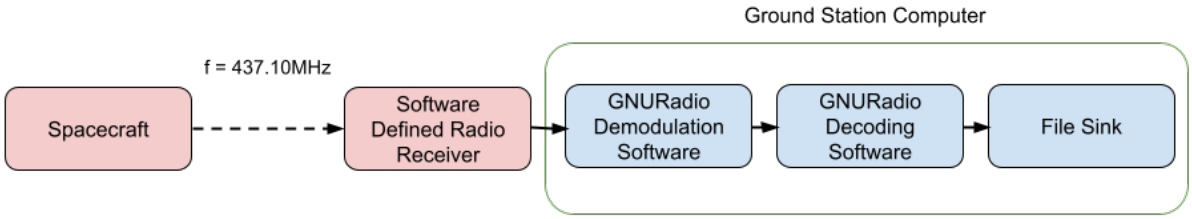

Fig. 10 The setup that was used to carry out the beacon test. The red blocks signify hardware components for the test with the blue signifying software components. The dotted line connecting the spacecraft to the Pluto SDR denotes an RF link.

Unlike the BER test that was carried out over coaxial cable, this test was carried out over RF using a small antenna in the EIRSAT-1 lab connected to the TX port of the CMC board and the antenna that came with the PlutoSDR. The frames that were recorded were passed through parsing software that was written by another member of the team to allow for extractions of the parameters that are stored in the house keeping beacon. For correct extraction of the data from the frames, the database definition that is used to keep track of the on-board parameters is needed as well as the the channel aggregator. The aggregator is the bit structure of the data contained in the channel.

The test itself ran for just under 72 hours, during which 4,183 frames in total were received by the ground station setup. To visualise the results, we extract parameters stored in the housekeeping beacon and plot them over the period of time the test was carried out. By plotting these parameters, we can see not only shows the correct functionality of the ground station downlink software, but also the performance of the on-board hardware components and logging and parsing software. The parameters that were extracted from the housekeeping packets were the on-board time that the row was created, the temperature of the CMC transmitter, and data from the one of the temperature sensors mounted externally on the spacecraft. The results are given in Figures 11, 12 and 13.

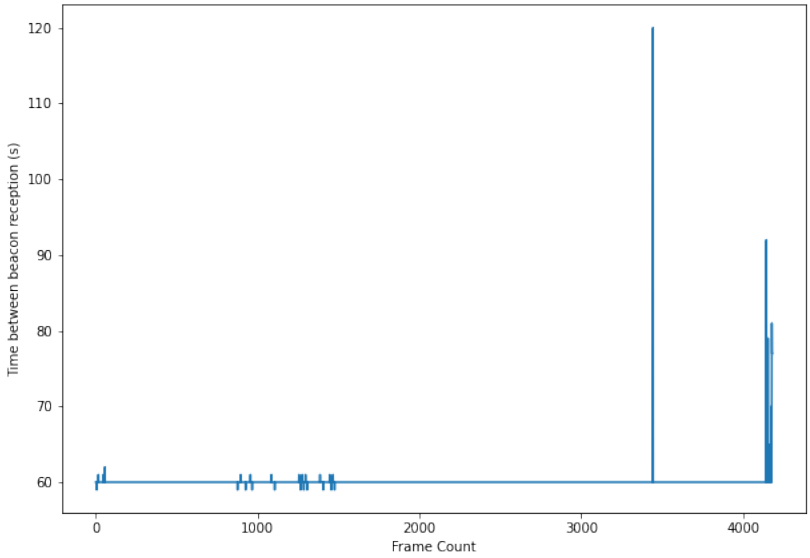

Fig. 11 Plot to show the difference between each of the timestamps contained within the received frames, where we subtract the time from the previous frame from the current frame. The expected time difference should be 60 seconds

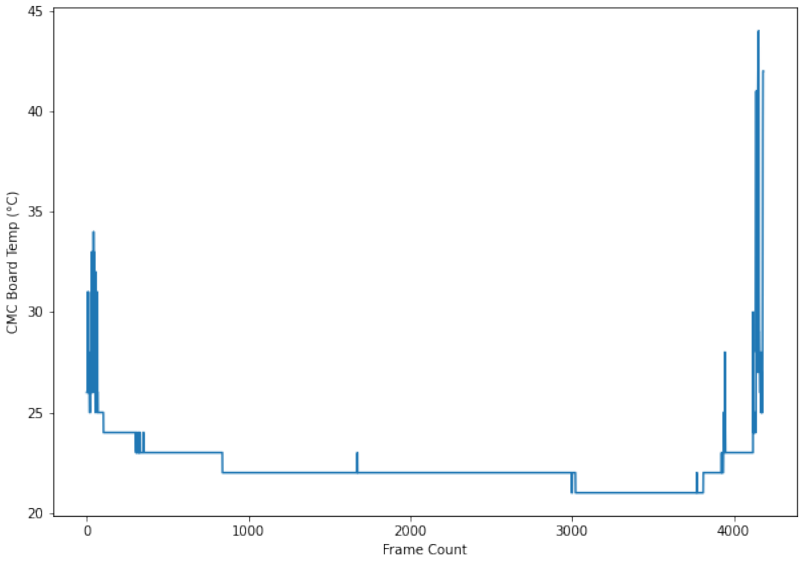

Fig. 12 The temperature of the transmitter on the CMC. The temperature on the Friday evening is high as just before the test began other team members were transmitting data. The temperature drops over the weekend and spikes again on the Monday on the right, when work began again on the spacecraft.

Figure 11 shows the time difference between each of the timestamps extracted from the housekeeping frames. The beacon is set to transmit once every 60 seconds. From the plot we observe that this is true in the majority of cases. It can be seen that at frame 3,445, the time difference between the received housekeeping frames is at 120 seconds. This signifies that there was a single frame dropped. From viewing of the output logs of GNURadio it was found that

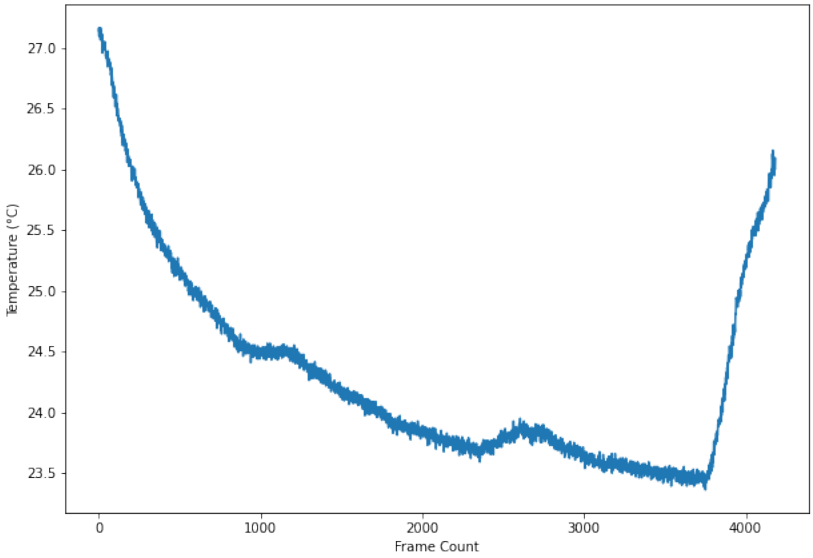

Fig. 13 The temperature, in °C, of the ADCS board using sensor 1(of 4). We observe the temperature dropping from 27 °C to 24 °over the weekend and rising again on the Monday on the right-hand side of the plot.

this frame was dropped due to too many errors contained within the frame for the RS decoder to correctly recover the frame. In this case, the errors could be over the limit of the number of errors the RS codes can correct. Figure 13 shows that towards the end of the test period, the beacon was often delayed. This occurs when a lot of data is being downlinked from the spacecraft as downlink data has a higher priority than the beacon. At frame 4140, a member of the team was carrying out tasks on the spacecraft and the level of data being transmitted increased. This had the effect of increasing the time between beacon transmissions. Figure 12 also shows that the team member was using the spacecraft. Throughout the weekend, the temperature of the transmitter was between 20 and 25°C, but on the Monday the temperature increases above 40°C. This rise in temperature aligns with the increased interval between housekeeping frames. Both of these occurrences are due to the increase in usage of the transmitter to transmit data with a higher transmit priority than that of the beacon.

The final plot, Figure 13, was generated from the housekeeping beacon data. It illustrates the temperature retrieved from one of the external temperature sensors. The data shows the correct functionality of the temperature sensor as the ambient temperature in the EIRSAT-1 cleanroom drops from 27°C to a minimum of 23.5°C and begins to rise again. These changes in temperature align with the power cycle of the heating system in the cleanroom.

VI. Conclusion

The EIRSAT-1 ground segment communications system has been successfully developed and tested at UCD. The results from the uplink BER test show the received signal strength threshold that must be attained for correct transmission and reception of TCs. A potential way to increase this threshold would be to introduce a method of FEC codes. FX.25, similar to AX.25 but including RS codes, could be introduced into the EIRSAT-1 communication system. An effect of incorporating the FEC codes to the uplink would be to increase computational power on-board, which could change the EIRSAT-1 power budget. Similarly, a new TNC would need to be purchased or designed to incorporate the added FEC codes into the framing as standard TNCs only implement the standard AX.25 framing. As the data is transmitted from the ground, the ICOM-9100 provides the possibility of transmitting up to a power of 100W. If the received signal strength is too low, then the ground does have the option of increasing the transmitting power and thereby increasing the received signal strength.

The SDR approach used for the downlink has given a great deal of flexibility in the communication system, while also supporting the robustness of the system. The BER downlink test demonstrated that, even down to a receiving signal strength of -120dB, the GNURadio flowchart is able to successfully recover all data that has been transmitted.

Test	Test Aim	Test Result	Figures
BER Uplink	Find the number of erroneous bits transmitted at varying attenuation levels to find the minimum received signal strength for stable uplink.	Stable link at received signal strengths above -100dB. Exponential drop off in the number of frames received below this received signal strength	6, 7
BER Downlink	Find the number of erroneous bits transmitted at varying attenuation levels to find the minimum received signal strength for stable downlink.	Stable link down to received signal strength of -120dB.	9
Long Period Beacon	Leave the ground station software running for a long period of time to collect transmitted beacon data from the spacecraft to ensure the reliability of the software for extended periods of time.	Successfully received 4183 beacon frames over the course of a weekend with a single frame being dropped.	11, 12, 13

Table 3 Summary of the three tests that were carried out to ensure the reliability and limitations of the ground segment communication software. The BER uplink, BER downlink and the Long Period beacon test are briefly summarised with the main results presented and the figures the results are presented.

Having no incorrect bytes received at this power level is a strong indication that, at a transmitting power of 0.5W, communication with the spacecraft will be effective throughout the mission. It should be noted that, if the test was repeated, the attenuation levels might be increased so that a lower received signal strength to find where the lower threshold to allow for correct reception and recovery of the transmitted telemetry. Similarly, the performance of the long period beacon test was encouraging. Having the system left run for almost a three day period and receiving all transmitted frames with only a single error indicates good stability of the system while the spacecraft is in flight, thereby eliminating the need for constant restarts prior to the initiation of the communication window.

A final test is yet to be undertaken due to the development of the space segment. This is a long distance test, where EIRSAT-1 will be positioned at a distance of approximately 20km within direct line of sight of UCD, so that the two-way communications can be tested. This will allow us to evaluate the full functionality of the ground segment, as well as the on-board communications system. This test will mark another significant milestone in the lead up to the EIRSAT-1 launch.

VII. Acknowledgments

The EIRSAT-1 project is carried out with the support of the Education Office of the European Space Agency under the educational Fly Your Satellite! programme. The EIRSAT-1 team acknowledges support from ESA under PRODEX contract 4000124425. RD acknowledges support from the Irish Research Council (IRC) under grant GOIPG/2019/2033. MD acknowledges support from the IRC under grant GOIP/2018/2564. FM acknowledges support from the School of Computer Science, UCD. SMB, DM and JM acknowledges support from Science Foundation Ireland under grant number 17/CDA/4723. LH acknowledges support from SFI under grant 19/FFP/6777 and from the EU AHEAD2020 under grant agreement 871158. DM and JT acknowledge support from the IRC under grants GOIPG/2014/453 and GOIPG/2014/684. JE and JR acknowledge scholarships from the UCD School of Physics. SW acknowledges support from the European Space Agency under PRODEX contract number 400012071. LS acknowledges support from the IRC under grant number GOIPG/2017/1525. This study was supported by The European Space Agency's Science Programme under contract 4000104771/11/NL/CBi. We acknowledge all students who have contributed to EIRSAT-1.

References

[1] Murphy, D., Flanagan, J., Thompson, J. W., Doyle, M., Erkal, J., Gloster, A., O'Toole, C., Salmon, L., Sherwin, D., Walsh, S., et al., "EIRSAT-1–the educational Irish research satellite," *Proceedings of the 2nd Symposium on Space Educational Activities*, Vol. 8859, 2018.

[2] Murphy, D., et al., "A compact instrument for gamma-ray burst detection on a Cubesat platform I:Design drivers and expected performance," In prep.

[3] Murphy, D., et al., "A compact instrument for gamma-ray burst detection on a Cubesat platform II:Detailed design and laboratory characterisation," In prep.

[4] Murphy, D., et al., "EIRSAT-1 - The Educational Irish Research Satellite," *Proceedings of the 69th International Astronautical Congress (IAC), Bremen, Germany, 1-5 October 2018*, 2018.

[5] O'Connor, W. J., et al., "Wave-based control of non-linear flexible mechanical systems," *Nonlinear Dynamics*, Vol. 57, No. 1-2, 2008, pp. 113–123. https://doi.org/10.1007/s11071-008-9425-4.

[6] Sherwin, D., et al., "Wave-based attitude control of EIRSAT-1, 2U CubeSat," *2nd Symposium on Space Educational Activities. April 11-13 2018, Budapest, Hungary*, 2018. Paper ID: SSEA-2018-93.

[7] Thompson, J., et al., "Double-dipole antenna deployment system for EIRSAT-1, 2U CubeSat," *2nd Symposium on Space Educational Activities. April 11-13 2018, Budapest, Hungary*, 2018. Paper ID: SSEA-2018-78.

[8] Walsh, S., Murphy, D., Doyle, M., Thompson, J., Dunwoody, R., Emam, M., Erkal, J., Flanagan, J., Fontanesi, G., Gloster, A., et al., "Assembly, Integration, and Verification Activities for a 2U CubeSat, EIRSAT-1," *arXiv preprint arXiv:2010.10425*, 2020.

[9] Dunwoody, R., Doyle, M., Murphy, D., Finnernan, G., O'Callaghan, D., Reilly, J., Thompson, J., Walsh, S., Erkel, J., Fontanesi, G., Kyle, J., Mangan, J., Marshall, F., Salmon, L., de Faoite, D., Greene, D., Hanlon, L., Martin-Carrillo, McKeown, D., O'Connor, W., Wall, R., and McBreen, S., "Development and Validation of the Operations Procedures and Manual, for a 2U CubeSat, EIRSAT-1, with Three Novel Payloads," SpaceOps 2021, SpaceOps-2021,2,x1349, 2021.

[10] Doyle, M., Gloster, A., O'Toole, C., Mangan, J., Murphy, D., Dunwoody, R., Emam, M., Erkal, J., Flanaghan, J., Fontanesi, G., et al., "Flight Software Development for the EIRSAT-1 Mission," *arXiv preprint arXiv:2008.09074*, 2020.

[11] Beech, W. A., Nielsen, D. E., Noo, J. T., and Ncuu, L. K., "Ax. 25 link access protocol for amateur packet radio, version: 2.2 rev," *Tucson Amateur Packet Radio Corp*, Citeseer, 1997.

[12] Chepponis, M., and Karn, P., "The KISS TNC: A simple Host-to-TNC communications protocol," *6th Computer Networking Conference*, Vol. 225, 1987, pp. 06111–1494.

[13] Ulversoy, T., "Software defined radio: Challenges and opportunities," *IEEE Communications Surveys & Tutorials*, Vol. 12, No. 4, 2010, pp. 531–550.

[14] Baldini, G., Sturman, T., Biswas, A. R., Leschhorn, R., Godor, G., and Street, M., "Security aspects in software defined radio and cognitive radio networks: A survey and a way ahead," *IEEE Communications Surveys & Tutorials*, Vol. 14, No. 2, 2011, pp. 355–379.

[15] "GNU Radio, the free & open software radio ecosystem," http://gnuradio.org, 2015.

[16] Sanchez, I. A., Moury, G., and Weiss, H., "The CCSDS space data link security protocol," *2010-MILCOM 2010 MILITARY COMMUNICATIONS CONFERENCE*, IEEE, 2010, pp. 219–224.

[17] Synchronization, T., and Coding, C., "Recommended Standard CCSDS 131.0-B-2," *Consultative Committee for Space Data Systems, Blue Book*, , No. 3, 2017.

[18] Estevez, D., "gr-satellites," , 2020. URL https://github.com/daniestevez/gr-satellites.

[19] Direwolf, "Direwolf Github," , 2020. URL https://github.com/wb2osz/direwolf.