Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 02, February-2021

Implementation of Reed-Solomon Codes for Stratospheric Balloon Probes and Nano-Satellites

Eduardo Valadez Campos
Department of astrophysics
Instituto Nacional de Astrofisica,
Optica y Electronica
Puebla, Mexico

Jose Eduardo Mendoza Torres
Department of astrophysics
Instituto Nacional de Astrofisica,
Optica y Electronica
Puebla, Mexico

Abstract – **The digital communications and data handling technics are essential in any system that require to store and transmit information from one point to another, and if the system has to work remotely, the software, as much as the hardware, is a key part when it comes to compensate errors in the data due to the environments with high electromagnetic noise, fluctuating power source or a changing visibility window to transmit. Usually, the purchase of a communication system with a sophisticated built-in Error Correction Codes (ECC) are quite expensive and, depending the country, always there is security restrictions. Therefore, in order to increase the performance of scientific data gathered from stratospheric balloon probes and nano-satellites using entirely low-cost commercial components, we develop a communication system to transmit/receive data files above 5KB with low-cost commercial RF transceivers, commercial ARM-based mini-PC and using an implementation of Reed-Solomon codes.**

Keywords* – *Tx; Rx; Reed-Solomon Codes; Error Correcting Codes; Components-Off-The-Shelf; ARM-based; Galois field; codec.

1. INTRODUCTION

The use of Components-Off-The-Shelf or COTS based solutions has allowed to develop radio-probes, payloads and mini-satellites with modern tools and a minimal cost that a couple decades ago, by getting full advantage of the economy of scale with cheap and plentiful components. Due to the typically shorter lifespan of these devices compared to traditional endeavors, it is possible to use newer, more innovative designs without running a high financial risk. This is interesting as it allows for rapid development and advancement in an otherwise conservative industry where only the large companies and agencies could afford such developments [1][2].

However, in order to increase the reliability of the COTS-based devices, it is necessary to make thoroughly analysis of the environment conditions where the devices are going to operate and to replicate such conditions. Then, identify the weak points where the device shows poor performance or a risk of failure, so the development team can improve/replace/optimize such features [3].

When it comes down to data gathering, the performance improvement of the digital communication system is possible. One way to do this is to increase in the signal-to-noise ratio (SNR) by increasing the transmission power or by changing the antenna gain [4], but usually the transceivers with high transmit power are expensive, without mention the transmit permits (depending of the country) required to do so. Also, the physical size restrictions of the nano-satellites and balloon probes makes unfeasible the increase of the power source [5]. The other method is the implementation of a Forward Error Correction (FEC), that is a type of ECC that does not require handshaking between the source and the destination. It is less costly than these techniques. Introducing channel coding in a communication system introduces the possibility of approaching the maximum transmission rate theoretically possible [4]. The FEC method is favorable because have a lower consumption of bandwidth and a high-efficiency error correction. This is the reason FEC is suitable for applications in wireless communication [6]. Hence the Reed-Solomon codes are chosen

The proposed solution is to have a R-S codec implemented in our communication system. The data is compressed in certain format (depending of the type of file), divided in smaller packages, encoded and transmitted through a RF transceiver. From the other side, the Receiver Control Software gather the packages, decode them and detect any erroneous byte and determine their exact location inside the package in order to intent to correct them.

To test that the algorithm does work, we make two tests:

1. Burst of errors: By passing the data packages through a Random Number Polynomial Generator. In this test we determine the number of errors inserted.
2. Random errors: By Adding an Additive White Gaussian Noise (AWGN) array, to the data package. The probability of error is independent from one transmitting byte to the next.

For the moment we are not conducting any further tests concerning to the RF signal or the transceivers.

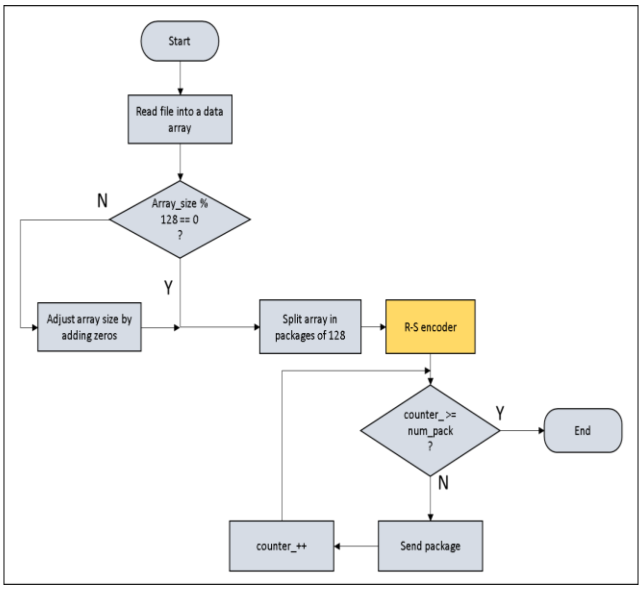

Start
Read file into a data array
Array_size % 128 == 0 ?
N
Y
Adjust array size by adding zeros
Split array in packages of 128
R-S encoder
counter_>= num_pack ?
Y
N
End
Send package
counter_++

Fig 1. *Flowchart of single-send mode*

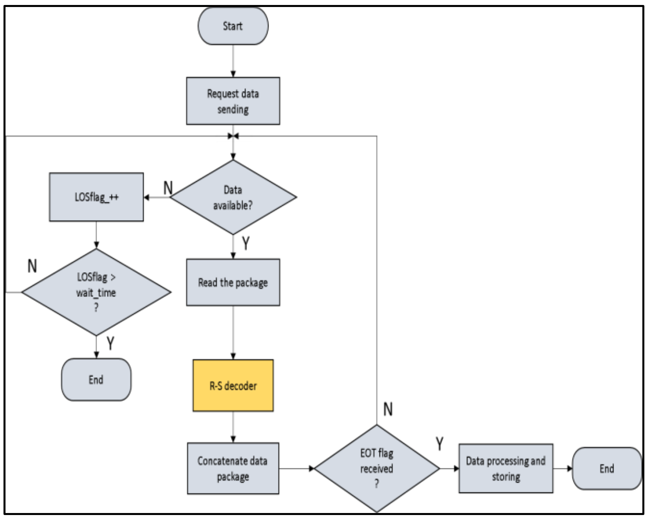


Fig 2. *Flowchart of the uni-directional receiver function in the Control Software*

2. REED–SOLOMON CODES

The Reed-Solomon (R-S) codes are a subset of BCH codes (Bose-Chaudhuri-Hocquenghem codes), that is a class of cyclic error-correcting codes that are constructed using polynomials over a Galois field [8][9][10]. With a wide range of applications in digital communications and storage, Reed–Solomon codes are able to detect and correct multiple symbol errors [7].

A Reed-Solomon code is specified as $RS(n,k)$ with m-bit symbols [14]. This means that the encoder takes "k" data symbols of "m" bits each and adds parity symbols to make an "n" symbol codeword. There are $n-k$ parity symbols of "m" bits each (Fig. 3). A Reed-Solomon decoder can correct up to t symbols that contain errors in a codeword, where $2t = n-k$.

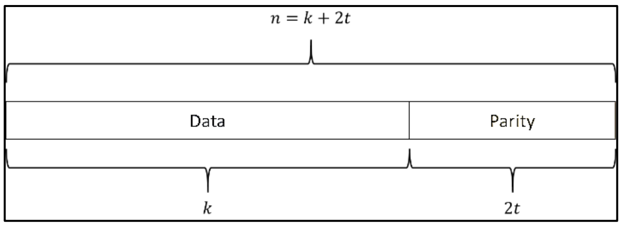


Fig 3. *Typical Reed-Solomon codeword*

Where:

$block\ length = n = k + 2t$

$Possible\ errors\ to\ detect = 2t$

$Possible\ errors\ to\ correct = t$

The R-S codes may be shortened by (conceptually) making a number of data symbols zero at the encoder, not transmitting them, and then re-inserting them at the decoder.

In our software, we already know that this zero-bytes are not important, so we adapt the calculations so we can ignore these added zero-bytes and save computing resources.

We define the package size to encode/decode:

$$k = 128\ bytes$$

The proposed the number of bytes of parity and hence the max number of error detection per package k:

$$2t = 32\ bytes$$

Number of bytes than can be corrected per package of k:

$$t = 16\ bytes$$

When a target file is requested to the on-board computer, the data array is divided into small pieces of 128 bytes, then it is encoded and concatenated with its respective parity bytes and then send. For the purpose of this paper, we are only going to describe briefly the mathematic procedure followed to implement the R-S codes without taking into details.

2.1. FINITE (GALOIS) FIELD

Reed-Solomon codes are based on a specialist area of mathematics known as Galois-fields algebra. A Galois-field or finite-field is a set of elements in which we can do addition, subtraction, multiplication, and division without leaving the set. The operation of addition and multiplication must satisfy the commutative, associative, and distributive laws. The number of elements in a field is called the order of the field and, according to Galois, in order for a filed to be finite, the numbers of elements are P^m where P is a prime number and m is a positive integer (Eq. 1).

$$GF(P^m) = \{0,1, \ldots., P^m - 1\} \qquad (1)$$

As we are dealing with information represented in binary, we are going to focus in the binary field $GF(2)$ and the extension field $GF(2^m)$ (Eq. 2 and 3), where addition is EXCLUSIVE OR (XOR) and multiplication is AND. Since the only invertible element is 1, division is the identity function.

$$GF(2) = \{0, 1\} \tag{2}$$

$$GF(2^m) = \{0,1,\alpha,\alpha^2,\alpha^3,\ldots,\alpha^{2^m-2}\} \tag{3}$$

The elements in the extended $GF(2^m)$ are m-bit symbols. Since we are dealing with arrays of bytes, hence:

$$GF(2^8) = GF(256) = \{0,1,\alpha,\alpha^2,\ldots,\alpha^{254}\} \tag{4}$$

2.2. PRIMITIVE POLYNOMIAL AND FIELD SYMBOLS

Polynomials over the binary field $GF(2^m)$ are any polynomials with binary coefficients. Each of these polynomials, denoted as $f(x)$, is simply the product of its irreducible factors. An irreducible polynomial $f(x)$ of degree m is said to be primitive $p(x)$ if the smallest positive integer n for which $f(x)$ divides $x^n + 1$ is $n = 2^m - 1$ The primitive polynomials are polynomials that generate the elements contained in a finite field which in turn are needed to define the R-S Codes.

The primitive polynomial determined for a $GF(2^8)$ are shown in the table 1:

TABLE 1. PRIMITIVE POLYNOMIALS OF GF (256)

$x^8+x^4+x^3+x^2+1$
$x^8+x^5+x^3+x+1$
$x^8+x^5+x^3+x^2+1$
$x^8+x^6+x^3+x^2+1$
$x^8+x^6+x^4+x^3+x^2+x+1$
$x^8+x^6+x^5+x+1$
$x^8+x^6+x^5+x^2+1$
$x^8+x^6+x^5+x^3+1$
$x^8+x^6+x^5+x^4+1$
$x^8+x^7+x^2+x+1$
$x^8+x^7+x^3+x^2+1$
$x^8+x^7+x^5+x^3+1$
$x^8+x^7+x^6+x+1$
$x^8+x^7+x^6+x^3+x^2+x+1$
$x^8+x^7+x^6+x^5+x^2+x+1$
$x^8+x^7+x^6+x^5+x^4+x^2+1$

The equation 5 is the primitive polynomial used in our algorithm:

$$p(x) = x^8+x^4+x^3+x^2+1 \tag{5}$$

The field symbols or primitive elements $\propto^i$ that conform $GF(2^m)$ are determined using the modulo method (Eq. 6)

$$\alpha^i = \alpha(x)\ \ mod\ \ p(x) \tag{6}$$

TABLE 2. SAMPLE OF THE FIRST 27 ELEMENTS OF GF (256)

$0 = 0\ mod\ p(x) = 0$	$\alpha^8 = x^8\ mod\ p(x) = x^4+x^3+x^2+1$	$\alpha^{17} = x^{17}\ mod\ p(x) = x^7+x^4+x^3$
$1 = 1\ mod\ p(x) = 1$	$\alpha^9 = x^9\ mod\ p(x) = x^5+x^4+x^3+x$	$\alpha^{18} = x^{18}\ mod\ p(x) = x^5+x^3+x^2+1$
$\alpha = x\ mod\ p(x) = x$	$\alpha^{10} = x^{10}\ mod\ p(x) = x^6+x^5+x^4+x^2$	$\alpha^{19} = x^{19}\ mod\ p(x) = x^6+x^4+x^3+x$
$\alpha^2 = x^2\ mod\ p(x) = x^2$	$\alpha^{11} = x^{11}\ mod\ p(x) = x^7+x^6+x^5+x^3$	$\alpha^{20} = x^{20}\ mod\ p(x) = x^7+x^5+x^4+x^2$
$\alpha^3 = x^3\ mod\ p(x) = x^3$	$\alpha^{12} = x^{12}\ mod\ p(x) = x^7+x^6+x^3+x^2+1$	$\alpha^{21} = x^{21}\ mod\ p(x) = x^6+x^5+x^4+x^2+1$
$\alpha^4 = x^4\ mod\ p(x) = x^4$	$\alpha^{13} = x^{13}\ mod\ p(x) = x^7+x^2+x+1$	$\alpha^{22} = x^{22}\ mod\ p(x) = x^7+x^6+x^5+x^3+x$
$\alpha^5 = x^5\ mod\ p(x) = x^5$	$\alpha^{14} = x^{14}\ mod\ p(x) = x^4+x+1$	$\alpha^{23} = x^{23}\ mod\ p(x) = x^7+x^6+x^3+1$
$\alpha^6 = x^6\ mod\ p(x) = x^6$	$\alpha^{15} = x^{15}\ mod\ p(x) = x^5+x^2+x$	$\alpha^{24} = x^{24}\ mod\ p(x) = x^7+x^3+x^2+x+1$
$\alpha^7 = x^7\ mod\ p(x) = x^7$	$\alpha^{16} = x^{16}\ mod\ p(x) = x^6+x^3+x^2$	$\alpha^{25} = x^{25}\ mod\ p(x) = x+1$

2.3. GENERATOR POLYNOMIAL

A R-S codeword is generated using this polynomial, it is defined by the size of the parity-bytes and composed of the minimal polynomial of the $GF(2^m)$. It is used for the encoding and decoding the message [12].
Once the size of the parity-bytes $(2t)$ is chosen, that is powers of $x \in GF(2^m)$, it is proceeded to compute all their minimal polynomials in order to produce the generator polynomial. For each consecutive power of α, the minimal polynomial will be $x - \alpha^i$ and the generator polynomial is constructed using the next expression:

$$g(x) = \prod_{i=FCR}^{FCR+2t-1} (x - \alpha^i) \tag{7}$$

Where FCR is the power of the first consecutive root in $g(x)$. The proposed value $2t = 32$ and the first consecutive root $FCR = 0$, therefore:

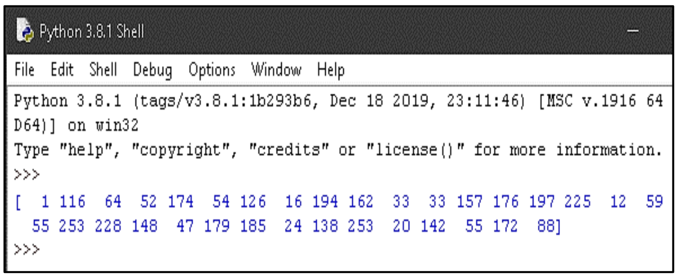
$$g(x) = \prod_{i=0}^{31}(x - \alpha^i) \tag{8}$$

$$g(x) = (x-1)(x-\alpha)(x-\alpha^2)(x-\alpha^3)\cdots(x-\alpha^{31}) \tag{9}$$

Using Galois algebra for a binary field we obtain our generator polynomial (Eq. 10):

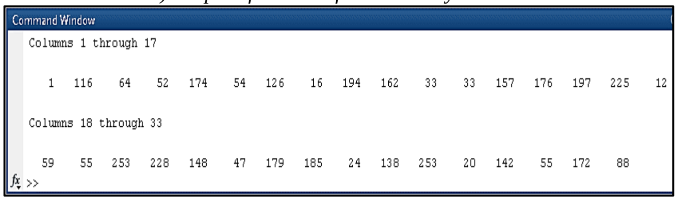
$$g(x) = x^{32} + \alpha^{10}x^{31} + \alpha^{6}x^{30} + \alpha^{106}x^{29} + \alpha^{190}x^{28} + \alpha^{249}x^{27} + \alpha^{167}x^{26} + \alpha^{4}x^{25} + \alpha^{67}x^{24} + \alpha^{209}x^{23} + \alpha^{138}x^{22} + \alpha^{138}x^{21} + \alpha^{32}x^{20} + \alpha^{242}x^{19} + \alpha^{123}x^{18} + \alpha^{89}x^{17} + \alpha^{27}x^{16} + \alpha^{120}x^{15} + \alpha^{185}x^{14} + \alpha^{80}x^{13} + \alpha^{156}x^{12} + \alpha^{38}x^{11} + \alpha^{69}x^{10} + \alpha^{171}x^{9} + \alpha^{60}x^{8} + \alpha^{28}x^{7} + \alpha^{222}x^{6} + \alpha^{80}x^{5} + \alpha^{52}x^{4} + \alpha^{254}x^{3} + \alpha^{185}x^{2} + \alpha^{220}x + 241 \qquad (10)$$

In the fig. 4, it is shown the results of our generator polynomial function in the communication software.



```
Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 23:11:46) [MSC v.1916 64
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
[  1 116  64  52 174  54 126  16 194 162  33  33 157 176 197 225  12  59
  55 253 228 148  47 179 185  24 138 253  20 142  55 172  88]
>>>
```

a) Output of our own function in Python



```
Columns 1 through 17

   1   116    64    52   174    54   126    16   194   162    33    33   157   176   197   225    12

Columns 18 through 33

  59    55   253   228   148    47   179   185    24   138   253    20   142    55   172    88
fx >>
```

b) Output using the "rsgenpoly" function in MATLAB

Fig 4. *Test of our generator polynomial function where we obtain the coefficients*

2.4. R-S ENCODING

Now with the generator polynomial, the parity-check polynomial can be defined by using the next expression:

$$parity(x) = \left(x^{n-k}\, M(x)\right) \;\; mod \;\; g(x) \qquad (11)$$

And then we can define the codeword as:

$$c(x) = x^{n-k}\, m(x) + parity(x) \qquad (12)$$

2.5. R-S DECODING

We define the received data as $r(x)$ and is the sum of the original codeword $c(x)$ plus the errors/erasures $E(x)$

$$r(x) = c(x) + E(x) \qquad (13)$$

The decoding process takes a quite large process that the R-S encoding:

1. Calculate the syndrome components from the received word
2. Calculate the error-locator word from the syndrome components
3. Calculate the error values from the syndrome components and the error-locator numbers
4. Calculate the decoded code word from the received word

2.6. SYNDROME CALCULATIONS

Every codeword is created so that it can be divided by the generator polynomial $g(x)$, therefore, if we evaluate a word in any of the roots of $g(x)$, we will obtain 0 (no errors) as result only if it is a codeword.

A R-S codeword has $2t$ syndromes that depend only on errors (NOT on the transmitted codeword). We can get the syndromes by evaluating the word in the roots of the generator polynomial $g(x)$ are the roots of the generator.

The i-th syndrome is defined as:

$$S_i = c(\alpha^i) + E(\alpha^i) = 0 + E(\alpha^i)$$

$$S_i = \sum_{j=1}^{v} E_{i_j}(\alpha^i) \qquad (14)$$

Let's define the error locators as $X_j = \alpha^{i_j}$ and the error values as $Y_j = E_{i_j}$, then the syndromes can be written in terms of these error locators and error values as:

$$S_i = \sum_{j=1}^{v} Y_j X_j^i$$

$$= Y_1 X_1^i + Y_2 X_2^i + \cdots Y_v X_v^i \qquad (15)$$

The syndromes give a system of $n - k \geq 2v$ equations in $2v$ unknowns.

$$\begin{bmatrix} S_1 \\ S_2 \\ \vdots \\ S_{2t} \end{bmatrix} = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_v \end{bmatrix} \begin{bmatrix} X_1^1 & X_2^1 & \dots & X_v^1 \\ X_1^2 & X_2^2 & \dots & X_v^2 \\ \vdots & \vdots & \ddots & \vdots \\ X_1^{2t} & X_2^{2t} & \dots & X_v^{2t} \end{bmatrix} \qquad (16)$$

If every syndrome is zero, then we can be sure that the codeword has no errors, by the other hand, if just a syndrome is different from zero, the codeword does not have one of the roots, therefore it cannot be divided by $g(x)$. If that case happens, it is necessary to detect the errors.

2.7. ERROR LOCATOR POLYNOMIAL

In order to solve the Eq. 16, it is precise to determine the Error-locator polynomial (Eq. 17), which has got $X_1^{-1}, X_1^{-1}, \cdots X_v^{-1}$ as roots.

$$\Lambda(x) = \prod_{j=1}^{v} (1 - xX_j) \qquad (17)$$

$$\Lambda(x) = 1 + \Lambda_1 x + \cdots + \Lambda_{v-1}x^{v-1} + \Lambda_v x^v \qquad (18)$$

$$\Lambda(X_j^{-1})Y_j X_j^{i+v} = 0 \qquad (19)$$

For all the errors (different values of k):

$$\sum_{j=1}^{v} Y_j X_j^{i+v} + \Lambda_1 \sum_{j=1}^{v} Y_j X_j^{i+v-1} + \cdots + \Lambda_v \sum_{j=1}^{v} Y_j X_j^{i} = 0 \qquad (20)$$

Substitute the Eq. 15 in Eq. 20 and we get the next linear system

$$S_{i+v} + \Lambda_1 S_{i+v-1} + \cdots + \Lambda_v S_i = 0 \qquad (21)$$

To solve such system, there are several algorithms to do so, the most popular are the Berlekamp-Massey algorithm [10][11] and the Peterson–Gorenstein–Zierler algorithm [15]; and to find the roots Λ(x) of the error locator polynomial it can be used the Chien search algorithm [16].

2.8. ERROR CORRECTION

Once the error locations X_k are known, the next step is to try to estimate the original value of Y_k. This can be achieved by using the Forney algorithm [11][13].

$$E_{i_j} = -X_j \frac{\Omega(X_j^{-1})}{\Lambda'(X_j^{-1})} \qquad (22)$$

The error evaluator polynomial is computed as follow:

$$\Omega(x) \ mod \ x^{2t+1} = \Lambda(x)S(x) \qquad (23)$$

And polynomial $\Lambda'(x)$

$$\Lambda'(x) = \sum_{j}^{t} \Lambda_j \, x^{j-1} \qquad (24)$$

The error values are then used to correct the received values at those locations to recover the original codeword.

3. PERFORMANCE METRICS OF DIGITAL COMMUNICATIONS

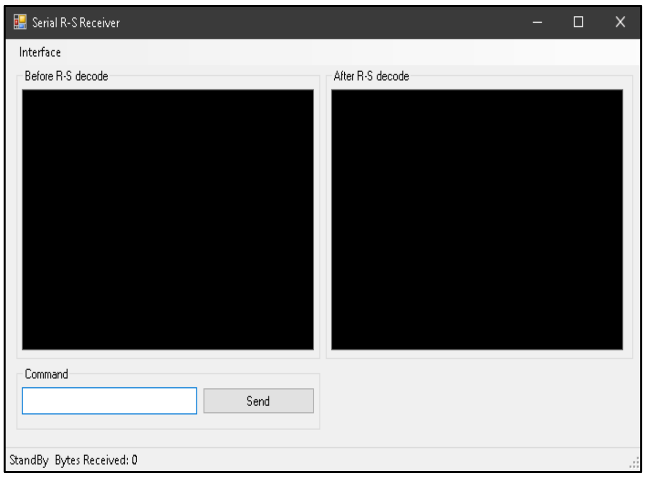
We can almost say for sure than the main goal of the digital communications systems is the accurate transfer of data bits as fast as possible [19]. The primary measure we are going to use to evaluate the R-S performance is the *Bit Error Rate* or BER.

The BER can be defined as the estimated probability of bit errors, in practical testing, it is measured by transferring a finite number of bits through the system and count the incorrectly received bits by comparing with the original data (Eq. 25) [20].

$$BER = \frac{Number\ of\ bit\ errors\ in\ package}{Total\ of\ bits\ in\ package} \qquad (25)$$

4. PROGRAM TESTING

The R-S codec has been implemented in two programming languages: Python for the on-board software running in the mini-PC and in C# for a laptop with Windows OS (Fig. 5).



StandBy Bytes Received: 0

Fig 5. *Receiver Control Software*

The transceiver (Fig. 6) work in the band of 433MHz, with a bandwidth of 1KHz, the modulation is GFSK. The majority of the COTS transceivers have a low bit-transfer rate (bps) as well as a low size of buffer-I/O of no more than a hundred of bytes, so to avoid an overrun of the buffer, it is necessary to send every package and wait an interval of time, depending of the transceiver model, to finish the sending [2][18].

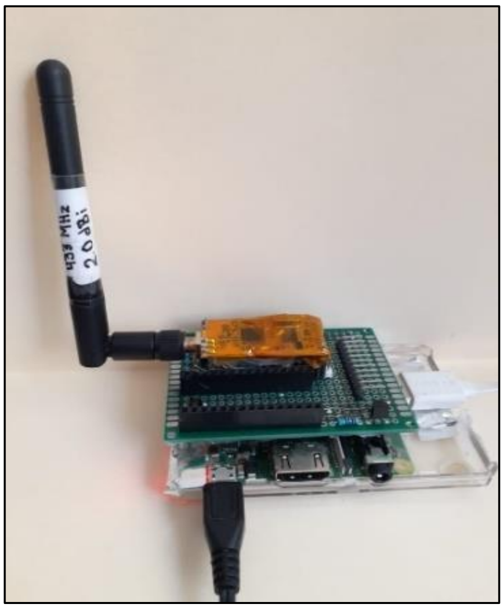


Fig 6. *Raspberry Pi with the transceiver*

The JPEG format is selected as a target file (Fig. 7) due to the Huffman encoding [2][17], even a single erroneous byte set in the wrong place can compromise the entire file, corrupting it. The image is a picture of the Large Millimeter Telescope "Alfonso Serrano" located at the Volcán Sierra Negra.

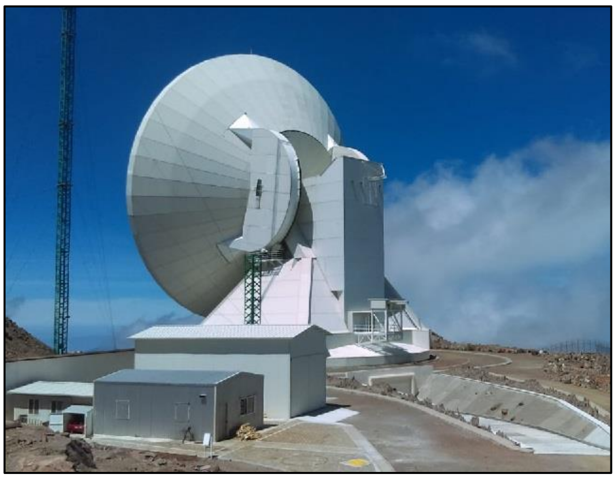
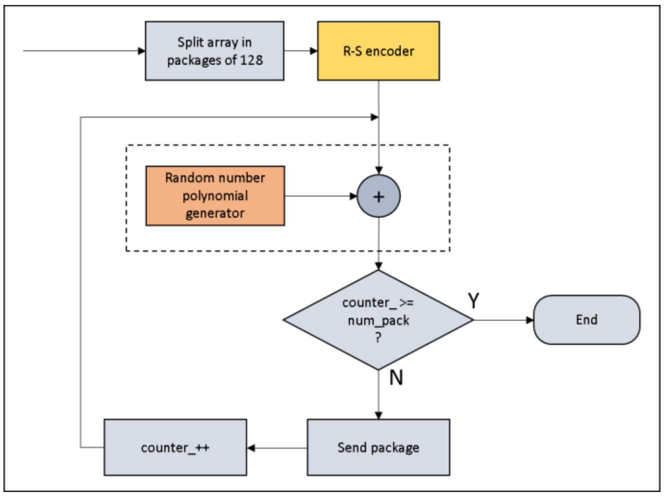


Fig 7. *Original image target for the transmission tests*

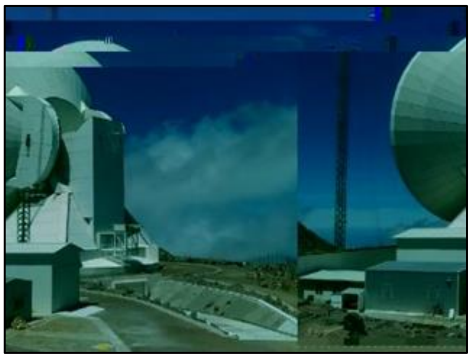
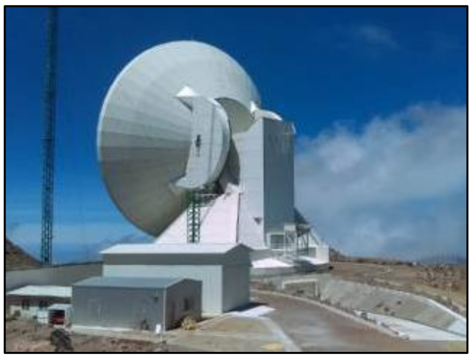
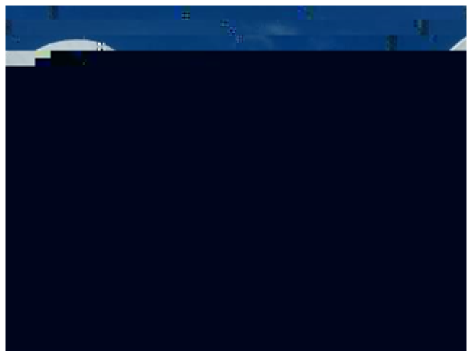
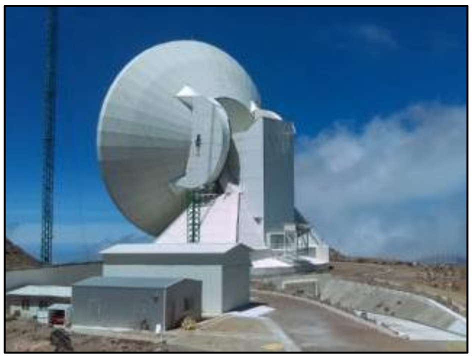
As mentioned before, during the transmission, the package passes through an error burst generator block (Fig. 8), where a certain random number function generates a burst of errors to be inserted in the data, the quantity of errors is determined by us manually during every test. For the AWGN test, the block generates a white-noise signal array where we only determine the standard deviation and the mean of the signal noise, so it is expected than the number of errors be above of 2t in some cases, meaning that some images will not be fully recovered.


Split array in packages of 128
R-S encoder
Random number polynomial generator
+
counter_ >= num_pack ?
Y
End
N
counter_++
Send package
Fig 8. Flowchart of the Random number generator is placed in the program for testing the R-S decoder

5. EXPERIMENTAL RESULTS

The algorithm has been able to receive the image data, decode every package and concatenate it to restore the original file as it is shown in the table 3. As a comparison, it is also shown the image file as it would be before the R-S decoding.

TABLE 3. IMAGE FILE RECEIVED THROUGH RADIO DURING THE BURST ERROR TEST.

Format: JPEG; Size (bytes): 19,421	
Error inserted	
10 bytes	 **Before R-S decoding**
	 **After R-S decoding**
700 bytes	 **Before R-S decoding**
	 **After R-S decoding**

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 02, February-2021

The AWGN test results are shown in the table 4 and 5, where it is observed than, since we cannot set the quantity of errors to be below 2t, the file is partially recovered but legible.

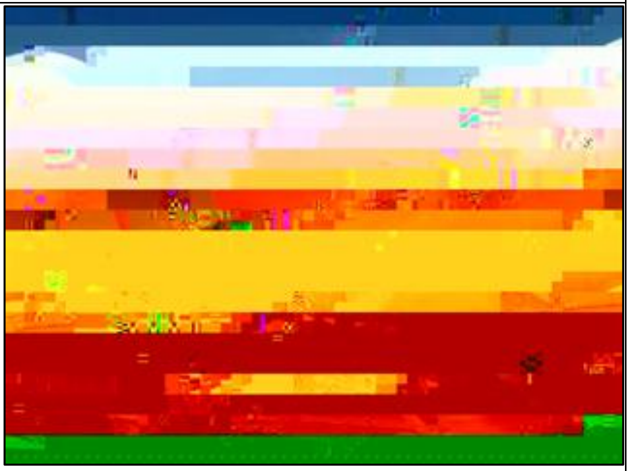
To measure the performance of the data transmission, we calculate the BER before and then after the R-S decoding. This is achieved by comparing the original data sent, before adding the AWGN, with the data decoded in the Receiver Control Program (Table 4). This process was done several times using different image data sizes.

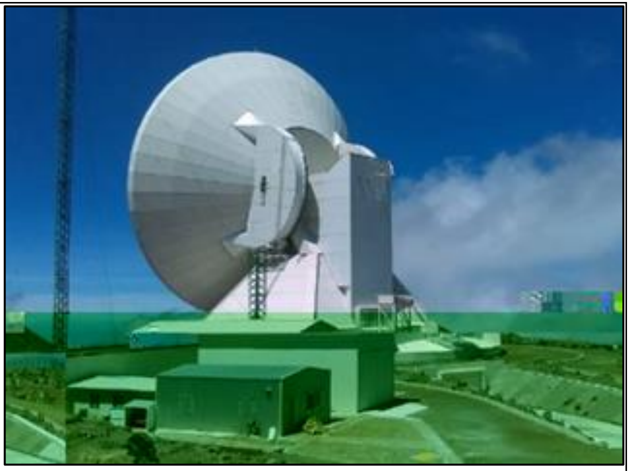
TABLE 4. BER CALCULATIONS USING THE DATA RECEIVED AND DECODED

No. test	Total bits	Total error bits	Error bits unable to correct	BER before R-S decoding	BER after R-S decoding
1	32,552	1,080	40	3.3×10^{-2}	1.2×10^{-3}
2	63,000	2,280	35	3.6×10^{-2}	5.6×10^{-4}
3	83,568	3,082	47	3.7×10^{-2}	5.6×10^{-4}
4	155,368	6,215	105	4×10^{-2}	6.8×10^{-4}

TABLE 5. IMAGE FILE RECEIVED DURING THE AWGN TEST

Format: JPEG; Size (bytes): 19,421


Before R-S decoding


After R-S decoding

As we can observe in the table 4, the BER value decreased, meaning an improvement in the transmission. For example, in the test No. 4, the BER before the R-S decoding is 4×10^{-2}, that is for every 100 bits, there are 4 erroneous bits. After the R-S decoding the BER goes down to 6.8×10^{-4}, meaning that for every 10,000 bits, there is just around 7 erroneous bits.

6. CONCLUSIONS

We have been able to implement a working R-S codec that can be used in ARM-based OBC as well as ported to Windows OS. We testes their performance using entirely low-cost COTS transceivers and determined that it is capable to find and correct errors in data files of different sizes. To improve the performance of the R-S codec, it is required further analysis of the time execution.

REFERENCES

[1] Lovascio, A., D'Orazio, A. and Centonze, V., 2020. Characterization of a COTS-Based RF Receiver for Cubesat Applications. Sensors, 20(3), p.776.

[2] Valadez-Campos, E., Mendoza-Torres, E. and DeRoa-Campoy, A., 2021. Low-cost testing platform for high-altitude balloon flights. Educative Clues (Pistas Educativas), 42(136), p.21. Available at: <http://www.itcelaya.edu.mx/ojs/index.php/pistas/article/view/2425/0> [Accessed 16 February 2021].

[3] Langer, Martin & Bouwmeester, Jasper., 2016. Reliability of CubeSats – Statistical Data, Developers' Beliefs and the Way Forward. Proceedings of the AIAA/USU Conference on Small Satellites, SSC16-X-2.

[4] Sklar, B. and Harris, F., 2004. The ABCs of linear block codes. IEEE Signal Processing Magazine, 21(4), pp.14-35.

[5] De Milliano, M. and Verhoeven, C., 2010. Towards the next generation of nanosatellite communication systems. Astronautic Act (Acta Astronautica), 66(9-10), pp.1425-1433.

[6] Phat Nguyen Huu and Vinh Tran-Quang†, Takumi Miyoshi., 2012. Multi-hop Reed-Solomon encoding scheme for image transmission on wireless sensor networks. Fourth International Conference on Communications and Electronics (ICCE).

[7] Sanjana P. Choudhari, Megha B. Chakole., 2017, Reed Solomon code for WiMAX network, 2017 International Conference on Communication and Signal Processing (ICCSP), pp. 176−179.

[8] N. Ambramson., Information Theory and Coding, 1963, McGraw-Hill, USA.

[9] E.R. Berlekamp Algebraic Coding Theory, 1968, McGraw-Hill, New York (USA).

[10] R.T. Moenck. Practical fast polynomial multiplication. In Proc. 3rd ACM Symposium on Symbolic and algebraic computation, pages 136–148. ACM, 1976.
[11] Blahut R. E., 2003, Algebraic Codes for Data Transmission, Cambridge University Press.
[12] Hamming, R., 1986. *Coding and information theory*. Englewood Cliffs, N.J.: Prentice-Hall.
[13] Forney, G., 1965. On decoding BCH codes. *IEEE Transactions on Information Theory*, 11(4), pp.549-557.
[14] Tilavat, V., & Shukla, Y. (2014). Simplification of procedure for decoding Reed–Solomon codes using various algorithms: an introductory survey. International Journal of Engineering Development and Research, 2(1), 279-283.
[15] Peterson, W., 1960. Encoding and error-correction procedures for the Bose-Chaudhuri codes. IEEE Transactions on Information Theory, 6(4), pp.459-470.
[16] Chien, R., 1964. Cyclic decoding procedures for Bose- Chaudhuri-Hocquenghem codes. *IEEE Transactions on Information Theory*, 10(4), pp.357-363.
[17] H. T. Sencar and N. Memon, 2009. Identification and recovery of JPEG files with missing fragments, Digital Investigation, vol. 6, pp. S88–S98
[18] Freebsd.org. 2020. Serial and UART Tutorial. [online] Available at: <https://www.freebsd.org/doc/en_US.ISO8859-1/articles/serial-uart/index.html> [Accessed 18 February 2021].
[19] Mitic, D., Lebl, A. and Markov, Z., 2012. Calculating the required number of bits in the function of confidence level and error probability estimation. Serbian Journal of Electrical Engineering, 9(3), pp.361-375.
[20] Sklar, B., 2001. Digital communications. Upper Saddle River, N.J.: Prentice Hall PTR.