AjaRemote

Version 2022-3

Contents

Introduction

AJARemote is part of the AJAOne family of devices that use the [ESP32](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 most anything controllable via Infrared.

Although the InfraRed library is very extensive there is no guarantee that every device can be controlled.

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

License Model

MIT License

Copyright (c) 2022 Nick Tucker

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and Hardware and associated documentation files (the "Software" and "Hardware"), to deal in the Software and Hardware without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software and Hardware, and to permit persons to whom the Software and Hardware is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software and Hardware.

THE SOFTWARE AND HARDWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER

LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE AND HARDWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE AND HARDWARE.

Protocols

AJARemote supports most of the major IR Protocols, such as, but not limited to:

Infrared

- Denon
- Dish
- JVC
- Lego
- NEC
- Panasonic
- [Pronto](#)
- Sony
- Samsung
- Sharp

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 SSID (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. Any standard USB charging block can be used.
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 **AJARemoteAP**. The password is **password**.
4. Depending on your device, the configuration page may show automatically. If not, open a browser to 192.168.4.1.

AJAOneAP

WiFiManager

Configure WiFi

Configure WiFi (No Scan)

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

Test Access point 1 🔒 88%

Test Access point 2 🔒 58%

Test Access point 3 58%

Test access point 4 🔒 18%

SSID

password

mqtt server

1883

mqtt user

mqtt password

AJAOne_B3DC

save

Scan

AjaRemote Version 2022-3, July 2022
Nick Tucker

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 twice 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 blink 4 times and repeat 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 **AJARemote_XXXX** where XXXX is the last four digits of the device MAC address. For example, **AJARemote_AD34**.

LED Codes

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

| 1 blink	• Indicates data has either been sent or received from the MQTT Broker • Indicates a web page has been sent.
2 blinks and repeats.	Booting. Connecting to WiFi and MQTT. Can also indicate the AJARemote needs to be setup with WiFi credentials via the **AJARemoteAP** access point.
3 blinks	Boot process complete and entering normal operations.
4 blinks and repeats	Attempting to connect to the MQTT Broker.
5 blinks and repeats	Setting new WiFi SSID and / or password.

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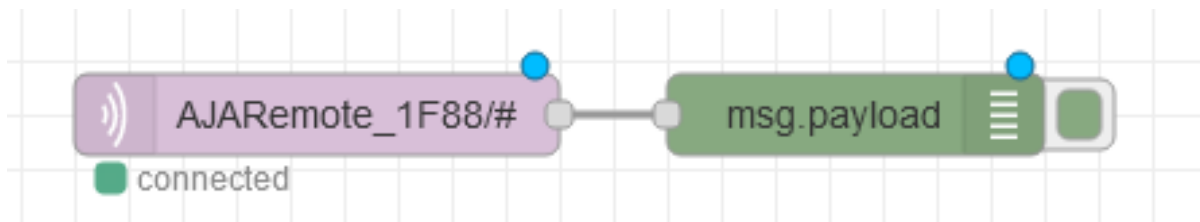
And set the Topic on the **MQTT In** object to be the address of the AJARemote_xxxx followed by "/#". Replace "xxxx" with the address of the current AJARemote. Deploy this flow and click on the debug tab.

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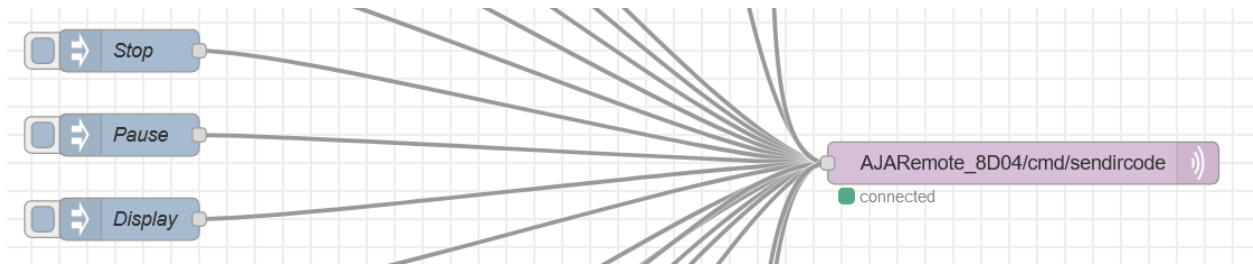
1. cmd/sendircode
2. cmd/sendbtcode
3. cmd/reboot

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 **AJARemote_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)

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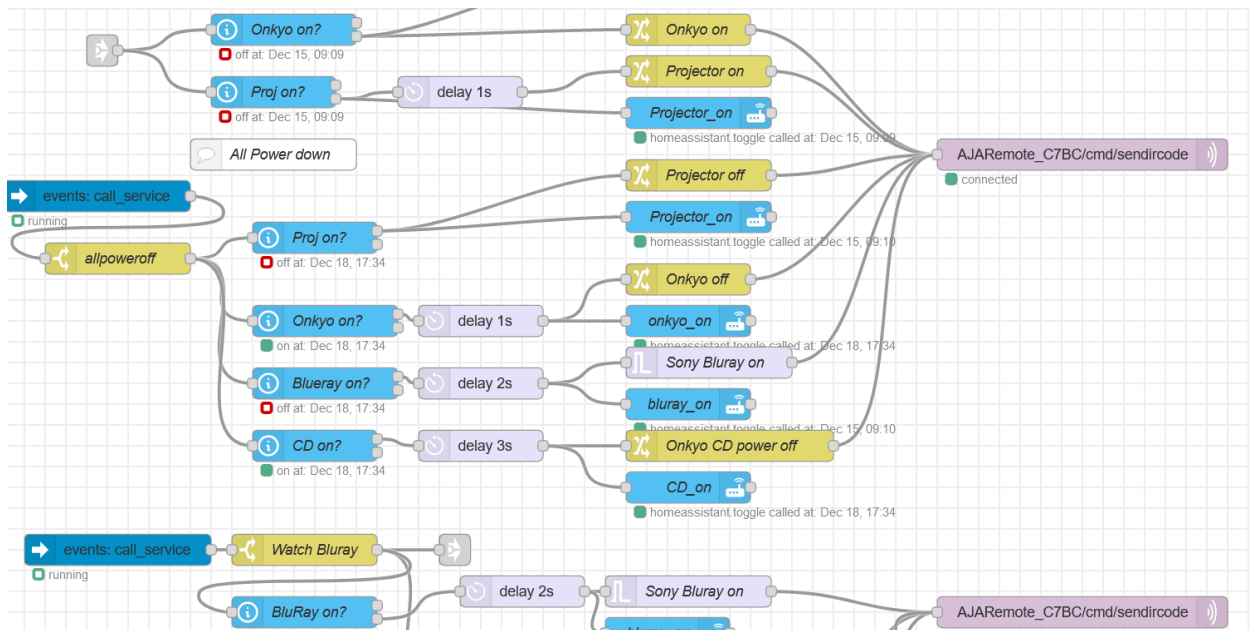
By pressing the "***Watch Bluray***" button the Node-Red sequence will be initiated that will:

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.

AjaRemote Version 2022-3, July 2022
Nick Tucker

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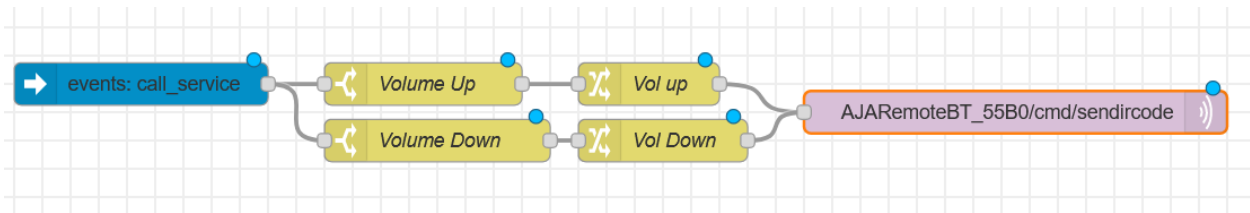
1. Edit Configuration.yaml and add an empty script, one for **volumeup** and one for **volumedown**

```
14 ▾ script:¬
15 ▾ ··volumeup:¬
16 ····sequence:¬
17 ▾ ··volumedown:¬
18 ····sequence:¬
19   ¬
```

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:

Properties ⚙ 📄

Name   Vol up

≣ Rules

Set ⌄   ▾ msg. payload

to   ▾ {} {"protocol":"PRONTO","data":"0000 ⋯

{"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 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:

Properties ⚙ 📄

🌐 Server   HA Mosquitto   ✏

≣ Topic   AJARemoteBT_55B0/cmd/sendircode

✴ QoS   ⌄   🔁 Retain   ⌄

🏷 Name   Name

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

```
 1  type: vertical-stack
 2  title: Volume
 3  cards:
 4    - type: entity-button
 5      tap_action:
 6        action: call-service
 7        service: script.turn_on
 8        service_data:
 9          entity_id: script.volumeup
10      entity: script.volumeup
11      name: Onkyo Volume Up
12      icon: 'mdi:volume-plus'
13      icon_height: 25px
14    - type: entity-button
15      tap_action:
16        action: call-service
17        service: script.turn_on
18        service_data:
19          entity_id: script.volumedown
20      entity: script.volumedown
21      name: Onkyo Volume Down
22      icon: 'mdi:volume-minus'
23      icon_height: 25px
```
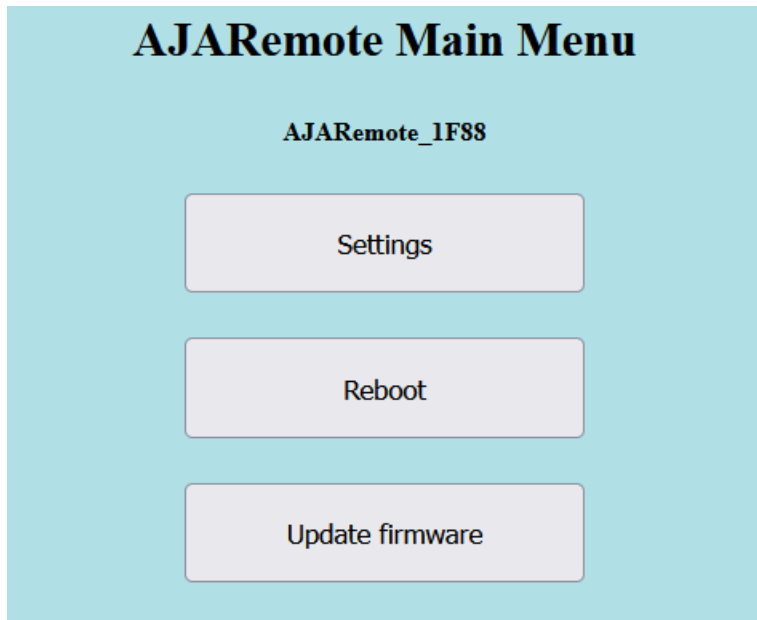
10. Save the dashboard and test.

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

Web Page

The AJARemote runs an internal web server, on port 80, to access some further functionality. Open a web browser and enter the IP address of the AJARemote, for example, **_HTTP://192.168.10.10_**
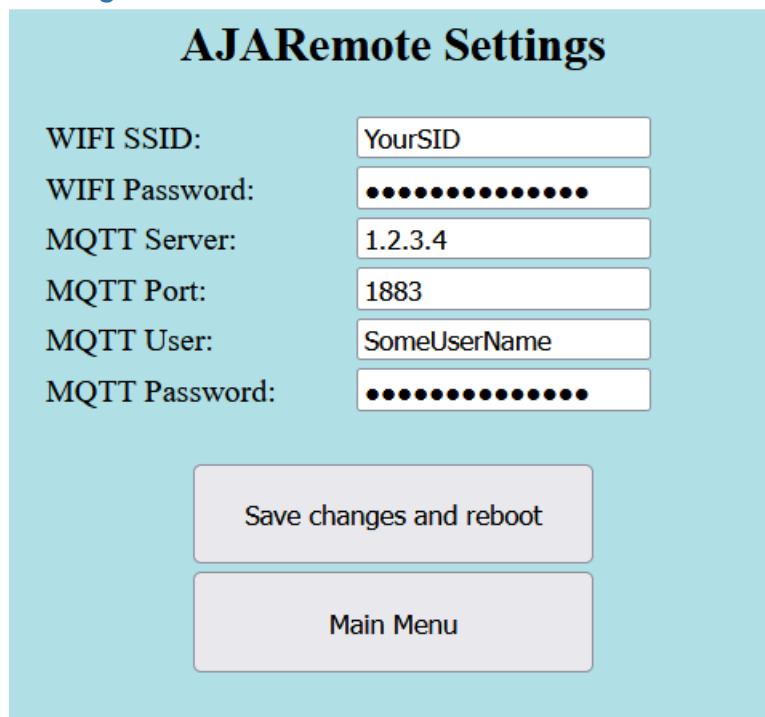
Main Menu

Figure 1 - Main Menu

- Settings can be updated such as WiFi credentials as well as the MQTT credentials.
- The AJARemote can be rebooted
- The AJARemote Firmware can be updated.
- The AJARemote address is shown.

Settings

Figure 2 – Settings

From the **Settings** page the items set during the initial setup can be changed. Make edits as needed and click on the **Save changes and reboot** button.

If the WiFi settings are changed incorrectly (wrong password or wrong SSID) the AJARemote will open up the access portal AJARemoteAP. Connect to this WiFi portal and correct the settings and try again.

Reboot
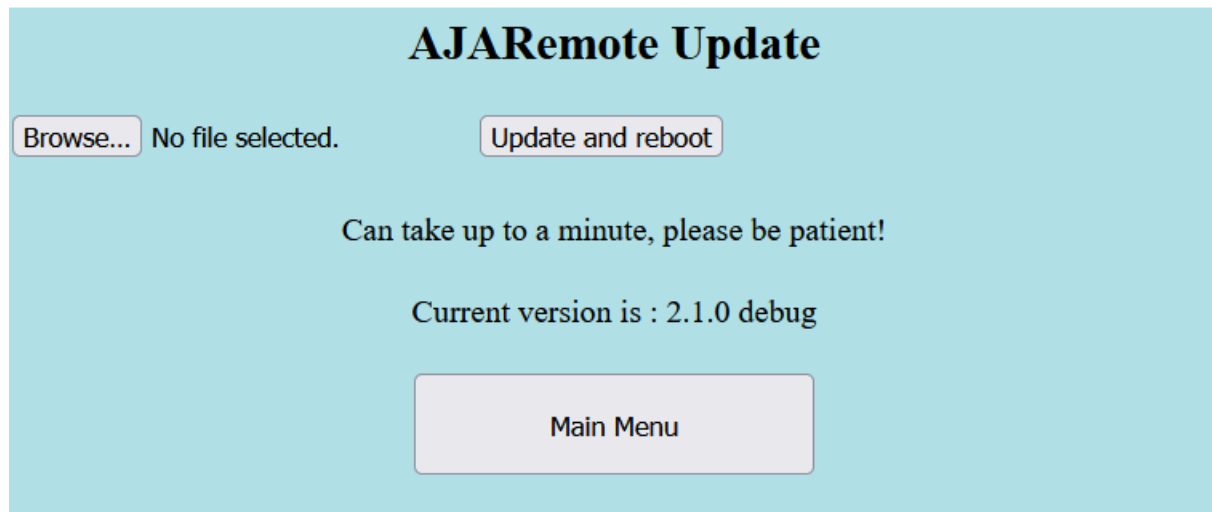As the name implies this will reboot the AJARemote immediately.

Update Firmware

AJARemote Update

Browse... No file selected.   Update and reboot

Can take up to a minute, please be patient!

Current version is : 2.1.0 debug

Main Menu

Figure 3 - Update firmware

New to AJARemote is the ability for field updatable firmware. In the event updates are made, they can be obtained from the AJARemote GIT page.

1. Download the new firmware.
2. Click on the **Browse** button and select the new file
3. Click on **Update and reboot**. DO NOT CLICK ANY BUTTONS UNTIL THE PROCESS IS COMPLETE.
4. Once complete the AJARemote will reboot and the main menu will be displayed.

This page also shows the current firmware version.

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 on solid. 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.

Official software and Eagle schematics are available here on GITHub.