



# Arduino, Elixir and Nerves

A Deep Dive into the Firmata Protocol



# Firmata

Firmata is a protocol for communicating with microcontrollers from software on a computer (host) based on the MIDI protocol



# Firmata -> MIDI

0xD0 - 0xDF + 1 byte

MIDI: Channel + Pressure

**Firmata:** Report digital port + enable/disable

# Outgoing Messages

## I2C Request Message

Start Sysex	Sysex Command	Address LSB	Address MSB	Data		End Sysex
0xF0	0x76	7-bits	7-bits	7-bits LSB	7-bits MSB	0xF7

slave address (MSB) + read/write and address mode bits

- {7: always 0} + {6: reserved} + {5: address mode, 1 means 10-bit mode} +
- {4-3: read/write, 00 => write, 01 => read once, 10 => read continuously, 11 => stop reading} +
- {2-0: slave address MSB in 10-bit mode, not used in 7-bit mode}

## Servo Config Message

Start Sysex	Sysex Command	MinPulse LSB	MinPulse MSB	MaxPulse LSB	MaxPulse MSB	Angle LSB	Angle MSB	End Sysex
0xF0	0x70	7-bits	7-bits	7-bits	7-bits	7-bits	7-bits	0xF7

## I2C Config Message

Start Sysex	Sysex Command	Delay LSB	Delay MSB	End Sysex
0xF0	0x78	7-bits	7-bits	0xF7

## Extended Analog Message

Start Sysex	Sysex Command	Pin	Value	End Sysex
0xF0	0x6F	7-bits	1..3 lsb->msb 7-bits	0xF7

The value can be 1 up to 3 7-bit values

## Sampling Interval Message

Start Sysex	Sysex Command	Interval LSB	Interval MSB	End Sysex
0xF0	0x7A	7-bits	7-bits	0xF7

## Modes

Input = 0  
Output = 1  
Analog = 2  
PWM = 3  
Servo = 4  
Shift = 5  
I2C = 6

## Query Capability

Start Sysex	Sysex Command	End Sysex
0xF0	0x6B	0xF7

## Analog Mapping Query

Start Sysex	Sysex Command	End Sysex
0xF0	0x69	0xF7

## Query Firmware Name and Version

Start Sysex	Sysex Command	End Sysex
0xF0	0x79	0xF7

## Pin State Query Message

Start Sysex	Sysex Command	Pin	End Sysex
0xF0	0x6D	7-bits	0xF7

## Set Analog Output

Analog Output	lsb	msb
0xE0 – 0xEF	7-bits	7-bits

B1110xxxx - Where xxxx the digital pin (max 16 pins)

## Set Digital Output

Digital Output	lsb	msb
0x90 – 0x9F	7-bits	7-bits

B1001xxxx - Where xxxx the digital port (max 16 ports)

## Set Pin Mode

Set Pin Mode	Pin #	Pin State
0xF4	7-bit (0-127)	mode

## Query Version

Request Version
0xF9

## Reset

Reset
0xFF

## Report Digital Port

Toggle Digital	Enable/Disable
0xD0 – 0xDF	1-bit (B0000000x)

B1101xxxx - Where xxxx the digital port (max 16 ports)

Digital port = 8 pins

## Report Analog Pin

Toggle Analog	Enable/Disable
0xC0 – 0xCF	1-bit (B0000000x)

B1100xxxx - Where xxxx the analog pin (max 16 pins)

## Sysex Messages

Outgoing from Host

## Incoming Messages

### General Sysex Message

Start Sysex	Sysex Command	String	End Sysex
0xF0	0x00 – 0x7F	0..1024 7-bit split chars	0xF7

### Firmware Message

Start Sysex	Sysex Command	Major Version	Minor Version	Firmware name	End Sysex
0xF0	0x79	7-bits	7-bits	0..1022 7-bit split chars	0xF7

### Analog Mapping Message

Start Sysex	Sysex Command	Analog Channel	End Sysex
0xF0	0x6A	0..pins 7-bits	0xF7

### String Message

Start Sysex	Sysex Command	String	End Sysex
0xF0	0x70	0..1024 7-bit split chars	0xF7

### Pin State Message

Start Sysex	Sysex Command	Pin Number	Pin Mode	Pin State			End Sysex
0xF0	0x6E	7-bits	mode	7-bits LSB	7-bits Middle Byte	7-bits MSB	0xF7

Middle and MS bytes may be omitted if they have no info

### Capability Query Message

Start Sysex	Sysex Command	Pin Capabilities				End Sysex	
0xF0	0x6C	0..pins				0xF7	0xF7
		0..modes					
		7-bits mode	7-bits resolution				

### I2C Response Message

Start Sysex	Sysex Command	Address LSB	Address MSB	Register LSB	Register MSB	Data		End Sysex
0xF0	0x77	7-bits	7-bits	7-bits	7-bits	0..data 7-bits LSB 7-bits MSB		0xF7

## Sysex Messages

### Protocol Version Message

Protocol Message	Major Version	Minor Version
0xF9	7-bit	7-bit

### Analog Message

Analog Message	LSB	MSB
0xE0 – 0xEF	7-bit	7-bit

B1110xxxx -> xxxx the analog pin  
(max 16 pins)

### Digital Message

Analog Message	LSB	MSB
0xE0 – 0xEF	7-bit	7-bit

B1110xxxx -> xxxx the digital port  
(max 16 ports) Digital port = 8 pins

### Modes

Input = 0  
Output = 1  
Analog = 2  
PWM = 3  
Servo = 4  
Shift = 5  
I2C = 6

Incoming to Host



# System Exclusive (Sysex) Messages

Few message types map 1 - 1 to MIDI protocol, basically analog and digital pin IO, firmware version, etc.

Mapping to a MIDI message must match MIDI message length exactly

Sysex Messages can be any length

Sysex Messages are used prominently for Firmata functionality

Start Sysex	Sysex Command	String	End Sysex
0xF0	0x00 – 0x7F	0..1024 7-bit split chars	0xF7



# Full Disclosure

Keyvan Fatehi (<https://github.com/kfatehi>) wrote the original Elixir Firmata Client (<https://github.com/entone/firmata>) and is responsible for the majority of the architecture and parsing logic.

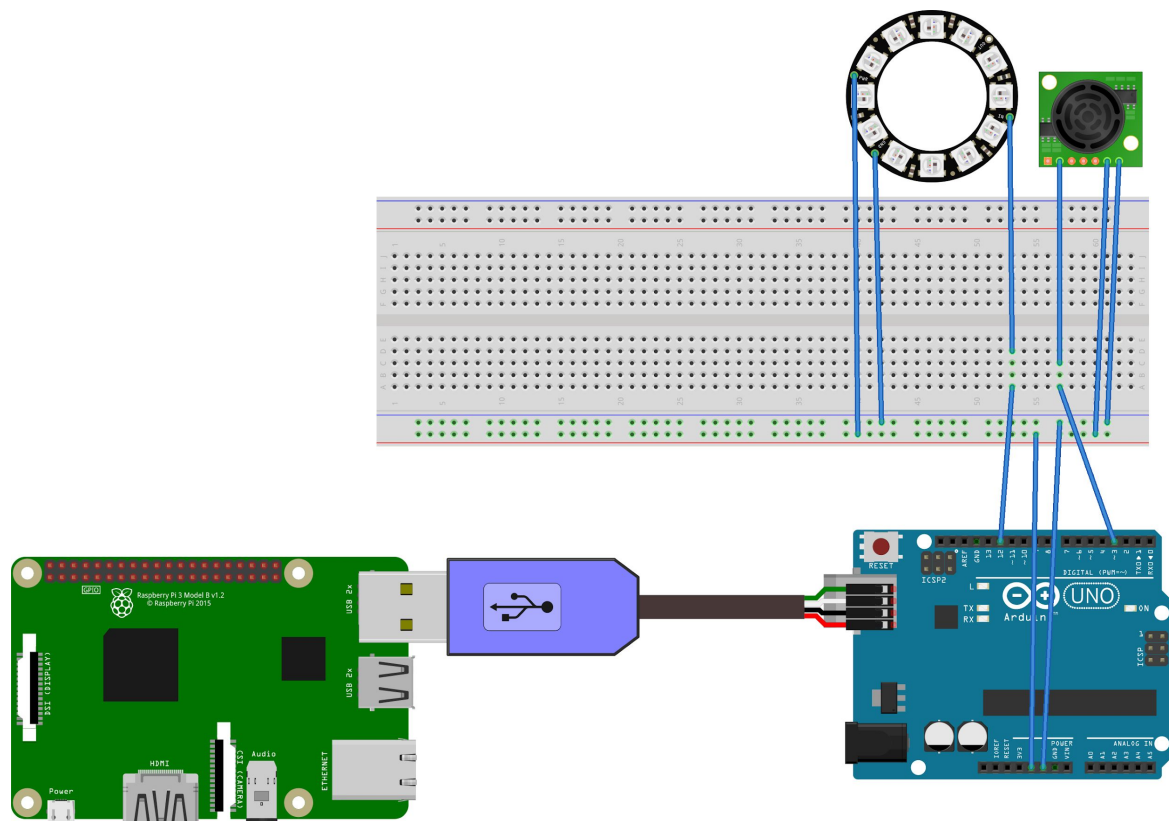


# My contributions (maintainer)

- I2C read/write
- String data
- Analog pin <--> process mapping architecture.
- Ultrasonic sensor
- NeoPixel



I2C	Analog	Digital	Servo	Custom
FirmataExample.Board				
Firmata.Board				
Nerves.UART				
Stream				
Arduino				
I2C	Analog	Digital	Servo	Custom



fritzing

Typical Firmata Architecture

```
def init({port, opts}) do
  speed = opts[:speed] || 57600
  uart_opts = [speed: speed, active: true]

  {:ok, serial} = Nerves.UART.start_link
  :ok = Nerves.UART.open(serial, port, uart_opts)

  Nerves.UART.write(serial, <<0xFF>>)
  Nerves.UART.write(serial, <<0xF9>>)

  state =
    @initial_state
    ▷ Map.put(:serial, serial)
    ▷ Map.put(:interface, opts[:interface])
  {:ok, state}
end
```

Firmata.Board.init

```
def handle_info({:nerves_uart, _port, data}, state) do
  {outbox, parser} = Enum.reduce(data, {state.outbox, state.parser}, &Firmata.Protocol.parse(&2, &1))
  Enum.each(outbox, &send(self, &1))
  {:noreply, %{state | outbox: [], parser: parser}}
end
```

Parse UART input



# Firmata.Protocol

```
defmodule Firmata.Protocol do
  use Firmata.Protocol.Mixin
  alias Firmata.Protocol.Sysex, as: Sysex

  def parse({outbox, {}}, <<@report_version>>) do
    {outbox, {:report_version}}
  end

  def parse({outbox, {:report_version}}, <<major>>) do
    {outbox, {:report_version, major}}
  end

  def parse({outbox, {:report_version, major}}, <<minor>>) do
    {[{:report_version, major, minor} | outbox ], {} }
  end

  def parse({outbox, {}}, <<@start_sysex>> = sysex) do
    {outbox, {:sysex, sysex}}
  end

  def parse({outbox, {:sysex, sysex}}, <<@end_sysex>>) do
    {[ Sysex.parse(sysex) | outbox ], {} }
  end

  def parse({outbox, {:sysex, sysex}}, byte) do
    {outbox, {:sysex, sysex << byte }}
  end

  def parse({outbox, {}}, <<byte>>) when byte in @analog_message_range do
    {outbox, {:analog_read, byte &&& 0x0F}}
  end

  def parse({outbox, {:analog_read, pin}}, <<lsb>>) do
    {outbox, {:analog_read, pin, lsb}}
  end

  def parse({outbox, {:analog_read, pin, lsb}}, <<msb>>) do
    {[{:analog_read, pin, lsb || (msb <<< 7)} | outbox], {} }
  end
end
```



# Custom Firmata Additions

```
#include <Servo.h>
#include <Wire.h>
#include <Firmata.h>
#include <NewPing.h>
#include <Adafruit_GFX.h>
#include <Adafruit_NeoMatrix.h>
#include <Adafruit_NeoPixel.h>
```

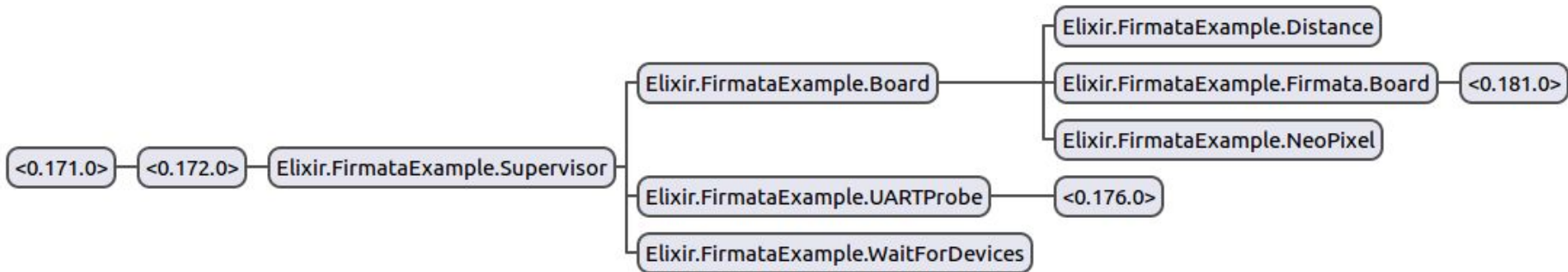
```
#include <Servo.h>
#include <Wire.h>
#include <Firmata.h>
^
^
^
^
```



# Modifying the Firmata Client

```
# custom for sonar range sensors  
@sonar_config 0x62 # configure pins to control a Ping type sonar distance device  
@sonar_data 0x63 # distance data returned  
  
#custom for neopixels  
@neopixel_register 0x74 #arg0 pin_number, arg1 num_pixels  
@neopixel_brightness 0x73 #arg0 brightness  
@neopixel 0x72 #arg0 pixel_index, arg1 red, arg2 green, arg3 blue
```

[https://github.com/entone/firmata\\_example](https://github.com/entone/firmata_example)



- cicada
- cowboy
- elixir
- gettext
- hackney
- hardware**
- hex
- iex
- inets
- interface
- kernel
- logger
- mdns
- mix
- mqtt
- nerves\_interim\_wifi
- nerves\_network\_inte
- nerves\_wpa\_supplika
- os\_mon
- plug
- ranch
- sasl
- ssdp
- ssl
- timex
- tzdata







# Thank You

@entropealab

@CRTLabs

<https://github.com/entone>

<http://code.crtlabs.org>

[https://github.com/entone/firmata\\_example](https://github.com/entone/firmata_example)