Home automation

Welcome to the awesome world of home automation! Since Batocera is Wake on LAN (WoL) enabled by default, you can now integrate your Batocera system into most any advanced home automation system.

Integrate Batocera with MQTT

MQTT is a simple messaging protocol used in home automation to allow devices to talk to each other. It helps smart home gadgets, like lights and sensors, send and receive messages quickly and efficiently. This means you can control and monitor your home devices in real-time, as well as trigger automations that respond to changes in those devices.

Batocera provides an MQTT client, which can publish all of Batocera's game start/stop and EmulationStation events to an MQTT broker (server) that you provide.

Requirements

- A separate MQTT broker (server) to publish events to. Batocera only contains tools to read & write to an MQTT broker, but not the broker itself.

Use a Custom User Service to Forward Events

The following user service listens for the game start/stop and EmulationStation events that are generated by Batocera, and relays them to the MQTT broker.

When the service is enabled, it will install several small script files that take advantage Batocera's custom script features to capture the events.

1. Copy this script to `/userdata/system/services/mqtt_events`
2. Modify the configuration variables at the top of the file with the connection info for your MQTT broker
3. In EmulationStation, enable the `mqtt_events` service
4. Reboot to complete install, or via SSH, run `/userdata/system/services/mqtt_events start`

The service is designed to capture events and publish them to the following topics:

```
/batocera/<HOSTNAME>/emulationstation/<EVENT_NAME>
/batocera/<HOSTNAME>/game/<EVENT_NAME>
/batocera/<HOSTNAME>/system/<EVENT_NAME>
```

The messages published to those topics are a JSON dictionary containing 2 fields:

- `data`: An array of the arguments passed to the custom even script. The number of arguments and their values will vary depending on the event. Refer back to the docs for more info.

- `timestamp`: An ISO-formatted timestamp of when the event was published

mqtt-events

```
MQTT_HOST=your-mqtt-server.lan
MQTT_PORT=1883
MQTT_USER=username
MQTT_PASSWORD=password
MQTT_BASE_TOPIC=batocera/$(hostname)

SERVICE_NAME=$(basename "$0" .${0##*.})
ES_EVENTS=(game-start game-end game-selected system-selected theme-
changed settings-changed controls-changed config-changed quit reboot
shutdown sleep wake achievements screensaver-start screensaver-stop)


################################################################################
########
# Publish an event message to the MQTT broker as JSON dictionary.
#
# The first argument is the subtopic to publish to (source/event_name)
# The remaining arguments are converted to a list of strings, which
# is stored in the 'data' field of the message
# An ISO-formatted 'timestamp' field is automatically added to the
message
#
# Only executes if this service is enabled
################################################################################
########
publish() {
    if [[ " $(/usr/bin/batocera-settings-get system.services) " == *"
$SERVICE_NAME "* ]]; then
        subtopic="$1"
        shift
        # Initialize an empty JSON array
        data='['
        # Iterate over the remaining arguments
        for arg in "$@"; do
            # Convert each argument to a JSON string and append it to
the array
            data+="$(jq -Rn --arg arg "$arg" '$arg'),"
        done
        # Remove the trailing comma and close the JSON array
        data="${data%,}]"
        mosquitto_pub -h "$MQTT_HOST" -p "$MQTT_PORT" -u "$MQTT_USER" -
P "$MQTT_PASSWORD" -t "$MQTT_BASE_TOPIC/$subtopic" -m '{"data":
'"$data"', "timestamp": "'"$(date --iso-8601=ns)"'"}' -r
    fi
}

################################################################################
```

```
########
# Start the MQTT service.
# - Installs any missing support files
# - Publishes a system startup event
################################################################################
########
start() {
    install
    publish "system/startup"
}

################################################################################
########
# Stop the MQTT service.
# - Publishes a system shutdown event
################################################################################
########
stop() {
    publish "system/shutdown"
}

################################################################################
########
# Create scripts that capture events and publish them to the MQTT
broker.
################################################################################
########
install() {
    # EmulationStation events
    for event in ${ES_EVENTS[@]}; do
script_dir="/userdata/system/configs/emulationstation/scripts/$event"
        mkdir -p "$script_dir"
        script="$script_dir/$SERVICE_NAME.sh"
        if [[ -f "$script" ]]; then
            echo "WARNING: Skipping installation of '$script', as it
already exists"
            continue
        fi
        echo "Installing '$script'"
        cat <<-EOF > "$script"
#!/bin/bash
/userdata/system/services/$SERVICE_NAME publish
"emulationstation/$event" "\$@"
EOF
    chmod +x "$script"
    done

    # Game events
    script="/userdata/system/scripts/$SERVICE_NAME.sh"
    echo "Installing '$script'"
    cat <<-EOF > "$script"
```

```
#!/bin/bash
/userdata/system/services/$SERVICE_NAME publish "game/\$1" "\${@:2}"
EOF
}

################################################################################
########
# Delete scripts that capture events and publish them to the MQTT
broker.
################################################################################
########
uninstall() {
    # Delete EmulationStation event scripts, and remove event
directories if empty
    for event in ${ES_EVENTS[@]}; do
script_dir="/userdata/system/configs/emulationstation/scripts/$event"
        script="$script_dir/$SERVICE_NAME.sh"
        echo "Deleting '$script'"
        rm "$script"
        if [ -z "$(ls -A "$script_dir")" ]; then
            echo "Deleting empty directory '$script_dir'"
            rm -rf "$script_dir"
        fi
    done

    # Delete Game event script
    script="/userdata/system/scripts/$SERVICE_NAME.sh"
    echo "Deleting '$script'"
    rm "$script"
}

################################################################################
########
# Script entrypoint
################################################################################
########
if [ $# -eq 0 ]; then
    echo "ERROR: No arguments provided"
    echo "Usage: $SERVICE_NAME publish|start|stop|install|uninstall
[args]"
    exit 1
fi
if [[ $(type -t "$1") == function ]]; then
    FUNCTION="$1"
    shift
    $FUNCTION "$@"
else
    echo "ERROR: '$1' is not defined"
    echo "Usage: $SERVICE_NAME publish|start|stop|install|uninstall
[args]"
    exit 1
```

```
fi
```

Control Batocera with a Logitech Harmony remote via Home Assistant (HA)

This tutorial will cover on how you can integrate your Batocera system into Home Assistant (HA) so that you can startup/shutdown Batocera with just one click on your Logitech Harmony remote control. Note that for consistency reasons we will cover two shutdown scenarios here:

1. Scenario 1: You are logged in into Batocera's Kodi integration, meaning you have Kodi up and running on your system's screen while you want to shutdown your system. To keep Kodi's database consistent it would be best to use Kodi's API to shutdown the system so data consistency can be guaranteed.
2. Scenario 2: Your are logged out of Batocera's Kodi integration, meaning you have Kodi not running on your system's screen, instead you have Batocera's Emulation Station up and running on your system's screen while you want to shutdown your system.

Depending on how you are using Batocera (with or without its Kodi implementation), it would obviously make sense to assume that all Batocera users are using both systems, Kodi *and* Batocera. Therefore we will create a solution which always covers both situations so you don't have to worry about it in any way. In short: In the end we will have a solution which sends a shutdown command by using the Kodi API first (which will be ignored when Kodi is not running), waits 10 seconds, and then sends a shutdown command by sending a simple SSH command to Batocera's underlying Linux system (which will be ignored when the system is not running anymore in case it has already being shut down by the Kodi API shutdown command).

Let's do it!

Requirements

This tutorial will cover an example on how to turn on/off your Batocera system with just one click on your Logitech Harmony remote control via Home Assistant (HA). For this you will need basically three things:

- A Wake on LAN (WoL) capable/enabled Batocera system having a static IP address or a hostname provided by your DHCP/DNS server (mostly your router)
- A running Logitech Harmony (hub based) remote control setup, with the Logitech Harmony Hub having a static IP address or a hostname provided by your DHCP/DNS server (mostly your router)
- A running HA instance (For automatic device discovery and latency reasons it is strongly recommended to have the HA instance within the same Layer 2 subnet as your Logitech Harmony Hub!)

This tutorial was made based on Home Assistant version **2022.9**

Note that for all of the following steps, if you are using hostnames instead of static IP addresses, you have to make sure to have all of your DNS records up to date within your DNS server (mostly your router)!

Note that you will have to open up all according ports in your firewall between the according subnets/devices if you are not going to have all of your devices within the same Layer 2 subnet as suggested in this tutorial!

Installation

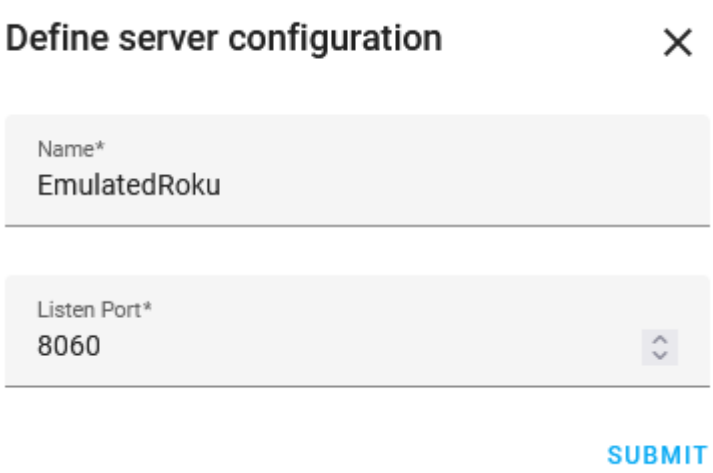
Note that for simplifying reasons we are not going to cover here on how to implement scripts/automations via separated *.yaml* files in HA, but for advanced users, of course you can feel free to do so. Also we are going to use manual *.yaml* code as little as possible here, instead we will configure most everything with the GUI.

Preparing Home Assistant

First of all you want to install three required HA **integrations** and one required HA **add-on**. All three installations are set up very quickly, so for the parts not mentioned within the following steps, just follow the instruction steps from the HA GUI, it's all straight forward.

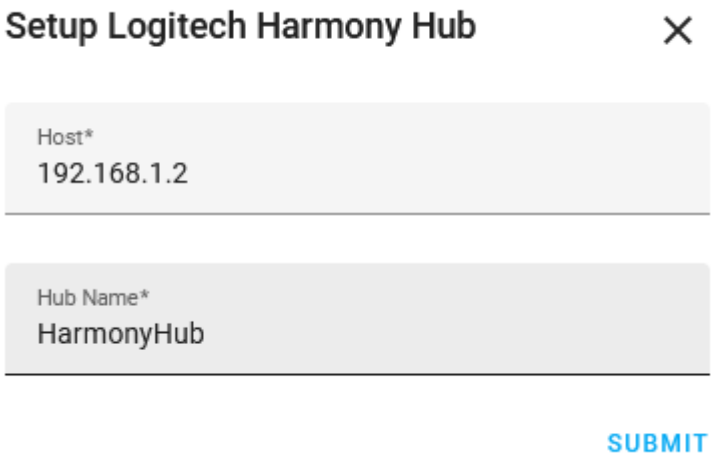
First, add the following three required **integrations** into HA (you can install them via the HA GUI if you are using at least a *supervised* HA installation (which is strongly recommended anyway or *-even way better(!)-* an official HAOS (Home Assistant Operating System) instance):

- Emulated Roku → When installing the integration it asks you to add an Emulated Roku instance automatically. Do so by giving it the name `EmulatedRoku` and by using the default port setting `8060` and then click on `SUBMIT`:



EmulatedRoku integration

- Logitech Harmony Hub → When installing the integration it asks you to add the informations about your Logitech Harmony Hub. Do so by inserting the according static IP address/hostname of the Logitech Harmony Hub (e.g. `192.168.1.2`) and give it the name `HarmonyHub` and then click on `SUBMIT`:

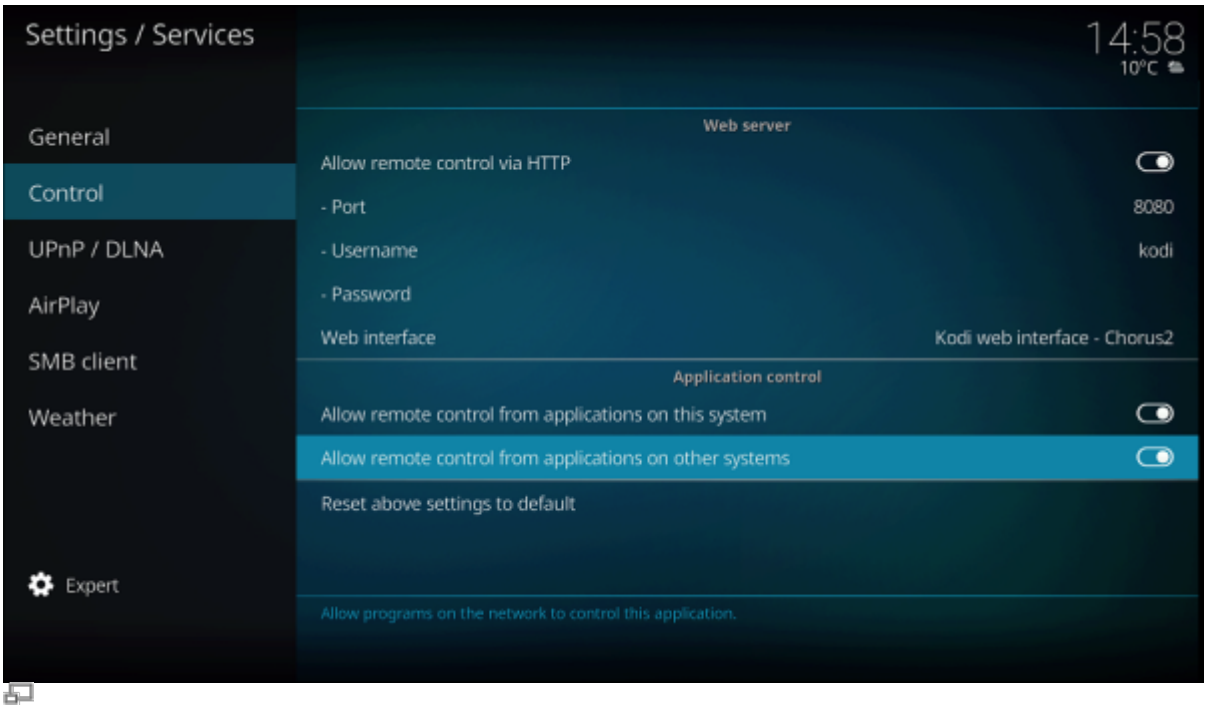


Logitech Harmony integration

- Kodi → When installing the integration it asks you to add the informations about your Kodi instance. Do so by inserting the according static IP address/hostname of your Kodi instance (= your Batocera system) and by using the default port setting `8080` and then click on `SUBMIT`:

Kodi integration

Remember to also enable the Kodi web interface and to allow remote control within Kodi settings on your Batocera system:



Kodi remote control settings

In addition you want to install the following HA **add-on** by using the HA's `Add-ons` menu:

- Terminal & SSH → Don't forget to enable the add-on after installing it! After installation you will have a new tab called `Terminal` in the HA sidebar. Click on it and you will be prompted to the command line. Now, for automating purposes you have to make your Batocera system accessible to HA via SSH without a specific user login, instead it will use a secure authentication key for passwordless login. To accomplish this, from the HA command line, execute the

following two commands (if you have already created an SSH key for your `root` user in the past on your HA instance, skip the first command!)...

```
ssh-keygen
```

Confirm *everything* without inserting *anything* and by just hitting the `[Enter]` key on your keyboard!

Now execute the following command:

```
cat /root/.ssh/id_rsa.pub | ssh root@<my_IPaddress/hostname> 'cat >>
/userdata/system/.ssh/authorized_keys'
```

...where `<my_IPaddress/hostname>` has to be replaced with the according static IP address/hostname of your Batocera system. Insert the Batocera `root` user's password (Default password: `linux`).

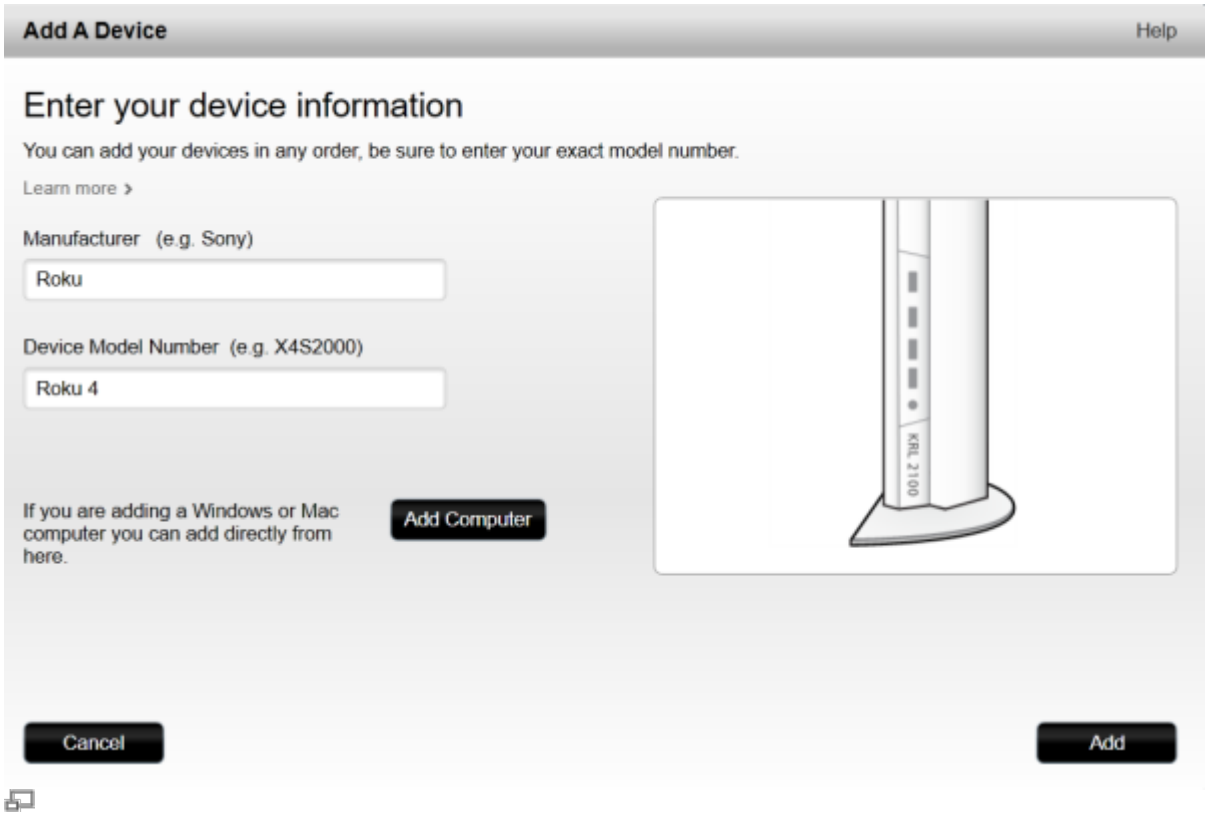
Now test the passwordless SSH login from your HA instance to your Batocera system. If it works, go further.

Note: You only have to do all of this just once unless you change the SSH authentication key manually or install Batocera from scratch again.

Preparing Logitech Harmony

With the official Logitech Harmony application for Android/iOS add a new device by letting the app search automatically for new devices. It will find the Roku 4 device easily (which we have installed on HA earlier) if you have your Logitech Harmony Hub within the same Layer 2 subnet as your HA instance. If for some reason it does not detect the Roku 4 device automatically, add it manually by using the official “Logitech Harmony Desktop” application for Windows/macOS with the following device values:

Manufacturer	Device Model Number
Roku	Roku 4

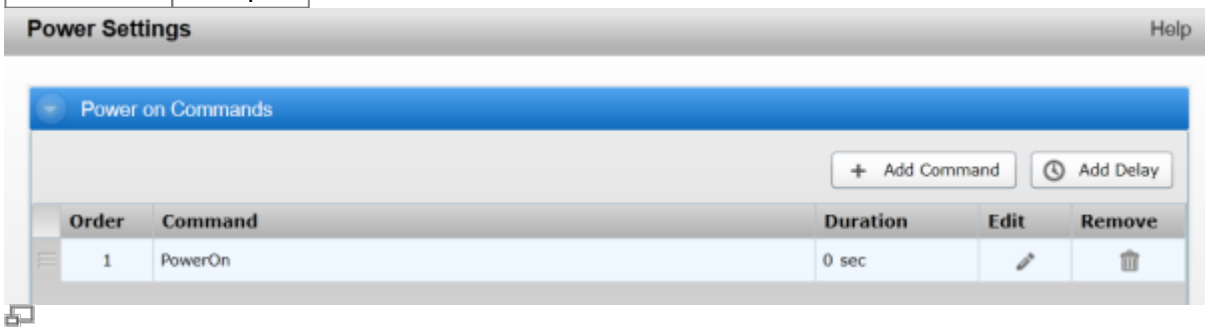


Logitech Harmony *Roku 4* device configuration

Click on Add and then just follow the instructions on the Logitech Harmony Desktop application, it's all straight forward.

Note that the default configuration of the Roku 4 commands is misconfigured somehow. For example: If you press the Roku 4's Home button on the Logitech Harmony remote, HA recognizes it as `Info` command and vice versa. While this misconfiguration is only true for a couple of buttons, most of them are recognized normally. Just be sure that for default *Power on* and *Power off* settings the according keys are mapped. From the Logitech Harmony Desktop application's main menu you can set those settings by navigating to: `Devices` → `Change device settings` → `Power settings` → `Next` → `I want to turn off this devicce when not in use.` → `Next` → `I press two different buttons for on and for off` → `Next`

Setting	**Mapping**
`Power on`	`PowerOn`
`Power off`	`Sleep`



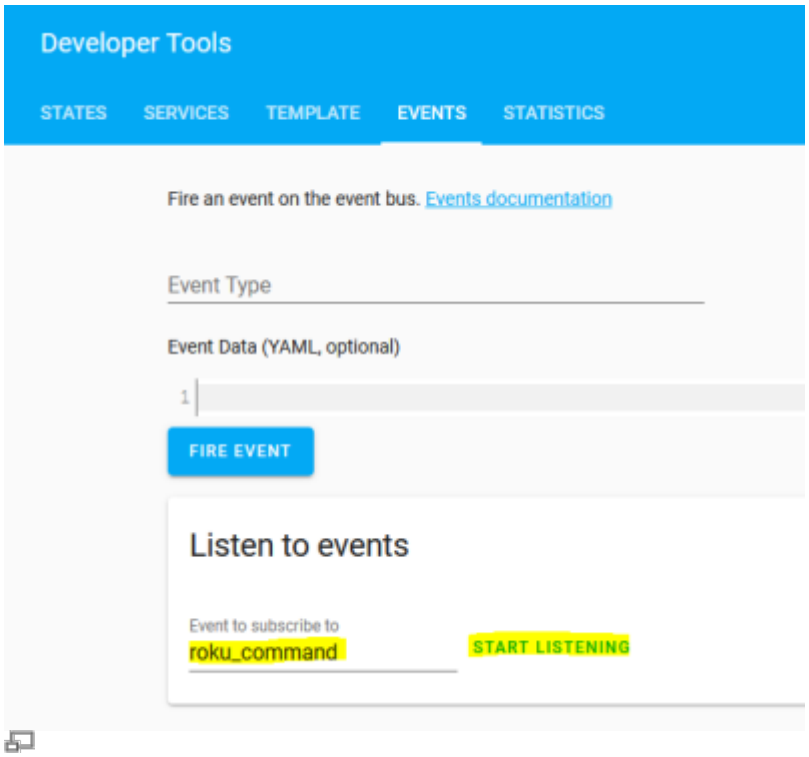
Power on mapping

Power off mapping

Before going further now, make sure all of your Logitech Harmony remote devices (e.g. your physical Harmony remote control and/or Logitech smartphone app) are synced with the newly added configuration.

Testing steps

You can now already test if your HA instance is able to communicate with your Logitech Harmony's `Roku 4` device by navigating from the HA main menu to:
`Developer Tools` → `EVENTS`
Within the `Event to subsrive to` input screen insert the following value:
`roku_command`
and then click on `START LISTENING`:



Home Assistant event listener

Now take your preferred Logitech remote control (Harmony remote control or Logitech smartphone app), select the `Roku 4` device and press a random key. The received command(s) should now show up in HA.

Example:

If this is not the case then something went wrong and you should go through the steps conscientiously again.

Now you know how to find out how a specific Roku 4 command sent from your Logitech Harmony remote is being recognized by HA. This gives you the possibility to use any of those commands for specific actions/automations in HA. Nevertheless, to prevent uneccessary issues, it is strongly suggested to use all the commands and settings the same way they are being used in this tutorial.

Also, if you want to check if the command misconfigurations mentioned above were being solved, you can test it anytime with the `roku_command` event listening method.

Creating startup/shutdown scripts in Home Assistant

Now comes the interesting part with a little bit of .yaml automation steps in HA.

We will have two main parts here: One for **startup** and one for **shutdown** you Batocera system.

Let's do the **startup** part first:

For your understanding: HA has its own Wake on LAN (WoL) client already installed by default so you can use it right out of the box.

From the HA main menu, navigate to:
`Configuration` → `Automations & Scenes` → `Scripts` → `+ ADD SCRIPT`
Configure the following values (leave the rest untouched on default settings):

Setting	**Value**
`Name`	<your_Custom_Script_Name> (e.g. `PowerOnBatocera`)
`Action type`	`Call service`
`Service`	`wake_on_lan.send_magic_packet`
`MAC address`	<your_Baterocera_system's_MAC_address> (e.g. `00:80:41:AE:FD:7E`)

Now click on `SAVE SCRIPT`.

Now let's do the **shutdown** part:
From the HA main menu go to the sidebar and click on `File editor` (if you don't have that option please install the File editor HA add-on first by using the HA's `Add-ons` menu). Then open the file `/config/configuration.yaml` and paste the following code:

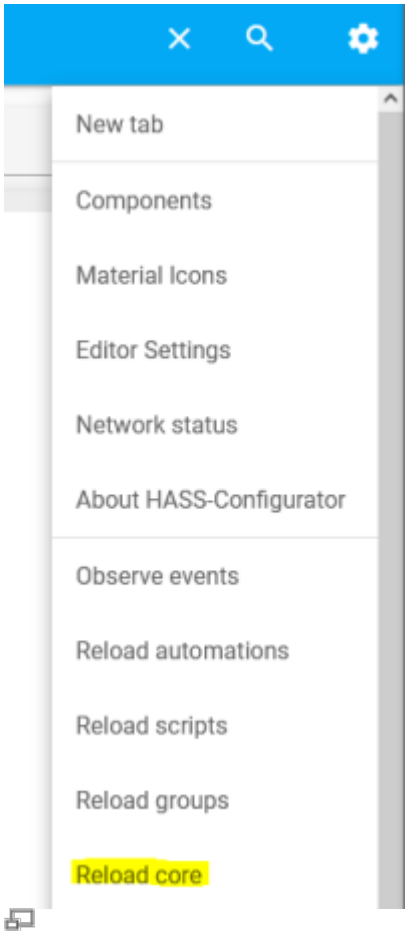
```
shell_command:
  batocera_poweroff: ssh -i /config/.ssh/id_rsa -o
'StrictHostKeyChecking=no'
root@<your_Baterocera_system's_static_IP_address/hostname> 'batocera-es-
swissknife --shutdown' > /dev/null 2>&1 &
```

Example:

```
# Shell commands:
# Beware: It is mandatory to place the SSH authentication keys inside the '/config/.ssh' directory! Other directories will not work for automations!
shell_command:
  batocera_poweroff: ssh -i /config/.ssh/id_rsa -o 'StrictHostKeyChecking=no' root@10.2.1.3 'batocera-es-swissknife --shutdown' > /dev/null 2>&1 &
```

configuration.yaml shell example

Now reload your HA core from within the file editor:



Reload core

Now, from the HA main menu, navigate to:
`Configuration` → `Automations & Scenes` → `Scripts` → `+ ADD SCRIPT`
Configure the following values (leave the rest untouched on default settings):

Setting	**Value**
`Name`	<your_Custom_Script_Name> (e.g. `PowerOffBatocera`)
`Action type`	`Call service`
`Service`	`kodi.call_method`
`Targets`	Click on `Choose entity` and select your Kodi instance which we had created before (e.g. `192.168.1.3`)

Setting	Value
Method	System.Shutdown

Now click on ADD ACTION...

Setting	Value
Action type	wait for time to pass (delay)
Duration	00:00:10:00

Now click on ADD ACTION...

Setting	Value
Action type	Call service
Service	shell_command.batocera_poweroff

Now click on SAVE SCRIPT.

Testing steps

You can now test if you can launch those scripts manually. If you launch the PowerOnBatocera script manually, the system should start. If you launch the PowerOffBatocera script manually, the system should power off. Test it by clicking on the accoridng *PowerOnBatocera* and *PowerOffBatocera* triangle button from the HA Configuration → Automation and Scripts → Scripts menu:

Launch scripts manually

If it does not work, something went wrong and you should go through the steps conscientiously again.

Creating automations in Home Assistant

Now we are going to do the real magic: The implementation for real home automation.
Let's do the **startup** automation first:
From the HA main menu navigate to:
Configuration → Automations & Scenes → Automations → + ADD AUTOMATION → Start with an empty automation
Configure the following values (leave the rest untouched on default settings):

Setting	Value
Name	<your_Custom_Automation_Name> (e.g. PowerOnBatocera)
Trigger type	Event
Event type	roku_command
Event data	source_name: EmulatedRoku and type: keypress and key: PowerOn

Setting	Value
Action type	Call service
Service	script.PowerOnBatocera

Now click on SAVE.

Now let's do the **shutdown** automation:
From the HA main menu navigate to:
Configuration → Automations & Scenes → Automations → + ADD AUTOMATION → Start with an empty automation
Configure the following values (leave the rest untouched on default settings):

Setting	Value
Name	<your_Custom_Automation_Name> (e.g. PowerOffBatocera)
Trigger type	Event
Event type	roku_command
Event data	source_name: EmulatedRoku and type: keypress and key: PowerOff
Action type	Call service
Service	script.PowerOffBatocera

Now click on SAVE.

That's it! Oh boy, how cool is that? You are now able to power on/off your Batocera system with just one click from your Logitech Harmony remote control. Feel free to integrate the Roku 4 device into your Logitech Harmony activities so you can power on/off all of your devices (e.g. TV, AV receiver, amplifier, HTPC, ...) with just one click with a single activity.