OSC tutorial

OSC

OSC (open sound control) is a protocol for networking sound synthesizers, computers, and other multimedia devices for purposes such as musical performance or show control.

UniOSC

OSC for Arduino

This is an Arduino and Teensy library implementation of the OSC (Open Sound Control) encoding.It was developed primarily by Yotam Mann and Adrian Freed at CNMAT where OSC was invented. It benefits from contributions from John MacCallum, Matt Wright, Jeff Lubow and Andy Schmeder and many beta testers.

Features:

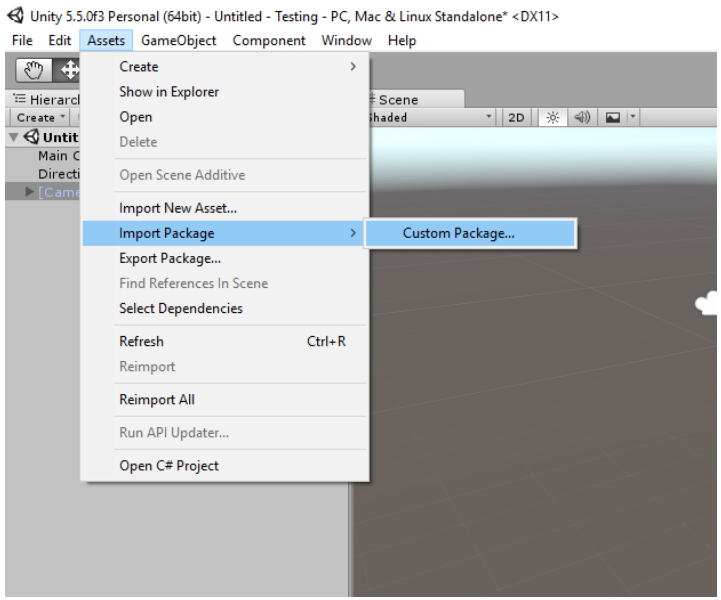
- Supports the four basic OSC data types (32-bit integers, 32-bit floats, strings, and blobs - arbitrary length byte sequences)
- Supports the optional 64-bit timetag data type and Booleans
- Address pattern matching
- Dynamic memory allocation
- Sends and receives OSC packets over transport layers that implements the Arduino Stream Class such as Serial and Ethernet UDP

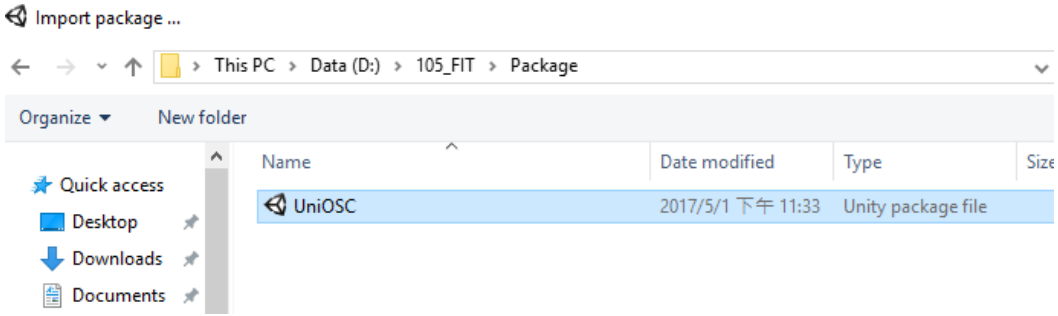
OSC features

- Open-ended, dynamic, URI-style symbolic naming scheme.
- Symbolic and high-resolution numeric data
- Pattern Matching language to specify multiple recipients of a single message
- High resolution time tags
- "Bundles" of messages whose effects must occur simultaneously

OSC on Unity (UniOSC package)

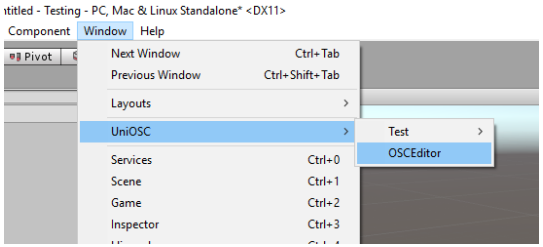
Import the UniOSC package.

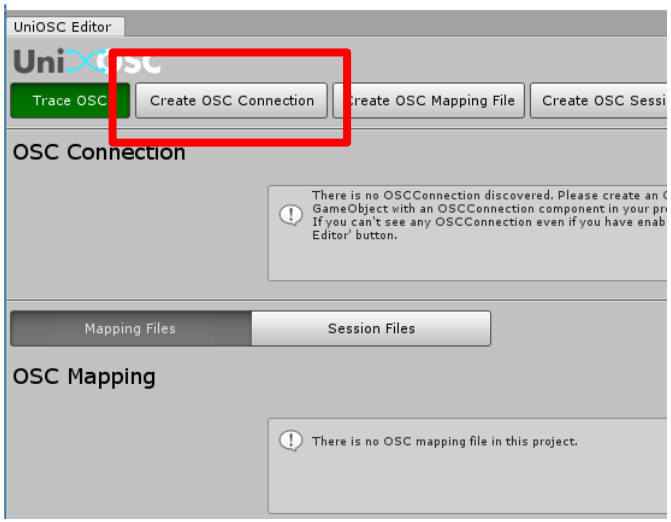


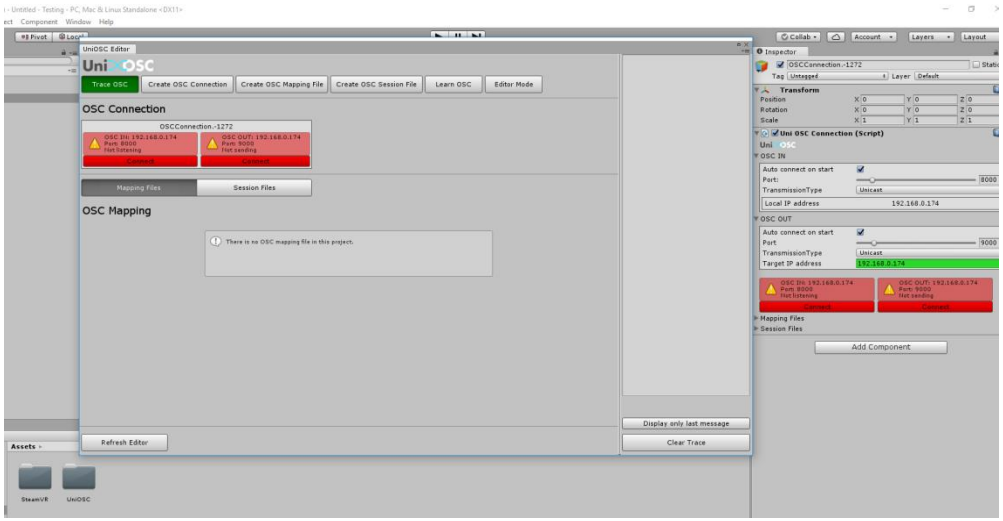


OSC on Unity (UniOSC package)

Open the OSCEditor Window and create a new OSC Connection.

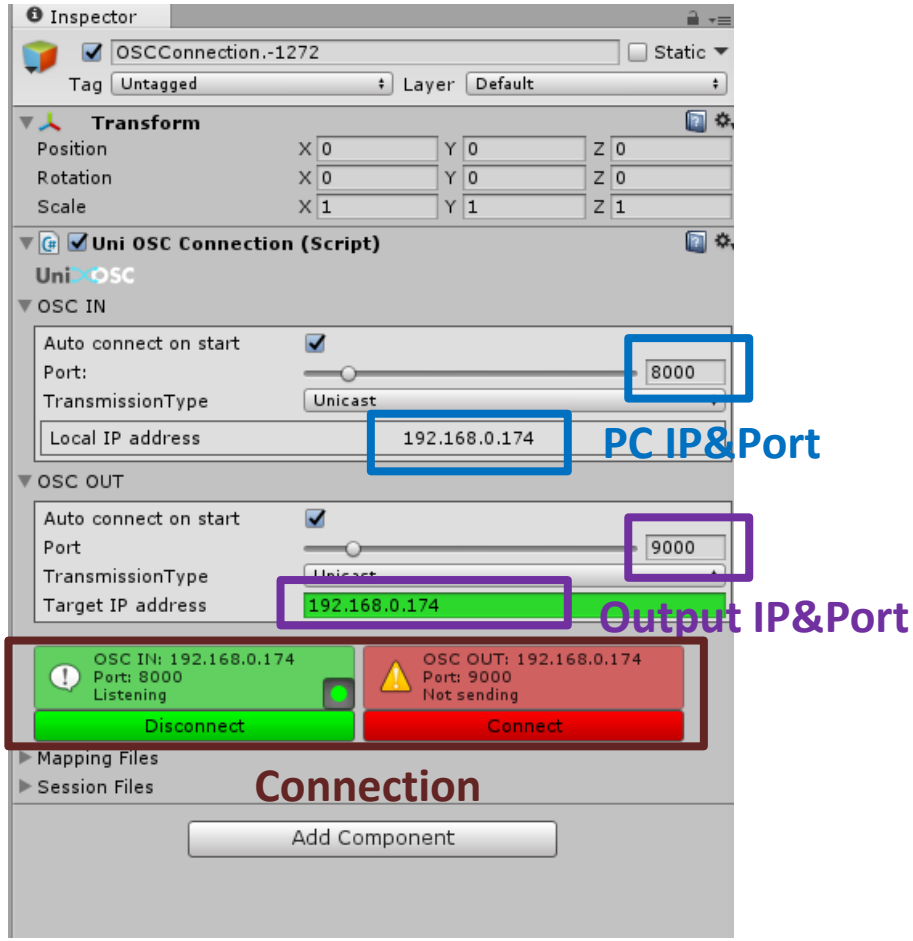






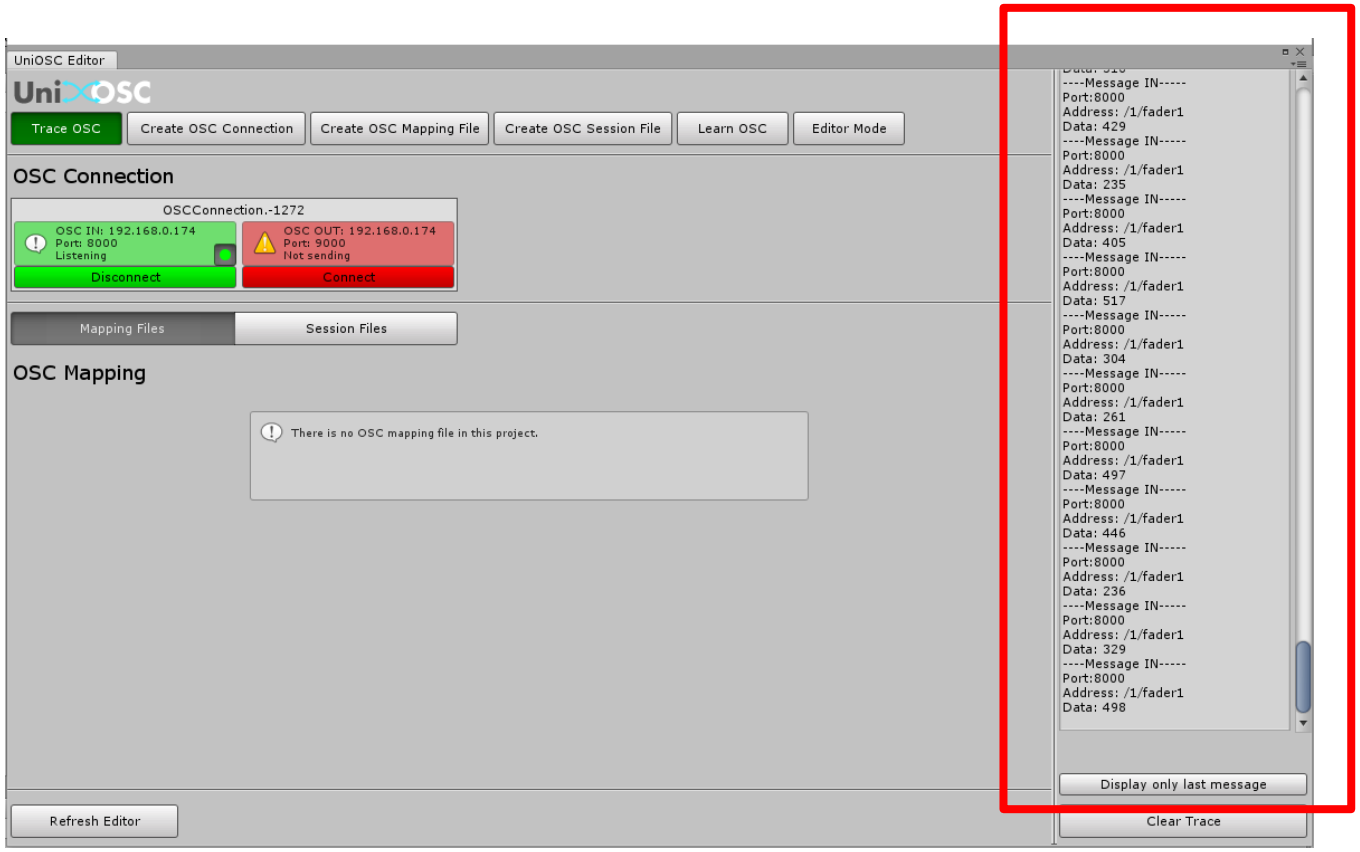
OSC on Unity (UniOSC package)

You can setup the OSC input(PC) and output(iPad or Arduino…etc) IP and Ports in the Inspector.



OSC on Unity (UniOSC package)

The OSCEditor Window will show you the all Input and Output Message if there is any.



OSC on Unity (UniOSC package)

Receiver Script:

Assembly-CSharp | OSCReciever

```
using UnityEngine.UI;
using UniOSC;
using OSCsharp.Data;

public class OSCReciever :  UniOSCEventTarget {

    public GameObject controlObject;

    // Use this for initialization
    void Start () {

    }

    // Update is called once per frame
    void Update () {

    }

    public override void OnOSCMessageReceived(UniOSCEventArgs args)
    {
        OscMessage msg = (OscMessage)args.Packet;
        if (msg.Data.Count < 1) return;

        int scale = (int)msg.Data[0];
        controlObject.transform.localScale = new Vector3((float)scale / 1023.0f, (

    }
}
```

Inheritance

Receive over here

OSC on Unity (UniOSC package)

Sender Script:

```
using UnityEngine.UI;
using OSCsharp.Data;
using UniOSC;

0 references
public class OSCSender : UniOSCEventDispatcher
{
    public Slider mainSlider;

    14 references
    public override void Awake()
    {
        base.Awake();
    }

    16 references
    public override void OnEnable()
    {
        //Here we setup our OSC message
        base.OnEnable();
        ClearData();
        //now we could add data;
        AppendData(123);//
        //AppendData(123f);
        //AppendData("MyString");
    }
    16 references
    public override void OnDisable()
    {
        //Don't forget this!!!!
        base.OnDisable();

    }
```

Inheritance

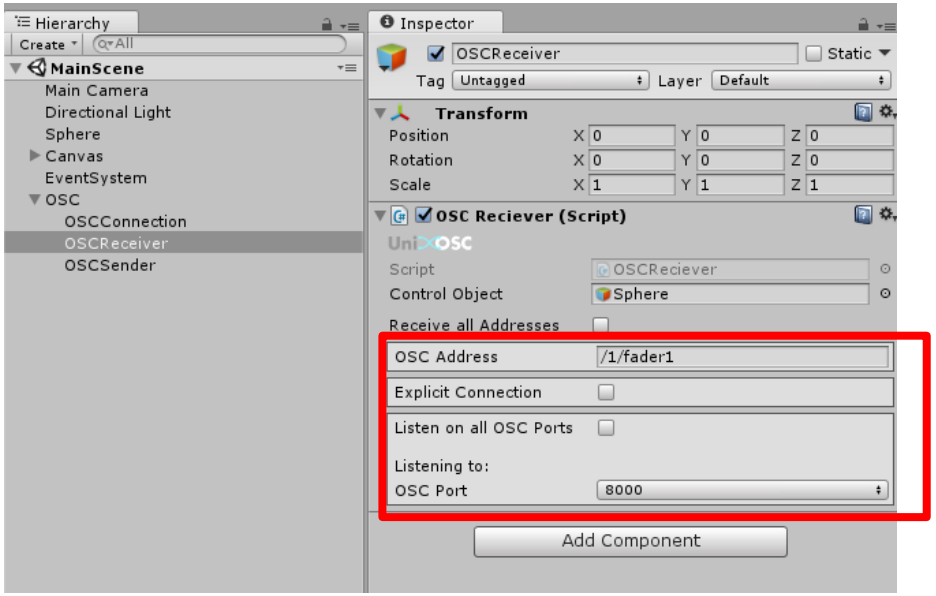
Initial Data

```
public void MySendOSCMessageTriggerMethod(){

    //Here we update the data with a new value
    //OscMessage msg = null;
    if (_OSCeArg.Packet is OscMessage)
    {
        //message
        OscMessage msg = ((OscMessage)_OSCeArg.Packet);
        _updateOscMessageData(msg);

    }
    else if (_OSCeArg.Packet is OscBundle)
    {
        //bundle
        foreach (OscMessage msg2 in ((OscBundle)_OSCeArg.Packet).Messages)
        {
            _updateOscMessageData(msg2);
        }
    }

    //Here we trigger the sending
    _SendOSCMessage(_OSCeArg);


}

2 references
private void _updateOscMessageData(OscMessage msg)
{
    msg.UpdateDataAt(0, (int)mainSlider.value);
    //msg.UpdateDataAt(1, dynamicFloatValue);
    //msg.UpdateDataAt(2, dynamicStringValue);
}
```

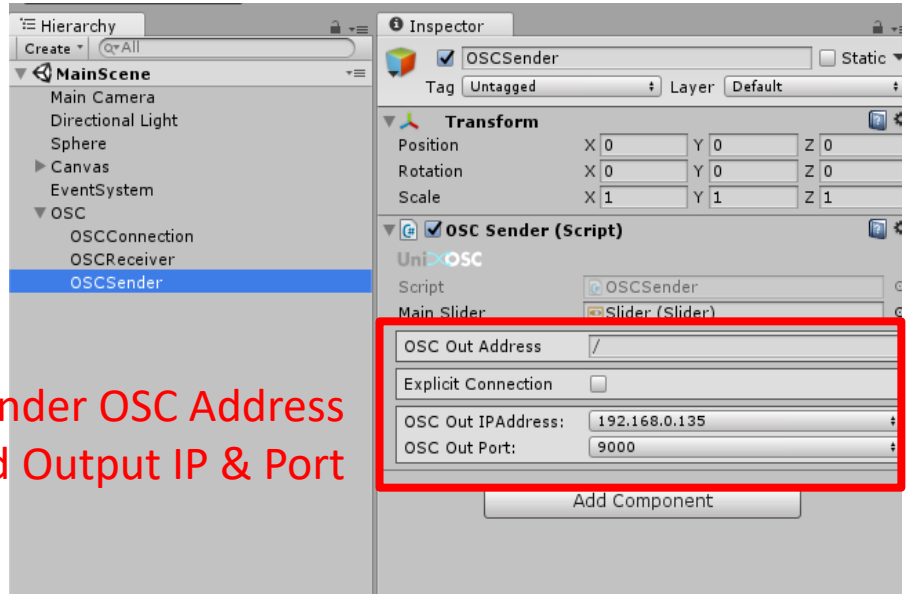
Trigger Sender

Update the OSC Message

OSC on Unity (UniOSC package)



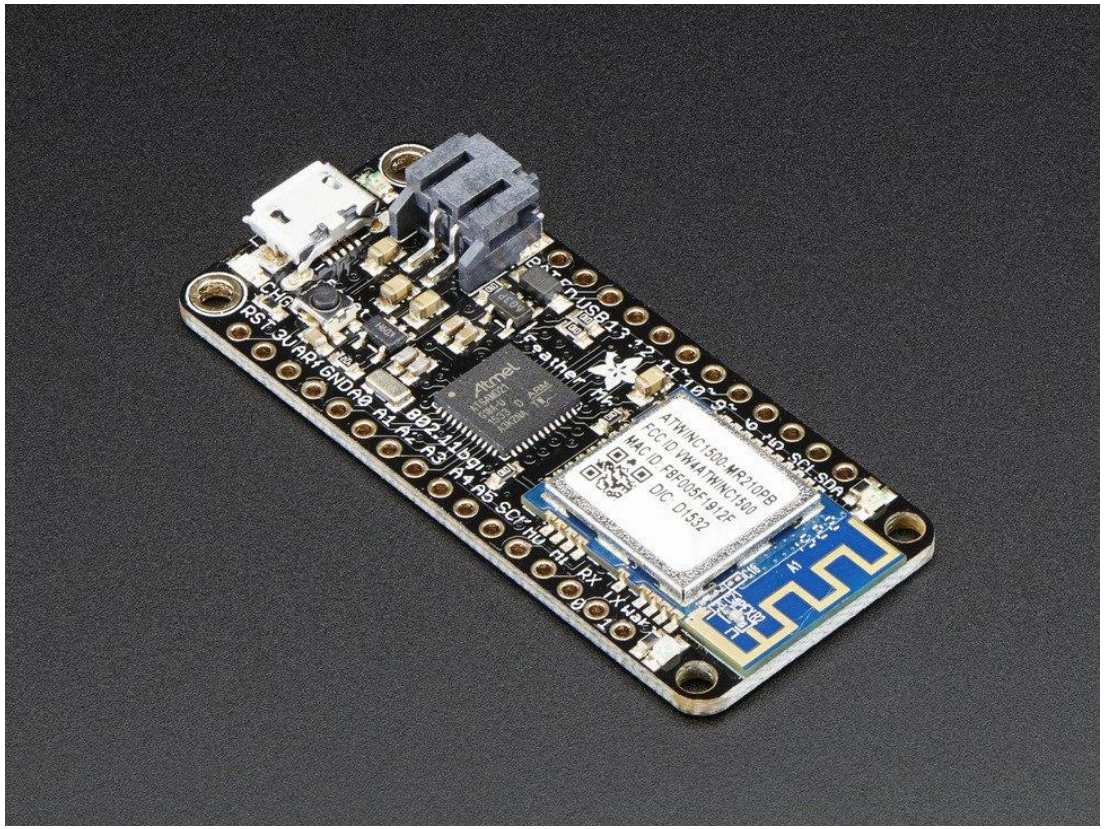
Setup the Receive Address and Listen Port



Setup the Sender OSC Address and Output IP & Port

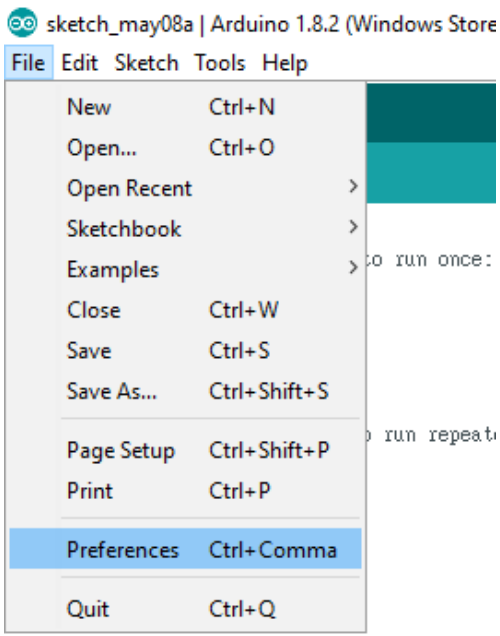
OSC on Arduino

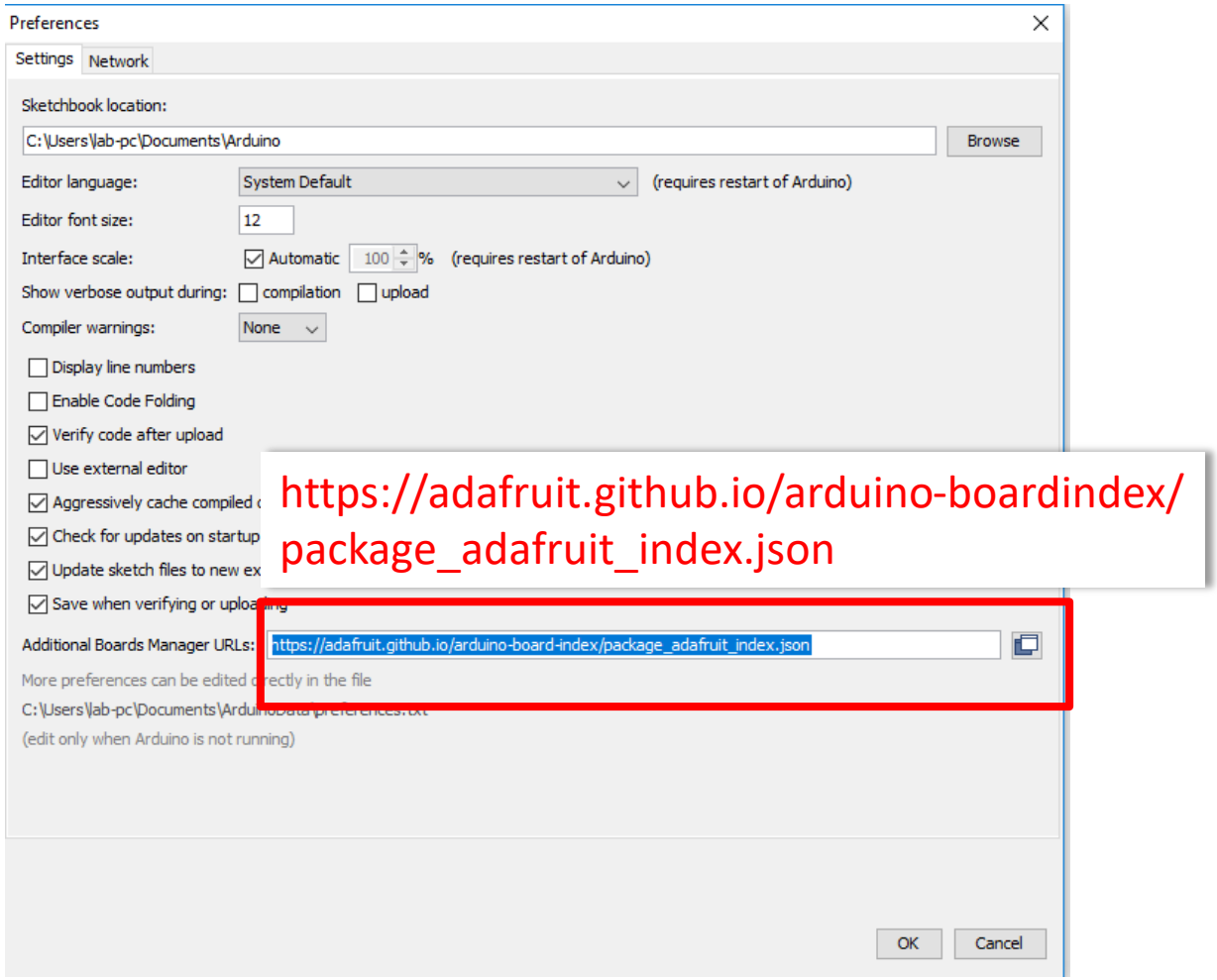
In this project, we will use the Adafruit feather m0 (wifi-ready arduino) for the OSC transmission.



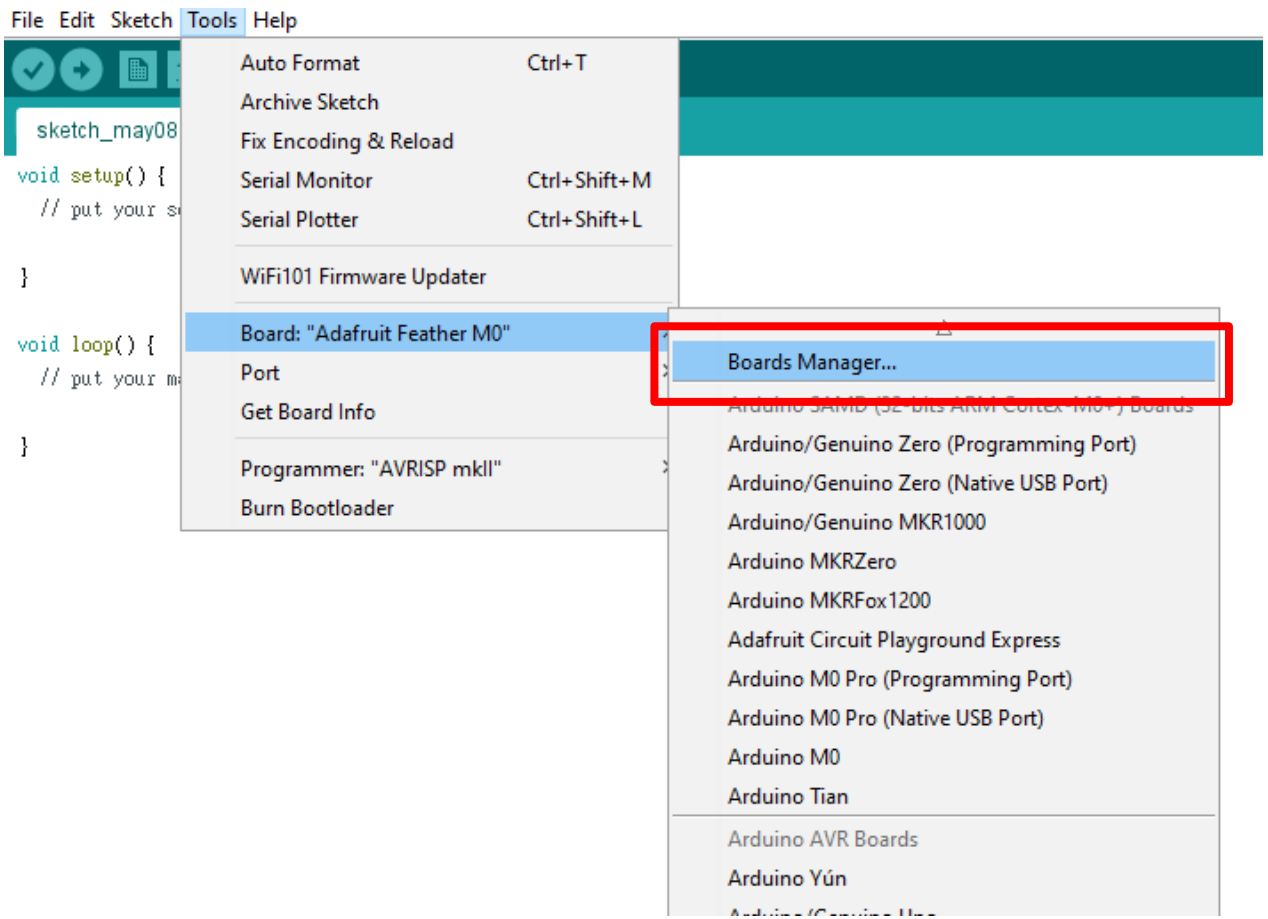
OSC on Arduino

Install the Adafruit wifi board package

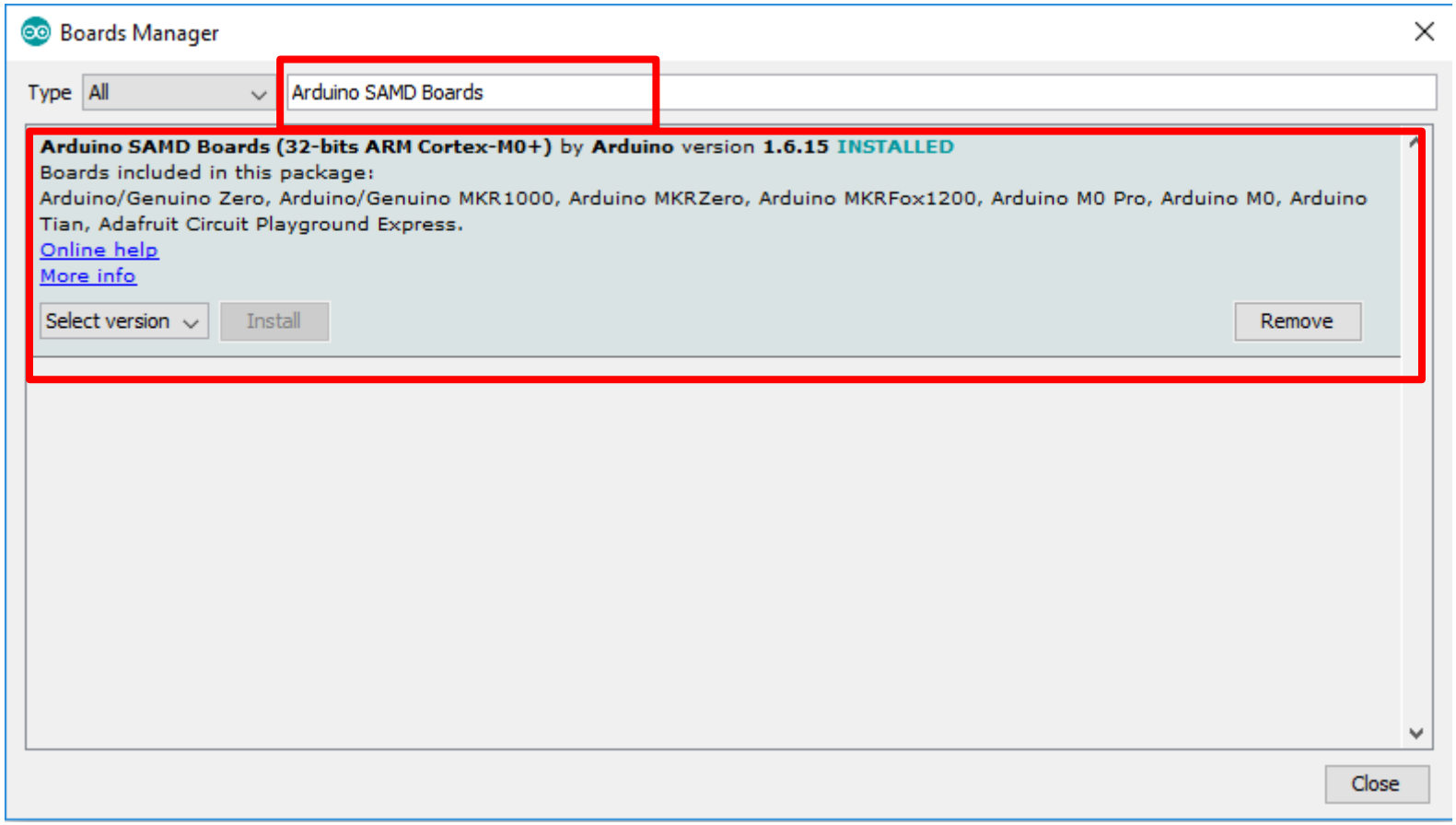




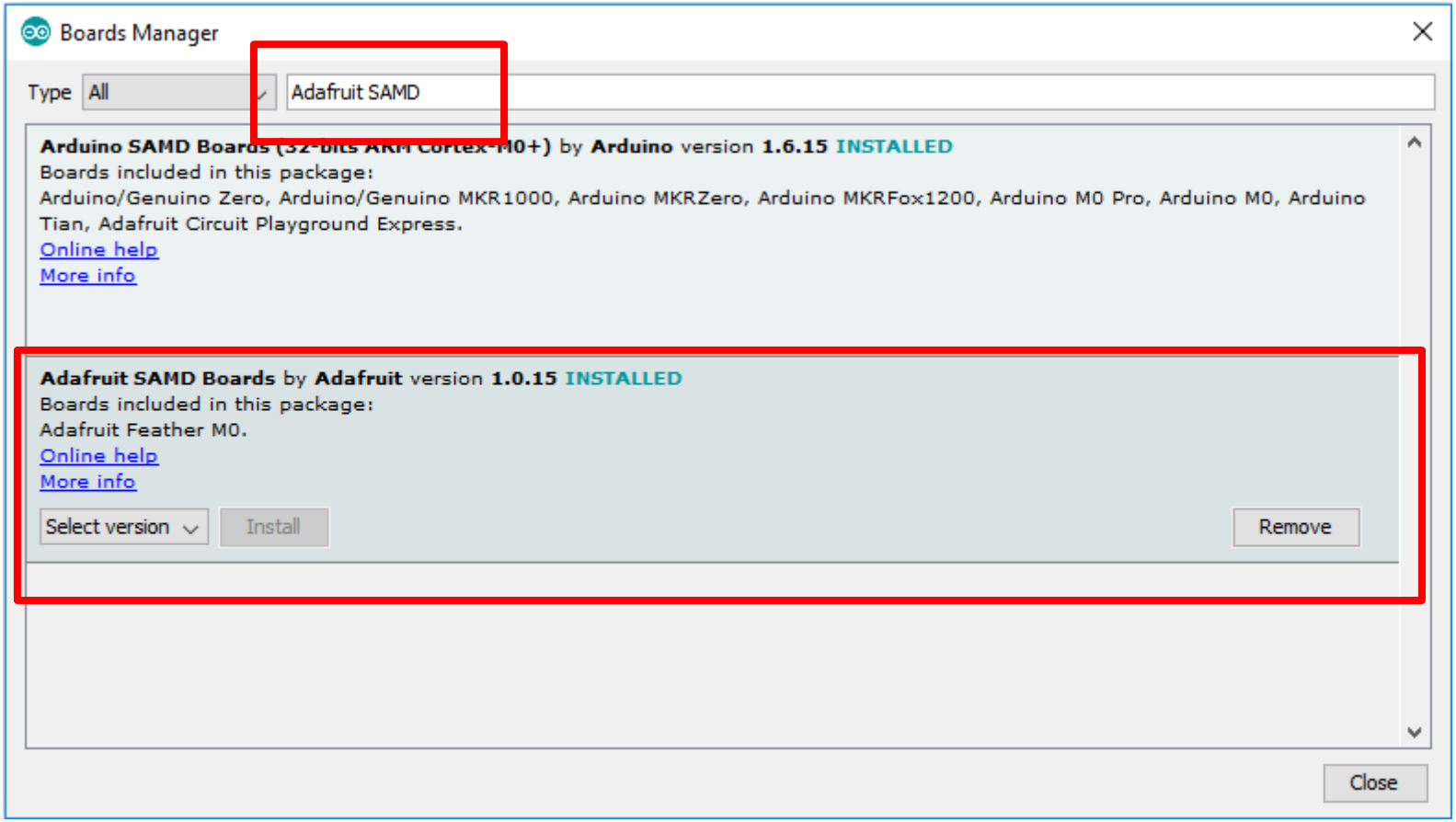
OSC on Arduino



OSC on Arduino



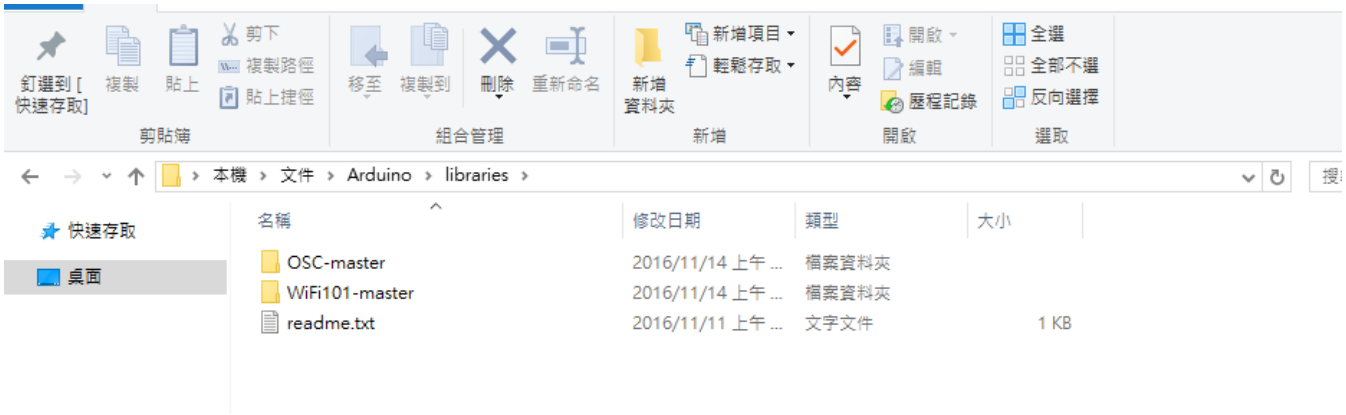
OSC on Arduino



OSC on Arduino

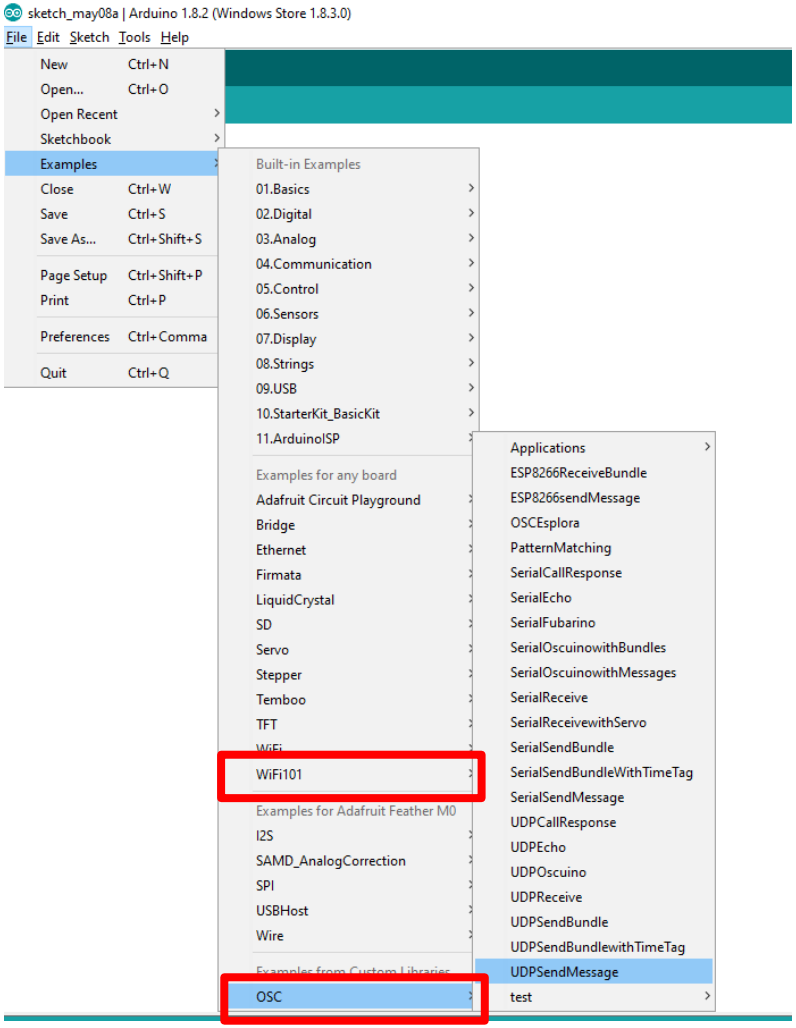
Install libraries
Wifi101 https://github.com/arduino-libraries/WiFi101
OSC https://github.com/CNMAT/OSC

Unzip them in the folder “Arduino/libraries/”



OSC on Arduino

After installing, you will find the example of Wifi101 and OSC in Arduino IDE.



OSC on Arduino

Open file “arduino_osc.ino”, this is a sample code for OSC connection between Unity and Arduino.

```
#include <OSCMessage.h>
#include <OSCBundle.h>
//input output pint
int ledPin = 10;
int sensorPin = A2;

int status = WL_IDLE_STATUS;
char ssid[] = "NextInterfaces Lab"; // your network SSID (name)
char pass[] = "nextinterfaces"; // your network password (use for WPA, or use as key for WEP)
int keyIndex = 0; // your network key Index number (needed only for WEP)

//IP setup
IPAddress sendToUnityPC_Ip(192, 168, 0, 174); // UnityPC's IP
unsigned int sendToUnityPC_Port = 8000; // UnityPC's listening port
unsigned int listenPort = 9000; // local port to listen on

char packetBuffer[255]; //buffer to hold incoming packet
char ReplyBuffer[] = "acknowledged"; // a string to send back
WiFiUDP Udp_send;
WiFiUDP Udp_listen;
```

Network IP and password

Unity IP and Port

Local Port to listen on

OSC on Arduino

```
}
void loop() {
    // Write
    OSCMessage msg("/1/fader1");
    msg.add((int)analogRead(sensorPin));
    Udp_send.beginPacket(sendToUnityPC_Ip, sendToUnityPC_Port);
    msg.send(Udp_send);
    Udp_send.endPacket();
    msg.empty();
    delay(10);
    // Read
    OSCMessage messageIn;
    int size;
    if( (size = Udp_listen.parsePacket())>0)
    {
        while(size--)
            messageIn.fill(Udp_listen.read());
        if(!messageIn.hasError()){
            int data = messageIn.getInt(0);
            Serial.println(messageIn.size());
            Serial.println(data);
            // setting intensity of the LED
            int fadeValue = data;
            analogWrite(ledPin, fadeValue);
        }
    }
}
void printWifiStatus() {
```

Send Message

Receive Message