Home automation

Since Batocera is Wake on LAN (WoL) enabled by default and provides a MQTT pub/sub infrastructure, you can now integrate your Batocera system into any advanced home automation system, like Home Assistant.

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. Solutions like Home Assistant have fairly easy packages MQTT brokers available.

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
}
```

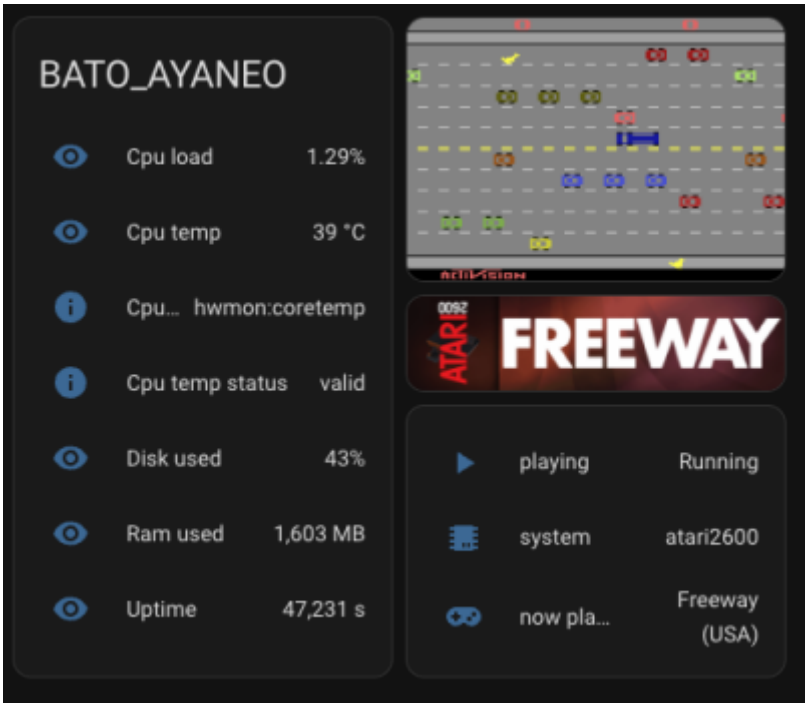
```
################################################################################
########
# Start the MQTT service.
# - Installs any missing support files
# - Publishes a system startup event
################################################################################
########
start() {
    install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

More Advanced Script

The following script is an extension of the previous script, that provides more data. Once enabled and integrated with a Home Assistant MQTT broker, you can transparently get cards like this:



To install this one:

1. copy this script into `/userdata/system/services/mqtt_events`
2. edit configuration variables at the top of the file with the credentials your MQTT broker
3. make it executable with `chmod +x /userdata/system/services/mqtt_events`
4. run `/userdata/system/services/mqtt_events install`
5. then, either go to EmulationStation and enable the service, or do it from the command line with `batocera-services enable mqtt_events` and then `batocera-services start mqtt_events`

mqtt_events

```
#!/bin/bash
# Batocera MQTT bridge (boxart + marquee + CPU-only telemetry pinned to
HWMON:temp)
# Initially from kit on Batocera Discord
# - CPU temp:
#   * Median-of-samples + sticky + last-good + hard range guards.
#   * Cached temp*_input path to avoid flicks and rescan churn.
# - Home Assistant MQTT discovery:
#   * Image: Box Art + Marquee (no YAML).
#   * Sensors: cpu_temp (+source/status), cpu_load, ram_used,
```

```
disk_used, uptime.
# - Debug topic: batocera/<host>/media_debug

#######################################
# Broker / topics
#######################################
MQTT_HOST=""     # Broker IP
MQTT_PORT=""     # Broker Port (usually 1883)
MQTT_USER=""     # username
MQTT_PASSWORD="" # Password

DISCOVERY_PREFIX="homeassistant"
HOSTNAME="$(hostname)"
MQTT_BASE_TOPIC="batocera/${HOSTNAME}"

# Telemetry
TELEMETRY_ENABLE=1
TELEMETRY_INTERVAL=5

# Media behavior
RETAIN_MEDIA_BYTES=1
PUBLISH_IMAGE_B64=1

# Paths / cache
PIDFILE="/var/run/mqtt_events.pid"
TMP_DIR="/tmp/mqtt_media"
mkdir -p "$TMP_DIR"
CORETEMP_PATH_CACHE="/tmp/mqtt_events_coretemp_path"
CPU_TEMP_STATE_FILE="/tmp/mqtt_events_cpu_temp_state"

# jq detection (optional)
HAS_JQ=0
command -v jq >/dev/null 2>&1 && HAS_JQ=1

#######################################
# ES service
#######################################
SERVICE_NAME=$(basename "$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)

#######################################
# Helpers
#######################################
_ts() { date -u +"%Y-%m-%dT%H:%M:%SZ"; }
_local_ts() { date +"%d/%m/%Y %H:%M:%S"; }

_json_array_from_args() {
  if [ "$HAS_JQ" -eq 1 ]; then
    local arr="["
```

```
    for arg in "$@"; do
      arr+=$(jq -Rn --arg a "$arg" '$a')
      arr+=","
    done
    echo "${arr%,}]"
  else
    local arr="["
    for arg in "$@"; do arr+="\"${arg//\"/\\\"}\","; done
    echo "${arr%,}]"
  fi
}

# Retained publishers
_pub() { mosquitto_pub -h "$MQTT_HOST" -p "$MQTT_PORT" -u "$MQTT_USER"
-P "$MQTT_PASSWORD" -t "$1" -m "$2" -r; }
_pub_str() { _pub "$1" "$2"; }

# Non-retained
_pub_event() { mosquitto_pub -h "$MQTT_HOST" -p "$MQTT_PORT" -u
"$MQTT_USER" -P "$MQTT_PASSWORD" -t "$1" -m "$2"; }

# File publisher (bytes)
_pub_file() {
  [ -f "$2" ] || return 1
  local args=(-h "$MQTT_HOST" -p "$MQTT_PORT" -u "$MQTT_USER" -P
"$MQTT_PASSWORD" -t "$1" -f "$2")
  [ "$RETAIN_MEDIA_BYTES" -eq 1 ] && args+=(-r)
  mosquitto_pub "${args[@]}"
}

_dbg() { _pub_event "${MQTT_BASE_TOPIC}/media_debug" "$1"; }

########################################
# Startup cleanup: purge any old GPU retained topics/entities
# Unused, kept as an example for clearing out entities
########################################
_clear_legacy_gpu_topics() {
  local ID="batocera_${HOSTNAME}"
  # HA discovery (GPU)
  for t in \
    "${DISCOVERY_PREFIX}/sensor/${ID}_gpu_temp/config" \
    "${DISCOVERY_PREFIX}/sensor/${ID}_gpu_temp_source/config" \
    "${DISCOVERY_PREFIX}/sensor/${ID}_gpu_temp_status/config"; do
    mosquitto_pub -h "$MQTT_HOST" -p "$MQTT_PORT" -u "$MQTT_USER" -P
"$MQTT_PASSWORD" -r -n -t "$t"
  done
  # State topics (GPU)
  for t in gpu_temp gpu_temp_source gpu_temp_status; do
    mosquitto_pub -h "$MQTT_HOST" -p "$MQTT_PORT" -u "$MQTT_USER" -P
"$MQTT_PASSWORD" -r -n -t "${MQTT_BASE_TOPIC}/sensors/${t}"
  done
```

```
}

########################################
# Converters: dwebp | magick | convert | ffmpeg
########################################
_convert_webp_to_png() {
  local in="$1" out
  out="$(mktemp -p "$TMP_DIR" media_XXXXXX.png)"
  if command -v dwebp >/dev/null 2>&1; then dwebp "$in" -o "$out"
>/dev/null 2>&1 && {
    echo "$out"
    return 0
  }; fi
  if command -v magick >/dev/null 2>&1; then magick "$in" "$out"
>/dev/null 2>&1 && {
    echo "$out"
    return 0
  }; fi
  if command -v convert >/dev/null 2>&1; then convert "$in" "$out"
>/dev/null 2>&1 && {
    echo "$out"
    return 0
  }; fi
  if command -v ffmpeg >/dev/null 2>&1; then ffmpeg -y -loglevel error
-i "$in" "$out" >/dev/null 2>&1 && {
    echo "$out"
    return 0
  }; fi
  return 1
}
_maybe_convert_to_png() {
  local in="$1" ext="${1##*.}"
  case "${ext,,}" in
  webp)
    local out
    out="$(_convert_webp_to_png "$in")" || {
      _dbg "CONVERT_MISSING $in"
      echo "$in"
      return 0
    }
    _dbg "CONVERTED_WEBP $in -> $out"
    echo "$out"
    ;;
  *) echo "$in" ;;
  esac
}

########################################
# Media discovery (box-art + marquee)
########################################
_find_game_image() {
```

```
  local sys="$1" romp="$2" romn="$3" base stem name p line candidate
  base="$(basename "$romp")"
  stem="${base%.*}"
  name="$romn"
  # 1) roms/<system>/images
  for ext in webp png jpg jpeg JPG JPEG PNG; do
    for candidate in \
      "/userdata/roms/$sys/images/${stem}-image.$ext" \
      "/userdata/roms/$sys/images/${stem}.$ext" \
      "/userdata/roms/$sys/images/${name//\//-}-image.$ext" \
      "/userdata/roms/$sys/images/${name//\//-}.$ext"; do [ -f
"$candidate" ] && echo "$candidate" && return 0; done
  done
  # 2) downloaded_images/<system>
  for root in
"/userdata/system/configs/emulationstation/downloaded_images/$sys"
"$HOME/.emulationstation/downloaded_images/$sys"; do
    [ -d "$root" ] || continue
    for ext in webp png jpg jpeg JPG JPEG PNG; do
      for candidate in \
        "$root/${name//\//-}-image.$ext" \
        "$root/${name//\//-}.$ext" \
        "$root/${stem}-image.$ext" \
        "$root/${stem}.$ext"; do [ -f "$candidate" ] && echo
"$candidate" && return 0; done
    done
  done
  # 3) <image> in gamelist.xml
  local gl="/userdata/roms/$sys/gamelist.xml"
  if [ -f "$gl" ]; then
    p="$(grep -F -n "<path>${romp}</path>" "$gl" | head -n1 | cut -d: -
f1)"
    if [ -n "$p" ]; then
      line="$(tail -n +$((p)) "$gl" | head -n 50 | grep -m1 -o
'<image>[^<]*</image>' | sed -e 's#<image>##' -e 's#</image>##')"
      [ -n "$line" ] && line="${line/#\~/$HOME}"
      case "$line" in ./*) line="/userdata/roms/$sys/${line#./}" ;;
esac
      [ -f "$line" ] && echo "$line" && return 0
    fi
  fi
  return 1
}
_find_marquee_image() {
  local sys="$1" romp="$2" romn="$3" base stem name p line candidate
  base="$(basename "$romp")"
  stem="${base%.*}"
  name="$romn"
  # 1) roms/<system>/images (also accept -wheel/-logo variants)
  for ext in webp png jpg jpeg JPG JPEG PNG; do
    for candidate in \
```

```
      "/userdata/roms/$sys/images/${stem}-marquee.$ext" \
      "/userdata/roms/$sys/images/${name//\//-}-marquee.$ext" \
      "/userdata/roms/$sys/images/${stem}-wheel.$ext" \
      "/userdata/roms/$sys/images/${name//\//-}-wheel.$ext" \
      "/userdata/roms/$sys/images/${stem}-logo.$ext" \
      "/userdata/roms/$sys/images/${name//\//-}-logo.$ext"; do [ -f
"$candidate" ] && echo "$candidate" && return 0; done
  done
  # 2) downloaded_images/<system>
  for root in
"/userdata/system/configs/emulationstation/downloaded_images/$sys"
"$HOME/.emulationstation/downloaded_images/$sys"; do
    [ -d "$root" ] || continue
    for ext in webp png jpg jpeg JPG JPEG PNG; do
      for candidate in \
        "$root/${name//\//-}-marquee.$ext" \
        "$root/${stem}-marquee.$ext" \
        "$root/${name//\//-}-wheel.$ext" \
        "$root/${stem}-wheel.$ext" \
        "$root/${name//\//-}-logo.$ext" \
        "$root/${stem}-logo.$ext"; do [ -f "$candidate" ] && echo
"$candidate" && return 0; done
    done
  done
  # 3) <marquee> or <thumbnail> in gamelist.xml
  local gl="/userdata/roms/$sys/gamelist.xml"
  if [ -f "$gl" ]; then
    p="$(grep -F -n "<path>${romp}</path>" "$gl" | head -n1 | cut -d: -
f1)"
    if [ -n "$p" ]; then
      for tag in marquee thumbnail; do
        line="$(tail -n +$((p)) "$gl" | head -n 60 | grep -m1 -o
"<$tag>[^<]*</$tag>" | sed -e "s#<$tag>##" -e "s#</$tag>##")"
        [ -n "$line" ] && line="${line/#\~/$HOME}"
        case "$line" in ./*) line="/userdata/roms/$sys/${line#./}" ;;
esac
        [ -f "$line" ] && echo "$line" && return 0
      done
    fi
  fi
  return 1
}

_publish_file_variants() {
  local kind="$1" fp="$2" ext
  [ -f "$fp" ] || {
    _dbg "NOT_FOUND $kind $fp"
    return 1
  }
  ext="${fp##*.}"
  if [ "${ext,,}" = "webp" ]; then
```

```
    local conv
    conv="$(_maybe_convert_to_png "$fp")"
    fp="$conv"
  fi
  _pub_event "${MQTT_BASE_TOPIC}/emulationstation/game-
selected/${kind}_path" "$fp"
  _pub_file "${MQTT_BASE_TOPIC}/emulationstation/game-
selected/${kind}_bytes" "$fp"
  if [ "$PUBLISH_IMAGE_B64" -eq 1 ] && command -v base64 >/dev/null
2>&1; then
    base64 "$fp" | tr -d '\n' | mosquitto_pub -h "$MQTT_HOST" -p
"$MQTT_PORT" -u "$MQTT_USER" -P "$MQTT_PASSWORD" \
      -t "${MQTT_BASE_TOPIC}/emulationstation/game-
selected/${kind}_b64" -s
  fi
}

_publish_media_for_selection() {
  local sys="$1" romp="$2" romn="$3" fp
  fp="$(_find_game_image "$sys" "$romp" "$romn")" &&
_publish_file_variants "image" "$fp" || _dbg "NO_IMAGE $sys $romp"
  fp="$(_find_marquee_image "$sys" "$romp" "$romn")" &&
_publish_file_variants "marquee" "$fp" || _dbg "NO_MARQUEE $sys $romp"
}

########################################
# Core publish
########################################
publish() {
  local subtopic="$1"
  shift
  local data ts
  data=$(_json_array_from_args "$@")
  ts=$(_ts)
  _pub "${MQTT_BASE_TOPIC}/${subtopic}" "{\"data\": ${data},
\"timestamp\": \"${ts}\"}"
  case "$subtopic" in
  emulationstation/game-start)
    local rom_path rom_name
    if [ "$HAS_JQ" -eq 1 ]; then
      rom_path=$(jq -r '.[0] // empty' <<<"$data")
      rom_name=$(jq -r '.[1] // .[0]' <<<"$data")
    else
      rom_path="${1:-}"
      rom_name="${2:-$rom_path}"
    fi
    [ -n "$rom_name" ] && _pub_str
"${MQTT_BASE_TOPIC}/emulationstation/rom" "$rom_name"
    _pub_str "${MQTT_BASE_TOPIC}/emulationstation/state" "playing"
    ;;
  emulationstation/game-end | game/gameStop | emulationstation/quit |
```

```
emulationstation/reboot | emulationstation/shutdown | system/shutdown |
emulationstation/screensaver-stop | emulationstation/sleep |
emulationstation/wake)
    _pub_str "${MQTT_BASE_TOPIC}/emulationstation/state" "menu"
    _pub_str "${MQTT_BASE_TOPIC}/emulationstation/rom" ""
    ;;
  emulationstation/system-selected)
    local sys
    if [ "$HAS_JQ" -eq 1 ]; then sys=$(jq -r '.[0] // empty'
<<<"$data"); else sys="${1:-}"; fi
    [ -n "$sys" ] && _pub_str
"${MQTT_BASE_TOPIC}/emulationstation/system" "$sys"
    ;;
  emulationstation/game-selected)
    local sys romp rom_name ts_local
    ts_local="$(_local_ts)"
    if [ "$HAS_JQ" -eq 1 ]; then
      sys=$(jq -r '.[0] // empty' <<<"$data")
      romp=$(jq -r '.[1] // empty' <<<"$data")
      rom_name=$(jq -r '.[2] // .[1] // .[0]' <<<"$data")
    else
      sys="${1:-}"
      romp="${2:-}"
      rom_name="${3:-${2:-$1}}"
    fi
    [ -n "$sys" ] && _pub_str
"${MQTT_BASE_TOPIC}/emulationstation/system" "$sys"
    [ -n "$rom_name" ] && _pub_str
"${MQTT_BASE_TOPIC}/emulationstation/rom" "$rom_name"
    if [ "$HAS_JQ" -eq 1 ]; then
      _pub_event "${MQTT_BASE_TOPIC}/emulationstation/game-
selected/pretty" "$(jq -n --argjson data "$data" --arg ts "$ts"
'{data:$data, timestamp:$ts}')"
    else
      _pub_event "${MQTT_BASE_TOPIC}/emulationstation/game-
selected/pretty" "{\"data\": ${data}, \"timestamp\": \"${ts}\"}"
    fi
    _pub_event "${MQTT_BASE_TOPIC}/emulationstation/game-
selected/human_ts" "$ts_local"
    _pub_event "${MQTT_BASE_TOPIC}/emulationstation/game-
selected/compact" "{\"data\": ${data}, \"timestamp\": \"${ts}\"}"
    _publish_media_for_selection "$sys" "$romp" "$rom_name"
    ;;
  game/gameStart)
    local sys
    if [ "$HAS_JQ" -eq 1 ]; then sys=$(jq -r '.[0] // empty'
<<<"$data"); else sys="${1:-}"; fi
    [ -n "$sys" ] && _pub_str
"${MQTT_BASE_TOPIC}/emulationstation/system" "$sys"
    _pub_str "${MQTT_BASE_TOPIC}/emulationstation/state" "playing"
    ;;
```

```
  esac
}

#######################################
# HA MQTT Discovery (images + CPU telemetry only)
#######################################
ha_discover() {
  local ID="batocera_${HOSTNAME}"

  # CPU/system sensors
  declare -A SENS=([cpu_temp]="°C" [cpu_load]="%" [ram_used]="MB"
[disk_used]="%" [uptime]="s")
  for key in cpu_temp cpu_load ram_used disk_used uptime; do
    _pub "${DISCOVERY_PREFIX}/sensor/${ID}_${key}/config" \
      "{
        \"name\": \"${key//_/ }\",
        \"unique_id\": \"${ID}_${key}\",
        \"state_topic\": \"${MQTT_BASE_TOPIC}/sensors/${key}\",
        \"unit_of_measurement\": \"${SENS[$key]}\",
        \"availability_topic\": \"${MQTT_BASE_TOPIC}/availability\",
        \"device\":
{\"identifiers\":[\"${ID}\"],\"name\":\"${HOSTNAME}\"}
      }"
  done

  # Diagnostics for CPU temp
  for diag in cpu_temp_source cpu_temp_status; do
    _pub "${DISCOVERY_PREFIX}/sensor/${ID}_${diag}/config" \
      "{
        \"name\": \"${diag//_/ }\",
        \"unique_id\": \"${ID}_${diag}\",
        \"state_topic\": \"${MQTT_BASE_TOPIC}/sensors/${diag}\",
        \"icon\": \"mdi:information\",
        \"availability_topic\": \"${MQTT_BASE_TOPIC}/availability\",
        \"device\":
{\"identifiers\":[\"${ID}\"],\"name\":\"${HOSTNAME}\"}
      }"
  done

  # Now Playing / System / Playing
  _pub "${DISCOVERY_PREFIX}/sensor/${ID}_now_playing/config" \
    "{
      \"name\":\"now playing\",
      \"unique_id\":\"${ID}_now_playing\",
      \"state_topic\":\"${MQTT_BASE_TOPIC}/emulationstation/rom\",
      \"availability_topic\":\"${MQTT_BASE_TOPIC}/availability\",
      \"icon\":\"mdi:gamepad-variant\",
      \"device\": {\"identifiers\":[\"${ID}\"]}
    }"
  _pub "${DISCOVERY_PREFIX}/sensor/${ID}_system/config" \
    "{
```

```
      \"name\":\"system\",
      \"unique_id\":\"${ID}_system\",
      \"state_topic\":\"${MQTT_BASE_TOPIC}/emulationstation/system\",
      \"availability_topic\":\"${MQTT_BASE_TOPIC}/availability\",
      \"icon\":\"mdi:chip\",
      \"device\": {\"identifiers\":[\"${ID}\"]}
    }"
  _pub "${DISCOVERY_PREFIX}/binary_sensor/${ID}_playing/config" \
    "{
      \"name\":\"playing\",
      \"unique_id\":\"${ID}_playing\",
      \"state_topic\":\"${MQTT_BASE_TOPIC}/emulationstation/state\",
      \"payload_on\":\"playing\",
      \"payload_off\":\"menu\",
      \"device_class\":\"running\",
      \"availability_topic\":\"${MQTT_BASE_TOPIC}/availability\",
      \"device\": {\"identifiers\":[\"${ID}\"]}
    }"

  # Buttons
  for cmd in reboot shutdown; do
    _pub "${DISCOVERY_PREFIX}/button/${ID}_${cmd}/config" \
      "{
        \"name\":\"${cmd^} ${HOSTNAME}\",
        \"unique_id\":\"${ID}_${cmd}\",
        \"command_topic\":\"${MQTT_BASE_TOPIC}/command\",
        \"payload_press\":\"${cmd}\",
        \"availability_topic\":\"${MQTT_BASE_TOPIC}/availability\",
        \"icon\":\"mdi:${cmd}\",
        \"device\": {\"identifiers\":[\"${ID}\"]}
      }"
  done

  # Images
  _pub "${DISCOVERY_PREFIX}/image/${HOSTNAME}/boxart/config" \
    "{
      \"name\": \"box art\",
      \"unique_id\": \"${ID}_boxart\",
      \"image_topic\": \"${MQTT_BASE_TOPIC}/emulationstation/game-
selected/image_bytes\",
      \"availability_topic\": \"${MQTT_BASE_TOPIC}/availability\",
      \"device\":
{\"identifiers\":[\"${ID}\"],\"name\":\"${HOSTNAME}\"}
    }"
  _pub "${DISCOVERY_PREFIX}/image/${HOSTNAME}/marquee/config" \
    "{
      \"name\": \"marquee\",
      \"unique_id\": \"${ID}_marquee\",
      \"image_topic\": \"${MQTT_BASE_TOPIC}/emulationstation/game-
selected/marquee_bytes\",
      \"availability_topic\": \"${MQTT_BASE_TOPIC}/availability\",
```

```
      \"device\":
{\"identifiers\":[\"${ID}\"],\"name\":\"${HOSTNAME}\"}
    }"
}

#######################################
# CPU temperature
#######################################
CPU_TEMP_SAMPLE_COUNT=3
CPU_TEMP_SAMPLE_DELAY_MS=150
CPU_TEMP_STICKY_SECONDS=600
CPU_TEMP_MIN_VALID=-40
CPU_TEMP_MAX_VALID=125

_norm_celsius() {
  local v="$1"
  [ -z "$v" ] && return 1
  v="$(echo "$v" | tr -d '[:space:]')"
  case "$v" in '' | *[!0-9-]*) return 1 ;; esac
  if [ "$v" -ge 1000 ] || [ "$v" -le -1000 ]; then echo $((v / 1000));
else echo "$v"; fi
}
_is_valid_temp() {
  local c="$1"
  [ -z "$c" ] && return 1
  [ "$c" -lt "$CPU_TEMP_MIN_VALID" ] && return 1
  [ "$c" -gt "$CPU_TEMP_MAX_VALID" ] && return 1
  return 0
}
_sleep_ms() { usleep "$(($1 * 1000))" 2>/dev/null || sleep "$(awk
"BEGIN{print $1/1000}")"; }

_resolve_coretemp_path() {
  # Use cached path if still valid
  if [ -f "$CORETEMP_PATH_CACHE" ]; then
    local p
    p="$(cat "$CORETEMP_PATH_CACHE" 2>/dev/null)"
    [ -n "$p" ] && [ -f "$p" ] && echo "$p" && return 0
  fi
  # Scan for 'coretemp' hwmon and select Package/Physical/CPU label;
else first found
  for h in /sys/class/hwmon/hwmon*; do
    [ -f "$h/name" ] || continue
    local name
    name="$(cat "$h/name" 2>/dev/null)"
    case "${name,,}" in *temp*)
      local first=""
      for n in "$h"/temp*_input; do
        [ -f "$n" ] || continue
        [ -z "$first" ] && first="$n"
        local lblfile="${n%_input}_label" lbl=""
```

```
        [ -f "$lblfile" ] && lbl="$(cat "$lblfile" 2>/dev/null)"
        if echo "$lbl" | grep -qiE 'package|physical|cpu'; then
          echo "$n" | tee "$CORETEMP_PATH_CACHE"
          return 0
        fi
      done
      if [ -n "$first" ]; then
        echo "$first" | tee "$CORETEMP_PATH_CACHE"
        return 0
      fi
      ;;
    esac
  done
  return 1
}

_read_coretemp_once() {
  local p
  p="$(_resolve_coretemp_path)" || return 1
  local raw
  raw="$(cat "$p" 2>/dev/null)" || return 1
  local c
  c="$(_norm_celsius "$raw")" || return 1
  echo "$c"
  return 0
}

_sample_coretemp_median() {
  local n="${CPU_TEMP_SAMPLE_COUNT:-3}"
d="${CPU_TEMP_SAMPLE_DELAY_MS:-150}"
  local vals=() i=0 v
  while [ $i -lt "$n" ]; do
    v="$(_read_coretemp_once)" || {
      _sleep_ms "$d"
      i=$((i + 1))
      continue
    }
    vals+=("$v")
    i=$((i + 1))
    [ $i -lt "$n" ] && _sleep_ms "$d"
  done
  [ "${#vals[@]}" -eq 0 ] && return 1
  IFS=$'\n' vals=($(printf "%s\n" "${vals[@]}" | sort -n))
  unset IFS
  local mid=$(((${#vals[@]} - 1) / 2))
  echo "${vals[$mid]}"
  return 0
}

_publish_cpu_temp() {
  local now c last_c last_src last_ts sticky_src sticky_until
```

```
  now="$(date +%s)"
  [ -f "$CPU_TEMP_STATE_FILE" ] && IFS=';' read -r last_c last_src
last_ts sticky_src sticky_until <"$CPU_TEMP_STATE_FILE"

  c="$(_sample_coretemp_median)" || c=""

  if _is_valid_temp "$c"; then
    _pub_str "${MQTT_BASE_TOPIC}/sensors/cpu_temp" "$c"
    _pub_str "${MQTT_BASE_TOPIC}/sensors/cpu_temp_source"
"hwmon:coretemp"
    _pub_str "${MQTT_BASE_TOPIC}/sensors/cpu_temp_status" "valid"
    local until=$((now + CPU_TEMP_STICKY_SECONDS))
    echo "${c};hwmon:coretemp;${now};hwmon:coretemp;${until}"
>"$CPU_TEMP_STATE_FILE"
  else
    if _is_valid_temp "$last_c"; then
      _pub_str "${MQTT_BASE_TOPIC}/sensors/cpu_temp" "$last_c"
      _pub_str "${MQTT_BASE_TOPIC}/sensors/cpu_temp_source"
"${last_src:-hwmon:coretemp}"
      _pub_str "${MQTT_BASE_TOPIC}/sensors/cpu_temp_status" "stale"
    else
      _dbg "CPU_TEMP_UNAVAILABLE (coretemp not found or unreadable)"
    fi
  fi
}

######################################
# Command listener
######################################
listen_commands() {
  mosquitto_sub -h "$MQTT_HOST" -p "$MQTT_PORT" -u "$MQTT_USER" -P
"$MQTT_PASSWORD" \
    -t "${MQTT_BASE_TOPIC}/command" -q 0 | while read -r payload; do
    case "$payload" in
    reboot) reboot ;;
    shutdown) shutdown -h now ;;
    esac
  done &
  CMD_PID=$!
}

######################################
# Telemetry loop
######################################
telemetry_loop() {
  [ "$TELEMETRY_ENABLE" -eq 1 ] || return 0
  while true; do
    _publish_cpu_temp

    # CPU load (1min)
    [ -r /proc/loadavg ] && _pub_str
```

```
"${MQTT_BASE_TOPIC}/sensors/cpu_load" "$(awk '{print $1}'
/proc/loadavg)"

    # RAM used (MB)
    if [ -r /proc/meminfo ]; then
      local mt ma
      mt="$(awk '/MemTotal:/ {print $2}' /proc/meminfo)"
      ma="$(awk '/MemAvailable:/ {print $2}' /proc/meminfo)"
      [ -n "$mt" ] && [ -n "$ma" ] && _pub_str
"${MQTT_BASE_TOPIC}/sensors/ram_used" $(((mt - ma) / 1024))
    fi

    # Disk used (% of /userdata)
    if df -P /userdata >/dev/null 2>&1; then
      local du
      du="$(df -P /userdata | awk 'NR==2{gsub(/%/,"",$5);print $5}')"
      [ -n "$du" ] && _pub_str "${MQTT_BASE_TOPIC}/sensors/disk_used"
"$du"
    fi

    # Uptime (s)
    [ -r /proc/uptime ] && _pub_str "${MQTT_BASE_TOPIC}/sensors/uptime"
"$(awk '{print int($1)}' /proc/uptime)"

    sleep "$TELEMETRY_INTERVAL"
  done &
  TEL_PID=$!
}

#######################################
# Install / Uninstall ES hooks
#######################################
install() {
  mkdir -p /userdata/system/configs/emulationstation/scripts
  for event in "${ES_EVENTS[@]}"; do
    local
script_dir="/userdata/system/configs/emulationstation/scripts/${event}"
    mkdir -p "$script_dir"
    local script="${script_dir}/${SERVICE_NAME%.sh}.sh"
    if [ ! -f "$script" ]; then
      cat >"$script" <<'EOF'
#!/bin/bash
event_name="$(basename "$(dirname "$0")")"
/userdata/system/services/SERVICE_NAME publish
"emulationstation/${event_name}" "$@"
EOF
      sed -i "s/SERVICE_NAME/${SERVICE_NAME}/" "$script"
      chmod +x "$script"
    fi
  done
```

```
  mkdir -p /userdata/system/scripts
  local game_script="/userdata/system/scripts/${SERVICE_NAME%.sh}.sh"
  if [ ! -f "$game_script" ]; then
    cat >"$game_script" <<'EOF'
#!/bin/bash
# Args: gameStart|gameStop system emulator core rompath
/userdata/system/services/SERVICE_NAME publish "game/$1" "${@:2}"
EOF
    sed -i "s/SERVICE_NAME/${SERVICE_NAME}/" "$game_script"
    chmod +x "$game_script"
  fi
}
uninstall() {
  for event in "${ES_EVENTS[@]}"; do
    local
script_dir="/userdata/system/configs/emulationstation/scripts/${event}"
    local script="${script_dir}/${SERVICE_NAME%.sh}.sh"
    [ -f "$script" ] && rm -f "$script"
    [ -d "$script_dir" ] && [ -z "$(ls -A "$script_dir")" ] && rmdir
"$script_dir" 2>/dev/null
  done
  rm -f "/userdata/system/scripts/${SERVICE_NAME%.sh}.sh"
}

#######################################
# Process Control
#######################################
getMPID() {
  X=$(cat "${PIDFILE}" 2>/dev/null)
  test -z "${X}" && echo "" && return 1
  if test -e "/proc/${X}"; then
    echo "${X}"
    return 0
  fi
  echo ""
  return 1
}

#######################################
# Start / Stop
#######################################
start() {
  P=$(getMPID)
  if test -n "$P"; then
    kill "$P"
  fi
  echo $$ >"${PIDFILE}"
  install
  _pub_str "${MQTT_BASE_TOPIC}/availability" "online"
  publish "system/startup"
  ha_discover
```

```
  listen_commands
  telemetry_loop
}
stop() {
  P=$(getMPID)
  if test -n "$P"; then
    kill "$P"
  fi
  /bin/rm "${PIDFILE}"
  publish "system/shutdown"
  [ -n "$CMD_PID" ] && kill "$CMD_PID" 2>/dev/null
  [ -n "$TEL_PID" ] && kill "$TEL_PID" 2>/dev/null
  _pub_str "${MQTT_BASE_TOPIC}/availability" "offline"
}

########################################
# Entrypoint
########################################
if [ $# -eq 0 ]; then
  echo "Usage: $SERVICE_NAME publish|start|stop|install|uninstall
[args...]"
  exit 1
fi
case "$1" in
publish)
  shift
  publish "$@"
  ;;
start) start ;;
stop) stop ;;
install) install ;;
uninstall) uninstall ;;
*)
  echo "Unknown command: $1"
  exit 1
  ;;
esac
```

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

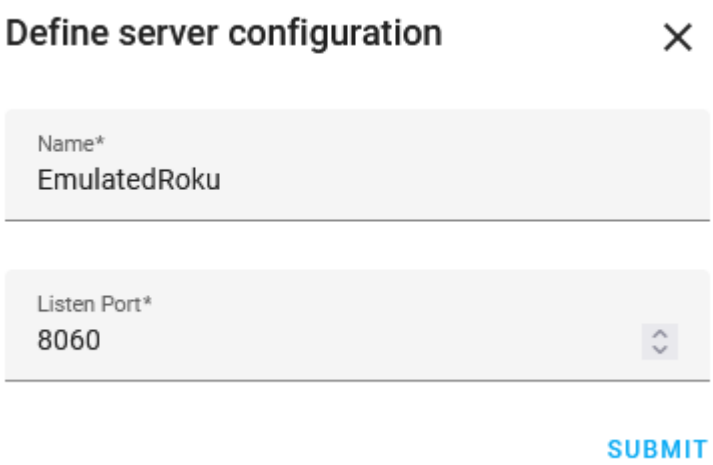
Note that for simplifying reasons we are not going to cover here on how to implement scripts/automations via separated *.yaml* files in HA, but for advanced users, of course you can feel free to do so. Also we are going to use manual *.yaml* code as little as possible here, instead we will configure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

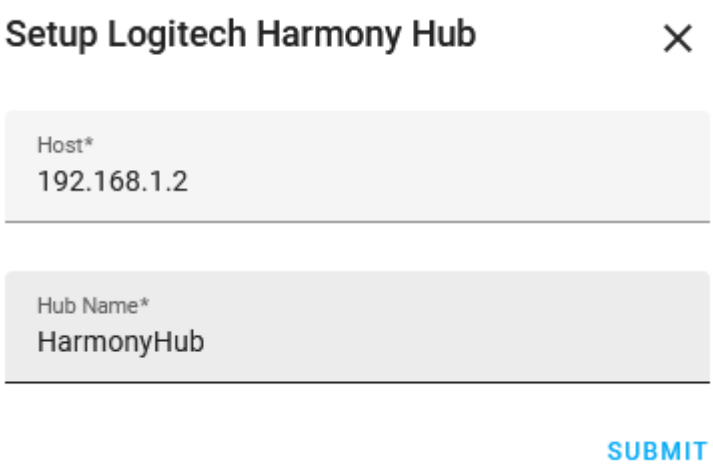
First, add the following three required **integrations** into HA (you can install them via the HA GUI if you are using at least a *supervised* HA installation (which is strongly recommended anyway or *-even way better(!)-* an official HAOS (Home Assistant Operating System) instance):

- 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`:



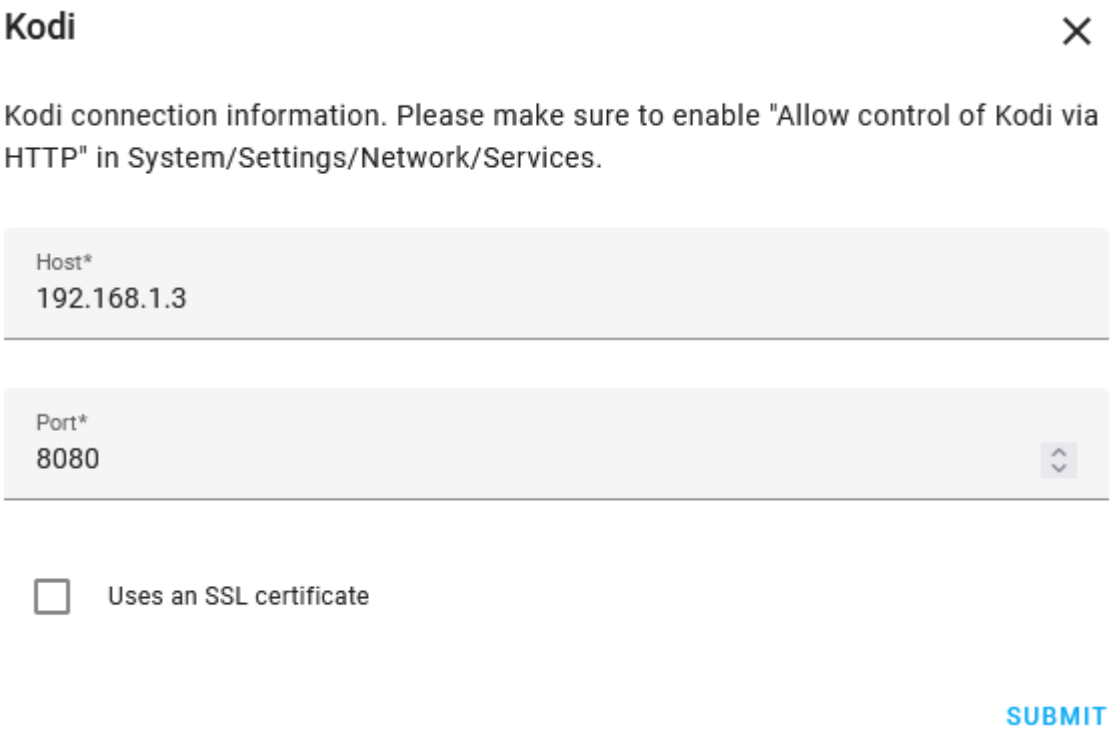
EmulatedRoku integration

- 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`:



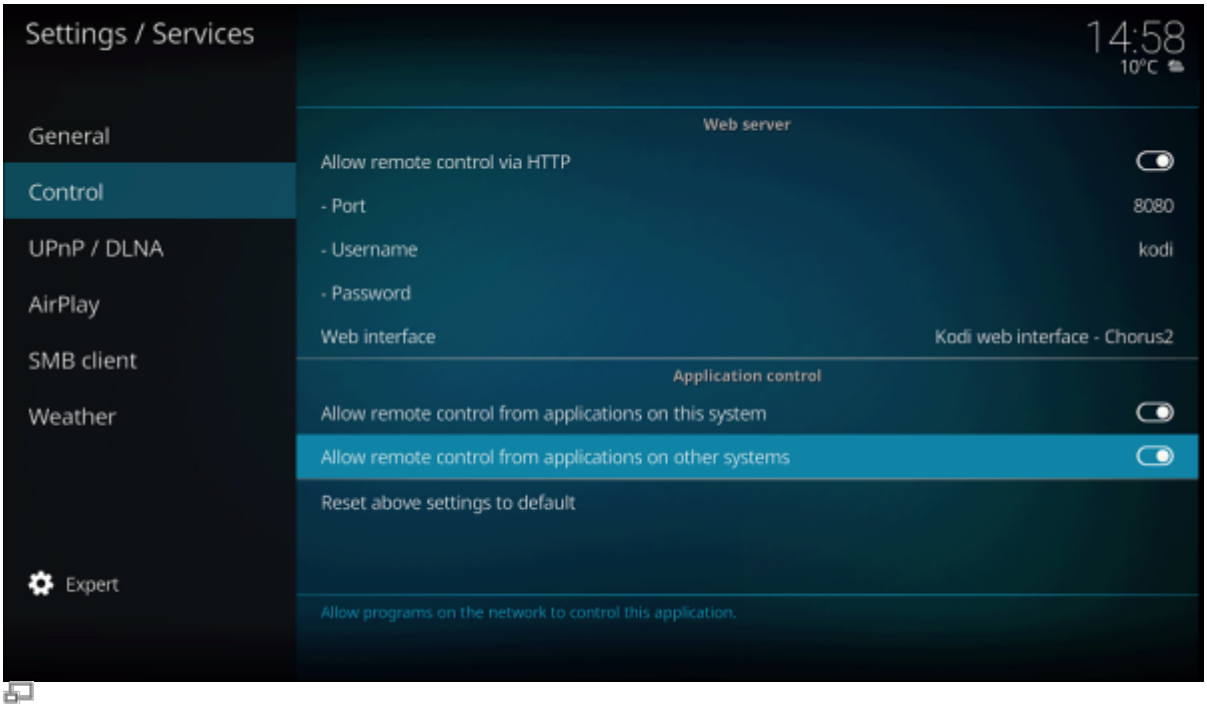
Logitech Harmony integration

- 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`).

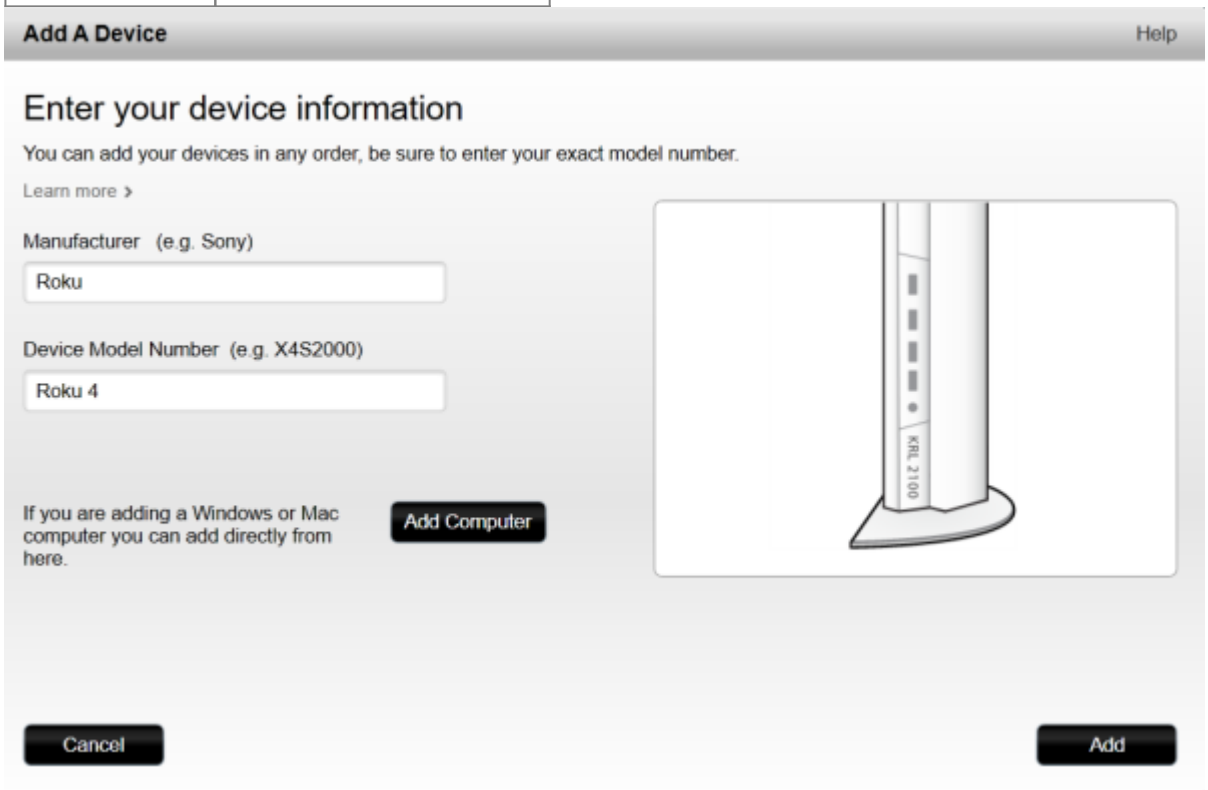
  Now test the passwordless SSH login from your HA instance to your Batocera system. If it works, go further.

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

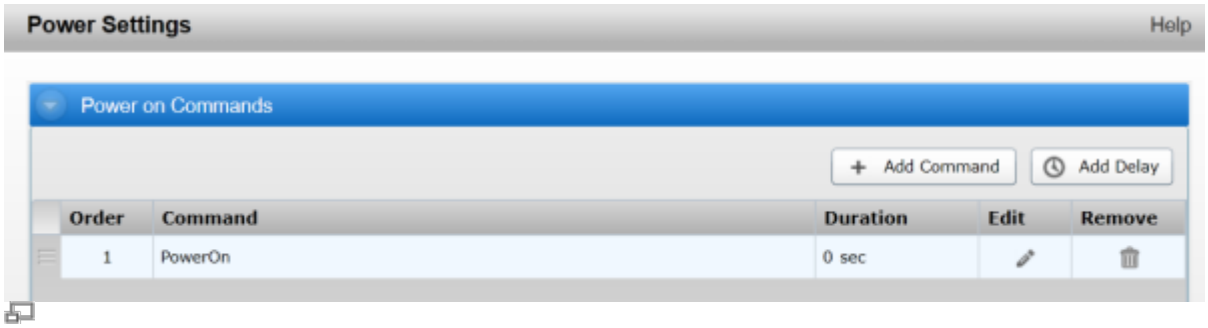


Logitech Harmony *Roku 4* device configuration

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`



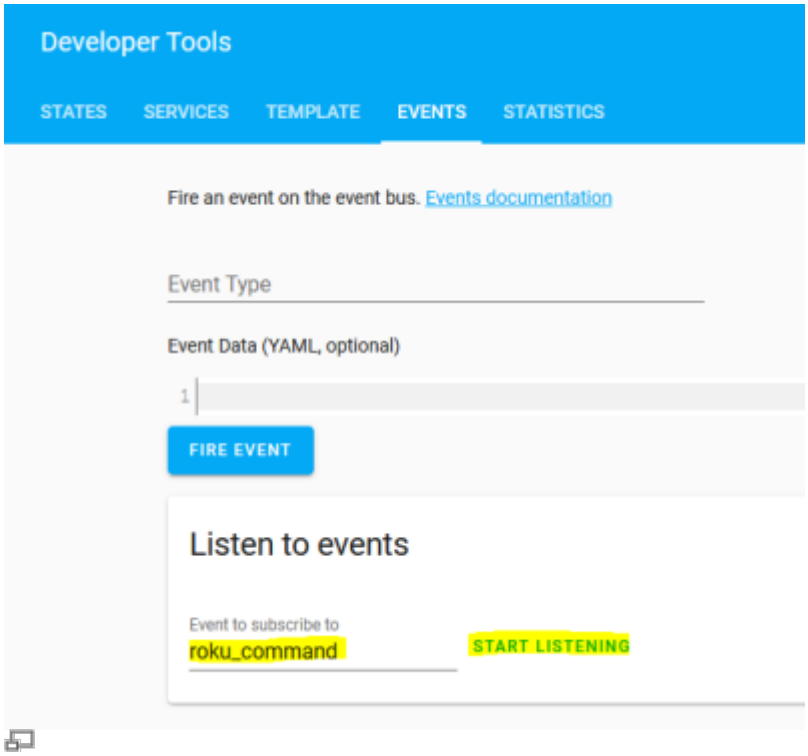
Power on mapping

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:

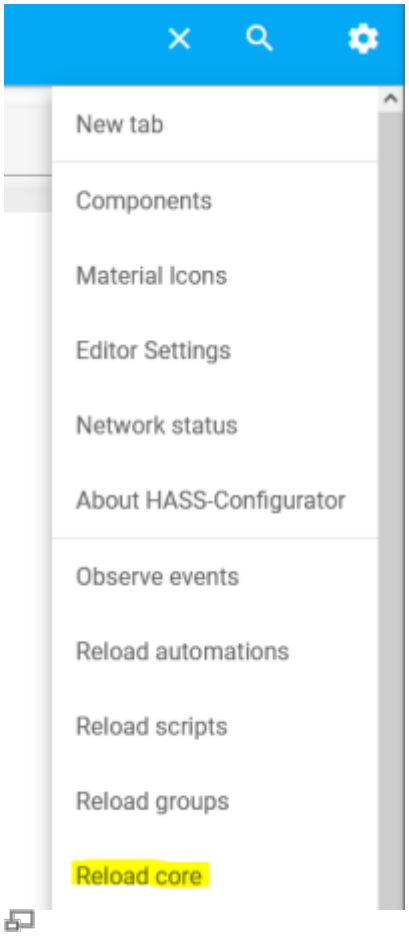
```
shell_command:
  batocera_poweroff: ssh -i /config/.ssh/id_rsa -o
'StrictHostKeyChecking=no'
root@<your_Baterocera_system's_static_IP_address/hostname> 'batocera-es-
swissknife --shutdown' > /dev/null 2>&1 &
```

Example:

```
64 # Shell commands:
65 # Beware: It is mandatory to place the SSH authentication keys inside the '/config/.ssh' directory! Other directories will not work for automations!
66 shell_command:
67   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`)
`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`
`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.

Advanced: Boot automatically into OSes on multiboot systems

In case you got Batocera set up as a dual- or multiboot system, you may be interested on how to set up PXE-boot (network boot) on your Batocera PC so you can boot automatically into one of the according installed OSes on your PC, based on Home Assistant automations. To accomplish this, please refer to this tutorial, which was made based on a Batocera and Windows 11 dual-boot installation.