RĪGAS TEHNISKĀ UNIVERSITĀTE

ELEKTRONIKAS UN TELEKOMUNIKĀCIJU FAKULTĀTE

Datu pārraide bezvadu sensoru tīklos

Laboratorijas darbs Nr.5

I REGV0

Ruslans Babajans 171REB152

Rīga, 2021

Darba uzdevums:

Laboratorijas darbā tiek modificēts kods lai iekļautu: ierīču adreses (un kādai ierīcei ziņas tiek adresētas), ziņu pāradresāciju, ziņu pārsūtījuma skaitu (Network lalyer). Kopumā tas veido "Flooding" datu maršrutēšanas algoritmu.

Izmantotā aparatūra: DISCO-L072CZ-LRWAN1 izstrādes platforma.

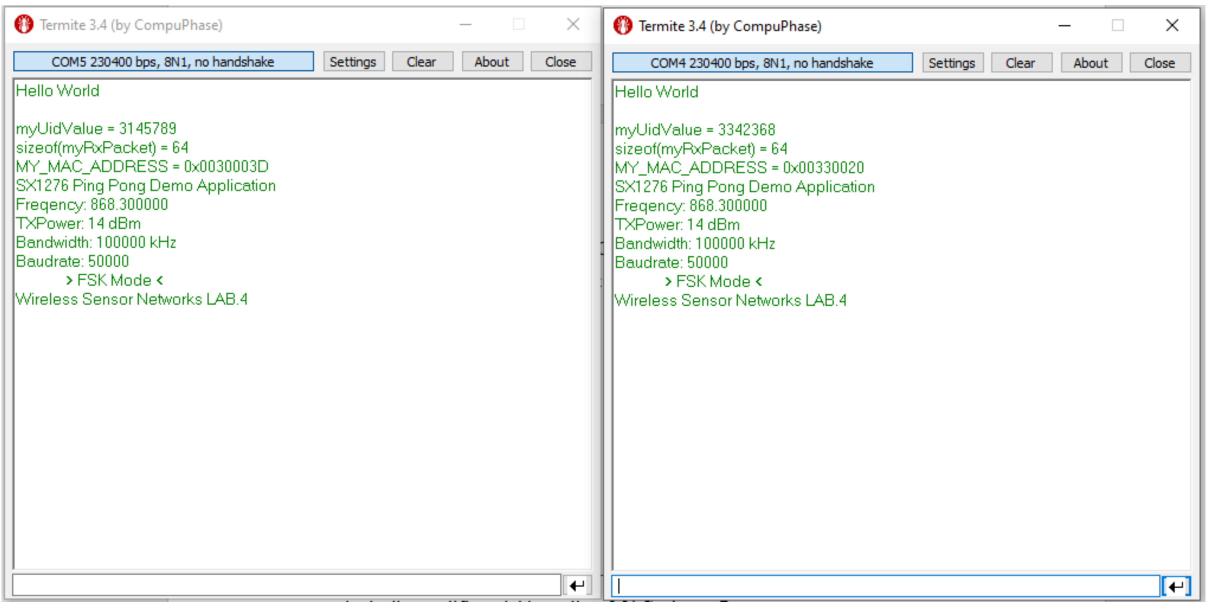
Izstrādātais c++ kods:

```
// UID of used kits are: 3145789, 3342368

#define MY_MAC_ADDRESS myUidValue
#define OTHER_MAC_ADDRESS 3342368
#include "mbed.h"
#include "PinMap.h"
#include "GenericPingPong.h"
#include "sx1276-mbed-hal.h"
#include "main.h"
#ifdef FEATURE_LORA
/* Set this flag to '1' to display debug messages on the console */
#define DEBUG_MESSAGE   1
/* Set this flag to '1' to use the LoRa modulation or to '0' to use FSK modulation */
#define USE_MODEM_LORA  0
#define USE_MODEM_FSK   !USE_MODEM_LORA
#define RF_FREQUENCY            868300000 // Hz
#define TX_OUTPUT_POWER         14                // 14 dBm
#if USE_MODEM_LORA == 1
#define LORA_BANDWIDTH          125000  // LoRa default, details in SX1276::BandwidthMap
#define LORA_SPREADING_FACTOR   LORA_SF7
#define LORA_CODINGRATE         LORA_ERROR_CODING_RATE_4_5
#define LORA_PREAMBLE_LENGTH    8       // Same for Tx and Rx
#define LORA_SYMBOL_TIMEOUT     5       // Symbols
#define LORA_FIX_LENGTH_PAYLOAD_ON  false
#define LORA_FHSS_ENABLED       false
#define LORA_NB_SYMB_HOP        4
#define LORA_IQ_INVERSION_ON    false
#define LORA_CRC_ENABLED        true

#elif USE_MODEM_FSK == 1
#define FSK_FDEV                25000     // Hz                H=1
#define FSK_DATARATE            50000     // bps
#define FSK_BANDWIDTH           100000    // Hz                TX_BW
#define FSK_AFC_BANDWIDTH       103473    // Hz                RX_BW
#define FSK_PREAMBLE_LENGTH     5         // Same for Tx and Rx
#define FSK_FIX_LENGTH_PAYLOAD_ON   false
#define FSK_CRC_ENABLED         true
```

```
#else
   #error "Please define a modem in the compiler options."
#endif
#define RX_TIMEOUT_VALUE    0   // in ms
#define MAX_RX_LENGTH                          64
#define MAX_HOP_COUNT                          3
#define LENGTH_OFFSET       11
/*
 *  Global variables declarations
 */
typedef enum
{
   IDLE,

   RX_COMPLETE,
   RX_TIMEOUT,
   RX_ERROR,

   TX_START,
   TX_WAITING_COMPLETE,
   TX_COMPLETE,
   TX_TIMEOUT,

} AppStates_t;
volatile AppStates_t State = IDLE;

/*!
 * Radio events function pointer
 */
static RadioEvents_t RadioEvents;

/*
 *  Global variables declarations
 */
SX1276Generic *Radio;
typedef __packed struct {
            uint8_t length;
            uint8_t hopCounter;
            uint32_t destinationAddress;
            uint32_t sourceAddress;
            uint8_t payload[54];
} myPacket_t;

myPacket_t myRxPacket, myTxPacket;
```

```
volatile uint8_t lastRxLength;

uint32_t buttonPressCounter;

InterruptIn mybutton(USER_BUTTON);

void myButtonInterruptFunction(){
        if(State == IDLE){
                State = TX_START;
                buttonPressCounter++;
        }
}
int SX1276PingPong()
{
        uint32_t * addressUID = (uint32_t *)(0x1FF80050 + 0x14);
        uint32_t myUidValue = *addressUID;
        dprintf("myUidValue = %u", myUidValue);
        srand(myUidValue);
        dprintf("sizeof(myRxPacket) = %u", sizeof(myRxPacket));
        dprintf("MY_MAC_ADDRESS = 0x%08X", MY_MAC_ADDRESS);
        mybutton.fall(&myButtonInterruptFunction);

    Radio = new SX1276Generic(NULL, MURATA_SX1276,
                LORA_SPI_MOSI, LORA_SPI_MISO, LORA_SPI_SCLK, LORA_CS, LORA_RESET,
        LORA_DIO0, LORA_DIO1, LORA_DIO2, LORA_DIO3, LORA_DIO4, LORA_DIO5,
        LORA_ANT_RX, LORA_ANT_TX, LORA_ANT_BOOST, LORA_TCXO);

    uint8_t i;

    dprintf("SX1276 Ping Pong Demo Application" );
    dprintf("Freqency: %.6f", (double)RF_FREQUENCY/1000000.0);
    dprintf("TXPower: %d dBm",  TX_OUTPUT_POWER);
#if USE_MODEM_LORA == 1
    dprintf("Bandwidth: %d Hz", LORA_BANDWIDTH);
    dprintf("Spreading factor: SF%d", LORA_SPREADING_FACTOR);
#elif USE_MODEM_FSK == 1
    dprintf("Bandwidth: %d kHz",  FSK_BANDWIDTH);
    dprintf("Baudrate: %d", FSK_DATARATE);
#endif
    // Initialize Radio driver
    RadioEvents.TxDone = OnTxDone;
    RadioEvents.RxDone = OnRxDone;
    RadioEvents.RxError = OnRxError;
    RadioEvents.TxTimeout = OnTxTimeout;
    RadioEvents.RxTimeout = OnRxTimeout;
    if (Radio->Init( &RadioEvents ) == false) {
        while(1) {
                dprintf("Radio could not be detected!");
                wait( 1 );
        }
    }
    Radio->SetChannel(RF_FREQUENCY );
#if USE_MODEM_LORA == 1

    if (LORA_FHSS_ENABLED)
        dprintf("             > LORA FHSS Mode <");
    if (!LORA_FHSS_ENABLED)
        dprintf("             > LORA Mode <");
    Radio->SetTxConfig( MODEM_LORA, TX_OUTPUT_POWER, 0, LORA_BANDWIDTH,
                         LORA_SPREADING_FACTOR, LORA_CODINGRATE,
                         LORA_PREAMBLE_LENGTH, LORA_FIX_LENGTH_PAYLOAD_ON,
                         LORA_CRC_ENABLED, LORA_FHSS_ENABLED, LORA_NB_SYMB_HOP,
                         LORA_IQ_INVERSION_ON, 2000 );

    Radio->SetRxConfig( MODEM_LORA, LORA_BANDWIDTH, LORA_SPREADING_FACTOR,
                         LORA_CODINGRATE, 0, LORA_PREAMBLE_LENGTH,
                         LORA_SYMBOL_TIMEOUT, LORA_FIX_LENGTH_PAYLOAD_ON, 0,
                         LORA_CRC_ENABLED, LORA_FHSS_ENABLED, LORA_NB_SYMB_HOP,
                         LORA_IQ_INVERSION_ON, true );

#elif USE_MODEM_FSK == 1
    dprintf("             > FSK Mode <");
    Radio->SetTxConfig( MODEM_FSK, TX_OUTPUT_POWER, FSK_FDEV, 0,
                         FSK_DATARATE, 0,
                         FSK_PREAMBLE_LENGTH, FSK_FIX_LENGTH_PAYLOAD_ON,
                         FSK_CRC_ENABLED, 0, 0, 0, 2000 );

    Radio->SetRxConfig( MODEM_FSK, FSK_BANDWIDTH, FSK_DATARATE,
                         0, FSK_AFC_BANDWIDTH, FSK_PREAMBLE_LENGTH,
                         0, FSK_FIX_LENGTH_PAYLOAD_ON, 0, FSK_CRC_ENABLED,
                         0, 0, false, true );

#else
#error "Please define a modem in the compiler options."
#endif
        dprintf("Wireless Sensor Networks LAB.4");

    Radio->Rx( RX_TIMEOUT_VALUE );

    while( 1 )
    {
#ifdef TARGET_STM32L4
        WatchDogUpdate();
#endif

        switch( State )
        {
        case IDLE:
            // Do nothing - wait for button interrupt
            sleep();
            break;
        case RX_COMPLETE:
            if (myRxPacket.length == lastRxLength){
                    dprintf("Rx complete!");
                    dprintf("myRxPacket.length = %u",myRxPacket.length);
                    dprintf("myRxPacket.hopCounter = %u",myRxPacket.hopCounter);
                    dprintf("myRxPacket.sourceAddress = 0x%08X",myRxPacket.sourceAddress);
                    dprintf("myRxPacket.destinationAddress = 0x%08X",myRxPacket.destinationAddress);

                    if(myRxPacket.destinationAddress == MY_MAC_ADDRESS){
                        dprintf("This packet is for us!");
                        myRxPacket.payload[myRxPacket.length-LENGTH_OFFSET]=0; // Add Null at the end of text
manually
                        dprintf("Payload is %s\n",myRxPacket.payload);
                      }
                      else{
                        dprintf("Packet was not for us (different destinationAddress)");
                        if (myRxPacket.sourceAddress == MY_MAC_ADDRESS){
                            dprintf("This packet was sent for us, so we do not forward it again");
                        }
                        else{
                            if (myRxPacket.hopCounter >0){
                                dprintf("Hop counter > 0, forwarding packet");
                                myRxPacket.hopCounter--;
                                uint16_t random_wait_ms = rand() % 1000;
                                dprintf("Waiting %d ms before forwarding", random_wait_ms);
                                wait(random_wait_ms * 0.001);
                                Radio->Send( &myRxPacket, myRxPacket.length );
                                State = TX_WAITING_COMPLETE;
                                break;
                            }
                        }
                      }
            }
            else{
                dprintf("Packet length error!");
            }
            Radio->Rx( RX_TIMEOUT_VALUE ); // Put transceiver back to Rx
          State = IDLE;
          break;
        case RX_TIMEOUT:
            Radio->Rx( RX_TIMEOUT_VALUE ); // Put transceiver back to Rx
            dprintf("Rx Timeout happened\n");
            State = IDLE;
            break;
        case RX_ERROR:
            Radio->Rx( RX_TIMEOUT_VALUE ); // Put transceiver back to Rx
            dprintf("Rx CRC Error happened\n");
            State = IDLE;
            break;

        case TX_START:
            myTxPacket.hopCounter = MAX_HOP_COUNT;
            myTxPacket.sourceAddress = MY_MAC_ADDRESS;
            myTxPacket.destinationAddress = OTHER_MAC_ADDRESS;
            myTxPacket.length = sprintf((char*)myTxPacket.payload, "Ruslans Babajans says hello! buttonPressCounter=%d",
buttonPressCounter);
            myTxPacket.length += LENGTH_OFFSET; // Add null terminated character and header length

            Radio->Sleep( ); // First we need to put chip from Rx to Sleep state
            Radio->Send( &myTxPacket, myTxPacket.length );
            dprintf("Message sent! buttonPressCounter was %d", buttonPressCounter);
            State = TX_WAITING_COMPLETE;
            break;
        case TX_WAITING_COMPLETE:
                        sleep();
          break;
        case TX_COMPLETE:
            dprintf("Message sending complete!");
            Radio->Rx( RX_TIMEOUT_VALUE ); // Put transceiver back to Rx
            State = IDLE;
            break;
        case TX_TIMEOUT:
          dprintf("TX_TIMEOUT happened!");
            Radio->Rx( RX_TIMEOUT_VALUE ); // Put transceiver back to Rx
            State = IDLE;
            break;
        default:
          State = IDLE;
          break;
        }
    }
}
void OnTxDone(void *radio, void *userThisPtr, void *userData)
{
    Radio->Sleep( );
    State = TX_COMPLETE;
    if (DEBUG_MESSAGE)
        dprintf("> OnTxDone");
}
void OnRxDone(void *radio, void *userThisPtr, void *userData, uint8_t *payload, uint16_t size, int16_t rssi, int8_t snr)
{
    Radio->Sleep( );
    lastRxLength = size;
    if (size > MAX_RX_LENGTH){
            lastRxLength = 64;
    }
    memcpy( &myRxPacket, payload, size );
    State = RX_COMPLETE;
    if (DEBUG_MESSAGE)
        dprintf("> OnRxDone: RssiValue=%d dBm, SnrValue=%d", rssi, snr);
    dump("Data:", payload, size);
}
void OnTxTimeout(void *radio, void *userThisPtr, void *userData)
{
    Radio->Sleep( );
    State = TX_TIMEOUT;
    if(DEBUG_MESSAGE)
        dprintf("> OnTxTimeout");
}
void OnRxTimeout(void *radio, void *userThisPtr, void *userData)
{
    Radio->Sleep( );
    State = RX_TIMEOUT;
    if (DEBUG_MESSAGE)
        dprintf("> OnRxTimeout");
}
void OnRxError(void *radio, void *userThisPtr, void *userData)
{
    Radio->Sleep( );
    State = RX_ERROR;
    if (DEBUG_MESSAGE)
        dprintf("> OnRxError");
}
#endif
```

Katrai iekārtai tika pievienota MAC adrese vienāda ar mikrokontrolera UID adresi, kas ir unikāla katrai mikroshēmai. Ziņas tiek adresētas vienai no iekārtām un sākumā “lēcienu skaitītājam” maksimāla vērtība ir 3.



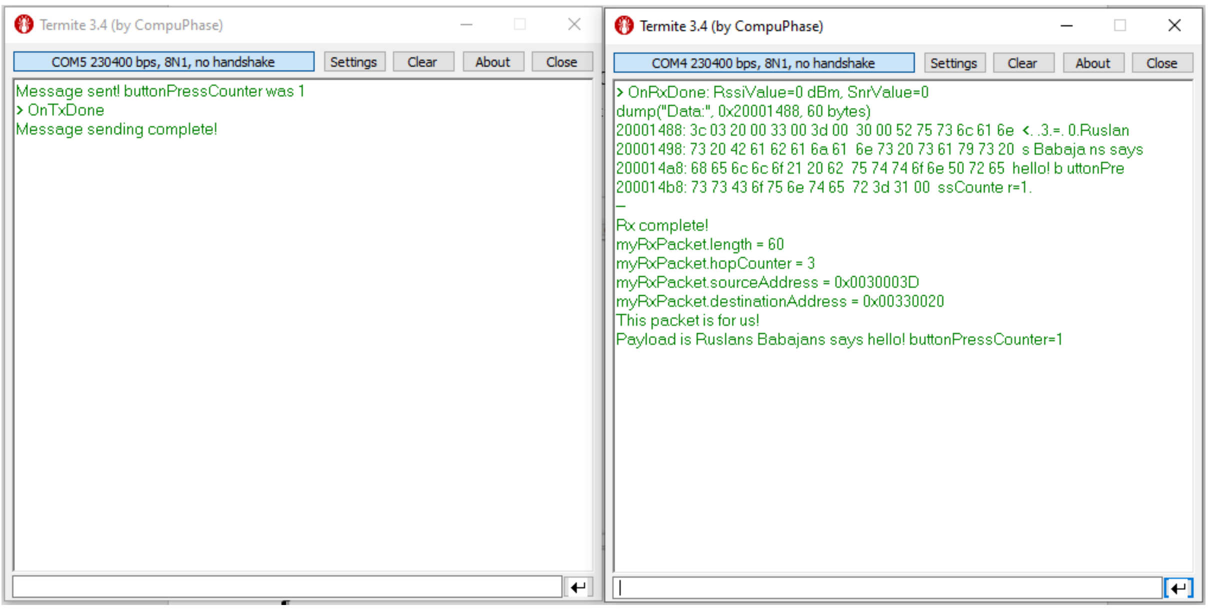
Termite 3.4 (by CompuPhase)
COM5 230400 bps, 8N1, no handshake
Settings
Clear
About
Close

```
Hello World

myUidValue = 3145789
sizeof(myRxPacket) = 64
MY_MAC_ADDRESS = 0x0030003D
SX1276 Ping Pong Demo Application
Freqency: 868.300000
TXPower: 14 dBm
Bandwidth: 100000 kHz
Baudrate: 50000
        > FSK Mode <
Wireless Sensor Networks LAB.4
```

Termite 3.4 (by CompuPhase)
COM4 230400 bps, 8N1, no handshake
Settings
Clear
About
Close

```
Hello World

myUidValue = 3342368
sizeof(myRxPacket) = 64
MY_MAC_ADDRESS = 0x00330020
SX1276 Ping Pong Demo Application
Freqency: 868.300000
TXPower: 14 dBm
Bandwidth: 100000 kHz
Baudrate: 50000
        > FSK Mode <
Wireless Sensor Networks LAB.4
```

1. att. Divu iekāru parametri



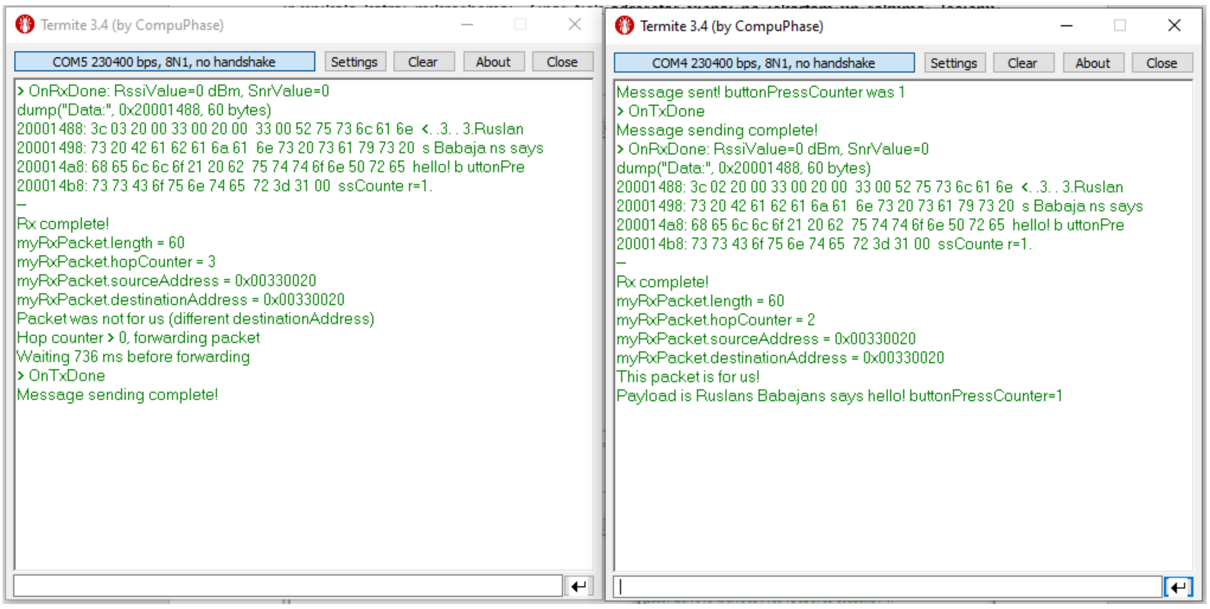
Termite 3.4 (by CompuPhase)
COM5 230400 bps, 8N1, no handshake
Settings
Clear
About
Close

```
Message sent! buttonPressCounter was 1
> OnTxDone
Message sending complete!
```

Termite 3.4 (by CompuPhase)
COM4 230400 bps, 8N1, no handshake
Settings
Clear
About
Close

```
> OnRxDone: RssiValue=0 dBm, SnrValue=0
dump("Data:", 0x20001488, 60 bytes)
20001488: 3c 03 20 00 33 00 3d 00  30 00 52 75 73 6c 61 6e  <. .3.=. 0.Ruslan
20001498: 73 20 42 61 62 61 6a 61  6e 73 20 73 61 79 73 20  s Babaja ns says
200014a8: 68 65 6c 6c 6f 21 20 62  75 74 74 6f 6e 50 72 65  hello! b uttonPre
200014b8: 73 73 43 6f 75 6e 74 65  72 3d 31 00  ssCounte r=1.
--
Rx complete!
myRxPacket.length = 60
myRxPacket.hopCounter = 3
myRxPacket.sourceAddress = 0x0030003D
myRxPacket.destinationAddress = 0x00330020
This packet is for us!
Payload is Ruslans Babajans says hello! buttonPressCounter=1
```

2. att. Pirmā iekārta sūt ziņu kas ir domāta otrai iekārtai


Termite 3.4 (by CompuPhase)
COM5 230400 bps, 8N1, no handshake
Settings
Clear
About
Close

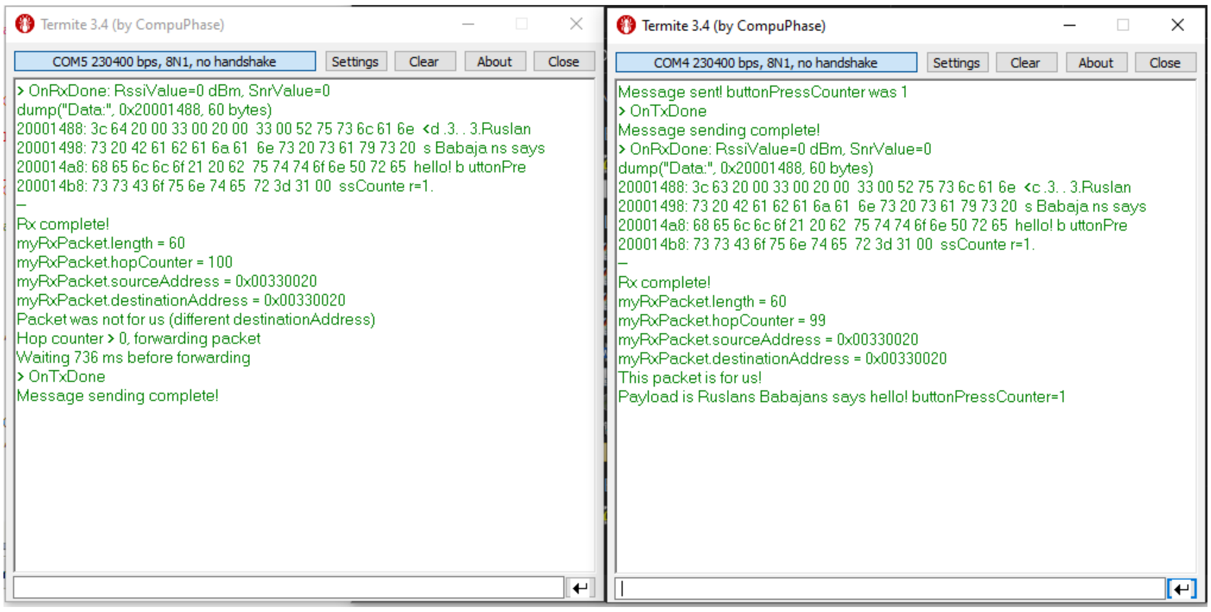
```
> OnRxDone: RssiValue=0 dBm, SnrValue=0
dump("Data:", 0x20001488, 60 bytes)
20001488: 3c 03 20 00 33 00 20 00  33 00 52 75 73 6c 61 6e  <. .3. . 3.Ruslan
20001498: 73 20 42 61 62 61 6a 61  6e 73 20 73 61 79 73 20  s Babaja ns says
200014a8: 68 65 6c 6c 6f 21 20 62  75 74 74 6f 6e 50 72 65  hello! b uttonPre
200014b8: 73 73 43 6f 75 6e 74 65  72 3d 31 00  ssCounte r=1.
–
Rx complete!
myRxPacket.length = 60
myRxPacket.hopCounter = 3
myRxPacket.sourceAddress = 0x00330020
myRxPacket.destinationAddress = 0x00330020
Packet was not for us (different destinationAddress)
Hop counter > 0, forwarding packet
Waiting 736 ms before forwarding
> OnTxDone
Message sending complete!
```

Termite 3.4 (by CompuPhase)
COM4 230400 bps, 8N1, no handshake
Settings
Clear
About
Close

```
Message sent! buttonPressCounter was 1
> OnTxDone
Message sending complete!
> OnRxDone: RssiValue=0 dBm, SnrValue=0
dump("Data:", 0x20001488, 60 bytes)
20001488: 3c 02 20 00 33 00 20 00  33 00 52 75 73 6c 61 6e  <. .3. . 3.Ruslan
20001498: 73 20 42 61 62 61 6a 61  6e 73 20 73 61 79 73 20  s Babaja ns says
200014a8: 68 65 6c 6c 6f 21 20 62  75 74 74 6f 6e 50 72 65  hello! b uttonPre
200014b8: 73 73 43 6f 75 6e 74 65  72 3d 31 00  ssCounte r=1.
–
Rx complete!
myRxPacket.length = 60
myRxPacket.hopCounter = 2
myRxPacket.sourceAddress = 0x00330020
myRxPacket.destinationAddress = 0x00330020
This packet is for us!
Payload is Ruslans Babajans says hello! buttonPressCounter=1
```

3. att. Otrā iekārta sūt ziņu kas ir domāta otrai iekārtai

2. att. var redzēt ka pirmā iekārta sūt ziņu, otrā saņem, un ziņu pāradresācija nenotiek. Bet kad otrā iekārta izsūt ziņu kas ir domāta viņai pašai, pirmā iekārta to uztver, samazina lēcienu skaitu uz 1, un pārsūt, jo ziņai nav domāta viņai. Viss strādā perfekti. Tālāk "lēcienu skaitītājam" ir uzstādīta maksimālā vērtība 100, un tiek pamēģināts 3. att. gadījums.


Termite 3.4 (by CompuPhase)
COM5 230400 bps, 8N1, no handshake
Settings
Clear
About
Close

```
> OnRxDone: RssiValue=0 dBm, SnrValue=0
dump("Data:", 0x20001488, 60 bytes)
20001488: 3c 64 20 00 33 00 20 00  33 00 52 75 73 6c 61 6e  <d .3. . 3.Ruslan
20001498: 73 20 42 61 62 61 6a 61  6e 73 20 73 61 79 73 20  s Babaja ns says
200014a8: 68 65 6c 6c 6f 21 20 62  75 74 74 6f 6e 50 72 65  hello! b uttonPre
200014b8: 73 73 43 6f 75 6e 74 65  72 3d 31 00  ssCounte r=1.
–
Rx complete!
myRxPacket.length = 60
myRxPacket.hopCounter = 100
myRxPacket.sourceAddress = 0x00330020
myRxPacket.destinationAddress = 0x00330020
Packet was not for us (different destinationAddress)
Hop counter > 0, forwarding packet
Waiting 736 ms before forwarding
> OnTxDone
Message sending complete!
```

Termite 3.4 (by CompuPhase)
COM4 230400 bps, 8N1, no handshake
Settings
Clear
About
Close

```
Message sent! buttonPressCounter was 1
> OnTxDone
Message sending complete!
> OnRxDone: RssiValue=0 dBm, SnrValue=0
dump("Data:", 0x20001488, 60 bytes)
20001488: 3c 63 20 00 33 00 20 00  33 00 52 75 73 6c 61 6e  <c .3. . 3.Ruslan
20001498: 73 20 42 61 62 61 6a 61  6e 73 20 73 61 79 73 20  s Babaja ns says
200014a8: 68 65 6c 6c 6f 21 20 62  75 74 74 6f 6e 50 72 65  hello! b uttonPre
200014b8: 73 73 43 6f 75 6e 74 65  72 3d 31 00  ssCounte r=1.
–
Rx complete!
myRxPacket.length = 60
myRxPacket.hopCounter = 99
myRxPacket.sourceAddress = 0x00330020
myRxPacket.destinationAddress = 0x00330020
This packet is for us!
Payload is Ruslans Babajans says hello! buttonPressCounter=1
```

4. att. Otrā iekārta sūt ziņu kas ir domāta otrai iekārtai

4. att. var redzēt ka no šīm izmaināmm sistēmas darbība nemainījās - lēcienu skaits tika samazināts uz 1, un ziņa tika pārsūtīta. Pēdējā solī tika izrēķināts maksimālais ziņu ilgums. MATLAB kods aprēķiniem:

```
%% Clear variables and close figures
format long, clear variables, close all
%==============================================%
%% FSK calculations
DEVIATION=25e3; % Hz
H=1; % modulation index
F0=868300000; % Hz central frequency
PPM=2; % ppm

DATA_RATE=(2*DEVIATION)/(H); % bps
TX_BW=2*DEVIATION+DATA_RATE; % Hz
RX_BW=TX_BW+F0*2*(PPM/1e6);  % Hz
%==============================================%
%% LAB. 5
bit_time=1/DATA_RATE; % DATA_RATE = 50000 bps

Max_packet_length = 512; % bits or 64B
CRC_16= 16; % bits
Preamble = 20; % bits
Sync_word = 16; % bits

Total_max_length = Max_packet_length+CRC_16+Preamble+Sync_word;

t_max=Total_max_length*bit_time; % 11.28 ms
%==============================================%
```

Aprēķinu rezultātā maksimālais ziņu ilgums ir vienāds ar 11.28 ms. Internetā neatradu ar ko salīdzināt (ieexplore un researchgate tas arī nav atrodams).

Secinājumi:

Šajā darbā tika izveidots kods kas iekļauj "Flooding" datu maršrutēšanas algoritmu. Algoritmam ir priekšrocība ka nav jāveido speciālas maršrutēšanas tabulas, bet ziņa netiek pārsūtīta pa visefektīvāko ceļu. "Flooding" algoritma realizācijai ir jāpievērš uzmanība, jo pastāv risks ka ziņa mūžīgi ceļos pa tīklu ja "lēcienu skaitītājs" un saistītas darbības ir realizētas nepareizi.