Department of Electronic & Telecommunication Engineering
University of Moratuwa

Simplex Line Coding Transceiver

Group 04

Ayodya W.K.H	190065K
Bandara D.P.S.D	190070V
Bandara D.R.K.W.M.S.D	190071B
Bandara E.M.D.A	190072E

Supervisor: Mr. Mevan Wijewardena

This report is submitted as partial fulfilment of module EN2090 - Laboratory Practice - II

Date of Submission: March 2022

Table of Content

Page

Simplex Line Coding Transceiver

Group 4

Dept. of Electronic and Telecommunication Engineering, University of Moratuwa

ABSTRACT

Line coding is a technique which is used to transmit a bit stream using a channel. There are several globally recognized standard line-coding schemes available in the world. This paper focuses on the development and implementation of a simplex line coding transceiver based on analog components and IC's such as OpAmps, transistors and signal generating ICs. This project aims to accomplish data between two microcontrollers via a pair of twisted wires transmission with a data length of 1000 bits at a minimum data rate of 10 kb/s.

1. INTRODUCTION

The simplex line coding transceiver transmits specific length bits via a pair of twisted wires in one direction from one location to another. The transmitter and the receiver operate on the same frequency. This project aims to transmit a large bit length at a high bit rate with a low error rate. We have designed and built an encoding circuit and a decoding circuit that includes all analog devices. We chose the Manchester encoding system as our encoding technology.

2. METHODOLOGY

Manchester Encoding scheme has been implemented in our design using separate encoder and decoder circuits connected by a 50cm long twisted pair cable.

2.1. Block Diagrams

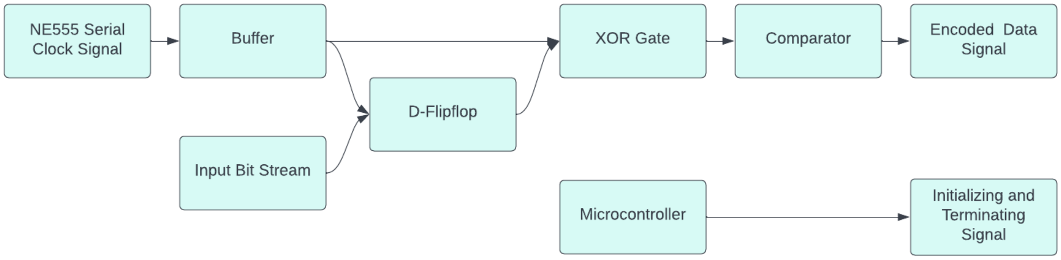


Figure 2.1.1: Encoder Block Diagram

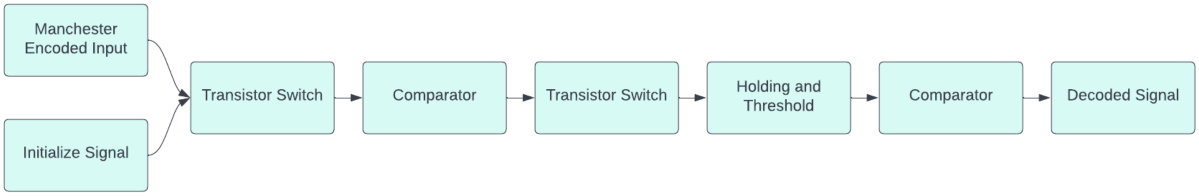


Figure 2.1.1: Decoder Block Diagram

2.2. Encoder Design

Encoder encodes digital data into a digital signal using Manchester encoding scheme. In Manchester encoding, ‘0’s are encoded as falling edges and ‘1’s are encoded as rising edges. The main advantage of using Manchester encoding is its self-clocking and synchronization on mid-bit transition. Manchester encoded data is achieved by sending digital data and clock through an XOR gate.

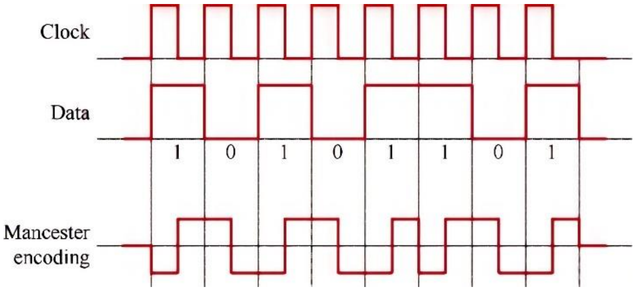


Figure 2.2: Manchester Encoding Scheme

2.2.1. Specifications

Baud Rate	: 10 kb/s
Current Drawn	: 46 mA
Operating Voltages	: 8 – 24 V
Impedance	: 92 kΩ

2.2.2. Circuit

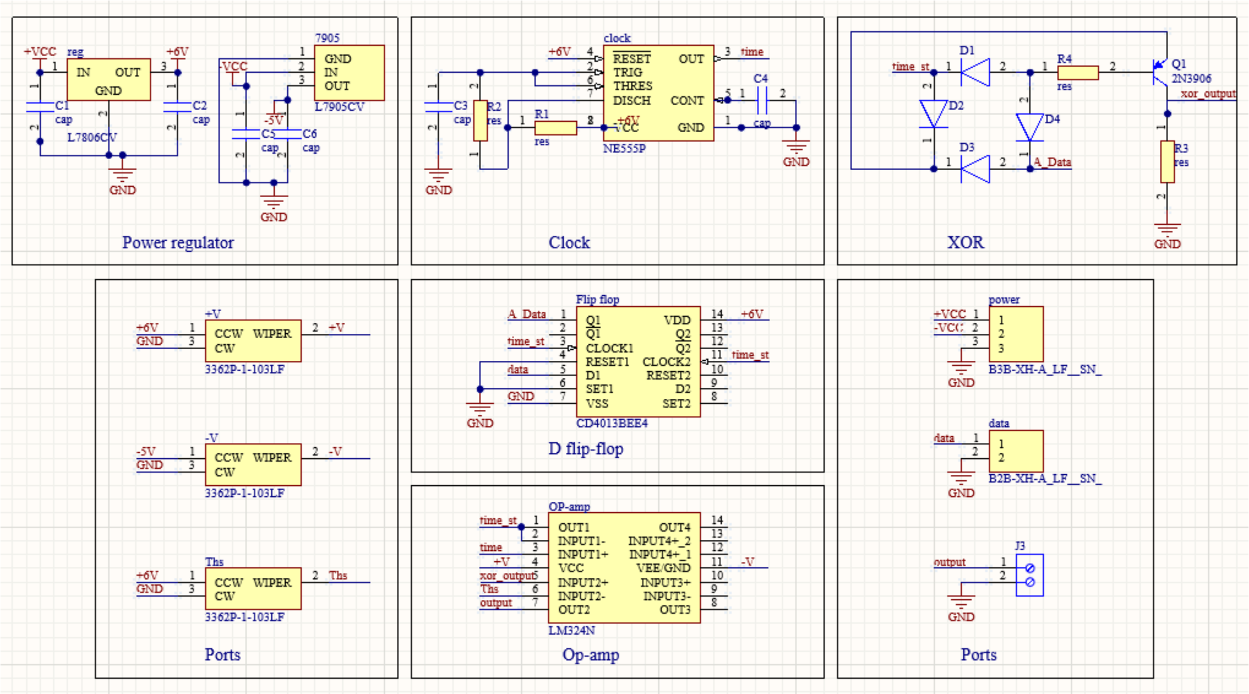


Figure 2.2.2: Encoder Schematic Design

2.2.3. Calculations

The following calculations are used to derive the clock frequency and the duty cycle of the NE555 timer.

Frequency of Oscillation,

R1 = 220 Ω
R2 = 5600 Ω
C1 = 10 nF

$$f = \frac{1}{T} = \frac{1 \cdot 44}{(R_1 + 2R_2)C_1} = \frac{1 \cdot 44}{(220 + 2 \times 5600) \times 10 \times 10^{-9}} = 12\ kHz$$

Duty Cycle,

$$\frac{t_1}{t_1 + t_2} = \frac{R_1 + R_2}{R_1 + 2R_2} = 1 - \frac{R_2}{R_1 + 2R_2} = 1 - \frac{5600}{220 + 2 \times 5600} = 50\%$$

2.2.4. Simulation

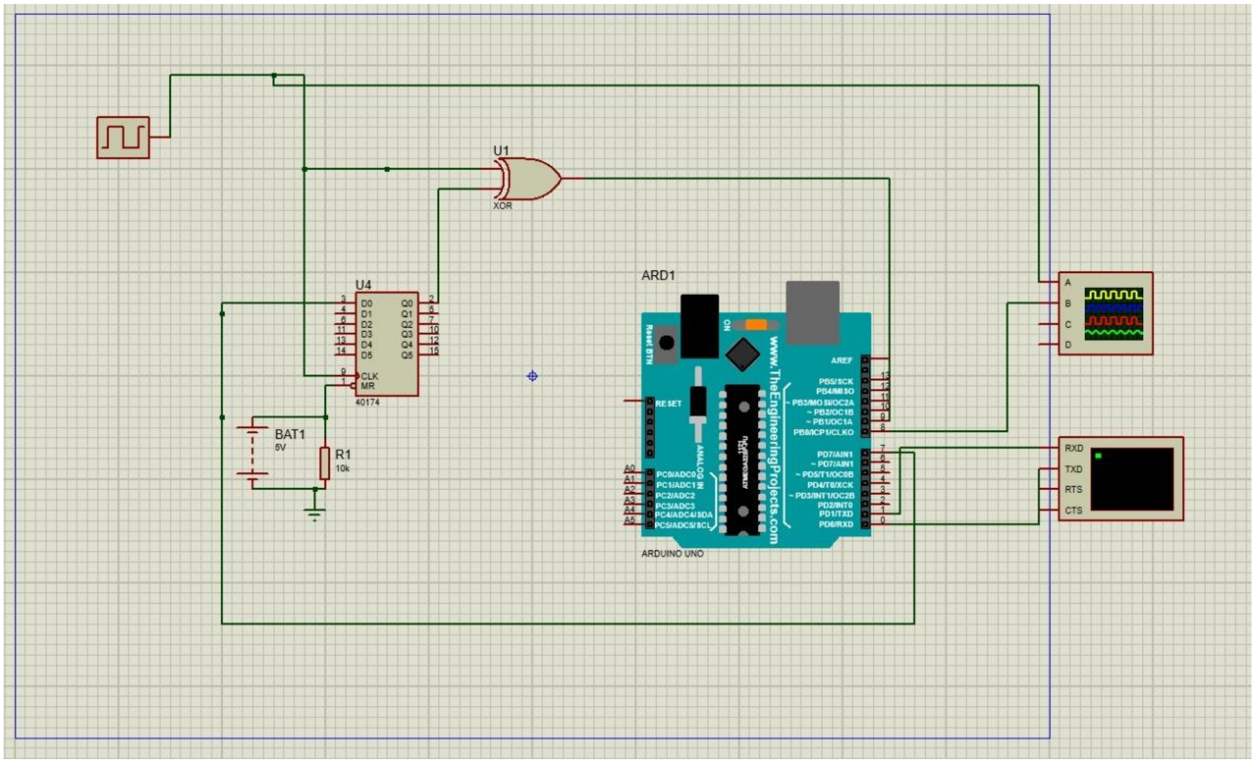


Figure 2.3.3: Encoder Simulation

2.2.5. PCB Design

Size of the PCB : 9cm x 5cm
Components mounted on Encoder PCB

- LM7905 Voltage Regulator – 1
- LM7806 Voltage Regulator – 1
- CD4013B D-Flipflop IC – 1
- NE555 Timer – 1
- LM324N OpAmp – 1
- BC556 PNP Transistor – 1
- Capacitors
  - 100 nF Polarized Cap – 2
  - 10 nF Ceramic Cap – 1
  - 100 nF Ceramic Cap – 1
  - 330 nF Polarized Cap – 2
- Resistors
  - 10 kΩ – 2
  - 5.6 kΩ – 1
  - 220 Ω – 1
- 1k Potentiometer – 3
- 1N4007 Diodes – 4
- 3 pin JST Connector – 1
- Screw Terminal – 1

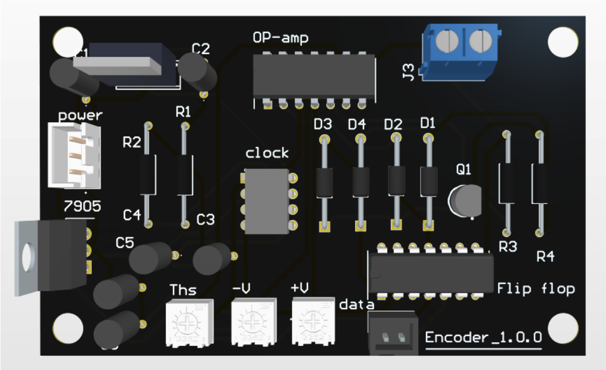


Figure 2.2.5.1: Encoder PCB 3D Model

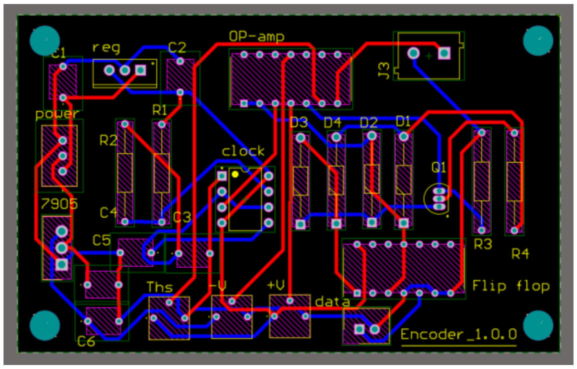


Figure 2.2.5.2: Encoder PCB Design

2.2.6. Enclosure Design

The enclosure is design with a glossy finish to get a futuristic appearance.

Material : PLA+ Plastics
Size of the Enclosure : 9cm x 5.6cm x 5.6cm

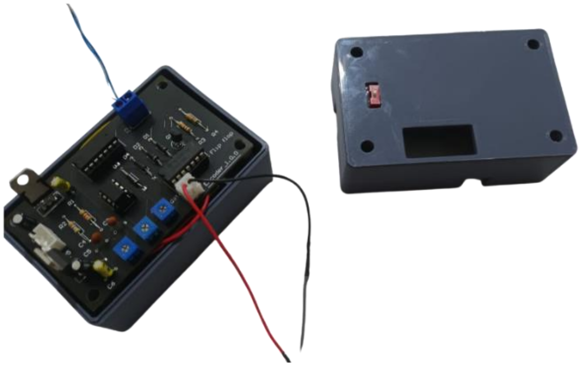


Figure 2.2.6: Encoder Enclosure Design

2.3. Decoder Design

The decoding circuit consist only analog components. When the encoded Manchester signal is received, the encoded data is decoded by the decoder. The decoder circuit operates at a frequency very similar to the encoder circuit.

2.3.1. Specifications

Baud Rate	: 10 kb/s
Current Drawn	: 17 mA
Operating Voltages	: 8 – 24 V
Impedance	: 92 kΩ

2.3.2. Circuit

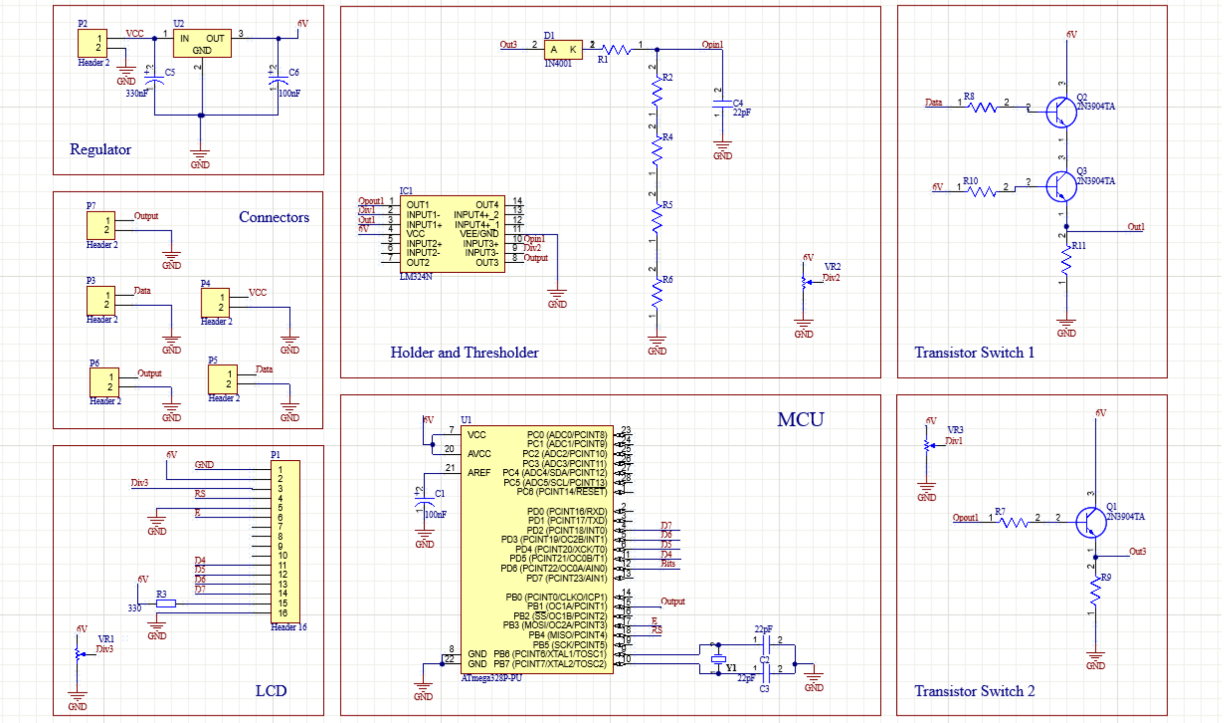


Figure 2.3.2: Decoder Schematic Design

2.3.3. Simulation

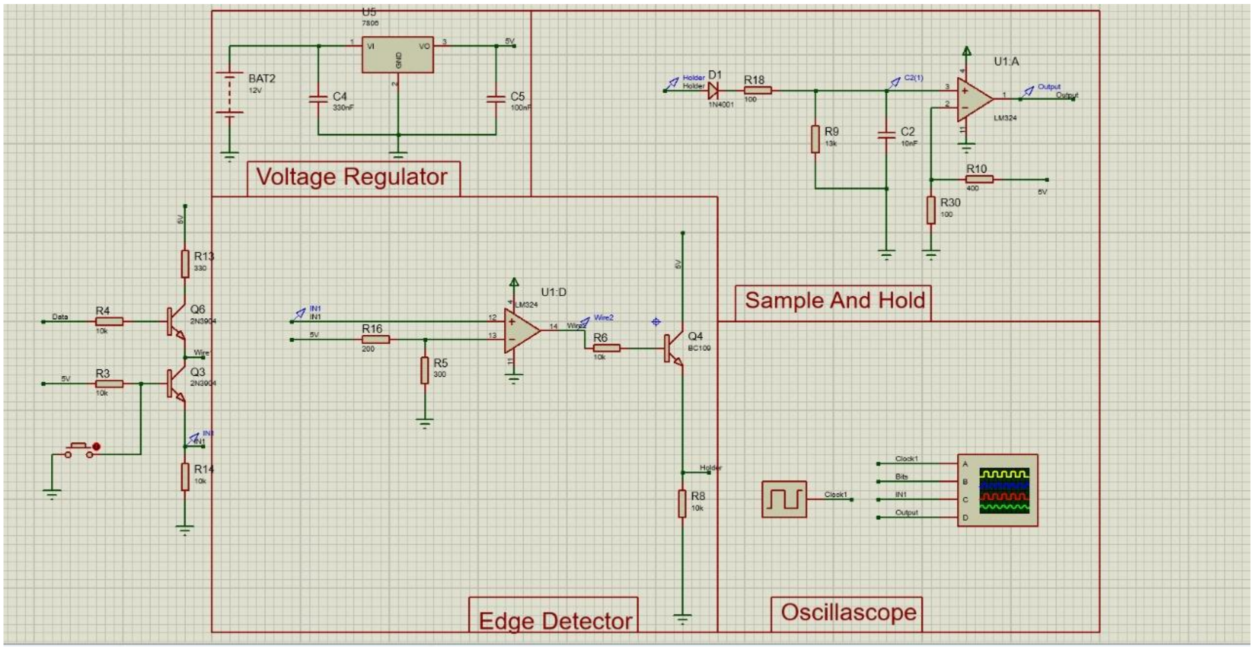


Figure 2.3.3: Decoder Simulation

2.3.4. PCB Design

Size of the PCB : 9cm x 5cm

Components mounted on Encoder PCB

• LM7806 Voltage Regulator	- 1
• LM324N OpAmp	- 1
• 2N3904TA NPN Transistor	- 1
• Capacitors	
○ 100nF Polarized cap	- 1
○ 330nF Polarized cap	- 1
○ 10nF ceramic Cap	- 1
• Resistors	
○ 100Ω	- 1
○ 10kΩ	- 6
○ 1KΩ	- 3
• 1k Potentiometers	- 2
• 1N4001 Diodes	- 1
• 2 pin JST Connectors	- 4

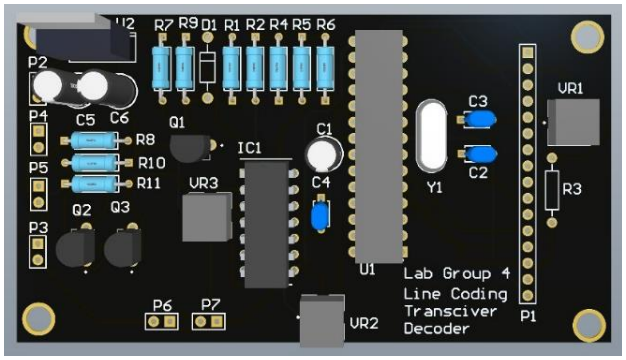


Figure 2.3.4.1: Decoder PCB 3D Model

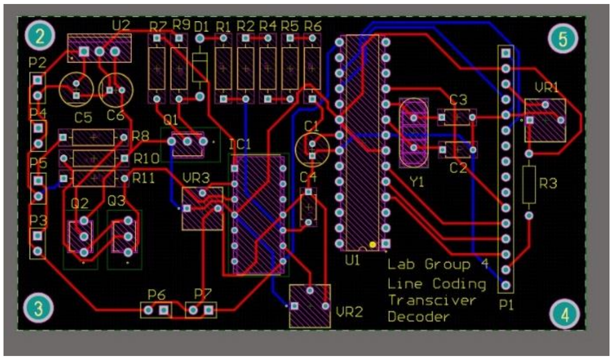


Figure 2.3.4.2: Decoder PCB Design

2.3.5. Enclosure Design

The enclosure is design with a glossy finish to get a futuristic appearance.

Material : PLA+ Plastics

Size of the Enclosure : 13cm x 9cm x 4.5cm

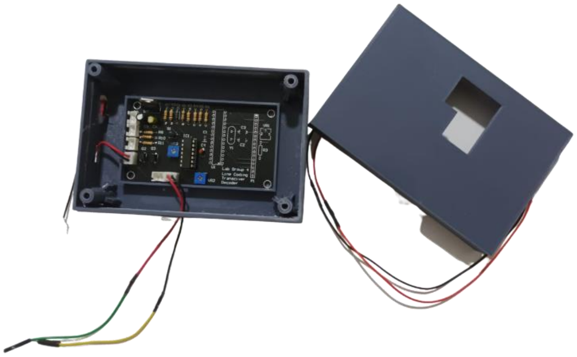
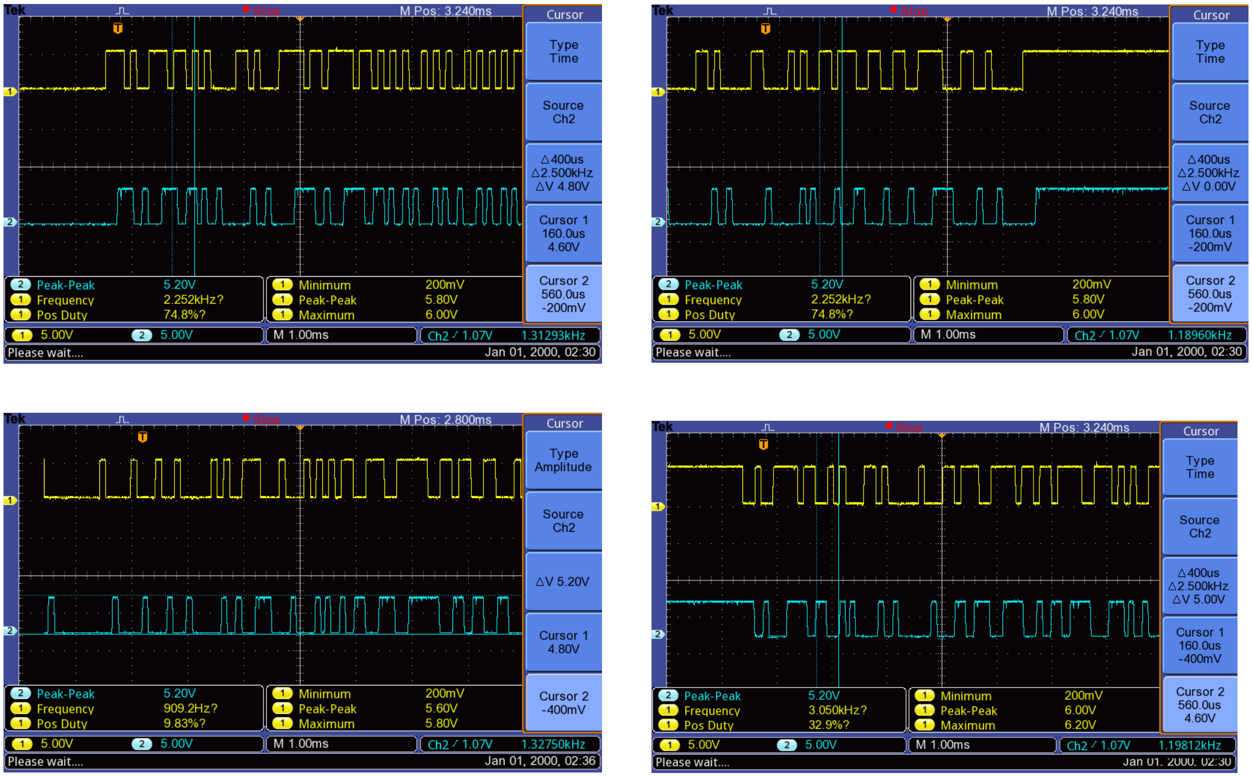


Figure 2.3.5: Decoder Enclosure Design

3. RESULTS

10kb/s input has been input to the simplex line code transceiver using an Arduino Uno microcontroller. The bit pattern consists of a repetitive bit pattern with length of 10bits. Yellow colour signal in the oscilloscope output indicates transmitting signal. The encoder has converted the binary bit pattern to a Manchester signal with peak values of +5V and -5V. The decoder has used pulse width identification mechanism to decode the signal. Blue color waveform in the oscilloscope indicates the output waveform. The delay between the encoded Manchester signal and the decoded output is around 10µs. There is no observable delay between the input signal and the Manchester encoded signal.



Tek Stop M Pos: 3.240ms
Cursor: Type Time; Source Ch2; Δ400us Δ2.500kHz ΔV 4.80V; Cursor 1 160.0us 4.60V; Cursor 2 560.0us -200mV
2 Peak-Peak 5.20V; 1 Frequency 2.252kHz?; 1 Pos Duty 74.8%?; 1 Minimum 200mV; 1 Peak-Peak 5.80V; 1 Maximum 6.00V
1 5.00V 2 5.00V M 1.00ms Ch2 ⁄ 1.07V 1.31293kHz
Please wait.... Jan 01, 2000, 02:30

Tek Stop M Pos: 3.240ms
Cursor: Type Time; Source Ch2; Δ400us Δ2.500kHz ΔV 0.00V; Cursor 1 160.0us -200mV; Cursor 2 560.0us -200mV
2 Peak-Peak 5.20V; 1 Frequency 2.252kHz?; 1 Pos Duty 74.8%?; 1 Minimum 200mV; 1 Peak-Peak 5.80V; 1 Maximum 6.00V
1 5.00V 2 5.00V M 1.00ms Ch2 ⁄ 1.07V 1.18960kHz
Please wait.... Jan 01, 2000, 02:30

Tek Stop M Pos: 2.800ms
Cursor: Type Amplitude; Source Ch2; ΔV 5.20V; Cursor 1 4.80V; Cursor 2 -400mV
2 Peak-Peak 5.20V; 1 Frequency 909.2Hz?; 1 Pos Duty 9.83%?; 1 Minimum 200mV; 1 Peak-Peak 5.60V; 1 Maximum 5.80V
1 5.00V 2 5.00V M 1.00ms Ch2 ⁄ 1.07V 1.32750kHz
Please wait.... Jan 01, 2000, 02:36

Tek Stop M Pos: 3.240ms
Cursor: Type Time; Source Ch2; Δ400us Δ2.500kHz ΔV 5.00V; Cursor 1 160.0us -400mV; Cursor 2 560.0us 4.60V
2 Peak-Peak 5.20V; 1 Frequency 3.050kHz?; 1 Pos Duty 32.9%?; 1 Minimum 200mV; 1 Peak-Peak 6.00V; 1 Maximum 6.20V
1 5.00V 2 5.00V M 1.00ms Ch2 ⁄ 1.07V 1.19812kHz
Please wait.... Jan 01. 2000. 02:30

Figure 3.1: Data Signal and Decoded Output Signal

4. DISCUSSION

We had to face a lot of challenges in designing and implementing the Simplex line coding transceiver.

Problems faced while carrying on the project

- Several iterations of tuning needed when connecting sub-sections of the circuit
- Some ICs and components bought from local vendors operated differently than the specifications in official datasheets
- Scarcity of some components has lead to difficulties when building the circuit

Solutions for these problems

- Planning input-output configurations and setting benchmarks for each subsection
- Pre-testing each component before adding to circuit to verify its behavior.
- Using readily available ICs and components for the design. Contact electronic equipment vendors and check the availability of components before adding them to circuit

5. ACKNOWLEDGEMENT

We would like to pay our gratitude to our supervisor Mr. Mevan Wijewardena who always encouraged self-studying and constantly guided us whenever we needed help. The meetings held by our supervisor were very helpful to clear our doubts and discuss issues regarding the project. In addition, we would like to convey our sincere gratitude to all the lecturers and instructors who were always willing to share their knowledge with us. It is our duty to thank our ENTC family, without whom we could not have accomplished this project. Everyone in our batch helped each other and learnt everything together.

We developed team spirit working together in this project for about 3 months and all our group members gave their maximum contribution to succeed the given project. Further, we would like to thank all the people who are not mentioned here for the great support provided.

6. REFERENCES/BIBLIOGRAPHY

[1] https://www.allaboutcircuits.com/technical-articles/how-to-generate-manchester-encoded-data-in-hardware-and-firmware/

[2] https://www.allaboutcircuits.com/technical-articles/how-to-decode-manchester-encoded-data-using-hardware/

[3] https://www.ti.com/lit/ds/symlink/lm555.pdf

7. CONTRIBUTION

190065K	Ayodya W.K.H	Decoder Circuit Design, Decoder Enclosure Design, PCB Soldering
190070V	Bandara D.P.S.D	Encoder Circuit Design, Encoder PCB Design, PCB Soldering
190071B	Bandara D.R.K.W.M.S.D	Decoder Circuit Design, Decoder PCB Design, PCB Soldering
190072E	Bandara E.M.D.A	Encoder Circuit Design, Encoder Enclosure Design, PCB Soldering

8. APPENDICES

GitHub Repository : https://github.com/Lab-Project-Group-Team-4/Line-coding-Transciver/tree/master

Enclosure CAD Designs :

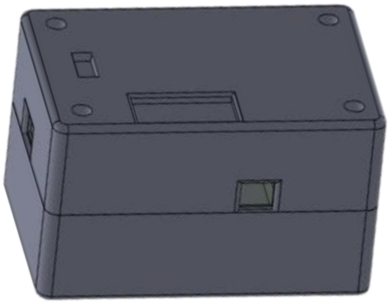


Figure 7.1: Encoder CAD Design

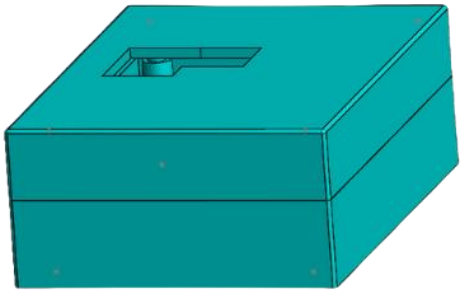


Figure 7.2: Decoder CAD Design

Encoder Code :

```
//including libraries for interrupt and delay
#include <avr/io.h>
#include <util/delay.h>
//delay amount in micro-seconds
#define delay_amount 70

//Data String Given
String Data = "1010101010";
int len = 10; //length of the String
int index = 0; //Current bit index in patter
int i = 0; //transmitted bit value

void setup() {
DDRB |= (1<<PB0); // Configuring Port B pin0 as an output

}

void loop() {
  if ( i < 1000 )
  {
    //toggle output based on the bit value
    if (Data[index%len] == '0'){
      PORTB &= ~(1<<PB0);
    }
    else{
      PORTB |= (1<<PB0);
    }
    index = (index + 1)%len;
    _delay_us(delay_amount);
  }

  i++;
}
```

Decoder Code :

```
#include <SPI.h>
#include <SD.h>
#define length_arr 300
File myFile;

int bit_array[length_arr] = {0};
int bit_pattern[10] = {0, 1, 0, 1, 0, 1, 0, 1, 0, 1};
int index = 0;


volatile bool start_read = false;
volatile bool read_val = false;
void blink_LED(int delay_amount);
float accuracy(int* encoded, int* decoded ,int len_message);

void setup() {



    DDRD &= ~(1<<PD7); // Set pin D7 as input
    DDRD |= (1<<PD6);  // Set pin D6 as output
    PORTD |= (1<<PD7); // Pull-Up the pin 7

    // TIMER 0 for interrupt frequency 10000 Hz:
    cli(); // stop interrupts
    TCCR0A = 0; // set entire TCCR0A register to 0
    TCCR0B = 0; // same for TCCR0B
    TCNT0  = 0; // initialize counter value to 0
    // set compare match register for 10000 Hz increments
    OCR0A = 199; // = 16000000 / (8 * 10000) - 1 (must be <256)
    // turn on CTC mode
    TCCR0B |= (1 << WGM01);
    // Set CS02, CS01 and CS00 bits for 8 prescaler
    TCCR0B |= (0 << CS02) | (1 << CS01) | (0 << CS00);
    // enable timer compare interrupt
    TIMSK0 |= (1 << OCIE0A);

  sei();

}

void loop() {


  if (read_val && index < length_arr) {
    bit_array[index] = ((PIND & ( 1 << PD7 ))) >> 7;
    read_val = false;
    index++;
  }

  else if ( index == length_arr) {
    Serial.begin(9600);
    for ( int i = 0; i < length_arr; i++) {
      Serial.print(bit_array[i]);
      Serial.print(";");
      if ( i%50 == 0){
        Serial.println();
        }
    }
    float accuracy_val;
    accuracy_val = accuracy(bit_pattern, bit_array, length_arr);
    Serial.println();
    Serial.end();
    index++;
  }

}

ISR(TIMER0_COMPA_vect) {
  read_val = true;
```

```
}

float accuracy(int* encoded, int* decoded ,int len_message){
 int correct = 0;
 int message_index = 0;
 for ( int i = 0; i<len_message ; i++){
  if (decoded[i] == encoded[message_index] ) {
    correct++;
   }

   message_index = (message_index + 1 ) % 10;
  }
 }
```