Management of Broadlink RM Mini 3 from eedomus

Since its May 29th, 2017 update, ConnectedObject offers support for the Broadlink RM Pro but, what about those of us who did not want to scratch our pocket and buy the Mini?

Well, what happens is that it does not work.

The truth is that they put us a little green with envy, as they show a php script (http://<IP_EEDOMUS>/script/) called broadlink.php, and they warn you that you will not be able to modify it.

```
// eedomus script for Broadlink RM Pro infrared hub
// this script will not validate if your upload it manually
```

In this script, they make use of functions that are not allowed in the subset of development functions that they offer:

```
$cmd = "python /mnt/flash/puch/modules/broadlink_python/locatrm.py";
exec ($cmd, $output);
```

Even so, it is not bad motivation to try it, so let's go.

First, we have to look for information in the Internet, and of course, there has already been someone who has done the hard work for us (https://github.com/davorf/BlackBeanControl).

Davorf, based on the mjg59 work (https://github.com/mjg59/python-broadlink) it's able to connect with the RM Mini from the command line using python.

Now, we have to see how to bring all this stuff to our eedomus.

Obviously, with the small set of functions we have, we must find an imaginative solution. The first step we have to do is to install the BlackBeanControl from davorf following his instructions.

Then, once installed, you have to configure it following the steps at https://github.com/davorf/BlackBeanControl#configuration where we will register our Mini RM. The IP and MAC Addresses can be obtained looking at the DHCP of our router.

And now is where the interesting stuff begins: doing a bit of debugging and looking at mjg59 code, you can see that each command that is sent to the RM Mini splits into two: a kind of authentication (AUTH), just the same for all commands and then the command itself.

The command that is sent, is a binary that is encrypted using AES and with a final checksum, so it does not seem very easy to implement it in eedomus.

I have faced this making a base 64 encode of both, the authorization and the command, and then print it. This will be the only things we will need for our script at eedomus.

We will slightly modify mjg59 script to print the base 64 encoded command before sending it to RM Mini. For this purpose, in line 14 of file __init__.py (which is installed in step: Download python-broadlink package - you can find it on the github by the package name (github user: mjg59)) we will add:

```
import binascii
```

in such a way that it remains:

```
import sys
import threading
import binascii
def gendevice (devtype, host, mac):
```

And between lines 276 and 277 we add the following:

```
print ('Encoded:', binascii.b2a_base64 (packet))
```

So that it is:

```
try:
print ('Encoded:', binascii.b2a_base64 (packet))
self.cs.sendto (packet, self.host)
```

So now we can use the BlackBeanControl.py to record the commands we need. BlackBeanControl will record the learned commands in the BlackBeanControl.ini file in the same directory that t iresides, but they will be useless for us, because they are not encrypted neither signed. With the modifications that we have made previously, when learning the command, will produce an output like:

('Ecoded:',
'WqWqVVqlqlUAAAAAAAAAAAAAAAAAAAAAAAAAAAAAD4AAAAKidqALH8tEMN3IwjAgAAAMvH
AAAg5ISMRV+Whk88Tf/XEIbI2WFChsEOsXNhuYJZm57WdtbplweAPWZiJbH1LvyobpCDpLv1jwPRpKkNo0jPg
biXgCrRnGwLHXkO50EGzce4tpQaDAFCYsUg/BoZP5EA33o7zCETwcxS/+gWyc1ENDlg\n')

('Ecoded:',
'WqWqVVqlqlUAAAAAAAAAAAAAAAAAAAAAAAAAAAAABo9gAAKidlAEVbtEMN3IwjAAAAAKHDA
ABFNFLn+S7alYNEkwg175pt+2www3ChsEOsXNY/u1Ot+giBTKf4z0FxADKOVww7hslNBXCESaOJ4prhBFQ2oF
vd3ALBYa8TJeh+GbD30c4GjeUbYZFWh20zjP87mR5AzbE=\n')

Being the first one the AUTH and the second the command, so the command to use would be:

WqWqVVqlqlUAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABo9gAAKidlAEVbtEMN3IwjAAAAAKHDA
ABFNFLn+S7alYNEkwg175pt+2www3ChsEOsXNY/u1Ot+giBTKf4z0FxADKOVww7hslNBXCESaOJ4prhBFQ2oF
vd3ALBYa8TJeh+GbD30c4GjeUbYZFWh20zjP87mR5AzbE=

We can dismiss the \n.

At this point we find that the base decode64 function is not available at the eedomus, so it will be then the first thing we have to implement (sdk_decode_base64 at broadlinkrmmini.php).

Once this is resolved, we have to see how to send the command over our network.
Taking a look at the code of mjg59 we see this:

```
cs = socket.socket (socket.AF_INET, socket.SOCK_DGRAM)
cs.setsockopt (socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
cs.setsockopt (socket.SOL_SOCKET, socket.SO_BROADCAST, 1)
```

So we have to open a socket to send some UDP packets. Fortunately eedomus will allow us to open a UDP socket:

```
$socket = socket_create (AF_INET, SOCK_DGRAM, SOL_UDP);
socket_set_option ($socket, SO_BROADCAST);
```

Once we have all the pieces we can finish configuring everything in the script that we will upload to the eedomus (broadlinkrmmini.php):

We have to add to the script the IP and the port that has the RM mini, these data were already obtained when configuring the BlackBeanControl, so we will set it in the variables that are in the 'vars to initialize section':

```
$broadlink_ip = <IP_BROADLINK >;
$broadlink_port = 80;
```

and the same for each command that we want to send, we will set up them in the zone marked as:

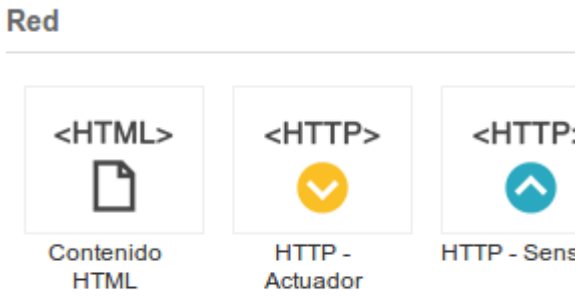
```
################### VARS TO INITIALIZE ###################################
$AUTH = '<obtained data>';
$AC_ON = '<obtained data>';
$AC_OFF = '<obtained data >';
################### END VARS TO INITIALIZE ################################
```

Once the commands have been set, we upload the script to eedomus. At this point we no longer need all that we have had to install from BlackBeanControl, since our goal was to get the encrypted and signed commands, so we can now test it using:

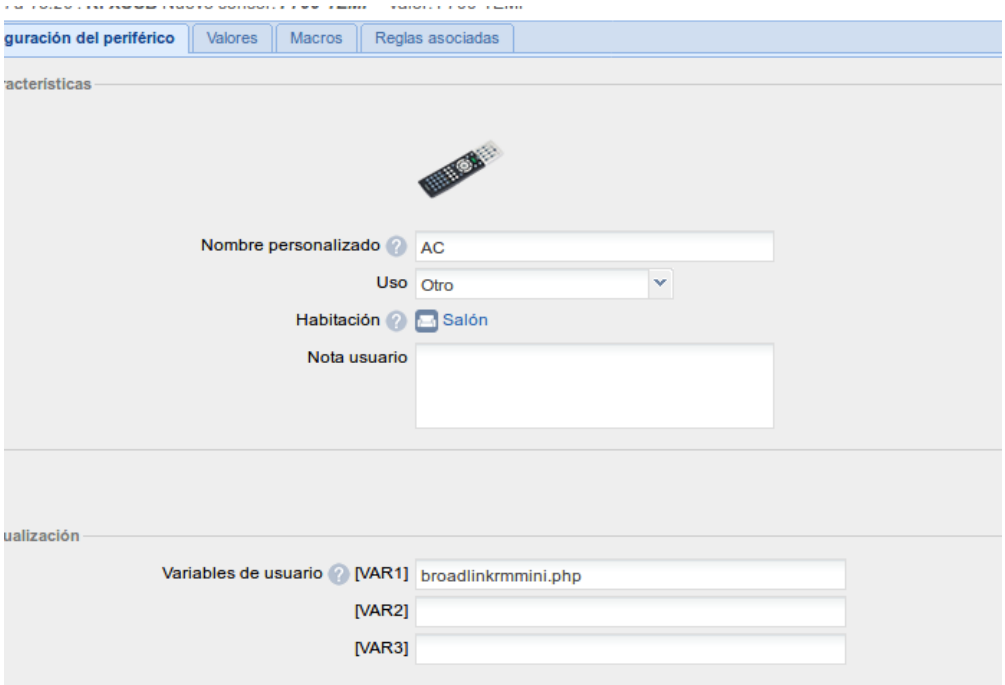
```
http://<IP_EEDOMUS>/script/?Exec=broadlinkrmmini.php&comando=xxx
```

replacing xxx with the command we want to test (AC_ON, AC_OFF, ...).

When it works, we can create an actuator in the eedomus interface:
We create an HTTP network actuator.



To make the things easier, we add in [VAR1] the name that we have given to the script (broadlinkrmmini.php in this case)



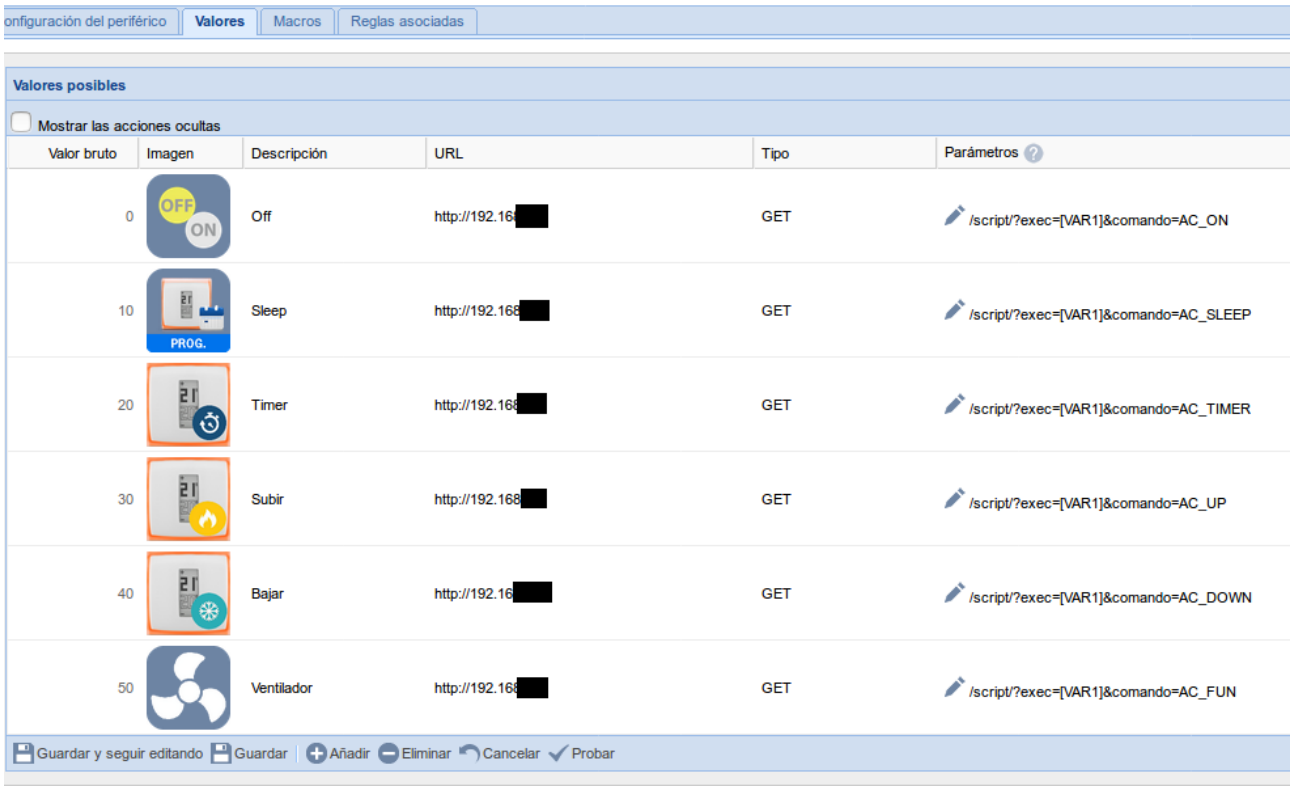
In the values tab we need to add the previously saved options:

The value field is a sequential that will only indicate the last function that has been done with the actuator, in URL we will have to put the internal URL of the eedomus, the type will be GET and the Script should be

/script/?exec=[VAR1]&comando=AC_ON

where the variable *comando* (spanish) should have been registered in broadlinkrmmini.php as we said above:

$AC_ON = '<obtained data>';
$AC_OFF = '<obtained data>';



Note: We must be clear that we will not get feedback in the eedomus if the command was executed correctly, we will have to see it in the device we are trying to control.