RĪGAS TEHNISKĀ UNIVERSITĀTE
ELEKTRONIKAS UN TELEKOMUNIKĀCIJU FAKULTĀTE

Datu pārraide bezvadu sensoru tīklos

Laboratorijas darbs Nr.4

I REGV0
Ruslans Babajans 171REB152

Rīga, 2021

Darba uzdevums:

Laboratorijas darbā tiek modificēts kods no 3. laboratorijas darba. Tiek ieviesta datu organizēšana paka, izmantojot "struct" tipa mainīgos ar apakšmainīgajiem jeb laukiem.

Izmantotā aparatūra: DISCO-L072CZ-LRWAN1 izstrādes platforma.

Izstrādātais c++ kods:

```
// For D:
#define MY_MAC_ADDRESS 0x1234ABCD
#define OTHER_MAC_ADDRESS 0xAABBCCDD
 // For E:
//#define MY_MAC_ADDRESS 0xAABBCCDD
//#define OTHER_MAC_ADDRESS 0x1234ABCD

#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
/*
 *  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;
            uint32_t destinationAddress;
            uint32_t sourceAddress;
            uint8_t payload[55];
} 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()
{
        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)
```

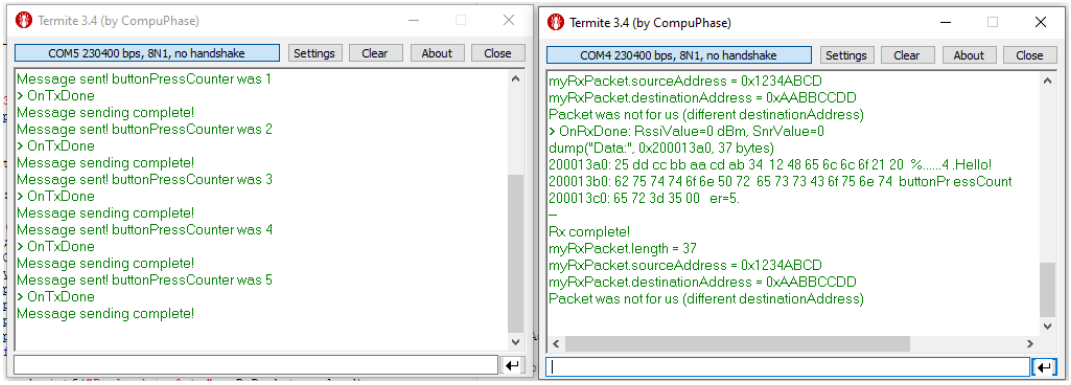
```
    dprintf("            > LORA Mode <");
  Radio->SetTxConfig( MODEM_LORA, TX_OUTPUT_POWER, 0, LORA_BANDWIDTH,
                      LORA_SPREADING_FACTOR, LORA_CODINGRATE,
                      LORA_PREAMBLE_LENGTH, LORA_FIX_LENGTH_PAYLOAD_ON,
                      LORA_CRC_ENABLED, LORA_FHSS_ENABLED, 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.sourceAddress = 0x%08X",myRxPacket.sourceAddress);
                dprintf("myRxPacket.destinationAddress = 0x%08X",myRxPacket.destinationAddress);
                if(myRxPacket.destinationAddress == MY_MAC_ADDRESS){
                        myRxPacket.payload[myRxPacket.length-10]=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)");
                    }
        }
        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.sourceAddress = MY_MAC_ADDRESS;
        myTxPacket.destinationAddress = OTHER_MAC_ADDRESS;
        myTxPacket.length = sprintf((char*)myTxPacket.payload, "Hello! buttonPressCounter=%d", buttonPressCounter);
        myTxPacket.length += 10; // 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:
```

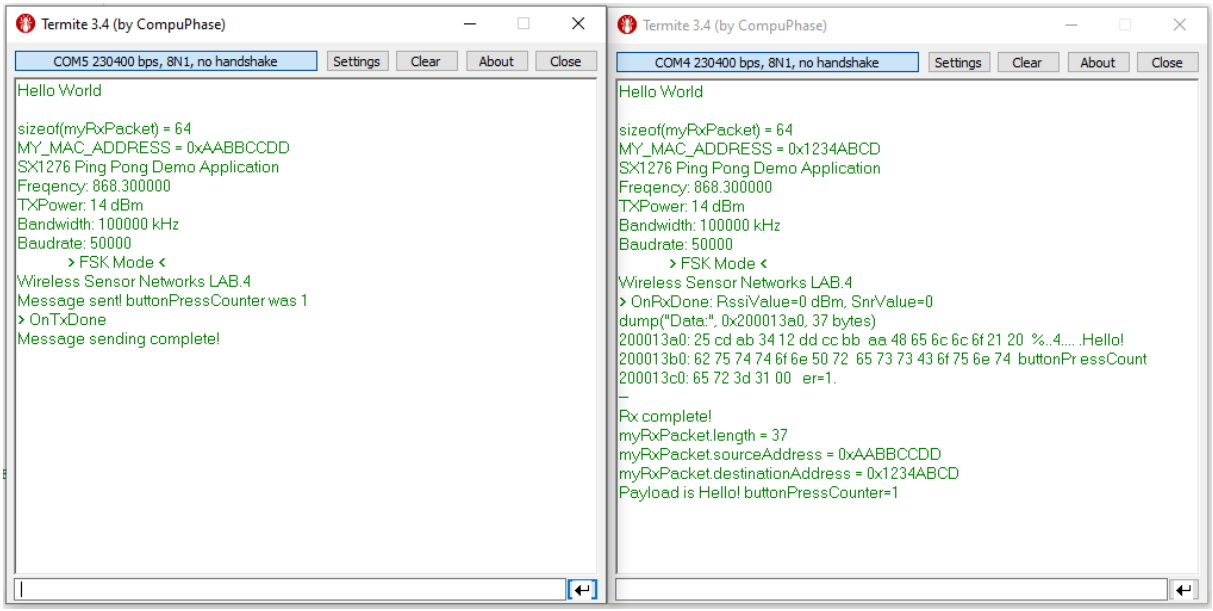
```
            dprintf("Message sending complete!");
            Radio->Rx( RX_TIMEOUT_VALUE ); // Put transceiver back to Rx
            State = IDLE;
            break;
        case TX_TIMEOUT:
          dprintf("TX_TIMEOUT happened!");
            Radio->Rx( RX_TIMEOUT_VALUE ); // Put transceiver 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
```

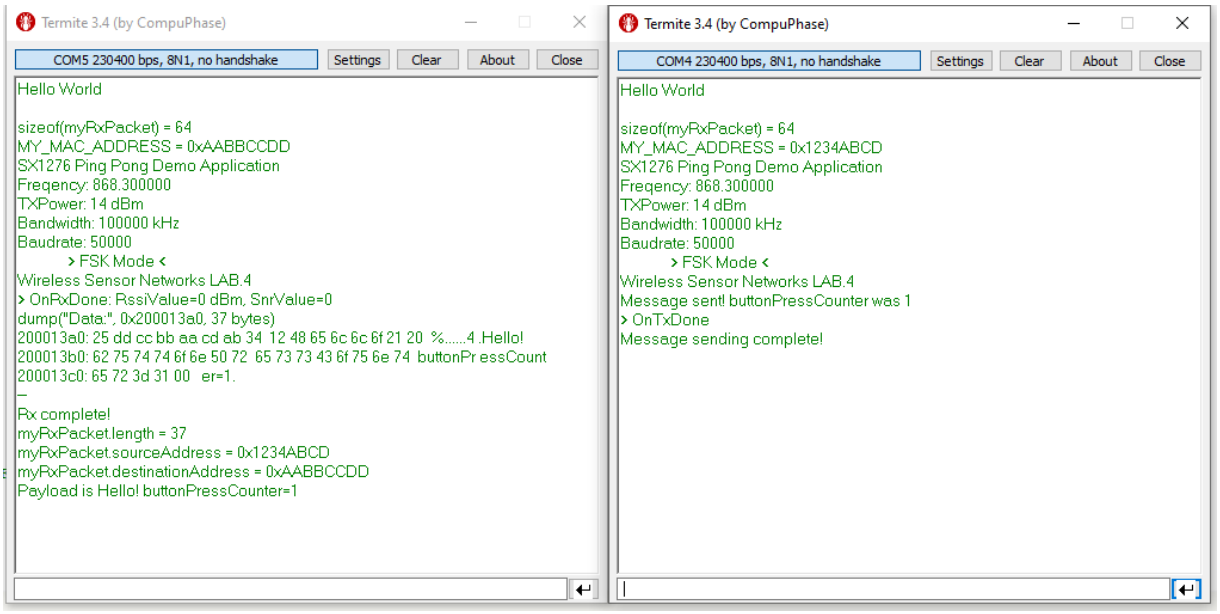
Sākumā abās iekārtās tika ielādēts kods kur ir identiska MAC adrese divām iekārtām, pēc tam kods tika modificēts lai būtu divas MAC adreses.



1. att. Termite ziņas ja iekārtam ir vienādas MAC adreses

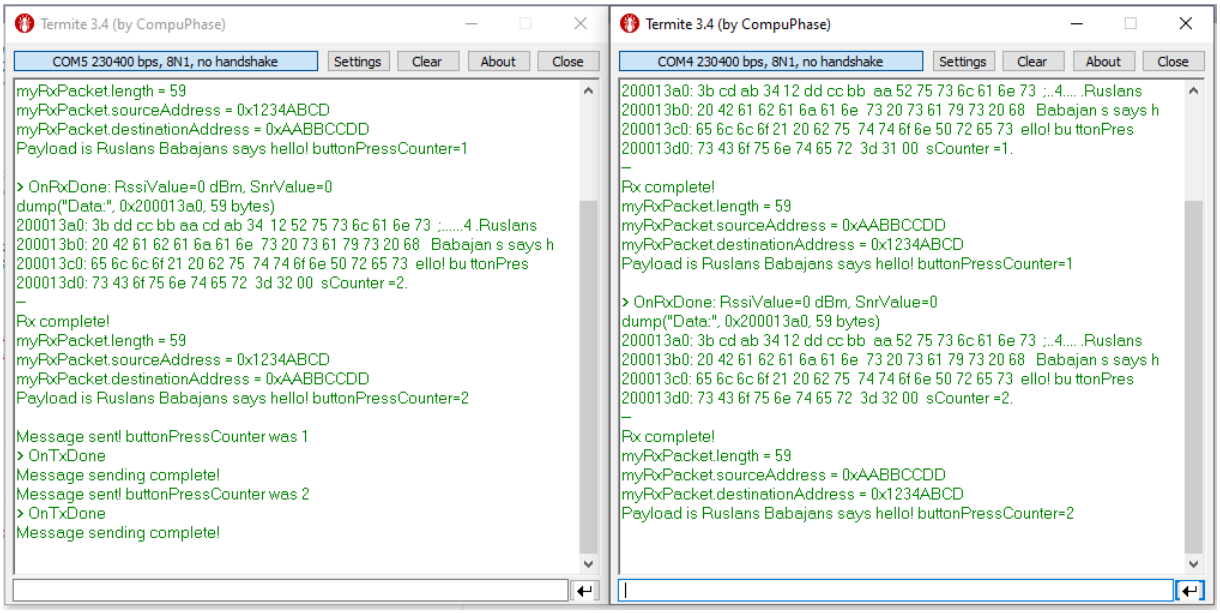


2. att. Termite ziņas ja iekārtam ir atšķirīgas MAC adreses, un poga tiek nospiesta uz vienas iekārtas



3. att. Termite ziņas ja iekārtam ir atšķirīgas MAC adreses, un poga tiek nospiesta uz otras iekārtas

Tālāk tiek uztaisīta neliela koda modifikācija lai payload būtu arī vārds un uzvārds.



4. att. Termite ziņas pie modificēta payload

Secinājumi:

Šajā darbā tika izveidots kods kas veido organizētās datu pakas papildinātas ar “source” un “destination” MAC adresēm. Izveidotas datu pakas atbilst “Data link layer”.