Manuals and Curriculum (http://playground.arduino.cc/Main/ManualsAndCurriculum)
Arduino StackExchange (http://arduino.stackexchange.com)
Board Setup and Configuration (http://playground.arduino.cc/Main/ArduinoCoreHardware)
Development Tools (http://playground.arduino.cc/Main/DevelopmentTools)
Arduino on other Atmel Chips (http://playground.arduino.cc/Main/ArduinoOnOtherAtmelChips)
Interfacing With Hardware (http://playground.arduino.cc/Main/InterfacingWithHardware)
Interfacing with Software (http://playground.arduino.cc/Main/InterfacingWithSoftware)
User Code Library (http://playground.arduino.cc/Main/GeneralCodeLibrary)
Suggestions & Bugs (http://code.google.com/p/arduino

nRF24L01

An Arduino port of the tinkerer.eu (http://www.tinkerer.eu/AVRLib/nRF24L01) library. It works with the Sparkfun nRF24L01+ (http://www.sparkfun.com/commerce/product_info.php?products_id=691) modules.

Note: This library supports a small (but useful) subset of the features provided by the nRF24L01 chip.

Download -> Mirf.zip (http://playground.arduino.cc/uploads/InterfacingWithHardware/Mirf.zip)

Also see github repository (http://github.com/aaronds/arduino-nrf24l01)

Another fork of Mirf that covers attiny cores github repository (http://github.com/stanleyseow/arduino-nrf24l01)

Another nRF24L01 library github RF24 repo (https://github.com/stanleyseow/RF24) - This libraries also includes Raspberry Pi libraries/drivers

Documentation

Pins:

- MISO -> 12
- MOSI -> 11
- SCK -> 13

Configurable:

- CE -> 8
- CSN -> 7

Properties:

byte cePin

CE Pin controls RX / TX, default 8.

byte csnPin

CSN Pin (Chip select not), default 7.

byte channel

RF Channel 0 - 127 or 0 - 84 in the US, default 0.

byte payload

Size in bytes, default 16, max 32.

/issues/list)
Electronics Technique (http://playground.arduino.cc/Main/ElectroInfoResources)
Sources for Electronic Parts (http://playground.arduino.cc/Main/Resources)
Related Hardware and Initiatives (http://playground.arduino.cc/Main/SimilarBoards)
Arduino People/Groups & Sites (http://playground.arduino.cc/Main/People)
Exhibition (http://playground.arduino.cc/Projects/ArduinoUsers)
Project Ideas (http://playground.arduino.cc/Projects/Ideas)
Languages (http://playground.arduino.cc/Main/Languages)

PARTICIPATE (http://playground.arduino.cc/Main/Participate)

NB: channel and payload must be the same for all nodes.

Methods:

void init(void)

Initialize the module, set the pin modes for the configurable pins and initialize the SPI module.

```
Example:
Mirf.csnPin = 9;
Mirf.cePin = 7;

Mirf.init();
```

void setRADDR(byte *addr)

Set the receiving address. Addresses are 5 bytes long.

```
Example:
Mirf.setRADDR((byte *)"addr1");
```

void setTADDR(byte *addr)

Set the sending address.

```
Example:
Mirf.setTADDR((byte *)"addr1");
```

void config(void)

Set channel and payload width. Power up in RX mode and flush RX fifo.

```
Example:
Mirf.payload = 32;
Mirf.channel = 2;
Mirf.config();
```

bool dataReady(void)

Is there data ready to be received?.

```
Example:
if(Mirf.dataReady()){
  //Get the data to play with.
}
```

void getData(byte *data)

Get the received data. 'data' should be an array of bytes Mirf.payload long.

```
Example:
byte data[Mirf.payload]
Mirf.getData(data);
```

void send(byte *data)

Send data. 'data' should be Mirf.payload bytes long.

bool isSending(void)

Return true if still trying to send. If the chip is still in transmit mode then this method will return the chip to receive mode.

```
Example:
Mirf.send(data);

while(Mirf.isSending()){
 //Wait.
}

//Chip is now in receive mode.
```

NB: Lots more information is available from the status registers regarding acknolagement or failure status. See Mirf.cpp:218.

bool rxFifoEmpty(void)

Is the RX Fifo Empty.

bool txFifoEmpty(void)

Is the TX Fifo Empty.

byte getStatus(void)

Return the status register.

void powerUpRx(void)

Power up chip and set to receive mode. Also clear sending interrupts.

void powerUpTx(void)

Power up tx mode.

Examples

See examples folder in zip file.

Share