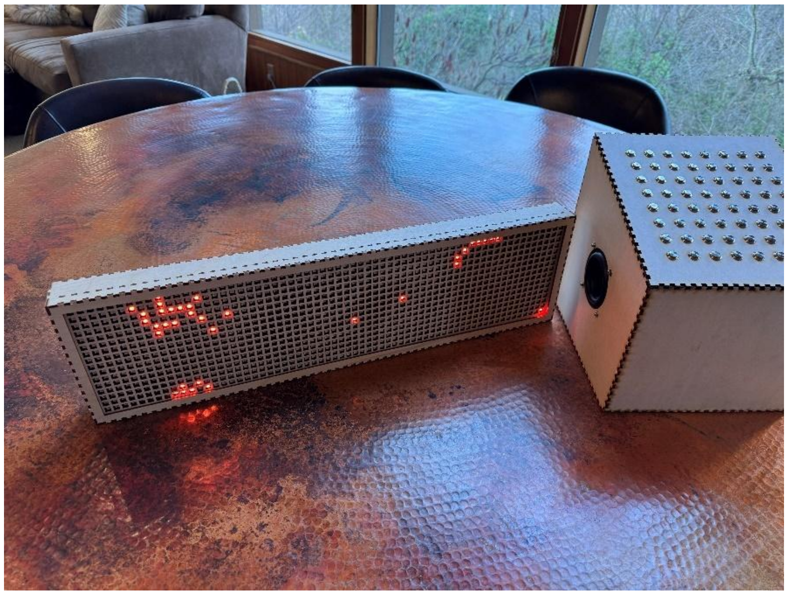
Network IV Rebooted

Andrew Seawright

Introduction

This is the user's guide and notes for the Network IV Rebooted system.



Operating Network IV Rebooted

1. Connect the network IV display power and USB. Turn on the audio amplifier (switch on back). Optionally connect an HDMI monitor and USB keyboard.

   The display USB connects in the back to the top USB. The display power (5V) is through the PowerPole connector (red+black). The HDMI, if connected, is directly to the micro-HDMI on top edge of the raspberry pi board. This is with a HDMI-to-micro-HDMI routed through the back "handle" opening. The optional keyboard can connect to the back bottom USB.

2. Plug in the pedestal unit to AC power. Wait for the system to boot up.

3. Using the HDMI monitor and keyboard, or ssh into the Raspberry Pi computer from another computer. The computer should be named "nw4". To login if needed, the

username is “pi” and the password is “raspberry”

4. Go to the network_iv directory:

```
cd network_iv
```

5. Source the setup settings file:

```
source setup.sh
```

This will setup some environment variables that specify which devices to use for the display and audio and set some other flags.

6. Run the nova emulator:

```
./nova
```

this will start the emulation which will load the Network IV program data, adjust settings, and start the program. There is a nova.ini file described below which is automatically loaded. At this point Network IV should be running.

7. Adjust audio volume (knob on amplifier in back)

Files

nova.ini	Nova startup file. Loads, setup, and run Network IV on startup. See default contents below.
nova	Nova executable (link to)
setup.sh	Set up file. Source this to setup device variables and flags before running ./nova
clear.sh	Source this to clear the display and quiet the audio. For example, if you broke out of the emulation and oscillators are running.
network_iv.tape.data.do	Network IV image converted from paper tape expressed as a series of SIMH memory deposit command address value pairs

gamelan.do	Gamelan pitch table as memory deposit commands

Nova.ini

The default contents of the **nova.ini** is the following:

```
do network_iv.tape.data.do

echo
echo ********************************
echo *** Network IV Program Loaded ***
echo ********************************
echo

do gamelan.do

d PC 00400
#break 02000
set throttle 150k
run
```

This basically loads the Network IV data, sets the PC to the starting address 00400, throttles the execution speed to a realistic value, and starts the program.

You can comment out some of the lines if you want to do start nova without executing stuff. For example, set breakpoints, single step, start from a different location.

If you comment out the "run" command in the .ini, then you will get a "sim>" prompt after startup and you can enter interactive commands.

Stop / Exit NOVA while running

Hitting **ctrl-e** will break out of the SIMH NOVA emulation to a SIMH "sim>" prompt. You can then inspect the system state, set breakpoints etc. See SIMH NOVA documentation (pointer below). To go back to linux you can type exit or quit.

Network IV SIMH Nova Details

The SIMH code is under the directory simh inside network_iv (/home/pi/network_iv/simh)

The NOVA source code (part of simh) is found in the directory:

```
        /home/pi/network_iv/simh/NOVA
```

The Network IV specific code that handles the IO to device 76 (octal) is the file:

```
        /home/pi/network_iv/simh/NOVA/network_iv.c
```

Other than a couple of small other changes this is the main addition to SIMH NOVA for Network IV.

To recompile a new "nova" executable do this:

```
        cd /home/pi/network_iv/simh
        make nova
```

For example, you would do this after making any changes to network_iv.c.

The diffs to SIMH NOVA are here:
https://github.com/NRGN2ART/Network-IV/blob/main/Rebooted/simh/diffs.txt

SIMH NOVA Documentation

The SIMH NOVA documentation can be found at this link:

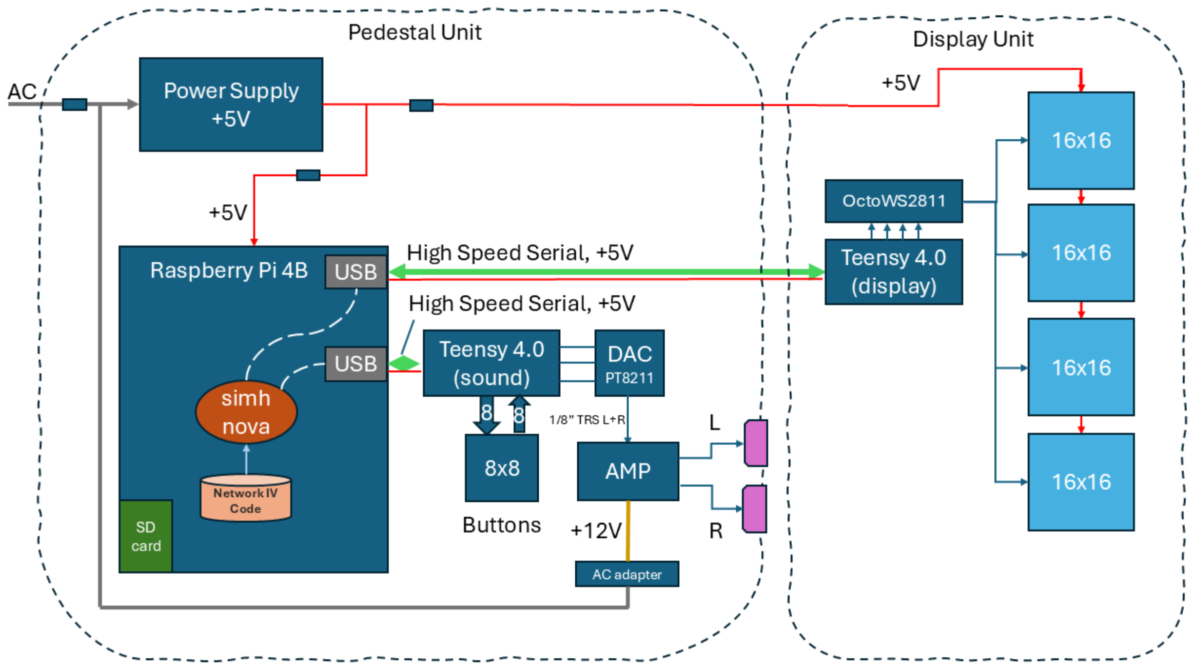
- https://simh.trailing-edge.com/pdf/nova_doc.pdf

Network IV Memory Map

00000	Program data and variables
00400	Start address – main program
02000	Interrupt handler
04000	Pitch list
04400	Filter list
05000	Envelope list
05700	Modify list
16000	Test Program 1 LTUNE the display code handling is tuned to the main Network IV program. To run this test source norow.csh before to change the display handling to work with this test.

	* cont after halt
16050	Test Program 2 STUNE
16100	Test Program 3 RTUNE

Network IV Rebooted Block Diagram



Github Location

The github for Network IV and Network IV rebooted is:
https://github.com/NRGN2ART/Network-IV/tree/main

Tape data	Raw: https://github.com/NRGN2ART/Network-IV/blob/main/Code/tape_nw4_final_program_obj_read1.txt As SIMH mem deposit: https://github.com/NRGN2ART/Network-IV/blob/main/Code/network_iv.tape.data.do
Annotated Program Listing Scan	https://github.com/NRGN2ART/Network-IV/blob/main/Code/NetworkIVprogramlisting.pdf
Engineering Drawings	https://github.com/NRGN2ART/Network-IV/blob/main/Documentation/NetworkIVdocumentation1.pdf

Information	https://github.com/NRGN2ART/Network-IV/blob/main/Documentation/NetworkIVdocumentation2.pdf
SIMH stuff	https://github.com/NRGN2ART/Network-IV/tree/main/Rebooted/simh
Laser Cut Files	https://github.com/NRGN2ART/Network-IV/tree/main/Rebooted/LaserCut

Display System

The Network IV Rebooted display system is implemented using a Teensy 4.0 microcontroller. See https://www.pjrc.com/store/teensy40.html.

The Teensy LED display driver board is used with the teensy:
https://www.pjrc.com/store/octo28_adaptor.html

The displays used are: https://www.amazon.com/gp/product/B07PB2P81N/?th=1

The display Teensy, driver board, and displays are housed in the display unit.

The most recent Teensy code for the display is "sketch_network_iv_display_v2.ino". See

https://github.com/NRGN2ART/Network-IV/blob/main/Rebooted/Teensy/sketch_network_iv_display_v2.ino

The Teensy is programed via the Arduino IDE setup with the Teensy extensions.

The Display Teensy is connected by USB to the Raspberry Pi. It will appear as "tty" device on the Raspberry Pi named something like **/dev/ttyACM1**. The network_iv.c code will open the display device set in the **NW4_DISPLAY_DEVICE** environment variable.

Basically the device is opened and commands sent by linux write() system commands. Linux read() commands get status back.

The device can be tested independently by using a terminal program, for example, in the Arduino IDE. There is a utility program "display" that can send display commands from the Raspberry Pi linux prompt described later.

The following table describes the Teensy display controller commands

Display Command	Description
?	Get status. Send (write) "?\n" then read back status info.
c	Clears the display – all lamps off

l <R> <C>	Light a single “lamp” at row <R> and column <C>. For example, sending “l 2 25\n” will cause lamp at row 2 column 25 to turn on. Column 0 is the leftmost column. Row 0 is the topmost row.
r <R> <HEXDATA>	Send display state for whole row <R> as a 64bit hex data string of 16 hex characters. This is much higher performance and what the network_iv.c does by default to update the display by row by row. For example, sending the command: “r 3 4000800000000001\n” would turn on lamps: row 3 column 1, row 3 column 16, and, row 3 column 63. All other lights on row 3 are cleared.

Sound / Input System

The sound and input system are implemented using another Teensy 4.0. This Teensy is inside the pedestal unit.

The most current Teensy sound program is “sketch_network_iv_sound_v6.ino”. See

https://github.com/NRGN2ART/Network-IV/blob/main/Rebooted/Teensy/sketch_network_iv_sound_v6.ino

The sound/input Teensy uses a PT8211 DAC board. See: https://www.pjrc.com/store/pt8211_kit.html

The input buttons are scanned using eight IO pins set as inputs with pull up (sense) and eight output as open drain.

The IO pins used for the DAC and the button scanning are defined in the Teensy sound program.

The Network IV I/O code connects to the sound system via the device named in the **NW4_SOUND_DEVICE** environment variable. The sound device is typically /dev/ttyACM0.

The following is a description of the commands recognized by the Sound / Input Teensy:

Command	**Description**
?	Get status

c	Clear sound state
i <SW>	Get button data as a vector for switch bank <SW> where SW is 0,1,2,3. Gets 16 bits (two rows of 8x8 layout) of switch (button) data. SW is the interpretation of switch bank in the Network IV program. Return data is of the form B:XXXX where XXXX are hex digits representing the state of 16 buttons
r <N>	Trigger resonator <N> where <N> is 0 to 31
s <N> <VAL>	Set sound parameter <N> to <VAL>. <N> mapping: 0: OSC 1 1: OSC 2 2: OSC 3 3: OSC 4 The oscillator values converted in the value_to_freq() function where value 101 is note A at 220Hz and each increment is a fifth of a half step change in frequency. This value to pitch is reverse engineered from the Gamelan table which gives values of notes (mapping between values and actual note pitches). 5: filter 1 7: filter 2 Filter parameters are set by Network IV but not yet used 14: VCA1 20: VCA2 24: VCA3 Values to gain by VAL/1024.0
d <N>	Set debug level to <N>

Button Matrix

Teensy IO #	IO Mode	#define Name	Wire color	Button Matrix
0	INPUT_PULLUP	BUTTON_IN_0	Brown	COL1
1	INPUT_PULLUP	BUTTON_IN_1	Red	COL2
2	INPUT_PULLUP	BUTTON_IN_2	Orange	COL3
3	INPUT_PULLUP	BUTTON_IN_3	Yellow	COL4
4	INPUT_PULLUP	BUTTON_IN_4	Green	COL5
5	INPUT_PULLUP	BUTTON_IN_5	Blue	COL6
6	INPUT_PULLUP	BUTTON_IN_6	Violet	COL7
8 (skip 7)	INPUT_PULLUP	BUTTON_IN_7	Gray	COL8
23	OUTPUT_OPENDRAIN	BUTTON_OUT_0	Brown	ROW1
22	OUTPUT_OPENDRAIN	BUTTON_OUT_1	Red	ROW2
19	OUTPUT_OPENDRAIN	BUTTON_OUT_2	Orange	ROW3
18	OUTPUT_OPENDRAIN	BUTTON_OUT_3	Yellow	ROW4
17	OUTPUT_OPENDRAIN	BUTTON_OUT_4	Green	ROW5
16	OUTPUT_OPENDRAIN	BUTTON_OUT_5	Blue	ROW6
15	OUTPUT_OPENDRAIN	BUTTON_OUT_6	Violet	ROW7
14	OUTPUT_OPENDRAIN	BUTTON_OUT_7	Gray	ROW8

Buttons are scanned by row. Each loop one row is scanned, by diving a row line to ground (open drain output). The column lines are pulled up by default. Thus, any buttons pressed in that selected row are read in as low. Button state is kept in an array of eight int for the eight rows.

Button state queries return the button state by Network IV switch bank numbers (0-3) organized as pairs of physical rows (16 bits each). See the original Network IV program listing fist page for the mapping diagram – that picture has most significant bits as # 0 in the Nova convention.

Utilities

In the *utility* directory under network_iv are two utilities:

1. *sound* to interact with the Sound/Input Teensy
2. *display* to interact with the Display Teensy

Sound Utility

The *sound* utility is compiled from sound.c. It takes command arguments and sends them as commands to the sound device. A ‘,’ comma (must be separated by whitespace on both sides) is converted to a ‘\n’ when sending.

Examples:

./sound r 5 , ? ,	Trigger resonator 5 and get status
./sound c , ? ,	Clear sound and get status
./sound d 1 , ? ,	Set debug level 1 and get status
./sound s 0 101 , ? ,	Set OSC1 to 101 (220Hz) and get status
./sound i 0 ,	Get input buttons bank SW 0

Display Utility

The *display* utility is like the *sound* utility. It sends commands to the display device.

Examples:

./display c	Clear the display
./display l 5 25	Turn on lamp row 5 col 25

MIDI?

Mapping the Network IV audio to MIDI is a future direction. The current thinking is to add environment variables to enable MIDI. One to enable MIDI and specify the MIDI device. Another to setup which instruments map to MIDI. This would allow a mixture of some audio from the Teensy and some from MIDI.

NW4_MIDI_DEVICE

NW4_MIDI_CONFIG_FILE

For example, we might map one instrument to a particular MIDI channel. Other parameters would describe how to map the notes etc.

Raspberry Pi Initial Setup

The initial Raspberry Pi SD card image was created using the Raspberry Pi imager utility: https://www.raspberrypi.com/news/raspberry-pi-imager-imaging-utility/

Choose the device as Raspberry Pi 4

Choose the OS option: Raspberry Pi OS (64-bit) (Recommended)

You can also specify the initial Wi-Fi setup for the Raspberry Pi

After imaging the SD card and booting up the Pi, you can change the boot mode to boot to console

At this point you would need to reinstall the SIMH and Network IV stuff.

Getting SIMH

This is how to get SIMH via git. This assumes a newly created SD card.

1. Make a network_iv dir in the /home/pi (home) directory

   ```
   cd ~
   mkdir network_iv
   cd network_iv
   ```

2. In the above directory do:

   ```
   git clone https://github.com/open-simh/simh.git
   ```

Building SIMH nova

```
cd network_iv/simh
make nova
```

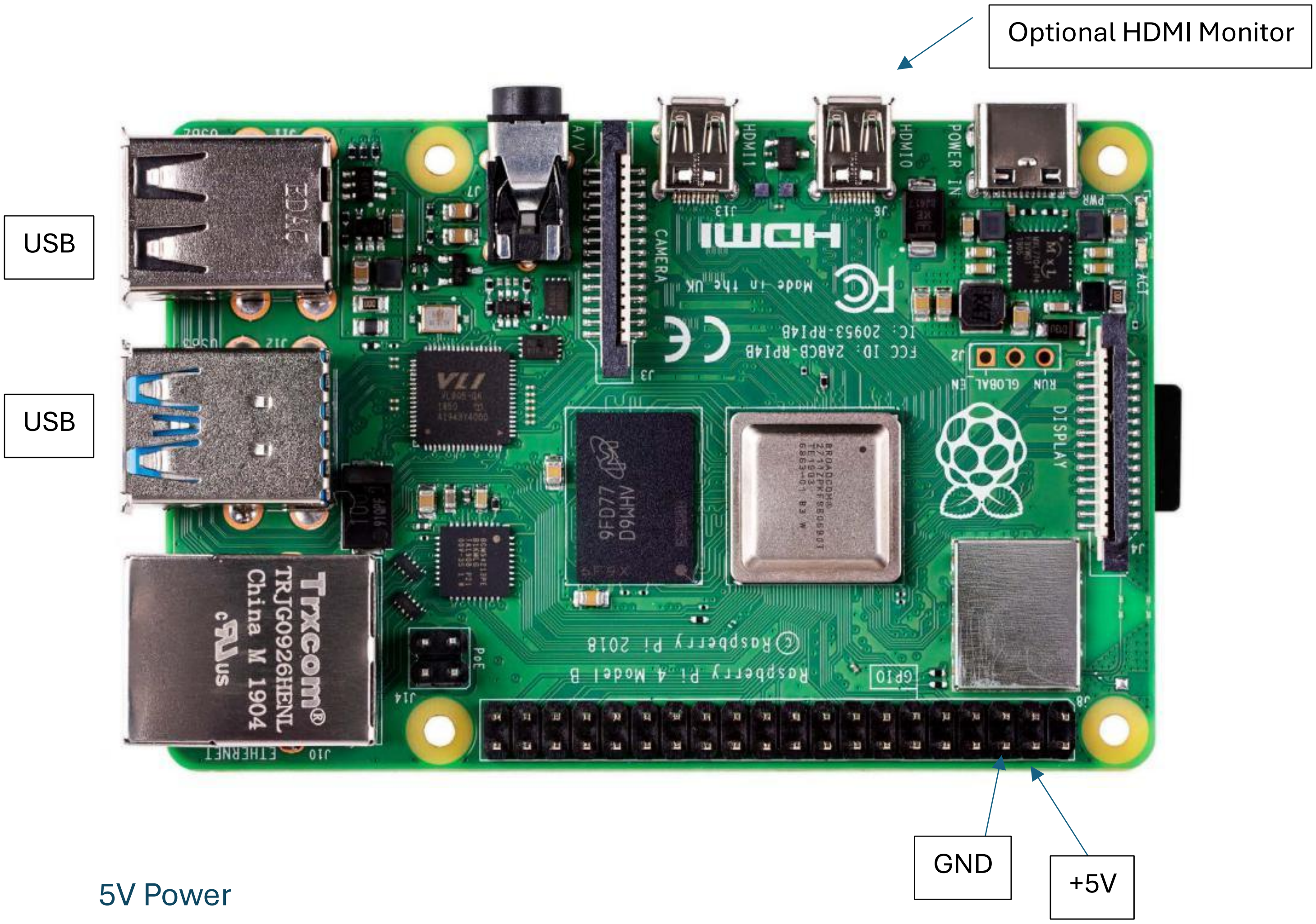
Adding network IV code

The SIMH NOVA additions for Network IV are described here:

https://github.com/NRGN2ART/Network-IV/tree/main/Rebooted/simh

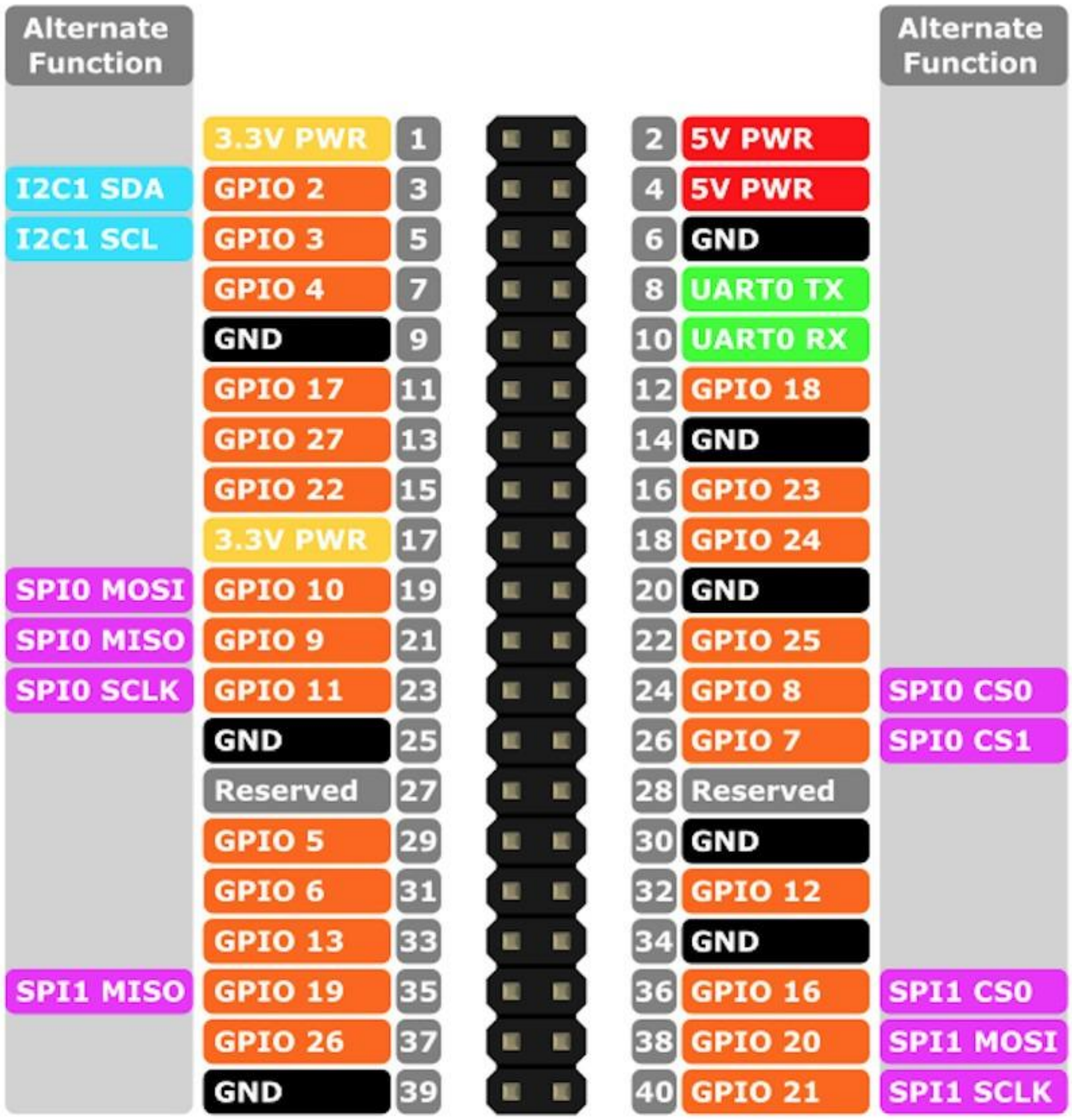
Raspberry PI 4B

Below is a picture of the Raspberry Pi 4B in the orientation mounted.



5V Power

The power supplied <u>to</u> the Network IV Raspberry Pi is through the 40 pin connector **pins 4 +5V** and **pin 6 GND**. Be careful if these connections to the power supply are disconnected, they are put back in the proper locations.



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	UART0 TX	
	GND	9	10	UART0 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	SPI0 CS0
	GND	25	26	GPIO 7	SPI0 CS1
	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	SPI1 CS0
	GPIO 26	37	38	GPIO 20	SPI1 MOSI
	GND	39	40	GPIO 21	SPI1 SCLK