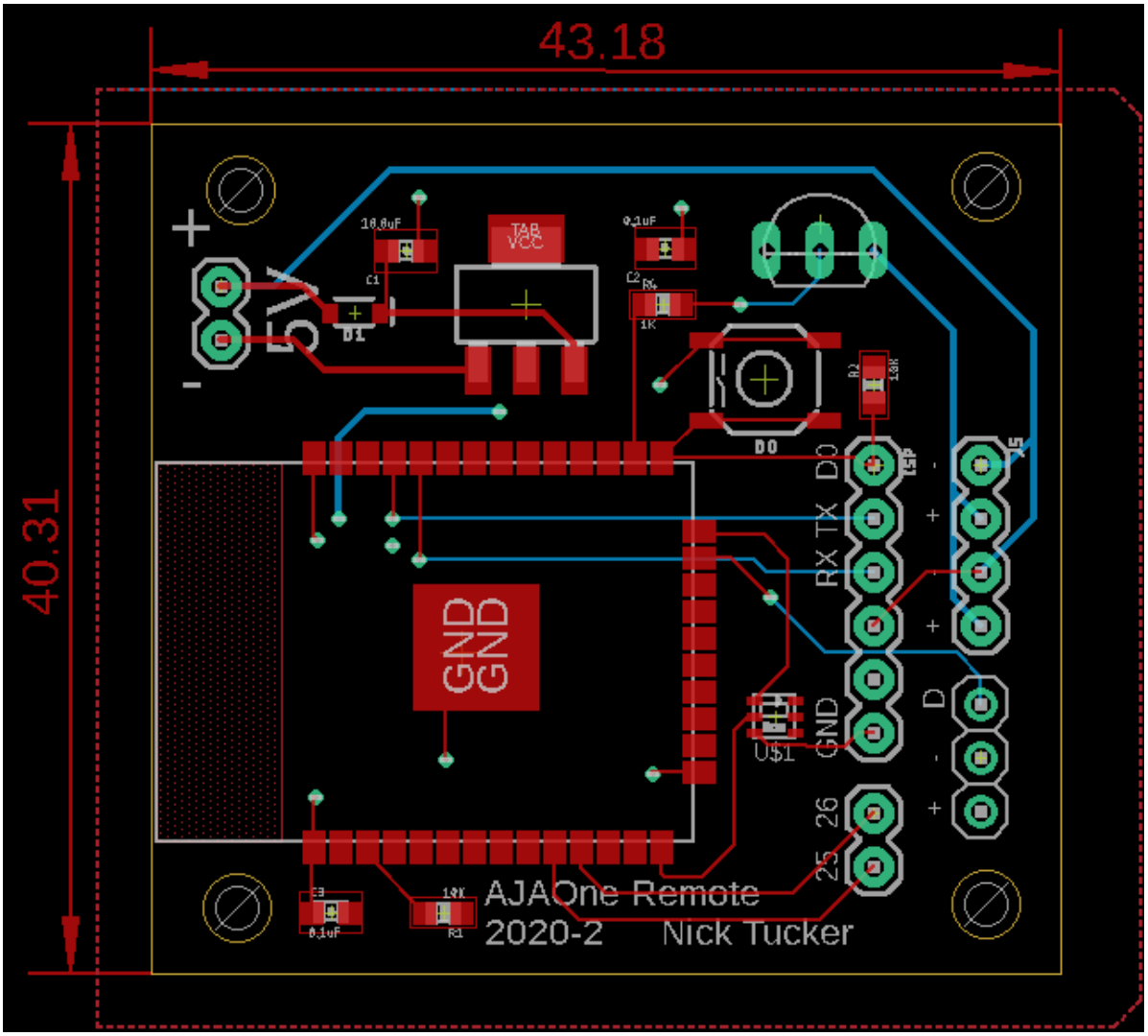
AjaRemote

Version 2020-2



Contents

Introduction

AjaRemote is part of the AjaOne family of devices that use the ESP32 and custom firmware to interface with an MQTT Broker. The AjaRemote is used to control most devices that use Infrared commands such as TV's, Blueray players, AV Receivers and so on.

Products such as Home Assistant, Node-Red and Mosquitto MQTT broker running on a Raspberry PI, and one or more AjaRemotes, make an excellent platform to automate anything controllable via Infrared.

Each AjaRemote has a built-in super bright 5mm infrared LED as well as an 38KHz infrared receiver. In addition, there is a 3.5mm jack to allow for additional infrared LEDs to be attached.

AjaRemote also provides basic HID keyboard controls via BTLe. These can be useful with such devices as Amazon's Fire Stick devices to move the cursor around.

License Model

Protocols

AjaRemote supports most of the major IR Protocols, such as:

Infrared

- Denon
- Dish
- JVC
- Lego
- NEC
- Panasonic
- Pronto
- Sony
- Samsung
- Sharp

Bluetooth LE

- Next Track
- Previous Track
- Stop
- Play / Pause
- Mute
- Volume up
- Volume down
- “Virtual Keys”.

Please see appendix for the full list.

Initial start up

In order to use the AjaRemote, it must first be connected to the local WiFi followed by the MQTT Broker.

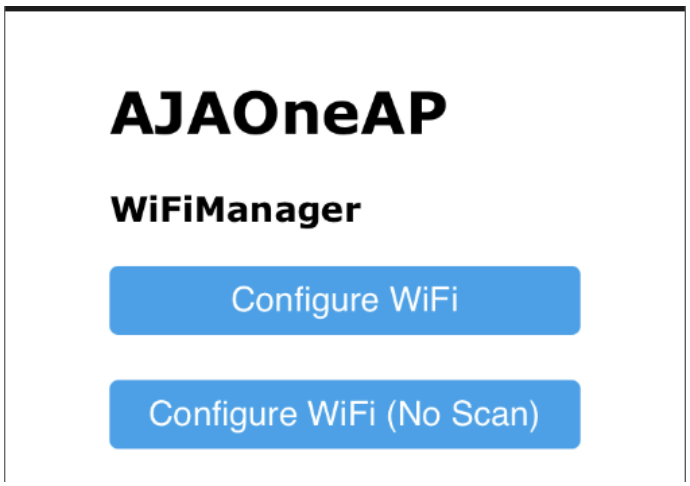
The information required will be:

1. WiFi SID (name of your WiFi access point)
2. WiFI password.
3. IP Address of the MQTT Broker.
4. MQTT Username (optional but recommended)
5. MQTT Password (optional but recommended).

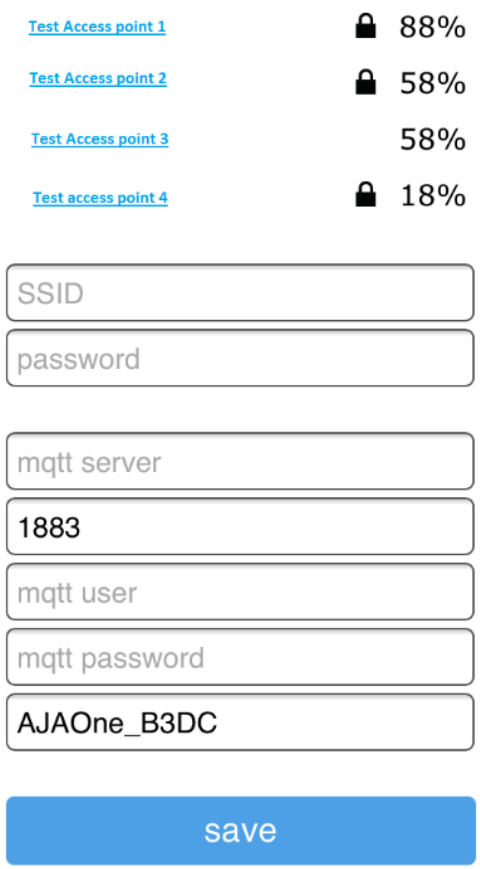
Configuration Steps

1. Connect the 5v power supply.
2. Wait approximately one minute for the system to configure itself and boot up.
3. Using a smart phone or tablet, go into Settings / WiFI and connect to the access point ***AJAoneAP***.

4. Depending on your device, the configuration page may show automatically. If not, open a browser to 192.168.4.1.



5. Click on ***Configure WiFI***. It will take a few moments to scan for all the available SIDs. If none are displayed, then move the AjaRemote closer to a WiFi access point.
6. Enter all the required information and click on ***Save***.



7. The AjaRemote will reboot and connect to the WiFI and the MQTT Broker.
8. If no Broker is found within five minutes, the AjaRemote will reboot and try again.

When the AjaRemote starts it's led will flash once to indicate the beginning of the boot sequence. Once the entire boot sequence is completed the led will flash three times.

If for some reason the MQTT Broker cannot be reached the LED will turn blue and remain on until a connection is made. If no connection is made in five minutes, it will reboot and try again.

Addressing

Each AjaRemote will have a unique name assigned to it. This name will be in the format of ***AJaRemoteBT_XXXX*** where XXXX is the last four digits of the device MAC address. For example, ***AjaRemoteBT_AD34***.

LED Codes

The on board RGB LED will indicate the current state of operation.

- Yellow - Booting.
- Purple - WiFi Starting.
- Blue - Attempting MQTT connection.
- Solid red - Performing reset.
- Blue-purple - A command was received.

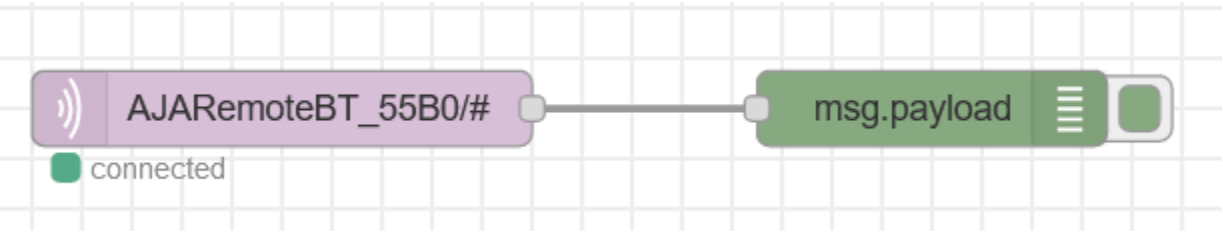
Operation

The document assumes that Mosquitto and Node-Red are being used for automation. Please adapt if other software stacks are being used. It is also assumed that Mosquitto and Node-Red have already been installed and configured appropriately and the operator has sufficient knowledge and understanding to create Node-Red sequences.

Learning Codes

In order to learn codes the existing remote controls can be used. Simply point them at the infrared receiver of the AJaRemote and press each button as needed.

In ***Node-Red***, create a sequence with an ***MQTT In*** object and a ***Debug*** object as shown here:



And set the Topic on the ***MQTT In*** object to be the address of the AJARemoteBT_xxxx followed by "/#". Replace "xxxx" with the address of the current AjaRemote. Deploy this flow and click on the debug tab.

Aim the infrared remote at the infrared receiver on the AjaRemote and press a button. The debug window in Node-Red should display the IR Code as well as the Pronto Code. These can

then be saved for later use when sequences are created to control the infrared devices. Repeat this process for all the buttons that are required. It's very useful to create a spreadsheet for all of the codes for future reference.

Sample captured IRCode:

{"protocol":"SONY","data":"0x42B47","bits":"20"}

And the associated Pronto codes:

{"protocol":"PRONTO","data":"0000 006D 0015 0000 005E 0015 0019 0015 002E 0015 0019 0015 0019 0015 0019 0015 0019 0015 0030 0015 0019 0013 0030 0015 0019 0015 0030 0015 0030 0015 0019 0015 002E 0015 0019 0015 0019 0015 0019 0015 0030 0015 0030 0013 0030 06C3 ","bits":"0"}

These codes can be directly used in NodeRed as JSON values.

Sending Codes

The AJARemote will subscribe to 3 topics with the MQTT Broker.

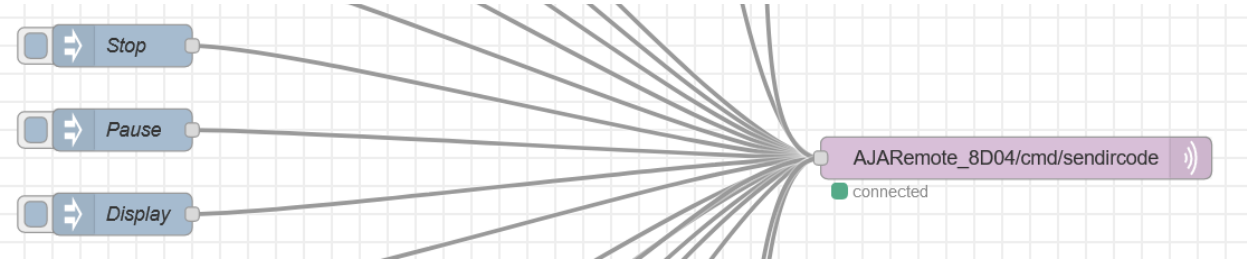
1. cmd/sendircode
2. cmd/sendbtcode
3. cmd/reboot

The first two commands will send a JSON string to the AJARemote. The "reboot" command will cause the AJARemote to reboot immediately.

Sending Infrared Codes

In most cases, devices can use the "non Pronto" commands to receive commands, but in some cases, the Pronto code needs to be used. It does not matter which one is used as long as it works. Some experimenting may need to take place. If nothing works, try the Pronto codes.

A basic Node-Red sequence may look similar to this:



1. Create an ***MQTT Out*** object and set the topic to ***AJARemoteBT_xxxx/cmd/sendircode*** (use the address of the AJARemote).

2. Create some sort of input to the ***MQTT Out*** object. In this example, a ***Trigger*** object is used.
3. Edit the properties of the Trigger object and add the appropriate JSON code to the "***msg.payload***". For example ***{"protocol":"NEC","data":"0x4B40718E","bits":"32"}***

In this example the ***NEC*** protocol is used. The actual code to be sent is ***0x4B40718E*** and the number of bits is ***32***. This exact string will be given by following the steps in the "***Learning Codes***" section.

Repeat this process for as many codes you wish to send.

The msg.payload for a Pronto command will look similar to:

{"protocol":"PRONTO","data":"0000 006D 0022 0000 015E 00AB 0017 0041 0015 0041 0015 0017 0015 0015 0017 0015 0015 0015 0017 0015 0017 003F 0017 0041 0015 0017 0015 0041 0017 0015 0015 0041 0015 0017 0017 003F 0015 0017 0015 0015 0017 0015 0015 0017 0015 0015 0017 0041 0015 0015 0017 0015 0017 003F 0017 0041 0015 0041 0015 0041 0017 003F 0017 0015 0017 003F 0017 0041 0015 0017 0015 258F","bits":"0"}

The value in "Bits" is actually ignored, but just needs to be present.

(This turns an Epson Projector on)

Sending Bluetooth commands

In order to use Bluetooth commands, the AjaRemote must be paired with the target device. Initially this functionality was added to wake up a Fire Stick by sending a "Home" button press; however, it can be paired with any device that recognizes an HID (Human Interface Device) keyboard / mouse. Start the pairing function on the target device and look for an HID called "BlueToothKB"

Sending Bluetooth commands is very similar to sending infrared commands by using the ***cmd/sendbtcode*** with the appropriate JSON msg.payload. There are two types of command "Keys", which refer to standard keys such as Up, Down, Return, Tab, and media specific commands such as "Next Track and "Previous Track"

The respective JSON msg.payloads are as follows, refer to the appendix for the full list of Keys and Media options.

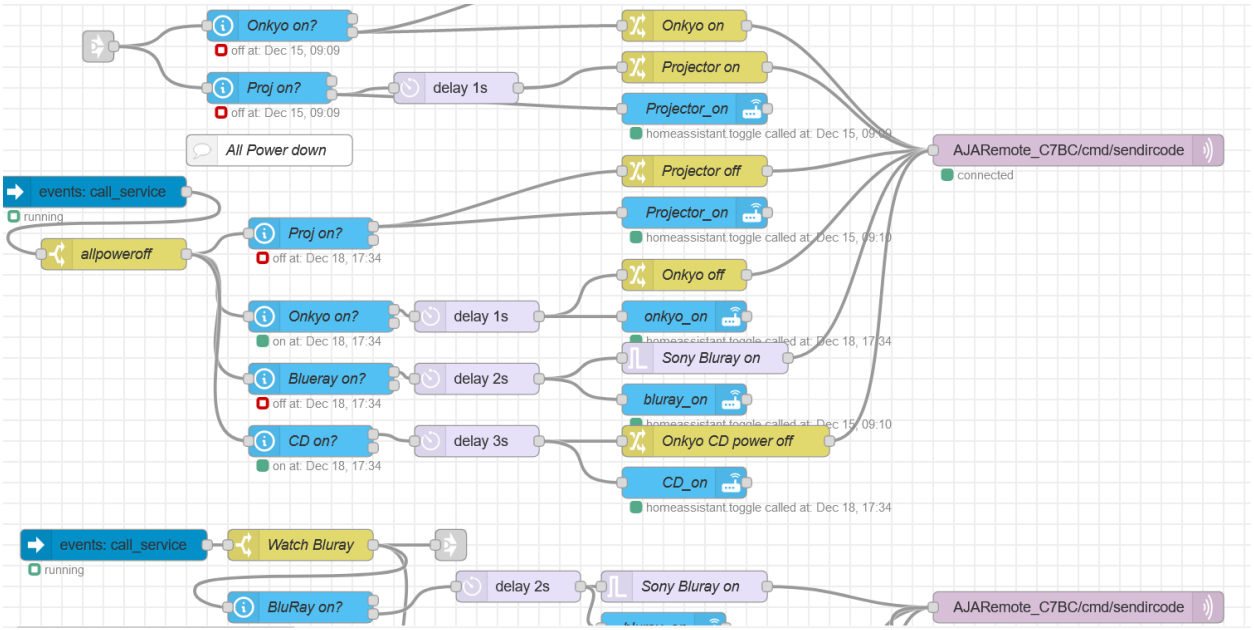
{"key":"KEY_HOME"}

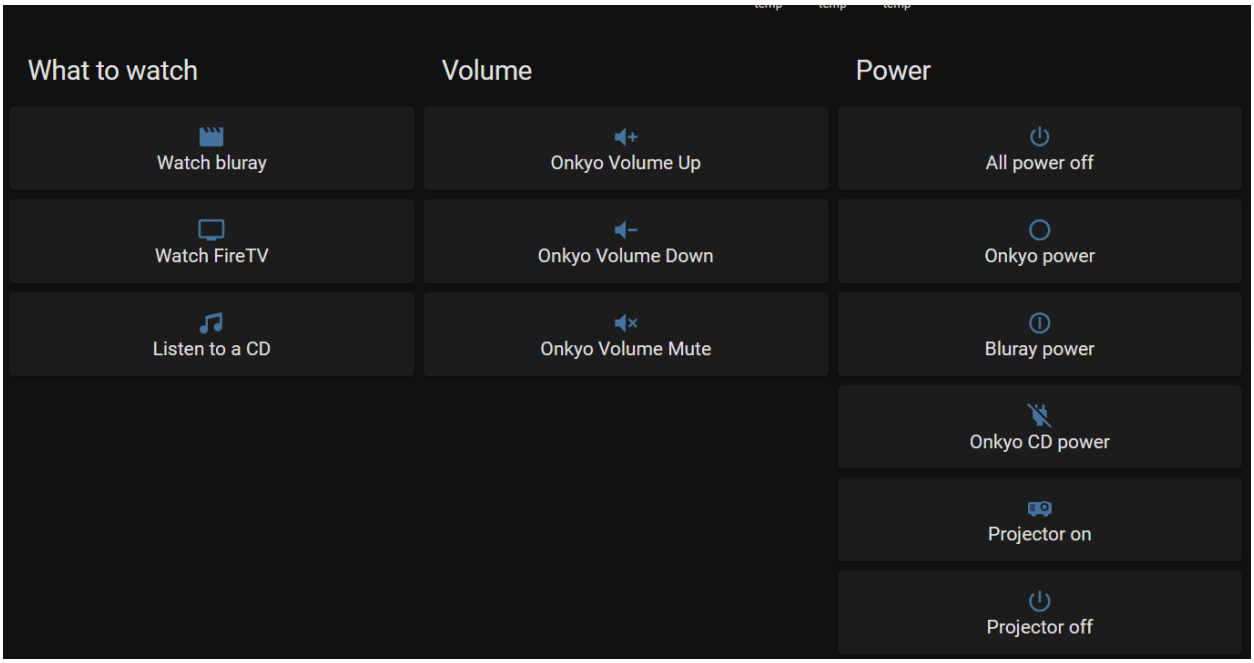
or

{"media":" KEY_MEDIA_NEXT_TRACK"}

Example

Node-Red sequences can be quite complex. The following screen shot shows a small portion of the author's configuration that is used in conjunction with Home Assistant.





By pressing the "***Watch Bluray***" button the Node-Red sequence will be initiated that will:

1. Check to see if the AV Receiver is turned on, turn it on if not.
2. Check to see if the Projector is turned on, turn it on if not.

3. Check to see if the Blueray player is on, turn it on if not.
4. Set the AV Receiver input to Blueray.

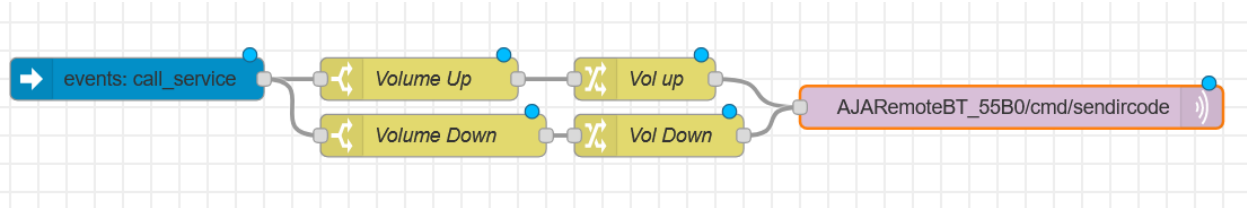
As each device is turned on, a Boolean flag is set to indicate its state. This way a device cannot be turned on twice, which in most cases will actually turn it off. This way the system can switch between "***Bluray***" and "***FireTV***" without turning the AV Receiver or Projector on and off.

An "***All Power off***" button will go through the list of devices and turn off anything that had previously been turned on.

AjaRemote can also be used to control lights, curtains, blinds, AV equipment, and so on, as long as it is controlled with infrared.

Home Assistant Integration

There are many ways to control an MQTT device from Home Assistant to achieve the sequence described above. The following is just one example of controlling volume.

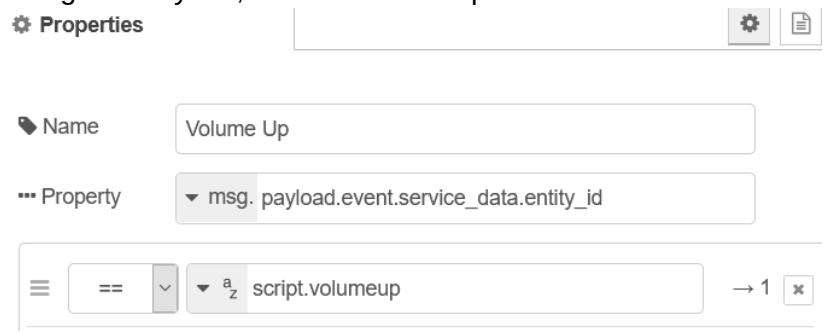


1. Edit Configuration.yaml and add an empty script, one for ***volumeup*** and one for ***volumedown***

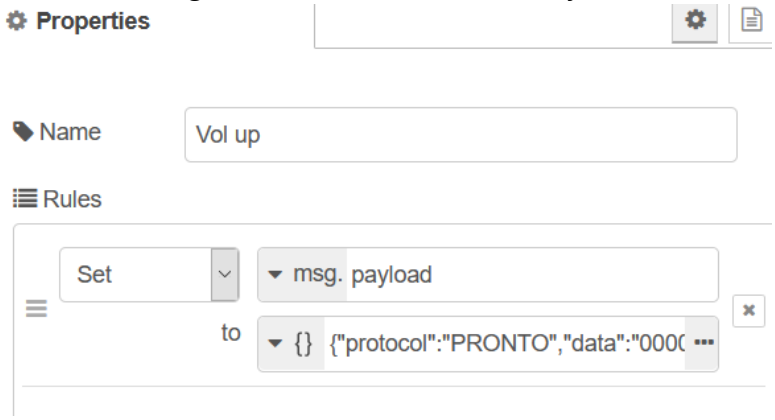
```
script:
  volumeup:
    sequence:
  volumedown:
    sequence:
```

2. Check and save the file.
3. Restart Node-Red to refresh all the entities.
4. From the "Home Assistant" pallet, drag an "***Events: all***" object to the flow and set the properties as follows:

5. Use two switch nodes and configure them using the script names from configuration.yaml, one for volume up and one for volume down.

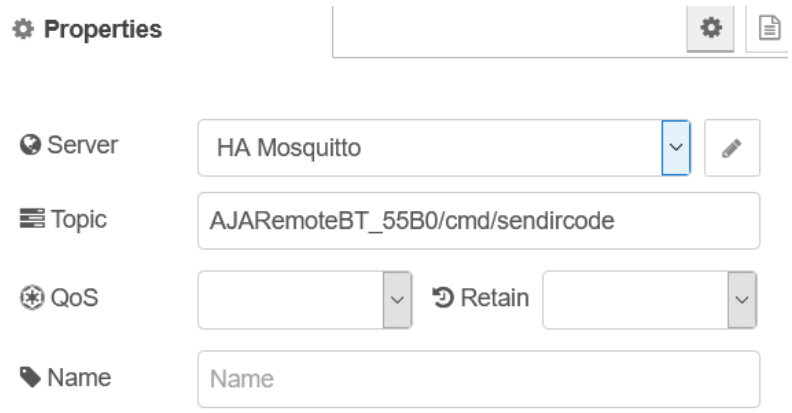


6. Use two ***Change*** nodes to set the necessary IR Codes to send:



{"protocol":"PRONTO","data":"0000 006D 0022 0002 0156 00AB 0015 0015 0015 003F 0015 0015 0015 0015 0015 003F 0015 0015 0015 003F 0015 003F 0015 003F 0015 0015 0015 003F 0015 003F 0015 0015 0015 003F 0015 003F 0015 0015 0015 0015 0015 003F 0015 0015 0015 0015 0015 0015 0015 0015 0015 0015 0015 0015 003F 0015 0015 0015 003F 0015 003F 0015 003F 0015 003F 0015 003F 0015 003F 0015 05CC 0156 0055 0015 0E47","bits":"32"} in this case.

7. Connect the two ***Change*** nodes to an ***MQTT Out*** node and set as follows using the address of the AjaRemote being used; Set as follows:



8. Deploy the sequence.
9. Create a new dashboard in Home Assistant and create a ***Vertical Stack*** and configure as follows:

Vertical Stack Card Configuration

```yaml
type: vertical-stack
title: Volume
cards:
  - type: entity-button
    tap_action:
      action: call-service
      service: script.turn_on
      service_data:
        entity_id: script.volumeup
    entity: script.volumeup
    name: Onkyo Volume Up
    icon: 'mdi:volume-plus'
    icon_height: 25px
  - type: entity-button
    tap_action:
      action: call-service
      service: script.turn_on
      service_data:
        entity_id: script.volumedown
    entity: script.volumedown
    name: Onkyo Volume Down
    icon: 'mdi:volume-minus'
    icon_height: 25px
```

10. Save the dashboard and test.

As mentioned this is one way of many to configure the automation flow under Home Assistant. Adapt as needed.

Technical

Expansion

There is a 3.5mm mono jack located on the rear panel of the AJARemote case for adding additional IR Leds.

The jack is center pin positive and can easily drive 3 additional LED's such as these

"IR Infrared Emitter Extender Cable Extension (10 Feet) Triple Head3 Eye 3.5mm Jack Infrared Red Transmitter Blaster Blink Eye Wire Cord Compatible with IR Repeater Extender System Kit, Xbox One"

Resetting

In order to reset the AJaRemote to factory defaults hold down the ***D0*** button for approximately 10 seconds until the LED turns a solid red. Wait for the LED to go out and then follow the "***Initial startup***" procedures.

Programming

Although the AjaRemote can be used as is, it is also open for modification and expanded functionality. With the appropriate knowledge of C++ and the ESP32 in conjunction with the necessary hardware to program the AjaRemote, extended functionality can be added. GPIO 25 and 26 are available on a two-pin header.

A standard ISP header is available and uses 5V logic. Please do not connect 5V power and 5v from the programmer at the same time. Disconnect the power before plugging in the programmer.

Holding the ***D0*** button down during power up will put the ESP32 into "download mode". Cycle the power after programming.

OTA cannot be used due to the available memory size. OTA can be used if the Bluetooth functionality is removed. If using the Arduino IDE, remember to set the partition size accordingly.

Official software and Eagle schematics are available here on GITHub.

Appendix

Bluetooth Keyboard Codes

KEY_RETURN

KEY_UP_ARROW

KEY_DOWN_ARROW

KEY_LEFT_ARROW

KEY_RIGHT_ARROW

KEY_BACKSPACE

KEY_TAB

KEY_ESC

KEY_INSERT

KEY_DELETE

KEY_PAGE_UP

KEY_PAGE_DOWN

KEY_HOME

KEY_END

Bluetooth Media Codes

KEY_MEDIA_NEXT_TRACK

KEY_MEDIA_PREVIOUS_TRACK

KEY_MEDIA_STOP

KEY_MEDIA_PLAY_PAUSE

KEY_MEDIA_MUTE

KEY_MEDIA_VOLUME_UP

KEY_MEDIA_VOLUME_DOWN

KEY_MEDIA_WWW_HOME

KEY_MEDIA_LOCAL_MACHINE_BROWSER

KEY_MEDIA_CALCULATOR

KEY_MEDIA_WWW_BOOKMARKS

KEY_MEDIA_WWW_SEARCH

KEY_MEDIA_WWW_STOP

KEY_MEDIA_WWW_BACK

KEY_MEDIA_CONSUMER_CONTROL_CONFIGURATION

KEY_MEDIA_EMAIL_READER

More codes can be added over time as need dictates.