



instructables

RotorHazard OTA Updater



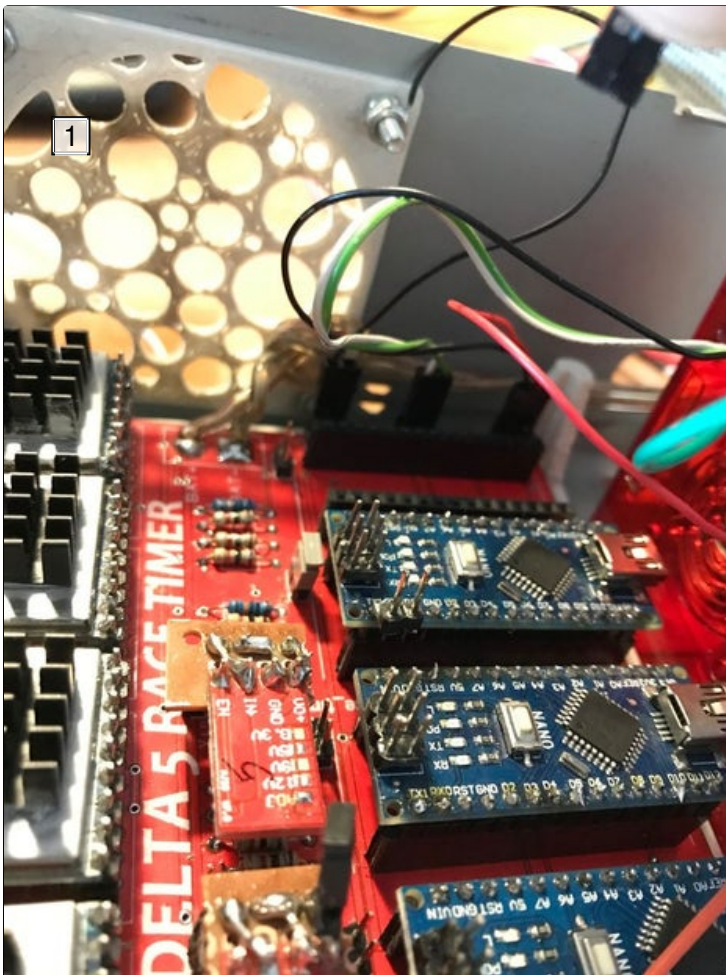
by szafranski

Collect all of required components.

Make sure that you have internet connection.

Supplies:

1. RotorHazard race timer
2. About 20 cm of thin wire + jumper wires or tool for attaching female gold-pins to the wire
3. Soldering iron + solder
4. 2 resistors: 5kOhm and 10 kOhm - or any other combination with 1/2 ratio in 5-20 kOhm range
OR logic level converter - using resistors is easier
5. Few gold pins - optional
6. PC connected to the internet



1. my friend is as architect - he designed this...

Step 1: Make Additional PCB Wiring

Connect all of the TX pins together and all of RX pins together - on Arduinos.
You can do it underneath the PCB so it will still look clean at the top..

Only the horizontal yellow wires - according to photo - are required.

If you have 2 PCBs connected together - connect all of the pins from both PCBs.



Step 2: Connect Arduino Nodes to Raspberry Pi

The common Arduinos RX line has to be connected to Raspberry's TX pin (GPIO 8).

Arduinos TX pins have to be connected to Raspberry's RX pin (GPIO 10) via logic level converter or with voltage divider. It is caused by the difference of voltage tolerance between the Arduino and the Raspberry. Arduino operates at 5V logic level and Raspberry at 3.3V. Converting the logic level is only required on the line where Arduinos TXes are connected to Raspberry's RX, cause Arduino transmits at 5V.

On the other line the Pi is transmitting at 3.3V and Arduinos are only receiving so it is save without voltage divider or level converter.

I soldered two male gold-pins to one of the Arduinos so I have RX and TX line exposed near the Raspberry.

You can do it the same way. Next I placed small PCB with the voltage divider on those pins. You don't have to do it this way. I just didn't want to have more cables than needed in my timer.

Besides that RST (reset) pin of every Arduino has to be connected to Raspberry's GPIO pins. It can be done according to default pins assignment - table on the attached photo.

If you want to use another pins - make changes in the update.py file accordingly. Just remember to use general purpose pins. Not reserved ones etc. You can use this command for make changes:

'nano ~/RH-ota/update.py'

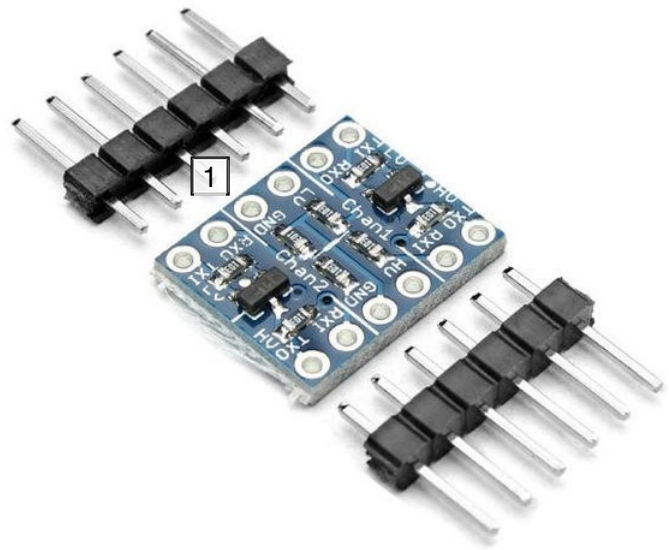
Alternate Function					Alternate Function
	3.3V PWR	1		2	5V PWR
I2C1 SDA	GPIO 2	3		4	5V PWR
I2C1 SCL	GPIO 3	5		6	GND
	GPIO 4	7		8	UART TX
	GND	9		10	UART RX
	GPIO 17	11		12	GPIO 18
	GPIO 27	13		14	GND
	GPIO 22	15		16	GPIO 23
	3.3V PWR	17		18	GPIO 24
SPI0 MOSI	GPIO 10	19		20	GND
SPI0 MISO	GPIO 9	21		22	GPIO 25
SPI0 SCLK	GPIO 11	23		24	GPIO 8
	GND	25		26	GPIO 7
	Reserved	27		28	Reserved
	GPIO 5	29		30	GND
	GPIO 6	31		32	GPIO 12
	GPIO 13	33		34	GND
SPI1 MISO	GPIO 19	35		36	GPIO 16
	GPIO 26	37		38	GPIO 20
	GND	39		40	GPIO 21
					SPI1 CS0
					SPI1 CS1
					SPI1 MOSI
					SPI1 SCLK

1. UART pins
2. pins used as reset pins by default

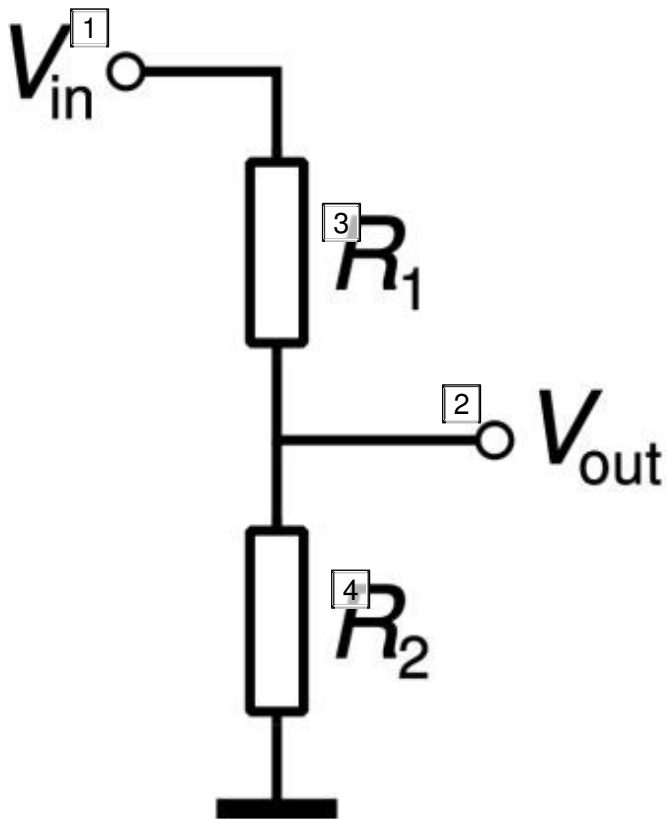
1. UART pins
2. reset pin is doubled on the ISP header - you can use that one if you want
3. reset pin

Arduino	¹ Raspberry's GPIO pin
1	12
2	16
3	20
4	22
5	6
6	13
7	19
8	26

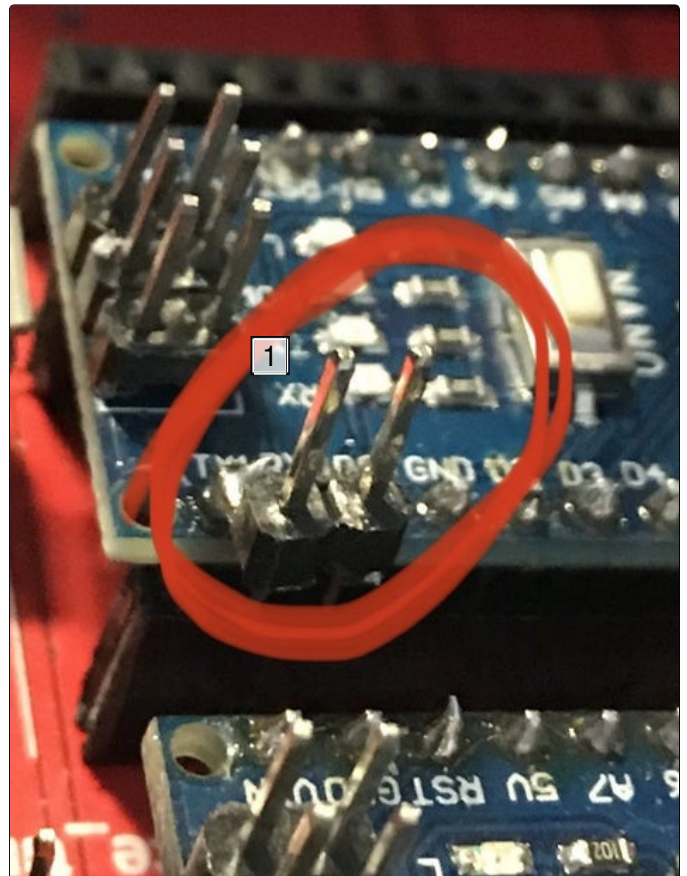
1. default pins assignment



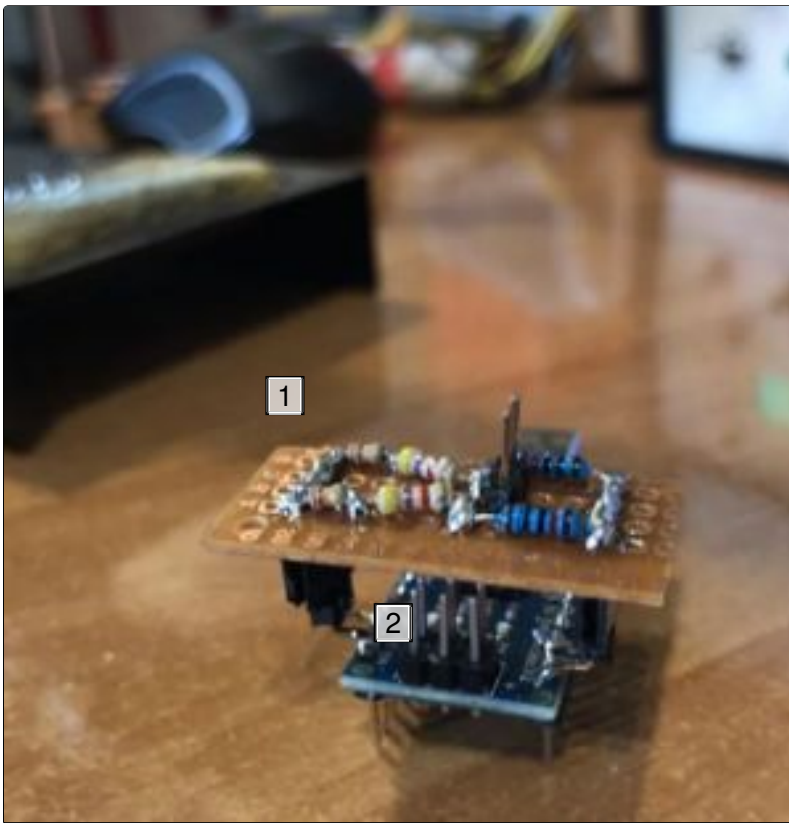
1. logic level converter



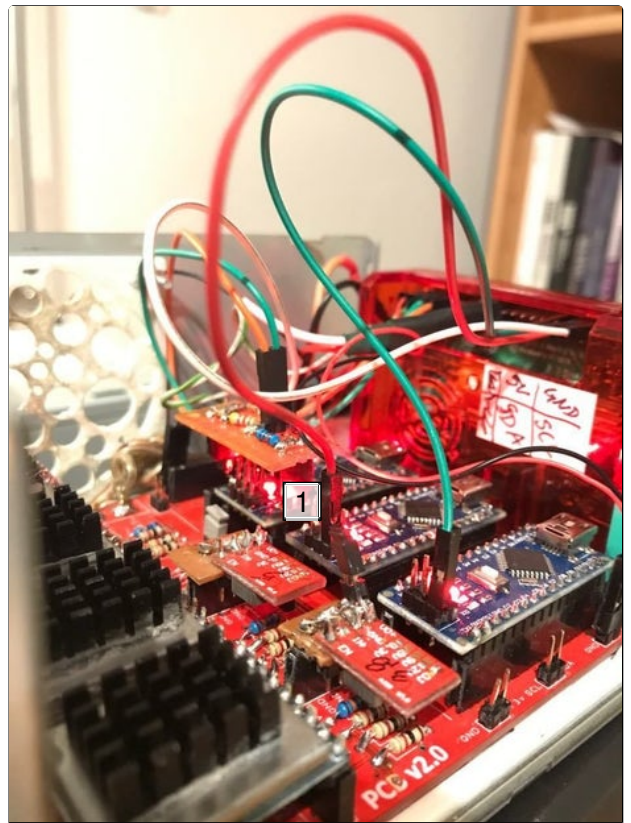
- 1. Arduino - TX - 5V
- 2. Raspberry - RX - 3.3V
- 3. 5kOhm
- 4. 10kOhm



1. additional gold-pins - UART



1. voltage divider PCB
2. one of the nodes



1. reset wires

Step 3: Login Into Raspberry Via SSH and Download the Updater

Open the ssh connection with your Raspberry.
Establish connection to the internet.

You can also hook up display to the Raspberry and
login into Raspbian at the Raspberry itself.

After logging into Pi download repository from
github page using those commands:

```
git clone https://github.com/szafranski/RH-ota.git
```

Enter downloaded folder:

```
cd RH-ota
```

And open update script:

```
python update.py
```

If you got an error after entering first command you
probably have to install git from apt.

Use command: `sudo apt install git`

```
pi@raspberrypi:~/RH-ota
C:\Users\sssh pi@192.168.137.120
pi@192.168.137.120's password:
Linux raspberrypi 4.19.97-v7+ #1294 SMP Thu Jan 30 13:15:58 GMT 2020 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Fri Feb 21 14:22:43 2020 from 192.168.137.1

SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set a new password.

pi@raspberrypi:~$ git clone https://github.com/szafranski/RH-ota.git
Cloning into 'RH-ota'...
remote: Enumerating objects: 792, done.
remote: Total 792 (delta 0), reused 0 (delta 0), pack-reused 792
Receiving objects: 100% (792/792), 5.72 MiB | 1.86 MiB/s, done.
Resolving deltas: 100% (390/390), done.
pi@raspberrypi:~$ ls
backup_RH_data  bme280  pi_ina219  RH-ota  RotorHazard  RotorHazard.old  rpi_ws281x
pi@raspberrypi:~$ cd RH-ota
pi@raspberrypi:~/RH-ota$ ls
firmware  how_to  README.md  resources  rpi_soft.py  update.py
pi@raspberrypi:~/RH-ota$ python update.py
```

Step 4: Prepare Raspbian OS

If you are doing this for the first time you have to install software that connects with Arduino and has ability to program it. Do it by entering Additional Features menu and select point 1 - "Install avrdude".

Next you have to enable serial port on GPIO header (UART protocol) and prepare it to be connected with external device. It is utilized to be the console output

by default so it is basically useless.

Enter Additional Features menu and point 2 - "Enable serial protocol".

Next you will be asked to reboot the Raspberry. Do it.

```
#####
###                               ###
###           RotorHazard         ###
###                               ###
### You are about to flash nodes firmware. Please do not interrupt this operation. ###
###                               ###
#####

          FEATURES MENU

1 - Install avrdude
2 - Enable serial protocol
3 - Fix GPIO pins state
4 - Raspberry as Access Point - coming soon
5 - Useful aliases - coming soon
6 - Go back
```

Step 5: Use Downloaded Software

After rebooting use downloaded software.

python ./RH-ota/update.py

If you want to update or install (or downgrade) server software enter point 1.

If you want to flash firmware on Arduinos enter point 2.

Follow the instructions on the screen.

It should works automagically :)

```
#####
##                                     ##
##          RotorHazard              ##
##                                     ##
##  You are about to flash nodes firmware. Please do not interrupt this operation!
##                                     ##
#####

MAIN MENU
1 - Server software installation and update
2 - Nodes flash and update
3 - Start the server now
4 - Additional features
5 - This is my first time - READ!
6 - Exit
```

```
AUTOMATIC UPDATE AND INSTALLATION OF ROTORHAZARD RACING TIMER SOFTWARE

This script will automatically install or update RotorHazard software on your Raspberry Pi.
All additional software depedancies and libraries also will be installed or updated.
Your current database and config file should stay on the updated software.
After rebooting please check by typing 'sudo raspi-config' if I2C, SPI and SSH protocols are active.

Source will be 

1

 ter repository of RotorHazard software on github - or version choosen by you.
Make sure that you are logged as us 

2

 .
You can change those by oppening file 'rpi_soft.py' in text editor - like 'nano'.

Enjoy!

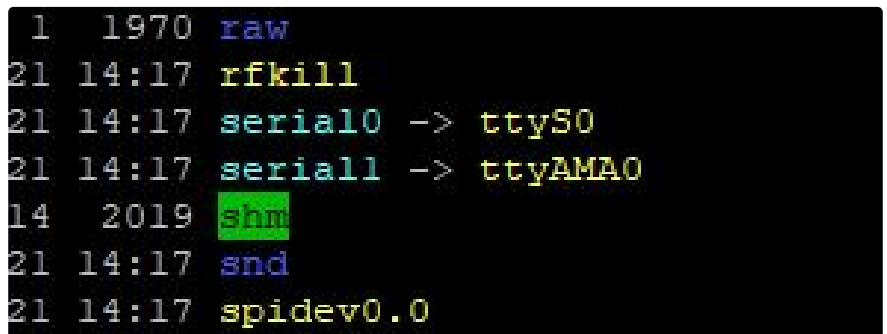
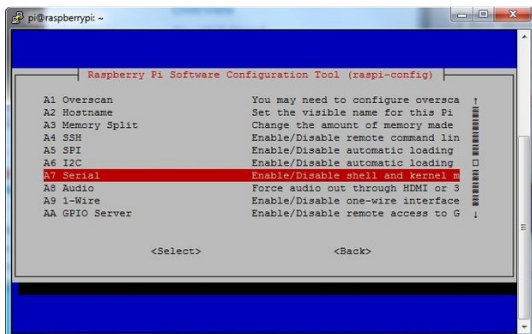
'i' - Install software from skcratch
'u' - Update existing installation
'a' - Abort
```

1. can be changed
2. should match username of the Raspberry

Step 6: Things Worth Mentioning / Troubleshooting

Solutions to possible problems:

- Arduinos can't be flashed - check the wiring. Remember that EVERY Arduino has to be connected via reset pin and UART (RX/TX pins). All Arduinos are being reset when each of them is being flashed. It is caused by the communication on the UART line which is normally being performed if node is active. So they have to be in reset state for so of them can be flashed - software do it automatically. If even one of them is active, none of them can be programmed.
- Updating script doesn't want to open - check if python is installed and install it by using command: 'sudo apt install python'
- Software works but in some cases it crashed or exits to system - make sure that your username in both files (update.py and rpi_soft.py) matches your system username.
- Updating server software takes long time - when server is being updated, the Raspberry itself is being updated as well. If you haven't updated Raspbian (Raspberry's OS) for some time it can take up to 20 minutes. If you get only errors - check internet connection.
- Few nodes updated with no problem but few of them were being flashed very slowly and I can't see them after opening the server - use option "Flash each node individually" and flash problematic nodes this way.
- I tried few times and I really think that my Raspberry's UART isn't working properly - open configuration assistant on your Pi by typing 'sudo raspi-config' -> Interfacing options -> Serial and after hitting enter choose "no" for first question and "yes" to second one. Reboot if asked. You can also type 'cd /dev' and then 'ls -l' and look for 'serial0 -> ttyS0'. It should look as on attached image. If it is, it definitively means that your UART is working.
- After enabling Serial on the Raspberry and rebooting I can't connect to Raspberry via SSH - probably some communication is being performed on UART line and prevents Raspberry from booting. Unplug those wires for a moment and reboot the Pi.



Step 7: Things I Had to Go Through / History of Development

At first I was trying to utilize the SPI protocol. I have connected spi1 (second SPI bus of the Raspberry Pi - first is occupied by LEDs) to Arduinos. After some hustle I was able to program the nodes but it was risky cause SPI protocol has ability to erase whole chip - bootloader included. So if something went wrong Arduino became useless. I had to pull it out and program with external programmer. Besides that 2 of 5 Arduinos used during testing are completely bricked now cause SPI protocol can change "fuses" of the chip. It is very low level programming stuff and can't be restored easily. The worst thing was that if SPI bus was connected to the Raspberry it couldn't be used to changing the channels of the 5808 receivers. I tried to use SoftwareSPI on Arduinos but it required changing pins assignment, more wiring etc. It was messy and unelegant. Besides, Arduinos had to be flashed with 'hex' files without the bootloader, files had to have additional delay during boot etc. Moreover special version of avrdude - program used for programming the Arduino - had to be used with special programmer

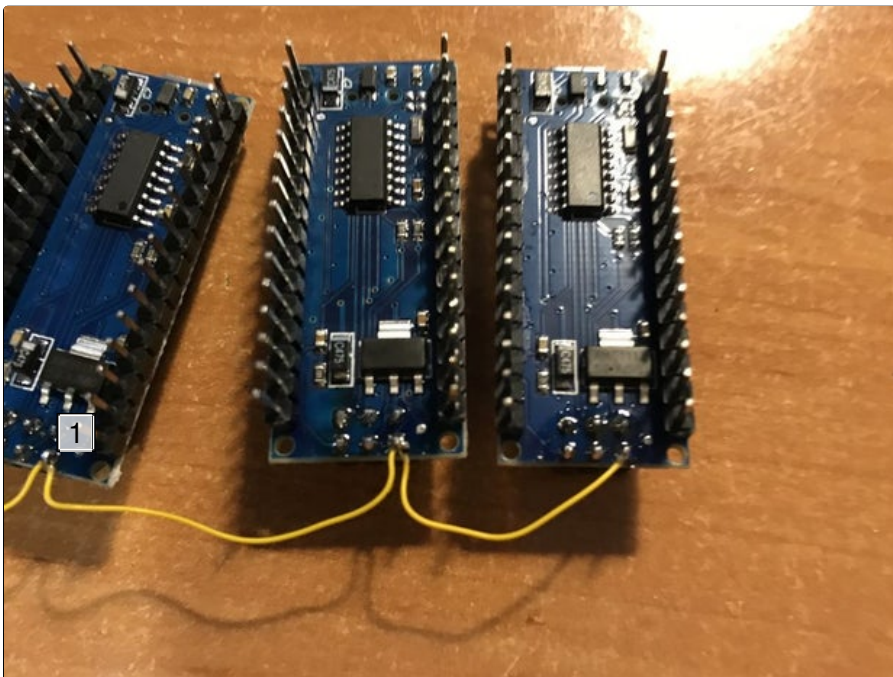
compiled... Bad things. For me it was still worthy but I was worried that no one would want to do it too and that someone could brick Arduinos before race or something.

I then realized that Raspberry has an UART line and that Arduino normally can be very easily programmed with this protocol. Basically every time Arduino is connected via USB it uses UART - and USB to serial converter (this small chip at the bottom of the Nano. I have connected the dots and after some troubleshooting process - enabling the UART on the Pi is not that obvious - I successfully connected Raspberry's UART to PC's COM port. At that point I was sure that Arduino can be programmed this way. Luckily programming via UART is a standard way of flashing so normal version of avrdude can be used. Even the software programmer is called "arduino". Only thing left was to make a way to establish connection with each Arduino before programming so it can be ready to be flashed. Right now resetting the nodes is done "manually" in python script which is easy to control and doesn't require changes in avrdude config files etc.

The biggest advantages of the way that this program operates right now are:

- bootloader stays on it's place, Arduinos can't be broken (or odds are as small as with programming with the PC)
- even if power outage occurs during flashing it shouldn't cause any issues
- code on the nodes is EXACTLY the same as standard code from RotorHazard - you can unplug Arduinos and plug to the PC and use with server via USB etc.

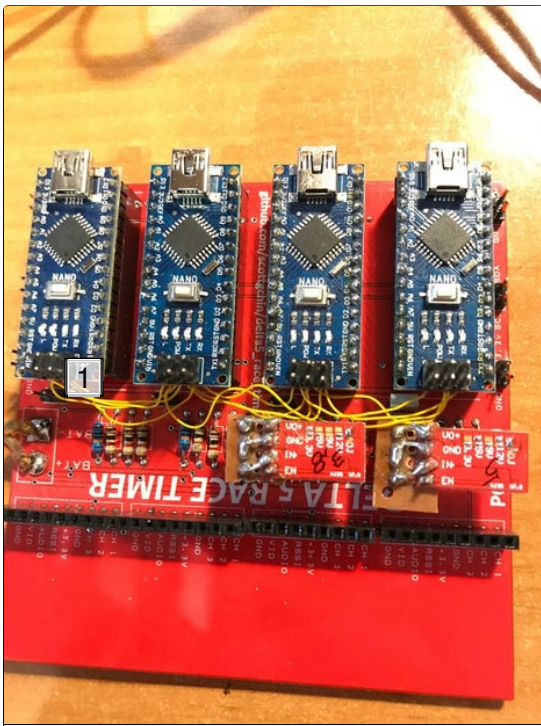
At the end I combined two separate programs so now you can update both - Raspberry's server and Arduino-based nodes using same program. Some additional features like making Raspberry as standalone Access Point automatically will come later.



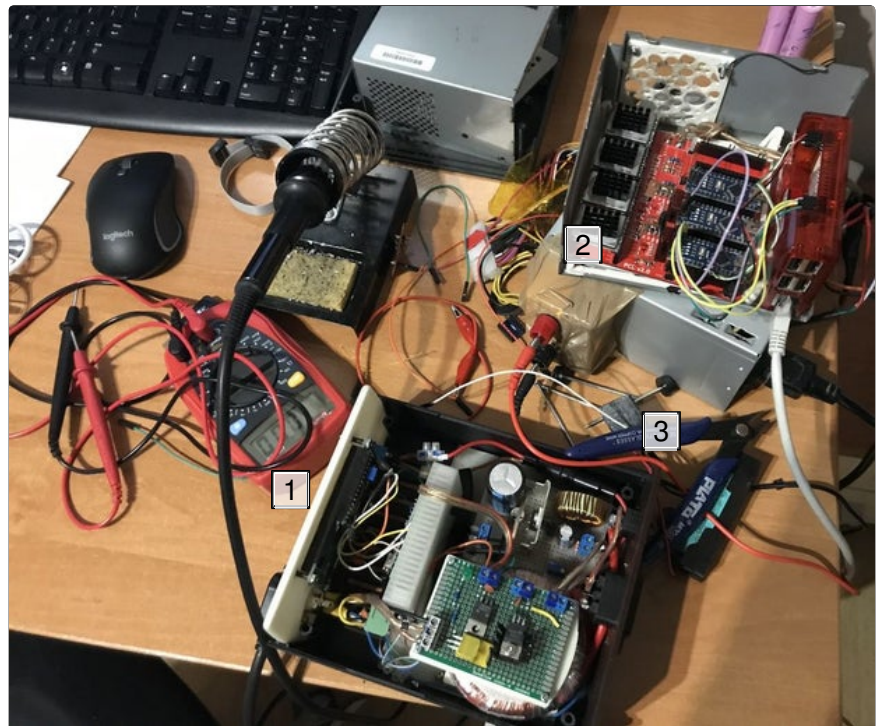
1. first attempts



1. big mess



1. SPI lines connected



1. DIY soldering station
2. test bench
3. but those if you don't have them



1. turning point - too much wires and changes

Step 8: Thanks + Disclaimer

Thanks for all the people from the RotorHazard Facebook group and all of developers (Delta5 timer as well). This is my small contribution to the project but without you this awesome timer couldn't exist. Thanks for all advices to the people on the Group. You are awesome. Especially thanks for people that were willing to test this software and this mod and people who responded to my questions on Facebook and gave me some ideas and feedback or just a kind word. Also thanks for Rodrigo Cardenas for additional testing and being first user.

Disclaimer: I've done my best to make this instruction as clear as possible. I am also sure that this procedure can't broke your electronics or software. I did lots of

testing. From the other hand I don't have control over your soldering skills or special situations with software like unusual Raspberry's configuration etc. You perform this mod at you own responsibility.

Feel free to give me the feedback and comment if you wish.

Github page of RotorHazard project:
<https://github.com/RotorHazard/RotorHazard>

Github of this project:
<https://github.com/szafranski/RH-ota>



RotorHazard
FPV RACE TIMING

1. <https://www.facebook.com/groups/rotorhazard>



Thanks for sharing :)



Thanks :) This instructable will be polished soon.