# Documentation

## Controlling MIDI via OSC and applying it to a two-handed layout for a keyboard

**Purpose**:
The purpose of the software is to take MIDI inputs from a QWERTY Keyboard and send it to the computer or other speaker output device.

**Class communicateOSC:**

This class is responsible for the communication between the member functions of midiKeyboard and User interface software (in our case, Max8). The purpose of the class is to define necessary communication protocols on the basis of the Open sound control library of python.

When a user calls the asynchronous function ***create*** of class **communicateOSC,** following steps get executed:
1. **Async function *create*** takes three arguments i.e. IP address, port_in and port_out and start initialization of data members.
2. It creates an object of AsyncIOOSCUDPServer() to start listening for the inputs from the UI(Max8) using the provided IP address and port_in value.
3. It also creates an object of SimpleUDPClient() which will send the messages out to UI(Max8) using the provided IP address and port_out value.
4. So, the asynchronous network connection is established between UI and python server using Open sound control library of python.

**Communication messages:**
After creation of the network, the client(Max8 here) and server(Python here) start communicating using following messages(Address and Arguments):

| **Input Messages(Address)** | **Output Messages(Address)** | **Description** |
|---|---|---|
| /inputs/key_mapping/keyboard, args = ("q2w3er5t6y7u",) | /outputs/key_mappings_spaced, ['q', '2', 'w', '3', 'e', 'r', '5', 't', '6', 'y', '7', 'u'] | Assigns the 12 keys of the octave to the 12 characters of the string passed. The keys will be mapped with values from 0 to 11.<br>Output message will return the list of characters to update the UI. |
| /inputs/key_mapping/octave,<br>args = ("o",) | /outputs/octave, "C3" | Assigns the key "o" to update the octave value.<br>Output message will contain the current Octave value i.e. "C3" which will update the Octave |

| | | |
|---|---|---|
| | | column in the UI. |
| /inputs/key_mapping/transpose,<br>args = (“p”,) | /outputs/transpose, 0 | Assigns the key “p” to update the transpose value.<br>Output message will contain the current transpose value i.e. “0” which will update the transpose column in the UI. |
| /inputs/key_mapping/pitch_bend,<br>args = (“9”,) | None | Assigns the key “9” to update the pitch bend value with the mouse movement. |
| /inputs/key_mapping/modulation,<br>args = (“0”,) | None | Assigns the key “0” to update the modulation value with the mouse movement. |
| /inputs/key_on/gen_MIDI,<br>args = (1,) | /outputs/gen_MIDI,[msg.note, msg.velocity] | It picks one of the sample file named as 'gen_midi_samples/sample_' + str(sample_no) + '.mid' and plays it. For example: 'gen_midi_samples/sample_21.mid’<br>“sample_no” varies from 21 to 30. |
| /inputs/key_on/gen_MIDI,<br>args = (0,) | None | It stops playing the sample midi. |
| /inputs/key_on, args = (113, 0) | /outputs/key, [48, 127] | When the key “q” is pressed(one of the key which is assigned to the 12 notes of the octave), it returns the processed value 48 (0(notes) + 48(Octave) + 0(Transpose)) and 127 which indicates the note is on.<br>[48,127] will also update the UI in the last key played column.<br>Note: ASCII value of “q” is 113. |
| /inputs/key_on, args = (112, 0) | /outputs/transpose, 1 | When the key “p” is pressed, it returns the current transpose value which is 1 here. And 1 will be updated in the transpose column of the UI.<br>Note: ASCII value of “p” is 112. |
| /inputs/key_on, args = (111, 0) | /outputs/octave, C4 | When the key “o” is pressed, it returns the current octave value which is C4 here. And C4 will |

| | | |
|---|---|---|
| | | be updated in the octave column of the UI.<br>Note: ASCII value of “o” is 111. |
| /inputs/key_on, args = (57, 0) | /outputs/pitch_bend, 64 | When the key “9” is pressed, it returns the current pitch_bend value which is 64 here. And 64 will be updated in the FE bars of pitch_bend column of the UI.<br>Note: ASCII value of “9” is 57. |
| /inputs/key_on, args = (48, 0) | /outputs/modulation, 44 | When the key “0” is pressed, it returns the current modulation value which is 44 here. And 44 will be updated in the FE bars of modulation column of the UI.<br>Note: ASCII value of “0” is 48. |
| /inputs/key_off, args = (113, 0) | /outputs/key, [48, 0] | When the key “q” is released(one of the key which is assigned to the 12 notes of the octave), it returns the processed value 48 (0(notes) + 48(Octave) + 0(Transpose)) and 0 which indicates the note is off.<br>Note: ASCII value of “q” is 113. |
| /inputs/key_off, args = (112, 0) | None | According to the example, if any key other than 'q', '2', 'w', '3', 'e', 'r', '5', 't', '6', 'y', '7', and, 'u' is released, it will return None. |
| Default | print(fUnknown Input - {address}: {args}) | If Unknown address is passed, it will print the default message. |

**NOTE: The arguments(args) used here are specific to current application and customizable as needed.**