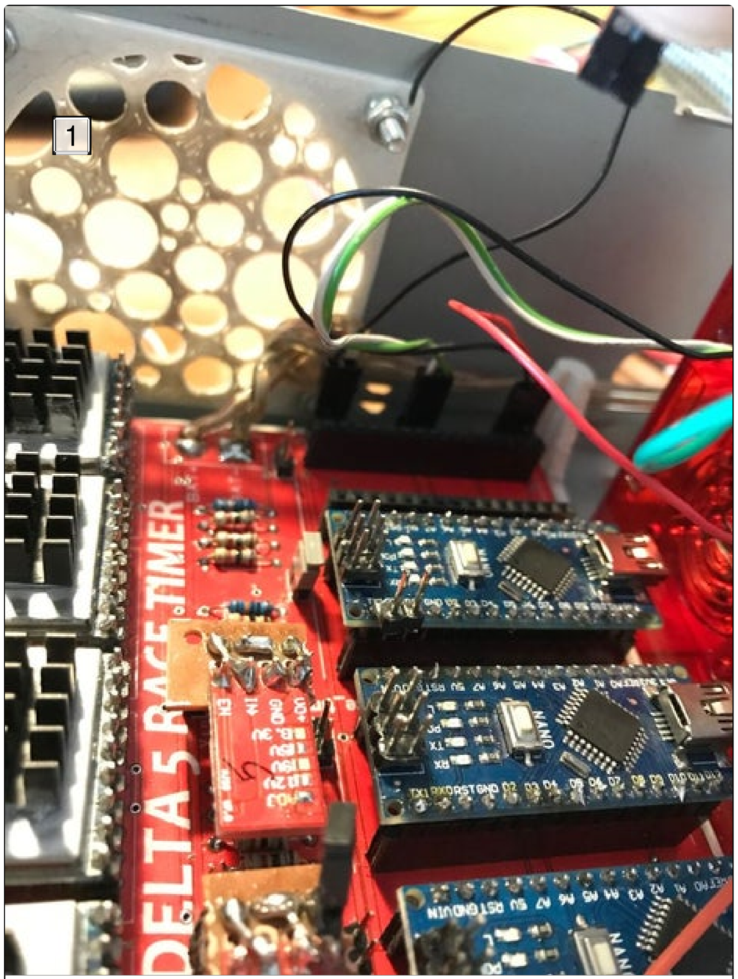
RotorHazard OTA Updater

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
5. PC connected to the internet



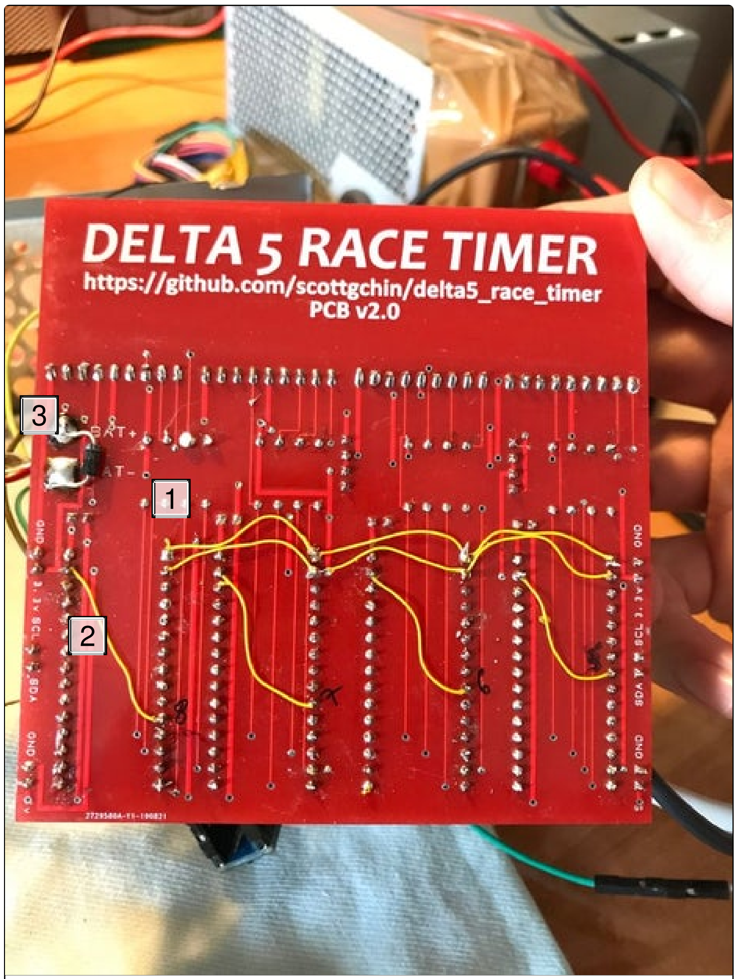
1. my friend is as architect - he designed this...

Step 1: Make Additional PCB Wiring

Connect all of the TX pins together and all of RX pins together - Arduinos ones.
You can do it underneath the PCB so it will be nicely 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.



1. RX and TX lines
2. ground auto-numbering mod
3. diode - some basic protection against reversed voltage

Step 2: Connect Arduino Nodes to Raspberry Pi

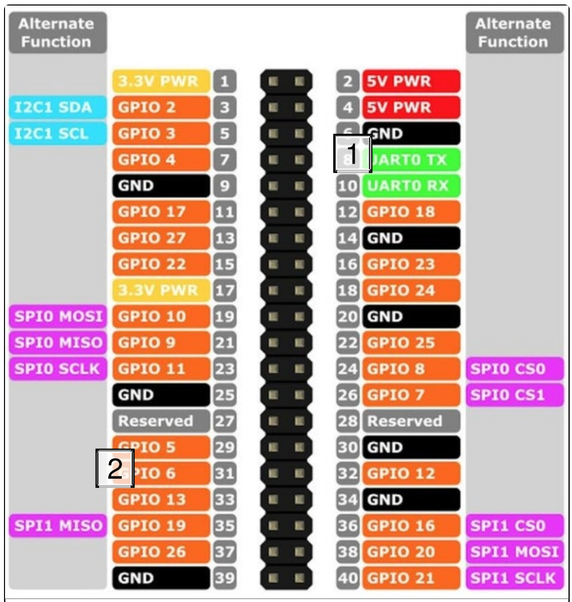
The common Arduinos RX line has to be connected to Raspberry's TX pin (GPIO 8).
Arduinos TX pins have to be connected to Raspberry's RX pin (GPIO 10) via logic level converter or with voltage divider. It is caused by difference of voltage tolerance of the Arduino and Raspberry. Arduino operates at 5V logic level and Raspberry at 3.3V. Converting the logic level is only required on the line where Ardunos TXes are connected to Raspberry's RXes, cause Arduino transmits at 5V.
On the other line the Pi is trasmitting and Arduinos are receiving so it is save without voltage divider or llc.

I solderer two male gold-pins to one of the Arduinos. 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.

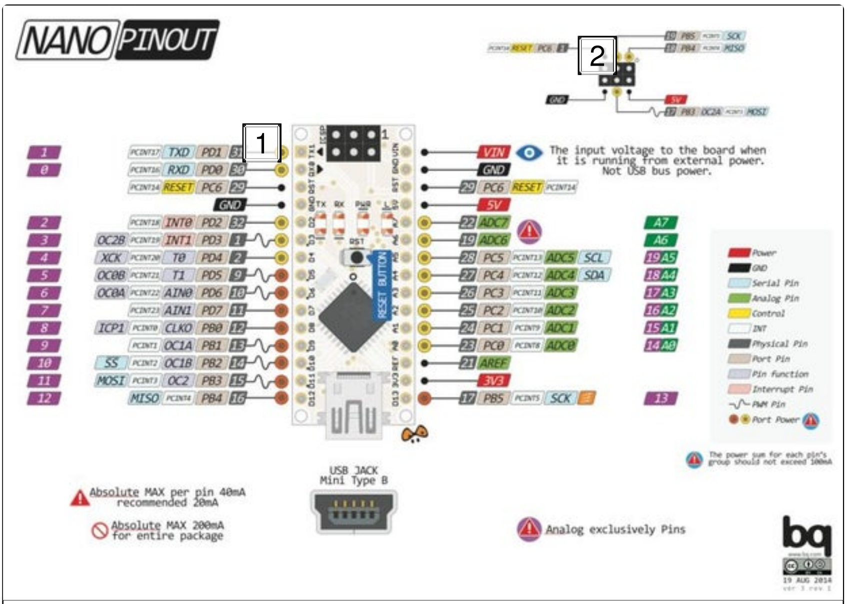
Besides that RST (reset) pin of every Arduino has to be connected to Raspberry's GPIO pins. It can be done according to default pins assignment - table on the attached photo.

If you want to use another pins - make changes in the update.py file. You can use this command:

nano ~/RH-ota/update.py

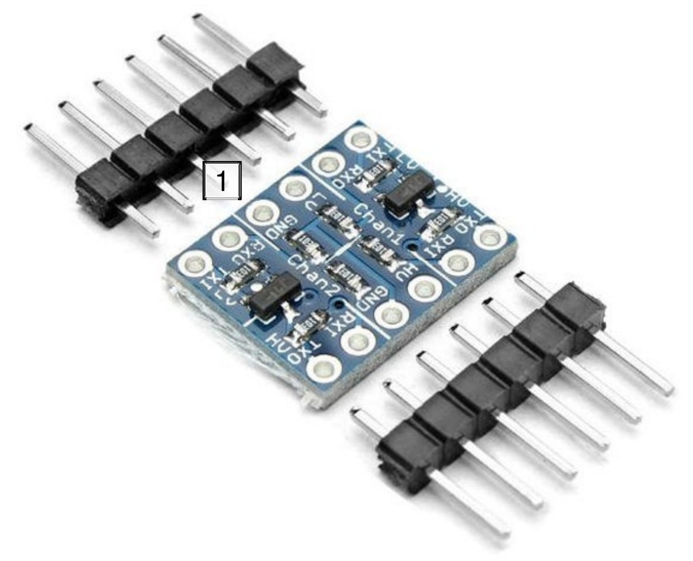


1. UART pins
2. pins used as reset pins by default

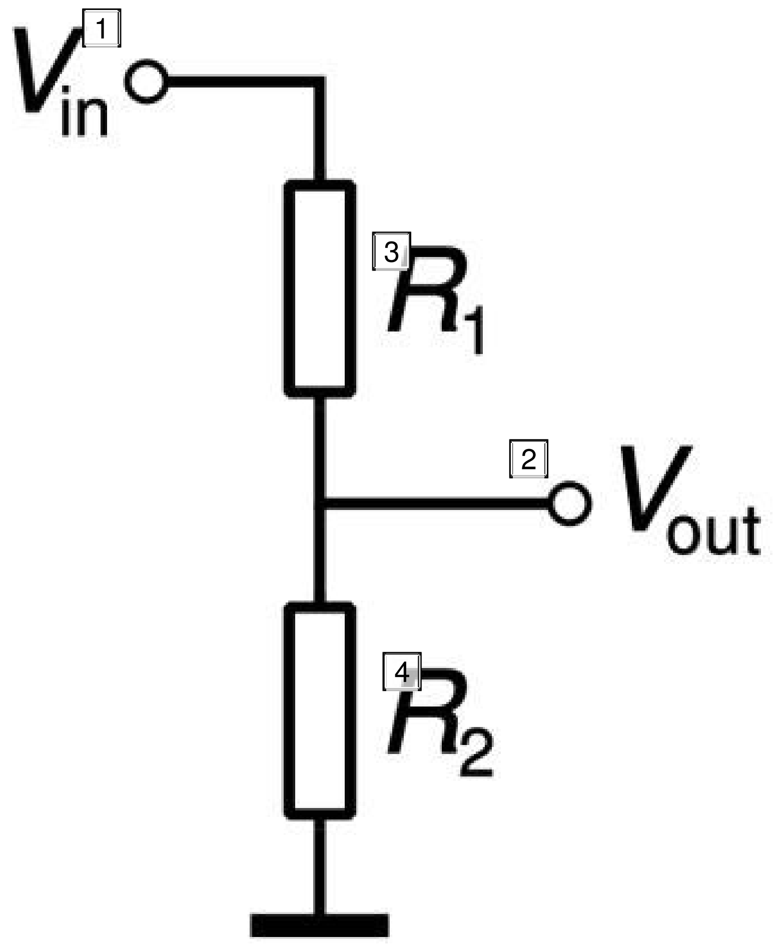

NANO PINOUT
1
2
The input voltage to the board when it is running from external power. Not USB bus power.
RESET BUTTON
USB JACK Mini Type B
Absolute MAX per pin 40mA recommended 20mA
Absolute MAX 200mA for entire package
Analog exclusively Pins
The power sum for each pin's group should not exceed 100mA
Power
GND
Serial Pin
Analog Pin
Control
INT
Physical Pin
Port Pin
Pin function
Interrupt Pin
PWM Pin
Port Power
bq

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

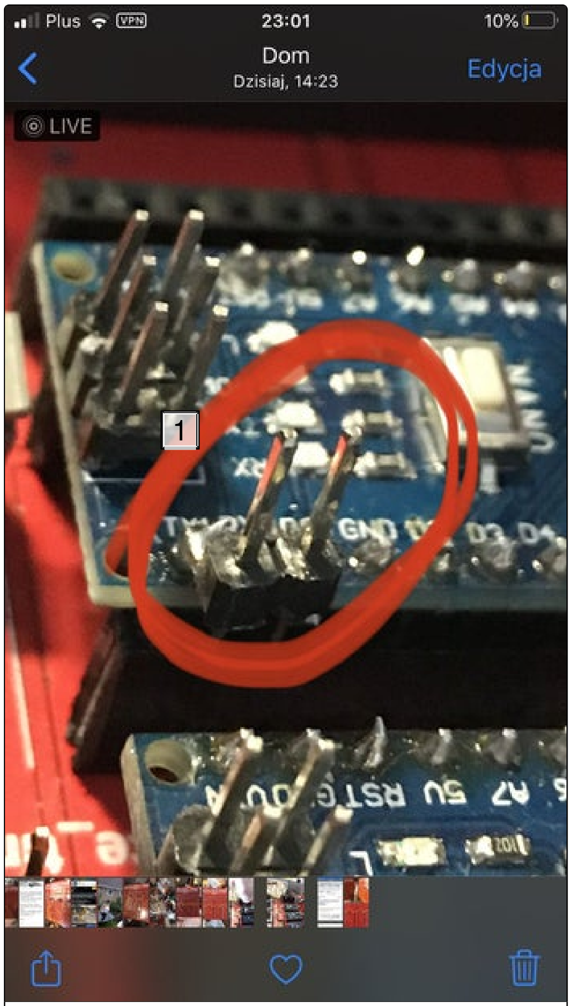
Arduino	Raspberry's GPIO pin
1	12
2	16
3	20
4	22
5	6
6	13
7	19
8	26



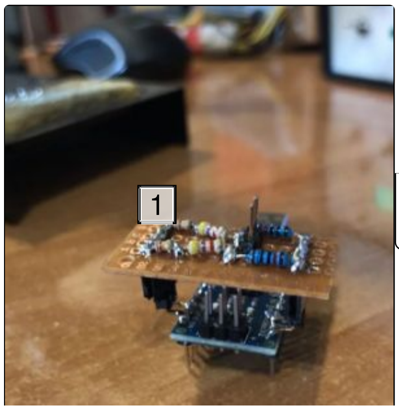
1. logic level converter



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



1. additional gold-pins on UART



1. voltage divider PCB

Step 3: Login Into Raspberry Via SSH and Downloading 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 and error after entering first command you probably have to install git from apt.

Use command: sudo apt install git

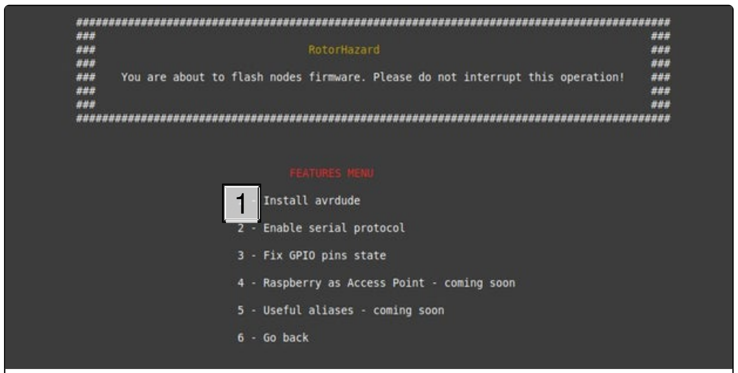
Step 4: Prepare Raspbian OS

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
```

1. enter '1' and '2' before using rest of the software

Step 5: Use Downloaded Software

After rebooting use downloaded software.

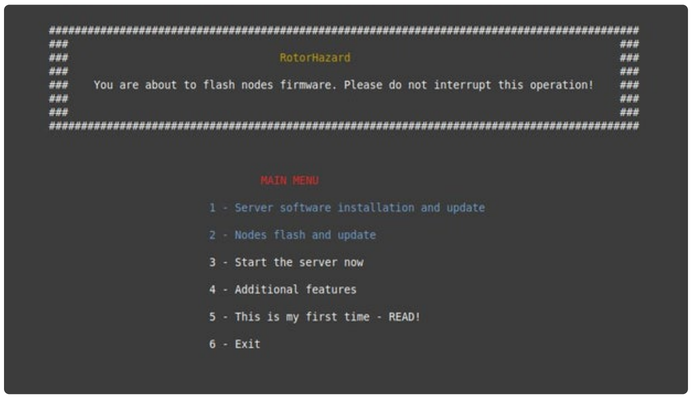
python RH-ota/update.py

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

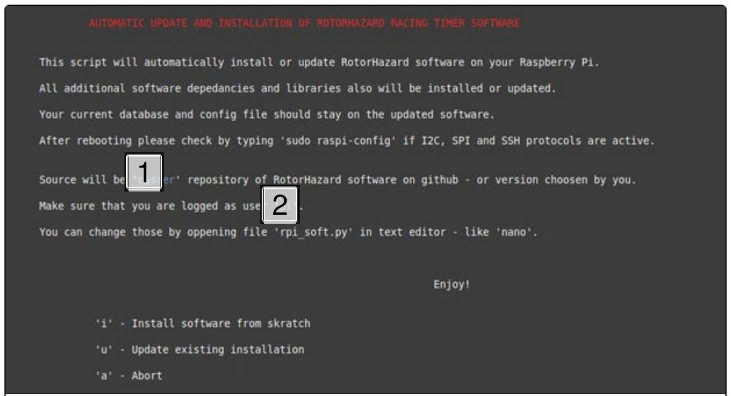
If you want to flash firmware on Arduinos enter point 2.

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
```



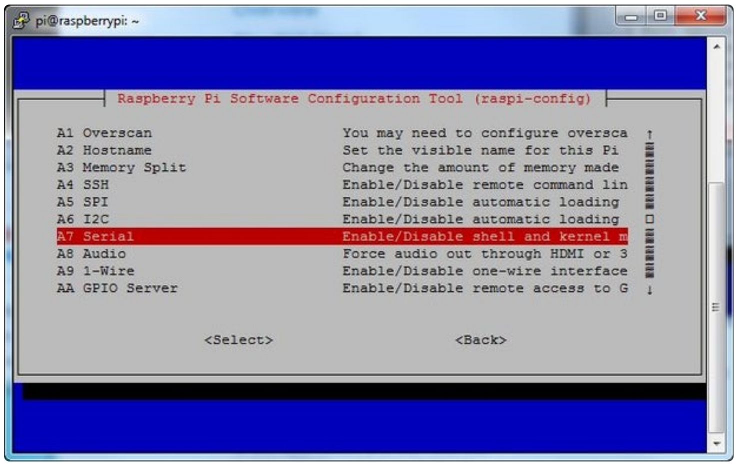
```
        AUTOMATIC UPDATE AND INSTALLATION OF ROTORHAZARD RACING TIMER SOFTWARE

    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 'master' repository of RotorHazard software on github - or version choosen by you.

    Make sure that you are logged as user 'pi'.

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


                                          Enjoy!


            'i' - Install software from skratch

            'u' - Update existing installation

            'a' - Abort
```

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

Step 6: Things Worth Mentioning / Troubleshoothing

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'
- Updating server software takes long time - when server is being updated, the Raspberry itself is being updated as well. If you haven't update Raspbian (Raspberry's OS) for some time it can take up to 20 minutes. It 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 'sudo cd /dev' and than 'ls -l' and look for ttyS0. It should look as on attached image



```
pi@raspberrypi: ~

Raspberry Pi Software Configuration Tool (raspi-config)

A1 Overscan                    You may need to configure oversca
A2 Hostname                    Set the visible name for this Pi
A3 Memory Split                Change the amount of memory made
A4 SSH                         Enable/Disable remote command lin
A5 SPI                         Enable/Disable automatic loading
A6 I2C                         Enable/Disable automatic loading
A7 Serial                      Enable/Disable shell and kernel m
A8 Audio                       Force audio out through HDMI or 3
A9 1-Wire                      Enable/Disable one-wire interface
AA GPIO Server                 Enable/Disable remote access to G

            <Select>                       <Back>
```

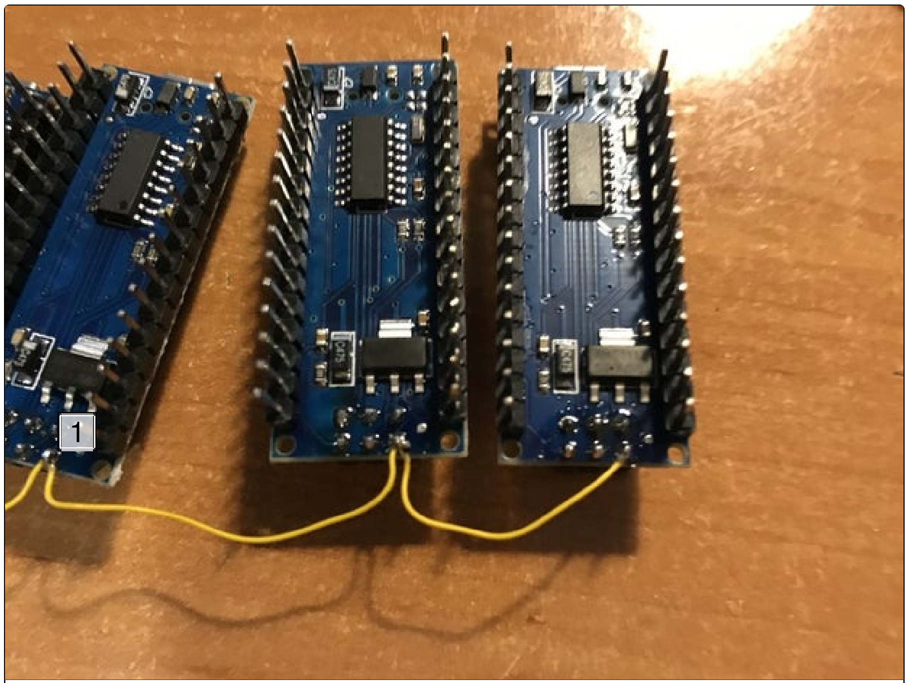
Step 7: Things I Had to Gone Through / History of Development

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 could be used to changing the channels of the 5808 receivers. I tried to use SoftwareSPI on Arduinos but it required changing pins assignment, more wireing 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.

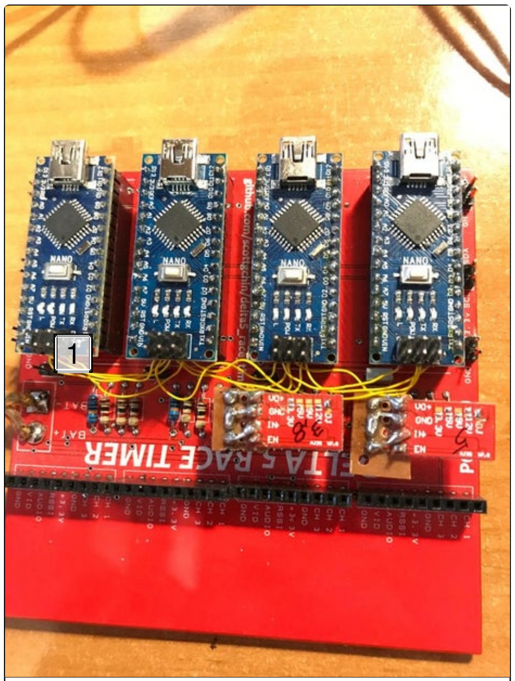
I than realized that Raspberry has an UART line and that Arduino normally can be very easily programmed with this protocol. Basically every time Arduino is connected via USB it uses UART. I have connected the dots and after some troubleshooting process - enabling the UART on the Pi is not that obvious - 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. 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 advantage of the way that this program operates right now is that 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 occures 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.

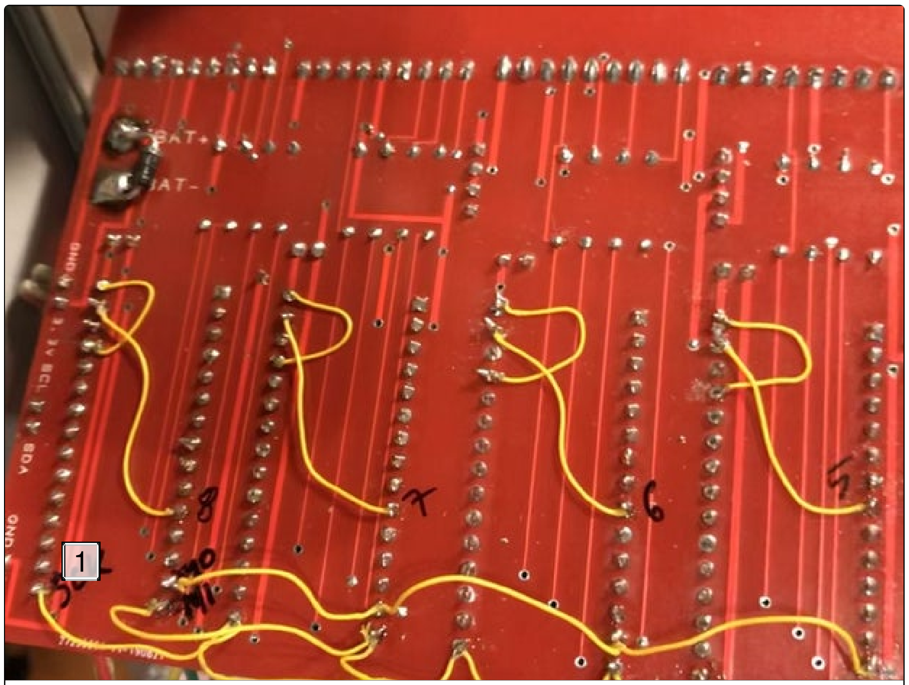
At the end I combined two separate programs so now you can update both - Raspberry's server and Arduino-based nodes using same program. Some additional features will come later.



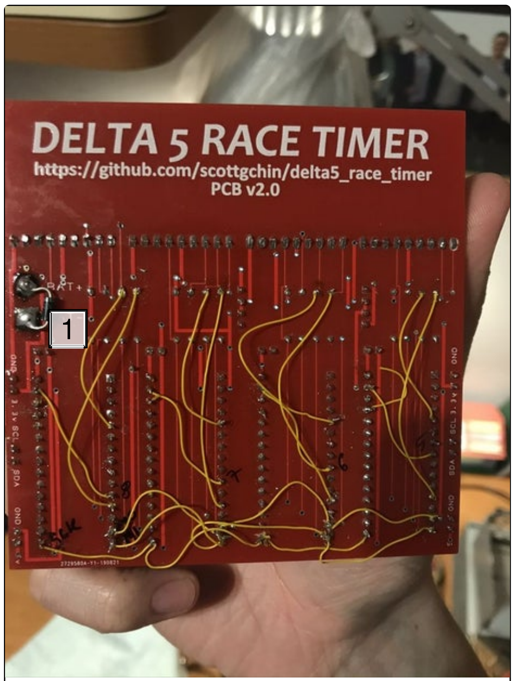
1. first attempts



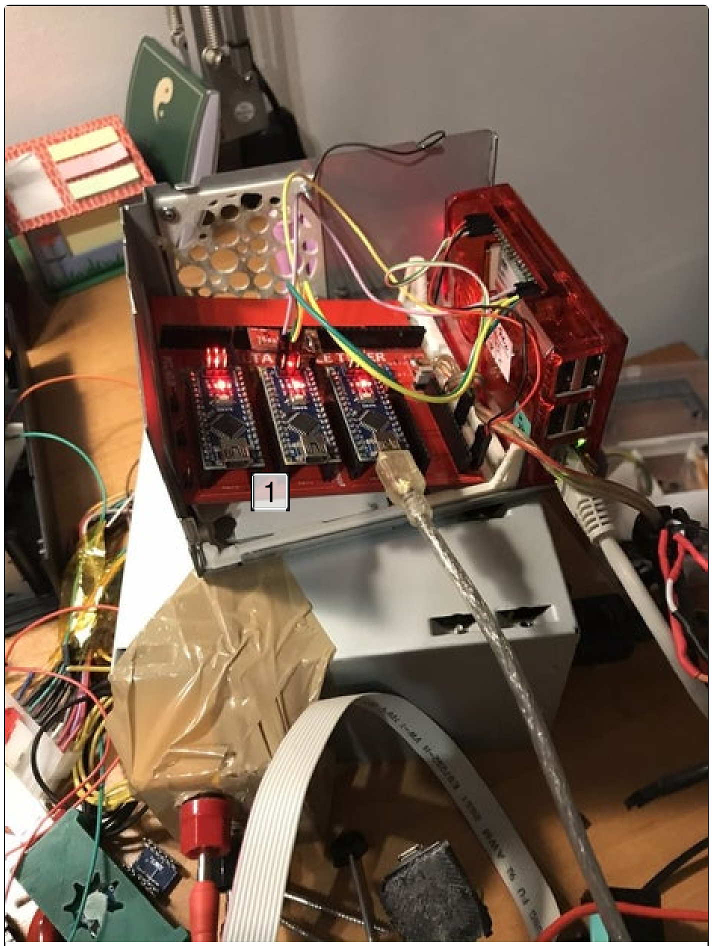
1. SPI lines connected



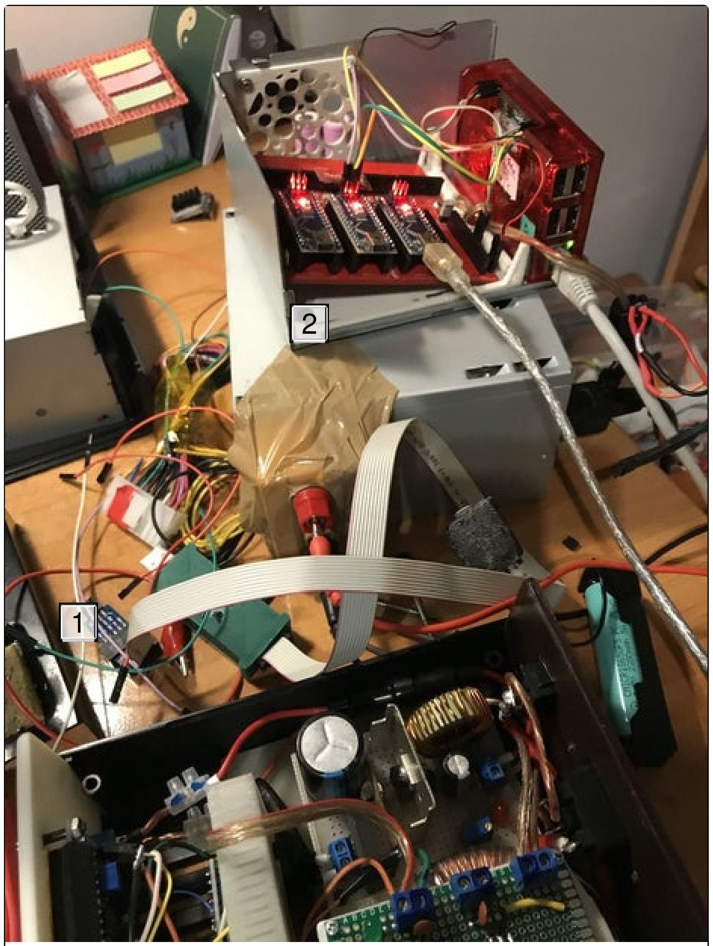
1. SPI lines



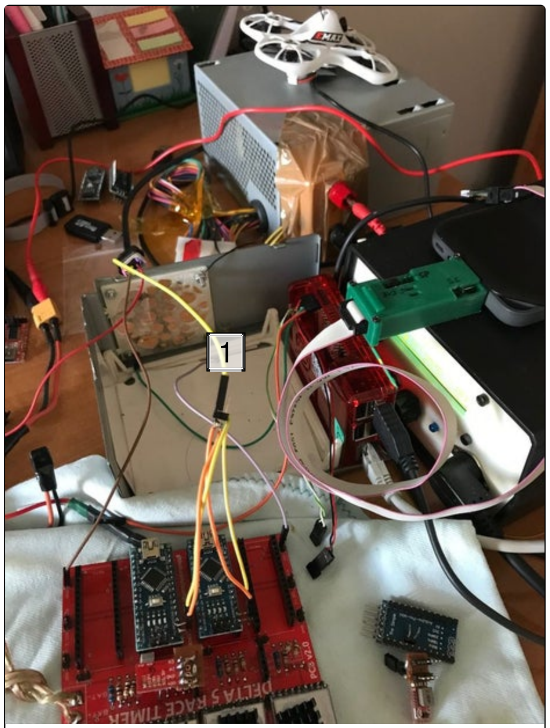
1. big mess



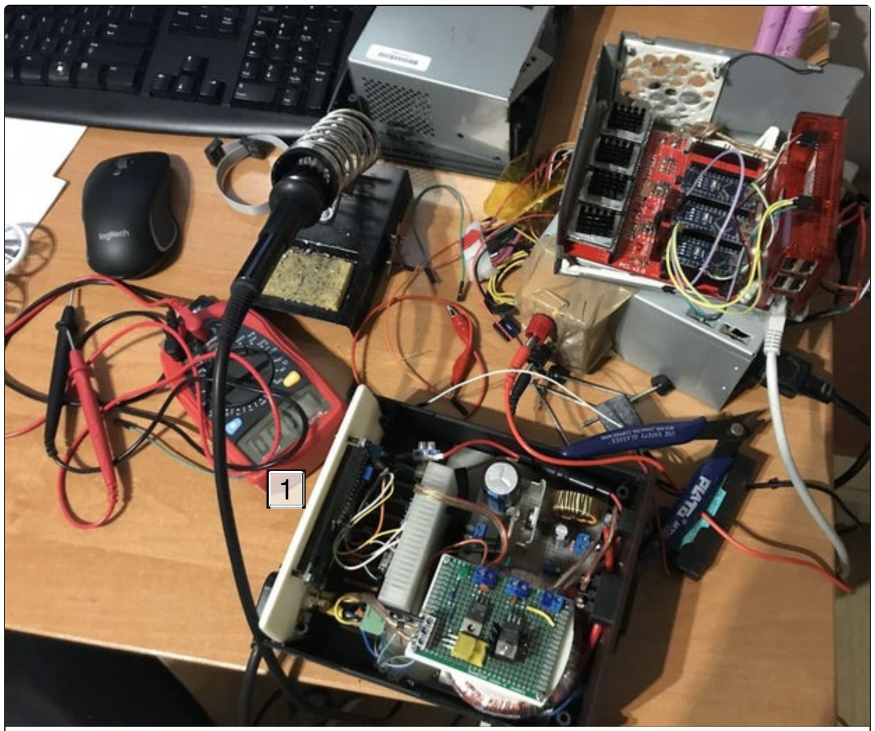
1. test bench



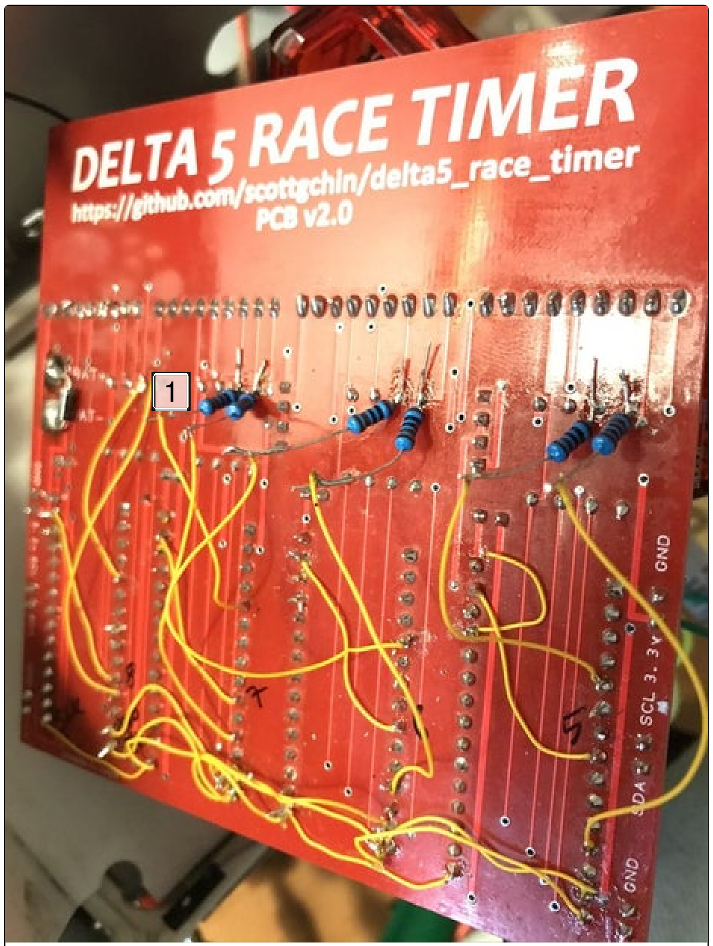
1. DIY soldering station
2. test bench



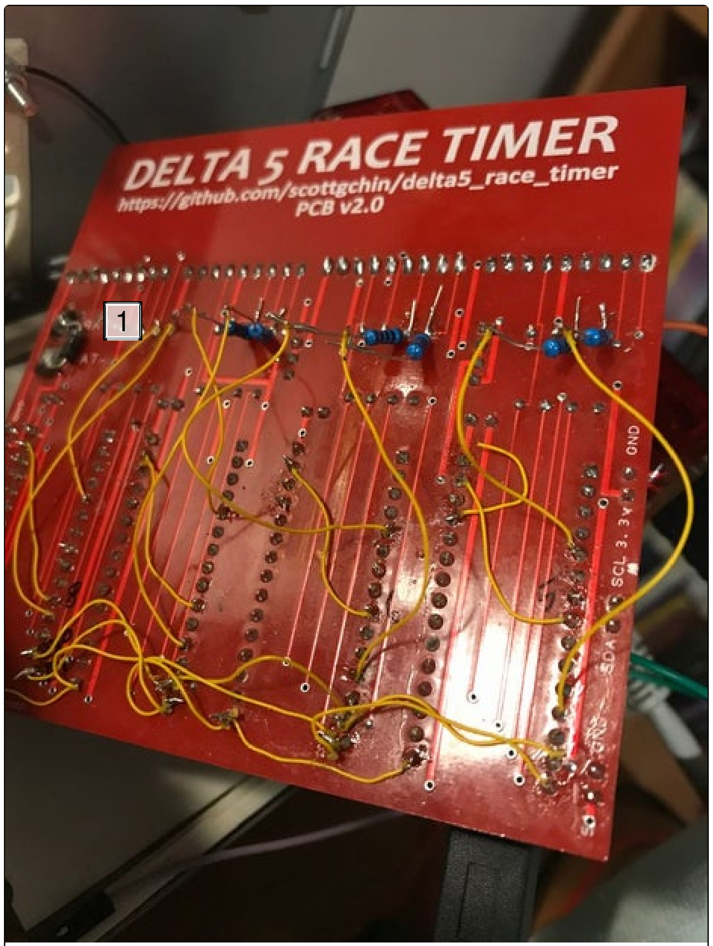
1. test bench



1. DIY soldering station



1. turning point - too much wires and changes



1. testing different pins for SPI

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.

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

Thanks for sharing :)

Thanks :) This instructable will be polished soon.