**Arduino 101 Setup with Unity**

**Initial Setup:**

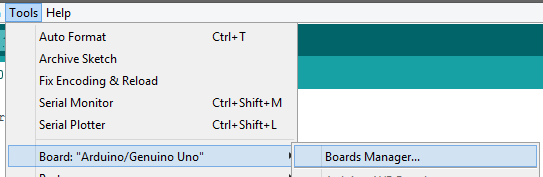
Plug your Arduino board into your computers USB port.

Download Arduino software from <https://www.arduino.cc/en/Main/Software>

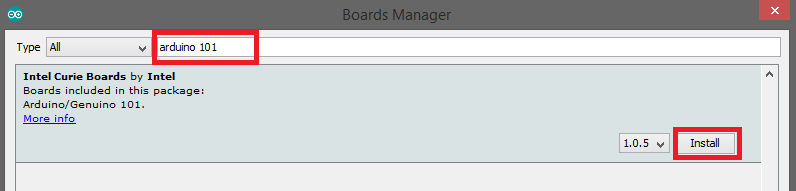
Source code for Curie libraries <https://github.com/01org/corelibs-arduino101>

Arduino Reference Page <https://www.arduino.cc/en/Reference/HomePage>

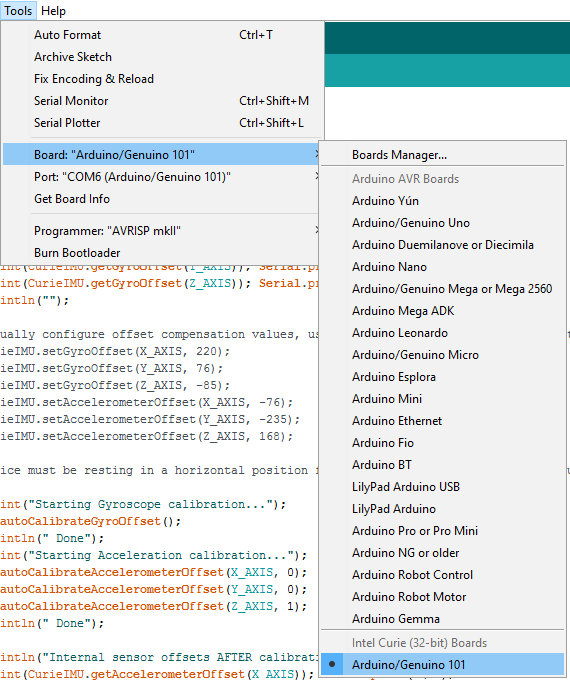
Once installed go to Tools > Boards Manager



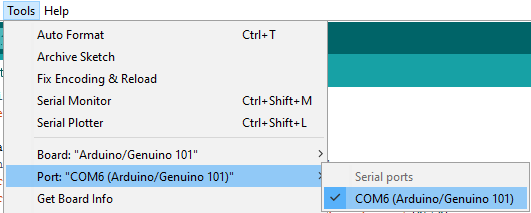
Search for Arduino 101, when located, select it in the list and click install.



After the install completes choose Board > Arduino/Genuino 101 from the menu.

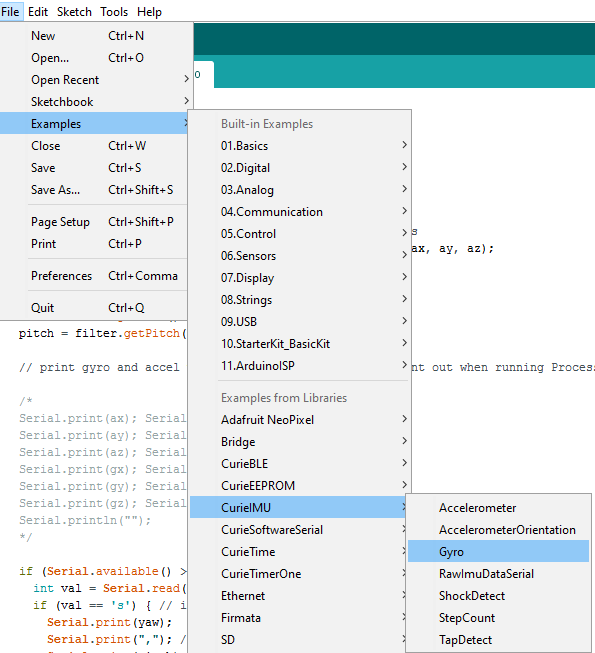


Choose Tools > Port from the main menu and select the port for your Arduino, if you are unsure, unplug your device and see which port is removed from the list.

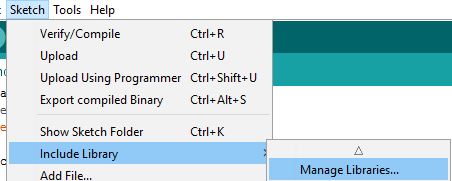


**Deplying Code:**

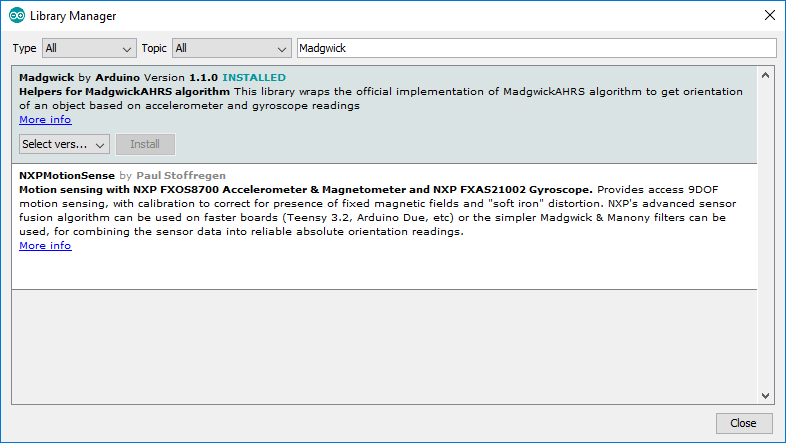
To run a sample program on your Arduino select File > Examples from the main menu, there should be several Curie specific examples available. You can also use the ArduinoUnitySample.Ino file from the Sketch folder in the CurieUnitySandbox project.



To run the Gyrometer or Unity example you will also need to install a library called MadgwickAHRS, from the main menu choose Sketch > Include Library > Manage Libraries…



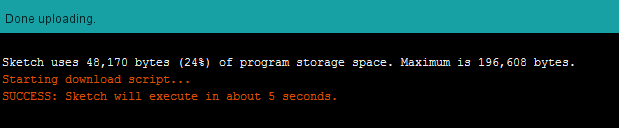
In the Library Manager search for Madgwick. Select the Madwick library and click Install.



You can use this process to install additional libraries as needed. Now that you have everything configured you can run your program by clicking the Upload Icon in the Toolbar.

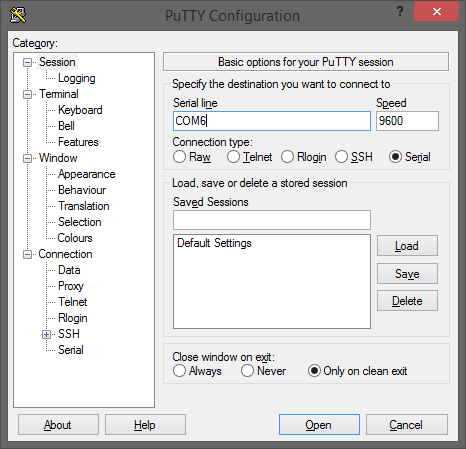


You should see output in the console similar to the following image.

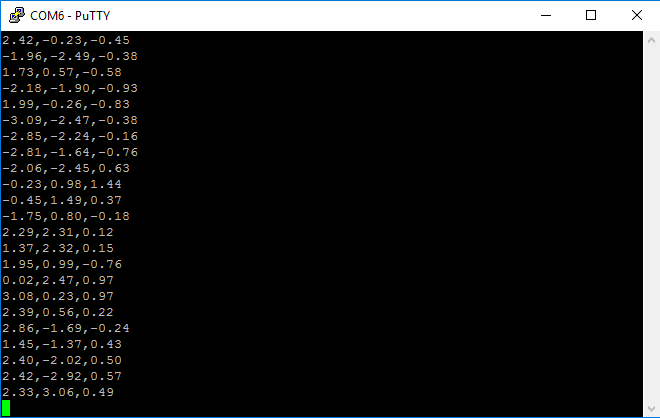


**Debugging:**

If you want to view communication through your COM port you can use a client like PuTTY to connect. You can connect with PuTTY by choosing Serial as your connection type. You’ll also need to add the name of your COM port to the Serial Line field and a baud rate that matches the baud rate in the Speed field.

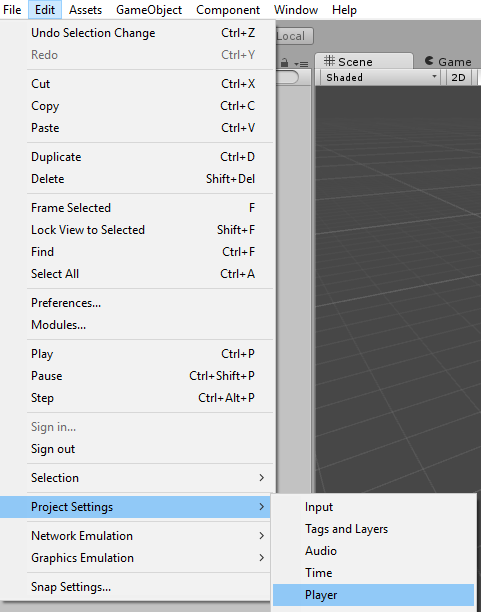


You can now send commands and receive output from your device. The following example is a sample of the gyrometer output.

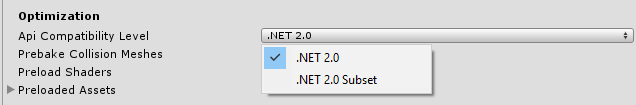


**Unity Setup:**

You can also communicate with your device over serial protocol in any language. In Unity I created a C# script called SerialGyro it opens a serial port and communicates with the connected device. In order to use the correct libraries you need to go to Edit > Project Settings > Player. This option should be preconfigured in the UnityCurieSandbox project.



In the player settings Inspector find Api Compatibility Level and choose .NET 2.0



**Unity Scripting:**

There are two sample scripts in the Assets folder of the UnityCurieSandbox project, SerialGyro and SerialAccelerometer. To create your own script you will need to reference System.IO.Ports.

using System.IO.Ports;

Create a new SerialPort property in your class.

private SerialPort serial;

In the Start() method you can open the serial port by creating a SerialPort object and calling the Open() function.

// Use this for initialization

void Start () {

// Setup your port object.

serial = new SerialPort(

SerialPort.GetPortNames()[0], 9600, Parity.None, 8, StopBits.One

);

// Set additional paramters.   
serial.DtrEnable = true;

serial.RtsEnable = true;

serial.ReadTimeout = 1000;

//allow time for setup. Unity has issues without this.

Thread.Sleep(100);

// Open the port.

serial.Open();  
}

In your Update() method you can communicate with your device using the SerialPort object. Use Write() or WriteLine() to send a command to your device and either Read() or ReadLine() to read from the device.

void Update() {

// Send a command to your device.

serial.Write("s");

// Read Responses from the serial port.

var msg = serial.ReadLine();

// Parse the values from the response.

if (msg != string.Empty) {

string[] ypr = msg.Split(',');

yaw = float.Parse(ypr[0]); // convert to float yaw

pitch = float.Parse(ypr[1]); // convert to float pitch

roll = float.Parse(ypr[2]); // convert to float roll

}  
}

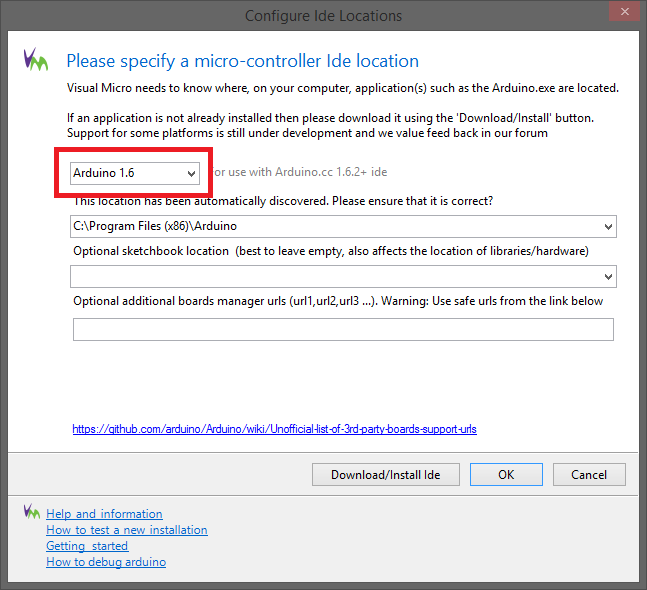
**Arduino Plugin for Visual Studio**

In addition to the Arduino IDE you can use a plugin for Visual Studio to deploy your sketches to your Arduino. The Arduino IDE is a prerequisite for installation so make sure you follow the steps in the Initial Setup section.

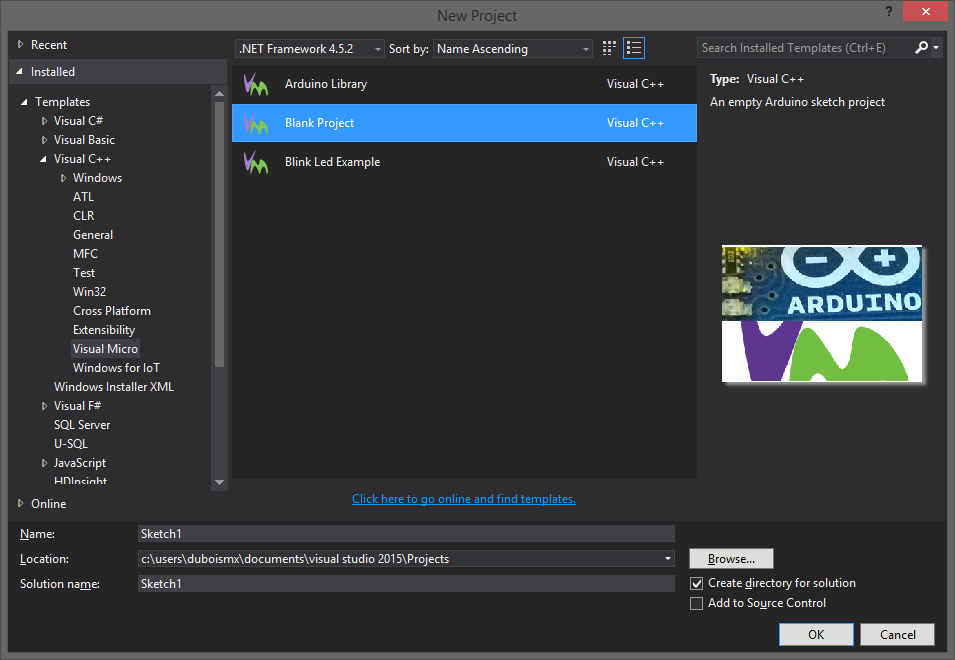
To install the plugin download the package from:

<https://visualstudiogallery.msdn.microsoft.com/069a905d-387d-4415-bc37-665a5ac9caba>

When you open Visual Studio again you’ll be prompted to configure the Arduino plugin. Choose the version of your Arduino IDE from the first dropdown and click OK.



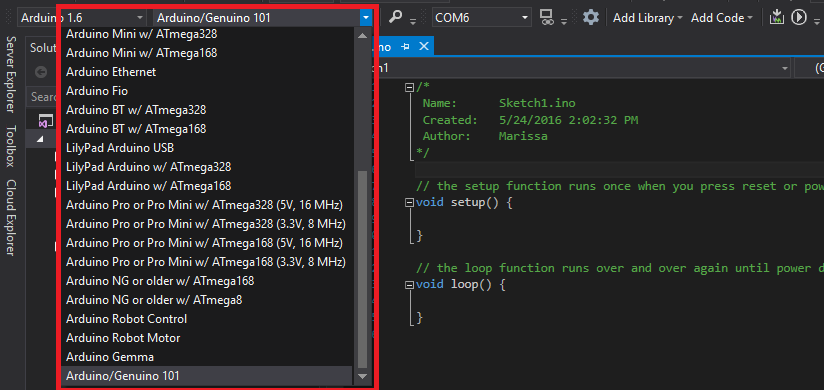
Create a new Visual Studio Project. You will now have a new Visual Micro option under the C++ node of the New Project dialog. Choose Blank Project to start from scratch.



Once the project opens you need to select your device and COM port. You should have a new toolbar in Visual Studio.



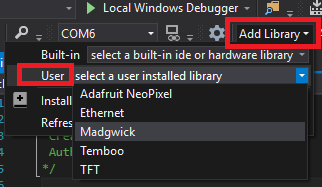
If you followed the Arduino 101 setup earlier you should have the Arduino/Genuino 101 available in the board Dropdown of the toolbar.



To the right of the board dropdown is the COM port dropdown. Select the correct COM port for your device.



To use Libraries like Madgwick you will need to click the Add Library dropdown and locate the previously installed library from the User dropdown.



Once your code is ready for Deployment you can choose the Build and Deploy Icon. This will build your Arduino Sketch, upload it to your device and start the Visual Studio Arduino Debugger.



By default your project will be configured for Debug release, this will load additional code onto your Arduino to assist with debugging. However, this can cause issues when trying to use your program outside of the visual studio environment. You just need to change your project from Debug to Release configuration before you deploy to your Arduino.

