PA197 Secure network design

Basic wireless networking

Petr Švenda svenda@fi.muni.cz
Faculty of Informatics, Masaryk University

CROCS
Centre for Research on
Cryptography and Security

Laboratory

- Start of implementing ad-hoc networks based on Arduino with RF module
  - – Basic Arduino programming model
  - – RF library – send packet between two nodes
  - – Neighbours discovery (logical communication group)

Laboratory

- Download and run Arduino IDE
  - https://www.arduino.cc/en/Main/Software
- (On Linux: usermod –a –G dialout your_username)
- Plug in JeeNode
- Select COM port
  - Can be assigned to different values
  - Try other ports if selected does not work
- Board: Arduino Mini
- Processor: ATMega328

File→Examples→01.Basics→Blink

- Basic application, should blink the LED
- During upload, Rx and Tx small leds are blinking
- After upload, blue LED should blink (1 second)

- You should now be able to compile and upload app
  - If LED is not blinking, check PIN value
  - Should be 9 for JeeNode => 13->9

Blink.ino

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin as an output.
  pinMode(13, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);              // wait for a second
}
```

- (Note that PIN used for LED can be different on different boards, 9 on JeeNode)

Troubleshooting

- Check if you have proper board and processor
  - Arduino Mini, ATMega328
- Don’t have serial monitor running if going to upload new app
- Try to re-plug jeenode
- Try to plug into different USB port
- Try to restart Arduino IDE
- Check if you have same serial port speed on Arduino and port monitor
  - Try different speeds, otherwise you will see garbled data
- Try again (anything ☺)

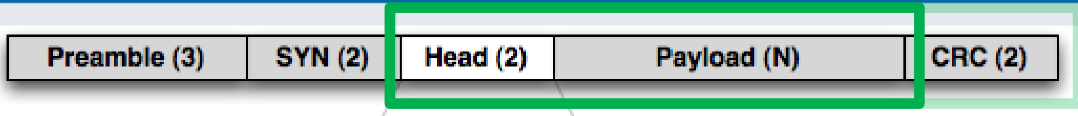
File→Examples … →DigitalReadSerial

- Original code prints state of button to serial port
- Run Serial monitor
  - Will automatically restart Arduino board
  - Observe data as print out
- Modify to print out loop counter (instead of button)
  - Small red LED should blink during data transfer
- You should now be able to upload application and see data via serial port
- You may use any other application to capture data
  - https://github.com/gskielian/Arduino-DataLogging/tree/master/PySerial

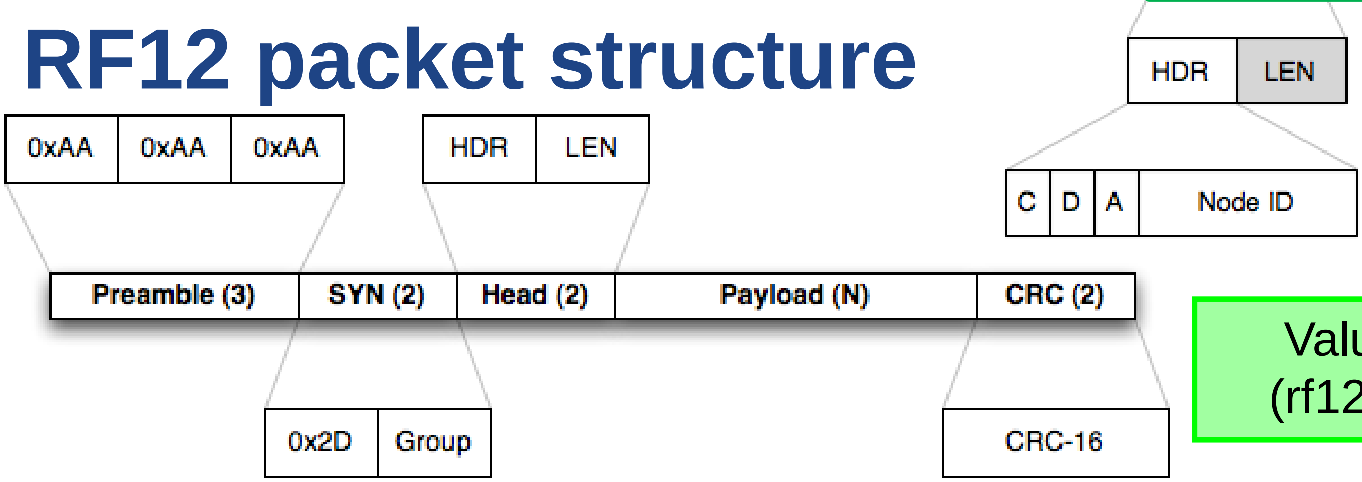
RF NETWORKING WITH JEELIB

JeeLib library

- Provides support for JeeNode radio module
- Download Jeelib-master.lib
  – https://github.com/jcw/jeelib/archive/master.zip
- Documentation: http://jeelabs.org/pub/docs/jeelib/index.html
- Add library into Arduino IDE
  – Sketch → Include library → Add .ZIP library
    - (Check Include library again, Contributed libraries)
  – Examples are now available: Examples → jeelib-master



RF12 packet structure



Values set/visible via (rf12_send, rf12_recv)

- C = CTL, D = DST, A = ACK, 5-bit node ID
  - A bit (ACK) – indicates if sender wants to get ACK back
  - D bit (DST) – indicates if node ID bits specify destination or source node
  - C bit (CTL) – 1 if packet is ACK (and A must be 0)
- To send packet only to node with nodeID
  - rf12_sendNow(**RF12_HDR_DST | nodeID**, &data, dataLen);
- Warning: radio is always broadcast in nature, filtering only in driver!
- http://jeelabs.org/2011/06/09/rf12-packet-format-and-design/index.html
- http://jeelabs.org/2011/06/10/rf12-broadcasts-and-acks/index.html

test1.ino

```
#include <JeeLib.h>

const byte LED = 9;
byte counter;

// turn the on-board LED on or off
static void led (bool on) {
  pinMode(LED, OUTPUT);
  digitalWrite(LED, on ? 0 : 1); // inverted logic
}

void setup () {
  // this is node 1 in net group 100 on the 868 MHz band
  rf12_initialize(1, RF12_868MHZ, 100);
}

void loop () {
  led(true);

  // actual packet send: broadcast to all, current counter, 1 byte long
  rf12_sendNow(0, &counter, 1);
  rf12_sendWait(1);

  led(false);

  // increment the counter (it'll wrap from 255 to 0)
  ++counter;
  // let one second pass before sending out another packet
  delay(1000);
}
```

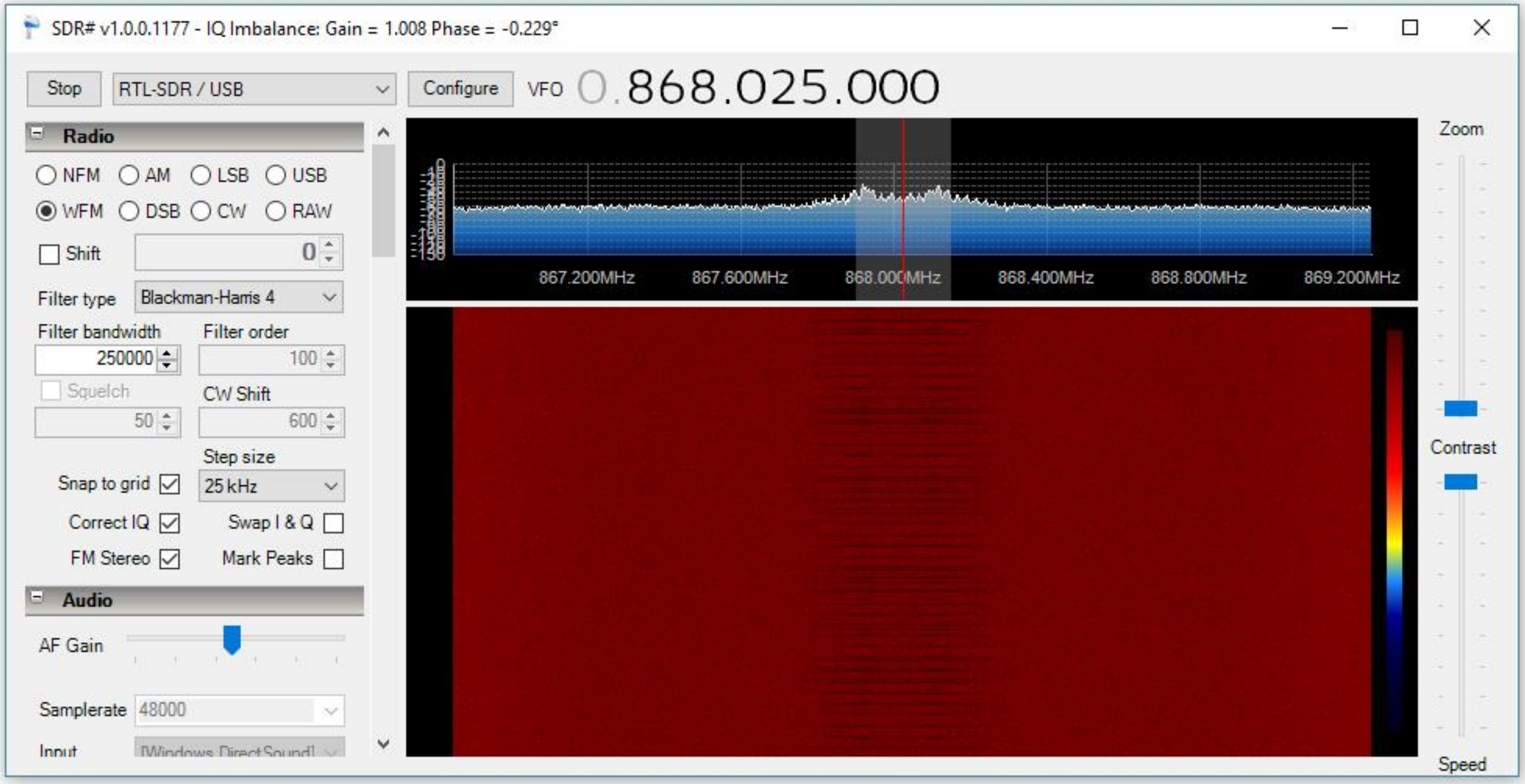
```
rf12_sendNow(RF12_HDR_DST | nodeID, …);
rf12_sendNow(RF12_HDR_DST | nodeID | RF12_HDR_ACK|, …);
```

File→Examples→jeelib-master→DINJ → test1

Basic beacon application

- Select File→Examples→jeelib-master→DINJ→ test1
  - – Compile, upload
  - – Application sends packet with counter every second
- Try to change your node ID (1..31 possible)
  - – rf12_initialize(1, RF12_868MHZ, 100);
  - – 31 is special ID for promiscuous mode (receives everything)
- Try to change your group
  - – rf12_initialize(1, RF12_868MHZ, 100);
  - – You will hear only messages within your group

(Another option to monitor packets – SDR!)


SDR# v1.0.0.1177 - IQ Imbalance: Gain = 1.008 Phase = -0.229°

Stop | RTL-SDR / USB | Configure | VFO 0.868.025.000

Radio

○ NFM ○ AM ○ LSB ○ USB
● WFM ○ DSB ○ CW ○ RAW

☐ Shift 0

Filter type: Blackman-Harris 4

Filter bandwidth: 250000
Filter order: 100

☐ Squelch: 50
CW Shift: 600

Step size: 25 kHz

Snap to grid ☑
Correct IQ ☑ Swap I & Q ☐
FM Stereo ☑ Mark Peaks ☐

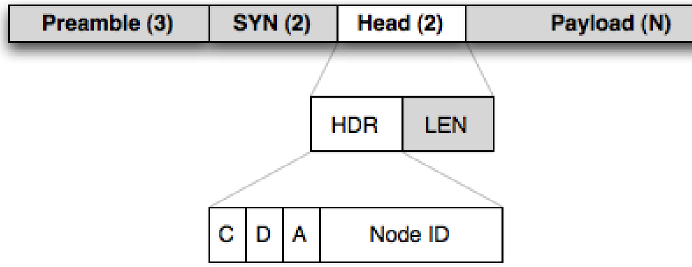
Audio

AF Gain

Samplerate: 48000

Input: [Windows DirectSound]

Basic beacon application – send packet



- rf12_sendNow(T, &counter, 1);
  - T = 0 is broadcast
  - T = 1..31 concrete target node ID
  - sendNow takes pointer to data and its length (&counter, 1B)
    - Busy waiting until send can be done (free channel check)
- rf12_sendWait(1);
  - Waits until a packet send is done
- Maximum length of payload data RF12_MAXDATA
  - 66 bytes, but don’t push it too close (unreliable)
  - Stay below 60

Sniffer.ino

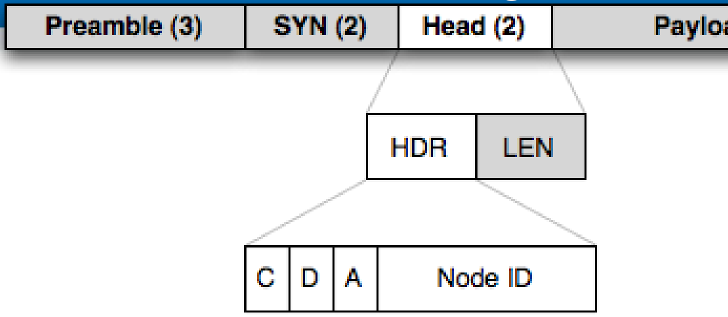
```
#include <Ports.h>
#include <RF12.h>

byte saveHdr, saveLen, saveData[RF12_MAXDATA];
word saveCrc;

void setup () {
  Serial.begin(57600);
  Serial.println("\n[sniffer] 868 MHz group 100");
  rf12_initialize(31, RF12_868MHZ, 100);
}
void printPacket(byte saveHdr, byte saveLen, byte saveData[RF12_MAXDATA]){
// … nice print of packet via Serial port, see full code at IS
}
void loop () {
  if (rf12_recvDone()) {
    // quickly save a copy of all volatile data
    saveLen = rf12_len;
    saveCrc = rf12_crc;
    saveHdr = rf12_hdr;
    if (saveLen <= sizeof(saveData)) { memcpy(saveData, (const void*) rf12_data, saveLen); }
    else { memset(saveData, 0xff, sizeof(saveData));}
    rf12_recvDone(); // release lock on info for next reception

    if (saveCrc != 0) {
      Serial.print("CRC error #");
      Serial.println(saveLen, DEC);
    } else { printPacket(saveHdr, saveLen, saveData);}
  }
}
```

Sniffer application

- Download sniffer code from IS (sniffer.ino)
  - File→New, Paste sniffer code
  - Compile and upload
- App listens for RF12 packets and prints it via Serial port
  - `rf12_initialize(31, RF12_868MHZ, 100);`
  - `rf12_recvDone()` – true if packet received
  - `rf12_recvDone()`
  - `rf12_len, rf12_crc, rf12_hdr, rf12_data`
    - Global variables set by radio module
  - Local copy of global variables (rf12_len → saveLen) made to:
    - Prevent overwrite by another packet
    - Enable radio module to start receiving next packet

Basic transmission: one hop

- Pair together with one other colleague
  - Write app that will blink LED X-times based on value inside received packet
- First node is beacon sending counter
  - Send is to particular second node (not broadcast to all)
  - Use different group (than 100)
- Second node – receiver blinking counter % 5
  - Use sniffer application as basic, change nodeID
  - `rf12_initialize(17, RF12_868MHZ, group);`
  - Don't forget to set LED output pin
- How far you can transmit? (try hall space)



Packet acknowledgements

- Send packets can frequently get lost (noise, collision) or missed by sender (performing other task)
  - How can be sender sure that the packet was delivered?
  - Sometimes does not care (broadcast or "UDP"-like transmission)
- If care, thank special message back from receiver can be expected/required (ACK)
  - Create packet for target node with bit flag that (A)CK = 1
  - Receiver reply with special ACK packet upon successful reception
    - Same header as received packet, but with (A)CK bit = 0 and (C)TL=1

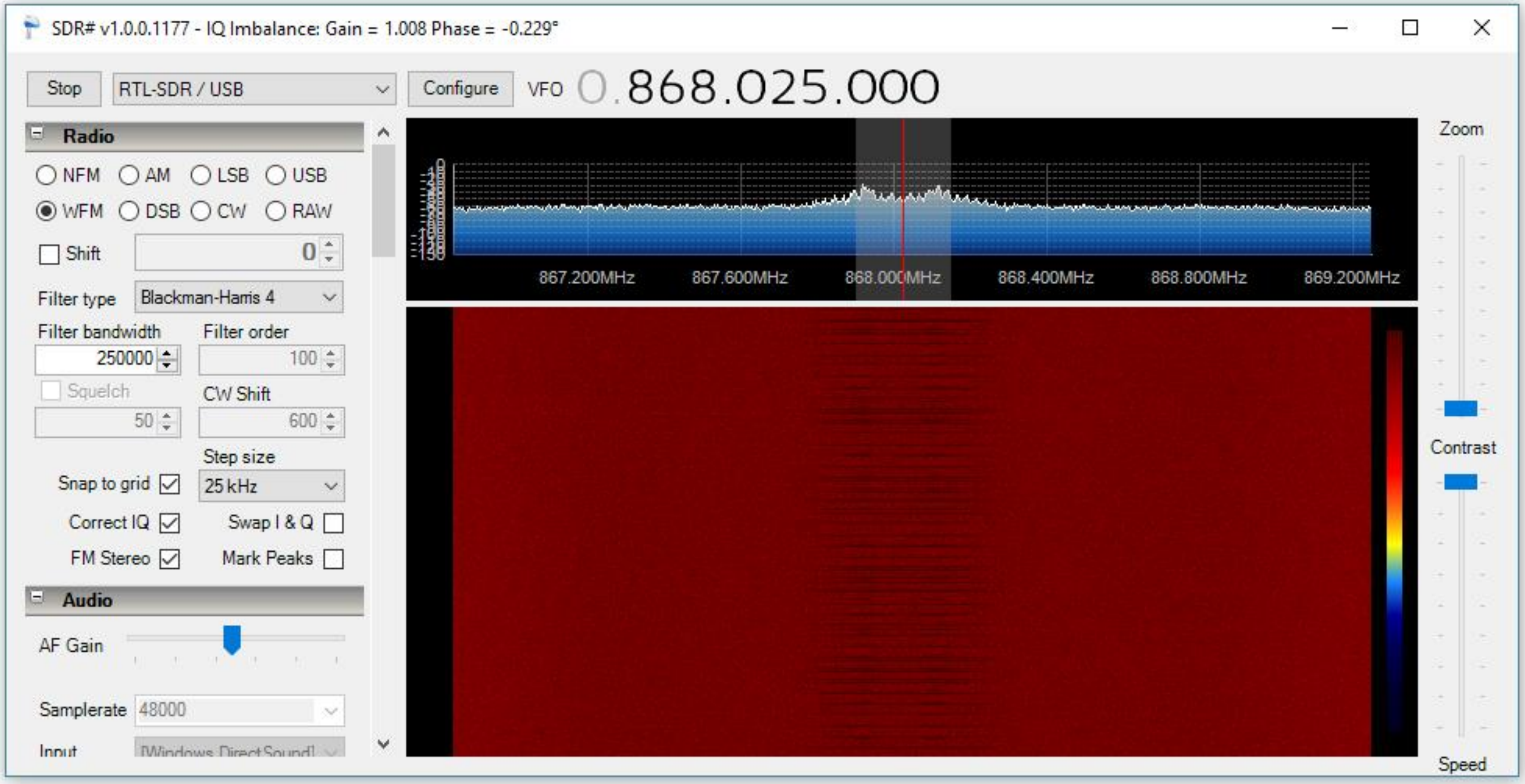
```
byte createHeader(boolean requireACK, byte destID){
  byte header = requireACK ? RF12_HDR_ACK : 0;
  header |= RF12_HDR_DST | destID;
  return header;
}
```

```
if(RF12_WANTS_ACK){
  rf12_sendStart(RF12_ACK_REPLY,0,0);
}
```

Non-discriminative jammer

- Occupies/distorts whole channel
  - Sending packets all the time without waiting for clear channel
  - No need to send correctly formatted packets or use same type of radio
- How to detect non-discriminative jammer?
  - Abnormally high number of lost packets
  - Abnormally high send waiting time (clear channel)
  - Abnormally low RSSI
  - Detection by whole channel monitoring (e.g., SDR)

Non-discriminative jammer : SDR



SDR# v1.0.0.1177 - IQ Imbalance: Gain = 1.008 Phase = -0.229°

RTL-SDR / USB

VFO 0.868.025.000

Radio

NFM AM LSB USB
[✓ WFM] DSB CW RAW

Shift 0

Filter type: Blackman-Harris 4

Filter bandwidth: 250000

Filter order: 100

Squelch: 50

CW Shift: 600

Step size: 25 kHz

Snap to grid ✓

Correct IQ ✓

Swap I & Q ☐

FM Stereo ✓

Mark Peaks ☐

Audio

AF Gain

Samplerate: 48000

Input: [Windows DirectSound]