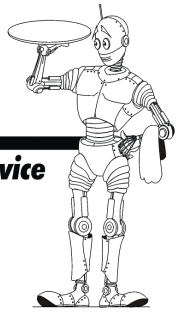
RobotShop INC.

La robotique à votre service



Arduino Xbee Shield

Overview

The Xbee shield allows an Arduino board to communicate wirelessly using Zigbee. It is based on the Xbee module from MaxStream. The module can communicate up to 100 feet indoors or 300 feet outdoors (with line-of-sight). It can be used as a serial/usb replacement or you can put it into a command mode and configure it for a variety of broadcast and mesh networking options. The shields breaks out each of the Xbee's pins to a through-hole solder pad. It also provides female pin headers for use of digital pins 2 to 7 and the analog inputs, which are covered by the shield (digital pins 8 to 13 are not obstructed by the shield, so you can use the headers on the board itself).

The Xbee shield was created in collaboration with Libelium, who developed it for use in their SquidBee motes (used for creating sensor networks).

Jumper Settings

The Xbee shield has two jumpers (the small removable plastic sleeves that each fit onto two of the three pins labelled Xbee/USB). These determine how the Xbee's serial communication connects to the serial communication between the microcontroller (ATmega8 or ATmega168) and FTDI USB-to-serial chip on the Arduino board.

With the jumpers in the **Xbee** position (i.e. on the two pins towards the interior of the board), the DOUT pin of the Xbee module is connected to the RX pin of the microcontroller; and DIN is connected to TX. Note that the RX and TX pins of the microcontroller are still connected to the TX and RX pins (respectively) of the FTDI chip - data sent from the microcontroller will be transmitted to the computer via USB as well as being sent wirelessly by the Xbee module. The microcontroller, however, will only be able to receive data from the Xbee module, not over USB from the computer.

With the jumpers in the **USB** position (i.e. on the two pins nearest the edge of the board), the DOUT pin the Xbee module is connected to the RX pin of the *FTDI chip*, and DIN on the Xbee module is connected to the TX pin of the FTDI chip. This means that the Xbee module can communicate directly with the computer - however, *this only works if the microcontroller has been removed from the Arduino board*. If the microcontroller is left in the Arduino board, it will be able to talk to the computer normally via USB, but neither the computer nor the microcontroller will be able to talk to the Xbee module.

Addressing

There are multiple parameters that need to be configured correctly for two modules to talk to each other (although with the default settings, all modules should be able to talk to each other). They need to be on the same network, as set by the **ID** parameter (see "Configuration" below for more details on the parameters). The modules need to be on the same channel, as set by the **CH** parameter. Finally, a module's destination address (**DH** and **DL** parameters) determine which modules on its network and channel will receive the data it transmits. This can happen in a few ways:

- If a module's **DH** is 0 and its **DL** is less than 0xFFFF (i.e. 16 bits), data transmitted by that module will be received by any module whose 16-bit address **MY** parameter equals **DL**.
- If **DH** is 0 and **DL** equals 0xFFFF, the module's transmissions will be received by all modules.
- If **DH** is non-zero or **DL** is greater than 0xFFFF, the transmission will only be received by the module whose serial number equals the transmitting module's destination address (i.e. whose **SH** equals the transmitting module's **DH** and whose **SL** equals its **DL**).

Again, this address matching will only happen between modules on the same network and channel. If two modules are on different networks or channels, they can't communicate regardless of their addresses.

Configuration

Here are some of the more useful parameters for configuring your Xbee module. For step-by-step instructions on reading and writing them, see the guide to the Xbee shield. Make sure to prepend `AT` to the parameter name when sending a command to the module (e.g. to read the `ID` parameter, you should send the command `ATID`).

Command	*Description*	*Valid Values*	*Default Value*
`ID`	The network ID of the Xbee module.	0 - 0xFFFF	3332
`CH`	The channel of the Xbee module.	0x0B - 0x1A	0X0C
`SH` and `SL`	The serial number of the Xbee module (`SH` gives the high 32 bits, `SL` the low 32 bits). Read-only.	0 - 0xFFFFFFFF (for both `SH` and `SL`)	different for each module
`MY`	The 16-bit address of the module.	0 - 0xFFFF	0
`DH` and `DL`	The destination address for wireless communication (`DH` is the high 32 bits, `DL` the low 32).	0 - 0xFFFFFFFF (for both `DH` and `DL`)	0 (for both `DH` and `DL`)
`BD`	The baud rate used for serial communication with the Arduino board or computer.	0 (1200 bps) 1 (2400 bps) 2 (4800 bps) 3 (9600 bps) 4 (19200 bps) 5 (38400 bps) 6 (57600 bps) 7 (115200 bps)	3 (9600 baud)

Note: although the valid and default values in the table above are written with a prefix of "0x" (to indicate that they are hexadecimal numbers), the module will not include the "0x" when reporting the value of a parameter, and you should omit it when setting values.

Here are a couple more useful commands for configuring the Xbee module (you'll need to prepend `AT` to these too).

Command	*Description*
`RE`	Restore factory default settings (note that like parameter changes, this is not permanent unless followed by the `WR` command).
`WR`	Write newly configured parameter values to non-volatile (long-term) storage. Otherwise, they will only last until the module loses power.
`CN`	Exit command mode now. (If you don't send any commands to the module for a few seconds, command mode will timeout and exit even without a `CN` command.)

For more details on configuring the Xbee module, consult the product manual from MaxStream.