4.8 Wireless communications

4.8.1 RFComm Server

The models *RFComm_server_[MODELNAME].mdl* are a number of simple test models which can be used to experiment with the radio-frequency (RF) wireless communication blocks of the toolbox. This latest addition to the toolbox provides a simple interface for wireless communications using *Nordic nRF24L01* radio modules (2.4 GHz, maximum data throughput: 2 MBit/s). These powerful little communication modules have been integrated in the easy to use transceiver circuits *MiRF v2* (distributed by SparkFun, less than US$20, 2006). In our laboratory, we use these transceiver chips on our mobile robots (Figure 4-21).

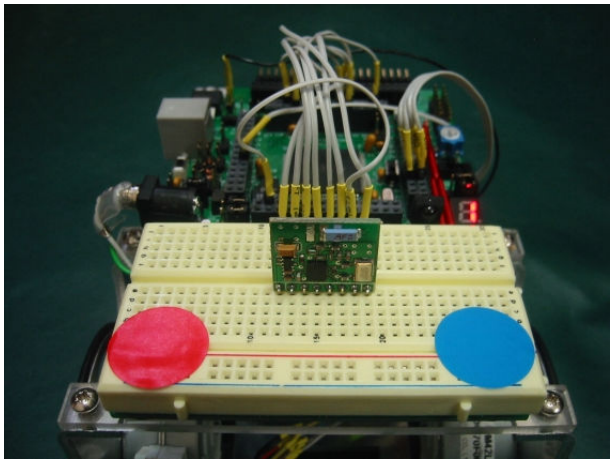

Figure 4-21 RF modules on the mobile robots used at The University of Adelaide

The software interface to the *MiRF v2* modules, written by one of our students (Stephen Craig) in support of a final year project (design of an autonomous paraglider) has been integrated in a series of blocks for the toolbox. The structure of these blocks resembles that of the FreePort communication blocks. At present the toolbox supports one radio module per microcontroller. The built-in transmission layer of these modules allows for bidirectional communications, i. e. each module can be used to transmit as well as receive using one of 128 separate RF channels. Set-up as a transmitter, the *RF Server* block can transmit data to up to 5 clients. Figure 4-22 shows the sample model *RFComm_server_TX_freePort0_5x2_formatted_noExt.mdl*. This model sets up two server transmit (TX) blocks using logical channels ‘0’ and ‘1’. Per client, up to 10 of these logical channels can be chosen. These channels resemble the channel number of the FreePort block. It is thus possible to use one and the same RF channel (0 … 127) for up to 10 separate logical channels.

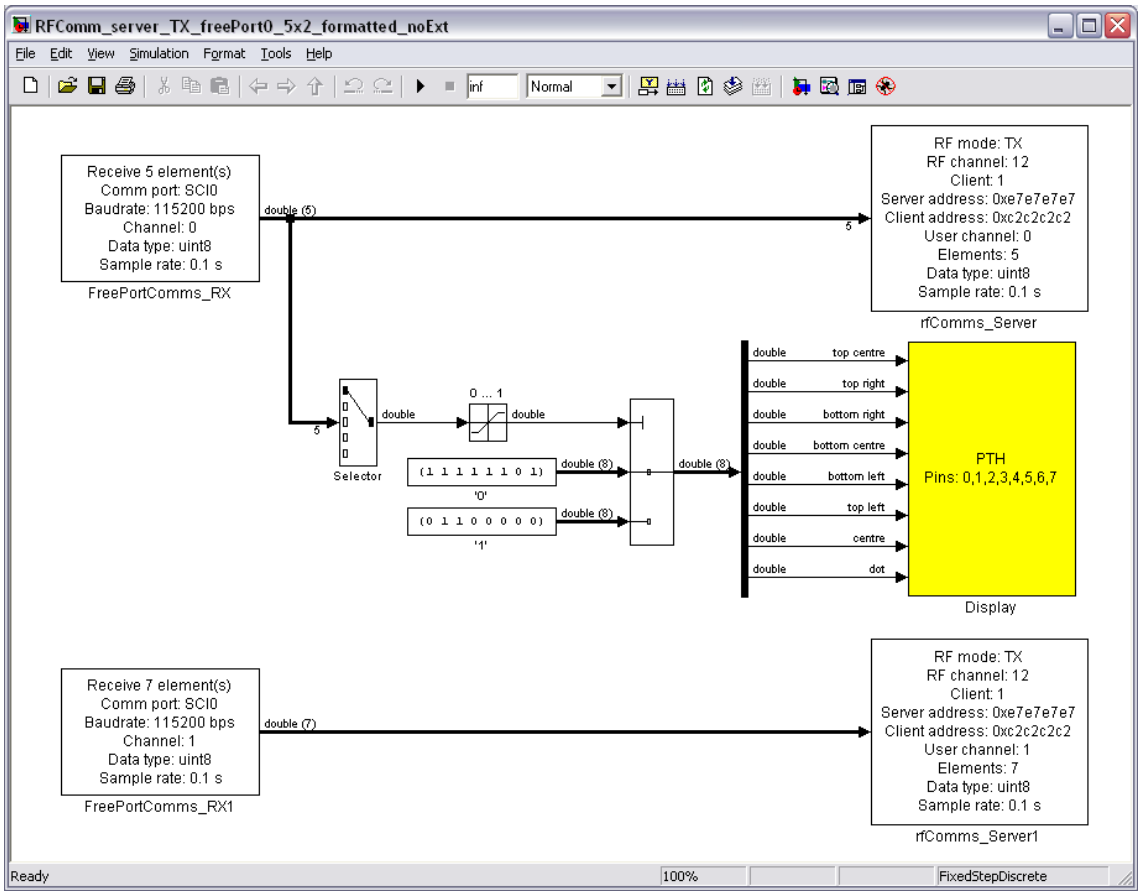

Figure 4-22 Sample model *RFComm_server_TX_freePort0_5x2_formatted_noExt*

Note that the underlying code currently limits the maximum number of bytes to be sent in any RF transmission to 32. This has implications on how much data can be sent through a RFComms block at anyone time. Using the formatted data transmission of the RFComms blocks, each telegram has a size of

4 + number_of_elements x element_size bytes.

In this equation, the *number_of_elements* defines the width of the block input, whereas the *element_size* is given by the width of each data type.

The following data types are supported:

type	width
single	*4*
uint8	*1*
int8	*1*
uint16	*3*
int16	*3*
int32	*4*
int32	*4*
boolean	*1*

This means that it should be possible to send up to 7 single precision numbers per transmission, or 28 individual data bytes.

The above model (Figure 4-22) runs on a MiniDragon+ board. The first of the 5 elements received on FreePort channel '0' (serial communication interface SCI0) is used to select one of two displays on 7-segment display of the board (connected to port H). All 5 elements – here: uint8 – are sent via the wireless link to client number '1' using logical channel '0'. Note that this channel number has nothing to do with the one used by the corresponding FreePort block.

The second FreePort channel ('1'), also connected to serial port SCI0, receives 7 bytes (uint8) and sends them to the same client via the wireless link. Logical channel '1' is used for this transmission. Again, this channel number has nothing to do with the corresponding FreePort channel. The radio module acts as server of the RF communication network. RF channel '12' is used and the server and client addresses have been set to 0xe7e7e7e7 and 0xc2c2c2c2c2, respectively. Note that the model runs in 'normal mode' (as opposed to 'External mode'); once compiled, it is not controlled by MATLAB anymore.

The data which is sent to this model through serial communication interface SCI0 is generated by a small m-file script which sends data through the COM1 serial port of the host computer *(rfComms_server_TX_freePort0_5x2_test_formatted.m)*. A set of 5 bytes (uint8, data type: '2') is sent using user channel '0', followed by a set of 7 bytes on user channel '1':

```
% test program, 'RFComm_server_TX_freePort0_5x2_formatted_noExt.mdl'

i = 0;
while(1)
    myData1 = i*[1 2 3 4 5]
    myData2 = i*[1 2 3 4 5 6 7]
    disp(['sending [' num2str(myData1) ']']);
    freePortSend(1, 115200, 0, 5, 2, myData1)
    disp(['sending [' num2str(myData2) ']']);
    freePortSend(1, 115200, 1, 7, 2, myData2)
    i = i + 1;
    if(i == 2) i = 0; end
    pause
end

disp('never reached')
```

Figure 4-23 shows the block parameters of the *rfComms_Server* block in transmit (TX) mode. Note the similarity of this block and the FreePort block. The only parameters which distinguish this block from the FreePort block are the *radio frequency channel* (RF channel, 0 … 127), the *server address* (hexadecimal, 4 bytes), the *client address* (hexadecimal, 4 bytes) and the number of clients to be serviced (1 – 5).

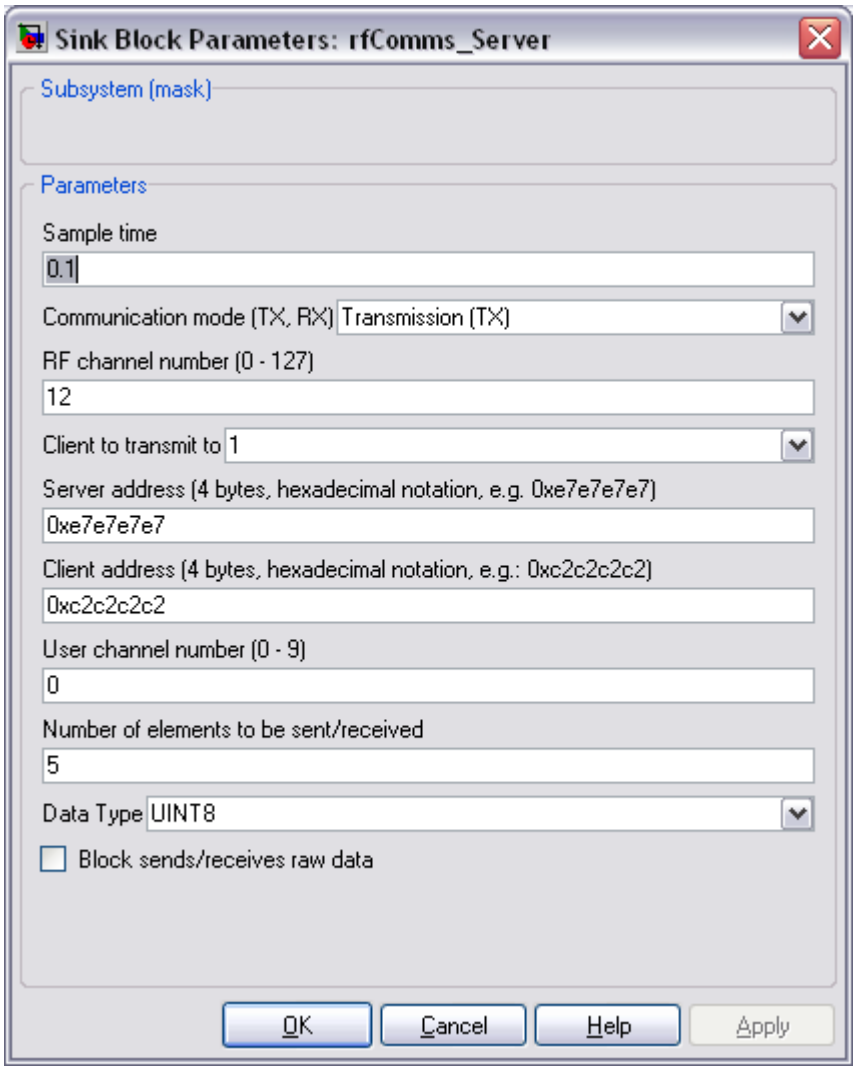

Figure 4-23 Block parameters of block *rfComms_Server*

The Server and client addresses can be chosen feely – they simply have to match on all communication partners (server, clients). Using more than one set of RF modules in the same area requires each communication network to be run on a different RF channel. This is for instance important when multiple mobile robot experiments are to be run simultaneously within the same laboratory. Assigning a different RF channel to each group allows for up to 128 parallel communication networks – more than enough for most applications of this toolbox. Figure 4-24 shows an example of a multi-robot configuration we used during experiments in our laboratory. The two robots on the right-hand side of the image are transmitters on channel ‘12’ and ‘13’, respectively. The robot in the middle is client #1 of the channel ‘13’ network, whereas the outer robots are clients #1 to #5 of the channel ’12’ network.

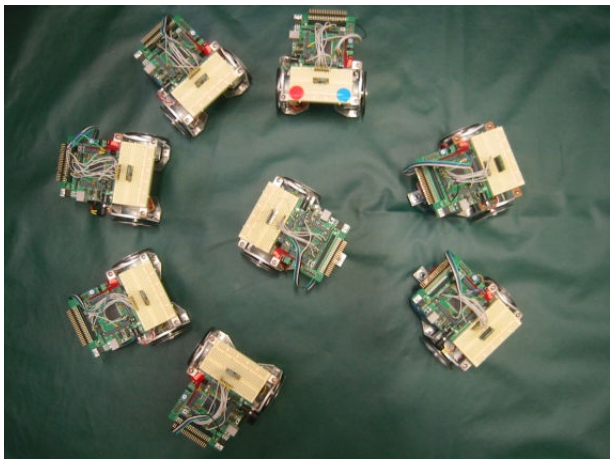

Figure 4-24 Multi-robot experiment

Figure 4-25 shows the client model which could be used to form the receiving end of this model: *RFComm_client_RX_freePort0_5x2_formatted_noExt*. It has a very similar structure to that of the above model. The first element of the 5 bytes received on logical channel '0' is filtered out and controls the 7-segment display on the MiniDragon+ board. In addition, the data is sent through serial communication interface SCI1 to a model running on the host. The latter is shown in Figure 4-26.

Note that, like the FreePort communication blocks, the rfComms blocks can be configured to send and receive unformatted data streams. In this case, all incoming bytes (up to 32) are sent without adding a telegram header (number of bytes, channel number, data type).

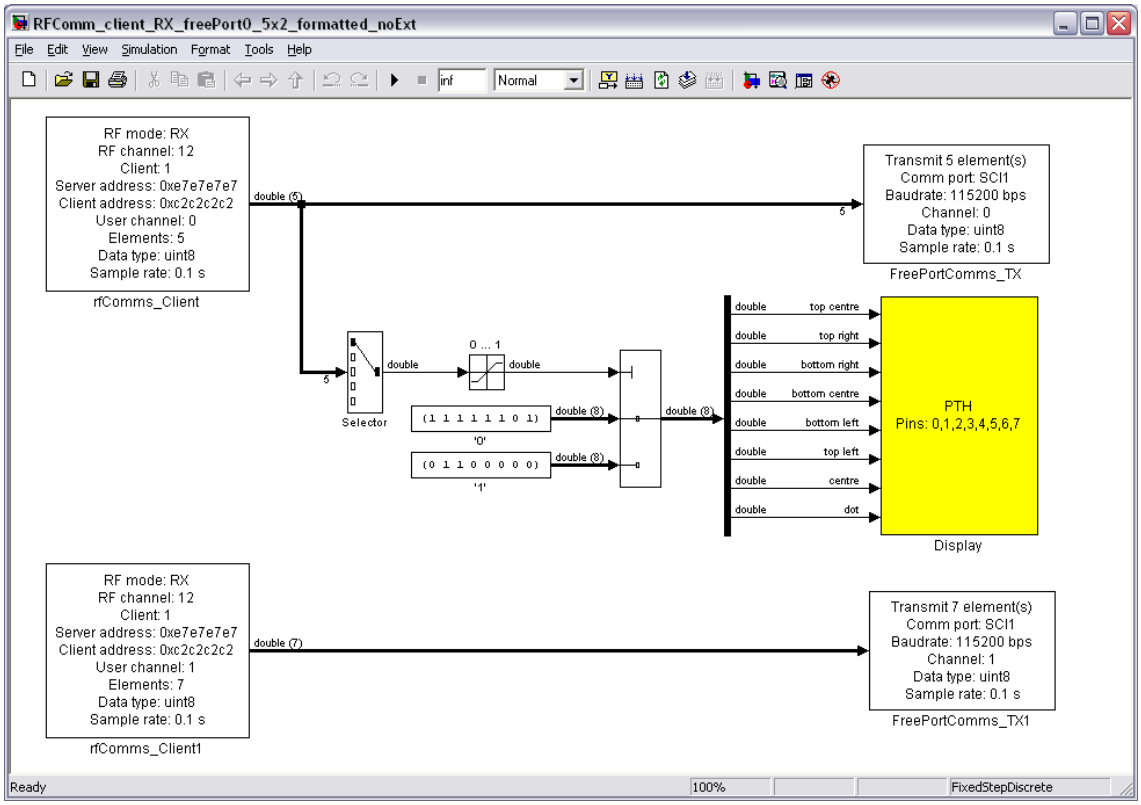

Figure 4-25 Sample model *RFComm_client_RX_freePort0_5x2_formatted_noExt*

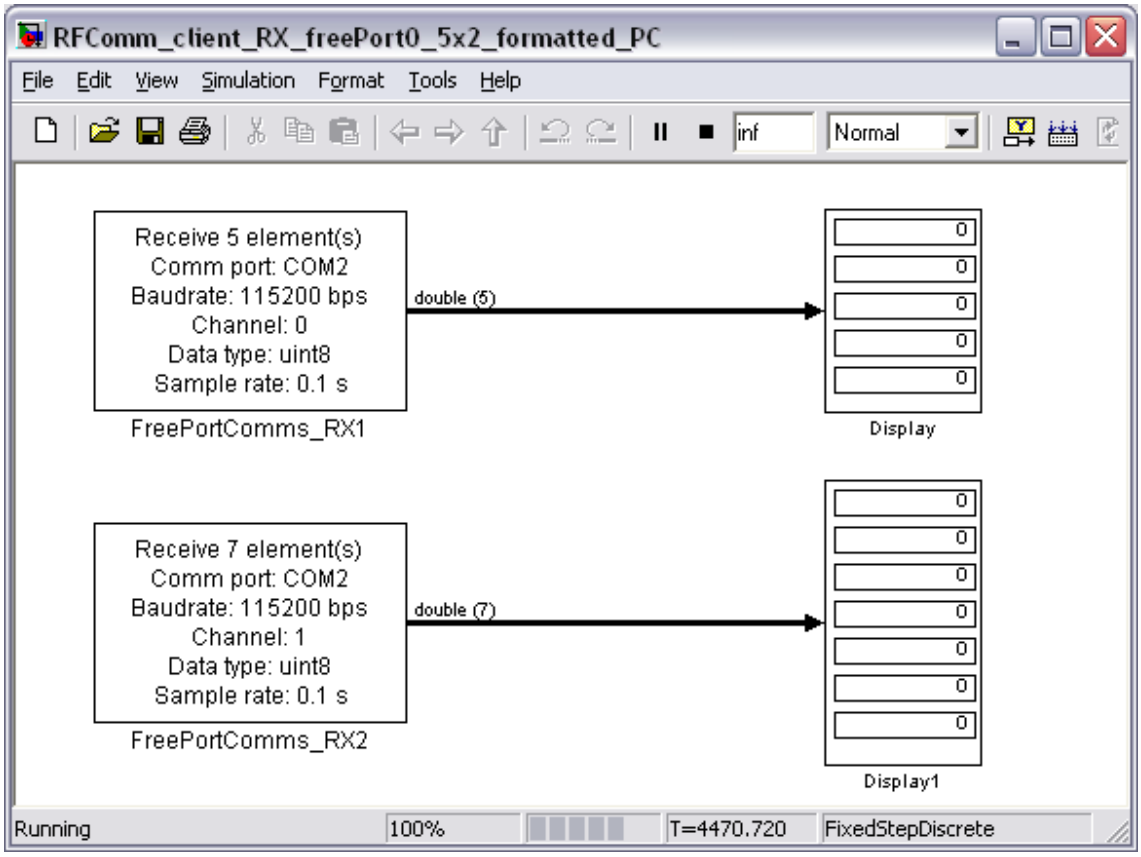

Figure 4-26 Host model for *RFComm_client_RX_freePort0_5x2_formatted_noExt*

4.8.2 RFComm Client

The models *RFComm_client_[MODELNAME].mdl* are simple test models which can be used to experiment with the radio-frequency (RF) wireless communication blocks of the toolbox. The block parameters of the client blocks are very similar to those of the server blocks (Figure 4-27). The client number now specifies which client this block belongs to.

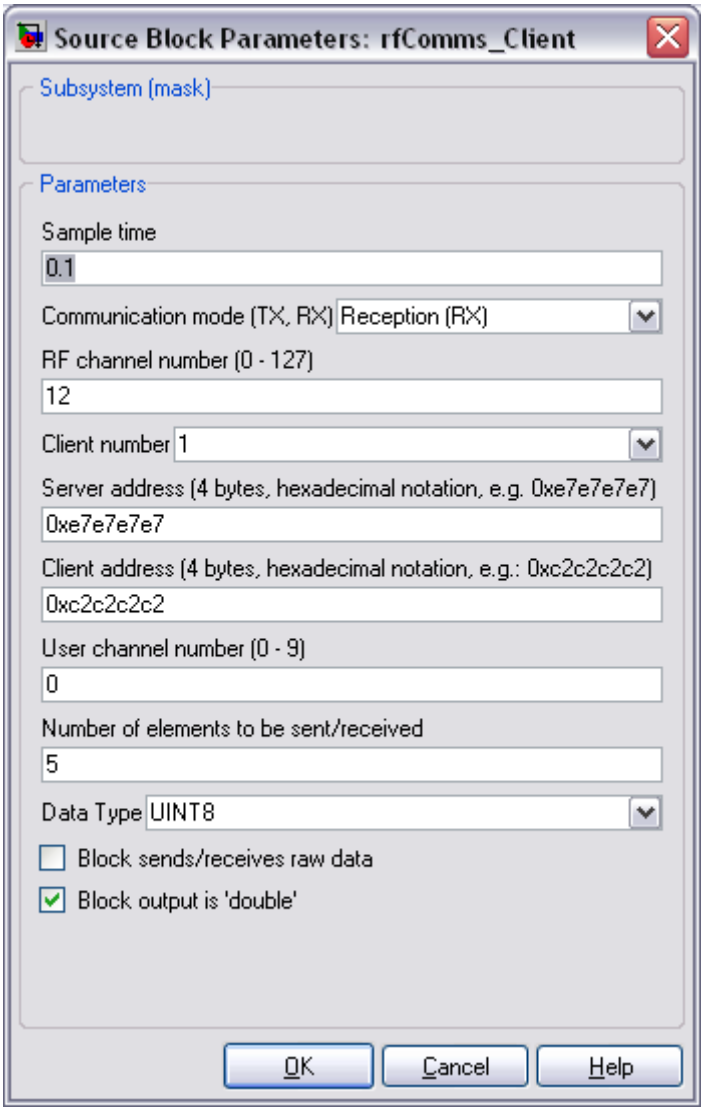

Figure 4-27 Block parameters of block *rfComms_Client*

The nRF24L01 is very versatile. Many more configurations are possible. One module could for instance be configured to act as a central base station, receiving data through up to 6 separate data pipes. This might be of interest in applications in which a number of sensors need to be connected to a central hub. Figure 4-28 shows the star configuration with a primary receiver and 6 transmitting nodes.

The same network topology can be used to transmit data to up to 5 different clients. Figure 4-29 shows an example, *RFComm_server_TX_counter_5clients_noExt.mdl*, which services 5 clients simultaneously. In this example, the same information is relayed to all 5 clients and the data transmission via the radio link is not formatted. The model also runs in 'normal' mode ('noExt').

The corresponding client models are *RFComm_client_RX_PTH_client1.mdl* to *RFComm_client_RX_PTH_client5.mdl* (Figure 4-30). A formatted version exists as well. It is important to match the transmission parameters on both ends of a RF link. This includes the channel number, number of elements, data type for formatted transmissions and consists of the number of bytes to be sent/received for unformatted transmissions.

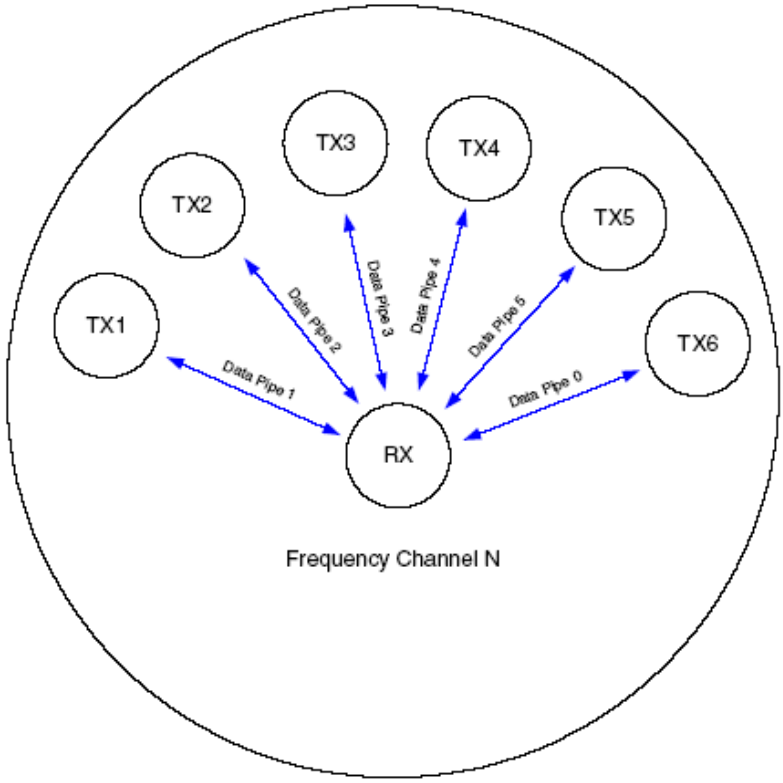

Figure 4-28 Using the nRF24L01 in *star* configuration

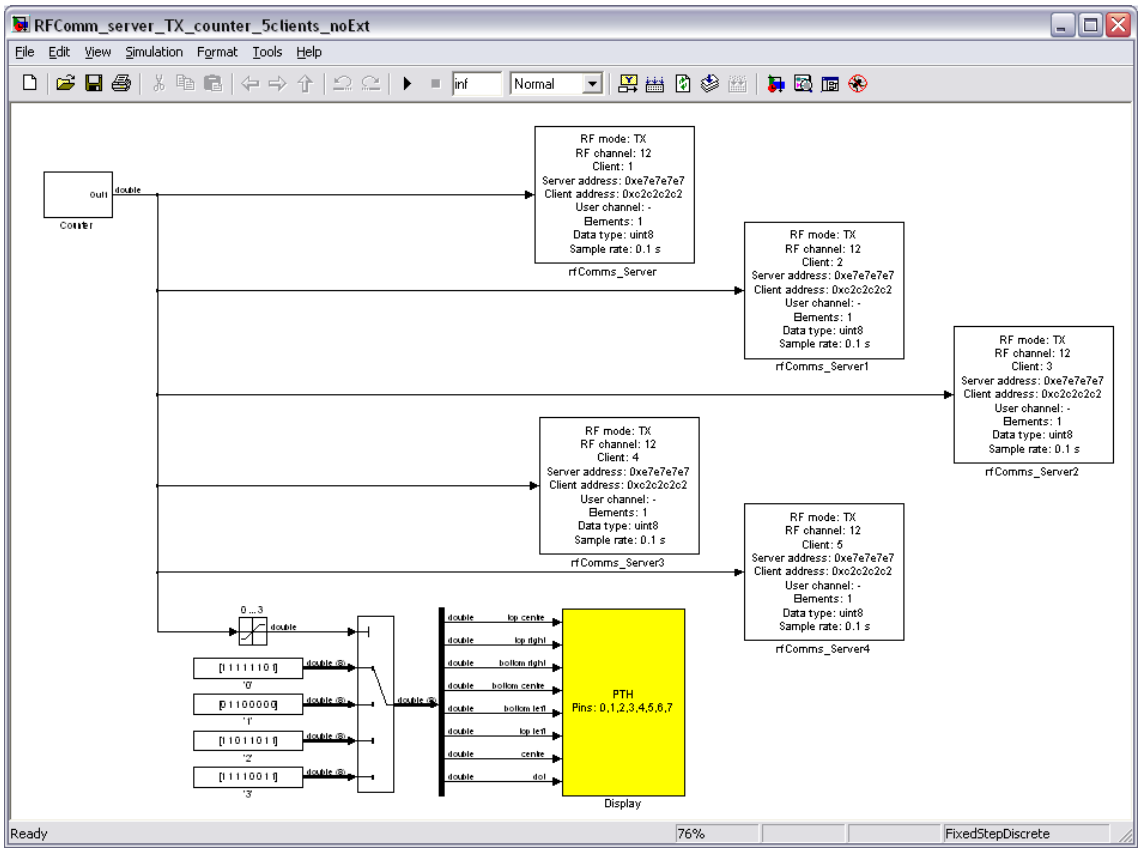

Figure 4-29 Sample model *RFComm_server_TX_counter_5clients_noExt*

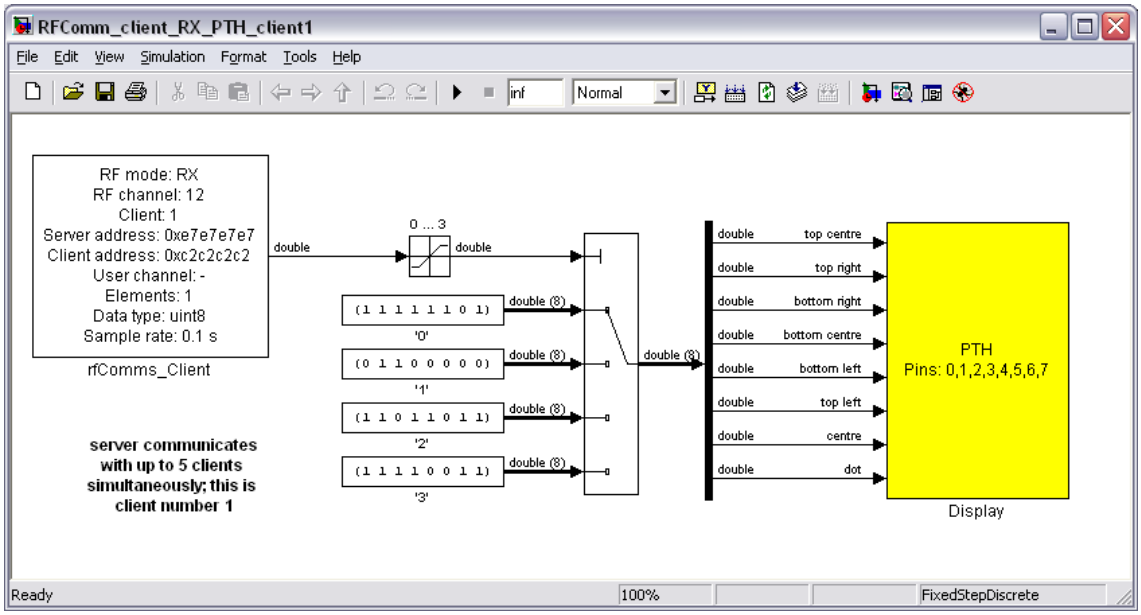

Figure 4-30 Sample model *RFComm_client_RX_PTH_client1.mdl*

4.8.3 Other RFComms models

The remaining *RFComm_server* and *RFComm_client* models should make it easy to experiment with the radio modules. The names of these models give an indication of what a model is trying to achieve. For example, server model *RFComm_server_TX_counter_formatted_noExt.mdl* (Figure 4-31) implements a simple counter (0 to 3) which controls the on-board 7-segment display of the MiniDragon+ board. Data values are transmitted to a single client using formatted data telegrams (logical channel '0', '1' element of type 'uint8'). The model runs in 'normal' mode (no 'External mode').

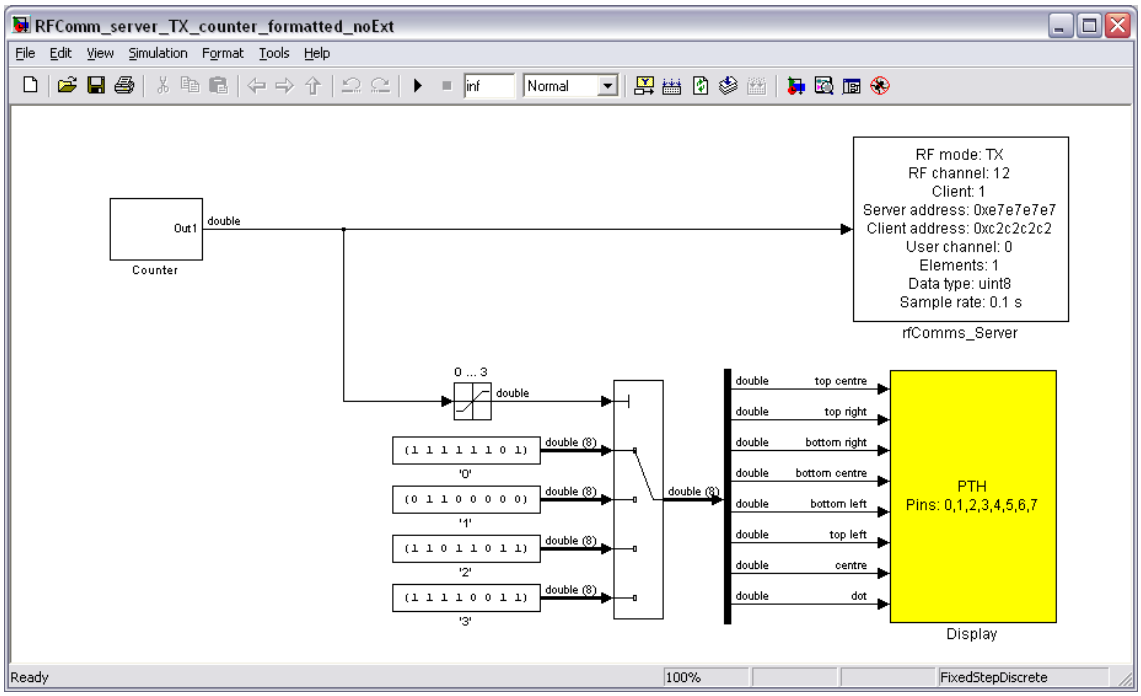

Figure 4-31 Sample model *RFComm_server_TX_counter_formatted_noExt.mdl*

Whenever a sample model receives data from the host, the name includes ‘freeport’. An example is *RFComm_server_RXTX_formatted_noExt.mdl*, with client model *RFComm_client_RXTX_formatted_noExt.mdl*. Figure 4-32 shows the server end of affairs. This model receives 5 elements of type ‘uint8’ from serial port SCI0 and sends them via the RF link to the client (formatted transmission).

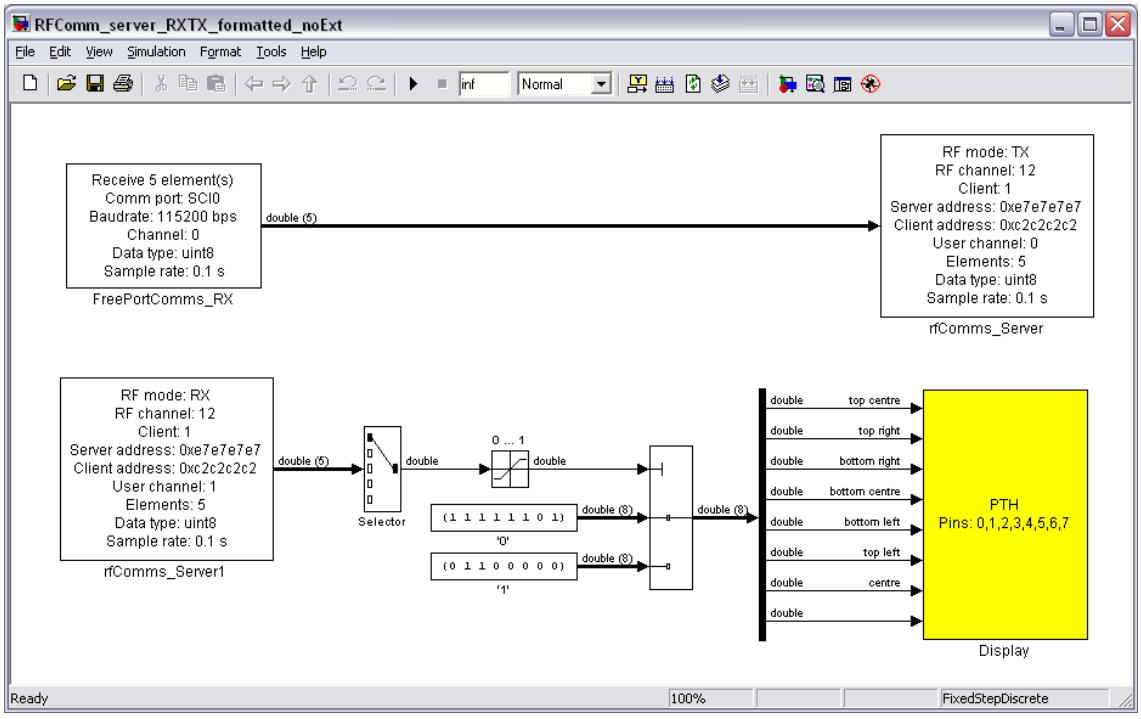

Figure 4-32 Sample model *RFComm_server_RXTX_formatted_noExt.mdl*

Figure 4-33 shows the client side of this model. The model receives the 5 bytes on logical channel ‘0’ and loops them back using logical channel ‘1’. The first element of the received data is used to control the 7-segment display of the receiving MiniDragon+ board (displaying either ‘0’ or ‘1’). The server model receives the returned data and also displays either ‘0’ or ‘1’, depending on the first element of the 5 incoming bytes. This sample model thus demonstrates that both the server as well as the client can be used to send and receive at the same time.

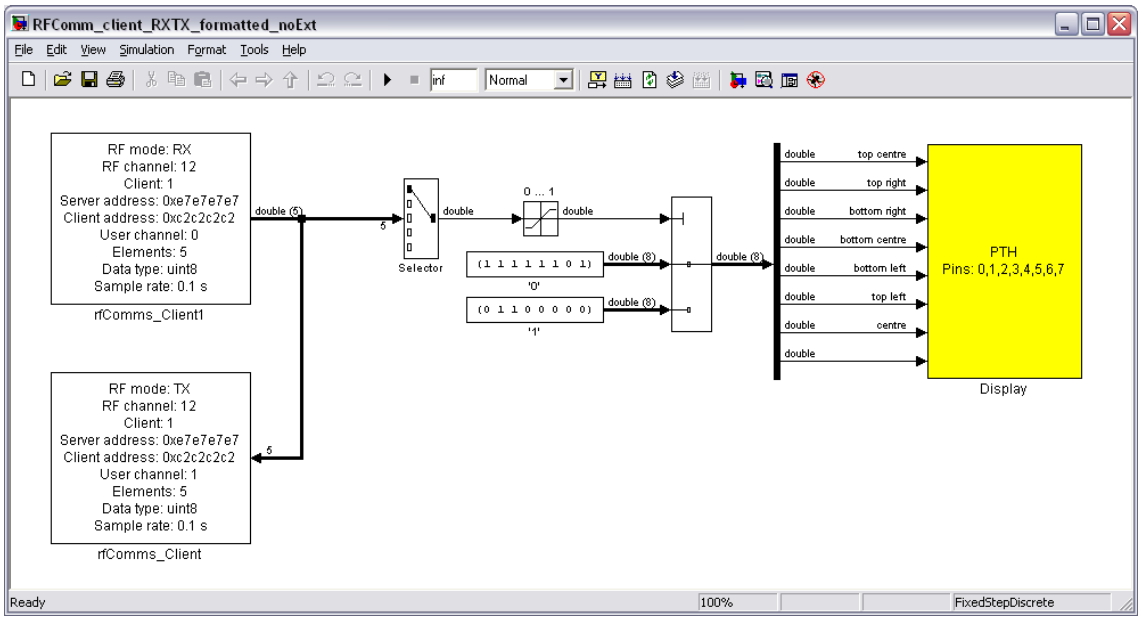

Figure 4-33 Sample model *RFComm_client_RXTX_formatted_noExt.mdl*

When experimenting with the remaining RF sample models, try to find matching server and client models and run them according to what is implied by their file name (‘External mode’, ‘normal mode’, etc.). Note that some models have associated m-file scripts which send data to the corresponding FreePort blocks.

The underlying code is fairly robust, taking care of temporary loss of transmission (receiver out of range), resending of erroneous data packages, etc. The RF interface thus provides an inexpensive, yet reliable means of communication between microcontrollers. Using antennas we have managed to achieve a total range of around 80 meters.