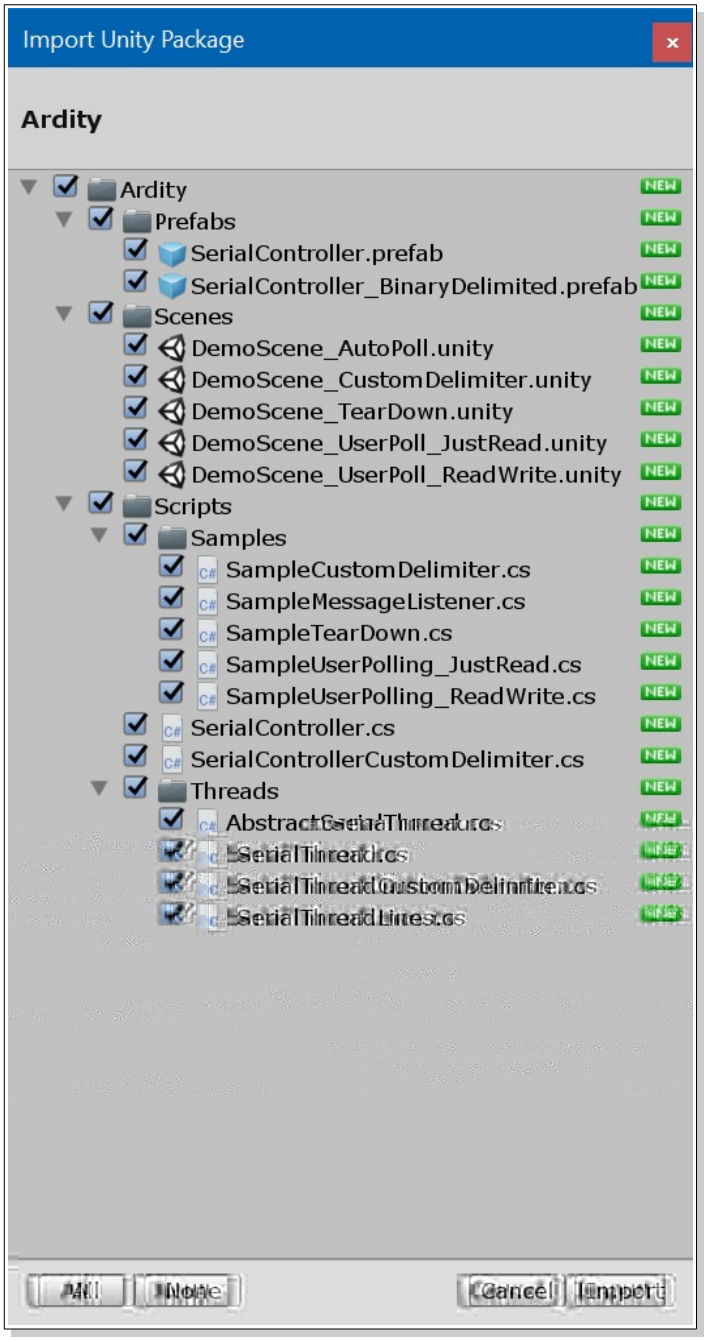
Setup Guide

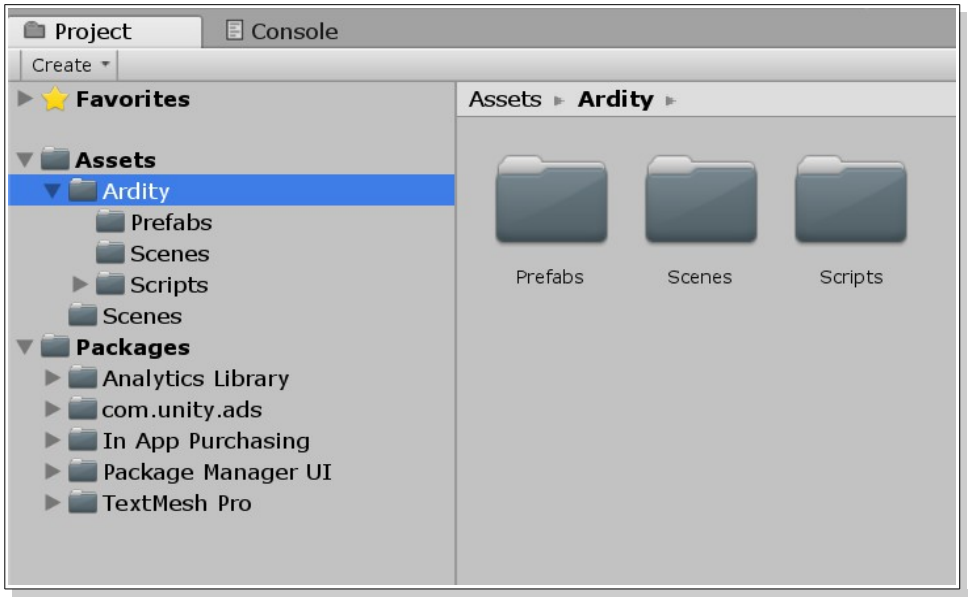
https://ardity.dwilches.com

Setup Guide

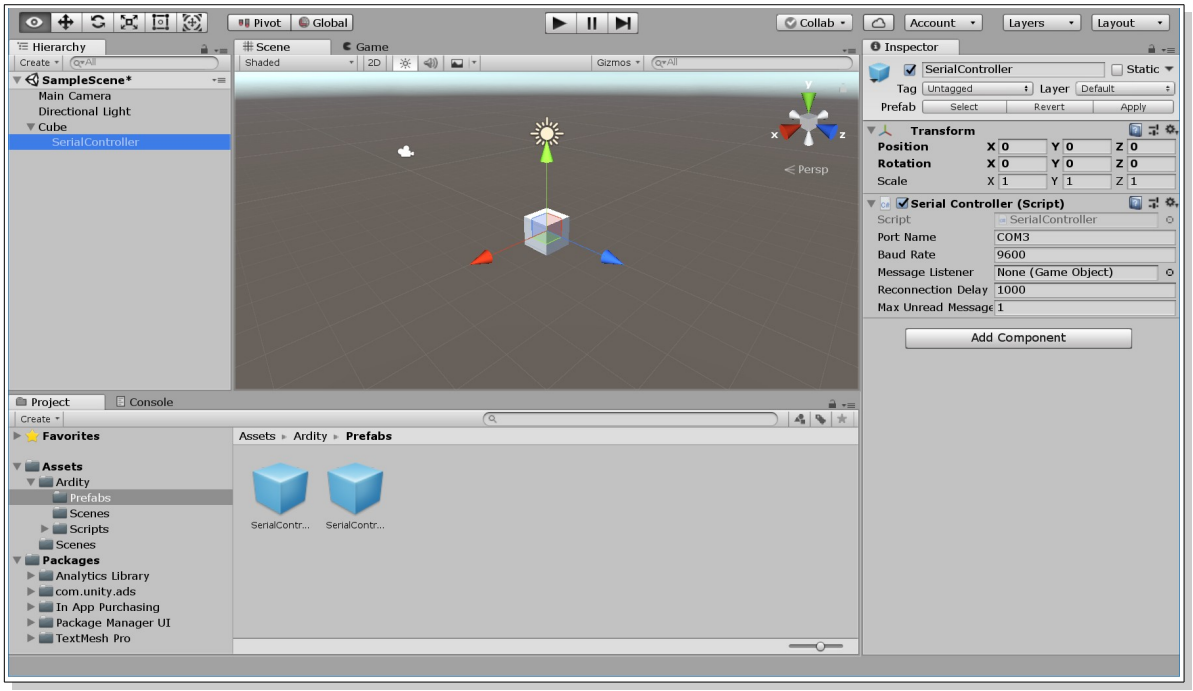
1. Double-click on the file *Ardity.unitypackage* or download it from the Asset Store, and click on the Import button:



2. The package files will be imported in the Ardity folder:



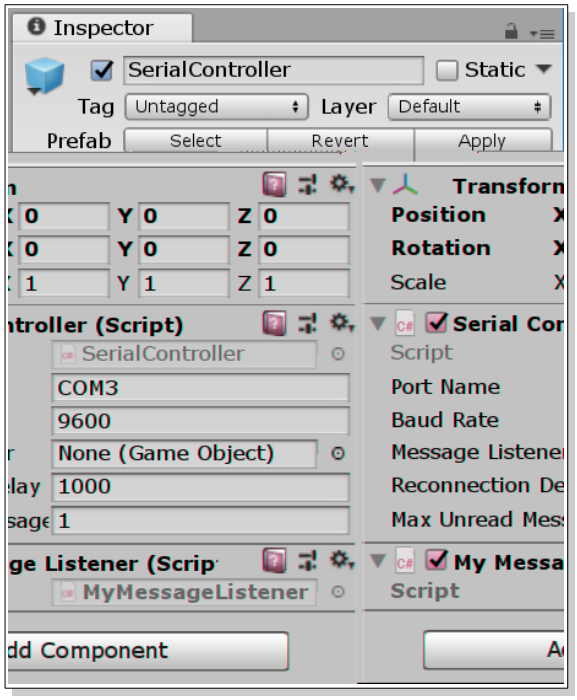
3. There are some sample scenes inside the Unity package, they are a good starting point, but basically what you need to get *Ardity* up and running is adding the SerialController prefab to one of your GameObjects ("Cube" in this image):



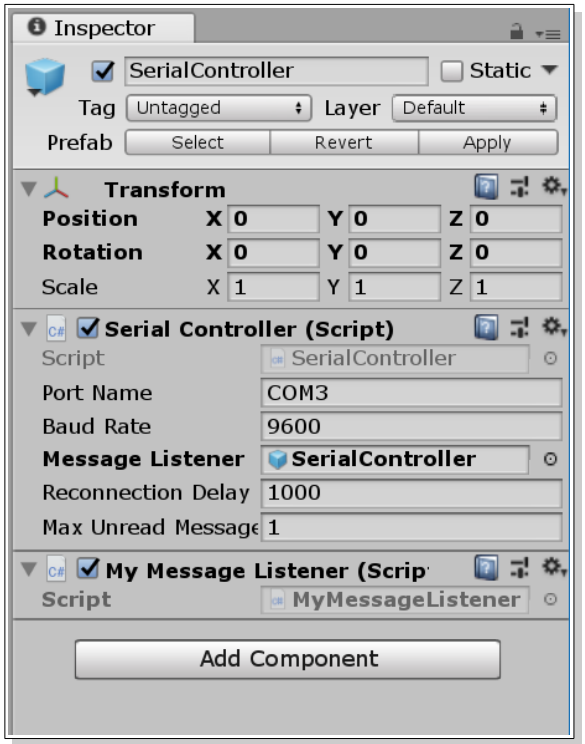
4. You have two options to get the events of your Arduino program at this point:
   - If you let the "Message Listener" field unset, then you need to poll Ardity each time you want to get a message.
   - If you want Ardity to automatically call your own callback whenever a message arrives, then set the “Message Listener” field to a GameObject that has the following methods “OnMessageArrived” and “OnConnectionEvent”.
5. In this manual we’ll follow the second option: creating a Message Listener.
6. Right click on your Project’s “Assets” folder, select “Create” and then “C# Script”. We’ll call our file “MyMessageListener”.
7. Double click on the new file and write the following inside it:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class MyMessageListener : MonoBehaviour {

   // Use this for initialization
   void Start () {
   }

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

   }

   // Invoked when a line of data is received from the serial device.
   void OnMessageArrived(string msg)
   {
      Debug.Log("Arrived: " + msg);
   }

   // Invoked when a connect/disconnect event occurs. The parameter 'success'
   // will be 'true' upon connection, and 'false' upon disconnection or
   // failure to connect.
   void OnConnectionEvent(bool success)
   {
      Debug.Log(success ? "Device connected" : "Device disconnected");
   }
}
```

8. Save the file and go back to Unity.
9. Now add the script "MyMessageListener" to your "SerialController" GameObject (it can be any GameObject actually, but in this example we'll reuse the SerialController):

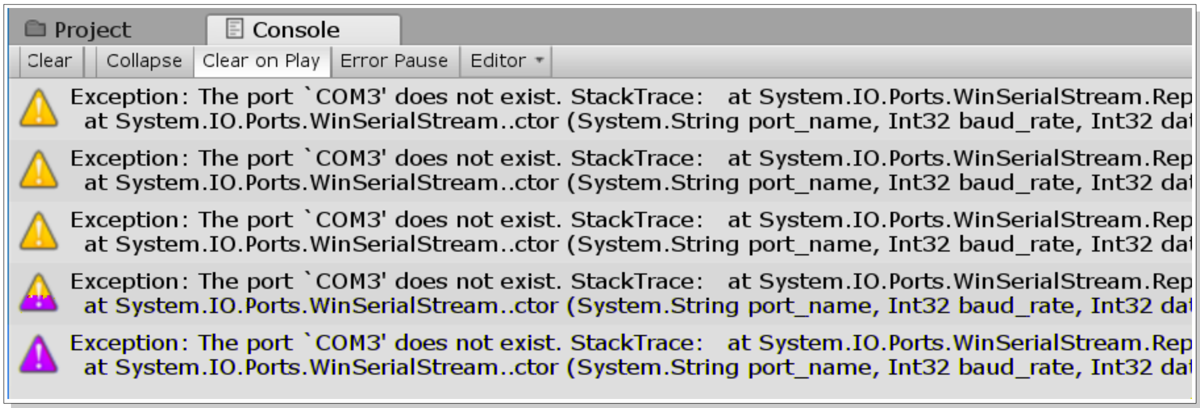


10. Now set the "Message Listener" field to the GameObject that contains the "MyMessageListener" script:



11. Run your scene and check the console.

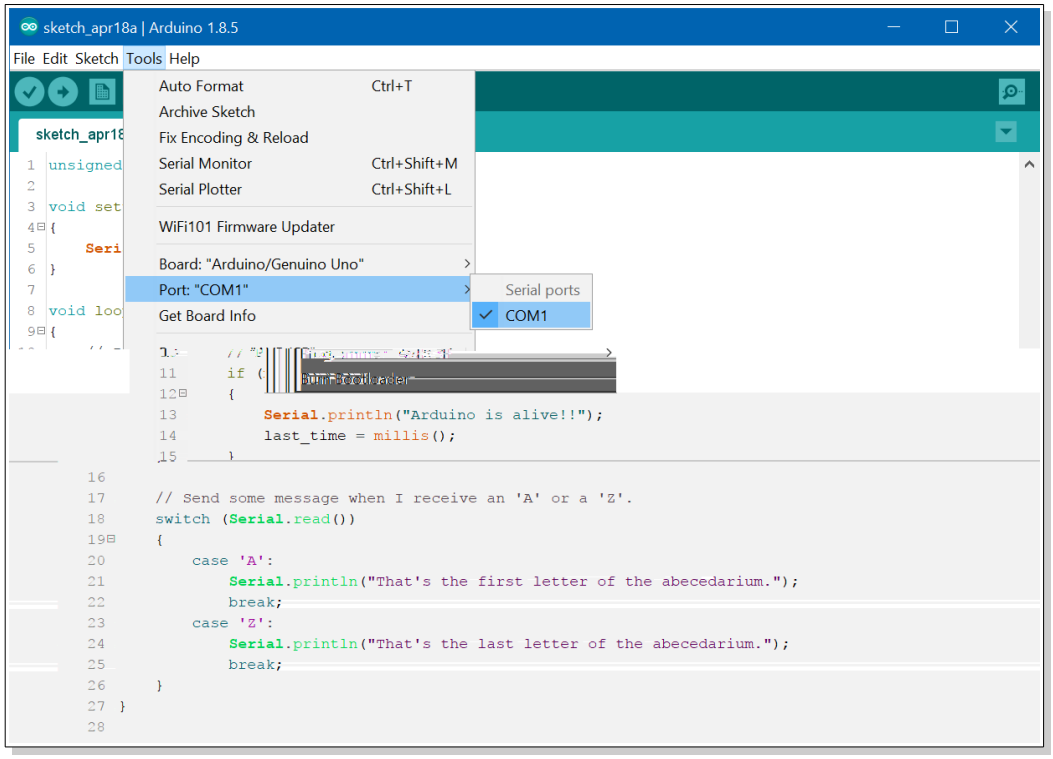
12. At this point, Ardity is trying to read from the COM3 port, you should see a series of warnings like in the following image if there is no Arduino device connected on COM3:



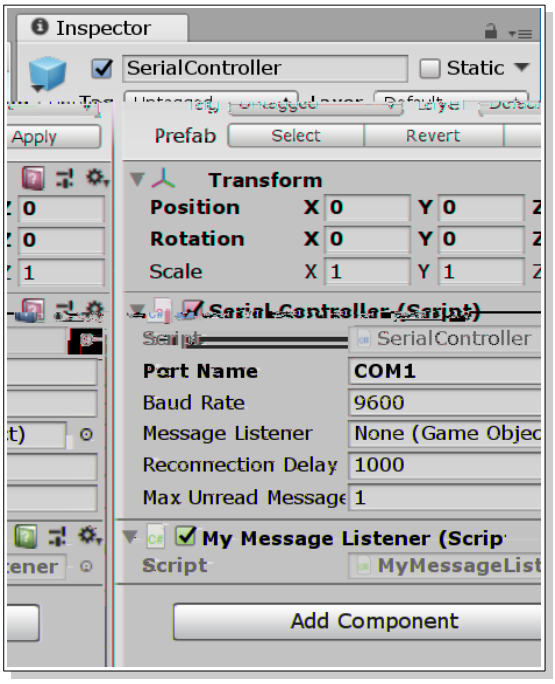
13. To test that Ardity indeed works, copy the following program in the Arduino IDE, and upload it to your Arduino:

```
unsigned long last_time = 0;

void setup() {
  Serial.begin(9600);
}

void loop()
{
  // Print a heartbeat
  if (millis() > last_time + 2000)
  {
    Serial.println("Arduino is alive!!");
    last_time = millis();
  }

  // Send some message when I receive an 'A' or a 'Z'.
  switch (Serial.read())
  {
    case 'A':
      Serial.println("That's the first letter of the abecedarium.");
      break;
    case 'Z':
      Serial.println("That's the last letter of the abecedarium.");
      break;
  }
}
```

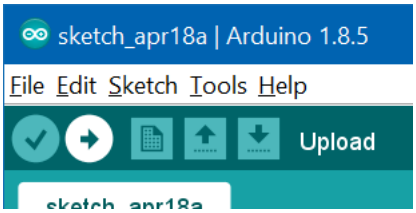
14. Note that COM3 may not be the port that your system assigns to your Arduino, validate it in the Arduino IDE in the “Tools” menu and then “Port: COMX” submenu:



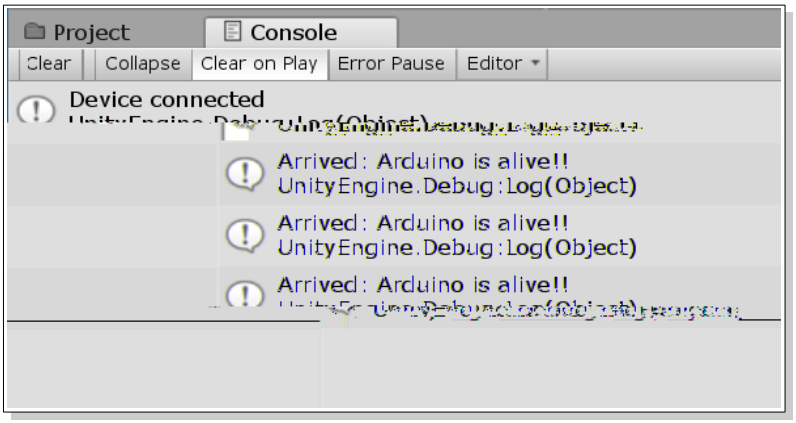
15. As in this example the port is COM1, let’s change it in the Unity project to match:



16. Now run the Arduino code by pressing the “Upload” button:



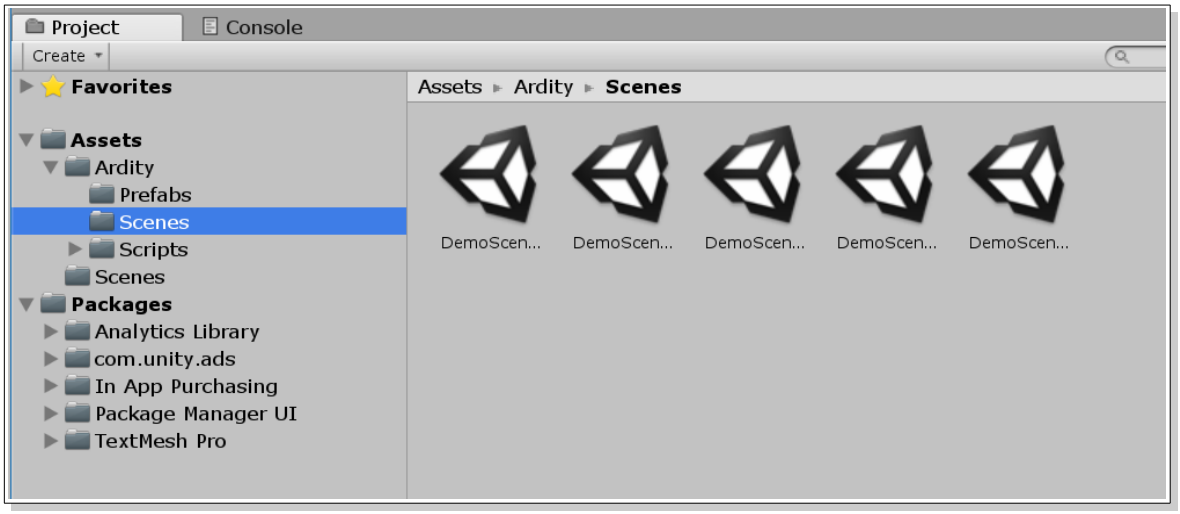
17. And then run your Unity scene. Look at the console again:



18. It works! The messages from the Arduino are being received by Ardity.

Where to go now?

1. From now on, you can check the remaining sample scenes where you can see sample implementations of more advanced features:



2. These sample scenes cover the following topics:
   - How to handle messages that are not delimited by newlines
   - How to send data back to the Arduino
   - How to register a “TearDown” callback

FAQ

1. **How do I do X?**

   If there is something that is troubling you, create an Issue in GitHub:

   https://github.com/DWilches/Ardity/issues

   And I'll do my best to get back to you (normally within some hours).

2. **I'm getting the error "The type or namespace name 'Ports' does not exist"**

   By default, Unity uses a subset of the .NET framework, so if you get this error message:

   ```
   Assets/Ardity/Scripts/SerialThread.cs(9,17): error CS0234:
   The type or namespace name 'Ports' does not exist in the
   namespace 'System.IO'. Are you missing an assembly
   reference?
   ```

   It's because the current "API Compatibility Level" of your Unity project is set to ".NET 2.0 Subset", which doesn't contain the classes necessary for serial communication.

   To solve the problem, go to **Edit** → **Project Settings** → **Player**, and under **"Other Settings"** find an option that reads **"Api Compatibility Level"** and change it from **".NET 2.0 Subset"** to **".NET 2.0"**.

3. **Does this library only support communication based on lines of strings?**

   Yes, but you can change it easily to support your own protocol. Check the **SerialThread** script.