RRoble Blind Motor Remote

Centralized remote control for 433MHz rolling blind motors

Revision History		
Version date	**Author**	**Change Summary**
Tuesday, June 21, 2023	Jesus Amozurrutia	First Draft

Contents

1. Introduction

This project consists of a centralized remote-control (the Device) to manage and control roller blinds and shades compatible with 433 band RF remotes like the BF-305:

The goal is to have one Device to register and control all the blinds in an area or room of the house and control each motor through ***Home Assistant*** or another MQTT compatible hub using **Cover** entities.

The main features for the Device are:

- Allows to register up to 50 motors/blinds.
- Allows to register up to 20 remote-control codes.
- Control blinds via ***Home Assistant*** using MQTT using Cover entities.
- Support for partially open/close positions.
- Each motor can be added to scenes and automations.
- The communication between ***Home Assistant*** and the remote control is via WiFi, so there is flexibility in the location of the Devices.
- Easy to setup.

It is possible to do this same functionality with a universal RF remote control like Broadlink, but I had little success in some tests that I did with this type of devices and with 433Mhz RF signals and in the end it was more expensive and complicated to configure so I decided to try the DIY route.

2. Motors

In particular, I am using blind motors from RollerHouse, but there are other brands that use the same band and a similar RF protocol:

https://www.amazon.com/stores/Rollerhouses/page/018EE1FB-1C43-46A7-A7C9-61E1A8AE468D?ref_=ast_bln https://www.amazon.com/review/R2ZXHHNMNFB5C7/ref=cm_cr_srp_d_rdp_perm?ie=UTF8

In my experience other brands use a very similar RF protocol but sometimes the checksum digit changes, so the sketch may need to be modified accordingly (See `RF_CS_ADJ` in the program/sketch).

The configuration of these motors must be carried out according to the manufacturer's instructions, pairing each motor with the provided remote-control on a specific channel, as well as establishing the stop points for the open and closed positions.

These types of motors are relatively cheap, but they do not have a feedback mechanism, so the remote control and Home Assistant can only guess the actual position of the blind, in case a partial open/close position is set. In order to establish the position of the blind, the opening and closing time must be taken after setup (depending on the weight of the blind, opening and closing times can be very different) and the remote control will fully open or close using the set stop points and then the final position is estimated based on time.

BF-305 and similar remote controls can operate in continuous or step by step mode. Step by step mode is useful to establish a precise position of the blind.

The Device uses the step by step mode to go from closed to open in a certain number of steps on zebra or day/night blinds:

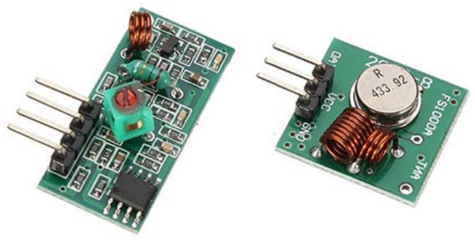
Zebra blinds.

3. Device Hardware

The hardware for the remote is based on an ESP-8266 module like the **D1 Mini** or the **Node MCU**, or even an ESP-12E/F, but this last one requires additional breakout hardware.

https://www.wemos.cc/en/latest/d1/d1_mini.html

The second set of elements are the 433MHz ASK transmitter and receiver modules such as the MX-05V and MX-FS-03V.



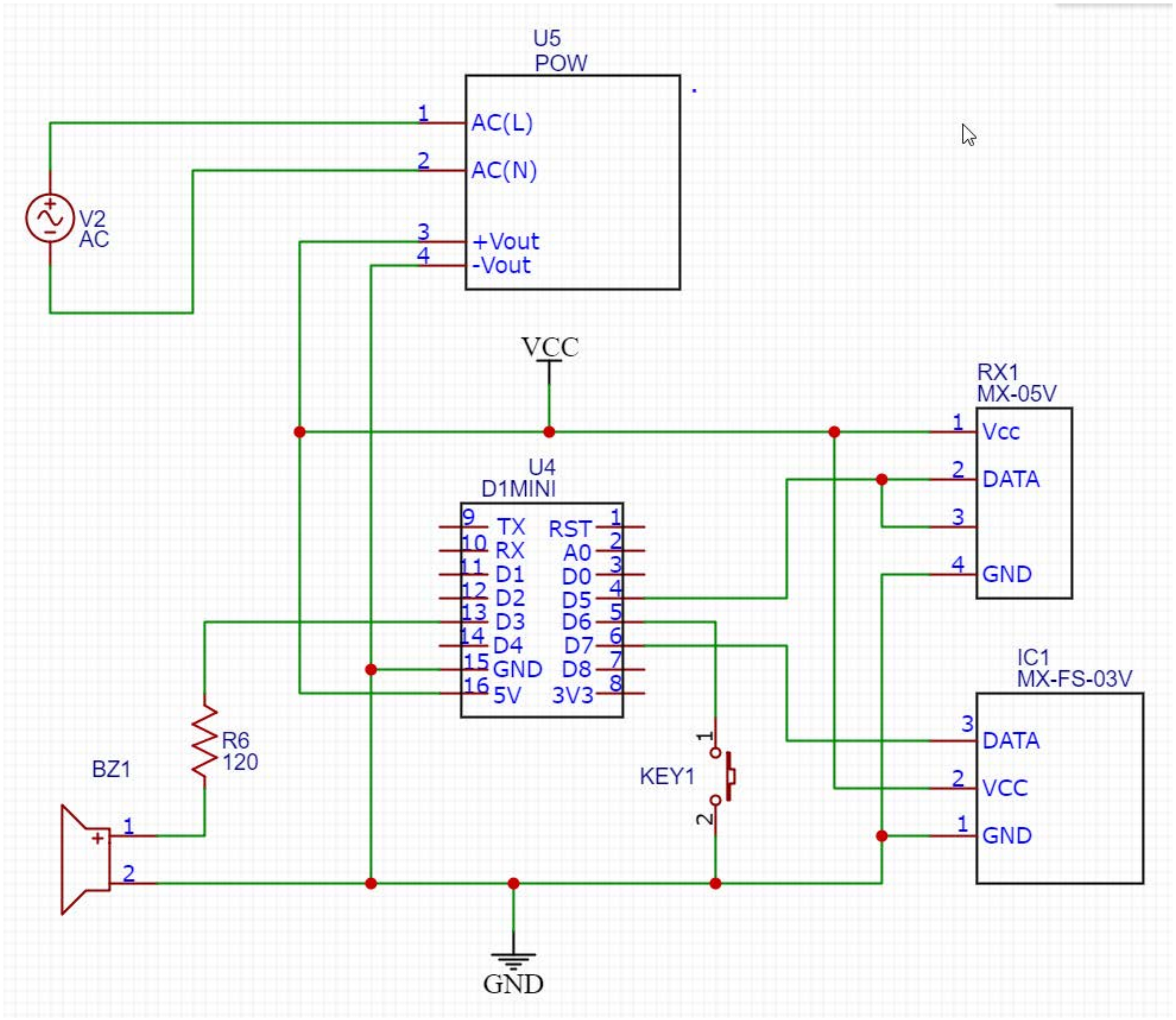
https://hobbycomponents.com/wired-wireless/1054-433mhz-transmitter-receiver-modules-with-antenna

Optionally you can set a button and a piezoelectric buzzer to receive feedback during configuration.

An inexpensive USB converter with at least 500mA output can be used as a power source.

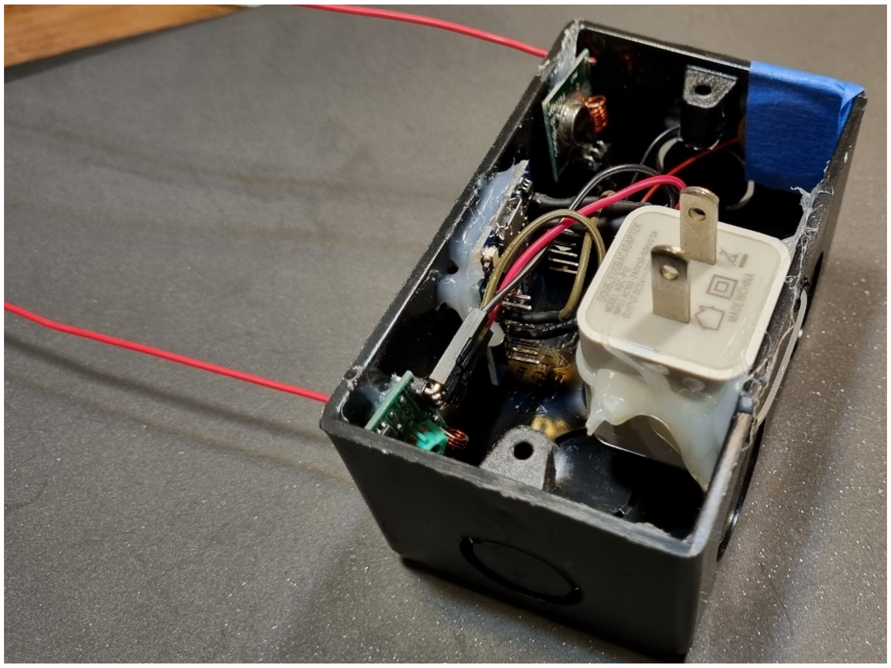
Below is the wiring diagram for the **D1 Mini**, but it can be easily replaced with a NodeMCU or even an ESP-32 module. Wiring is simple enough that it can be done without soldering with Dupont wires.

3.1. Schematic diagram



With this hardware, motors can be controlled up to 10 m (33 ft) away from the remote control, depending on the antenna of the RF modules, the type of construction and other factors of the environment.

In my case I set everything up in a generic box since it will be connected to an outlet hidden behind a piece of furniture.



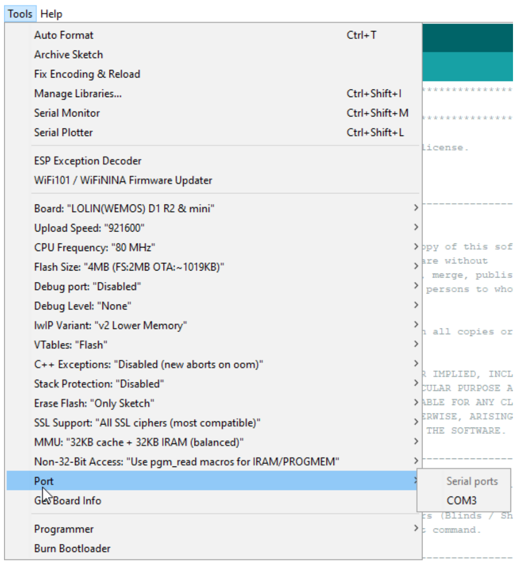
Full assembly.

4. Flashing Software

To upload the software for the first time, it is necessary to have a computer with the Arduino IDE installed and a USB cable for the **D1 Mini** or Node MCU modules. In this case the cable will provide power for the device, so it is not necessary to connect the power supply. In fact, it is highly recommended not to connect the power supply during this stage due to the risk of electric shock or damage to the computer.

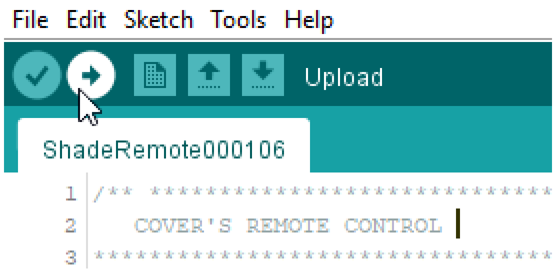
Open Arduino IDE and open the "BlindRemoteEsp_Current.ino" sketch. Check the "Sketch configuration" section near the top of the file to customize the sketch to you needs.

Under the Tools menu, select the Board to match your ESP module (DI Mini, Node MCU, etc.). If the board model matches the selection, the default parameters are usually adequate. Select the USB port where the cable has been connected:



Board: "LOLIN(WEMOS) D1 R2 & mini"
Upload Speed: "921600"
CPU Frequency: "80 MHz"
Flash Size: "4MB (FS:2MB OTA:~1019KB)"
Debug port: "Disabled"
Debug Level: "None"
lwIP Variant: "v2 Lower Memory"
VTables: "Flash"
C++ Exceptions: "Disabled (new aborts on oom)"
Stack Protection: "Disabled"
Erase Flash: "Only Sketch"
SSL Support: "All SSL ciphers (most compatible)"
MMU: "32KB cache + 32KB IRAM (balanced)"
Non-32-Bit Access: "Use pgm_read macros for IRAM/PROGMEM"

Click the upload button on the Arduino IDE to start flashing.



```
/** ******************************
    COVER'S REMOTE CONTROL
*********************************
```

After a while the device is ready to operate. If the DEBUG_MODE is enabled in the “Sketch configuration” section, you can open the serial monitor to check the messages sent by the Device.

5. WiFi Setup

Once the device is flashed, you can configure the parameters of the WiFi network and Home Assistant with the Setup Portal.

To activate the Setup Portal:

- Do a double reset (within 1.5 seconds) using the reset button in the **DI Mini** or **Node MCU**;
- If the button is installed on pin 6 you can activate the portal by doing 6 to 8 clicks within 4 seconds.

The on-board LED starts flashing every second. On a computer or mobile device, search for a WiFi network with a name like “RXRemote_XXXXXXXXXXXX” and select it.

Open a browser. It should automatically direct you to the Setup Portal. In case the portal does not open, you can go to the address “http://192.168.4.1/“ to open it manually.

Follow the on-screen instructions to set the WiFi configuration and MQTT parameters.

For more information on the Setup Portal see: https://github.com/tzapu/WiFiManager

If the WiFi and MQTT data are correct, the device should be auto-discovered in Home Assistant or other hubs that support MQTT auto-discovery.

The device will show up as a ***switch*** in Home Assistant.

6. Setup blinds

The general description of the process to make the configuration is:

- Install motors according to manufacturer's instructions.
- Pair the motor with the remote provided by the manufacturer and set the upper and lower limits for the blind.
- Take note of the time it takes to go from open to close and vice versa as accurately as possible.
- Register the manufacturer’s remote into the device.
- Add the motor configuration to the device.
- Test.

6.1. Register-Remote Codes

To register a new remote-control code:

a) Set the device into listening mode:
   - Press the pin 6 button on the device or
   - Turn on the switch auto-discovered in Home Assistant.

b) The device’s LED will start flashing and the buzzer will beep every second.

c) Press the up pr down buttons on the remote paired to the blind.

d) Once the device detects a valid code the device acknowledges with 3 short beeps and flashes. The new code is kept in memory for 90 seconds.

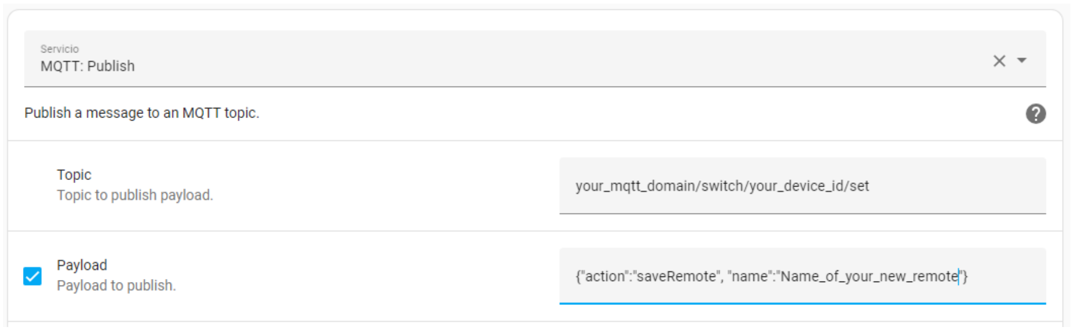
e) To permanently register the code, we need to send a MQTT message to the device. In Home Assistant go to "Developer Tools" then select “Services” and select the “MQTT: Publish” service in the dropdown.

f) Set the Topic using the parameters set during the device WiFi configuration:
   `YOUR_MQTT_DOMAIN/switch/YOUR_DEVICE_ID/set`

g) Set the Payload to something like:
   `{"action":"saveRemote", "name":"REMOTE_NAME"}`

h) Call the service.

i) The device acknowledges with 3 short beeps and flashes



Publish a message to an MQTT topic.

Topic
Topic to publish payload.
your_mqtt_domain/switch/your_device_id/set

Payload
Payload to publish.
{"action":"saveRemote", "name":"Name_of_your_new_remote"}

6.2. Add a Motor

Adding a new blind is also done via MQTT messages. It is necessary to register the Remote-Control Code first.

In Home Assistant, go to "Developer Tools" then select “Services” and select the “MQTT: Publish” service in the dropdown.

Use the same topic used in registering Remote-Control codes:

```
YOUR_MQTT_DOMAIN/switch/YOUR_DEVICE_ID/set
```

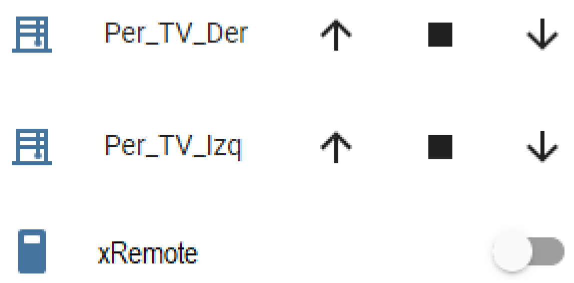
Set the Payload to something like:

```
{"action":"saveCover","name":"MOTOR_NAME","remote":"REMOTE_NAME","channel":1,"oTime":12,"cTime":1
2,"feedback":2,"mpos":8}
```

Where:

- **"name"** is the unique name for the blind in Home Assistant.
- **"remote"** is the name assigned to the remote code. See “Register Remote Codes”.
- **"channel"** Channel number assigned in the remote-control during pairing. For single channel remotes this is usually 1.
- **"oTime"** is the time in seconds it takes the blind to go from fully closed to fully open.
- **"cTime"** is the time in seconds it takes the blind to go from fully open to fully closed.
- **"feedback"** is the feedback mechanism for the motor.
  - 0 = Deletes the entity
  - 1 = None
  - 2 = Set position by time
  - 10 = Keep the same
- **"mpos"** is the number of steps it takes to go from a closed position to open in zebra or day/night blinds

Once the motor is added the auto-discovery mechanism is triggered and the new Cover is displayed in Home Assistant.


Per_TV_Der
Per_TV_Izq
xRemote

7. Appendix 1. Sketch

7.1. OTA Update

This sketch supports OTA updates for development purposes, using the Arduino IDE or an HTTP server.

Currently, HTTP updates only support unsecured http, therefore it is only recommended in a local environment for development purposes.

It is possible to implement secure https, as in the next example, if a public deployment is required: https://gist.github.com/igrr/24dd2138e9c8a7daa1b4, but at the moment it is out of scope.

> *Note: Temporarily disable the firewall in the computer while uploading a new image to the controller, to avoid disconnection problems when using OTA with the Arduino IDE.*

7.2. Libraries

The software uses the following libraries:

- ESP8266 core for Arduino (https://github.com/esp8266/Arduino).
- ESP8266 WiFi Connection manager (https://github.com/tzapu/WiFiManager).
- Double Reset Detect (https://github.com/jenscski/DoubleResetDetect).
- Arduino Json (https://github.com/bblanchon/ArduinoJson/)
- Asynchronous MQTT Client (https://github.com/marvinroger/async-mqtt-client/releases/tag/v0.8.1).

8. Appendix 2. RF Protocol

The RF protocol works using the 433.92MHz band with ASK modulating.

Each time a button is pressed on the remote-control, it sends a command to the motor.

Each command has 3 segments:

- 4 AGC sync. bits
- 41 command tribits
- Radio silence

The signal is divided in pulses. Each pulse width is 23µs.

The AGC sync. bits are:

- HIGH 216 pulses
- LOW 108 pulses
- HIGH 75 pulses
- LOW 15 pulses

The command bits are:

- Data 0 = HIGH (15 pulses), LOW (30 pulses)
- Data 0 = HIGH (30 pulses), LOW (15 pulses)

The 41 bits of the command are as follows:

- 16 bits for (unique) remote control ID, hard coded in each remote.
- 4 bits for channel ID: 1 - 15
- 4 bits for command:
  - DOWN = 1000
  - UP = 0011
  - STOP = 1010
  - CONFIRM/PAIR = 0010
  - LIMITS = 0100
  - CHANGE ROTATION DIRECTION = 0001
- 8 bits for remote control mode: step by step = 10000000, continuous mode = 10001000
- 8 bits for checksum This is the sum of the first 4 bytes, inverted (xored) + 2.
- Other remote brands may use a different constant for the checksum.
- 1 bit = 1

41 bits in total

Radio silence of 223 pulses