PASS Simulation Framework

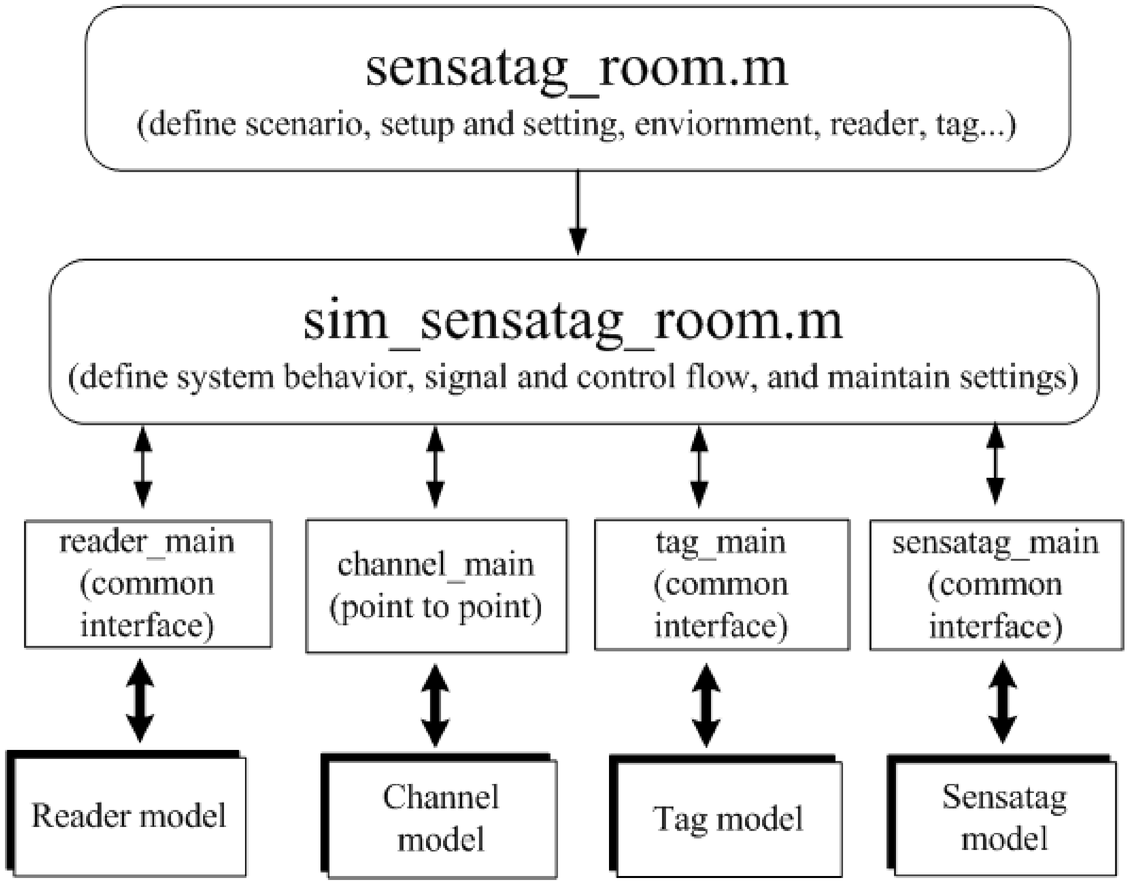
Jing Wang
Identigo RFID Research Group
uOttawa

March 1st, 2016

Overview: what is PASS

- PASS (**P**roximity-detection based **A**ugmented RFID **S**ystem **S**imulator) is a Matlab based system-level simulator designed for UHF RFID technology.
- PASS is designed for studying the recently developed augmented RFID system with sensatag.
- PASS is an open source software. You can modify and redistribute it. PASS is distributed in the hope that it is useful, but without any warranty.
- PASS simulation framework is developed based on PARIS simulator by Dr. Arnitz, Graz Uni. Of Tech., Austria.

Simulator Structure

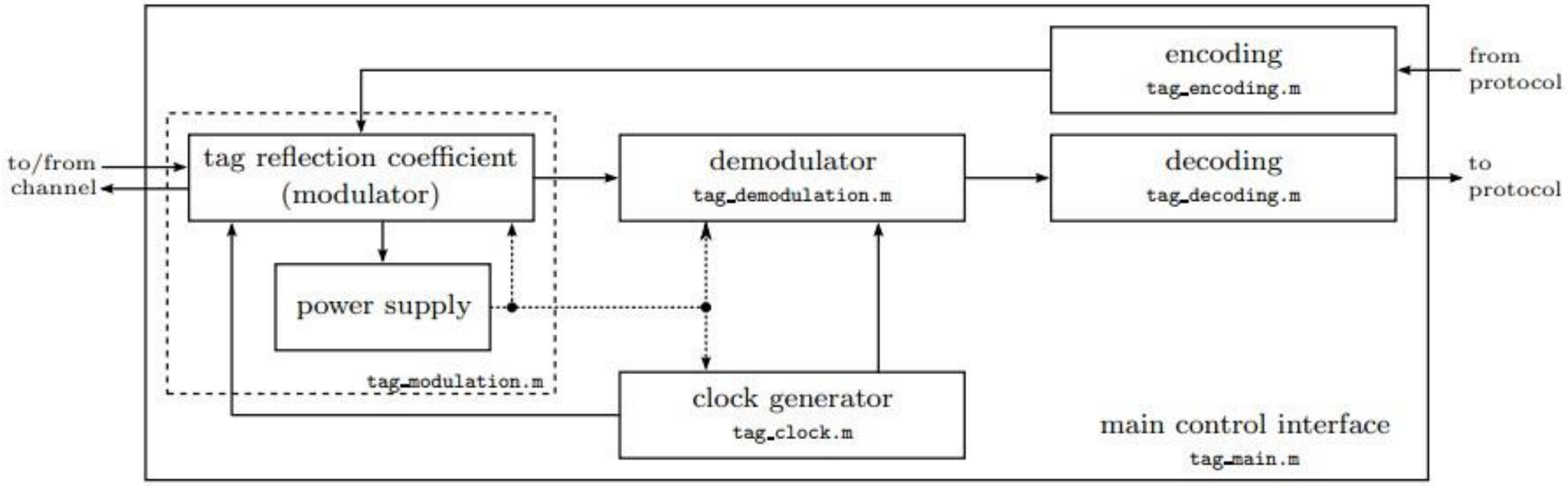

sensatag_room.m
(define scenario, setup and setting, enviornment, reader, tag...)
sim_sensatag_room.m
(define system behavior, signal and control flow, and maintain settings)
reader_main (common interface)
channel_main (point to point)
tag_main (common interface)
sensatag_main (common interface)
Reader model
Channel model
Tag model
Sensatag model

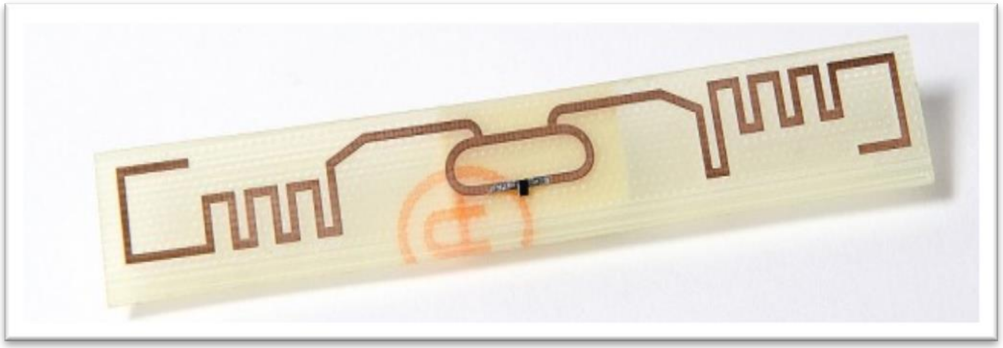
Simulator Structure

- The simulator is organized in a hierarchical fashion.
- The user script (*sensatag_room.m*) describes the simulation setup, which is scenario-dependent.
- The type of simulation, signal and control flow are defined in the behavior description script (*sim_sensatag_room.m*).
- The basic main functions control the behavior of reader, channel, tag and sensatag through underlying model and calculation.

PASS: Tag model

- The simulation is based on the model of an NXP UCODE G2XM chip soldered onto an NXP UCODE general purpose reference antenna.

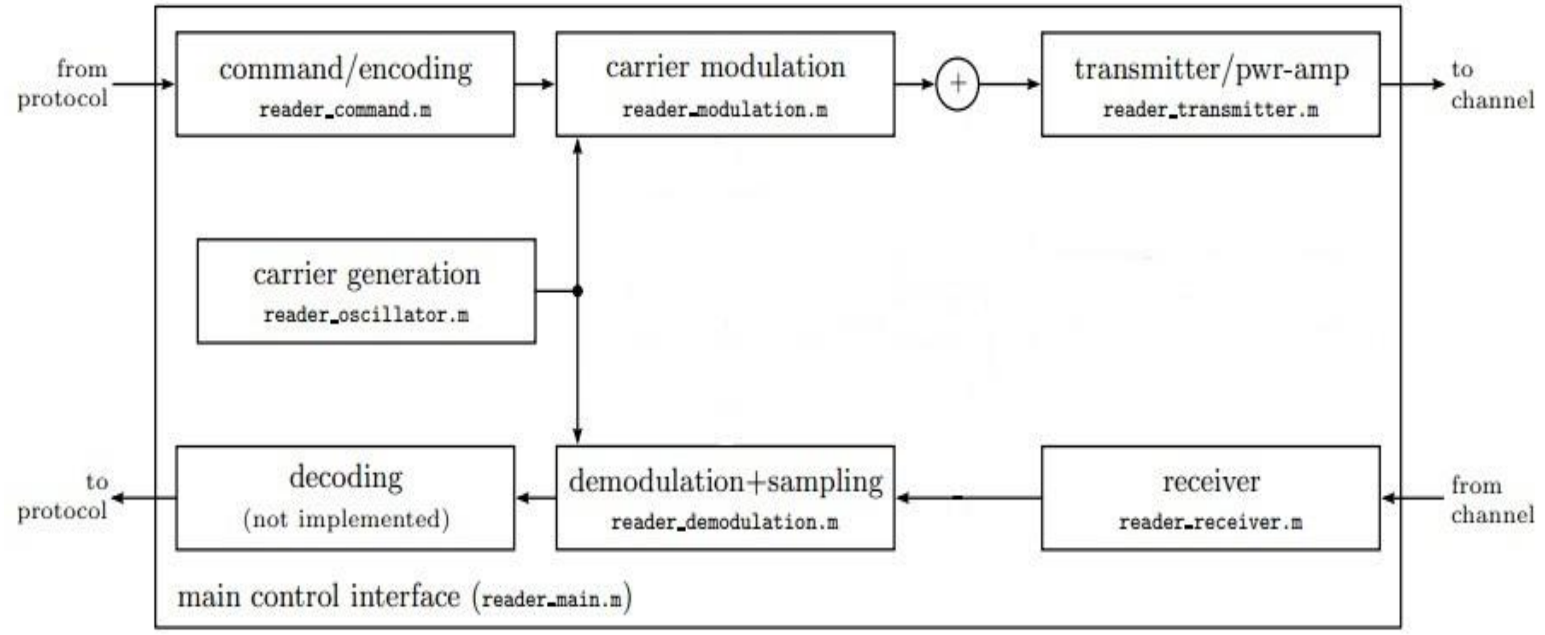




PASS: Tag model

- The most important part of the tag is the tag reflection. The reflection coefficient of a passive UHF RFID tag is highly nonlinear, frequency-dependent, and time-variant during tag modulation, which is provided by (*tag_modulation.m)* in the model. This function is by far the most complex tag model, based on physical characteristic and a large filterbank structure.
- On the receive path, the part of the incoming signal that is neither reflected nor lost in the modulator circuitry or parasitic is sent to the demodulator (*tag_demodulation.m*), where it is amplitude-demodulated and resampled to the tag clock rate.
- Decoding (*tag_decoding.m*) is implemented according to EPCglobal Class-1 Gen-2. The transmit path consists of the encoding block (*tag_encoding.m*), which sends the encoded data directly to the modulator.

PASS: reader model

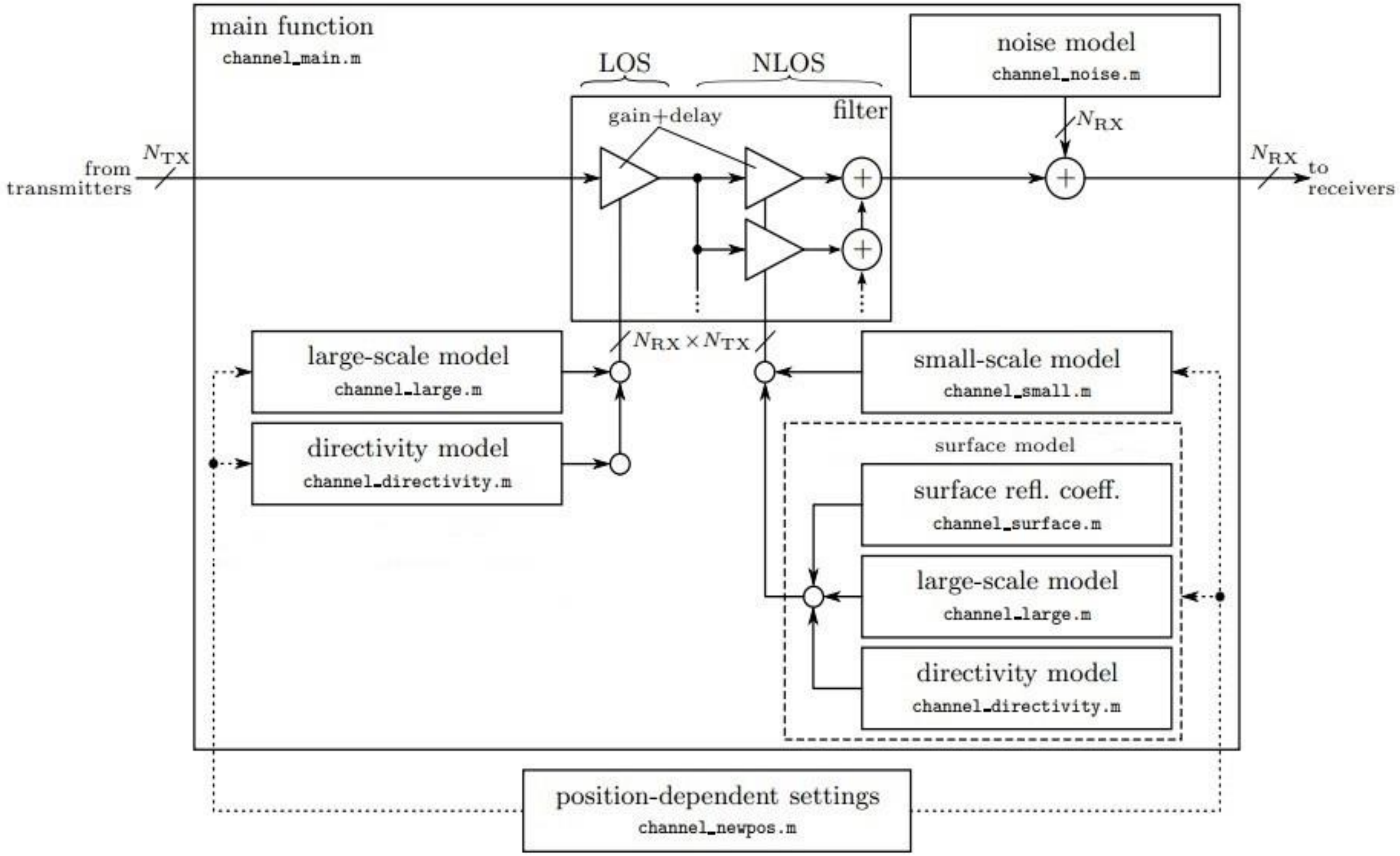


- The main reader function, (*reader_main.m*), provides the user with a simple-to-use interpreter interface. The reader structure implements a standard communication transceiver. A command is first encoded and then sent to the modulator which in turn modulates the carrier signal provided by the carrier generation block. This signal is sent over the transmitter, after which it enters the channel. The receiver path consists of the receiver, an I/Q demodulator, sampling and quantization, and the decoding block.

PASS: reader model

- For transmitting path command generation (*reader_command.m*) and modulation (*reader_modulation.m*) are implemented according to the EPCglobal Class-1 Gen-2 protocol. The created modulation waveform, which uses cosine rolloffs for band limitation, is subsequently modulated onto the reader carrier generated by (*reader_oscillator.m)*.
- Carrier generation supports cosine, sine, and complex exponential carriers with arbitrary frequency, phase and amplitude noise, as well as additive white Gaussian noise.
- The transmitter block (*reader_transmitter.m*) amplifies and filters the transmitted signal, emulating a nonlinear power amplifier with subsequent bandpass filtering. On the receive side, the receiver performs bandpass filtering and separate in-phase (I) and quadrature-phase (Q) signal.
- Demodulation is performed by (*reader_demodulation.m)* symmetrically on I and Q channels using a complex exponential carrier generated by (*reader_oscillator.m)*. The baseband signal is then filtered, resampled to the reader sampling frequency, and linearly quantized.

PASS: channel model



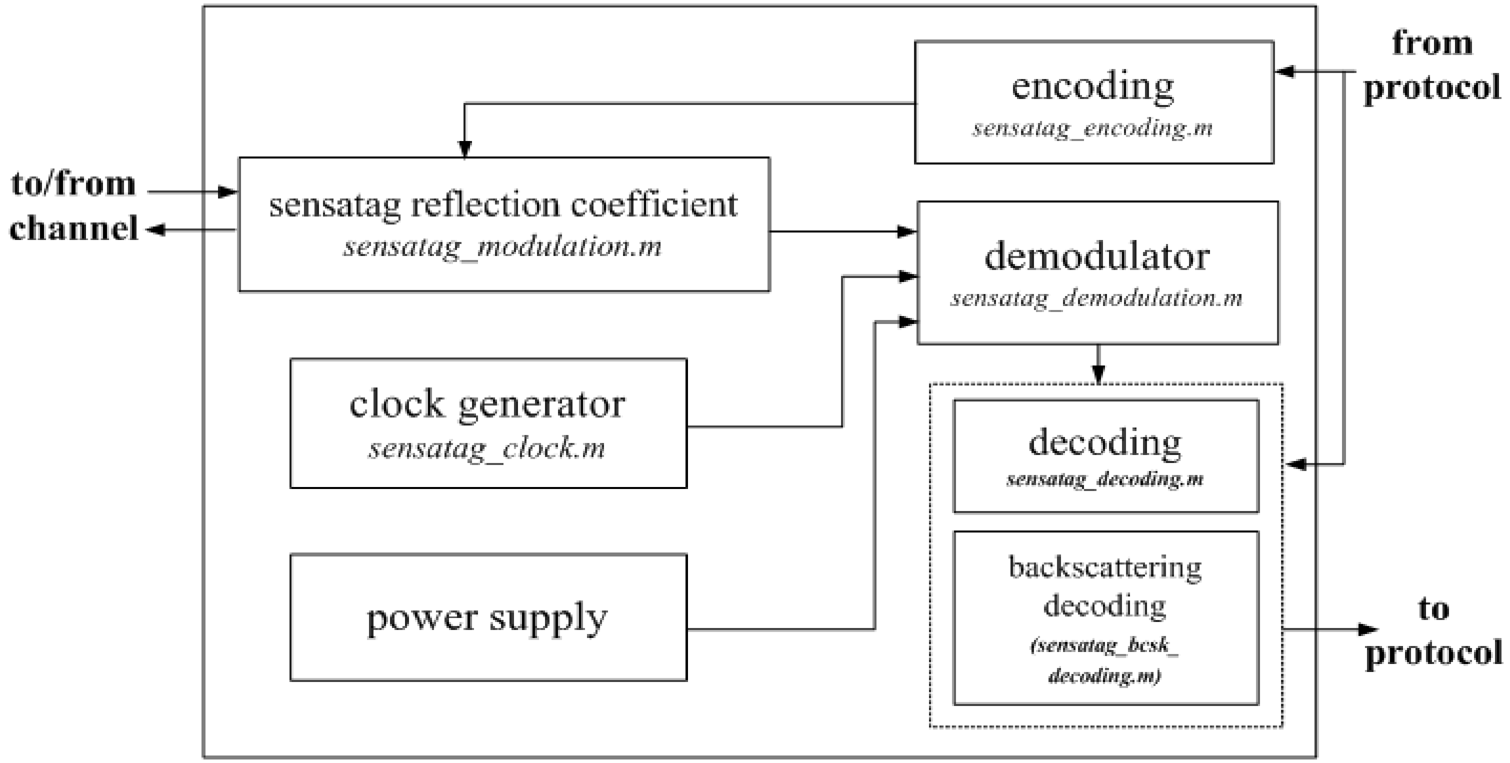
PASS: channel model

- Unlike the main reader and tag function, (*channel_main.m)* does not provide an interpreter interface, but merely calculates the output signals for a given set of input signals, subject to the channel settings. The function is designed to handle an arbitrary number of transmitters (inputs) and receivers (output).
- The channel between a transmitter and a receiver is formed by piecewise assembly of the channel impulse response. **The LOS component is created by the combined results of large-scale model (gain and delay), and directivity model (antenna gain pattern).**
- Large-scale and directivity gains are calculated by (*channel_large.m)* and (*channel_directivity.m)* respectively.

PASS: channel model

- **The NLOS part of the CIR is created by a stochastic small-scale model plus deterministic reflection in surfaces.** The latter include the regular reflection (reflection involving one surface) and specular inter-reflection (reflection involving more than one surface). Each of them employs large-scale and directivity model, as well as a surface reflection coefficient.
- The stochastic NLOS components created by (*channel_small.m)* are scaled by the large-scale LOS gain and by the directivity gain.

PASS: sensatag model



PASS: sensatag model

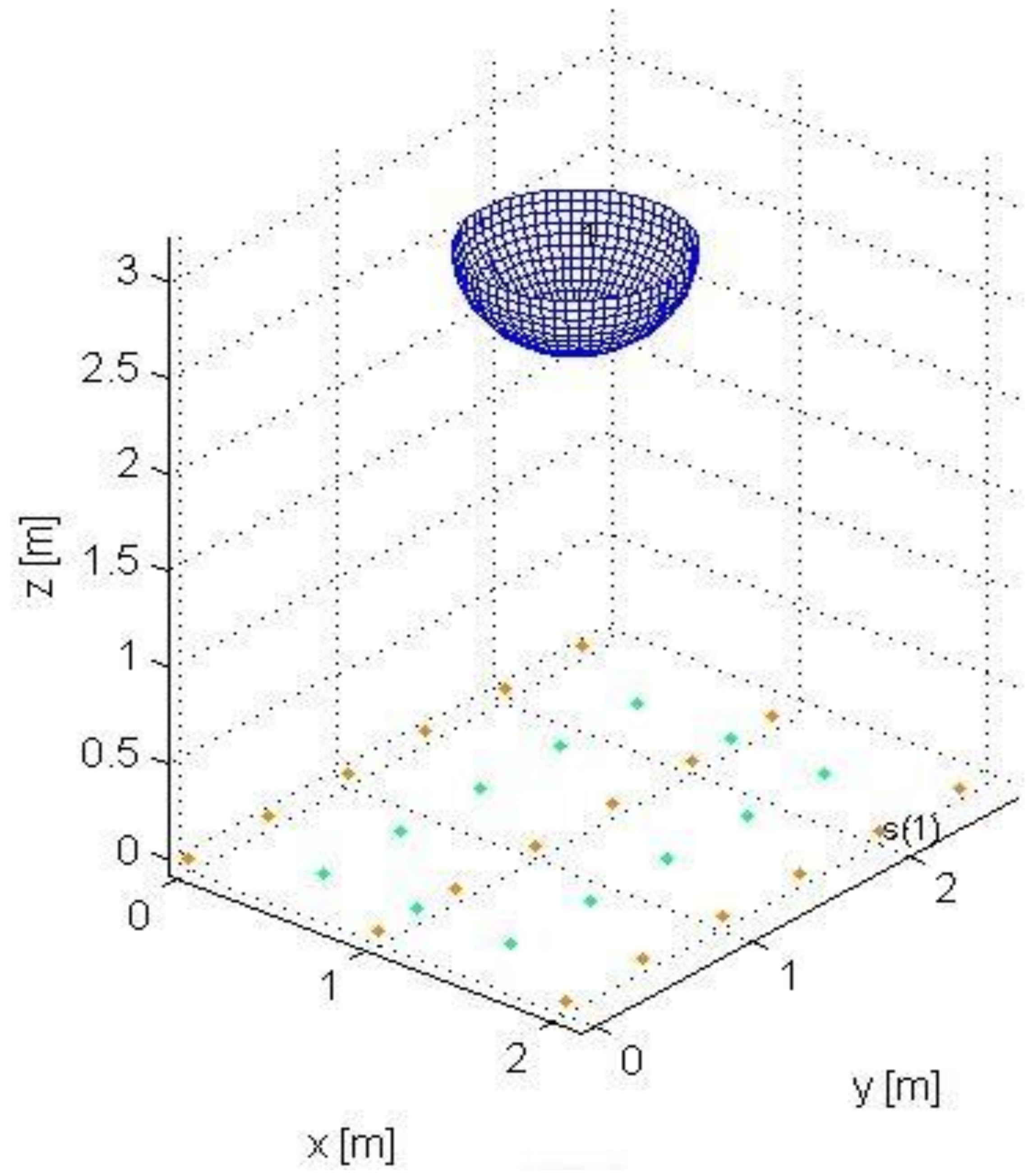
- The refection coefficient of sensatag is highly nonlinear, frequency-dependent and time variant, which is provided by (*sensatag_modulation.m)*. On the receive path.
- The part of the incoming signal that is neither reflected nor lost in the modulator circuitry or parasitic is sent to the demodulator (*sensatag_demodulation.m).*
- It is amplitude-demodulated by the envelop detector and resampled to the sensatag clock rate. Decoding (*sensatag_decoding.m)* and backscattering decoding (*sensatag_bcsk_decoding.m)* is implemented according to ISO 18000-6C, which is controlled by sensatag protocol. The transmit path consists of the encoding block (*sensatag_encoding.m)*, which sends the encoded data directly to the modulator.

Simulation Setup: *sensatag_room.m*

The main startup scripts define scenario and initialize the simulator.

1. Environment parameters
   Mainly, channel related parameters: path-loss factor, noise power density, small scale fading parameters (K_{LOS},τ_{RMS}).
2. Surface configuration
   Position of surface, reflection parameters of the surface. (to establish large scale fading).
3. Device configuration
   Initialization of reader, tag and sensatag, the trajectory of reader, tag and sensatag.

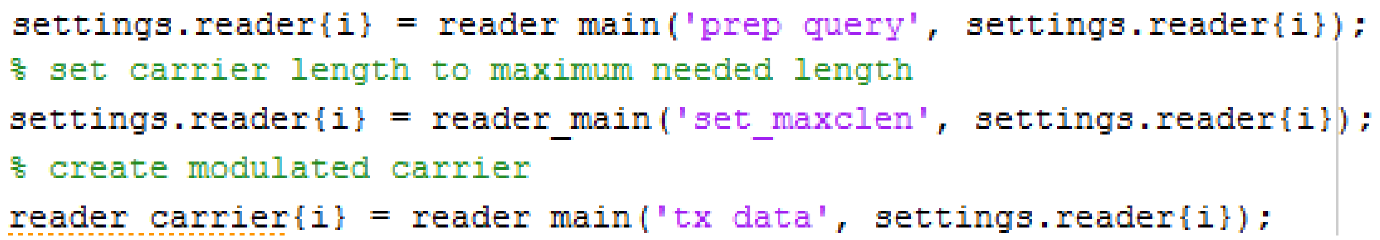
SIMULATION SETUP (VTX and surfaces: darker = higher gain)



Behavioral description script: *sim_sensatag_room.m*

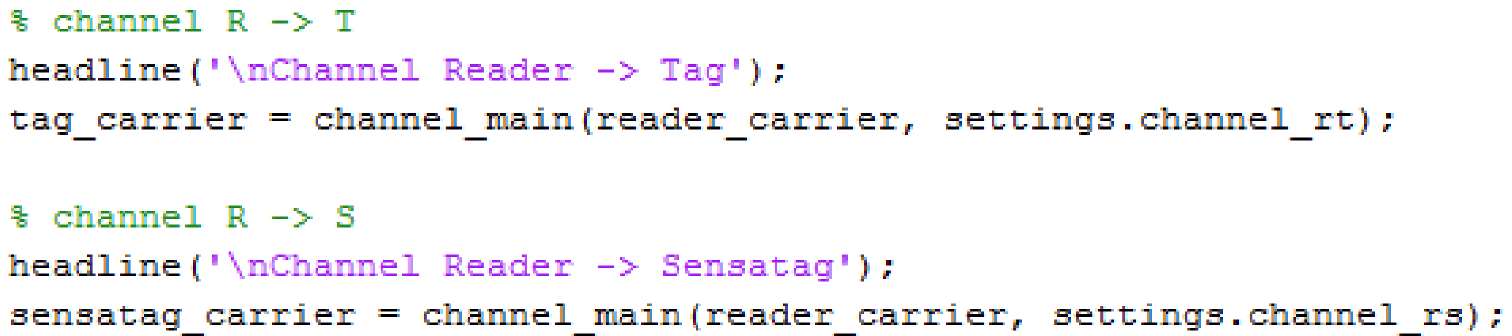
1. Setups: load settings from *sensatag_room.m*, establish point to point channel (reader <-> tag, reader<->sensatag, tag->sensatag);
2. Communication procedure

- Downlink: query command (reader->(tag, sensatag))



```
settings.reader{i} = reader_main('prep_query', settings.reader{i});
% set carrier length to maximum needed length
settings.reader{i} = reader_main('set_maxclen', settings.reader{i});
% create modulated carrier
reader_carrier{i} = reader_main('tx_data', settings.reader{i});
```

- Downlink: through the channel (reader->(tag, sensatag))

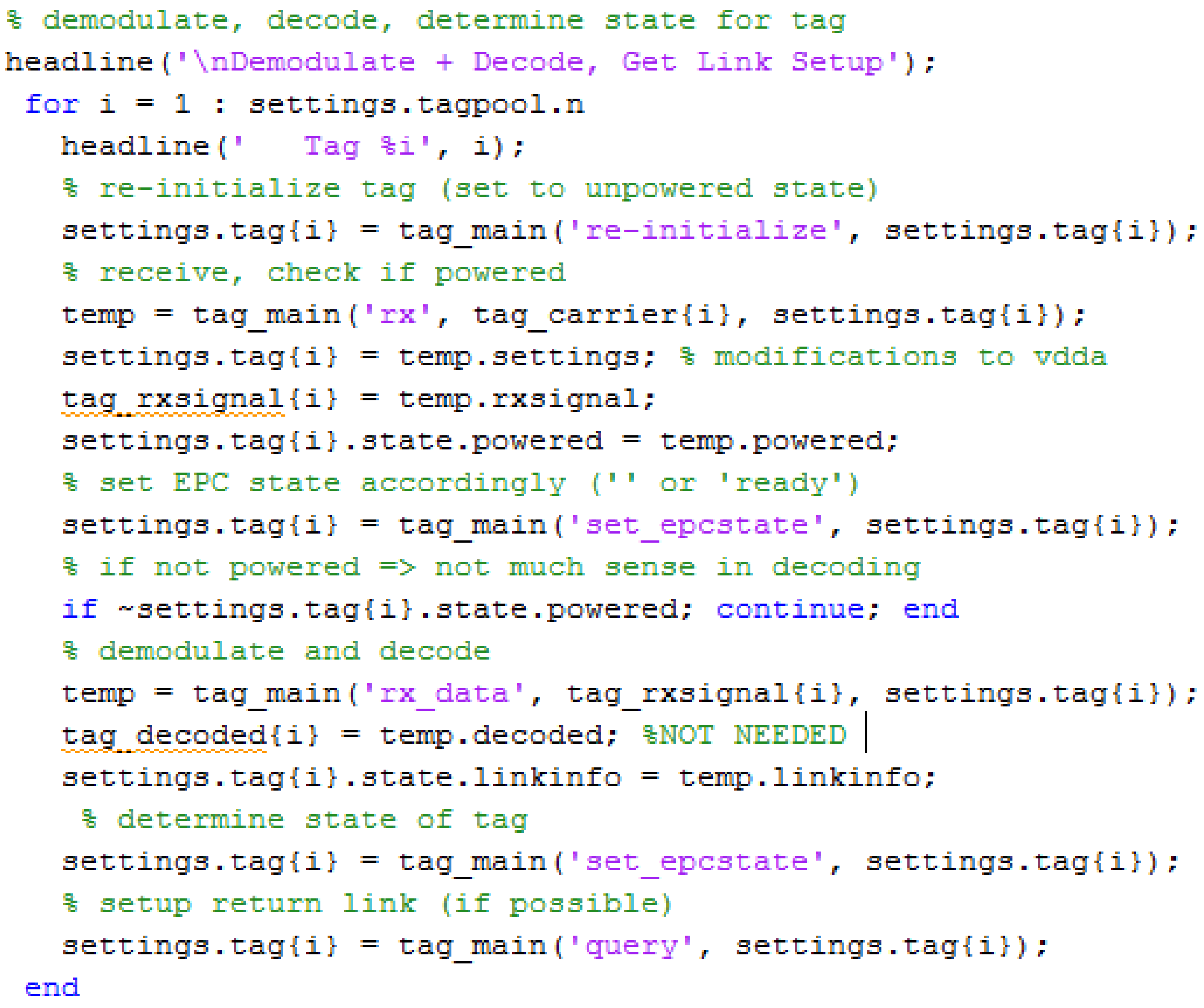


```
% channel R -> T
headline('\nChannel Reader -> Tag');
tag_carrier = channel_main(reader_carrier, settings.channel_rt);

% channel R -> S
headline('\nChannel Reader -> Sensatag');
sensatag_carrier = channel_main(reader_carrier, settings.channel_rs);
```

Behavioral description script: *sim_sensatag_room.m*

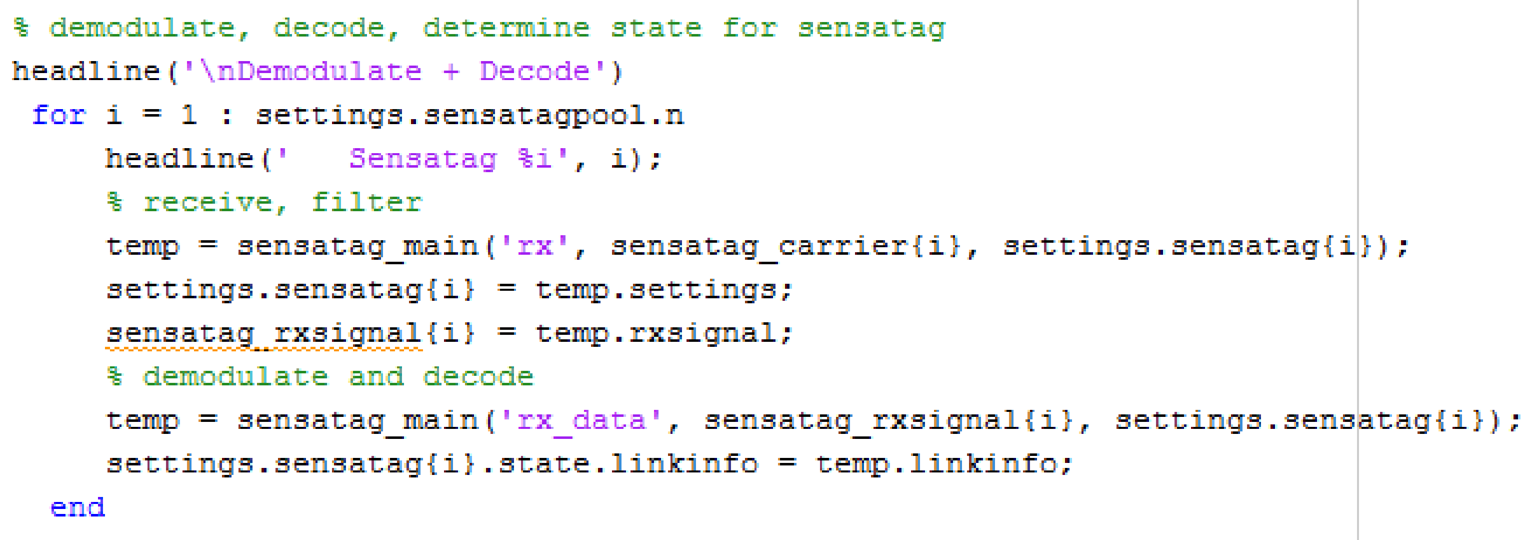
- Downlink: tag



```
% demodulate, decode, determine state for tag
headline('\nDemodulate + Decode, Get Link Setup');
 for i = 1 : settings.tagpool.n
   headline('   Tag %i', i);
   % re-initialize tag (set to unpowered state)
   settings.tag{i} = tag_main('re-initialize', settings.tag{i});
   % receive, check if powered
   temp = tag_main('rx', tag_carrier{i}, settings.tag{i});
   settings.tag{i} = temp.settings; % modifications to vdda
   tag_rxsignal{i} = temp.rxsignal;
   settings.tag{i}.state.powered = temp.powered;
   % set EPC state accordingly ('' or 'ready')
   settings.tag{i} = tag_main('set_epcstate', settings.tag{i});
   % if not powered => not much sense in decoding
   if ~settings.tag{i}.state.powered; continue; end
   % demodulate and decode
   temp = tag_main('rx_data', tag_rxsignal{i}, settings.tag{i});
   tag_decoded{i} = temp.decoded; %NOT NEEDED
   settings.tag{i}.state.linkinfo = temp.linkinfo;
    % determine state of tag
   settings.tag{i} = tag_main('set_epcstate', settings.tag{i});
   % setup return link (if possible)
   settings.tag{i} = tag_main('query', settings.tag{i});
 end
```

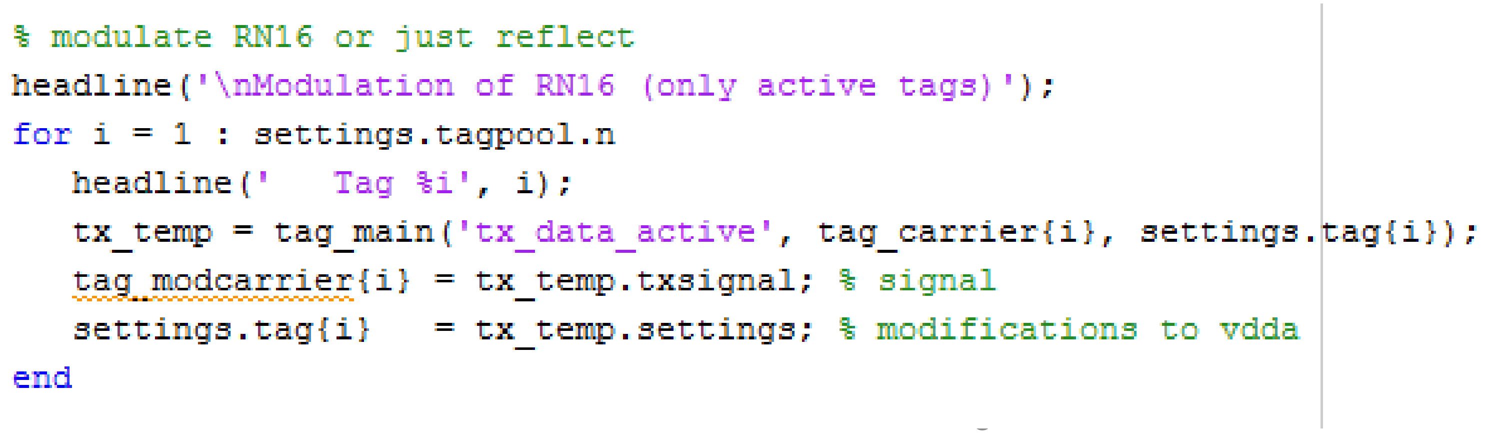
Behavioral description script: *sim_sensatag_room.m*

- Downlink: sensatag



```
% demodulate, decode, determine state for sensatag
headline('\nDemodulate + Decode')
 for i = 1 : settings.sensatagpool.n
     headline('    Sensatag %i', i);
     % receive, filter
     temp = sensatag_main('rx', sensatag_carrier{i}, settings.sensatag{i});
     settings.sensatag{i} = temp.settings;
     sensatag_rxsignal{i} = temp.rxsignal;
     % demodulate and decode
     temp = sensatag_main('rx_data', sensatag_rxsignal{i}, settings.sensatag{i});
     settings.sensatag{i}.state.linkinfo = temp.linkinfo;
  end
```

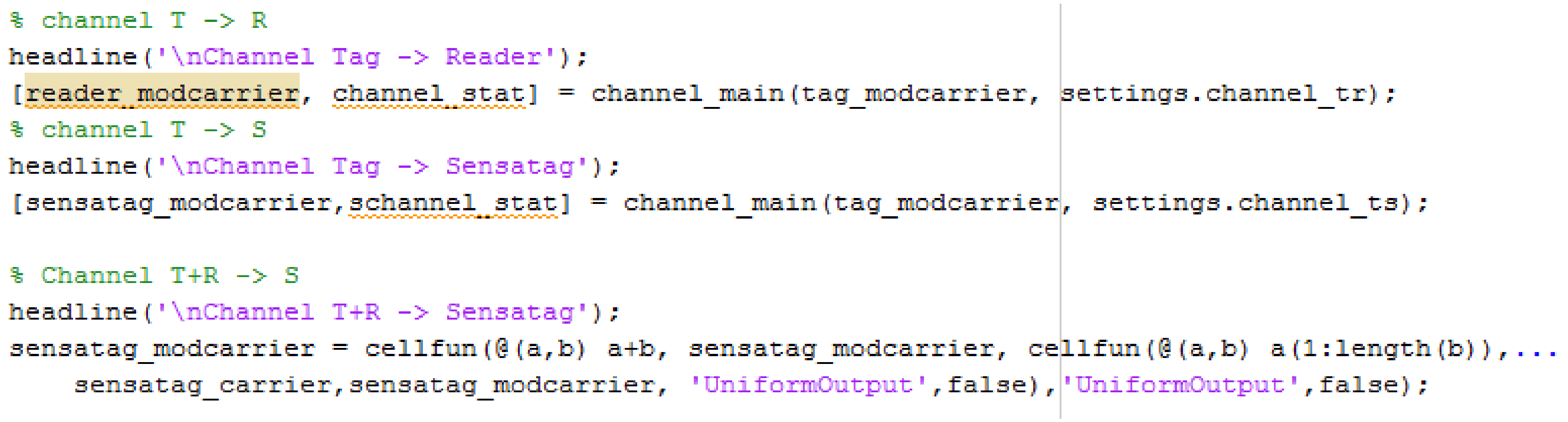
- Uplink: tag backscattering with RN16 (tag->(reader, sensatag))



```
% modulate RN16 or just reflect
headline('\nModulation of RN16 (only active tags)');
for i = 1 : settings.tagpool.n
    headline('    Tag %i', i);
    tx_temp = tag_main('tx_data_active', tag_carrier{i}, settings.tag{i});
    tag_modcarrier{i} = tx_temp.txsignal; % signal
    settings.tag{i}   = tx_temp.settings; % modifications to vdda
end
```

Behavioral description script: *sim_sensatag_room.m*

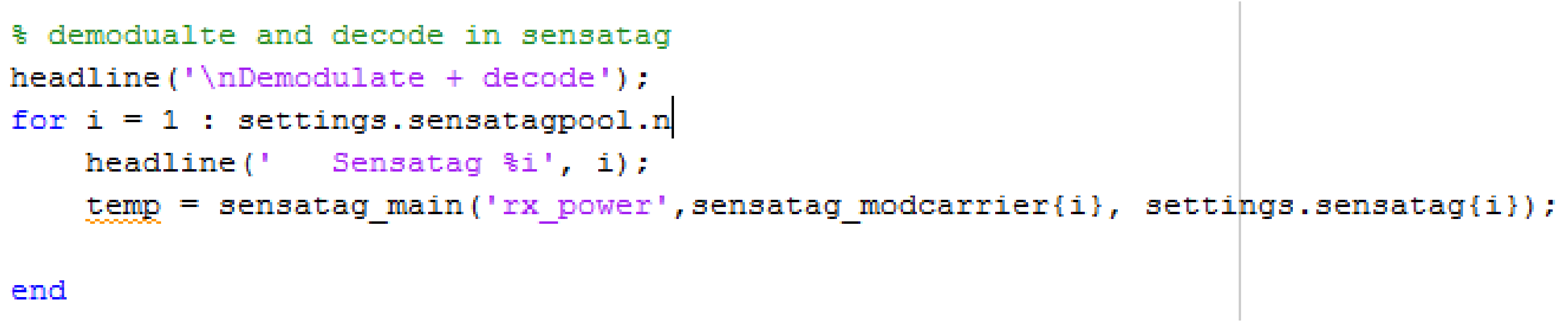
- Uplink: through channel



```
% channel T -> R
headline('\nChannel Tag -> Reader');
[reader_modcarrier, channel_stat] = channel_main(tag_modcarrier, settings.channel_tr);
% channel T -> S
headline('\nChannel Tag -> Sensatag');
[sensatag_modcarrier,schannel_stat] = channel_main(tag_modcarrier, settings.channel_ts);

% Channel T+R -> S
headline('\nChannel T+R -> Sensatag');
sensatag_modcarrier = cellfun(@(a,b) a+b, sensatag_modcarrier, cellfun(@(a,b) a(1:length(b)),...
    sensatag_carrier,sensatag_modcarrier, 'UniformOutput',false),'UniformOutput',false);
```

- Uplink: sensatag



```
% demodualte and decode in sensatag
headline('\nDemodulate + decode');
for i = 1 : settings.sensatagpool.n
    headline('    Sensatag %i', i);
    temp = sensatag_main('rx_power',sensatag_modcarrier{i}, settings.sensatag{i});

end
```

Conclusion

- The simulator could be used to establish various scenarios for UHF RFID system, which is convenient to design and test the localization and tracking algorithm.

- The simulator dose not implement the selection algorithm, such as Q selection. Thus the simulator could not be used for any traffic or throughput performance simulation.