

# Building the Firmware (optional)

Vapour Soft edited this page · [4 revisions](#)

## Contence:

- [Home](#)
- [Building the Firmware \(optional\)](#)
- [Building with PlatformIO](#)
- [Building with Arduino IDE](#)
- [Original Notes](#)
- [RTS/CTS handshaking and a dead spin loop issue](#)
- [Additional Notes](#)
- [Command Reference](#)
- [Flashing the firmware](#)
- [Hardware Assembly](#)
- [Notable Changes](#)
- [Quick Start](#)
- [Usage Notes](#)

## Building...

If you want to build the firmware yourself, rather than just downloading the pre-built binaries - here is some info:

**Jan 24/22:** It's been reported that the ESP8266 core is slightly snafu'd at the moment, and that it's breaking things in the modem software...

## Building with PlatformIO

platformio.ini currently pins the platform io core to espressif8266@2.6.3 as a workaround for the ESP8266 core issue mentioned above.

## Building with Arduino IDE

Please ensure you use ESP8266 core 2.7.4 as a workaround for the ESP8266 core issue mentioned above. I use PlatformIO as an IDE, so you will probably need to rename .cpp files to .ino to build with the Arduino IDE as a minimum.

## Original Notes

from RetroWifiModem's README.md

## RTS/CTS handshaking and a dead spin loop issue

Something I noticed with ESP8266 software that puzzled me was the number of places I saw a series of Serial print statements being broken up with calls to yield(), like so:

```
Serial.print("Hello world!\n"); yield(); Serial.print("How are you today?\n"); yield();
```

It didn't take long to figure out what was going on; The print() call was blocking, and at lower baud rates, even printing a few relatively short strings was enough to cause the watchdog to bark and cause a reset. So the repetitive yield() calls were an attempt to feed the watchdog often enough to keep it from barking.

What does this have to do with RTS/CTS handshaking? Simply put, lowering RTS for more than a few seconds was causing the watchdog to bark as well. So I started digging.

In cores/esp8266/uart.cpp I found the following function:

```
static void
uart_do_write_char(const int uart_nr, char c)
{
    while(uart_tx_fifo_full(uart_nr));

    USF(uart_nr) = c;
}
```

This is the low level function that everything calls to send a character out the serial port. The cause of the watchdog barking is in the dead spin while loop. It waits until there's room in the transmit FIFO to add another character. So if RTS/CTS handshaking is enabled, and RTS is low for longer than the watchdog likes: woof.

What I did to quiet the watchdog down both when sending long strings at low baud rates and during long waits for RTS to come active again was to add a yield() call to the dead spin while loop, like so:

```
static void
uart_do_write_char(const int uart_nr, char c)
{
    while(uart_tx_fifo_full(uart_nr))
        yield();

    USF(uart_nr) = c;
}
```

This way, no matter how long the code has to wait for space in the transmit FIFO, the watchdog is kept well fed and quiet.

## Additional Notes

If you build with PlatformIO the above patch should be applied automatically via the included python script as part of the build see platformio.ini and ./patches.