PA197 Secure network design

Basic wireless networking

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

CS
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
- Plug in JeeNode
- Select COM port
  - Can be assigned to different values
  - Try other ports if selected doe not work
- Board: Arduino Mini
- Processor: ATMega328

Test 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

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

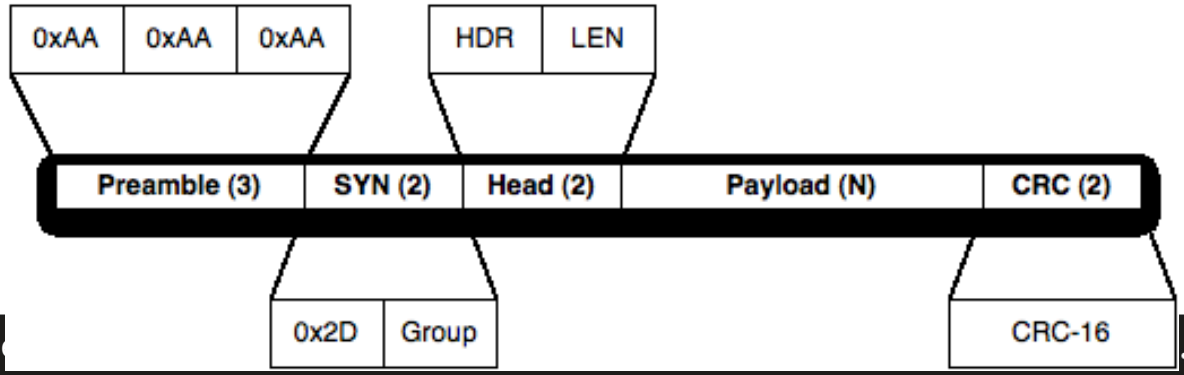
- Check if you have proper board and processor
  - Arduino Mini, ATMega328
- Don’t have serial monitor running if going to upload new app
- Try to re-plug jeenode
- Try to plug into different USB port
- Try to restart Arduino IDE
- Check if you have same serial port speed on arduino and port monitor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

File→Examples … →DigitalReadSerial

- Pre-prepared code that prints fixed message
  - Counter, Keep speed at 9600
- Run Serial monitor (will automatically restart Arduino)
  - Observe data print out
- Modify to print out loop counter
  - 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

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
  - Examples are now available: Examples → jeelib-master



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);
}
```

Basic beacon application

- Select File → Examples → 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

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
- Application listen 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 made to
    - Prevent overwrite by another packet
    - Enable radio module to start receiving next packet

Test1 + sniffer

- Collaborate two together
  - First node runs test1
  - Second node runs sniffer
- Make sure that same group is used
- Data transferred by first node should be captured sniffer

Basic transmission: one hop, two hops

- Pair together with one colleague
  - Write app that will blink LED X-times based on value inside received packet
- Pair together with two colleagues
  - Same as previous
  - Use one intermediate node (fixed routing topology)
    - Sender→Transmitter (receive, send)→Receiver (blink)
- If sending to specific node, ACK may be required
  - http://jeelabs.org/2010/12/11/rf12-acknowledgements/

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

Homework 11 – Network sniffing

- Create sniffer node that will capture as many packets as possible from single network run
  - 10 minutes transmission, 5 minutes silence (then repeat)
  - Try to capture packets from multiple runs and compare
- Produce short (1xA4) text description of solution
  - How network properties were found
  - How was traffic logged
  - How were packets analyzed
- Submit before: 12.5. 6am (full number of points)
  - Every additional started day (24h) means 1.5 points penalization

Example output you should submit

- #440#oes. They got the p
- #458#arades and fame and
- #464# love of the world.