

# The LEGO fireman robot at the Imagination Station

09/03/15 - 26/05/15  
Shogo Nishiguchi

## Fixing the hardware and the software

Firstly, I used the LEGO robot and a Unity Robot Engine developed by previous students. However, when I came here, the robot did not work because of some hardware troubles. I will show you what I did for a particular trouble.

As you can see, this robot is pretty small even though it has 6 relatively big actuators. What you have to take care is the head. A small rectangle board with a camera and a microphone is too close to an actuator. The board easily gets hot and it burns a wire of the actuator or melts the glue used inside head that causes the problem that the robot start shaking its head.

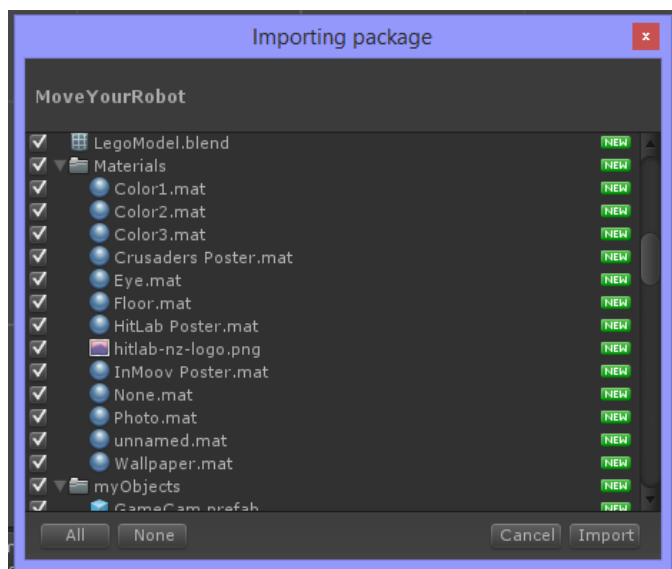
However it is impossible to make a space since the head is so small. Thus, after I changed the actuator burnt by the board, I covered the wire not to be burnt. It is not the best way but I am not sure if you can find a board that will not be hot. Additionally, I tilted the actuator in the head part a little bit so that it does not touch the board much. It makes the range of angle of the head smaller, but it is way better than it gets broken.

To find out more about how to use the software, please check the tutorial that previous interns made, however, much information in it might no longer be correct because I updated it.

In this report, I describe what I did with simple instruction of the Robot Engine.

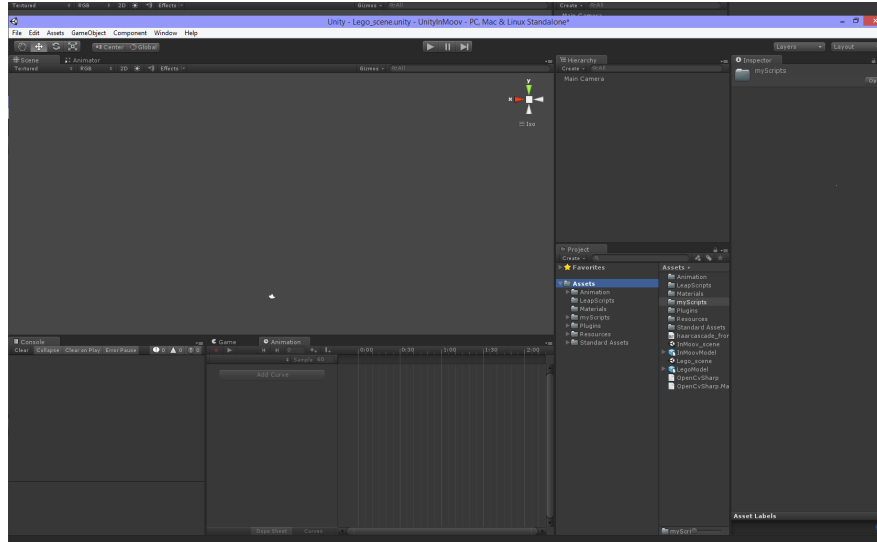
## Setup the Unity robot interface

Firstly, this software runs only on Windows PC. Install Unity 4.x.x (not 5.x.x) and then download MoveYourRobot package. Make a new Unity project and import it:  
Assets->Import Packages...



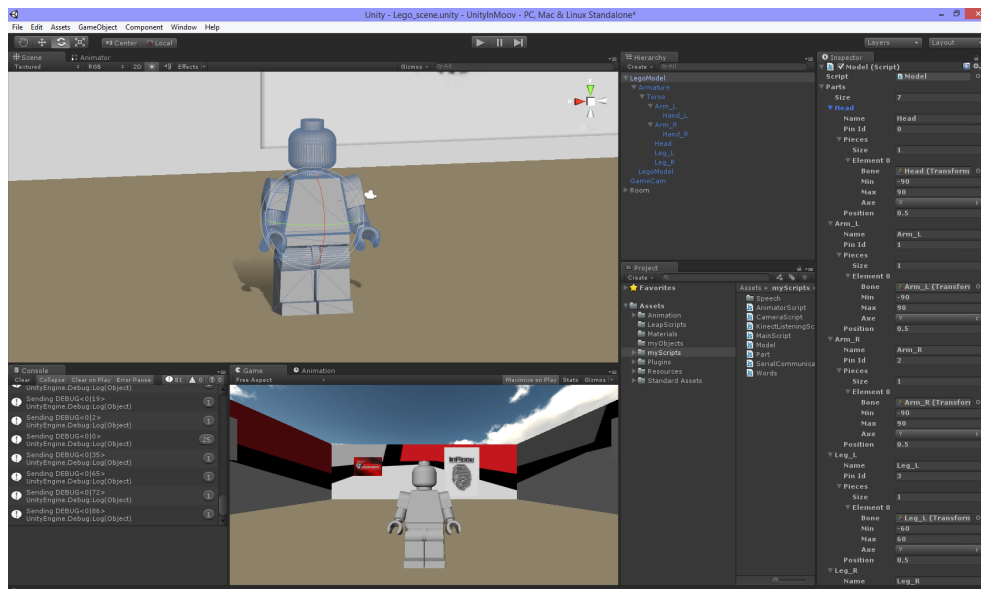
Import the model file (.blend for Blender): Assets -> Import New Assets.  
Go to File->Build Settings-> Player Settings , and then in the inspector, set API Compatibility Level to “.NET 2.0”.

Create a new scene: File->New Scene. Drag and Drop your model inside the scene.



(Remove the main cam object. Add Assets->MyObject->Room in the scene and set the position to (0,0,0). Place your model in the center of the scene. Rescale the model in the inspector to a normal scale.)

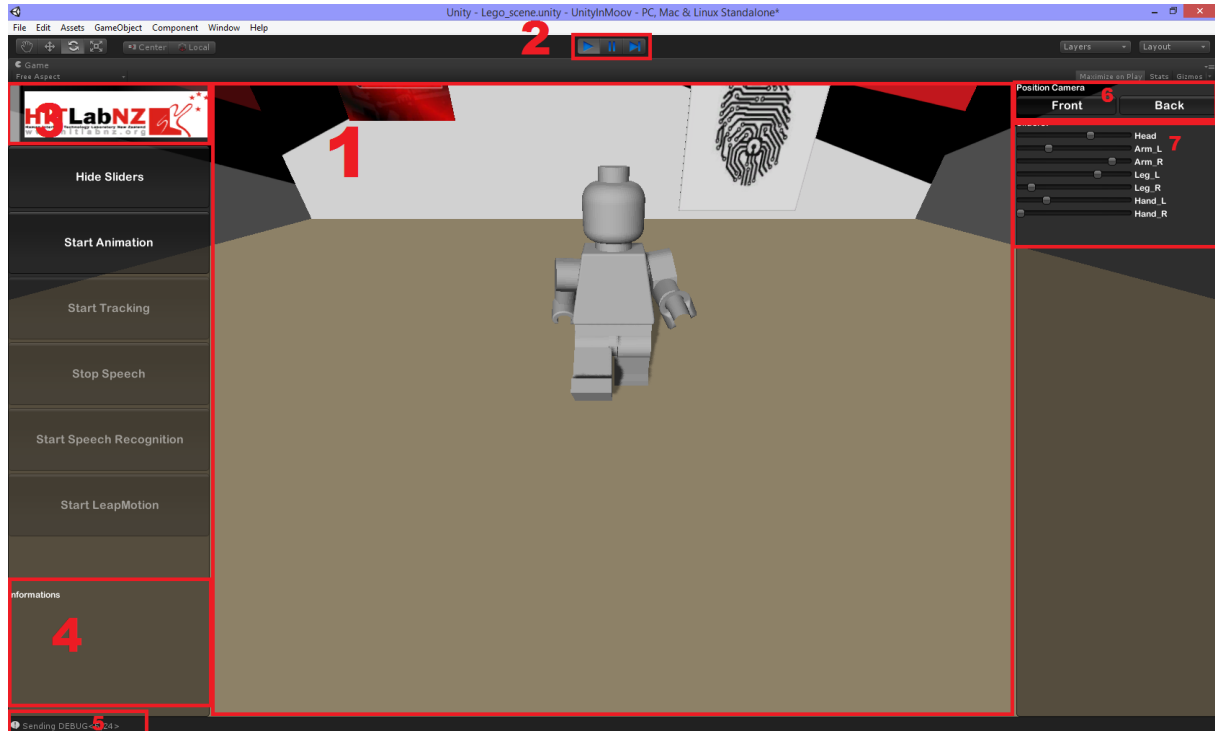
Add all the basics scripts in the MyScripts folder to your model that should have MainScript, Model, Serial Communication etc. You can use C#(.cs) and Unity Script(.js). I don't recommend you to use Boo (kind of Python) since people no longer use it and it will be unsupported on the next version of Unity. One of the advantages of the Unity is that you can use different language in the same project and they are compatible. Pay attention that source codes in *Plugin* or *Standard assets* are read before ones in *Myscripts*.



## Playing the animation

You are now ready to control your Model. Click on Play.

\*If it has an error about something related to html/php/ssh/MySQL..., you will solve it in the section of the conversational function. For now, just remove the object from the Inspector window or comment it out.



- 1: Scene: show the model and the field
- 2: Play/Stop
- 3: Click to Show/Hide Menus
- 4: Display information about contents
- 5: Show the debug log
- 6: Camera angle
- 7: Axis slider

## Setting the Arduino board

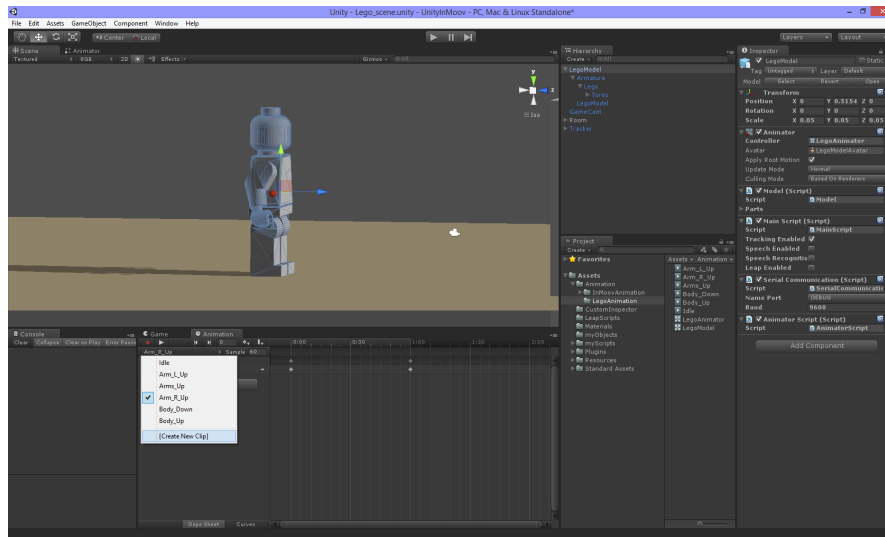
You can use an Arduino Uno, or Mega, and combine several boards if you have many servo motors. Set the NamePort of the SerialCommunication script. Use DEBUG if you don't want to connect your robot yet, you will see the data sent in the debug log.

I also attached a light sensor for my experiment. The communication between Unity and Arduino is bi-directional and for the LEGO robot, most of the Analog ports are released now. Thus, you are ready to use any analog sensors soon.

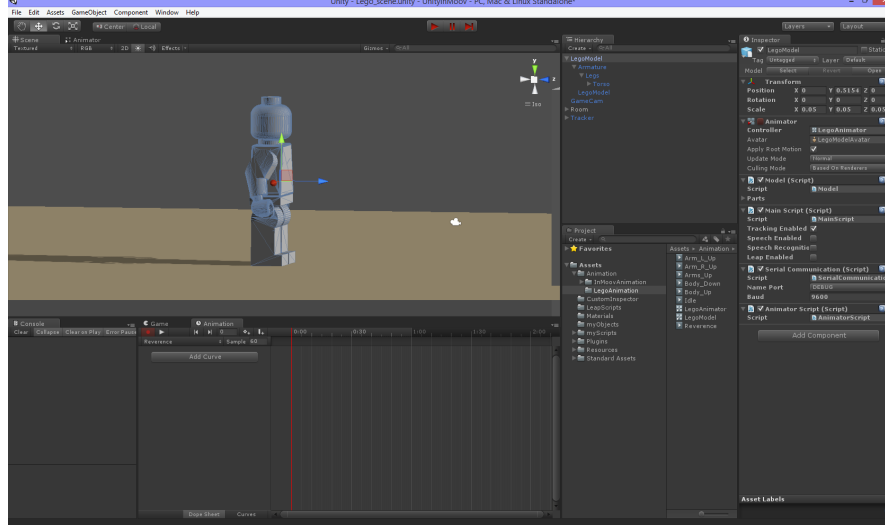
To find out more, check SerialCommunication.js, Arduino Sketch, and the connection of the light sensor.

## Animating the robot

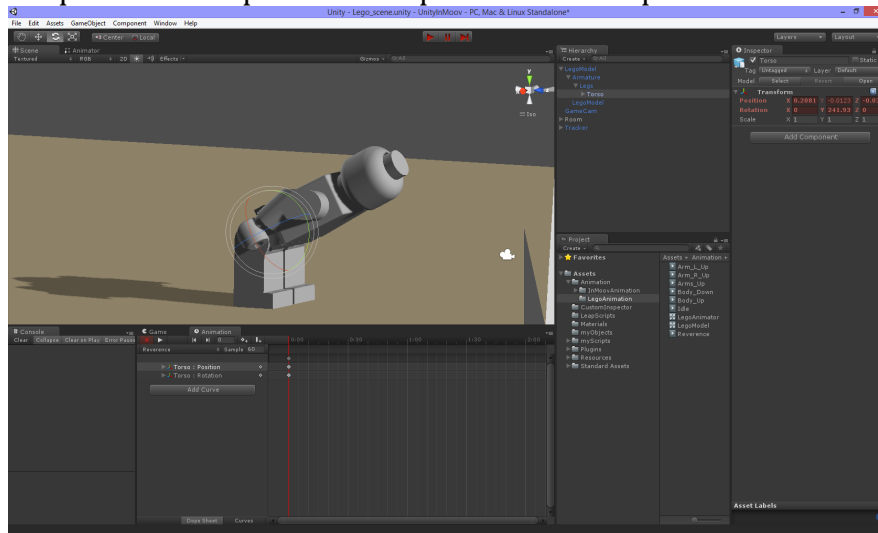
There are many ways to animate on Unity. Create a new Animator Controller and drag the new animator and drop on the Animator->Controller in the inspector window. In the Animation window, create a new clip.



Click at the 0:00 on the timeline and then a red line should appear.

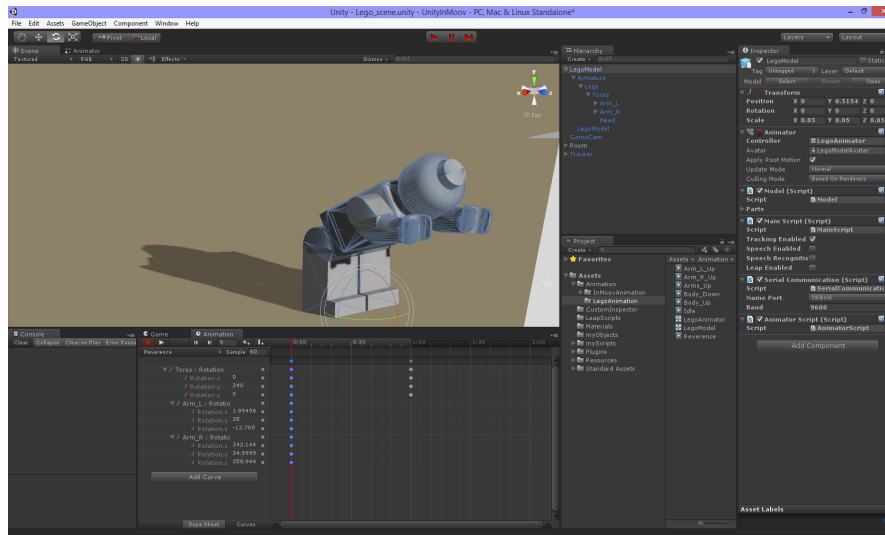


In the Hierarchy window, click on the part you want to move and in the scene window, rotate it using your mouse. In the Animation window, the part's name should appear with position and rotation components. The position component is not important.



Using a mouse to rotate its part is not very precise, for example when you want to raise its both arms, you have to rotate them to the same angle even though it is not easy. In such a case, type the number of the angles in the inspector window.

Be careful about the min and max values. In this stage, no program has not executed yet, so you can rotate any parts to any angles even if it is out of the range.



Now you are ready to use the animator controller to combine your animations and then program the behavior of your robot. Drag animations from Assets->Animation->MyRobotAnimation to the Animator window.

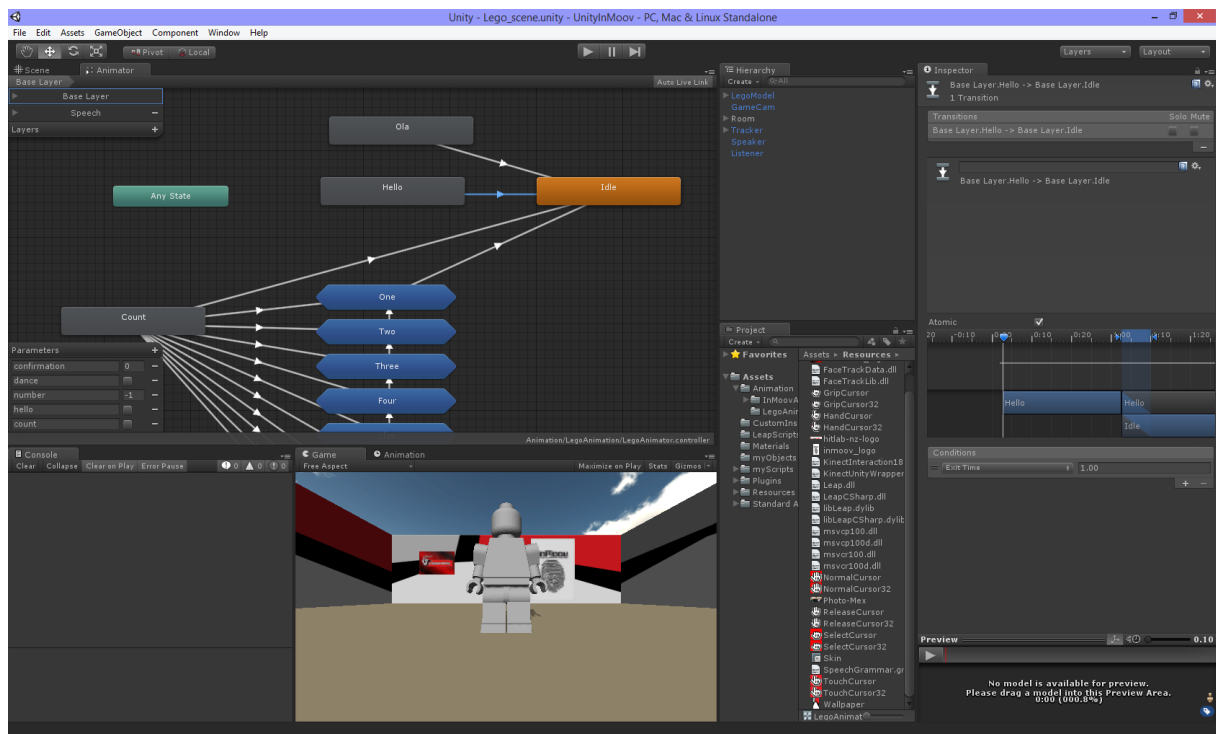
Right click on the animation that you want to play first and select "Set As Default". Then create transitions between states. Right click on a state, "Set Transition" and link it to

another state. Then you have to set the parameters of this transition, this is how you create the motion.

For the each animation, you can set a tag in the inspector window. I programed some ActionEvents. "Speech" is for the robot's speech, which enables the robot to talk at particular animation state, "Listenning" where the robot waits that people say something. "ChatBot" is for the robot's speech based on the Chatbot such as Cleverbot and Pandrabot.

Flags are used for the transitions.

- donation: it gets active when the light sensor reacts to something.
- facefind: it gets active when the camera find face.
- saidsomething: it gets active when microphone pick up a voice no matter what they say.
- speechend: it gets active when the speech of the robot finished.
- unrecognized: it gets active when people say what the robot cannot understand.
- yes,no,howareyou etc...: it gets active when the robot understand what they say based on the preprogrammed sentences.



There are some exceptions depending on what you want to do, but basically what you have to do is to change the flags in Animator.js., set the speech in Speech.js.

## Face Tracking

Connect the camera with your computer. The LEGO robot has a camera on its cap. Add the object *Tracker* to Assets->myObject.

At the Tracking Script in the Inspector:

*XTransform* = drag the part of your robot which will rotate horizontally

*YTransform* = drag the part of your robot which will rotate vertically

If you just have one DoF for the Tracking, you can drag the *Tracker* object in the empty parameter. If you just have one DoF for the Tracking, you can drag the Tracker object in the empty parameter. Now in the MainScript, check the TrackingEnabled checkbox.

## Background music

You can also play some music on the Unity.

I wrote the program to play “Everything is awesome” while it is dancing.

## Speech of the robot

Put Voice\_speaker.dll in

Windows 32B >> Copy 'Voice\_speaker.dll' in windows\system32 folder

Windows 64B >> Copy 'Voice\_speaker.dll' in windows\SysWOW64 folder

Add Speaker folder to Assets->myObjects.

Put “Speech” tag on the animation that you want to make the robot talk. Go to Speech.js and set the sentence to the array. This is the way that the robot talks based on the script that's prepared in advance (Static). On the other hand, put “ChatBot” tag instead of “Speech” one so that robot talks along a Chatbot (Dynamic).

I wanted to send/receive data with TCP/IP communication, but it did not work well. I guess the free version of Unity has some limitations? However I could not make it clear, so I used MySQL to register a log of the conversation. In the future work, you might want to use this robot as a shopkeeper at the LEGO store as all LEGO products has particular number. I have already made the program to access it from Unity to the database. Check the MySQL.cs.

The process of the conversation is as below.

1. Play the Unity Game.
2. Press *Start Animation* button
3. Open Google Chrome and run index2.html (Nothing will be displayed)
4. Press *Start Tracking* on Unity.
5. When the robot detects someone's face, something red will be displayed on the browser.
6. PHP file sets chatbotflag=0 in *flag* database, mySQL. (Speech recognition disabled)
7. The robot is going to say something.
8. PHP file sets chatbotflag=1 in *flag* database, mySQL. (Speech recognition enabled)
9. Something green is displayed on the browser.
10. Talk to the microphone, and then your speech will be shown on the browser.
11. PHP file registers the data to MySQL.
12. Unity accesses to MySQL and then you can process it if you want in mySQL.cs.
13. Something red is displayed on the browser.

14. PHP file sets chatbotflag=0 in flag database, mySQL. (Speech recognition disable)
15. It will be played in Speech.js.
16. Repeat.

I prepared the chatbotflag so that the robot does not detect its own voice and start talking to itself. "saidsomething" flag on the animator window is also prepared for this problem.

I used 3 APIs, speech recognition, Text-To-Speech and Chatbot, and so far it needs to access MySQL, so these factors take some time until the robot says something. The delay would be shortened by reviewing the program but anyway it is inevitable. I conducted the experiment at the LEGO Imagination Station. I displayed "Wait" with red background and "Talk" with green background on the browser that were the cue to customers. You have got to prepare such a cue somehow. People tend to respond, so even 100ms of delay would be fatal because the robot detects from the middle of the sentence. Thus, you can use a way that you like, but such a cue is necessary. As you know, even Siri developed by great engineers of Apple has some delay and cue.

The advantage of this function is that you can write your own conversation script for your research project and when people ask something that you did not expect, the robot answers using the chatbot. If you use a chatbot, the robot answers any kinds of questions, but there is no end and you cannot set the goal of the conversation.

On the other hand, if you use scripting, you have to write down all the possibilities of people's questions and answers although it is almost impossible. Thus, one of the biggest advantages of this program is that you can lead the people to the goal of the conversation and the robot can answer any random questions. It might be inappropriate, but it is a lot better than keeping being quiet to questions.

In this experiment, I wrote the main conversation script for the donation. "How are you?", "Did you enjoy the program today?", "Did you know I am made of LEGO?", "Do you want to see my dance?", "Please donate." and something like that. Check the Animator Controller in the Donation folder.

I named this robot Izzar because the robot answers like "My name is Izzar..." but if you want to change it, just replace the text and then use the text to speech. Of course, the chatbot is not perfectly clever, for example, when you ask it like how tall are you?, it is going to say "1.5 metre..." even though it is about 30 cm actually. You might want to set personal data such as its height, weight, age, gender, job and so on because people often ask it.

This is a way to use the Chatbot. We used EasyPHP.

1. Go to apache config directory  
`cd "C:\path\to\EasyPHP\binaries\apache\conf"`
2. Complete the PATH var with the path to openssl.exe  
`set PATH=%PATH%;C:\path\to\EasyPHP\binaries\apache\bin`

3. Answer the questions asked by the next command. It will create the public and private keys you need. For "Common name" entry, input "localhost".

```
openssl req -config openssl.cnf -new -out site.csr
```

4. You have to write the same pass word you choose before. It will generate SSL keys.

```
openssl rsa -in privkey.pem -out site.key
```

5. Here we sign the certificat. Beware that the browser will alert the user that the website cannot be trusted because of self-signed certificat (but a real certificat is expensive).

```
openssl x509 -in site.csr -out site.cert -req -signkey site.key -days 365
```

6. Convert to DER format, understood by Apache.

```
openssl x509 -in site.cert -out site.dert.crt -outform DER
```

7. Create a new folder 'ssl' in C:\path\to\EasyPHP\binaries\apache\conf\  
Then put in it the files created by the previous commands.

8. Find the httpd.conf file in C:\path\to\EasyPHP\binaries\apache\conf\.

-uncomment the line 177 :

```
LoadModule ssl_module modules/mod_ssl.so
```

-uncomment the line 823 :

```
Listen 443
```

-uncomment (or add) the code below at the line 870 :

```
<VirtualHost *:443>
```

```
    ServerName localhost
```

```
    SSLEngine on
```

```
    SSLCertificateFile
```

```
    "C:\path\to\EasyPHP\binaries\apache\conf\ssl\site.cert"
```

```
    SSLCertificateKeyFile
```

```
    "C:\path\to\EasyPHP\binaries\apache\conf\ssl\site.key"
```

```
</VirtualHost>
```

### Experiment result

I am trying to know the difference of the amount of the donation with the LEGO robot and without it at the Imagination Station. I need the data of how many people came and how many people interacted etc. (Still analyzing)

If you have any questions, feel free to ask me.

Nishiguchi Shogo, Osaka University (nishiguchi.shogo@irl.sys.es.osaka-u.ac.jp)