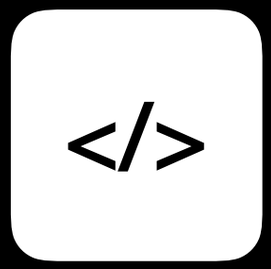
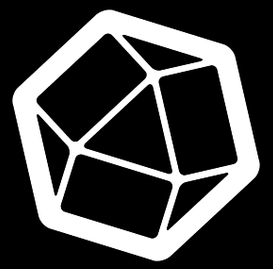
Digital Twins: Using Maths across Measurements

David G. Simmons
Senior Developer Evangelist, InfluxData
bit.ly/dgctalk



code.talks



InfluxData

@davidgsIoT

It’s about Digital Twins

But really it’s about maths

Photo by William Daigneault on Unsplash

Gas is easy: How much gas is there?

Ideal Gas Law

$PV=nRT$

Photo by Martin Adams on Unsplash

Heat Index: How hot does it *feel*

Heat index is *hard*

Steadman's Equation:

HI = 0.5 * {T + 61.0 + [(T-68.0)*1.2] + (RH*0.094)}

Rothsfuz Regression:

HI = -42.379 + 2.04901523*T + 10.14333127*RH - .22475541*T*RH - .00683783*T*T - .05481717*RH*RH + .00122874*T*T*RH + .00085282*T*RH*RH - .00000199*T*T*RH*RH

Photo by Zack Woolwine on Unsplash

There are lots of others

- Saturation Vapor Pressure
- Vapor Pressure Deficit
- Dew Point
- Air Density

Let’s talk about Flux

- Flux is a functional data scripting language that is written to be:
  - Useable
  - Readable
  - Composable
  - Testable
  - Contributable
  - Shareable
- Javascript-esque
- MIT License

Some Terminology

- **Bucket:** Named data source with a retention policy
- **Pipe-forward operator:** Used to chain operators together
- **Table:** Data is returned in annotated CSVs, represented as tables
- **Group key:** List of columns for which every row in the table has the same value

A Simple Flux Example

```
from(bucket:"telegraf/autogen")
  |> range(start: -1h)
  |> filter(fn: (r) =>
    r.measurement == “cpu” AND
    r.field == “usage_system” AND
    r.cpu == “cpu_total”)
  |> yield()
```

Let’s do some Maths in Flux!

- First, we build a tables:

```
CO2Raw = from(bucket: "telegraf")
    |> range(start: v.timeRangeStart)
    |> filter(fn: (r) => r._measurement == "k30_reader" and
(r._field == "co2"))
    |> aggregateWindow(every: 30s, fn: mean)
    |> filter(fn: (r) => exists r._value)
    |> keep(columns: ["_value", "_time"])
    |> yield(name: "Raw CO2”)
```

Here’s Your Table

Raw CO2			
result	table	_value	_time
	0	1259	2019-10-11T18:25:30Z
	0	1270	2019-10-11T18:26:00Z
	0	1269	2019-10-11T18:26:30Z
	0	1275	2019-10-11T18:27:00Z
	0	1256	2019-10-11T18:27:30Z
	0	1239	2019-10-11T18:28:00Z
	0	1239	2019-10-11T18:28:30Z
	0	1263	2019-10-11T18:29:00Z

Build some more Tables

- Temperature

```
Tmeas = from(bucket: "telegraf")
  |> range(start: v.timeRangeStart)
  |> filter(fn: (r) => r._measurement == "temperature" and (r._field == "temp_c"))
  |> aggregateWindow(every: 30s, fn: mean)
  |> keep(columns: ["_value", "_time"])
```

- Pressure

```
Pmeas = from(bucket: "telegraf")
  |> range(start: v.timeRangeStart)
  |> filter(fn: (r) => r._measurement == "environment" and (r._field == "pressure"))
  |> aggregateWindow(every: 30s, fn: mean)
  |> keep(columns: ["_value", "_time"])
```

There Can Be Only One*

- Join 2 of the tables together:

```
first_join = join(tables: {CO2meas: CO2Raw, Tmeas: Tmeas}, on: ["_time"])
```

- Make the big table:

```
second_join = join(tables: {first_join: first_join, Pmeas: Pmeas}, on:
["_time"])
    |>map(fn: (r) => ({_time: r._time, pressure: r._value,
gas:r._value_CO2meas, temperature:r._value_Tmeas}))
    |> filter(fn: (r) => exists r.pressure)
    |> filter(fn: (r) => exists r.gas)
    |> filter(fn: (r) => exists r.temperature)
    |> environmental.idealGasLaw()
   |> yield(name: "Compensated")
```

*

</> </> </> </> </> </> </> </> </> </> </> </> </> </> </> </> </> </> </> </>

@davidgsIoT

Let’s see the Maths

```
idealGasLaw = (tables=<-) => tables
    |> map(fn: (r) => ({_time: r._time, _value: r.gas *
(((r.temperature + 273.15) * 1013.25) / (r.pressure *
298.15))}))
```

The End Result

</> </> </> </> </> </> </> </> </> </> </> </> </> </> </> </> </> </> </> </> </> @davidgsIoT

We can Hide Complex Maths

- We can add more equations to our environmental package
- Soon, we can share that package with other users

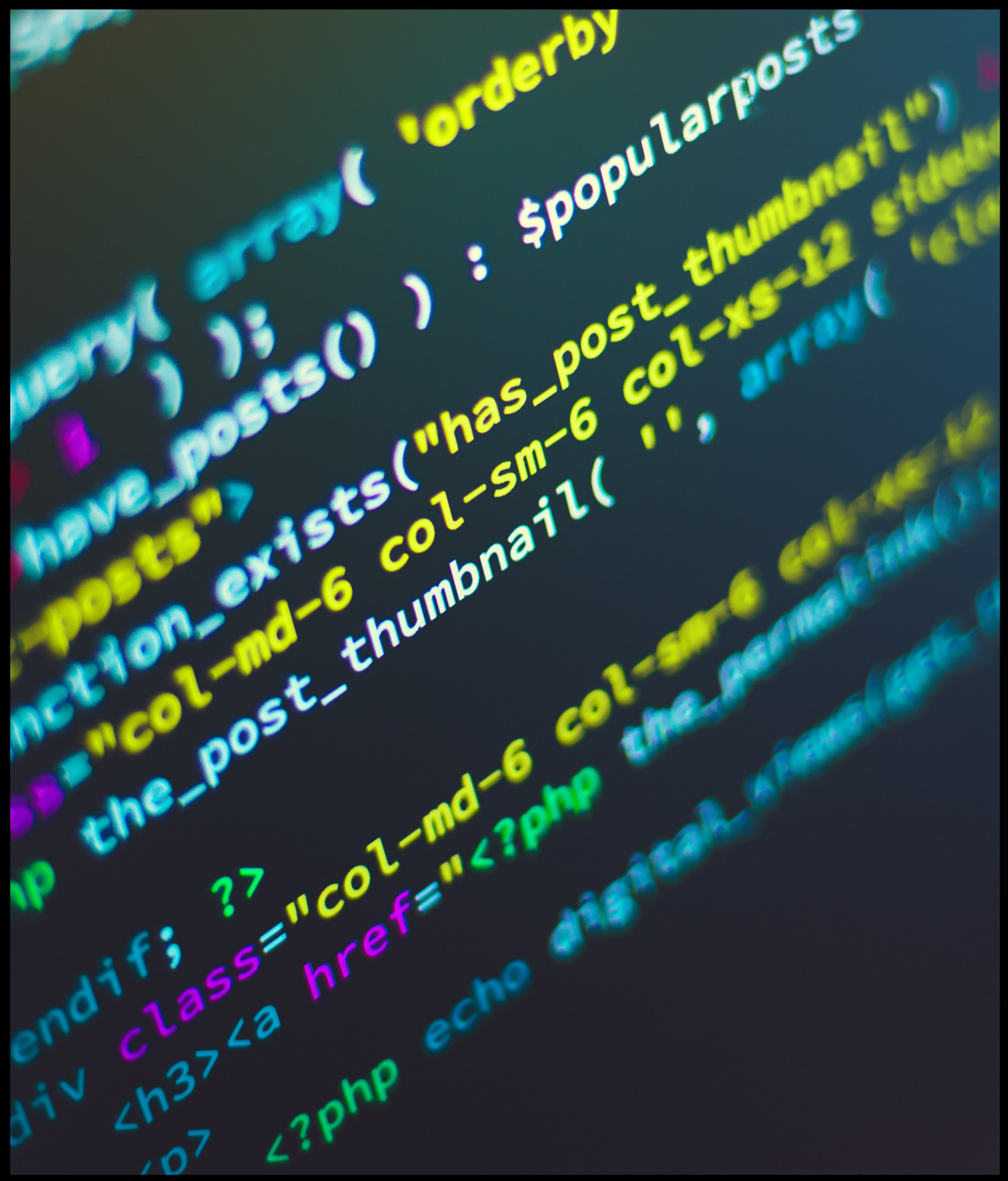


Photo by Shahadat Shemul on Unsplash

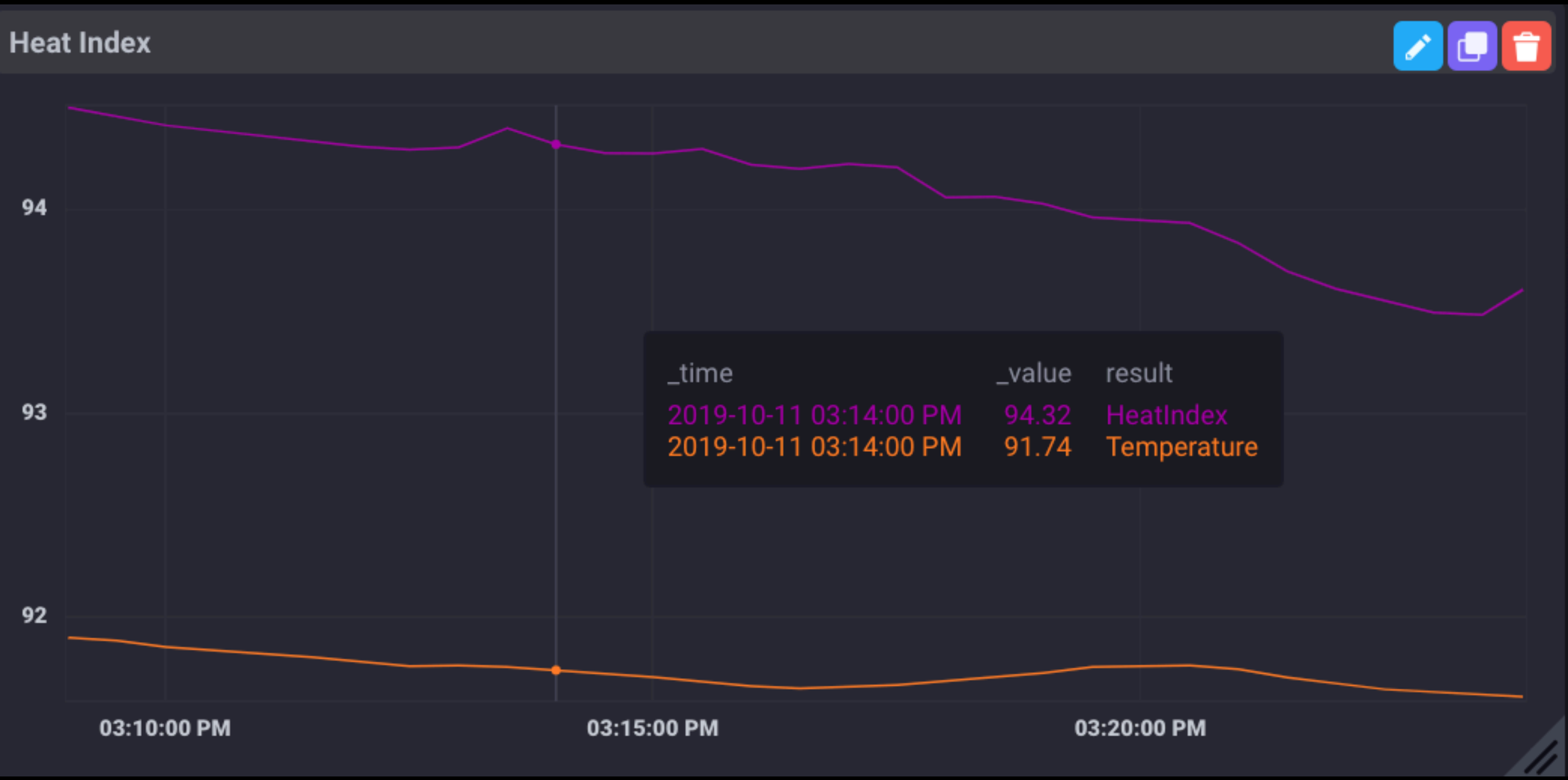
Steadman and Rothsfuz

```
// The 'simple' Steadman Equation
_steadman = (t,h) =>
    (0.5 * (t + 61.0 + ((t - 68.0)*1.2) + (h*0.094)))

// The more complex Rothsfuz Regression equation
_rothfusz = (t,h) =>
   -42.379 + 2.04901523*t + 10.14333127*h - .22475541*t*h - .00683783*t*t - .
05481717*h*h + .00122874*t*t*h + .00085282*t*h*h - .00000199*t*t*h*h

// required adjustment #1 to the Rothsfuz Regression
_roth_adjust1 = (t,h) =>
   ((13.0-h)/4.0)*math.sqrt(x: ((17.0-math.abs(x: (t-95.0))/17.0)))

// required adjustment #2 to the Rothsfuz Regression
_roth_adjust2 = (t,h) =>
   ((h-85.0 )/10.0) *((87.0-t)/5.0)
```

Heat Index is Hard

```
// Calculate the heat index given a table with Columns: temperature
and humidity
heatIndex = (tables=<-) => tables
    |> map(fn: (r) => ({
        r with _value:
            if  (_steadman(t: r.temperature, h: r.humidity) + r.temperature)/
2.0 < 80.0  then _steadman(t: r.temperature, h: r.humidity)
            else if ( r.humidty < 13.0 and r.temperature > 80.0) then
_rothfusz(t: r.temperature, h: r.humidity) - _roth_adjust1(t:
r.temperature, h: r.humidity)
            else if r.humidity > 85.0 and r.temperature >= 80.0 and
r.temperature <= 87.0 then _rothfusz(t: r.temperature, h: r.humidity) +
_roth_adjust2(t: r.temperature, h: r.humidity)
            else _rothfusz(t: r.temperature, h: r.humidity)
    }))
```

The End Result



</> </> </> </> </> </> </> </> </> </> </> </> </> </> </> </> </> </> </> </>

@davidgsIoT

Demo Time!

</> </> </> </> </> </> </> </> </> </> </> </> </> </> </> </> </> </> </> </>

@davidgsIoT

“I hate this damned machine
I wish that they would sell it.
It never does what I want
But only what I tell it.”

–My Mom

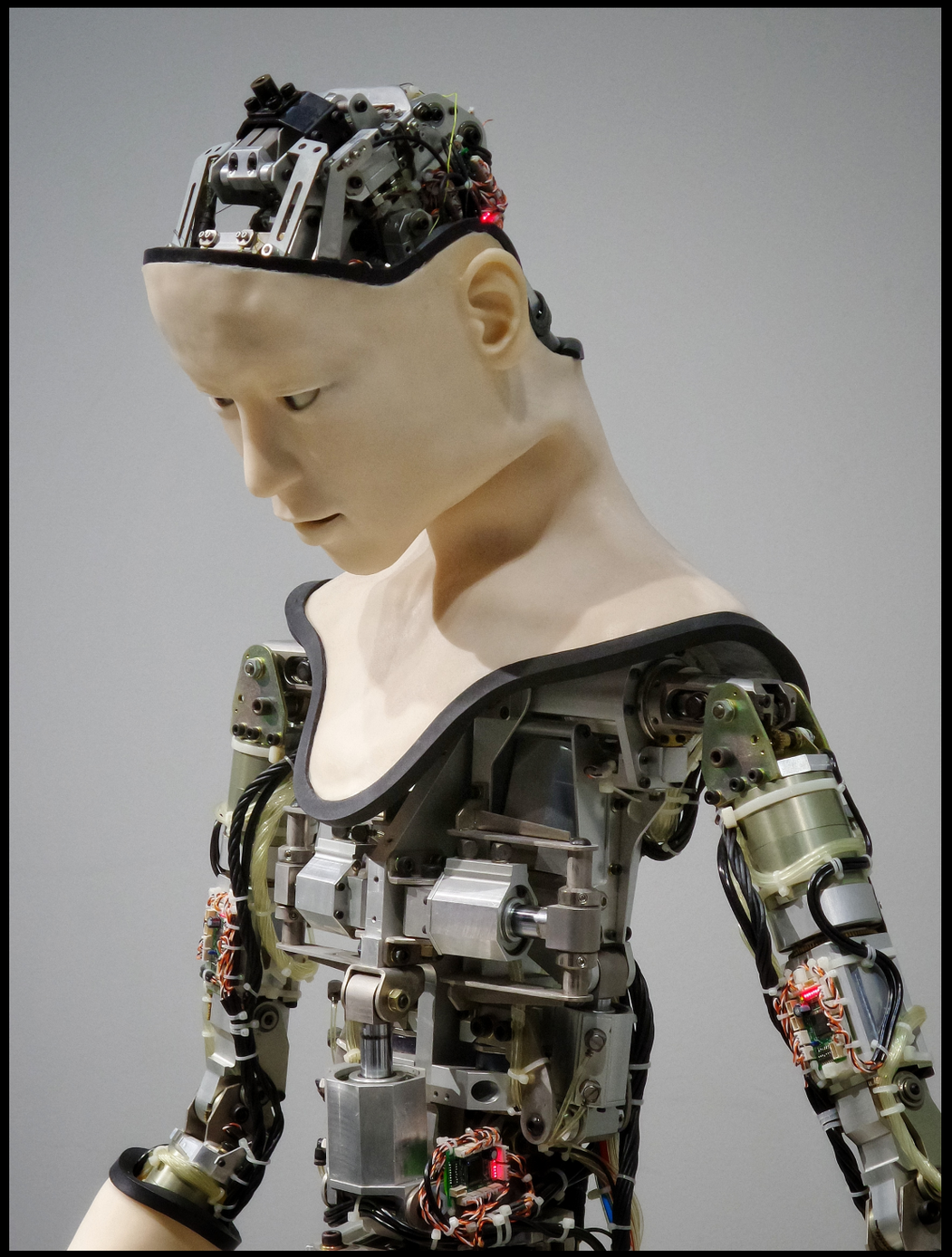


Photo by Franck V. on Unsplash

Resources

- Ideal Gas Law:
  - https://en.wikipedia.org/wiki/Ideal_gas_law
- Heat index:
  - https://www.weather.gov/media/ffc/ta_htindx.PDF
  - https://www.wpc.ncep.noaa.gov/html/heatindex_equation.shtml
- Flux:
  - https://v2.docs.influxdata.com/v2.0/query-data/
- This talk:
  - bit.ly/dgctalk

THANK YOU

influxdata®

Act in Time

@davidgsIoT