Arduino, Elixir and Nerves

A Deep Dive into the Firmata Protocol

Firmata

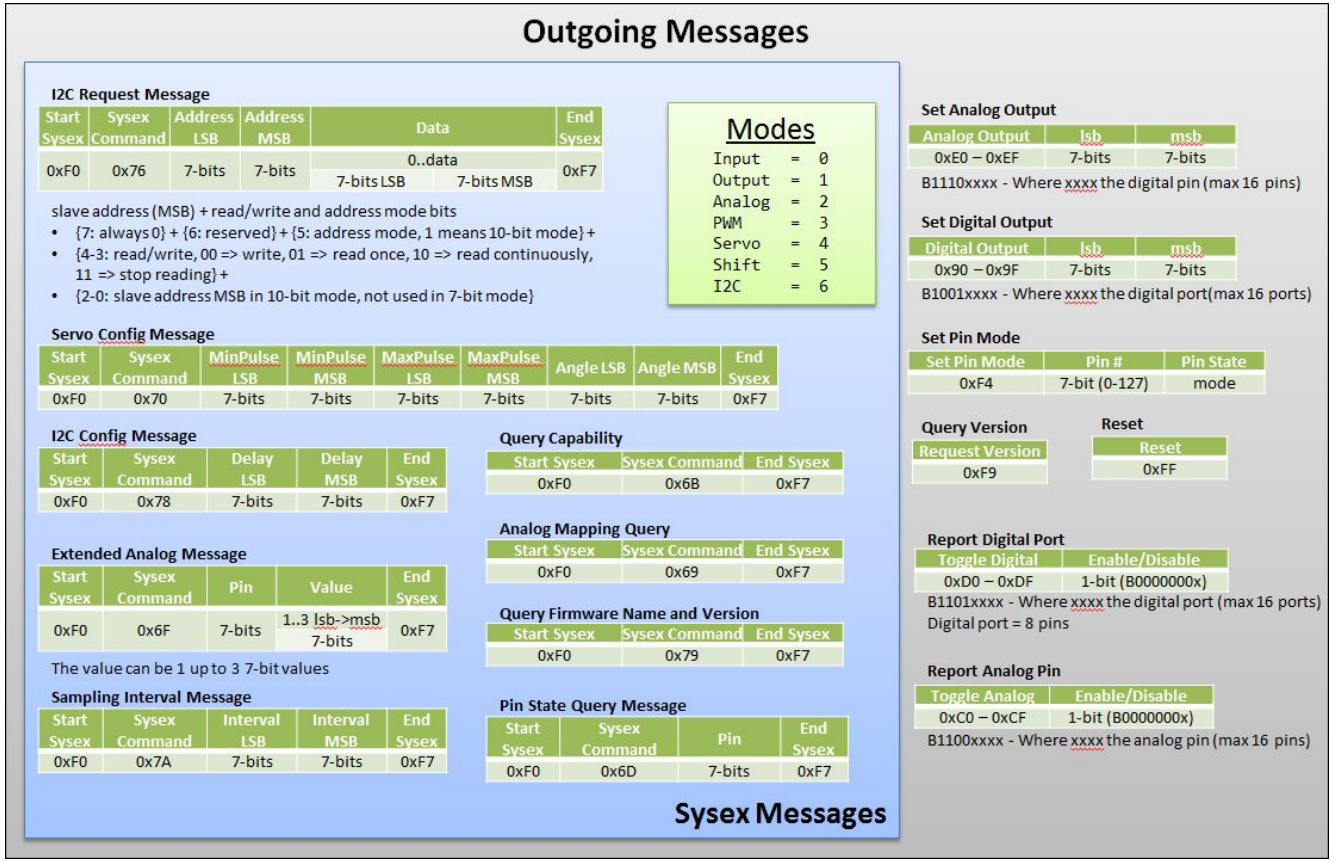
Firmata is a protocol for communicating with microcontrollers from software on a computer (host) based on the MIDI protocol

Firmata -> MIDI

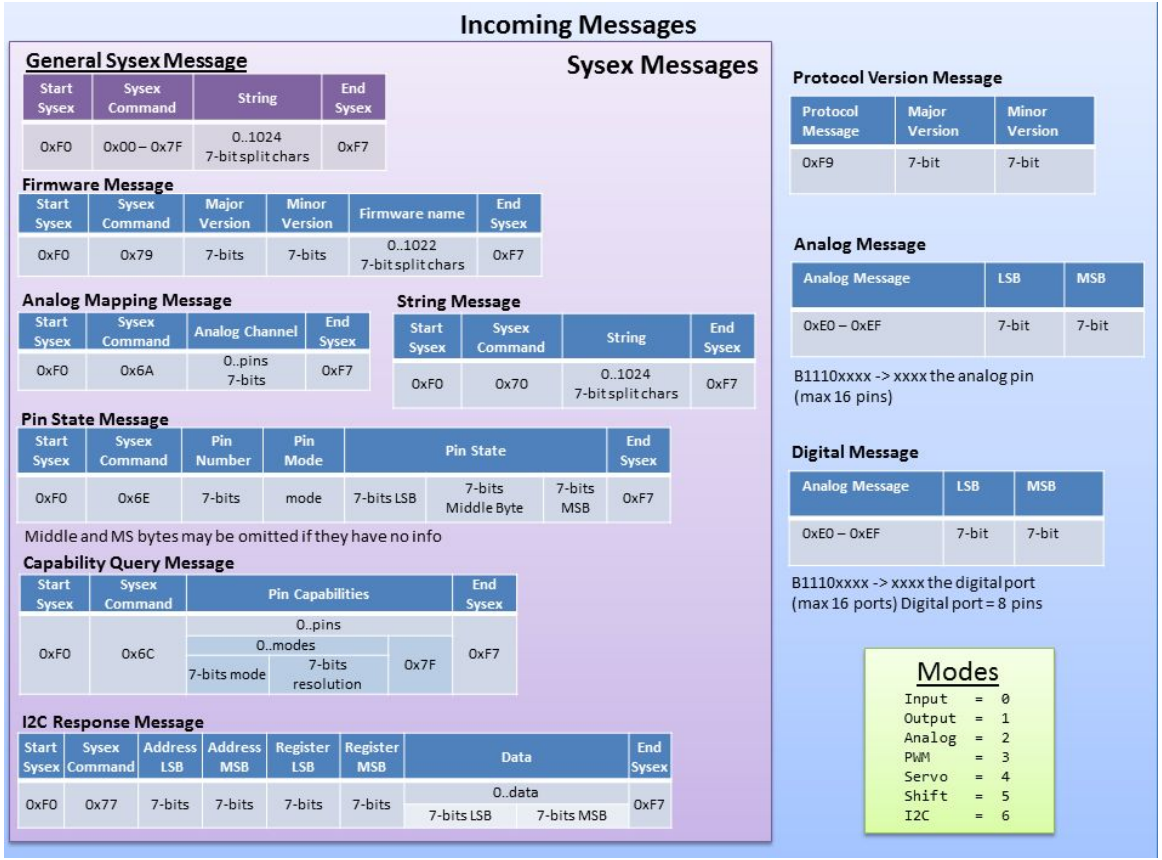
0xD0 - 0xDF + 1 byte

MIDI: Channel + Pressure

Firmata: Report digital port + enable/disable



Outgoing from Host



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.

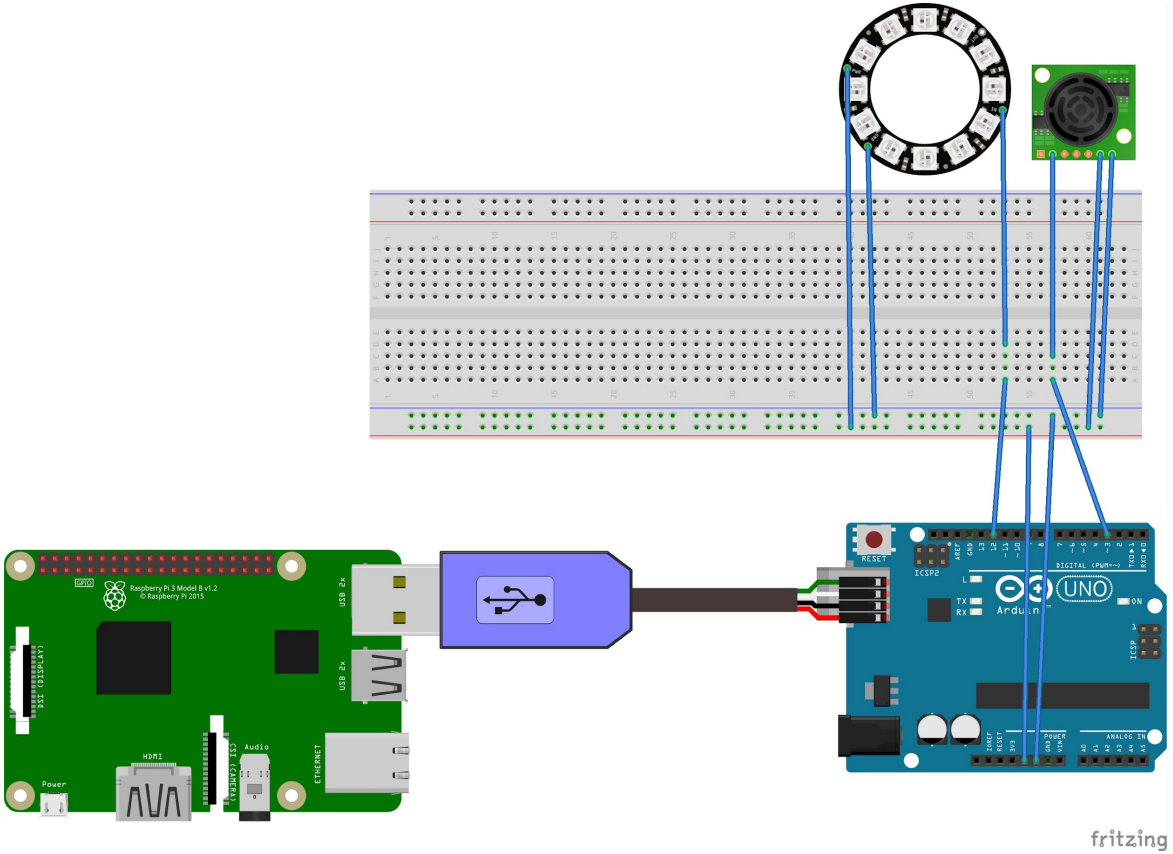
My contributions (maintainer)

- 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



Typical Firmata Architecture

```elixir
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

```elixir
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

```elixir
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
```

Custom Firmata Additions

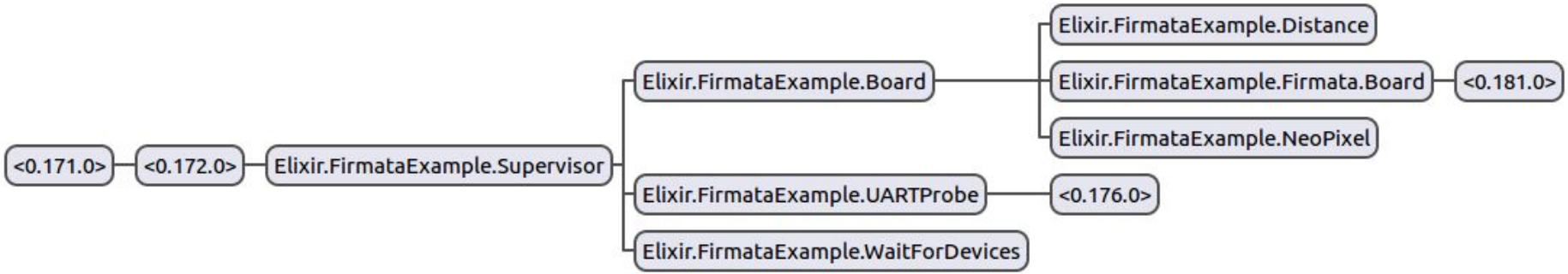
```
#include <Servo.h>                                    #include <Servo.h>
#include <Wire.h>                                     #include <Wire.h>
#include <Firmata.h>                                  #include <Firmata.h>
#include <NewPing.h>                                <
#include <Adafruit_GFX.h>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

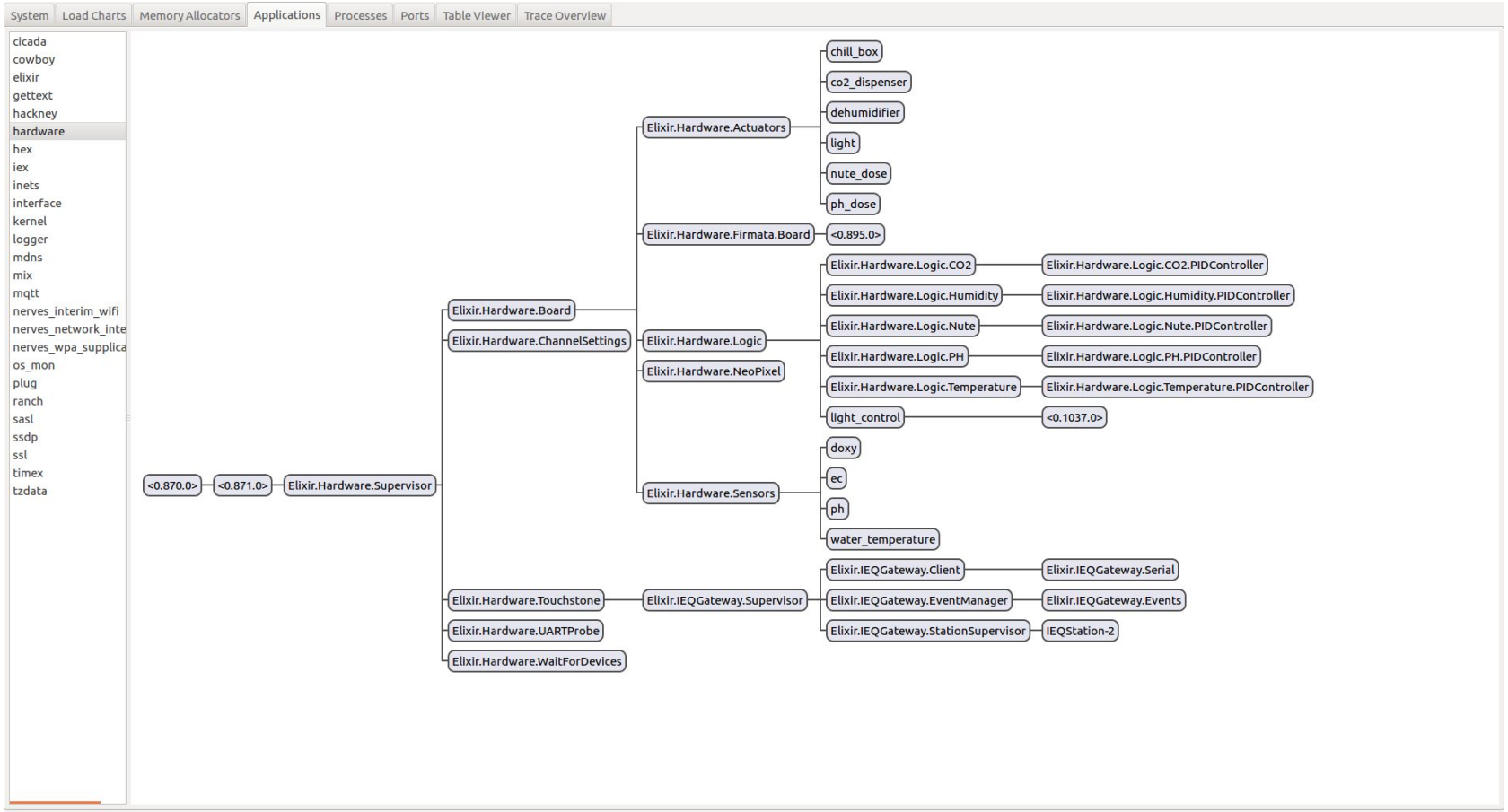
Modifying the Firmata Client

```
# 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




System
Load Charts
Memory Allocators
Applications
Processes
Ports
Table Viewer
Trace Overview
cicada
cowboy
elixir
gettext
hackney
hardware
hex
iex
inets
interface
kernel
logger
mdns
mix
mqtt
nerves_interim_wifi
nerves_network_inte
nerves_wpa_supplica
os_mon
plug
ranch
sasl
ssdp
ssl
timex
tzdata
<0.870.0>
<0.871.0>
Elixir.Hardware.Supervisor
Elixir.Hardware.Board
Elixir.Hardware.ChannelSettings
Elixir.Hardware.Touchstone
Elixir.Hardware.UARTProbe
Elixir.Hardware.WaitForDevices
Elixir.Hardware.Actuators
Elixir.Hardware.Firmata.Board
Elixir.Hardware.Logic
Elixir.Hardware.NeoPixel
Elixir.Hardware.Sensors
chill_box
co2_dispenser
dehumidifier
light
nute_dose
ph_dose
<0.895.0>
Elixir.Hardware.Logic.CO2
Elixir.Hardware.Logic.Humidity
Elixir.Hardware.Logic.Nute
Elixir.Hardware.Logic.PH
Elixir.Hardware.Logic.Temperature
light_control
Elixir.Hardware.Logic.CO2.PIDController
Elixir.Hardware.Logic.Humidity.PIDController
Elixir.Hardware.Logic.Nute.PIDController
Elixir.Hardware.Logic.PH.PIDController
Elixir.Hardware.Logic.Temperature.PIDController
<0.1037.0>
doxy
ec
ph
water_temperature
Elixir.IEQGateway.Supervisor
Elixir.IEQGateway.Client
Elixir.IEQGateway.EventManager
Elixir.IEQGateway.StationSupervisor
Elixir.IEQGateway.Serial
Elixir.IEQGateway.Events
IEQStation-2

Thank You

@entropealab

@CRTLabs

https://github.com/entone

http://code.crtlabs.org

https://github.com/entone/firmata_example