



# Elixir

100k Connected IoT Devices

# Contents



Electricity Grid

Supply vs Demand

Demand Response domain

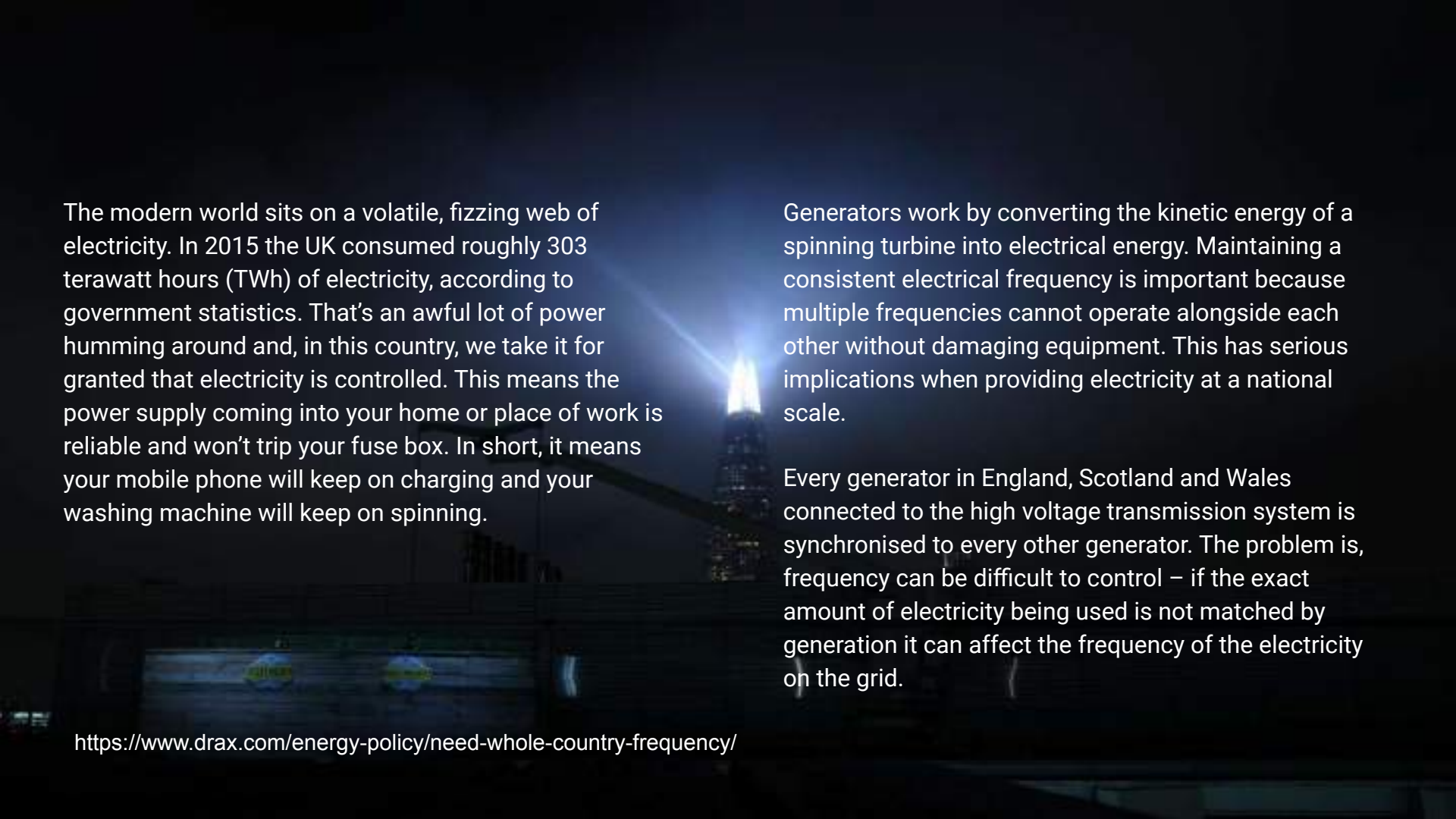
Customer Proposition

Elixir Solution

System Testing

Load Testing

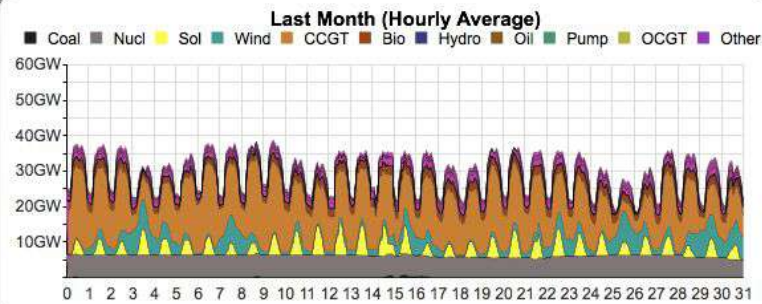
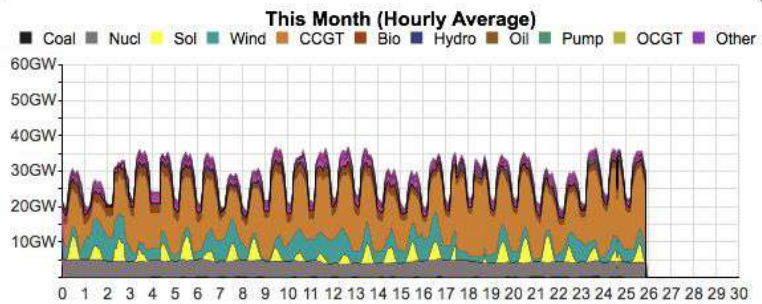
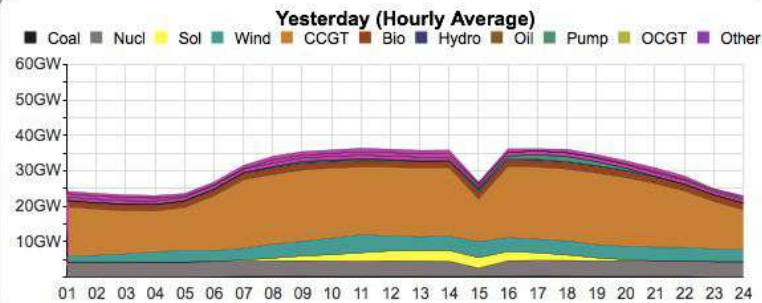
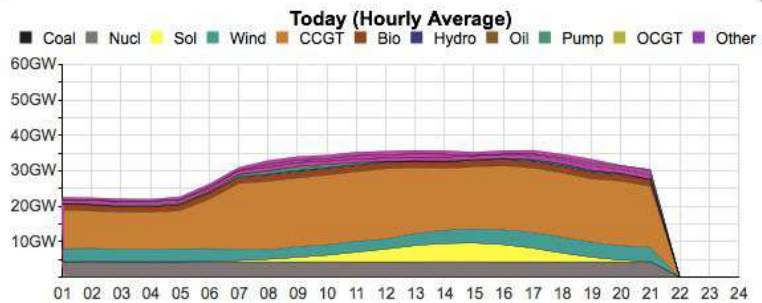
Conclusion (did they buy?)



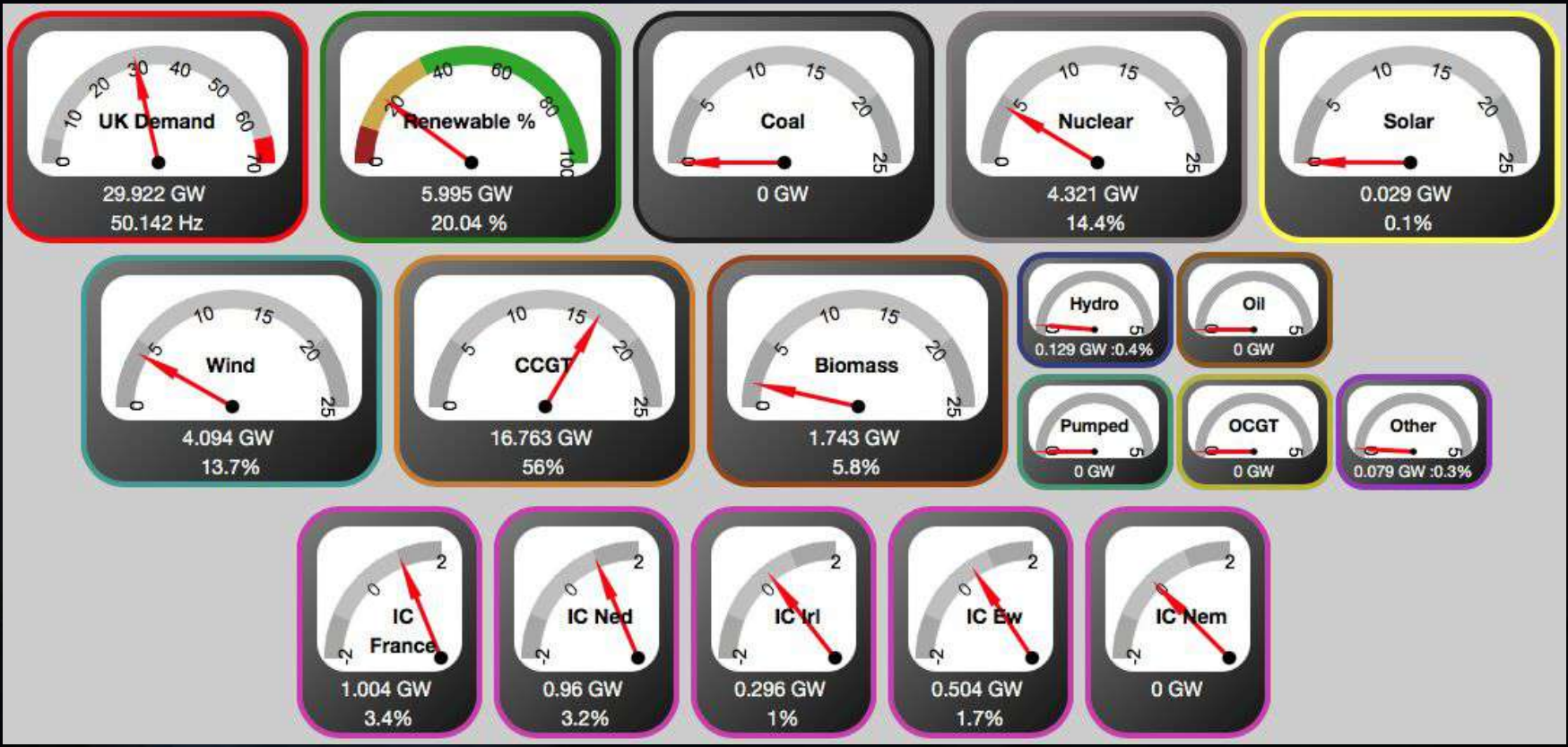
The modern world sits on a volatile, fizzing web of electricity. In 2015 the UK consumed roughly 303 terawatt hours (TWh) of electricity, according to government statistics. That's an awful lot of power humming around and, in this country, we take it for granted that electricity is controlled. This means the power supply coming into your home or place of work is reliable and won't trip your fuse box. In short, it means your mobile phone will keep on charging and your washing machine will keep on spinning.

Generators work by converting the kinetic energy of a spinning turbine into electrical energy. Maintaining a consistent electrical frequency is important because multiple frequencies cannot operate alongside each other without damaging equipment. This has serious implications when providing electricity at a national scale.

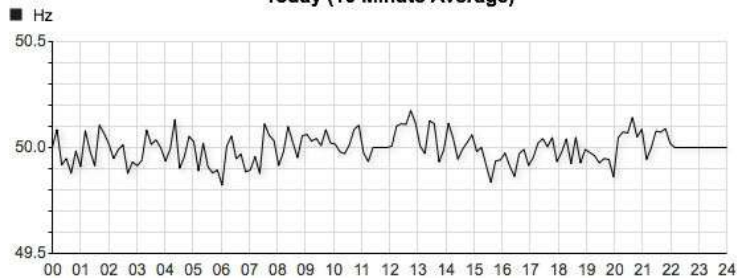
Every generator in England, Scotland and Wales connected to the high voltage transmission system is synchronised to every other generator. The problem is, frequency can be difficult to control – if the exact amount of electricity being used is not matched by generation it can affect the frequency of the electricity on the grid.





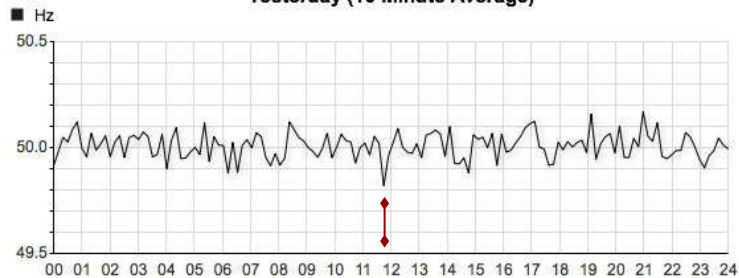


Today (10 Minute Average)



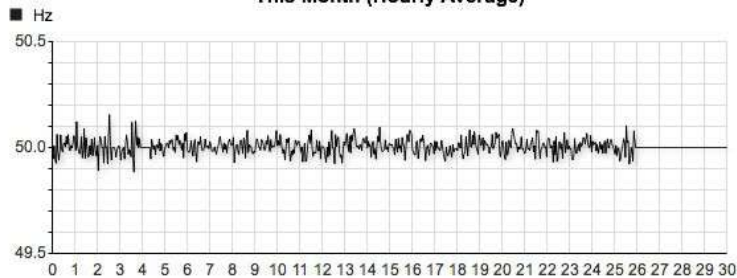
minimum : 49.823 Hz maximum : 50.174 Hz average : 49.997 Hz

Yesterday (10 Minute Average)



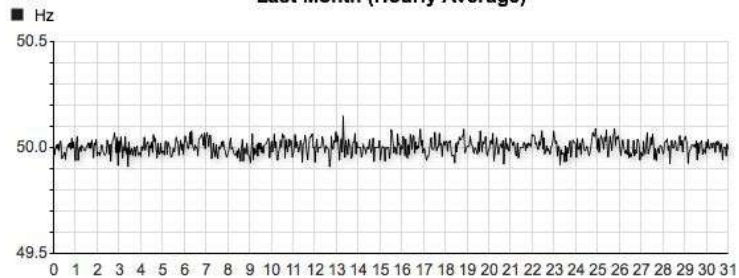
minimum : 49.82 Hz maximum : 50.169 Hz average : 50.009 Hz

This Month (Hourly Average)



minimum : 49.808 Hz maximum : 50.186 Hz average : 50.006 Hz

Last Month (Hourly Average)



minimum : 49.791 Hz maximum : 50.186 Hz average : 50.004 Hz

A close-up photograph of a large industrial turbine. The central focus is a long, polished metal shaft extending from the left towards the center. Behind the shaft is a large, curved, ribbed component, likely a turbine casing or a large flywheel, which is dark in color and has a textured surface. The background is slightly blurred, showing other parts of the machinery and a bright light source on the left. The overall scene is industrial and technical.

# Demand Response


Protect the turbines  
Save natural resources





An aerial view of a city skyline at dusk. The sky is a deep purple and blue. In the foreground, there are several buildings with signs. One prominent sign reads "F21 FOREVER 21". To the right, another sign features the "MODI" logo. The background shows a dense cluster of skyscrapers, including the Empire State Building, which is illuminated and stands out against the twilight sky. The overall scene is a mix of urban architecture and natural light from the setting sun.

In the past we match  
Supply to Demand

An aerial view of a city skyline at dusk, with a dense urban landscape in the foreground and a prominent skyscraper skyline in the background. The sky is a deep purple and blue. In the foreground, several buildings are visible, including one with a large sign that reads "F21 FOREVER 21" and another with a sign that reads "MODI". The text "With IoT and connected devices, we can also reduce the Demand" is overlaid in white, centered in the image.

With IoT and connected devices,  
we can also reduce the Demand



Customer Req

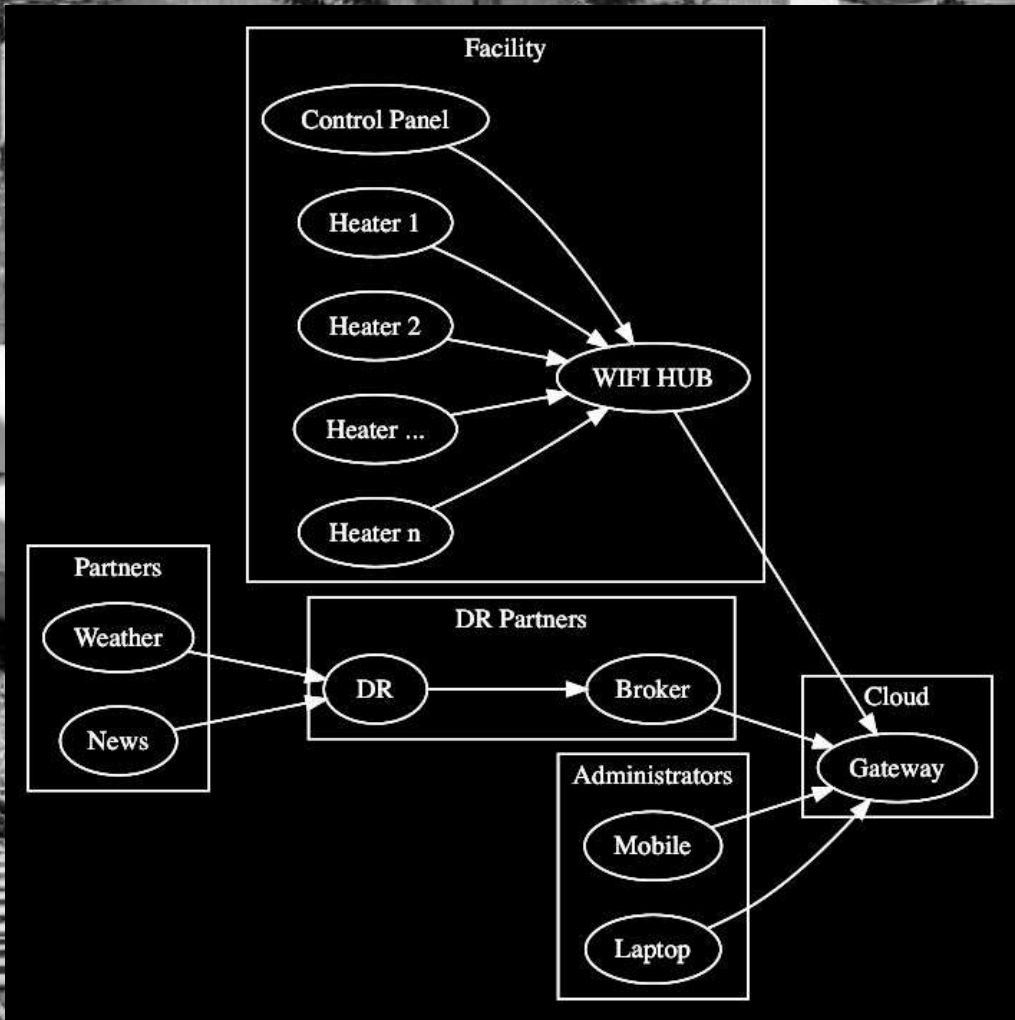
Demand Response solution

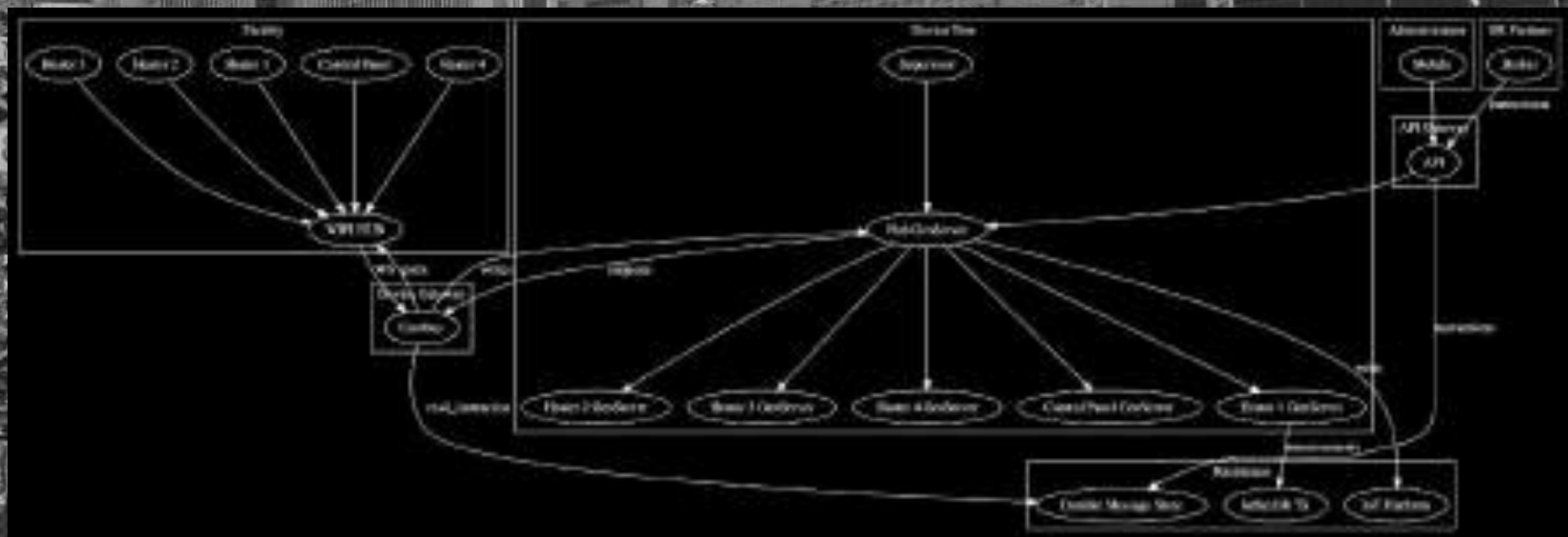
Scale to 100k connected devices

Scale product range (time to market)

Monitoring and metrics

Ease of administration





```
def unwrap_message(packet = <<
  128, _dest :: size(24),
  0x00      :: size(8),
  128, _src  :: size(24),
  0x00      :: size(8),
  msg_size  :: size(8),
  0x00,
  0x00,
  _command  :: size(8),
  _         :: binary-size(msg_size),
  crc_hi    :: size(8),
  crc_lo    :: size(8),
  trailer   :: binary >>) do
  <<header :: binary-size(@header_length), data :: binary-size(msg_size),
    _::binary>> = packet

  case :crc16.calc(header ◊ data) do
    <<^crc_hi, ^crc_lo>> → {header, data, trailing_uuid(trailer)}
    _                    → :corrupt
  end
end
```

```
def init(req =%{mac: mac}, _opts) do
  state = %{connected_at: now_s(),
            mac: mac,
            pending_command: nil,
            queued_commands: :queue.new(),
            sent_t: nil}
  {:cowboy_websocket, req, state}
end

def websocket_handle({:binary, payload}, state = %{sent_t: sent_t, mac: mac}) do
  case Packet.unwrap_message(payload) do
    {hdr, data, id} → handle_message(hdr, data, id, state)

    :invalid_packet → log(%{invalid: inspect(payload), mac: mac})
                     send_next_message(state)

    :corrupt → log(%{corrupt: inspect(payload), mac: mac})
               send_next_message(state)
  end
end
```



```
use ExUnit.Case
use ExCheck

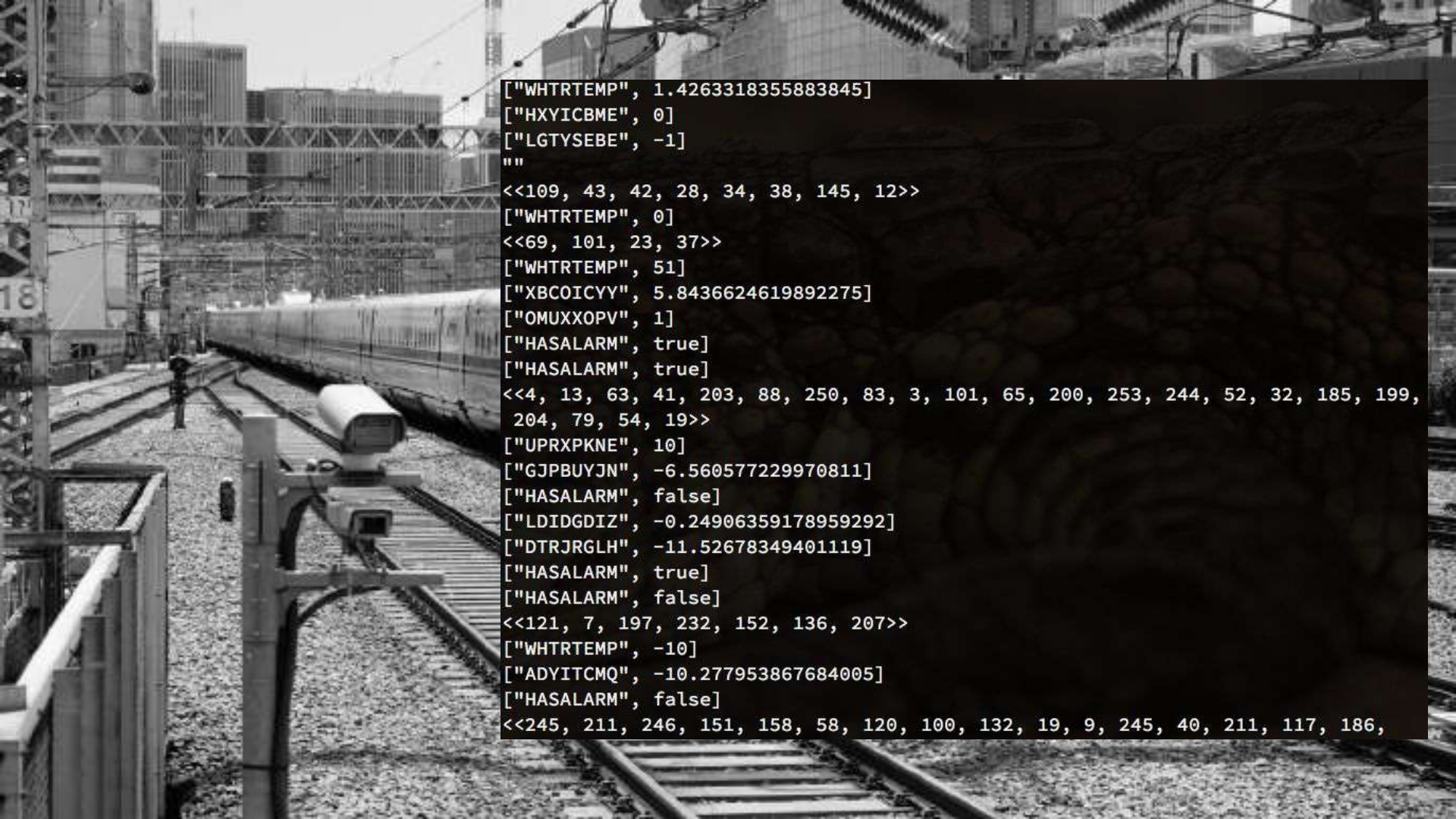
import :triq_dom

def gen_hasalarm, do: bool() ▷ bind(fn b → ["HASALARM", b] end)

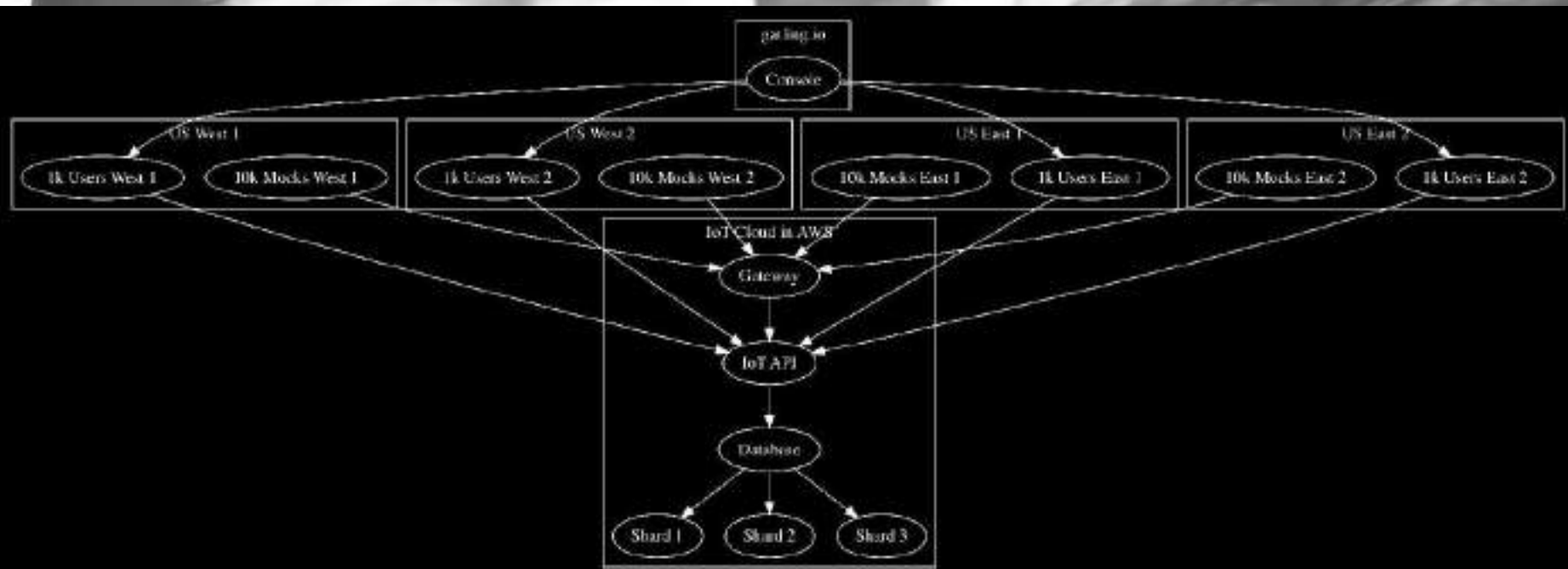
def gen_temp, do: oneof([oneof([0, 1]), choose(1, 99), int(), float()])
def gen_whtrtemp, do: gen_temp() ▷ bind(fn t → ["WHTRTEMP", t] end)

def gen_letter, do: ?A..?Z ▷ Enum.map(fn x → <<x :: utf8>> end) ▷ oneof()
def gen_field, do: vector(8, gen_letter()) ▷ bind(fn xs → Enum.join(xs) end)
def gen_value, do: oneof([int(), float(), unicode_binary()])
def gen_random, do: [gen_field(), gen_value()] ▷ bind(fn [f, v] → [f,v] end)

def gen_request, do: oneof([
  oneof([gen_hasalarm(), gen_whtrtemp()]),
  oneof([gen_hasalarm(), gen_whtrtemp(), gen_random()]),
  gen_random(),
  binary()
])
```

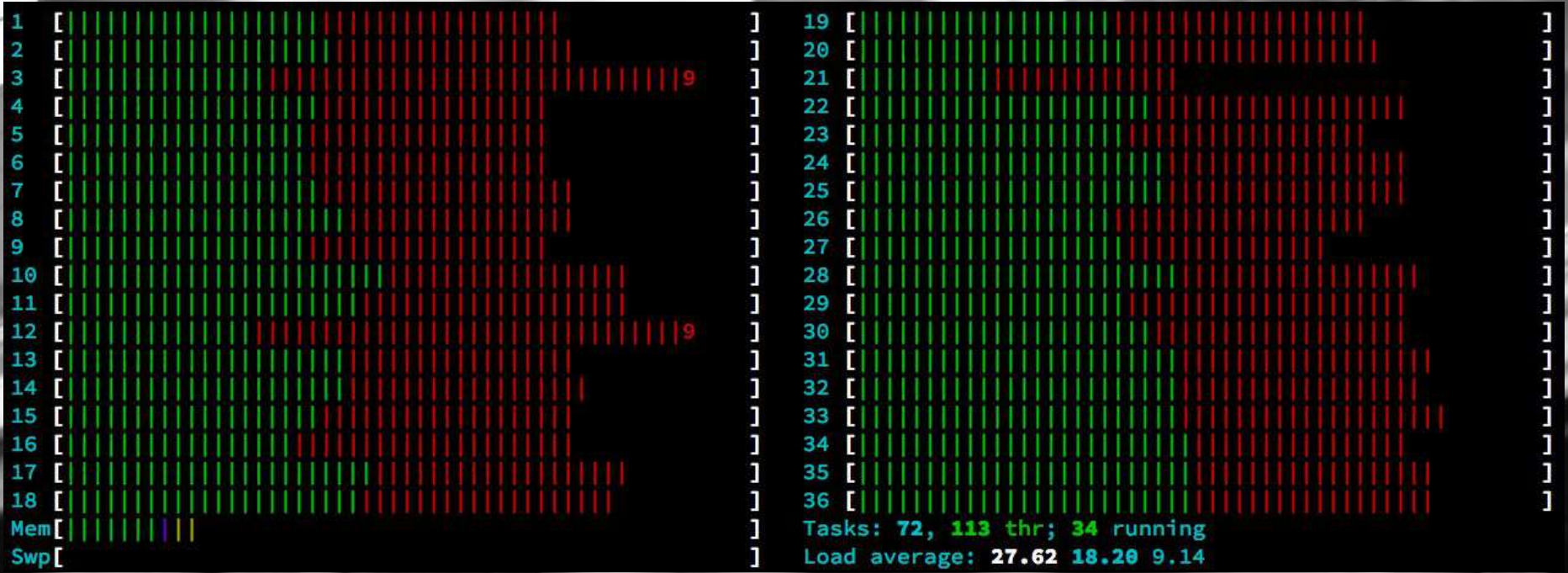


```
["WHTRTEMP", 1.4263318355883845]
["HXYICBME", 0]
["LGTYSEBE", -1]
""
<<109, 43, 42, 28, 34, 38, 145, 12>>
["WHTRTEMP", 0]
<<69, 101, 23, 37>>
["WHTRTEMP", 51]
["XBCOICY", 5.8436624619892275]
["OMUXXOPV", 1]
["HASALARM", true]
["HASALARM", true]
<<4, 13, 63, 41, 203, 88, 250, 83, 3, 101, 65, 200, 253, 244, 52, 32, 185, 199,
204, 79, 54, 19>>
["UPRXPKE", 10]
["GJPBUYJN", -6.560577229970811]
["HASALARM", false]
["LDIDGDIZ", -0.24906359178959292]
["DTRJRGLH", -11.52678349401119]
["HASALARM", true]
["HASALARM", false]
<<121, 7, 197, 232, 152, 136, 207>>
["WHTRTEMP", -10]
["ADYITCMQ", -10.277953867684005]
["HASALARM", false]
<<245, 211, 246, 151, 158, 58, 120, 100, 132, 19, 9, 245, 40, 211, 117, 186,
```



Load testing

```
1 [||||| 91 ] 5 [||||| 92. ]
2 [||||| 91. ] 6 [||||| 93. ]
3 [||||| 8 ] 7 [||||| 93. ]
4 [||||| ] 8 [||||| 94.3 ]
Mem [||||| ] Tasks: 40, 56 thr; 2 running
Swp [ ] Load average: 6.63 3.28 1.63
Uptime: 01:25:28
```



100k  
36vCPU 72GB

Decision



# Retrospective



Greenspun's Tenth Rule of Programming: any sufficiently complicated C or Fortran program contains an ad hoc informally-specified bug-ridden slow implementation of half of Common Lisp.

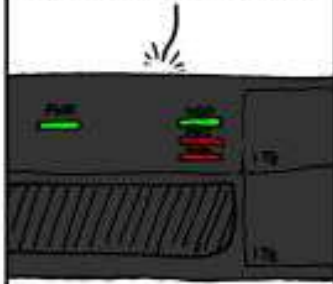
HI!  
I'M A SERVER!  
WHO ARE YOU?



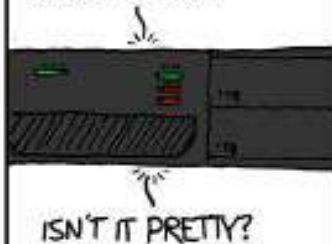
I'M A BROWSER.  
I'D LIKE TO SEE  
THIS ARTICLE.



OH BOY! I CAN HELP!  
LET ME GET IT FOR—  
...WHOA! YOU'RE A  
SMARTPHONE BROWSER?



YEAH.  
COOOOL! HEY, I'VE  
GOT THIS NEW MOBILE  
VERSION OF MY SITE!  
CHECK IT OUT!



ISN'T IT PRETTY?

SURE, BUT THIS IS  
JUST YOUR MOBILE  
SITE'S MAIN PAGE.  
WHERE'S THE  
ARTICLE I WANTED?



WHAT ARTICLE?  
THE ONE I— )  
/ WHO ARE YOU?  
I— /  
HI! I'M A SERVER!





Ex falso quodlibet



Principle of Explosion

-from false, anything follows

15

Imagine That ...



Shun Street  
順街 7-1

The Little Typer



David F. Friedman and David Thore Christensen  
Illustrations by Robert Meyer    Afterword by Carol McLeod  
Drawings by Orlan Gibby

Conclusion

