API

All

Log debug message:

Log.d()

Log warning message:

Log.w()

Log error message (also prints to STDERR):

Log.e()

Web server

Get list of running drivers:

getLoadedDrivers()

Check if remote module is present:

isModulePresent()

Send command to a driver

passCommand()

Driver

Send command to remote module:

Text: sendCommand()

Text: sendCommandAndWait()

Binary: sendBinary()

Send command to another driver:

passCommand()

Store data in database:

Text: storeTextData()

Binary: storeBinData()

Read data from database:

Text: readTextData()

Binary: readBinData()

Front-end (all request to the web server. Only example)

Web server host virtual documents, responds to GET requests on those documents with XML data

Get list of running drivers:

GET running\_drivers

Web server calls Coordinator.getLoadedDrivers(), formats ArrayList as xml, returns result

Get driver (and module type):

GET driver\_type?driver=led\_flash

Web server calls: Coordinator.getLoadedDrivers().contains(“led\_flash”), if result is true, then calls: (TODO, need to work this out, maybe pass a handle to the Driver so web server can directly call getModuleType())

Get widget xml

GET widget\_xml?driver=led\_flash

As above, still need to work out best method

Get full page xml

GET page\_xml?driver=led\_flash

As above, still need to work out best method

Protocols:

Version 0:

* Base version
* Text transmission only
* Only one remote module
* Requires sleeping ~2s between each transmission to avoid garbled text
* Simple setup, simple code
* Xbee in AT mode
* No newline characters allowed in header or command

Since each character in the transmission is sent separately there is no practical limit to the length of the message

Format:

[header | command | line-feed]

header is remote module name, terminated with colon

i.e.: “Destination\_name:command\n”

Version 1 (work in progress):

* Supports transmission in text or binary
* Should support many remote modules
* Should not require a sleep cycle between messages
* Xbee in API mode
* Requires breaking the command or binary data into chunks so that each chuck fits in a single Zigbee packet
* Allows newline characters in the data (text and binary)
  + Single linefeed chars replaced with doubled linefeed chars
* Added single byte to the start of the header (start byte)
* Allows the protocol to grow new features over time
* Allows coordinator to keep track of what protocol version each remote module speaks.
* Coordinator API remains the same for drivers
* Arduino sketches for remote modules must be updated

Format:

[start\_byte | destination | : | command | line-feed]

start byte:

[bit7 | bit6 … bit0]

bit7 : transmission type

1: binary

0: text

bit6 : reserved for future use

bit5 : reserved, always 1 (keeps the start byte from ever equaling the line-feed byte)

bits4 - 0 : protocol version

Version 2:

Need list of features desired (packet ordering