Dual-motes for the Zolertia Firefly: a system for WSN evaluation

The dual-motes are a system consisting of two wireles motes, the Zolertia Firefly, along with some necessary peripherals. The goal is the development of a hardware platform for the evaluation of experimental communication stacks using the IEEE 802.15.4 PHY layer with possible expansion to LoRa PHY.

The dual-motes consist of two wireless motes with parallel communication between the two. The goal is to run an experimental network stack on one mote, and several parameters will be evaluated so that a performance analyses can be done. The other mote monitors the first and collects data through the parallel communication and performs current measurements. The Monitoring Motes (MM) form a stable network with a sink mote, using a network stack with protocols that provide the most stable network possible.
Connections are also provided to be able to program the observed mote (OM) through the MM. The goal is to also enable over-the-air-programming, but at the time of writing this is still in development. Furthermore the dual-motes have a variable power supply, with output voltage ranging between 1.6V and 3.8V, and the input voltage range from 3V to 15V, with the possibility of using a 50V DC voltage regulator as well. The Input range was chosen to have multiple option to provide power, ranging from batteries to DC power lines at for example 48V. The variable output provides the opportunity to study the influence of supply voltage on CPU performance and power consumption. It needs to be set to 2V minimum to power the device.

Parallel communication

The dual-motes provide 15 bits of parallel communication between the two motes, through GPIO pins. One bit should reserved for raising an interupt on the MM when the OM sends data. The connection scheme is (MM -> OM):

- PA2 -> PC5
- PA3 -> PA3
- PA6 -> PA6
- PA7 -> PA7
- PB0 -> PB0
- PB4 -> PB4
- PC0 -> PC0
- PC1 -> PC1
- PC2 -> PC2
- PC3 -> PC3
- PC4 -> PA2
- PC5 -> PA4
- PC6 -> PA5
- PD0 -> PD0

- PD2 -> PD2

Remote programing of the observed mote

Programming the OM through the MM happens over an SPI connection. The SPI chip select pin is PA3 on both motes. The possibility to reset the OM by the MM is provided on pin PA5. As this uses some of the GPIO pins mentioned in the previous step, less bits are available for communication between OM and MM during normal operation if the devices are configured for remote programming.

This feature requires a binary file of the desired firmware, which preferably has been truncated of trailing 0xFF characters. A script is provided that performs this truncation.

The code for the MM to program the OM is available under the name zoul_dualmote2. Code using both this feature and the monitoring functionality is provided under the name tftp-monitor-upload-sender.

Current measurements

The MM can perform current measurements with shunt sensing. For this is uses following circuitry, and performs measurements on pin PA4/ADC2.

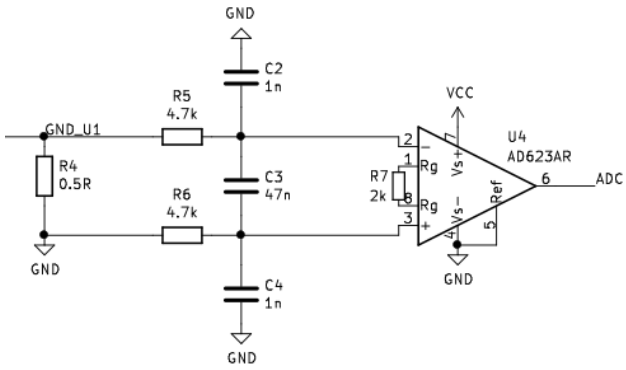

Figure 1: Current-sensing circuitry

The voltage drop over the 0.5 Ω shunt resistor is amplified by a facor $(1 + \frac{100k\Omega}{R_G})$, in this case 51. The 1nF resistors act as a low-pass filter to remove the high frequency component due to the clock speed from the CPU. Measurements at this accurate on the time axis are unnecessary.

Measurements are performed low-side (between the ground connection from the load and the actual ground) to avoid driving the amplifier to saturation into.

ng the dual-motes

Please look at the code provided. This is an example running a very simple p2p nullnet network on the OMs. Code is provided for:

- An OM sender
- An OM receiver
- A sink that is part of the OM network, that periodically receives energest data from the OMs
- MMs (monitor-sender)
- A sink that is part of the MM network

Several parameters are defined for each in `project-conf.h`. Important in the configuration is 'UART_CONF_ENABLE'. The UART communication over USB interferes with the parallel communication between OM and MM. As such it needs to be set to 0 for the dual-motes. On the sinks it has to be enabled, since otherwise no output from the sink is received on the host machine.

Delay measurements using the system

As mentioned before, the MM is notified whenever the OT sends or receives a message and some bits of additional info are also communicated to the MM (often data identifying the packet sent/received). At that moment the MM sends a packet to the data sink containing the following info:

- the data from the OM (if usefull)
- energy measurements
- time at which the MM mote was notified
- time at which the previous packet was sent to the sink

This data allows us to determine the delay between transmission and reception of a packet. We can assume that notifying the MM is near instantaneous, and the same for sender and receiver. There is however still a variable delay between notifying the MM and reception of the packet by the sink.

Here the parameters are:

- t_{sa} is the time sender's application sends the packet. In this case the moment the MM is notified.
- t_{sr} is the time the sender's radio transmits the packet. This can be variable due to collisions and retransmissions.
- t_{rr} is the time the receiver's radio receivers the packet. Here the receiver is the sink.

- t_{ra} is the time the receiver's applications receives the packet.

Keep in mind that the sender and receiver use an independant clock, meaning that the local timestamps are not comparable. These timestamps are recorded both at the sending and receiving OM. The latency between transmission and receptions by the radios is also assumed 0, meaning that $t_{sr} = t_{rr}$, and the

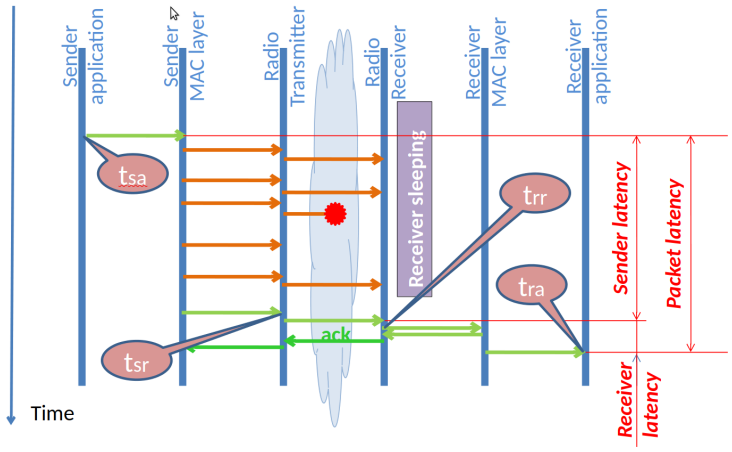

Figure 2: Delay between sending packet from MM to sink

packet latency is $(t_{sr} - t_{sa}) + (t_{ra} - t_{rr})$.

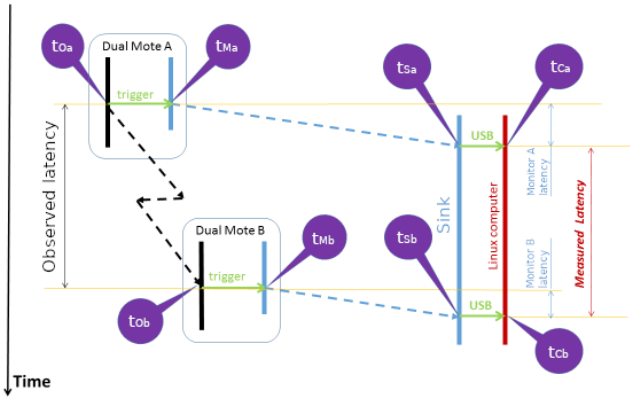

Figure 3: Timing of latency between two dual motes

Say a packet is transmitted by an OM at a time t_{Oa}, which is the same as t_{Ma}, and received by another OM at $t_{Ob} = t_{Mb}$. The MM's then transmit a packet to the sink, each with their own delay as mentioned above. At the sink we have the latency between the reception of both packets from the MM's and the internal latency values for both packets as well, meaning we can calculate the latency between the OM's: $Observed\ latency = Measured\ latency + (Monitor\ A\ latency - Monitor\ B\ latency)$.

Translated to the terms used in the figures this is $t_{Ob} - t_{Oa} = (t_{Cb} - t_{Ca}) - (t_{Sb} - t_{Mb}) + (t_{Sa} - t_{Ma})$.

The delays between MM's and sink $(t_{Sx} - t_{Mx})$ is the delay disscussed in the step above $(t_{sr} - t_{sa}) + (t_{ra} - t_{rr})$.

In the implementation it is not possible to obtain the timestamp of the packet currently being transmitted, only of the previously received or sent packet. Keep this in mind when performing calculations.