Lab 3

Protocol Layering with the RFM12B-S2 Radio Module

The aim of this third lab is to incorporate the existing functionality that you've developed into a basic layered model (i.e. basic PHYsical and Data Link Layers), using the rfm12 wireless transceiver with the Il Matto board.

Schedule

Preparation time : n/a

Lab time : 3 hours

Items provided

Tools : n/a

Components : RFM12B-S2 Radio Module (1 each)

Equipment : Oscilloscope, Logic Analyser

Software : n/a

Items to bring

Essentials. A full list is available on the Laboratory website at https://secure.ecs.soton.ac.uk/notes/ellabs/databook/essentials/

Revision History

October 26, 2015	Domenico Balsamo and Geoff Merrett	New lab for 2015/16

1 Aims, Learning Outcomes and Outline

These lab exercises will provide an introduction to computer networks and wireless communication using the Il Matto development board. In particular, in the second lab you became familiar with the network functionally without using any structured layering.

In this third lab, you will take the existing functionality that you developed in the previous labs (i.e. the hardware API and 0-persistent CSMA algorithm) and encapsulate them into a basic layered model (i.e. separate PHYsical and Data Link Layers, with appropriate interfaces and service definitions)

---

2 Preparation

None required.

---

3 Laboratory work

You should work in pairs for this lab exercise. Each pair should ensure that they have at least one working Il Matto between them, and that at least one person is familiar with the Il Matto and its programming.

Your objective for this lab is to incorporate all of the existing functionality that you developed in the previous two labs (i.e. frequencies to channels, 0-persistent CSMA, etc) into a basic layered model. This will involve encapsulating the functionality into separate layers for the PHYsical layer and Data Link Layer, and defining and complying with interfaces and service definitions between layers.

Layer 1: Physical

During the first lab, we have seen that physical layer functionality is defined in the `rfm12_config.h` file and they can be set within the function `rfm12_init()`.

In this lab, you have to structure the Physical basic functionalities into a proper layer with clearly and well-defined interface and service primitives.

As you already know, the physical layer is responsible for the direct control of RFM12, and the following properties are specified by rfm12 transceiver: band, FSK modulation, sync detection on 0x2D 0xD4. However, this still needs to be implemented and defined in a structured way:

- Baud rate (default: fixed channel baud rate assignment)
- Assignment of frequencies to channels
- Bandwidth of the channels
- Management of the transmission power/reception sensitivity.

A possible way to do this is by defining new files (`physical_layer.h` and `physical_layer.c`) with the APIs needed for physical layer:

APIs:

```
/* Layer 1: Physical */

void RFM12_PHY_modeTX(void);

void RFM12_PHY_modeRX(void);

void RFM12_setBaudrate(uint32_t baud);

void RFM12_setCenterFrequency(uint16_t freq);

void RFM12_setReceiverBandwidth(RFM12_VDI_t vdi, RFM12_RxBW_t bandwidth,
RFM12_GAIN_t gain, RFM12_RSSI_t drssi);

void RFM12_setPowerManagement(bool enRX, bool enBB, bool startTX, bool
enSynth, bool enOSC, bool enBat, bool enWkT, bool clkOff);

void RFM12_setTransmitPower(RFM12_Power_t power, RFM12_TxDev_t deviation);
```

Layer 2: Physical

During the second lab we have seen how to use a collision-avoiding approach (i.e. 0-persistent CSMA), and how it can be implemented using the function `rfm12_tick();`.

Again, you have to structure the MAC basic functionalities into a properly layer with clearly and well-defined interface. This has to be done in a well-structured and coded with clear and distinct interfaces and service definitions between the Physical and Data Link layers.

Moreover, because you have followed the principles of protocol layering, you should be able to implement an alternative MAC layer (that can easily be swapped with your existing mac_layer.c file, because it has the same interfaces and service definitions) which implements a 1-persistent CSMA algorithm (or a *p*-persistent CSMA algorithm – up to you). You should be able to demonstrate this working (i.e. the two .c files being swapped and still working). In order to indicate which CSMA algorithm is being used, you should include an additional header byte in your MAC frames:

	CSMA	16 bytes of data
Bits	8	8*16
Value	0 = 0-persistent 1 = 1-persistent 2 = n-persistent	Payload (data)

---

4 Resources

You can find more information about RFM12B-S2 on-line:

- https://www.sparkfun.com/products/12031

It is very useful to use/read the following documents:

- RFM12B-S2 Datasheet:
  - http://cdn.sparkfun.com/datasheets/Wireless/General/RFM12B.pdf

- Programming guide:
  - https://www.sparkfun.com/datasheets/Wireless/General/RF12B_code.pdf

- RFM12B IC Datasheet:
  - https://www.sparkfun.com/datasheets/Wireless/General/RF12B-IC.pdf

- RFM12 Library for Atmel AVR uC's (ATMega8):
  - http://www.das-labor.org/wiki/RFM12_library/en