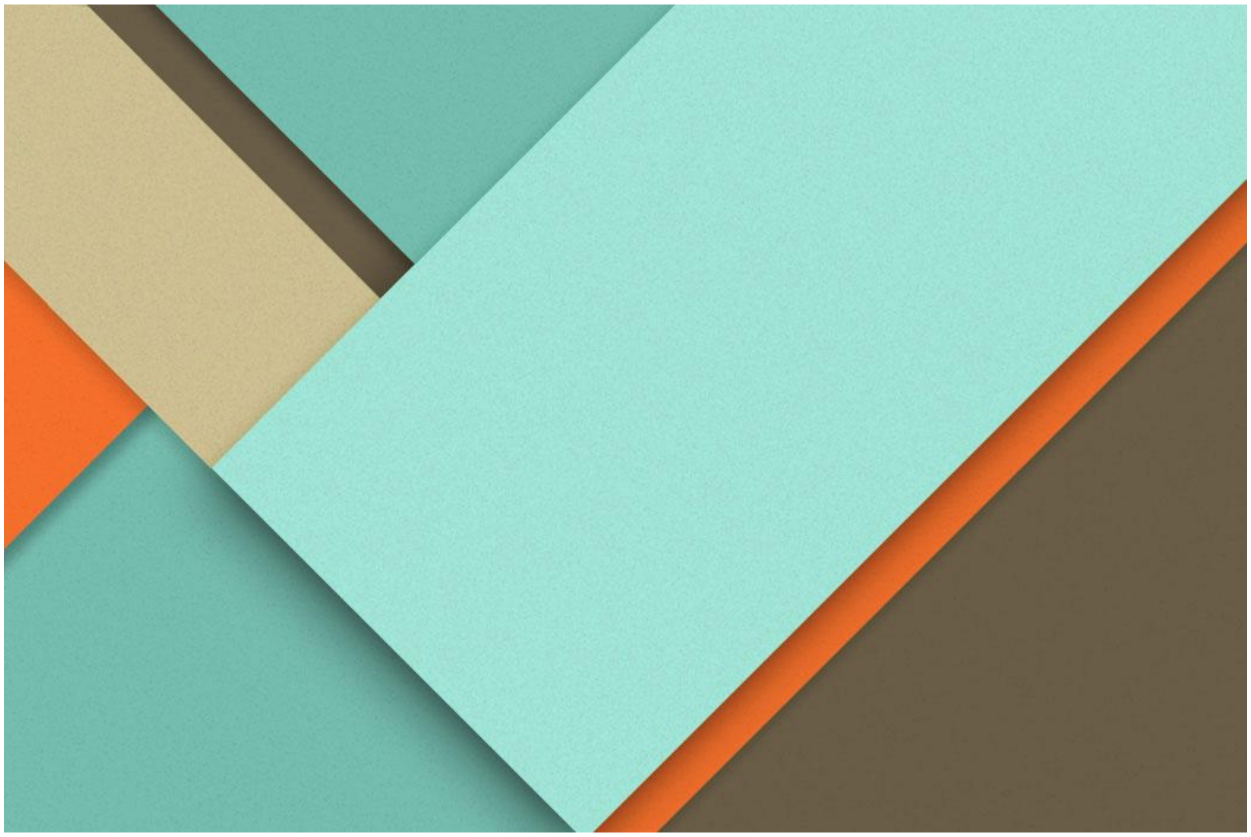


GEMINI. User Guide

Alvina Awan

Arosan Annalingam

Louis Härtel

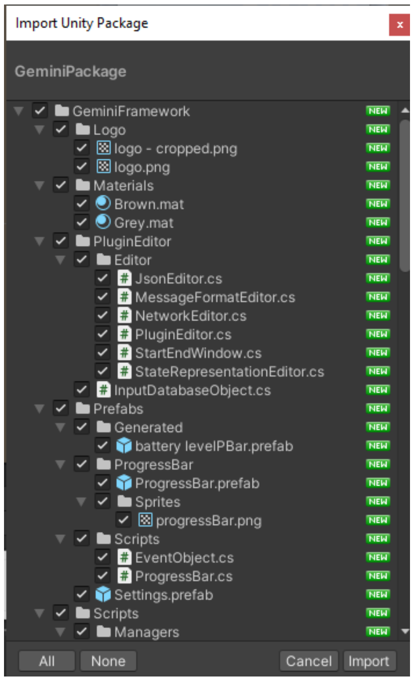
Manuel Pozor

Mohsin Muhammad

Yevheniia Kashperuk

Getting started

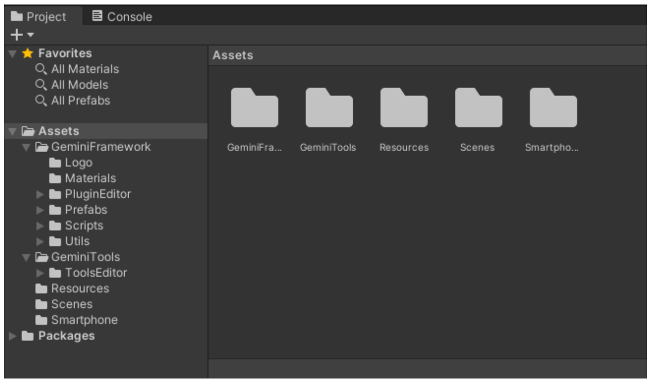
Start by adding **GeminiFramework** and **GeminiTools** folders to your projects asset folder.



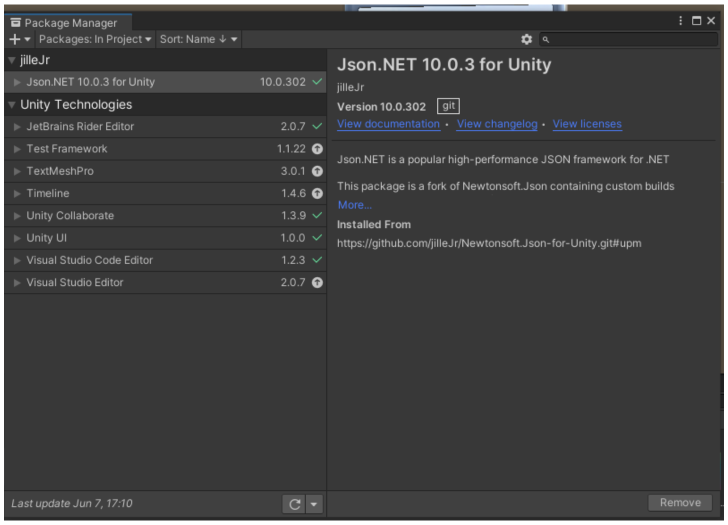
Import Unity Package

GeminiPackage

```
GeminiFramework
  Logo
    logo - cropped.png
    logo.png
  Materials
    Brown.mat
    Grey.mat
  PluginEditor
    Editor
      JsonEditor.cs
      MessageFormatEditor.cs
      NetworkEditor.cs
      PluginEditor.cs
      StartEndWindow.cs
      StateRepresentationEditor.cs
    InputDatabaseObject.cs
  Prefabs
    Generated
      battery levelPBar.prefab
    ProgressBar
      ProgressBar.prefab
      Sprites
        progressBar.png
    Scripts
      EventObject.cs
      ProgressBar.cs
    Settings.prefab
  Scripts
    Managers
```



To install needed requirements, simply open the Unity package manager through **Window > Package Manager**. Then click on **+** to add a package from this Git URL: https://github.com/jilleJr/Newtonsoft.Json-for-Unity.git#upm



jilleJr	
Json.NET 10.0.3 for Unity	10.0.302 ✓
Unity Technologies	
JetBrains Rider Editor	2.0.7 ✓
Test Framework	1.1.22
TextMeshPro	3.0.1
Timeline	1.4.6
Unity Collaborate	1.3.9 ✓
Unity UI	1.0.0 ✓
Visual Studio Code Editor	1.2.3 ✓
Visual Studio Editor	2.0.7

Json.NET 10.0.3 for Unity
jilleJr
Version 10.0.302 git
View documentation • View changelog • View licenses

Json.NET is a popular high-performance JSON framework for .NET

This package is a fork of Newtonsoft.Json containing custom builds
More...

Installed From
https://github.com/jilleJr/Newtonsoft.Json-for-Unity.git#upm

Last update Jun 7, 17:10

(You may need to reload Unity.)

Starting with an empty scene

Load the existing Gemini scene and customize it to your liking.

Modifying an already existing scene

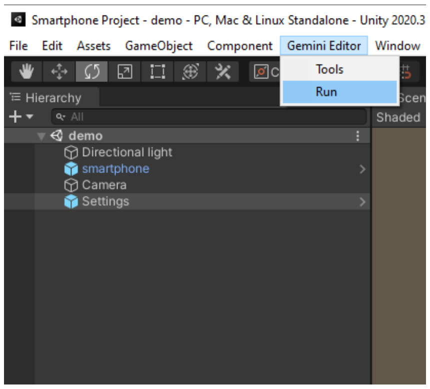
Import the manager scripts to your scene (simply drag&drop the **GeminiFramework/Prefabs/Settings** prefab).

Start your own DT configuration

Unity editor window

Here we show step by step how to use the Gemini plugin to import already existing Digital Twin configurations, to create a new setup and how to define and initialize new sensors. In the following documentation, we will go through each window of the plugin wizard.

First of all, open the setup wizard by clicking on ***Gemini Editor - Run*** in the Unity upper menu.

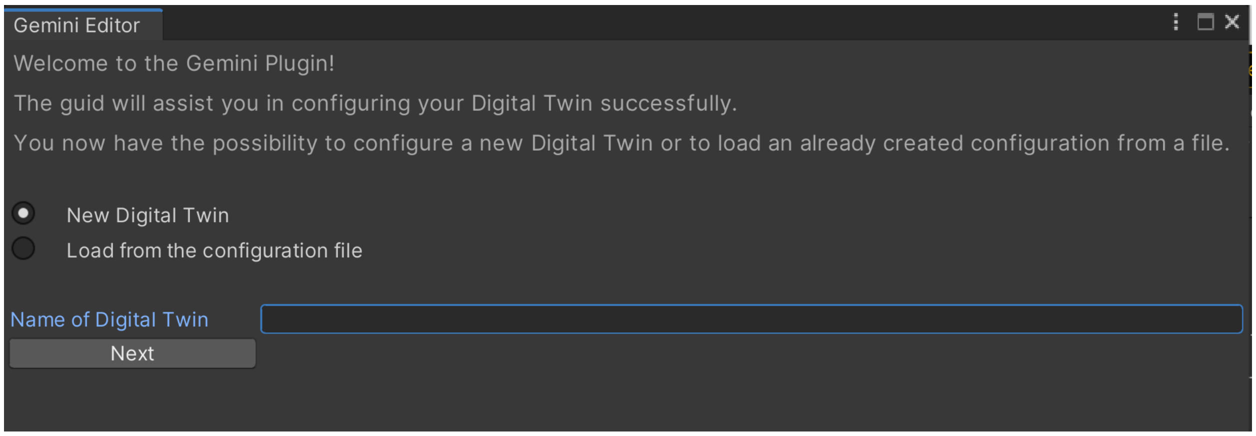


1. Import or create new configurations

Here you have the possibility to load already existing settings via a configuration file. Select one of the two points and click on the ***Next*** button.

If you are setting up Digital Twin for the first time, select the item ***New Digital Twin***.

If there is the configuration file from a pre-existing Digital Twin created with Gemini, the user can load his previous settings by selecting ***Load from the configuration file***.


Gemini Editor

Welcome to the Gemini Plugin!

The guid will assist you in configuring your Digital Twin successfully.

You now have the possibility to configure a new Digital Twin or to load an already created configuration from a file.

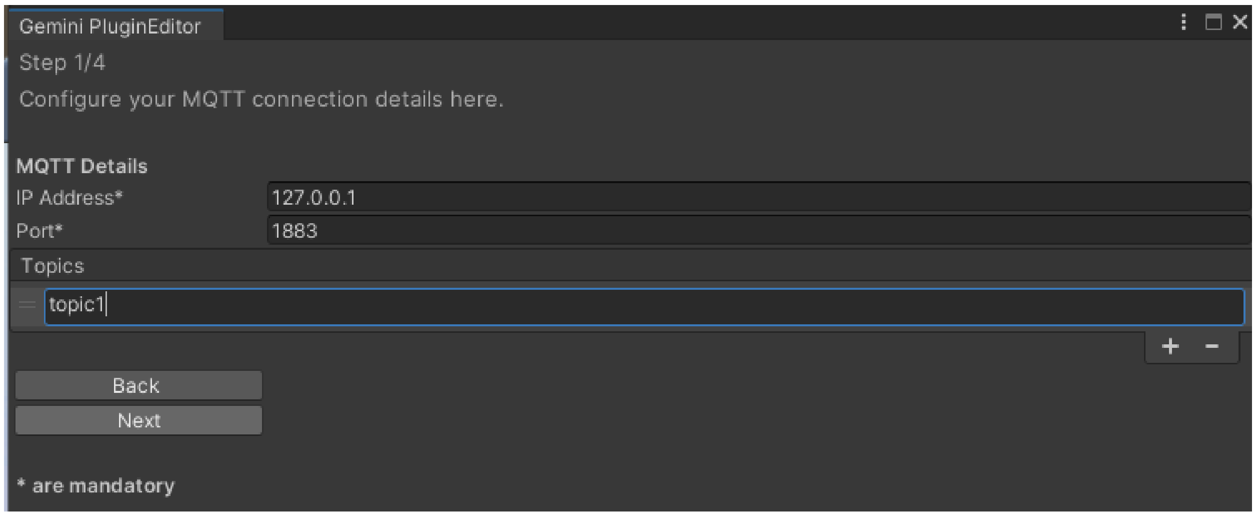
[✓ New Digital Twin]
Load from the configuration file

Name of Digital Twin

Next

2. Network protocol configuration

In the next step, the network protocol settings, that establish communication with the server, are configured. ***MQTT*** configuration data is required. Enter the IP address and the corresponding port of the MQTT broker in the IP ***Address*** and ***Port*** fields. Once the fields are filled click **Next** to go to the next step.


Gemini PluginEditor

Step 1/4

Configure your MQTT connection details here.

MQTT Details

IP Address* 127.0.0.1

Port* 1883

Topics

topic1

Back

Next

*** are mandatory**

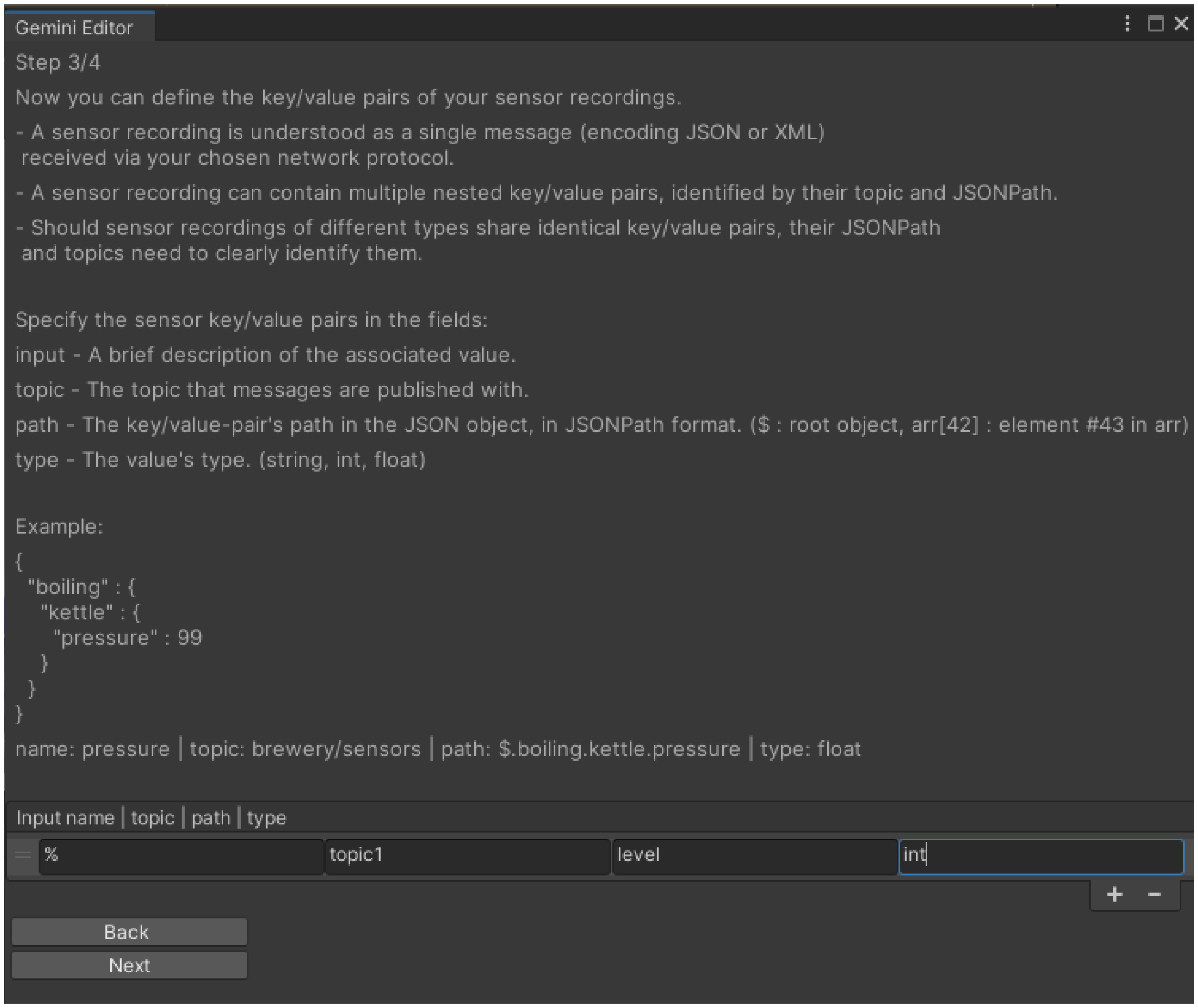
3. Select message format

Now we determine the format of the message received from the server. Currently only the ***JSON*** format is supported.

4. Select JSON format of arriving messages

The next step is to define the sensors received from the server. First, enter a unique name for the sensor in the ***Input*** name field. In the ***Path*** field, the path to the sensor object in the message from the server is defined. In the last ***Type*** field you define the type of value that a sensor takes (e.g. boolean, integer, object etc.).

Additional rows for sensors can be created using the ***Plus (+)*** button. The ***Minus (-)*** button deletes the last row.



Gemini Editor

Step 3/4

Now you can define the key/value pairs of your sensor recordings.

- A sensor recording is understood as a single message (encoding JSON or XML) received via your chosen network protocol.
- A sensor recording can contain multiple nested key/value pairs, identified by their topic and JSONPath.
- Should sensor recordings of different types share identical key/value pairs, their JSONPath and topics need to clearly identify them.

Specify the sensor key/value pairs in the fields:

input - A brief description of the associated value.

topic - The topic that messages are published with.

path - The key/value-pair's path in the JSON object, in JSONPath format. ($: root object, arr[42] : element #43 in arr)

type - The value's type. (string, int, float)

Example:

```
{
  "boiling" : {
    "kettle" : {
      "pressure" : 99
    }
  }
}
```

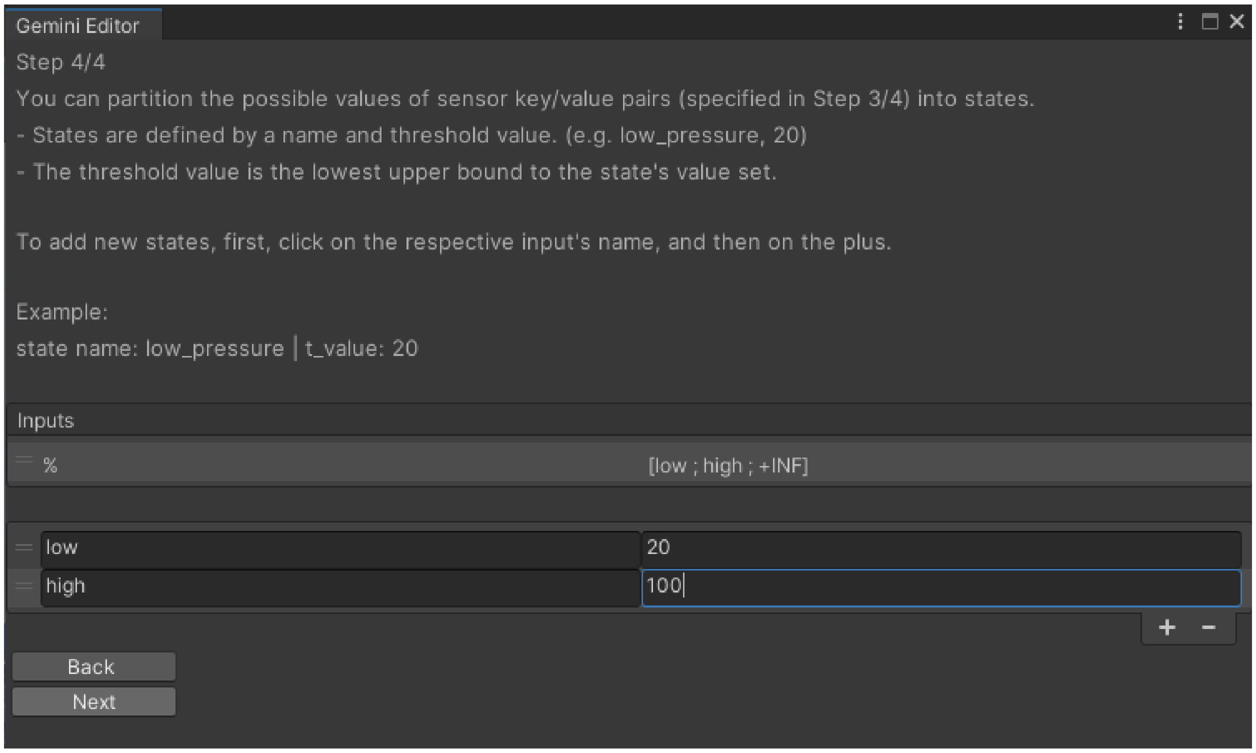
name: pressure | topic: brewery/sensors | path: $.boiling.kettle.pressure | type: float

Input name	topic	path	type
%	topic1	level	int

(**IMPORTANT**: At least one sensor must be defined).

5. Define States

In this step the states of the individual sensors are defined. To do this, click on the created sensors and give the states a name and an identifier. The identifier is used to define the state uniquely in the simulation.



Step 4/4

You can partition the possible values of sensor key/value pairs (specified in Step 3/4) into states.

- States are defined by a name and threshold value. (e.g. low_pressure, 20)
- The threshold value is the lowest upper bound to the state's value set.

To add new states, first, click on the respective input's name, and then on the plus.

Example:

state name: low_pressure | t_value: 20

%	[low ; high ; +INF]
low	20
high	100

6. Close the configuration

In the last step, navigate back to previous steps or finish the process.

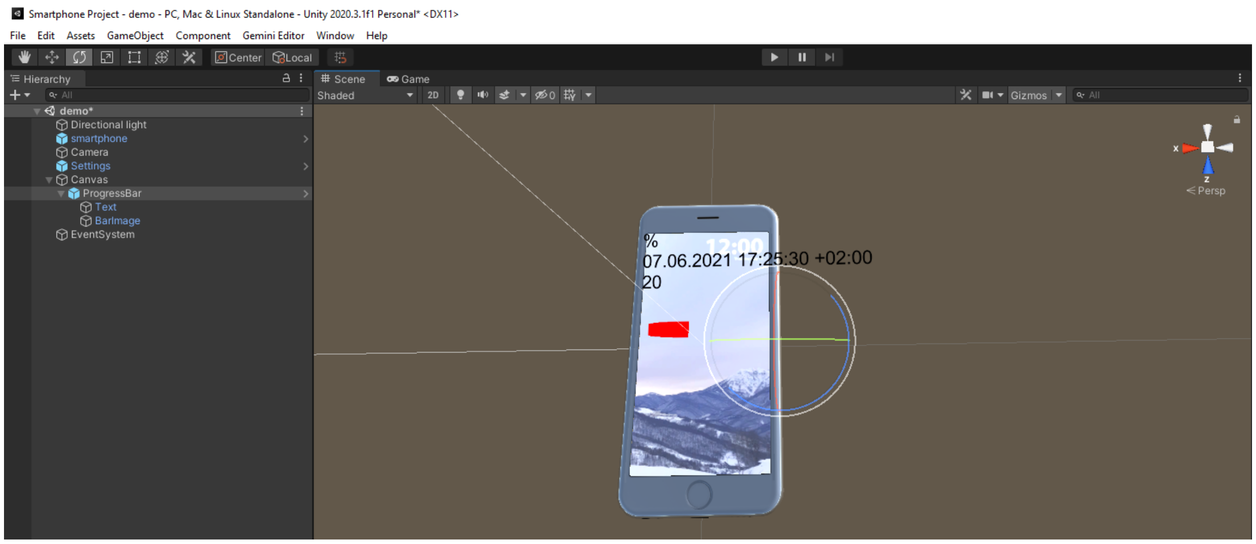
Adding a new Sensor

The following describes how to edit a new sensor to the scene. Your inputs should have already been generated and added to the hierarchy. Also, as you can see in the **Settings,** all inputs have been registered to the **Config** manager class.

testing: make sure mqtt broker is running and scene is playing; tools, configure topic

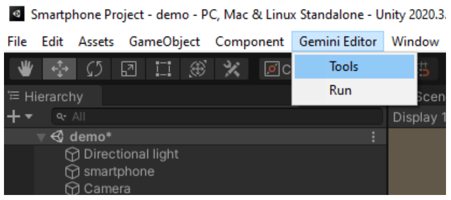
1. Create a Canvas

Start by creating a canvas in your project hierarchy. Add the desired prefabs to the canvas. Here, we selected a ***ProgessBar*** prefab that was generated during the configuration process.



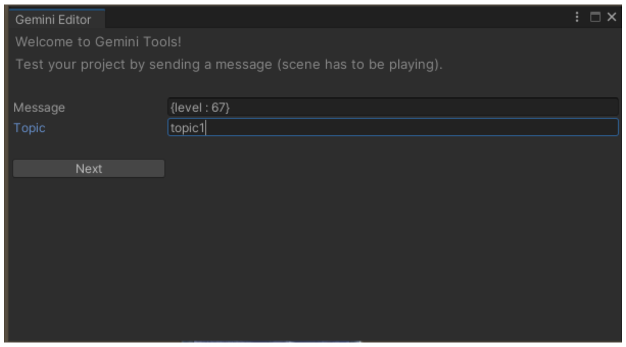
2. Test your MQTT connection

You can test whether your connection works by using our MQTT tool. Run it by clicking ***Gemini Editor - Tools.***


Smartphone Project - demo - PC, Mac & Linux Standalone - Unity 2020.3.
File Edit Assets GameObject Component Gemini Editor Window
Tools
Run
Hierarchy
demo*
Directional light
smartphone
Camera

(Important: scene has to be playing when starting tools)

Here, you can send quick test messages on a topic of your choice.

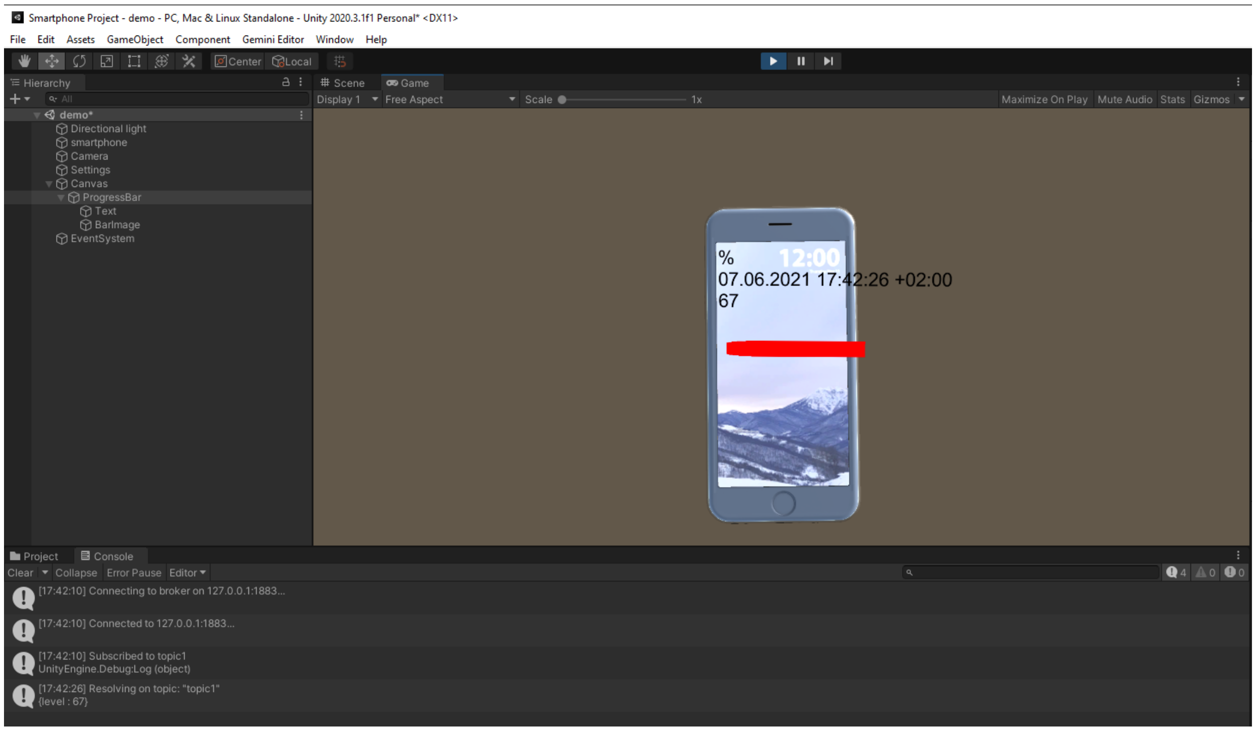


Welcome to Gemini Tools!
Test your project by sending a message (scene has to be playing).

Message: {level : 67}
Topic: topic1

(Pay attention to the correct JSON format of your messages!)

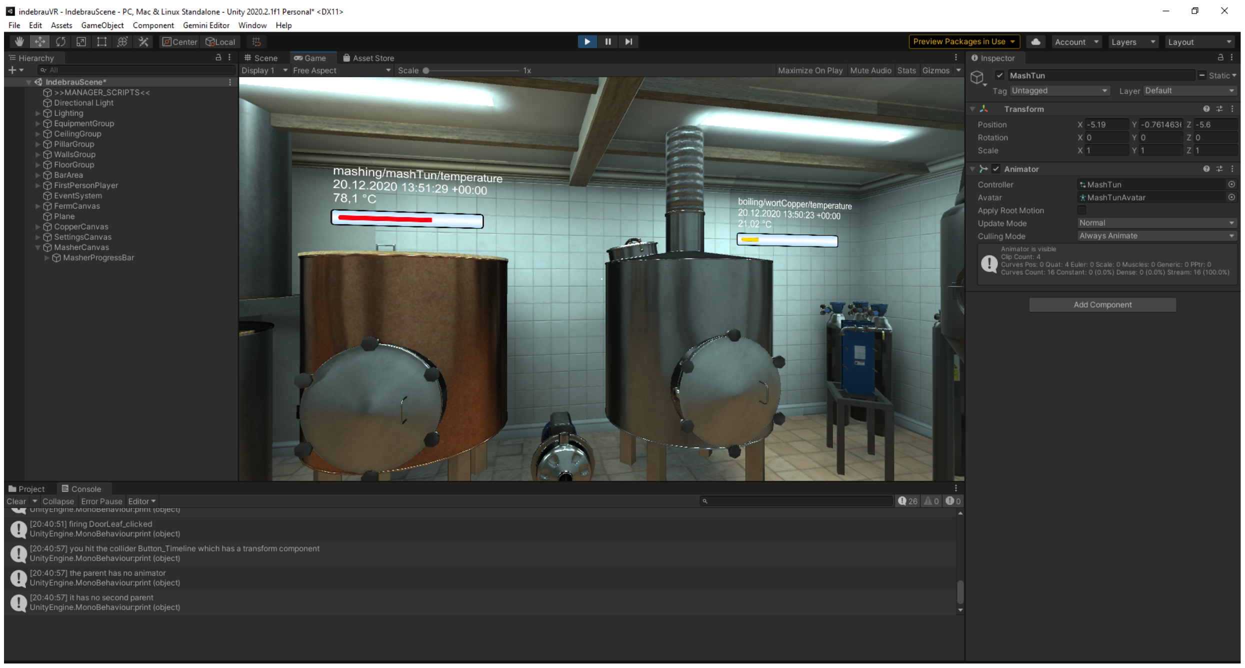
In the console, your message should be visible. In the scene, you can see your update.



```
[17:42:10] Connecting to broker on 127.0.0.1:1883...
[17:42:10] Connected to 127.0.0.1:1883...
[17:42:10] Subscribed to topic1
UnityEngine.Debug:Log (object)
[17:42:26] Resolving on topic: "topic1"
{level : 67}
```

The progress bar correctly displays our dummy battery status.

An example of our previous work can be seen below, here we were able to integrate sensors of a brewery into a digital scene and create custom replay behavior to relive the brewing process.


indebrauVR - IndebrauScene - PC, Mac & Linux Standalone - Unity 2020.2.1f1 Personal* <DX11>
File Edit Assets GameObject Component Gemini Editor Window Help
Center Local
Preview Packages in Use
Account Layers Layout
Hierarchy
Scene Game Asset Store
Display 1 Free Aspect Scale 1x
Maximize On Play Mute Audio Stats Gizmos
Inspector
Project Console
Clear Collapse Error Pause Editor
Add Component

```
IndebrauScene*
  >>MANAGER_SCRIPTS<<
  Directional Light
  Lighting
  EquipmentGroup
  CeilingGroup
  PillarGroup
  WallsGroup
  FloorGroup
  BarArea
  FirstPersonPlayer
  EventSystem
  FermCanvas
  Plane
  CopperCanvas
  SettingsCanvas
  MasherCanvas
    MasherProgressBar
```

mashing/mashTun/temperature
20.12.2020 13:51:29 +00:00
78,1 °C

boiling/wortCopper/temperature
20.12.2020 13:50:23 +00:00
21,02 °C

```
MashTun                      Static
Tag Untagged        Layer Default

Transform
Position    X -5.19   Y -0.761463(   Z -5.6
Rotation    X 0       Y 0            Z 0
Scale       X 1       Y 1            Z 1

Animator
Controller          MashTun
Avatar              MashTunAvatar
Apply Root Motion
Update Mode         Normal
Culling Mode        Always Animate

Animator is visible
Clip Count: 4
Curves Pos: 0 Quat: 4 Euler: 0 Scale: 0 Muscles: 0 Generic: 0 PPtr: 0
Curves Count: 16 Constant: 0 (0.0%) Dense: 0 (0.0%) Stream: 16 (100.0%)
```

```
UnityEngine.MonoBehaviour:print (object)
[20:40:51] firing DoorLeaf_clicked
UnityEngine.MonoBehaviour:print (object)
[20:40:57] you hit the collider Button_Timeline which has a transform component
UnityEngine.MonoBehaviour:print (object)
[20:40:57] the parent has no animator
UnityEngine.MonoBehaviour:print (object)
[20:40:57] it has no second parent
UnityEngine.MonoBehaviour:print (object)
```