Industrial Networks Project

Submitted by :

Ahmed Sherif Hosny 17P8028

Mohanad Maher 17P8150

Omar Adel Ahmed 17P8168

Ahmed Adel Ahmed 17P9039

Submitted to:

Prof. Ayman Bahaa

جامعة عين شمس

AIN SHAMS UNIVERSITY

INTRODUCTION

The universal asynchronous receiver-transmitter (UART) takes bytes of data and transmits the individual bits in a sequential fashion. At the destination, a second UART re-assembles the bits into complete bytes. Each UART contains a shift register, which is the fundamental method of conversion between serial and parallel forms. Serial transmission of digital information (bits) through a single wire or other medium is less costly than parallel transmission through multiple wires.

In this project, we were required to :

- In this driver, on the sending side, a function will take an array of bytes as an input and produces an ETHERNET compatible frame encapsulating the data.
- On the receiver side, the receiving function should validate the frame and extracts the data back.

IMPLEMENTED MICRO-CONTROLLERS

ARDUINO UNO

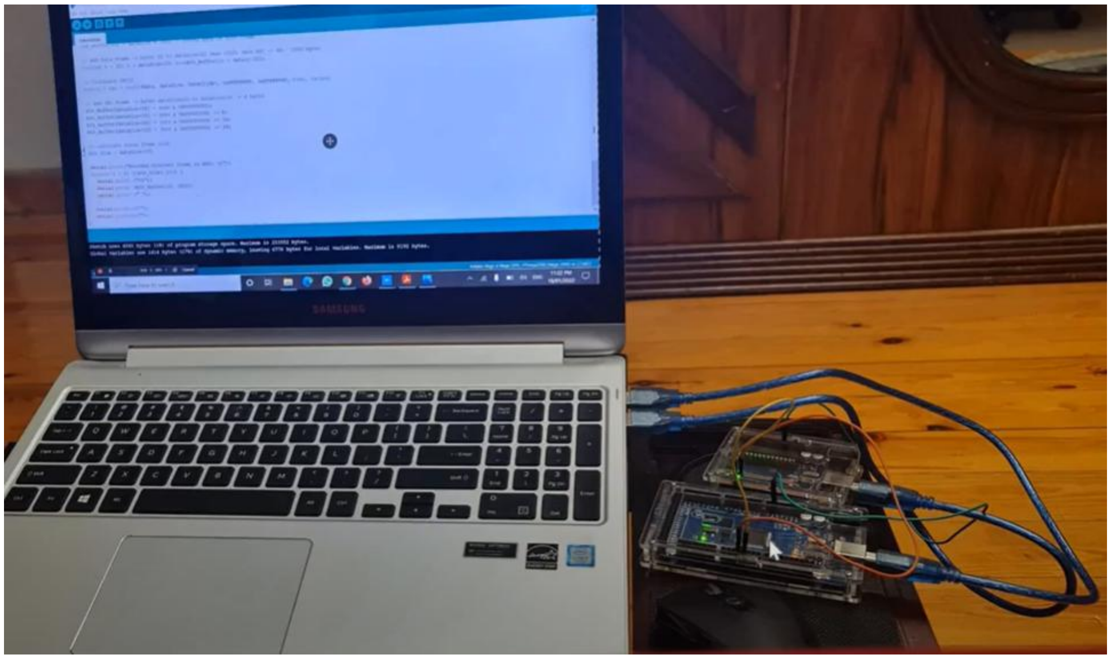
ARDUINO MEGA

THE CIRCUIT :

COMPONENTS:

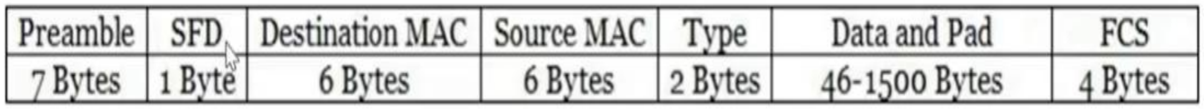
- Urduino UNO “Tx”
- Urduino Mega “RX”
- Wires “tx,rx,gnd”
- Laptop

CIRCUIT:



STEPS:

Before We start lets briefly describe the Ethernet frame



Preamble	SFD	Destination MAC	Source MAC	Type	Data and Pad	FCS
7 Bytes	1 Byte	6 Bytes	6 Bytes	2 Bytes	46-1500 Bytes	4 Bytes

- The data field is the field containing the real data that needs to be transmitted to the receiver side .
- A header is added to the data in the link layer containing:
  - Preamble : 7 bytes of 0x10101010
  - Start of frame : 1 byte
  - Src and destination MAC addresses : used to check if data is sent to current device or not and to know which device is communicating. A 6 byte hexadecimal address each.
  - Type: used to identify the type of the Network layer used usually IP
- A trailer is added to the data :
  - FCS: 4 bytes
  - CRC 32 : cyclic redundancy checksum to check that data is sent and received correctly

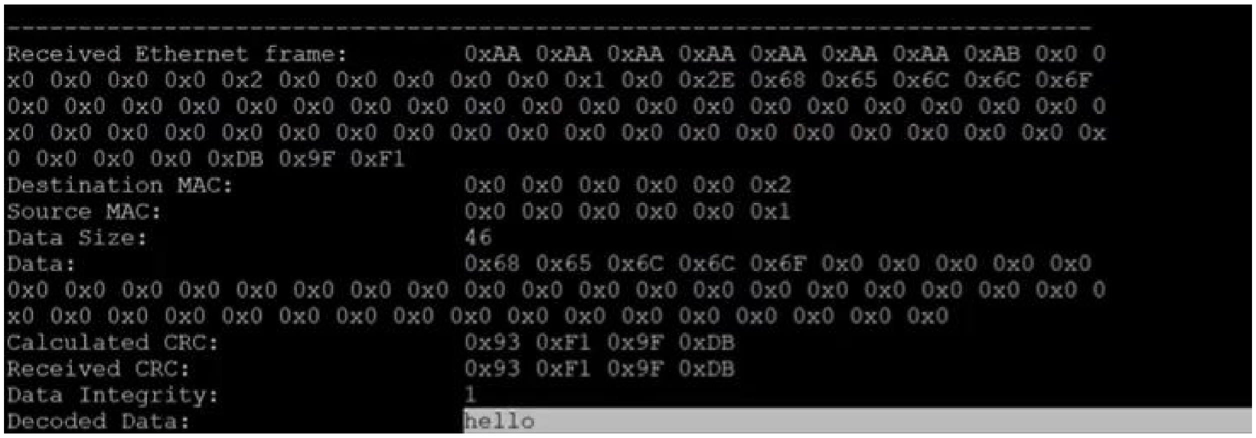
STEPS TO CREATE THIS FRAME FROM THE TRANSMITTER SIDE :

- We assumed max Ethernet frame size 512 bytes for space and memory limitations
- Created a buffer for storing input data by user
- Creation of unique Mac address for both transmitter and receiver
- Started serial communication between the two arduinos
- Get input from user and store it in buffer
- Encode the data and add the header and trailer as discussed :
    - Check size of data to add padding if less than minimum size .
    - Add preamble and start of frame
    - Add MAC addresses of destination then source
    - Calculate CRC
    - Add CRC to trailer
- Send data using serial communication

RECEIVER SIDE :

- Started a communication between putty and the receiver side
- Wait for Frame transmission (serial comm between two arduinos)
- Decode Ethernet frame as follow:
    - Check for destination address
    - Identify src address to be used if needed
    - Store data in internal buffer from it respective position in the frame
    - Calculate crc
    - Compare with crc sent with the frame
- If correct , print the incoming frame and its decoded contents.

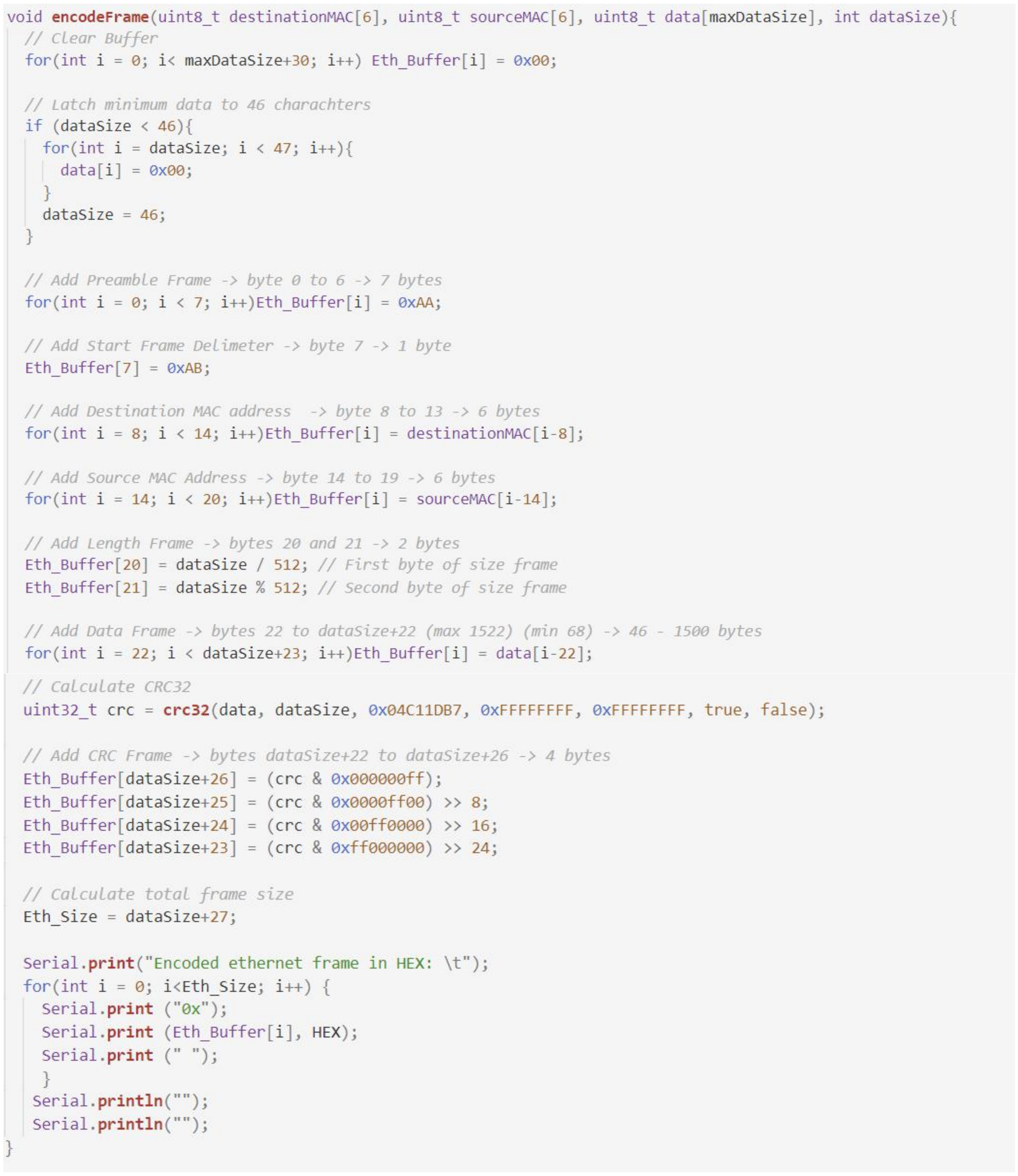
OUTPUT EXAMPLE:



```
---------------------------------------------------------------------------
Received Ethernet frame:        0xAA 0xAA 0xAA 0xAA 0xAA 0xAA 0xAA 0xAB 0x0 0
x0 0x0 0x0 0x0 0x2 0x0 0x0 0x0 0x0 0x0 0x1 0x0 0x2E 0x68 0x65 0x6C 0x6C 0x6F
0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0
x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x
0 0x0 0x0 0x0 0xDB 0x9F 0xF1
Destination MAC:                0x0 0x0 0x0 0x0 0x0 0x2
Source MAC:                     0x0 0x0 0x0 0x0 0x0 0x1
Data Size:                      46
Data:                           0x68 0x65 0x6C 0x6C 0x6F 0x0 0x0 0x0 0x0 0x0
0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0
x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0
Calculated CRC:                 0x93 0xF1 0x9F 0xDB
Received CRC:                   0x93 0xF1 0x9F 0xDB
Data Integrity:                 1
Decoded Data:                   hello
```

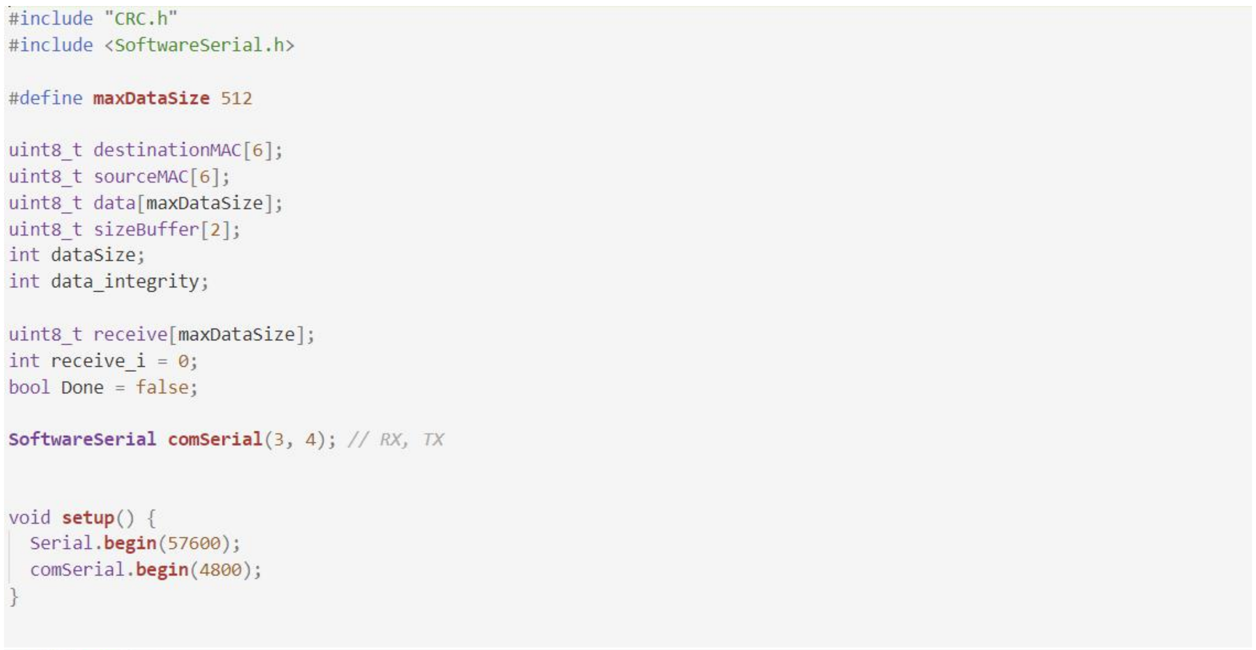
TRANSMITTER CODE

```
#include "CRC.h"
#include <SoftwareSerial.h>

#define maxDataSize 512

uint8_t Eth_Buffer[maxDataSize+30];
int Eth_Size = 0;

uint8_t source_MAC[6] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x01};
uint8_t destination_MAC[6] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x02};

uint8_t transmit[maxDataSize];
int transmit_i = 0;
bool Done = false;

SoftwareSerial comSerial(3, 4); // RX, TX

void setup() {
  Serial.begin(57600);
  comSerial.begin(4800);
}


 void loop() {
    if (Serial.available()) {
     uint8_t Input_Char = (uint8_t)Serial.read();
     if (Input_Char == '\n') {
       Done = true;
     }
     else{
       transmit[transmit_i] += Input_Char;
       transmit_i++;
     }
   }

   if (Done) {
     encodeFrame(destination_MAC, source_MAC, transmit, transmit_i);

     for(int i = 0; i<Eth_Size; i++) comSerial.write(Eth_Buffer[i]);
     comSerial.write('\n');

     Done = false;
     for(int i = 0; i < transmit_i+1; i++)transmit[i] = 0;
     transmit_i = 0;
   }
 }
```

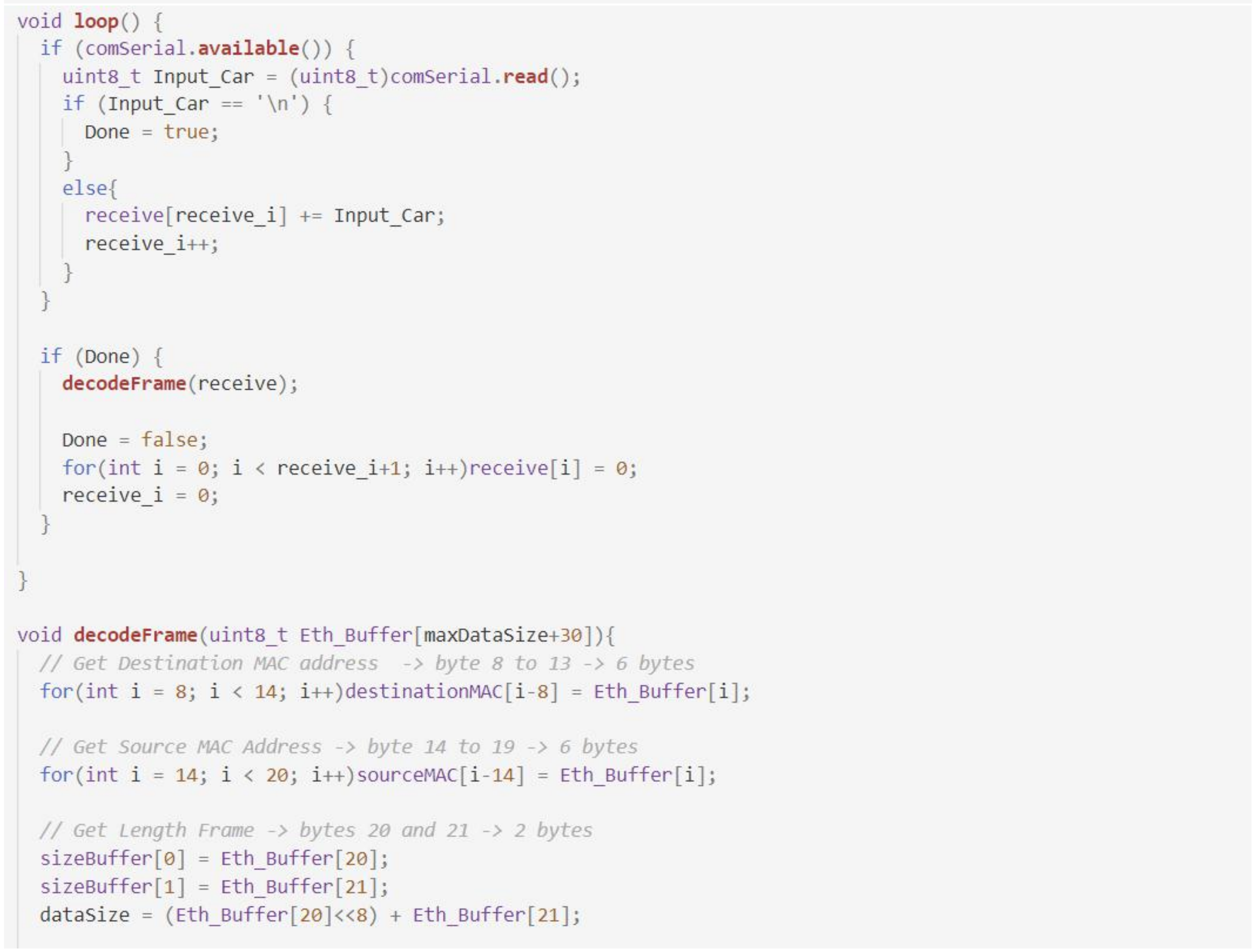


```
void encodeFrame(uint8_t destinationMAC[6], uint8_t sourceMAC[6], uint8_t data[maxDataSize], int dataSize){
  // Clear Buffer
  for(int i = 0; i< maxDataSize+30; i++) Eth_Buffer[i] = 0x00;

  // Latch minimum data to 46 charachters
  if (dataSize < 46){
    for(int i = dataSize; i < 47; i++){
      data[i] = 0x00;
    }
    dataSize = 46;
  }

  // Add Preamble Frame -> byte 0 to 6 -> 7 bytes
  for(int i = 0; i < 7; i++)Eth_Buffer[i] = 0xAA;

  // Add Start Frame Delimeter -> byte 7 -> 1 byte
  Eth_Buffer[7] = 0xAB;

  // Add Destination MAC address  -> byte 8 to 13 -> 6 bytes
  for(int i = 8; i < 14; i++)Eth_Buffer[i] = destinationMAC[i-8];

  // Add Source MAC Address -> byte 14 to 19 -> 6 bytes
  for(int i = 14; i < 20; i++)Eth_Buffer[i] = sourceMAC[i-14];

  // Add Length Frame -> bytes 20 and 21 -> 2 bytes
  Eth_Buffer[20] = dataSize / 512; // First byte of size frame
  Eth_Buffer[21] = dataSize % 512; // Second byte of size frame

  // Add Data Frame -> bytes 22 to dataSize+22 (max 1522) (min 68) -> 46 - 1500 bytes
  for(int i = 22; i < dataSize+23; i++)Eth_Buffer[i] = data[i-22];
  // Calculate CRC32
  uint32_t crc = crc32(data, dataSize, 0x04C11DB7, 0xFFFFFFFF, 0xFFFFFFFF, true, false);

  // Add CRC Frame -> bytes dataSize+22 to dataSize+26 -> 4 bytes
  Eth_Buffer[dataSize+26] = (crc & 0x000000ff);
  Eth_Buffer[dataSize+25] = (crc & 0x0000ff00) >> 8;
  Eth_Buffer[dataSize+24] = (crc & 0x00ff0000) >> 16;
  Eth_Buffer[dataSize+23] = (crc & 0xff000000) >> 24;

  // Calculate total frame size
  Eth_Size = dataSize+27;

  Serial.print("Encoded ethernet frame in HEX: \t");
  for(int i = 0; i<Eth_Size; i++) {
    Serial.print ("0x");
    Serial.print (Eth_Buffer[i], HEX);
    Serial.print (" ");
    }
   Serial.println("");
   Serial.println("");
}
```

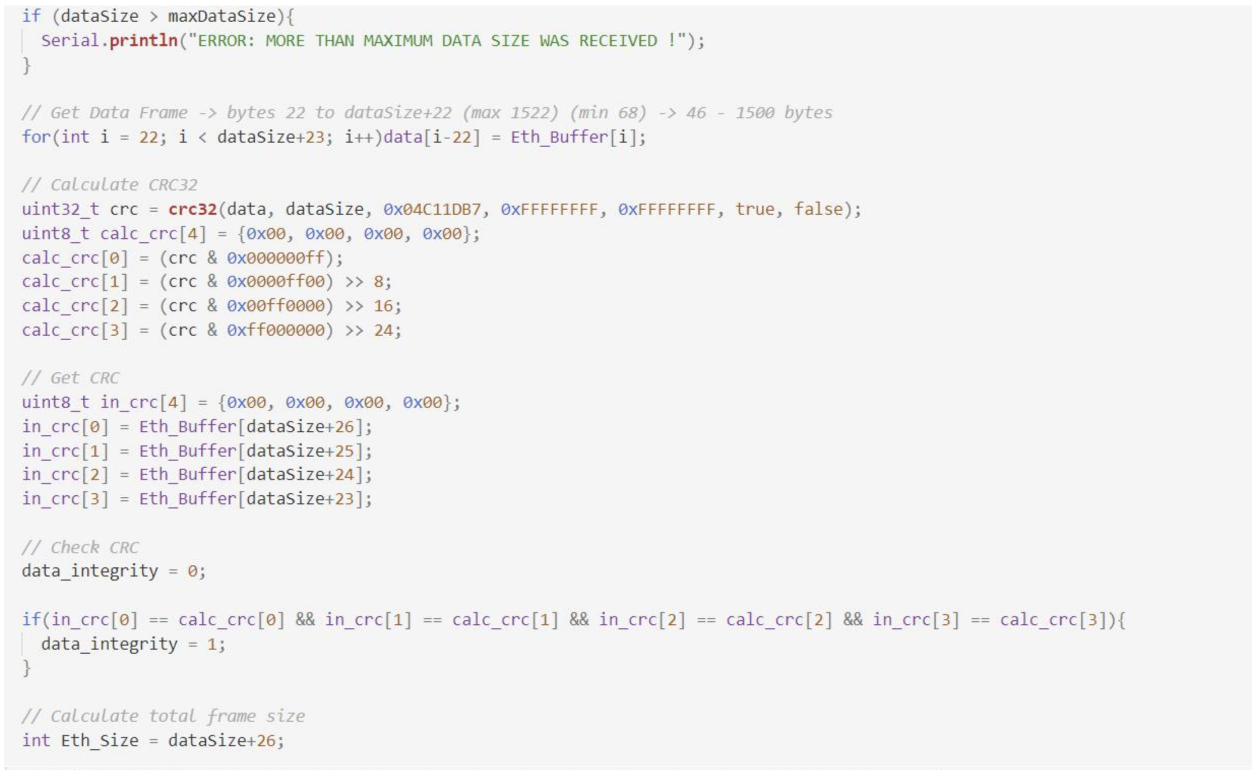
RECEIVER CODE



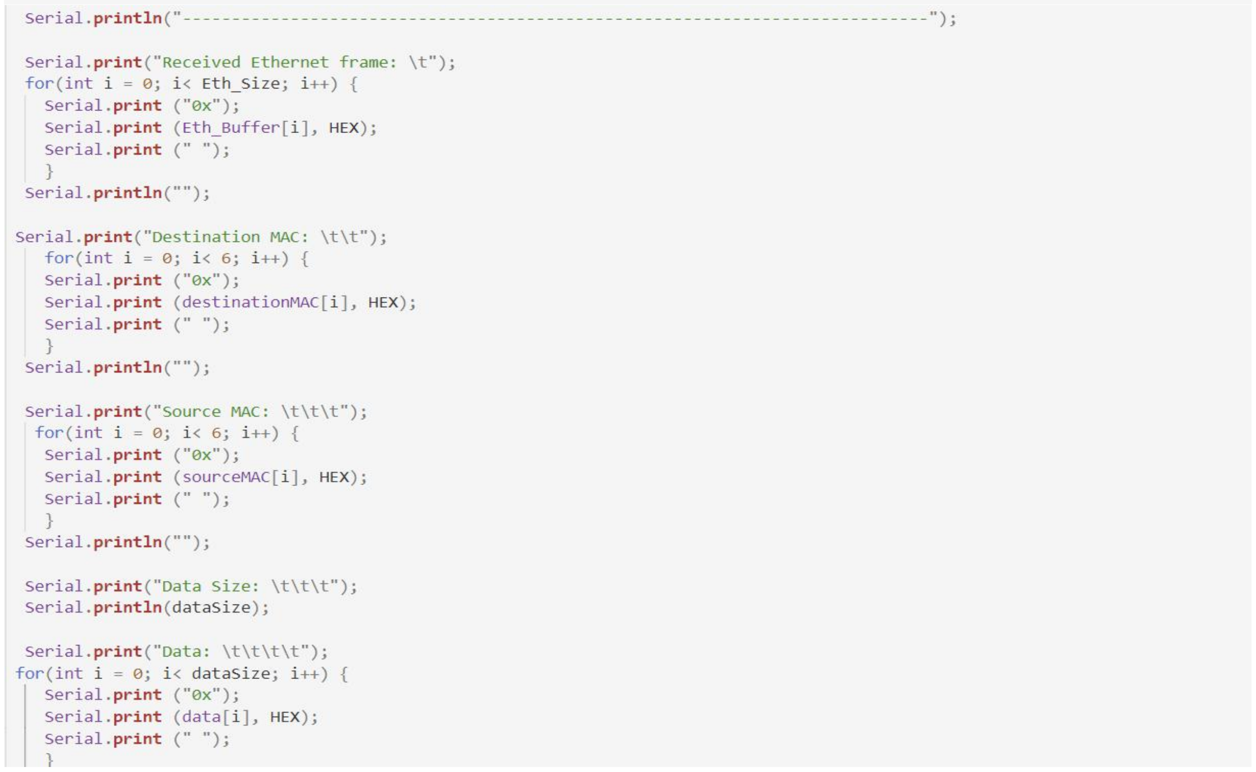
```
#include "CRC.h"
#include <SoftwareSerial.h>

#define maxDataSize 512

uint8_t destinationMAC[6];
uint8_t sourceMAC[6];
uint8_t data[maxDataSize];
uint8_t sizeBuffer[2];
int dataSize;
int data_integrity;

uint8_t receive[maxDataSize];
int receive_i = 0;
bool Done = false;

SoftwareSerial comSerial(3, 4); // RX, TX


void setup() {
  Serial.begin(57600);
  comSerial.begin(4800);
}
```



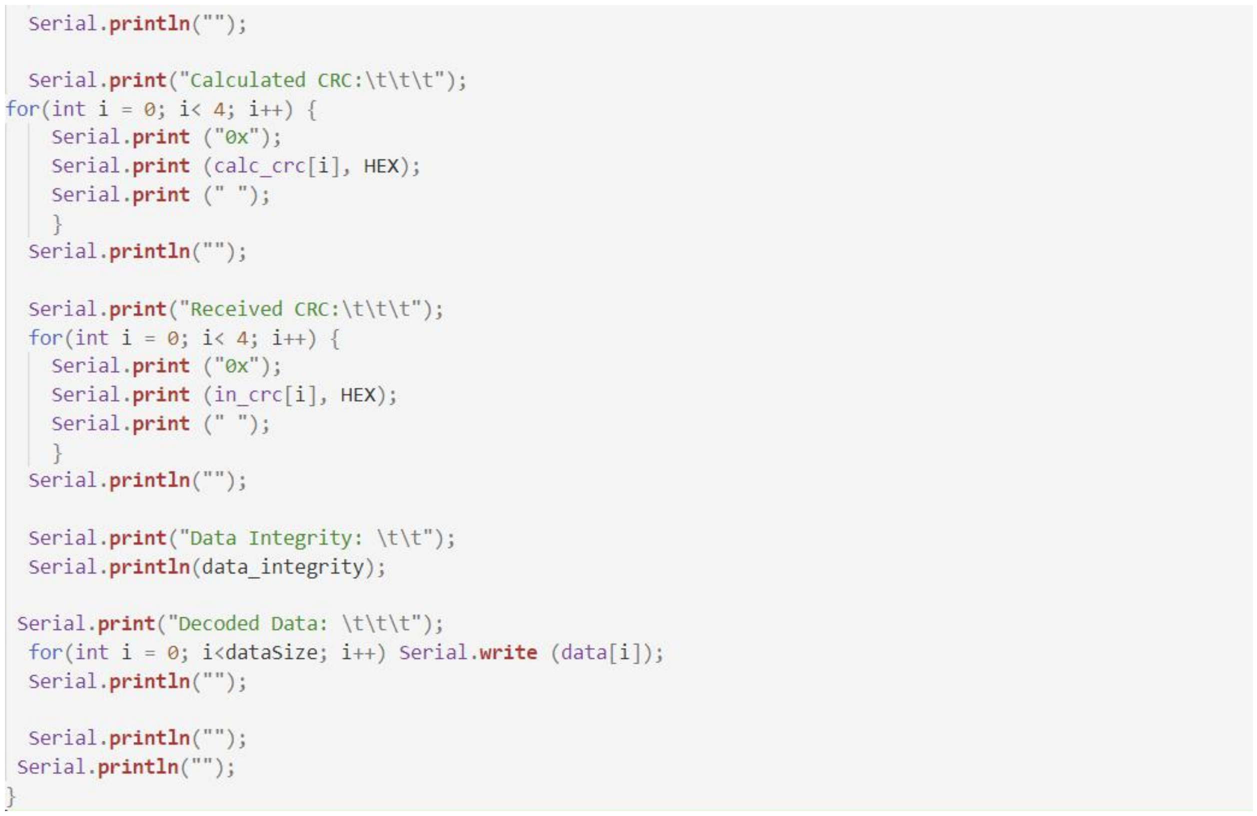
```
void loop() {
  if (comSerial.available()) {
    uint8_t Input_Car = (uint8_t)comSerial.read();
    if (Input_Car == '\n') {
      Done = true;
    }
    else{
      receive[receive_i] += Input_Car;
      receive_i++;
    }
  }

  if (Done) {
    decodeFrame(receive);

    Done = false;
    for(int i = 0; i < receive_i+1; i++)receive[i] = 0;
    receive_i = 0;
  }

}

void decodeFrame(uint8_t Eth_Buffer[maxDataSize+30]){
  // Get Destination MAC address  -> byte 8 to 13 -> 6 bytes
  for(int i = 8; i < 14; i++)destinationMAC[i-8] = Eth_Buffer[i];

  // Get Source MAC Address -> byte 14 to 19 -> 6 bytes
  for(int i = 14; i < 20; i++)sourceMAC[i-14] = Eth_Buffer[i];

  // Get Length Frame -> bytes 20 and 21 -> 2 bytes
  sizeBuffer[0] = Eth_Buffer[20];
  sizeBuffer[1] = Eth_Buffer[21];
  dataSize = (Eth_Buffer[20]<<8) + Eth_Buffer[21];
```



```
if (dataSize > maxDataSize){
  Serial.println("ERROR: MORE THAN MAXIMUM DATA SIZE WAS RECEIVED !");
}

// Get Data Frame -> bytes 22 to dataSize+22 (max 1522) (min 68) -> 46 - 1500 bytes
for(int i = 22; i < dataSize+23; i++)data[i-22] = Eth_Buffer[i];

// Calculate CRC32
uint32_t crc = crc32(data, dataSize, 0x04C11DB7, 0xFFFFFFFF, 0xFFFFFFFF, true, false);
uint8_t calc_crc[4] = {0x00, 0x00, 0x00, 0x00};
calc_crc[0] = (crc & 0x000000ff);
calc_crc[1] = (crc & 0x0000ff00) >> 8;
calc_crc[2] = (crc & 0x00ff0000) >> 16;
calc_crc[3] = (crc & 0xff000000) >> 24;

// Get CRC
uint8_t in_crc[4] = {0x00, 0x00, 0x00, 0x00};
in_crc[0] = Eth_Buffer[dataSize+26];
in_crc[1] = Eth_Buffer[dataSize+25];
in_crc[2] = Eth_Buffer[dataSize+24];
in_crc[3] = Eth_Buffer[dataSize+23];

// Check CRC
data_integrity = 0;

if(in_crc[0] == calc_crc[0] && in_crc[1] == calc_crc[1] && in_crc[2] == calc_crc[2] && in_crc[3] == calc_crc[3]){
  data_integrity = 1;
}

// Calculate total frame size
int Eth_Size = dataSize+26;
```



```
 Serial.println("---------------------------------------------------------------------------");

 Serial.print("Received Ethernet frame: \t");
 for(int i = 0; i< Eth_Size; i++) {
   Serial.print ("0x");
   Serial.print (Eth_Buffer[i], HEX);
   Serial.print (" ");
   }
 Serial.println("");

Serial.print("Destination MAC: \t\t");
   for(int i = 0; i< 6; i++) {
   Serial.print ("0x");
   Serial.print (destinationMAC[i], HEX);
   Serial.print (" ");
   }
 Serial.println("");

 Serial.print("Source MAC: \t\t\t");
  for(int i = 0; i< 6; i++) {
   Serial.print ("0x");
   Serial.print (sourceMAC[i], HEX);
   Serial.print (" ");
   }
 Serial.println("");

 Serial.print("Data Size: \t\t\t");
 Serial.println(dataSize);

 Serial.print("Data: \t\t\t\t");
for(int i = 0; i< dataSize; i++) {
   Serial.print ("0x");
   Serial.print (data[i], HEX);
   Serial.print (" ");
   }
```



```
  Serial.println("");

  Serial.print("Calculated CRC:\t\t\t");
for(int i = 0; i< 4; i++) {
    Serial.print ("0x");
    Serial.print (calc_crc[i], HEX);
    Serial.print (" ");
    }
  Serial.println("");

  Serial.print("Received CRC:\t\t\t");
  for(int i = 0; i< 4; i++) {
    Serial.print ("0x");
    Serial.print (in_crc[i], HEX);
    Serial.print (" ");
    }
  Serial.println("");

  Serial.print("Data Integrity: \t\t");
  Serial.println(data_integrity);

 Serial.print("Decoded Data: \t\t\t");
  for(int i = 0; i<dataSize; i++) Serial.write (data[i]);
  Serial.println("");

  Serial.println("");
 Serial.println("");
}
```

DRIVE LINK FOR VIDEO AND CODE

https://youtu.be/O7DGmUS7JsM