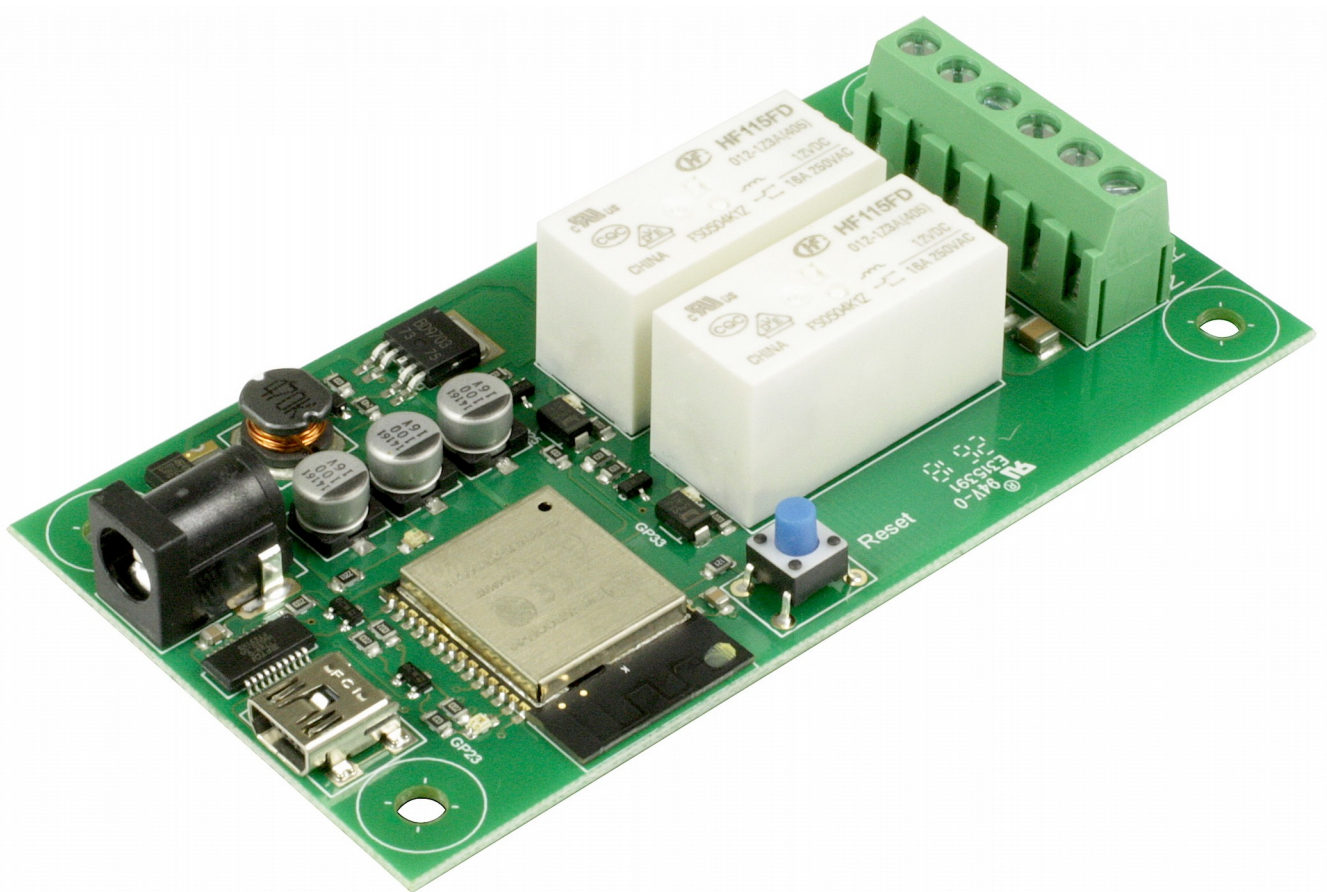


# ESP32LR20

User Manual Version 1.6



## Table of Contents

Changes from v1.5 to v1.6.....	3
Overview.....	3
Control Protocols.....	3
WiFi.....	3
Configuration.....	3
USB Configuration Commands.....	4
ST Status. Return the system status.....	4
RB ReBoot.....	5
IP Sets the modules IP address.....	5
SB Sets the SuBnet mask.....	5
GW Sets the GateWay address.....	5
PD Sets the Primary DNS.....	6
SD Sets the Secondary DNS.....	6
SS This sets the SSID.....	6
PW Sets your networks WIFI password.....	6
PA Sets the TCP/IP port number for the ASCII commands.....	6
AP Sets the ASCII password.....	7
MS Sets the MQTT broker address.....	7
MD Sets the MQTT ID for this module.....	7
MP Sets the MQTT broker's port.....	7
MU Sets the MQTT user name (V1.6+).....	7
MW Sets the MQTT password (V1.6+).....	8
R1-R2 Sets the MQTT topic this relay is subscribed to.....	8
TCP/IP Commands.....	9
SR Set Relay.....	9
GR Get Relay.....	9
Password.....	10
HTML Commands.....	10
Webpage.....	11
Schematics.....	12
CPU.....	12
Power Supply.....	13
Relay Outputs.....	14
PCB dimensions.....	15
Appendix 1.....	16
Programming the ESP32LR20 with Arduino studio.....	16
Notes.....	22

## Changes from v1.5 to v1.6

Setup commands MU and MW were added to provide an MQTT username and password.

## Overview

The ESP32LR20 is a WIFI connected relay module using the popular ESP32.

It provides 2 relays capable switching up to 16Amps. The normally open relay contacts have snubbers fitted across them, and can drive inductive loads such as contactors and solenoids.

Power for the board is 12v dc, which can be provided by a standard universal wall power supply. A 1A or greater supply should be selected.

## Control Protocols

1. Simple plain text commands sent to the module.
2. HTML commands
3. MQTT
4. A built in webpage

## WiFi

The ESP32LR20 connects via 2.4GHz WiFi to your network. Therefore it must be located in a position where it gets a good WiFi signal. The module should not be enclosed in a metal box/cabinet as this will shield the WiFi signal.

You can check the signal level by looking at the RSSI figure which is reported by the ST (STatus) command.

## Configuration

The ESP32LR20 is configured by connecting a USB cable to your PC and running a terminal program. PuTTY is a good option if you don't have any other preferences.

The serial port should be set to 115200 baud, 8 bit, 1 stop, no parity, no flow control.

## USB Configuration Commands

### ST Status. Return the system status

```
Status:  
Firmware Version: 1.6  
IP: 0.0.0.0 (192.168.0.19)  
Subnet: 0.0.0.0  
Gateway: 0.0.0.0  
Primary DNS: 0.0.0.0  
Secondary DNS: 0.0.0.0  
SSID: *****  
Password: *****  
ASCII TCP Port: 17123  
RSSI: -72  
MQTT Server: 192.168.0.115  
MQTT Port: 1883  
MQTT ID: ESP32LR20  
MQTT User: myUsername  
MQTT Password: *****  
Relay1 Topic: R1Topic  
Relay2 Topic: R2Topic
```

When the IP address is set to 0.0.0.0 this means that the IP address is being provided by your networks DHCP server. In that case the assigned IP address is also provided, as above.

When the SSID and Password are setup, they will be displayed until the next reset, After that they will only show as \*\*\*\*\*.

### RB ReBoot

This will restart the module. It may produce a lot of random characters as the ESP32's boot logging runs at a different baud rate. If it succeeds in connecting to your network it will report the IP address.

```
Re-Booting.. .  
嶮?????????#XL##C????5)????ia?b?????????命#??*?UY?o????? #i#U?5 ?Q?????
```

```
WiFi connected.  
IP address:  
192.168.0.6
```

## **IP Sets the modules IP address**

Enter IP followed by the required IP address. Entering address 0.0.0.0 means the IP will be obtained from your networks DHCP server. The new IP address will take effect after the next re-boot.

```
IP "192.168.0.123"  
OK. Saved IP Address: 192.168.0.123
```

## **SB Sets the SuBnet mask**

```
SB "255.255.255.0"  
OK. Saved Subnet Mask: 255.255.255.0
```

## **GW Sets the GateWay address**

This is normally the IP address of your router.

```
GW "192.168.0.1"  
OK. Saved Gateway Address: 192.168.0.1
```

## **PD Sets the Primary DNS**

The can be the IP address of your router which will then use your ISP provided DNS. You can also specify the DNS such as 8.8.8.8 for Googles DNS server.

```
PD "192.168.0.1"  
OK. Saved Primary DNS: 192.168.0.1
```

## **SD Sets the Secondary DNS**

The can be the IP address of your router which will then use your ISP provided DNS. You can also specify the DNS such as 8.8.4.4 for Googles DNS server.

```
SD "8.8.4.4"  
OK. Saved Secondary DNS: 8.8.4.4
```

## **SS This sets the SSID**

The SSID is the public name of your WIFI network Enter your WIFI's SSID here.

SS "Devantech"

OK. Saved SSID: Devantech

## **PW Sets your networks WIFI password**

PW "K]~kCZUV\*UGA6SG~"

OK. Saved Password: K]~kCZUV\*UGA6SG~

## **PA Sets the TCP/IP port number for the ASCII commands**

PA 17126

OK. Saved ASCII port number: 17126

## **AP Sets the ASCII password**

AP "MySecretPassword"

OK. Saved AsciiPassword: MySecretPassword

## **MS Sets the MQTT broker address**

MS "192.168.0.121"

OK. Saved MQTT Server: 192.168.0.121

## **MD Sets the MQTT ID for this module**

MS "UniqueModuleName"

OK. Saved MQTT ID: UniqueModuleName

## MP Sets the MQTT broker's port

Normally, this should be set to 1883.

```
mp 1883
```

```
OK. Saved MQTT port number: 1883
```

If you are not using MQTT, set the port to 0. This will turn off MQTT, otherwise it will continuously try to connect if there is no MQTT broker.

## MU Sets the MQTT user name (V1.6+)

This is for MQTT brokers that are setup to require a username and password. For open MQTT brokers that do not require a user name/password, these may be ignored.

```
MU "myUsername"
```

```
OK. Saved MQTT User: myUsername
```

## MW Sets the MQTT password (V1.6+)

This is for MQTT brokers that are setup to require a username and password.

```
MW "mySuperSecretPassword"
```

```
OK. Saved MQTT Password: mySuperSecretPassword
```

## R1-R2 Sets the MQTT topic this relay is subscribed to

```
R2 "Workshop/Heater"
```

```
OK. Saved Relay 2 Topic: Workshop/Heater
```

In use, the payload for relay topics should be a string with the first character set to '0' or '1' (ASCII characters 0x30/0x31).

## TCP/IP Commands

The ESP32LR20 has a built in TCP/IP command set which allows you to control the module remotely.

All commands are sent using plain ASCII text. PuTTY is a good cross platform terminal program to use for testing. The TCP/IP port is the one you set-up with the PA command during USB configuration. Do not use port 80 as that is reserved for the HTML commands and Webpage.

### SR Set Relay

This is used to turn a relay on or off

To turn Relay 1 on:

```
SR 1 1
```

The first number is the relay number from 1 to 2.

The second number is 1 or 0, on or off.

So turn turn relay 1 off again:

```
SR 1 0
```

The command will respond with ok or fail.

```
SR 1 1
```

```
ok
```

```
SR 1 6
```

```
fail
```

< 6 is not valid, only 1 or 0 for on/off

```
SR 9 1
```

```
fail
```

< relay 9 does not exist.

### GR Get Relay

Will return the state of the relay.

To get the status of relay 2:

```
GR 2
```

```
1
```

```
GR 2
```

```
0
```

```
GR 9
```

```
fail
```

< relay 9 does not exist.



## Password

From version 1.5 we have added a password to the ASCII commands, this can be set with the AP command over the USB connection. The password is passed as a prefix to any command.

For example if a password is set and relay 1 is required to be turned on, start with the password (example password of 1234), then the command, so it becomes:

```
1234 SR 1 1
```

## HTML Commands

There are a set of HTML commands that can be used to control the module.

?Rly2=1 This will turn on relay 2

?Rly2=0 This will turn off relay 2

?Rly2=2 This will toggle relay 2 to the opposite state.

You can enter the commands into a browser immediately after the IP address.

<http://192.168.0.3/?Rly2=1>

This will turn on relay 2.

In response the module will return an XML file, which your browser will display.

```
<ESP32LR20DATA>
```

```
<RELAYS>
```

```
<RLY1>off</RLY1>
```

```
<RLY2>on</RLY2>
```

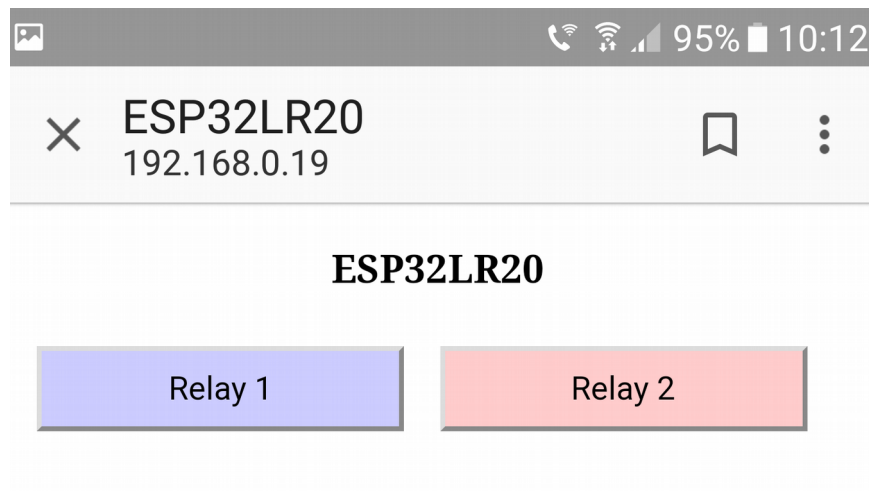
```
</RELAYS>
```

```
</ESP32LR20DATA>
```

The XML file is generated after the command has executed, therefore will reflect the new status of the relays.

## Webpage

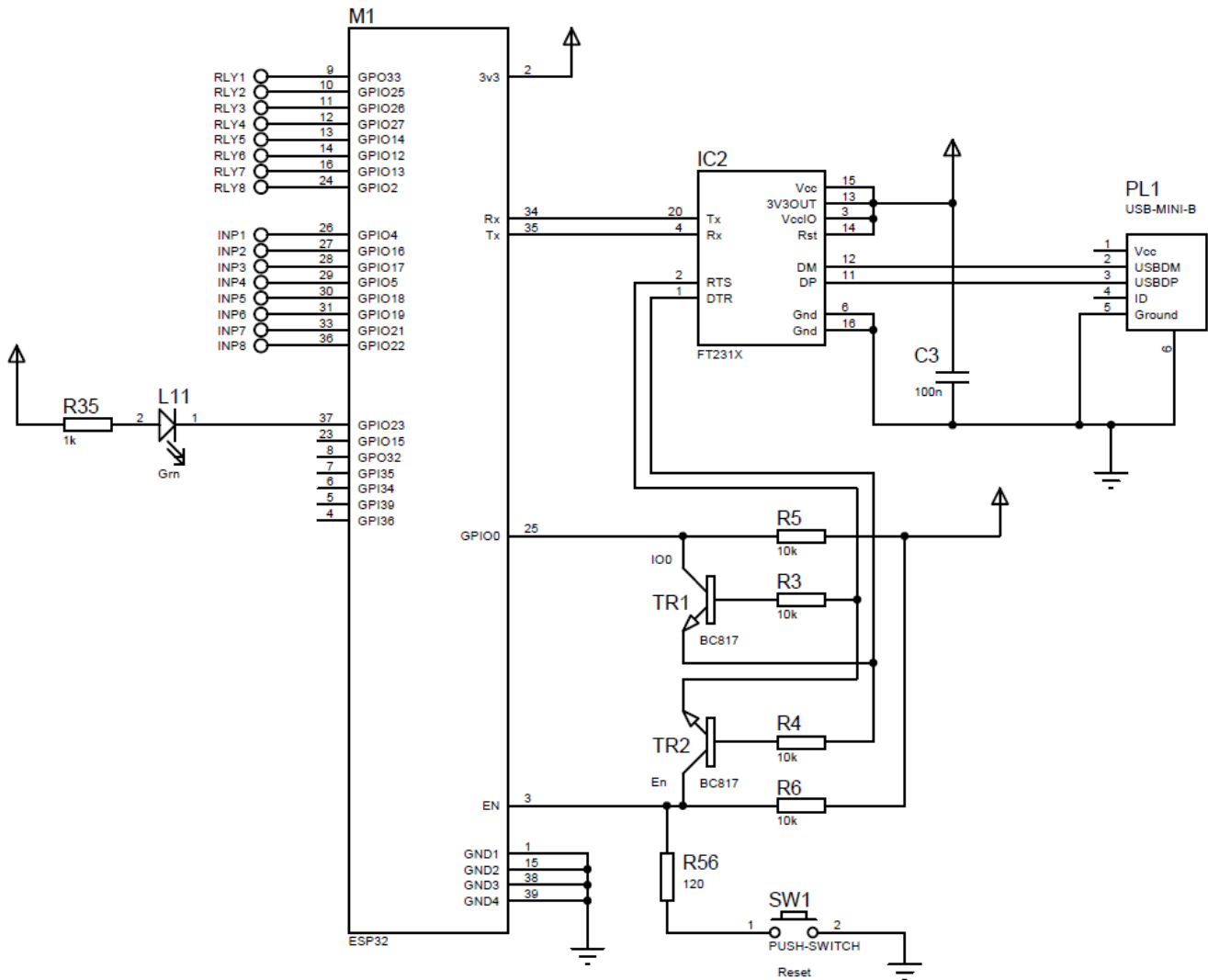
The built in webpage can be used as a remote app to monitor and control the relays. You can access the page as a default with just the IP address or by specifying index.htm.



The webpage contains the Javascript to send an HTML toggle command, as described in the previous section. It will send the toggle command each time a button is clicked. It then uses the responding XML file to colour the buttons.

## Schematics

### CPU

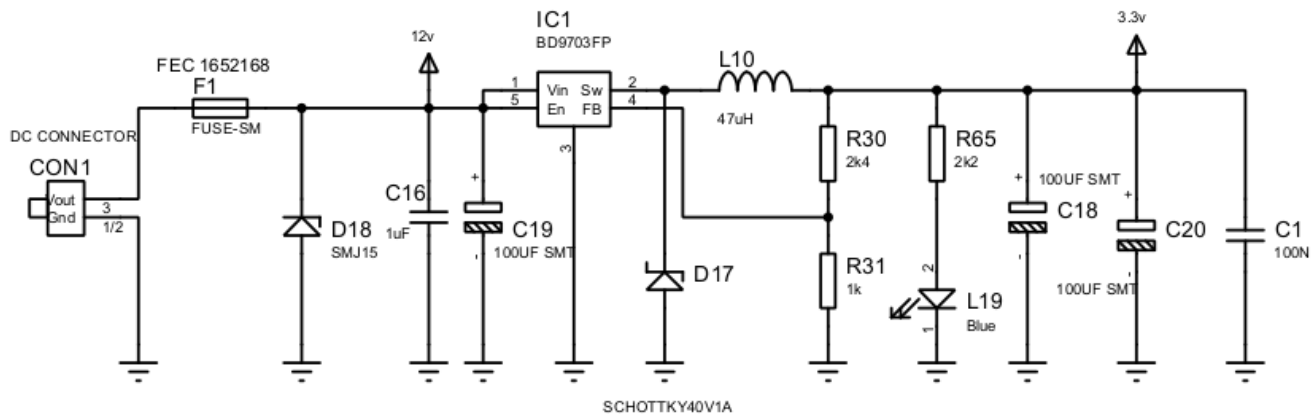


Note.

The CPU schematic is the same for the ESP32LR20, ESP32LR42 and ESP32LR88.

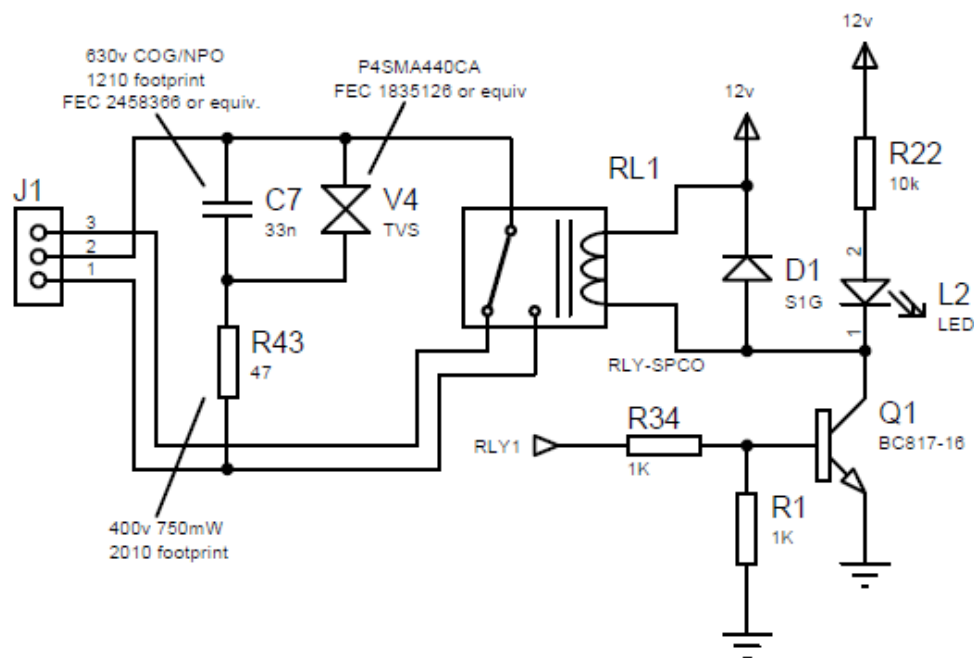
Relays 3-8 and Inputs 1-8 are not available on the ESP32LR20

## Power Supply



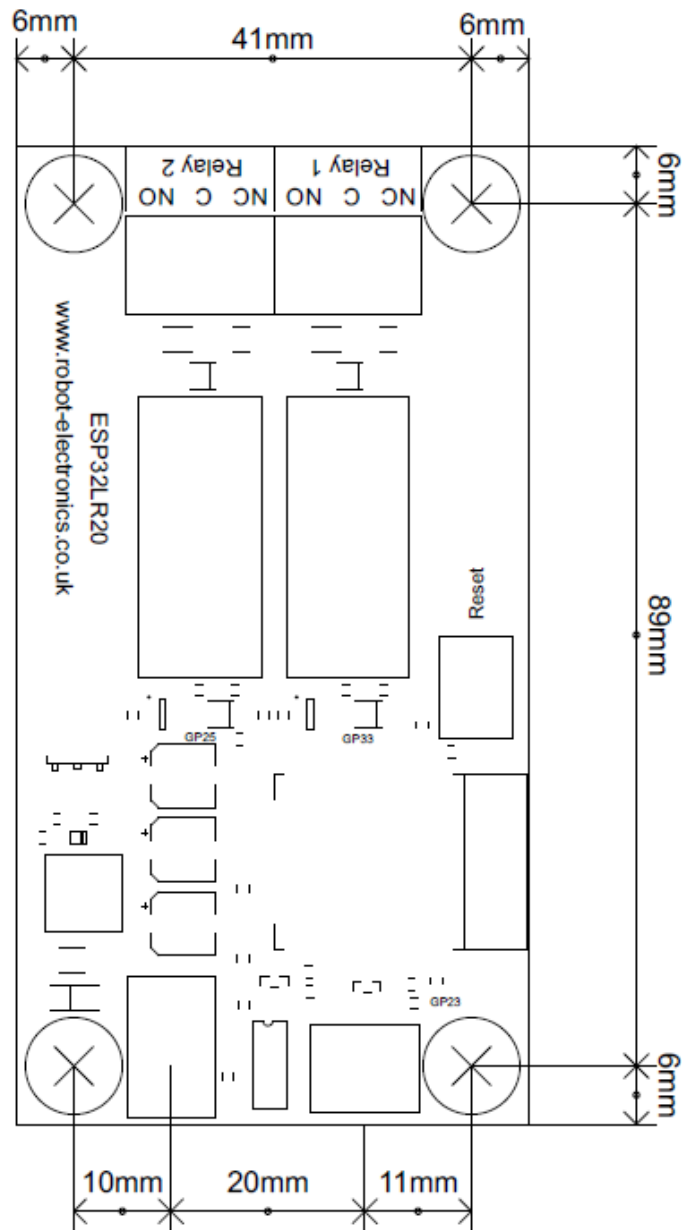
## Relay Outputs

1 of 2 identical circuits shown



The relays are capable of switching up to 16 Amps at 24vdc or 230vac. A data sheet of the relay can be found [here](#). Only the normally open (N/O) contact has the snubber circuitry.

## PCB dimensions



## Appendix 1

### Programming the ESP32LR20 with Arduino studio

Customising the ESP32LR20 can easily be achieved by using the Arduino studio and importing the required libraries.

#### **Step 1 – Arduino IDE installation**

Obtain and install the latest Arduino IDE from <https://www.arduino.cc/en/Main/Software> it should be the latest to ensure compatibility.

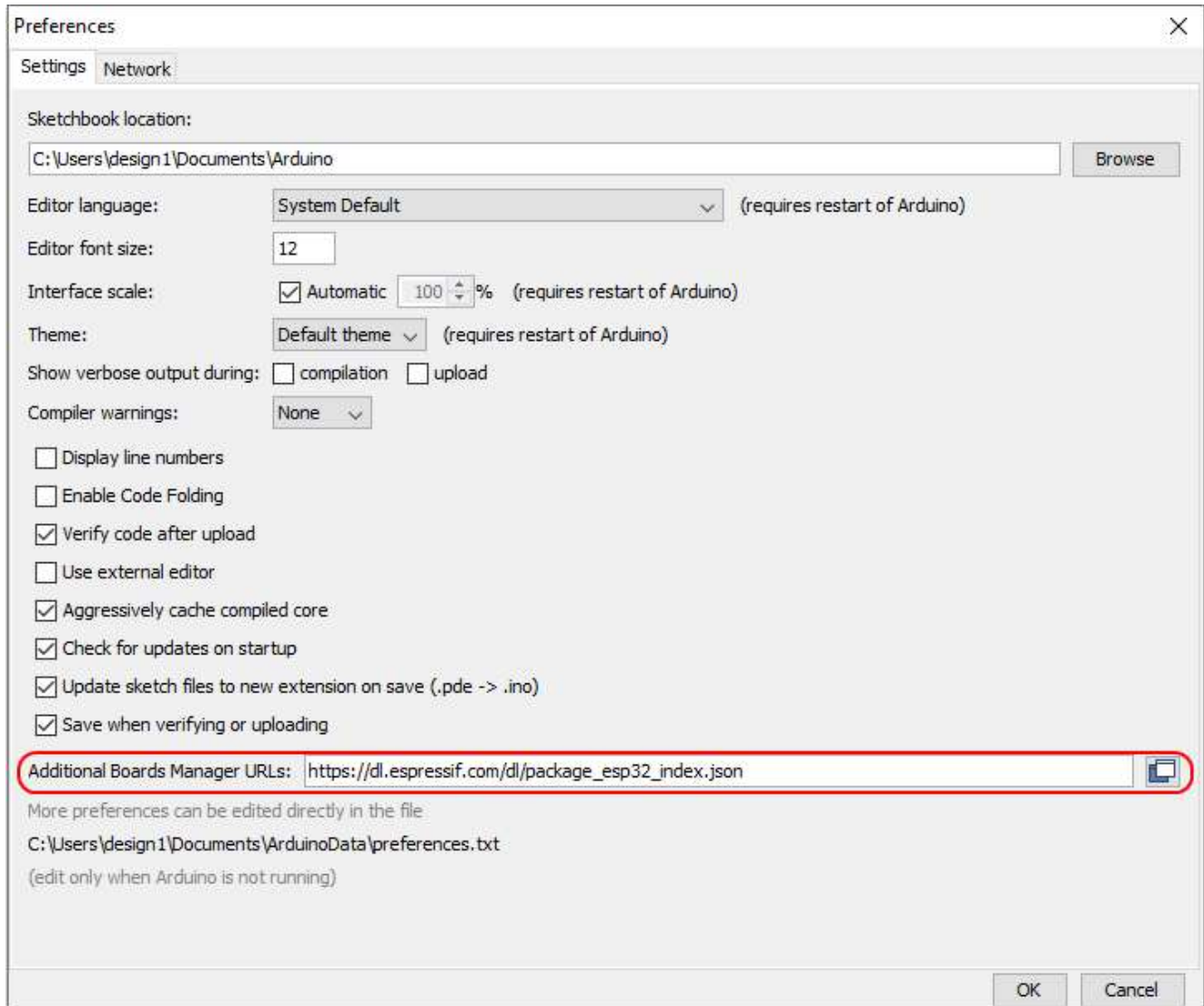
## Step 2 – Add the ESP32 library URL to the Arduino IDE

Go to File>Preferences

Now in the Preferences screen below we need to enter

[https://dl.espressif.com/dl/package\\_esp32\\_index.json](https://dl.espressif.com/dl/package_esp32_index.json)

into the “Additional Board Manager URLs” option. If you already have libraries added you may need to add a comma between the URLs

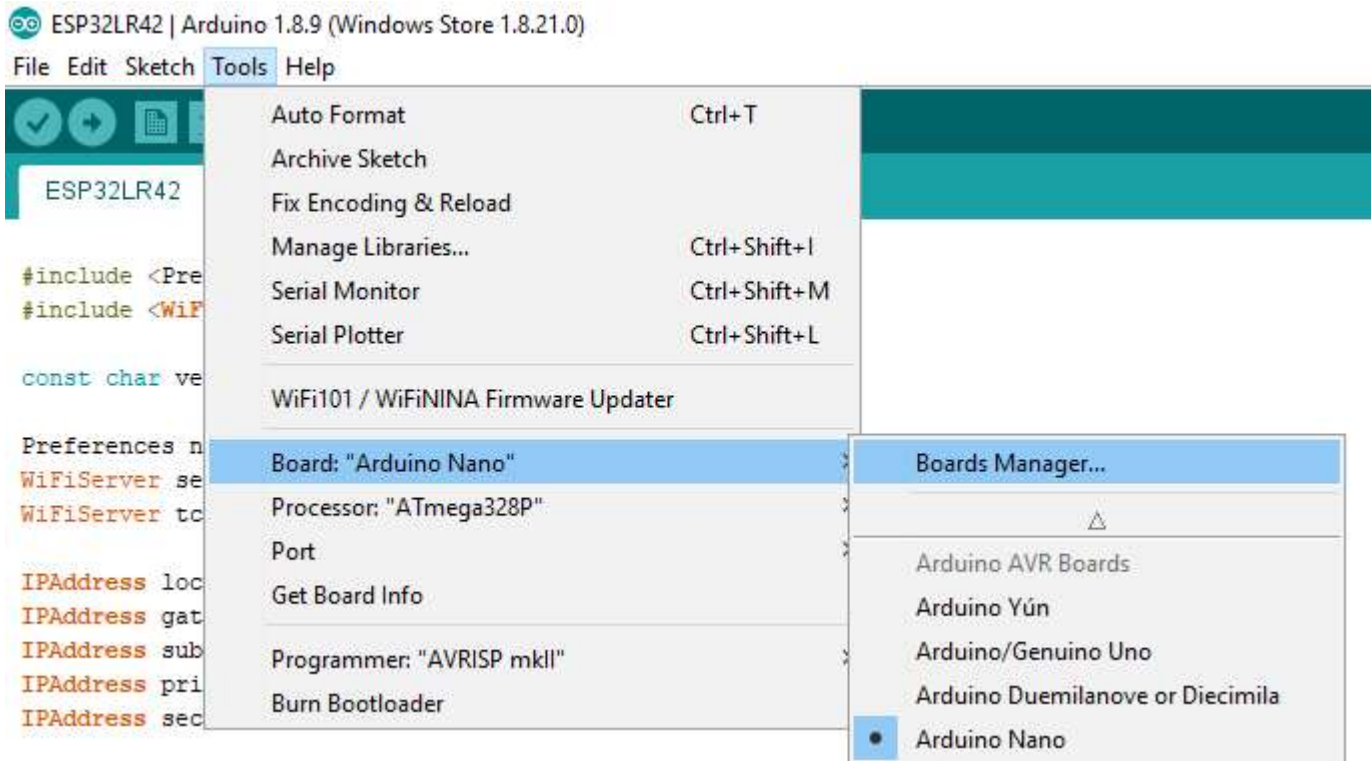


You can now click the OK button to finish with this screen.

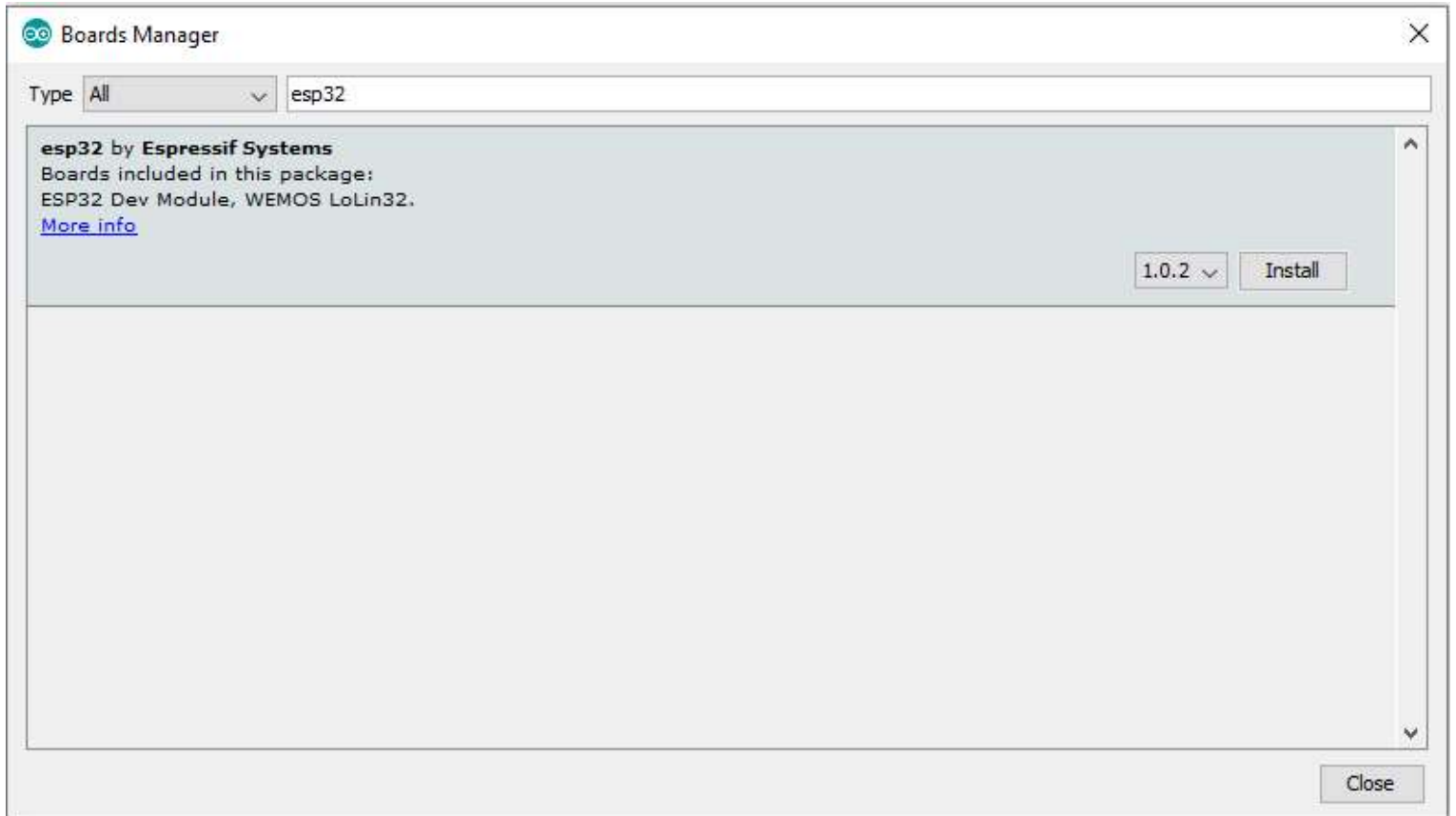


## Step 3 – Install the ESP32 library

Go to Tools>Board:>Boards Manager...



Now filter by “esp32” and install the Espressif Systems library



## Step 4 – Board selection

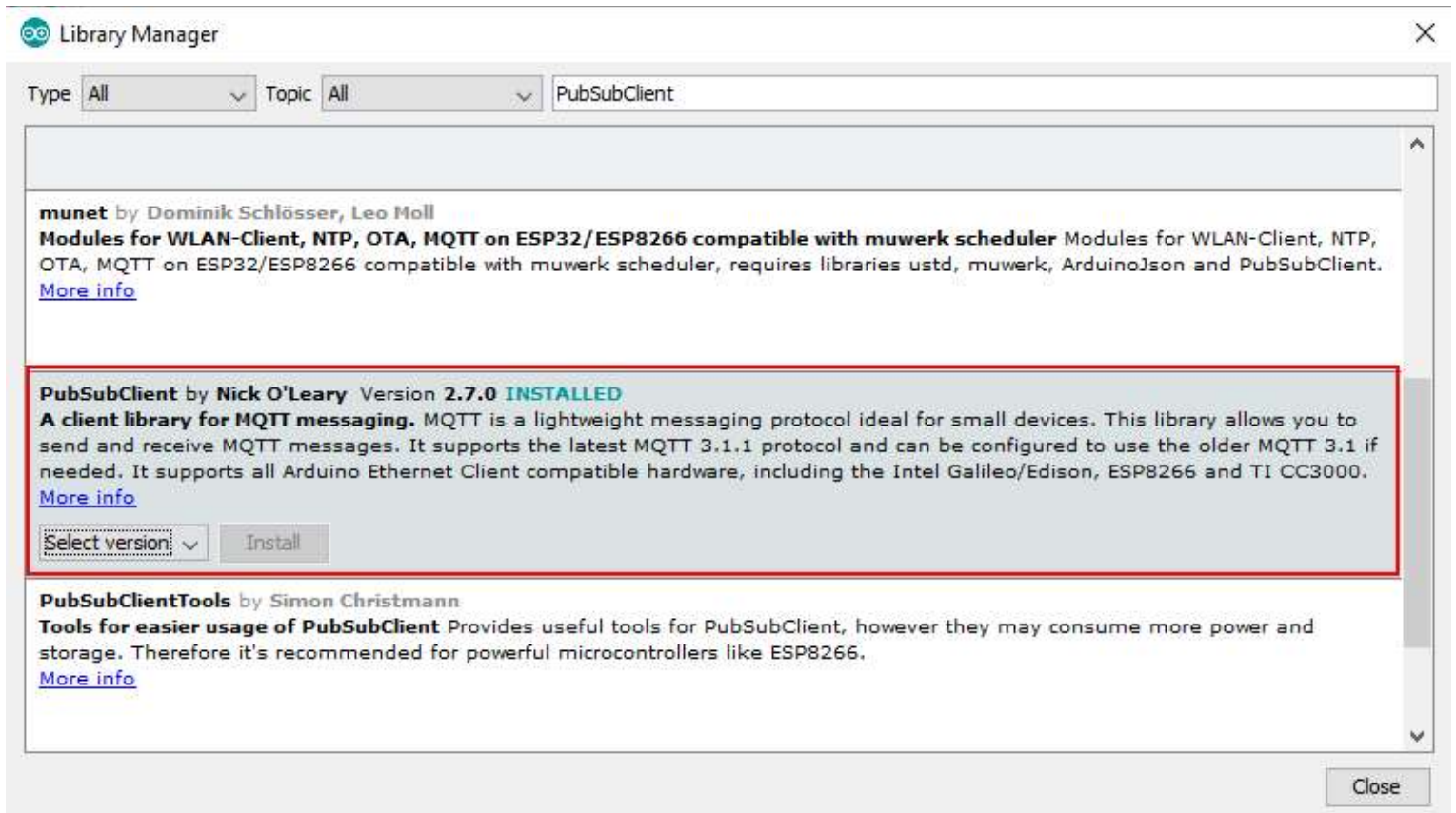
Go to Tools>Board:> and select ESP32 Dev Module



## Step 5 – Add MQTT library

Go to Tools>Manage Libraries...

Filter by PubSubClient and select PubSubClient by Nick O’Leary, then press the install button



That’s it! Your Arduino IDE should now be able to program the ESP32LR20 module. The factory shipped code is available Here: <https://github.com/devantech>

## Notes