Prosiding Seminar Nasional Sains dan Teknologi Seri 02
Fakultas Sains dan Teknologi, Universitas Terbuka
Vol. 1 No. 2 (2024)
e-ISSN: 3047-6569

IMPLEMENTASI PROSES *ENCODING* DAN *DECODING* KODE REED-MULLER MENGGUNAKAN PYTHON

Putri Arrum Nur Agustin*, Putranto Hadi Utomo
Program Studi Matematika, Universitas Sebelas Maret, Surakarta

**Penulis korespondensi: putriarrum@student.uns.ac.id*

ABSTRAK

Teori koding adalah salah satu ilmu pengetahuan yang mempelajari tentang metode pengiriman pesan melalui saluran komunikasi yang mengalami gangguan atau eror. Pada teori koding proses transmisi pesan dibagi menjadi dua yaitu *encoding* dan *decoding*. Proses *encoding* yaitu proses perubahan pesan menjadi sebuah *codeword*, dan proses *decoding* yaitu proses mengembalikan *codeword* yang telah di koreksi menjadi pesan asli yang dapat dibaca oleh penerima. Salah satu kode yang dapat digunakan untuk memperbaiki pesan yang eror adalah kode Reed-Muller. Kode Reed-Muller order pertama memiliki *rate* yang rendah, sehingga dimungkinkan untuk dapat memperbaiki banyak kesalahan. Metode penelitian yang digunakan dalam penelitian ini adalah studi literatur. Penelitian ini bertujuan menyusun algoritma pada pemrograman python untuk proses *encoding* dan *decoding* menggunakan kode Reed-Muller. Hasil yang diperoleh pada penelitian ini adalah algoritma *encoding* dan *decoding* menggunakan kode Reed-Muller. Diperoleh juga simulasi proses *encoding* dan *decoding* menggunakan kode Reed-Muller untuk kasus yang sederhana. Selain itu, juga disusun program untuk melakukan proses *encoding* dan *decoding* menggunakan kode Reed-Muller dengan bahasa pemrograman python.

Kata Kunci: *codeword*, *encoding*, *decoding*, kode reed-muller, transmisi pesan

1 PENDAHULUAN

Seiring berjalannya waktu teknologi akan terus berkembang menjadi semakin canggih. Contoh perkembangan teknologi dalam bidang komunikasi adalah semakin mudahnya kita mengirim pesan dan menerima pesan pada jarak jauh. Pesan yang dikirim akan melalui saluran pengiriman (*channel*) sebelum diterima oleh penerima. Akan tetapi, pesan yang diterima tidak akan selalu sesuai dengan pesan yang dikirimkan. Pesan terkadang mengalami gangguan atau eror (*noise*) sehingga akan mempengaruhi hasil pesan yang diterima. Fokus perhatian dalam transmisi pesan berisi informasi adalah integritas pesan, hal ini menjadi penting bagi *sender* untuk berkomunikasi dengan *receiver* supaya pesan yang ditransmisikan sama seperti pesan yang diterima.

Teori koding adalah salah satu ilmu pengetahuan yang mempelajari tentang metode pengiriman pesan melalui saluran komunikasi yang mengalami gangguan atau eror (*noise*). Proses yang terjadi pada teori koding dibagi menjadi dua. Proses pertama adalah *encoding*, yaitu proses perubahan pesan menjadi sebuah kode menggunakan *encoder* kemudian menjadi sebuah *input* dalam *channel*. Pada *channel* ini terdapat eror yang telah terjadi pada proses transmisi. Proses kedua adalah pesan yang sudah sampai di penerima akan di *decode* menjadi pesan asli dengan *decoder*.

Pada penelitian ini, akan dibahas bagaimana proses *encoding* dan *decoding* untuk kode Reed-Muller menggunakan bahasa pemrograman python untuk melihat hasil koreksi eror yang

terjadi. Untuk menggunakan kode Reed-Muller, perlu menganalisis beberapa parameter yang dibutuhkan lalu mencari generator matriksnya, selanjutnya akan dilakukan proses *encode* dan decode kode Reed-Muller. Alasan dipilihnya program python adalah bahasa python merupakan salah satu bahasa pemrograman yang mudah untuk dipelajari dan memiliki tingkat efesiensi tinggi pada struktur data serta memiliki fitur pembantu untuk program yang tinggi. Diharapkan dengan adanya program python pada kode Reed-Muller ini dapat memudahkan untuk memahami bagaimana kode Reed-Muller bekerja.

2 METODE

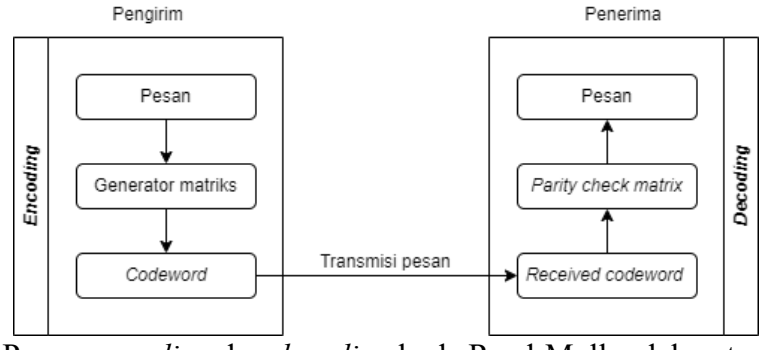
Metode penelitian yang digunakan dalam penelitian ini adalah studi literatur. Studi literatur yang dilakukan yaitu dengan mempelajari serta memahami konsep juga teori tentang *encoding* dan *decoding* menggunakan kode Reed-Muller dari beberapa referensi berupa buku, jurnal, tesis, maupun tulisan yang dimuat dalam situs web. Dari penelitian terdahulu, perkembangan yang terdapat dalam penelitian ini adalah pengimplementasian teori yang telah ada kedalam bahasa pemrograman python. Langkah-langkah yang akan dilakukan dalam penelitian ini adalah sebagai berikut.

1. Mempelajari definisi yang berkaitan dengan *encoding* dan *decoding* pada kode Reed-Muller.
2. Menetapkan pesan dalam bentuk biner yang hendak dikirimkan melalui *noisy channel*.
3. Melakukan simulasi proses *encoding* yaitu proses mengubah pesan yang ingin dikirim menjadi sebuah *codeword*. Setelah memeroleh *codeword*, pesan ditransmisikan.
4. Melakukan simulasi proses *decoding* yaitu proses mengembalikan *codeword* menjadi pesan yang benar sehingga dapat dibaca oleh penerimanya. Proses *decoding* terdiri dari *error-detection* yaitu mendeteksi apakah *codeword* yang telah ditransmisikan mengalami eror atau tidak, dan *error-correction* yaitu memperbaiki pesan yang eror agar kembali menjadi *codeword* yang asli. Setelah itu *codeword* dikembalikan menjadi pesan.
5. Mengimplementasi proses *encoding* dan *decoding* kode Reed-Muller menggunakan python.
6. Melakukan simulasi program yang telah dibuat.
7. Menarik kesimpulan.

3 HASIL DAN PEMBAHASAN

3.1 Algoritma *Encoding* dan *Decoding* Untuk Kode Reed-Muller

Pada proses transmisi pesan melalui *noisy channel*, terdapat *noise* atau kebisingan yang dapat menyebabkan pesan tersebut mengalami eror. Untuk mengatasi adanya gangguan atau eror pada saat proses transmisi pesan, diperlukan adanya algoritma *encoding* dan *decoding*. Skema proses *encoding* dan *decoding* menggunakan kode Reed-Muller dalam transmisi pesan dapat dilihat pada **Gambar 1**.



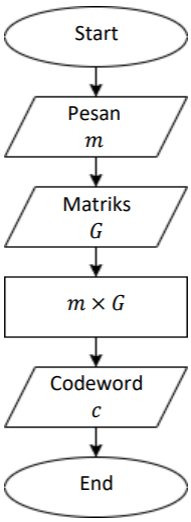
Gambar 1. Proses *encoding* dan *decoding* kode Reed-Muller dalam transmisi pesan

3.1.1 Algoritma *Encoding*

Langkah-langkah dalam proses *encoding* adalah sebagai berikut.

1. *Input* m, yaitu pesan dalam bentuk biner yang telah dijadikan matriks.
2. *Input* G, yaitu generator matriks dari kode yang akan digunakan.
3. Kalikan m dengan G untuk memperoleh *codeword*.
4. Didapatkan *output* c, yaitu *codeword* yang akan ditransmisikan melalui *noisy channel*.

Algoritma tersebut dapat digambarkan melalui *flowchart* pada **Gambar 2**.



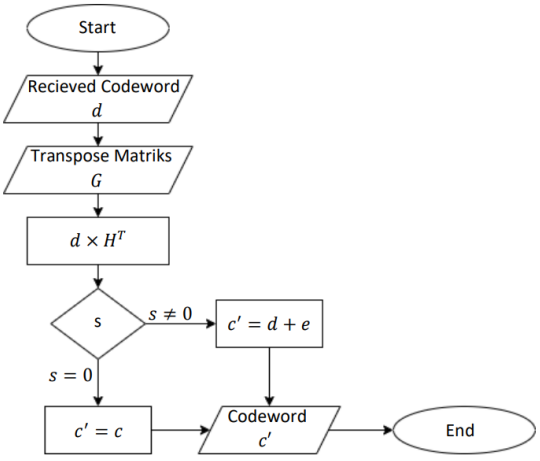
Gambar 2. Proses *Encoding* kode Reed-Muller

3.1.2 Algoritma *Decoding*

Langkah-langkah dalam proses *decoding* adalah sebagai berikut.

1. *Input* d, yaitu *codeword* yang telah di transmisikan melalui *noisy channel* (*received codeword*).
2. *Input* H^T , yaitu *parity check matrix* yang telah di-*transpose*.
3. Kalikan d dan H^T untuk memperoleh *syndrome* s.
4. Didapatkan *output* s.
5. Jika $s = 0$, maka d merupakan *codeword* (tidak terjadi eror).
6. Jika $s \neq 0$, maka d bukan merupakan *codeword* (terjadi eror).
7. Menentukan posisi eror e menggunakan matriks H.
8. Lakukan operasi XOR pada d dengan eror e.
9. Didapatkan *output* c' , yang harapannya merupakan *codeword* yang dikirim.

Algoritma tersebut dapat digambarkan melalui *flowchart* pada **Gambar 3**.



Gambar 3. Proses *Decoding* kode Reed-Muller

3.2 Contoh Proses *Encoding* dan *Decoding* Untuk Kode Reed-Muller Order Pertama $\boldsymbol{R(1,3)}$

Misalkan $m = (0 \quad 1 \quad 0 \quad 1)$ adalah sebuah pesan dalam bentuk biner yang hendak dikirimkan melalui *noisy channel*. Akan dilakukan proses *encoding* dan *decoding* menggunakan kode Reed-Muller order pertama $R(1,3)$, dengan generator matriks

$$G = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

dan *parity check matrix*

$$H = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

$$H^T = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix}.$$

Langkah-langkah untuk melakukan perhitungan adalah sebagai berikut.

1. Melakukan *encoding* untuk memeroleh *codeword* d melalui mengalikan pesan biner dengan generator matriks G.

$$c = mG$$

$$c = (0 \quad 1 \quad 0 \quad 1) \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

$$c = (0 \quad 1 \quad 0 \quad 1 \quad 1 \quad 0 \quad 1 \quad 0)$$

Diperoleh *codeword* $c = (0 \quad 1 \quad 0 \quad 1 \quad 1 \quad 0 \quad 1 \quad 0)$. Setelah memeroleh *codeword*, pesan akan ditransmisikan melalui *noisy channel*.

2. Selama proses transmisi melalui *noisy channel*, terdapat kemungkinan *codeword* mengalami eror. Oleh karena itu, dicari *syndrome* s untuk mengetahui apakah *codeword* yang telah di transmisikan mengalami eror atau tidak. *Syndrome* s diperoleh dari mengalikan pesan yang diterima d dengan *transposed parity check matrix* H^T . Jika $s = 0$, maka tidak terjadi eror, tetapi jika $s \neq 0$, maka terjadi eror. Lalu, menemukan posisi kolom pada H yang memuat s. Andaikan pada posisi ke-i, maka eror terjadi pada posisi ke-i. Melalui *noisy channel*, penerima meperoleh *received codeword* $d = (0 \quad 1 \quad 1 \quad 1 \quad 1 \quad 0 \quad 1 \quad 0)$. Akan dicari *syndrome* untuk mengetahui apalah terjadi eror atau tidak.

$$s = dH^T$$

$$s = (0 \quad 1 \quad 1 \quad 1 \quad 1 \quad 0 \quad 1 \quad 0)\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix}$$

$$s = (0 \quad 1 \quad 0 \quad 0)$$

Diperoleh *syndrome* $s = (0 \quad 1 \quad 0 \quad 0)$. Terlihat bahwa pada d terjadi eror. Pada H, s terletak pada kolom ketiga, hal ini berarti eror terletak pada bit ketiga.

3. Setelah posisi eror ditemukan, maka dilakukan *eror-correcting* untuk memperoleh *codeword* atau pesan yang corrected c' dengan menambahkan *codeword* yang diterima d dengan eror e. Melakukan eror correcting pada d.

$$c' = d + e$$

$$c' = (0 \quad 1 \quad 1 \quad 1 \quad 1 \quad 0 \quad 1 \quad 0) + (0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0)$$

$$c' = (0 \quad 1 \quad 0 \quad 1 \quad 1 \quad 0 \quad 1 \quad 0)$$

Diperoleh *codeword* $c' = (0 \quad 1 \quad 0 \quad 1 \quad 1 \quad 0 \quad 1 \quad 0)$.

4. Setelah didapatkan *codeword* c' , *codeword* tersebut harus dikembalikan ke dalam pesan asli agar dapat dibaca oleh penerima. Pengembalian *codeword* menjadi pesan asli dilakukan menggunakan sistem persamaan linear, dengan persamaan

$$(x_1 \quad x_2 \quad x_3 \quad x_4)G = c'$$

dengan x_1, x_2, x_3, dan x_4 merupakan elemen pesan yang jika disusun akan menghasilkan pesan asli, dan G merupakan generator matriks G.

$$(x_1 \quad x_2 \quad x_3 \quad x_4)G = c'$$

$$(x_1 \quad x_2 \quad x_3 \quad x_4)\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix} = (0 \quad 1 \quad 0 \quad 1 \quad 1 \quad 0 \quad 1 \quad 0)$$

$x_1 = 0$
$x_1 + x_2 = 1$
$x_1 + x_3 = 0$
$x_1 + x_2 + x_3 = 1$
$x_1 + x_4 = 1$
$x_1 + x_2 + x_4 = 0$
$x_1 + x_3 + x_4 = 1$
$x_1 + x_2 + x_3 + x_4 = 0$
$x_2 = 1$
$x_3 = 0$
$x_4 = 1$

Dari perhitungan sistem persamaan linear diatas dapat diperoleh pesan asli, yaitu $m = 0101$.

5. Didapatkan pesan asli $m = 0101$ yang dapat dibaca oleh penerima pesan.

3.3 Implementasi Proses *Encoding* dan *Decoding* Untuk Kode Reed-Muller Order Pertama $R(1,3)$ Menggunakan Program Python

Dalam penelitian ini simulasi proses *encoding* dan *decoding* kode Reed-Muller order pertama $R(1,3)$ akan diimplementasikan menggunakan program Python.

Tahapan simulasi proses *encoding* dan *decoding* kode Reed-Muller order pertama $R(1,3)$ adalah sebagai berikut.

1. Memasukan pesan berbentuk biner yang akan disimulasikan kedalam proses *encoding* dan *decoding*, seperti pada **Gambar 4**.

```
import numpy as np
'''INPUT PESAN'''
print('--INPUT PESAN--')
# Input Pesan yang akan di transmisikan
m = input("Masukan Pesan: ")
pesan = np.array([int(bit) for bit in m])
print('Pesan: ', pesan)
```

Gambar 4. Program proses *input* pesan

2. Setelah pesan dimasukan maka dimulailah proses *encoding* yaitu proses mengubah pesan menjadi *codeword* dengan mengalikan dengan generator matriks G. Proses *encoding* diawali dengan memasukan matriks G dan matriks H. Program proses *encoding* dapat dilihat pada **Gambar 5**.

```
'''PROSES ENCODING'''
print('--PROSES ENCODING--')

# Matriks G dan H
G = np.array([
    [1, 1, 1, 1, 1, 1, 1, 1],
    [0, 1, 0, 1, 0, 1, 0, 1],
    [0, 0, 1, 1, 0, 0, 1, 1],
    [0, 0, 0, 0, 1, 1, 1, 1]
])
H = np.array([
    [1, 0, 0, 1, 0, 1, 1, 0],
    [0, 1, 0, 1, 0, 1, 0, 1],
    [0, 0, 1, 1, 0, 0, 1, 1],
    [0, 0, 0, 0, 1, 1, 1, 1]
])

def encode_reed_muller(pesan):
    # Pastikan pesan memiliki panjang yang benar
    if len(pesan) != 4:
        raise ValueError("Panjang pesan harus 4 bit.")

    # Encoding pesan dengan mengalikan dengan matriks generator
    codeword = np.dot(pesan, G) % 2
    return codeword

# Codeword
codeword = encode_reed_muller(pesan)
print("Kata kode untuk pesan", pesan, ":", codeword)
```

Gambar 5. Program proses *encoding*

3. *Codeword* yang telah didapat dari proses *encoding* akan ditransmisikan melewati *noisy channel*. Diasumsikan transmisi pesan dengan menambahkan eror pada bit ketiga, seperti pada **Gambar 6**.

```
'''PROSES TRANSMISI PESAN'''
print('--PROSES TRANSMISI PESAN--')
error = [0,0,1,0,0,0,0,0]  # asumsi kesalahan di bit ketiga
codeword_error = (codeword + error) %2
print("Codeword yang telah tertransmisi:", codeword_error)
```

Gambar 6. Program proses transmisi pesan

4. Setelah pesan ditransmisikan melewati *noisy channel*, akan dilakukan proses *decoding* yang meliputi *error-detection* dan *error-correction* pesan, lalu kemudian *codeword* akan dikembalikan ke pesan asli. Program proses *decoding* dapat dilihat pada **Gambar 7**.

```
'''PROSES DECODING'''
print('--PROSES DECODING--')
def decode_reed_muller(codeword_error, HT):
    # Hitung syndrome (paritas dari pesan + kode)
    syndrome = np.dot(codeword_error, HT) % 2
    print('Syndrome: ',syndrome)
    # Periksa apakah ada kesalahan
    if np.all(syndrome == 0):
        print("Tidak terjadi error.")
        corrected_codeword = codeword_error  # Codeword asli
    else:
        # Temukan posisi kesalahan dari syndrome
        error_pos = -1
        for i in range(H.shape[1]):
            if np.array_equal(H[:, i], syndrome):
                error_pos = i
                break

        if error_pos == -1:
            print("Tidak dapat menemukan posisi error.")
            return codeword_error

        # Perbaiki kesalahan pada eror yang terdeteksi
        e = np.zeros_like(codeword_error)
        e[error_pos] = 1
        corrected_codeword = (codeword_error + e) % 2
    return(corrected_codeword)
HT = np.transpose(H)
corrected_codeword = decode_reed_muller(codeword_error, HT)
print("Codeword yang telah dikoreksi:", corrected_codeword)

# Periksa apakah corrected_codeword sama dengan codeword asli
def check_and_return(corrected_codeword, codeword, pesan):
    if corrected_codeword == codeword:
        return pesan
print("Pesan asli:", pesan)
```

Gambar 7. Program proses *decoding*

3.4 Hasil *Output* Dari Program Python

Pada simulasi transmisi pesan menggunakan kode Reed-Muller order pertama $R(1,3)$ dengan memasukan pesan 0101, diperoleh *output* yang dapat dilihat pada Gambar 8.

```
--INPUT PESAN--
Masukan Pesan: 0101
Pesan:  [0 1 0 1]
--PROSES ENCODING--
Codeword untuk pesan [0 1 0 1] : [0 1 0 1 1 0 1 0]
--PROSES TRANSMISI PESAN--
Codeword yang telah tertransmisi: [0 1 1 1 1 0 1 0]
--PROSES DECODING--
Syndrome:  [0 0 1 0]
Codeword yang telah dikoreksi: [0 1 0 1 1 0 1 0]
Pesan asli: [0 1 0 1]
```

Gambar 8. *Output* proses transmisi pesan

Dari *output* yang diperoleh, dapat dilihat bahwa program berhasil melakukan *encoding*, simulasi kesalahan, dan *decoding* dari pesan biner. Meskipun terdapat kesalahan dalam *codeword* yang diterima, algoritma kode Reed-Muller order pertama $R(1,3)$ dapat mendeteksi dan mengoreksi kesalahan tersebut, mengembalikan pesan asli dengan benar. Hasil ini menunjukkan bahwa program berfungsi dengan baik dalam mengatasi kesalahan dalam transmisi pesan biner.

4 KESIMPULAN

Proses transmisi pesan menggunakan kode Reed-Muller order pertama adalah meliputi proses *encoding* dan *decoding*. Proses tersebut dapat dilakukan dengan mencari generator matriks G terlebih dahulu untuk mendapatkan *codeword*, dan juga perlu mencari *parity check matrix* H untuk memperoleh *syndrome* dan mengetahui posisi eror. Proses *encoding* dan *decoding* menggunakan kode Reed-Muller order pertama dapat disimulasikan menggunakan bahasa pemrograman python, dan dapat diimplementasikan menggunakan software python. Pada penelitian selanjutnya diharapkan dapat meneliti terkait implementasi kode Reed-Muller selain order pertama menggunakan bahasa pemrograman Python.

UCAPAN TERIMA KASIH

Ucapan terima kasih disampaikan kepada semua pihak yang telah membantu dalam berjalannya penelitian dan penyusunan artikel ini. Semoga penelitian ini bermanfaat bagi semua pembaca.

DAFTAR PUSTAKA

Azizah, L. N. dan Syam, S. S. (2018). Hadamard Transform: Suatu Proses *Decoding* Pada Kode Reed Muller Orde Pertama, Universitas Negeri Yogyakarta, Indonesia.

Fayers, M. (2008). *MAS309 Coding Theory*, Queen Mary University of London, England.

Fraleigh, J.B. (2003). *A First Course In Abstract Algebra 7th Edition*, Addison-Wesley, Boston.

Gallian, J.A. (2006). *Contemporary Abstract Algebra 7th Edition*. Brooks/- Cole, USA.

Hirschfeld, J. W. P. (2014). *Coding Theory*, Mustansiriyah University, Iraq.

Jibril, M. (2011). *Algebraic Codes for Error Correction in Digital Communication Systems*, Doctor Degree Thesis, University of Plymouth, England.

Lindell, Y. (2010). *Introduction to Coding Theory Lecture Notes**, Bar-Ilan University, Israel.

Ling, S. and Xing, C. (2004). *Coding Theory A First Course*, Cambridge University New York, Inc., USA.

Meyer, L. (2021). *Coding and Decoding of Reed-Muller Codes*, Bachelor Degree Project.

Muhajir, F. (2016). Analisis Deteksi dan Koreksi Multi Bit Error dengan Partition Hamming Code.

Roth, R.M. (2006). *Introduction to Coding Theory*, IET Communications, 47(18-19), p.4.

Saptharishi, R., Shpilka, A. and Volk, B. L. (2016). *Efficiently decoding reedmuller codes from random errors*, Association for Computing Machinery.

Tilborg, H. C. A. (1993). *Coding Theory a first course*, Eindhoven University of Technology, Inc., Netherlands.

Udomkavanich, P. (2006). *2301532 : Coding Theory*, Chulalongkorn University, Thailand

Utomo, P. H. (2018). *Constrained Arrays and Erasure Decoding, Doctor Degree Thesis*, Eindhoven University of Technology, Netherlands.

Vanstone, S. A. and Oorschot P. C. (1989). *An Introduction to Error Correcting Codes with Applications*, Springer New York, USA.