

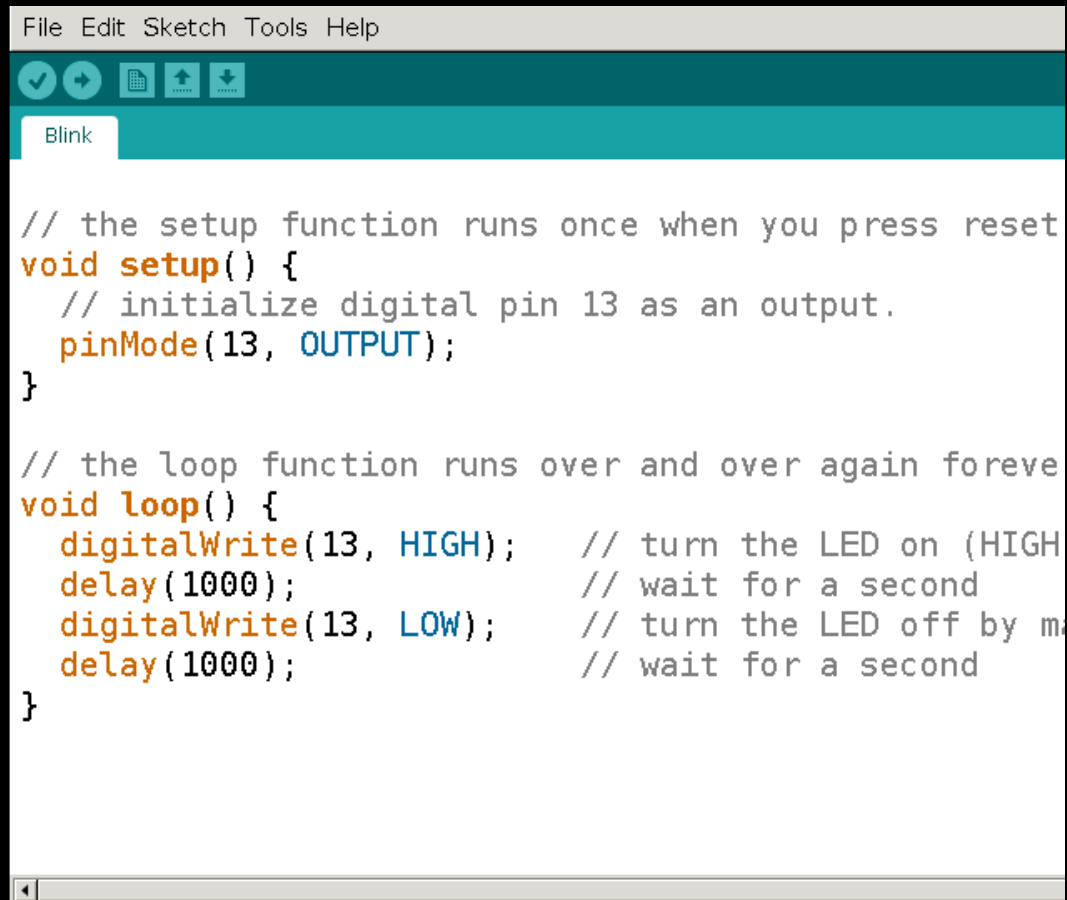
# MicroFlo

flow-based &  
visual programming  
for microcontrollers

<http://microflo.org>

Jon Nordby, MeshCon 2015

# Programming Today

A screenshot of the Arduino IDE interface. The menu bar at the top includes 'File', 'Edit', 'Sketch', 'Tools', and 'Help'. Below the menu bar is a toolbar with icons for opening, saving, and running. The main text area shows the code for a 'Blink' sketch. The code is written in C/C++ and uses syntax highlighting: keywords like 'void', 'setup', 'loop', 'pinMode', 'digitalWrite', and 'delay' are in orange; comments are in grey; and string literals like 'HIGH' and 'LOW' are in blue. The code defines a setup function to initialize pin 13 as an output and a loop function that turns the LED on and off with 1000ms delays.

```
File Edit Sketch Tools Help

Blink

// the setup function runs once when you press reset
void setup() {
  // initialize digital pin 13 as an output.
  pinMode(13, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(13, HIGH);   // turn the LED on (HIGH is the positive voltage)
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);              // wait for a second
}
```

“Code”

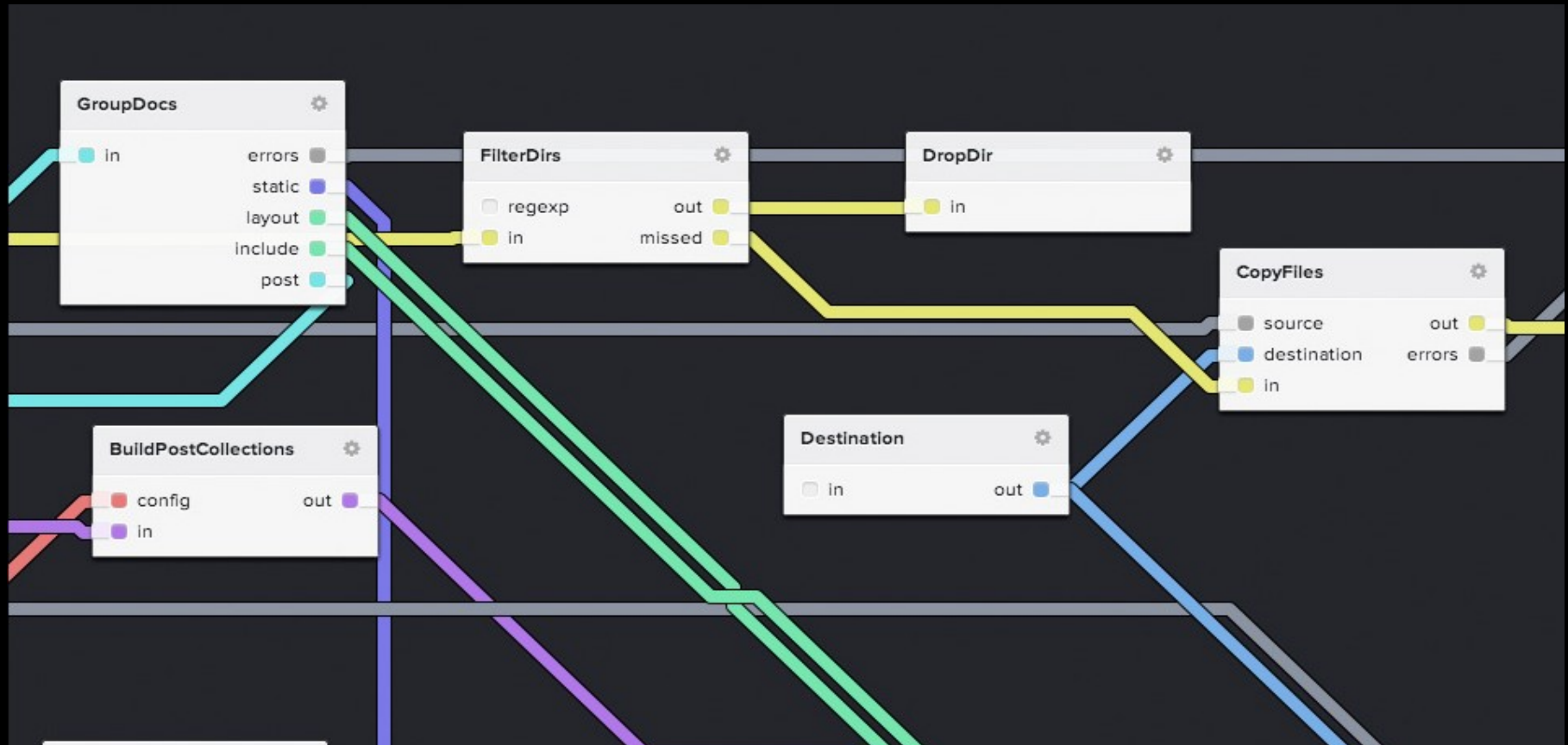
Text

C/C++

Imperative

write→compile→flash→restart→

# FBP 101



No source code generation!

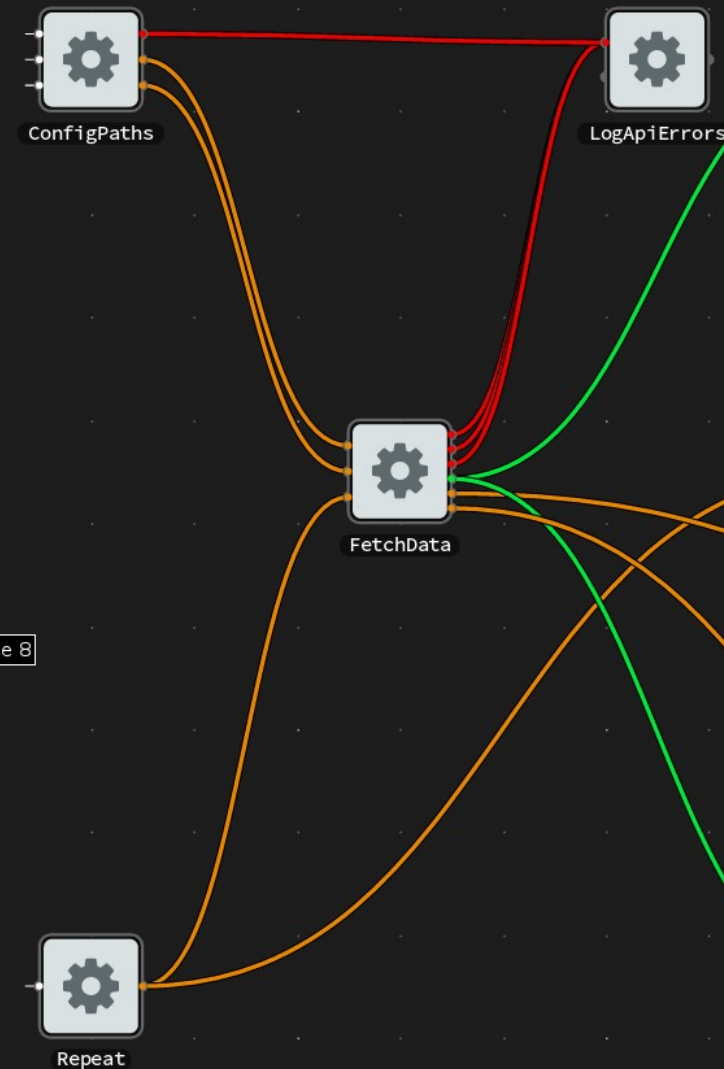
# Case: Ingress Table



<https://github.com/c-base/ingress-table>

# On Raspberry Pi

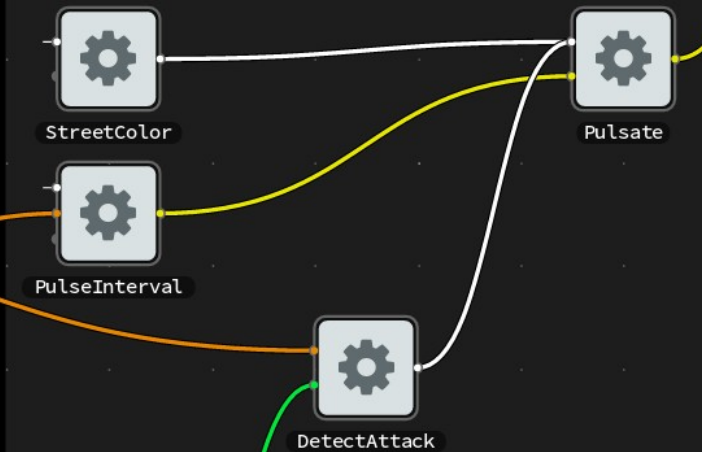
## Data Retrieval



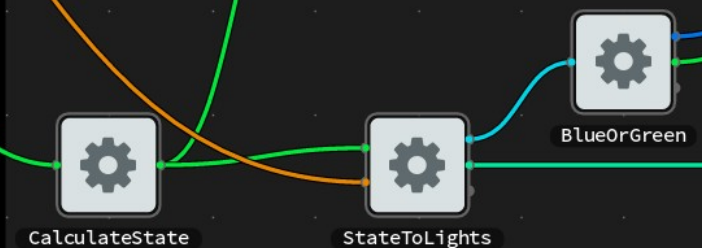
## c-beam



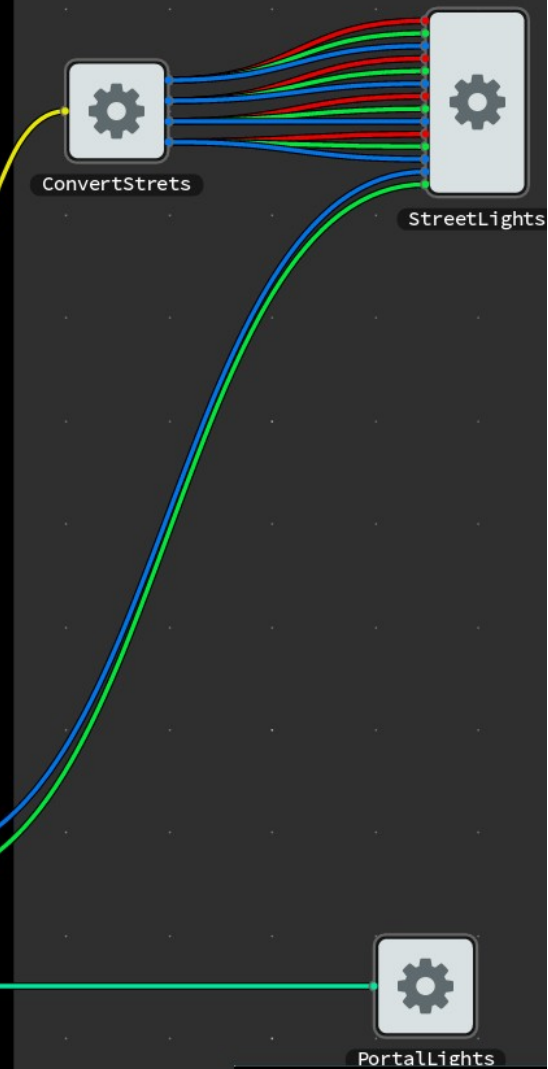
## Street Lights



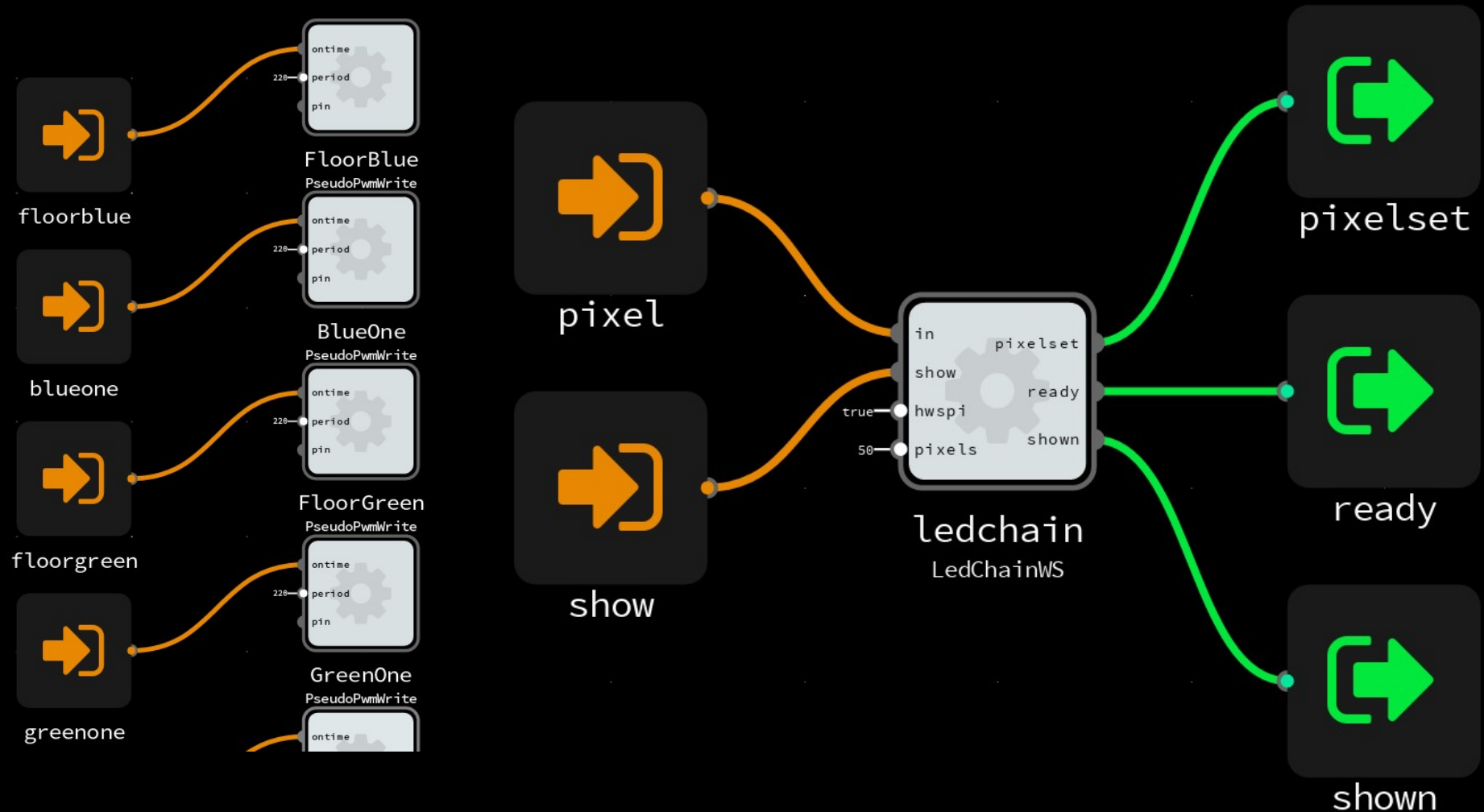
## Portal Status



## Ingress Table



# On microcontroller



# Inside component

```
/* microflo_component.yaml
name: Forward
description: Forward a packet from input to output
inports:
  in:
    type: all
    description: ""
outports:
  out:
    type: all
    description: ""
microflo_component */
```

```
class Forward : public SingleOutputComponent {
public:
    virtual void process(Packet in, MicroFlo::PortId port) {
        if (in.isData()) {
            send(in, port);
        }
    }
}
```



# Data-driven testing

```
1 topic: Forward
2 fixture:
3   type: 'fbp'
4   data: |
5     # @runtime microflo
6
7     INPORT=it.IN:IN
8     OUTPORT=it.OUT:OUT
9
10    it(Forward) OUT -> IN v(Forward)
11
12 cases:|
13 -
14   name: 'boolean true'
15   assertion: 'should be identical'
16   inputs:
17     in: true
18   expect:
19     out:
20       equals: true
21 -
```

<https://github.com/flowbased/fbp-spec>

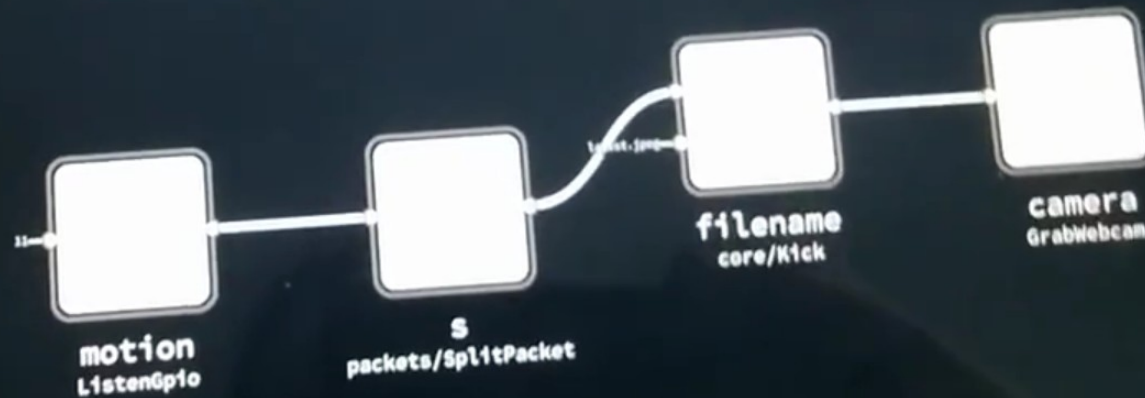


# Live mode

Flowhub live mode open from NFC tag

Opens IDE in browser

default/main



0:18 / 0:38

CC HD

<https://www.youtube.com/watch?v=EdgeSDFd9p0>

# Visual programming optional

```
1 # Thermostat
2 timer(Timer) OUT -> TRIGGER thermometer(ReadDallasTemperature)
3 thermometer() OUT -> IN hysteresis(HysteresisLatch)
4
5 # On/Off switch
6 hysteresis() OUT -> IN switch(BreakBeforeMake)
7 switch() OUT1 -> IN ia(InvertBoolean) OUT -> IN turnOn(DigitalWrite)
8 switch() OUT2 -> IN ic(InvertBoolean) OUT -> IN turnOff(DigitalWrite)
9 # Feedback cycle to switch required for synchronizing break-before-make logic
10 turnOn() OUT -> IN ib(InvertBoolean) OUT -> MONITOR1 switch()
11 turnOff() OUT -> IN id(InvertBoolean) OUT -> MONITOR2 switch()
12
13 # Config
14 '5000' -> INTERVAL timer() # milliseconds
15 '4' -> LOWTHRESHOLD hysteresis() # Celcius
16 '5' -> HIGHTHRESHOLD hysteresis() # Celcius
17 '["0x28", "0xAF", "0x1C", "0xB2", "0x04", "0x00", "0x00", "0x33"]' -> ADDRESS
   thermometer()
18 board(ArduinoUno) PIN9 -> PIN thermometer()
19 board() PIN12 -> PIN turnOff()
```

**.FBP** domain-specific language  
+ **embedding** APIs

# Get involved

MQTT **workshop at c-base**, Monday 18.00

At MeshCon, **chat afterwards**

**Visit** [flowhub.io](http://flowhub.io) [microflo.org](http://microflo.org)

Jon Nordby, [jonnor.com](http://jonnor.com), @jononor