AUTOMATION
INDUSTRIAL IOT

Software and Services Group
IoT Developer Relations, Intel

https://www.intel.com/content/www/us/en/big-data/pecan-street-smart-grid-technology-video.html

THE NEXT 10 YEARS AND AUTOMATION

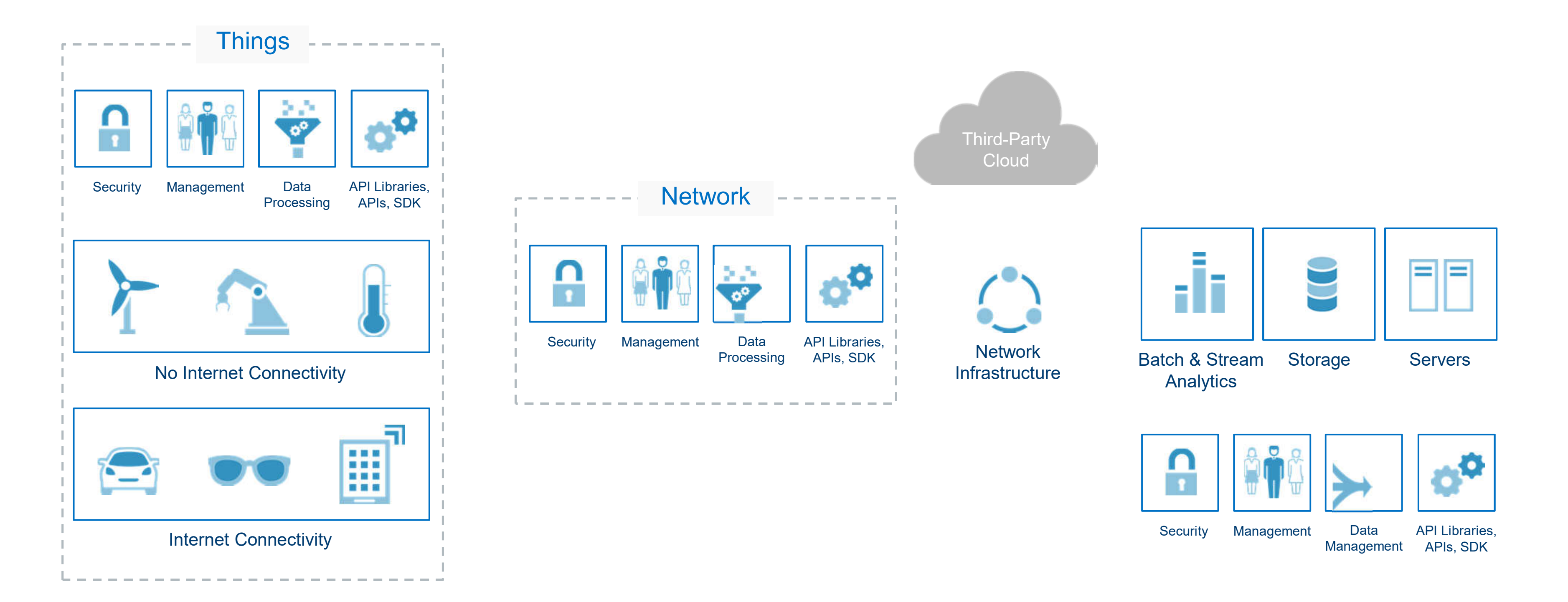
Big Data and the Internet of Things will drive automation, expert systems and artificial intelligence into common use.

The earliest system systems in artificial intelligence were rule based systems similar to many system in IoT

As IoT system increase in complexity and create huge amounts of data intelligent systems will be needed to understand and create actionable, intelligent processes.

INTEL® IOT PLATFORM – WHAT IS CLOSE LOOP MANAGEMENT?

Closed loop management is the ability of an IoT Edge Network to monitor sensors and automatically trigger actions based on predefined rules.



WHAT AND WHY?

- **Efficiency, safety and productivity -** Moving from a reactive to proactive maintenance operation has great effects.
- **Real time understanding of the status** of the critical factors for a factory or corporate installation will aid to accurately predict when an important or high-value piece of equipment may break-down, and then fixing the issue ahead of time, enables you to remain competitive and profitable.
- **Liability and Risk Reduction** - companies that are able to identify potential risks can minimize their liabilities.
- **Return on Investment** is fast becoming a core component of next generation predictive maintenance and optimization applications.
- **Accelerated learning and understanding** - The renaissance of cognitive sensing allows businesses to be intelligent with their time and resources.

APPROACHES TO AUTOMATION SYSTEMS

ANN - Artificial neural network

DCS - Distributed Control System

HMI - Human Machine Interface

SCADA - Supervisory Control and Data Acquisition

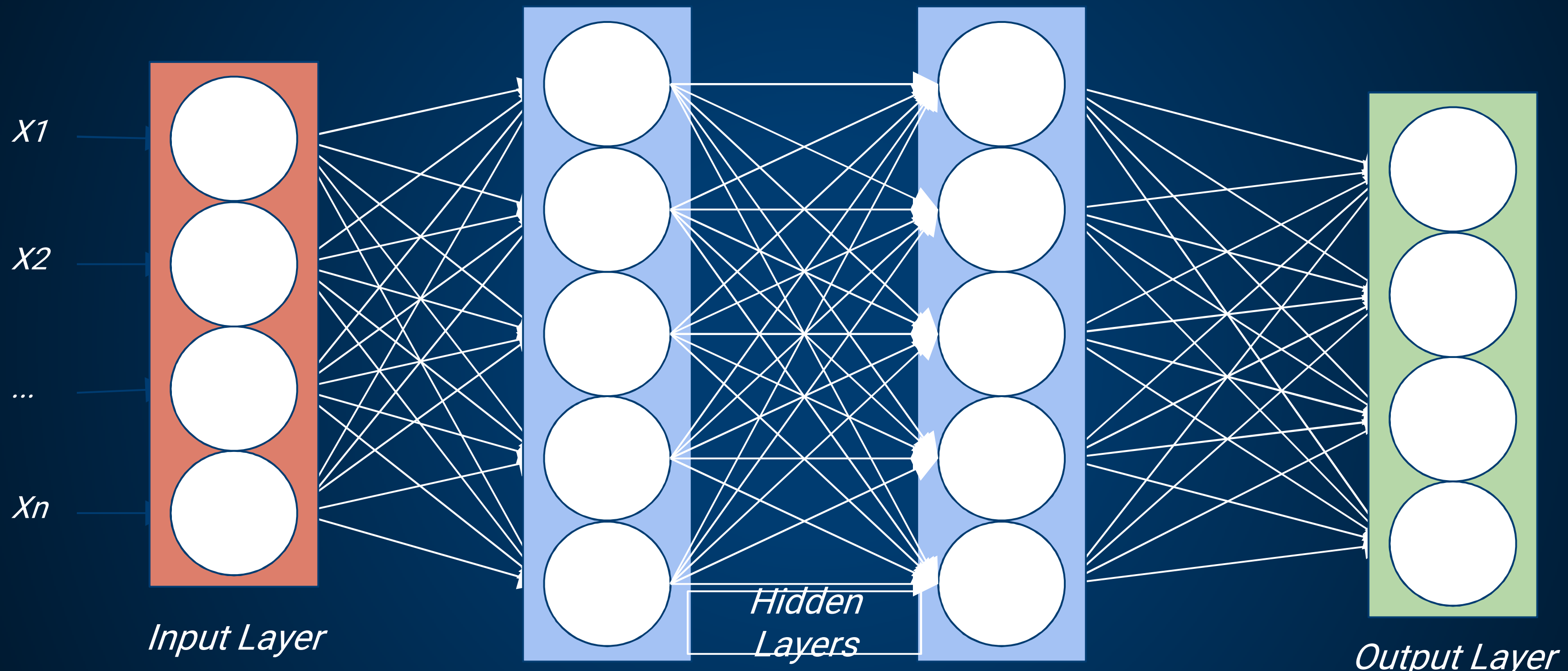
PLC - Programmable Logic Controller

APPROACHES TO AUTOMATION SYSTEMS

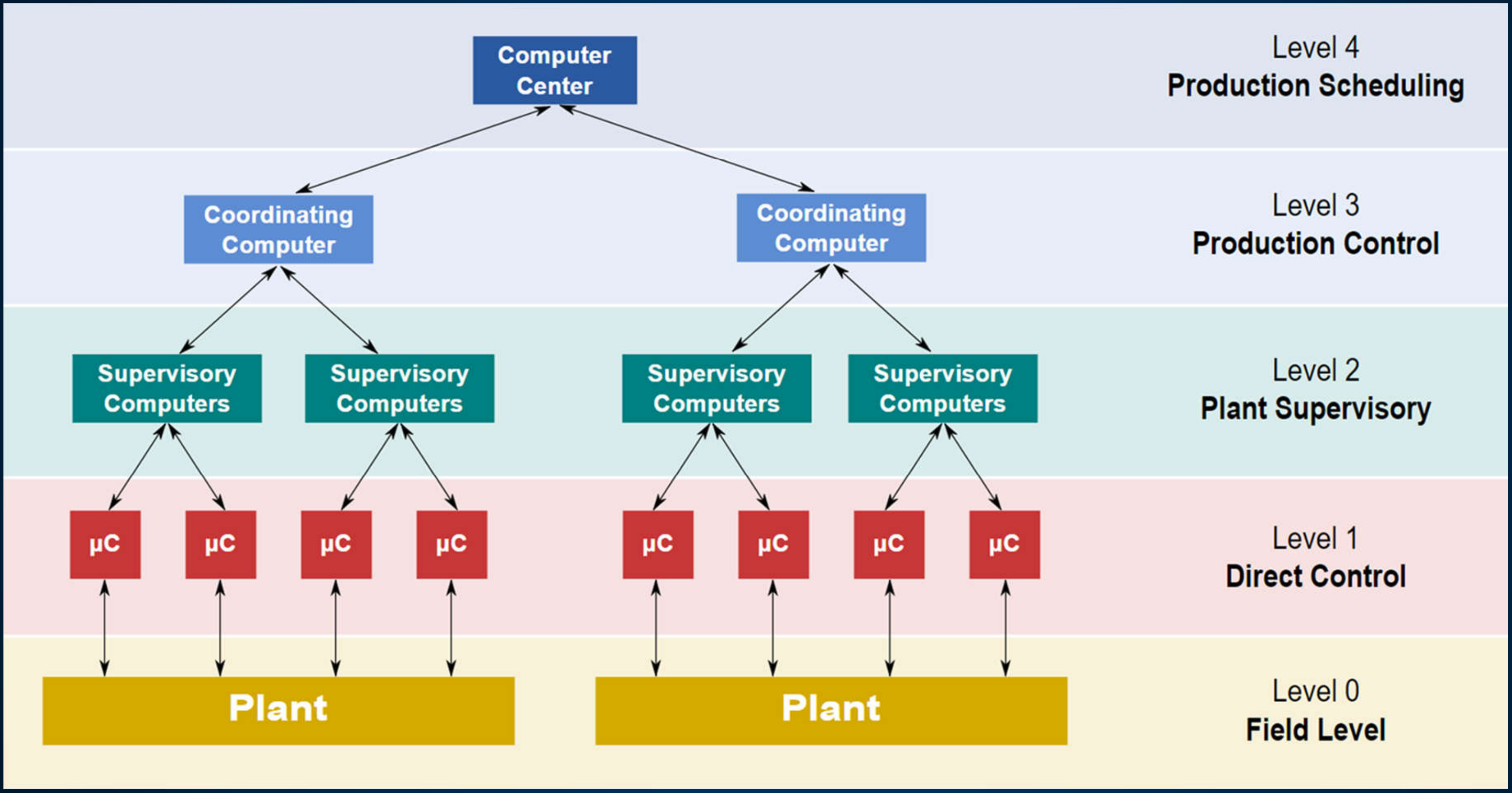
System	Control	Constraint	Interface
Rules based systems	Predicate System	Predefined behavior; does not handle unexpected	Sensors for events Administrators for behavior
ANN - Artificial neural network	Fuzzy Activation Logic	Behavior always defined not always expected	Trained reactive interface
DCS - Distributed Control System	Hierarchical	Conflicts in behavior may occur	Multi-layered
HMI - Human Machine Interface	Person	Interface must expose behavioral options through UI	Human Controls all actions
PLC - Programmable Logic Controller	Automatic	Predefined in the field but able to be repurposed	Software/Hardware Defined

Supervisory control an data acquisition (SCADA) is a control system architecture concept that uses computers, networked data communications and graphical user interfaces for high-level process supervisory management, but uses other peripheral devices such as programmable logic controllers and discrete PID controllers to interface to the process plant or machinery.

ARTIFICIAL NEURAL NETWORKS



DISTRIBUTED CONTROL SYSTEM



Computer Center

Coordinating Computer

Coordinating Computer

Supervisory Computers

Supervisory Computers

Supervisory Computers

Supervisory Computers

μC μC μC μC μC μC μC μC

Plant

Plant

Level 4
Production Scheduling

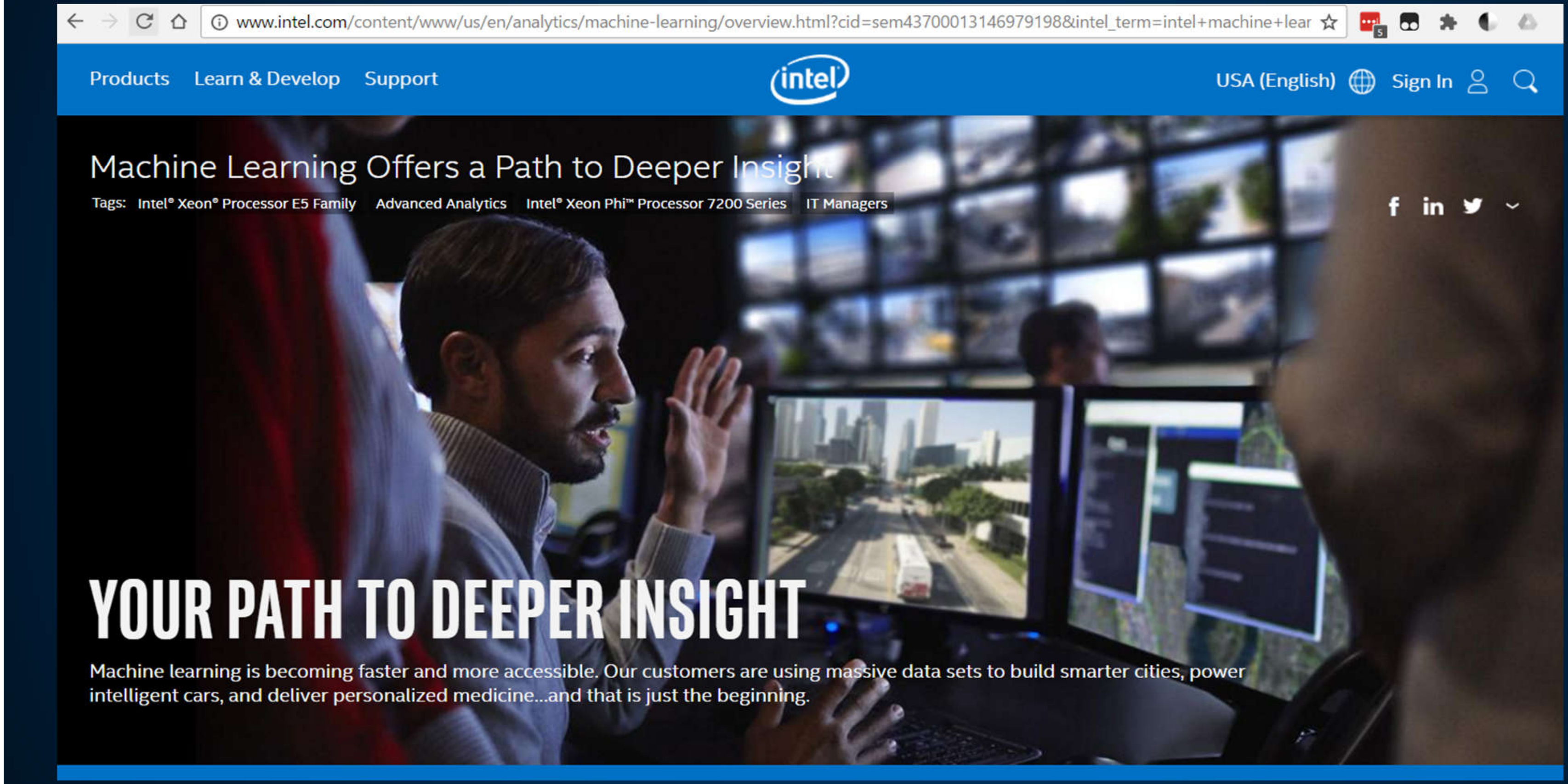
Level 3
Production Control

Level 2
Plant Supervisory

Level 1
Direct Control

Level 0
Field Level

INTEL OFFERS SOFTWARE AND DOCUMENTATION FOR MACHINE LEARNING



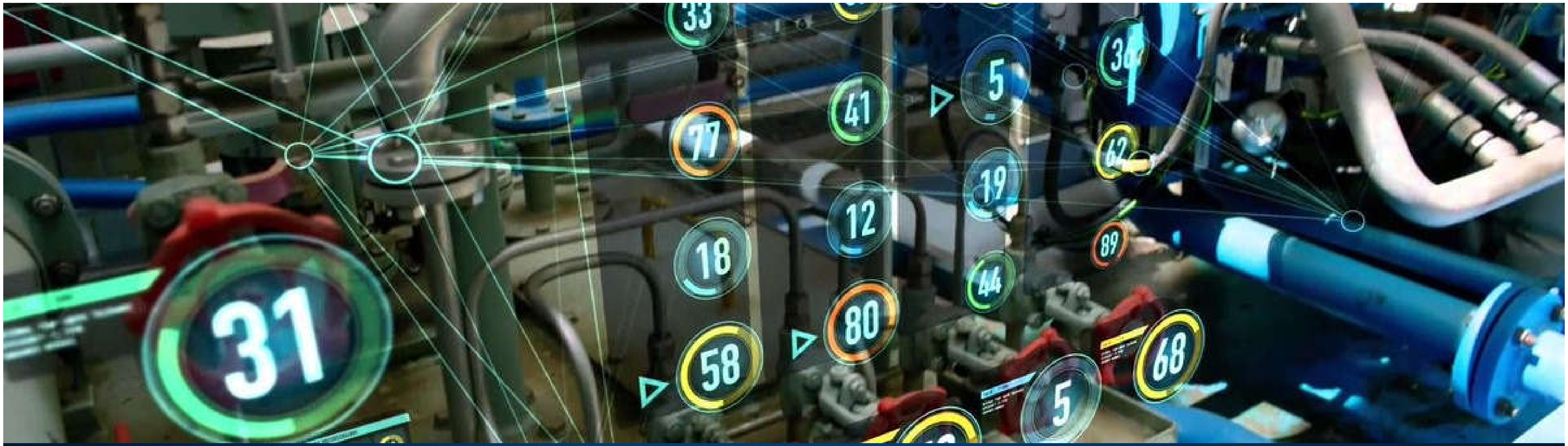
www.intel.com/content/www/us/en/analytics/machine-learning/overview.html?cid=sem43700013146979198&intel_term=intel+machine+lear
Products
Learn & Develop
Support
intel
USA (English)
Sign In

Machine Learning Offers a Path to Deeper Insight

Tags: Intel® Xeon® Processor E5 Family Advanced Analytics Intel® Xeon Phi™ Processor 7200 Series IT Managers

YOUR PATH TO DEEPER INSIGHT

Machine learning is becoming faster and more accessible. Our customers are using massive data sets to build smarter cities, power intelligent cars, and deliver personalized medicine...and that is just the beginning.

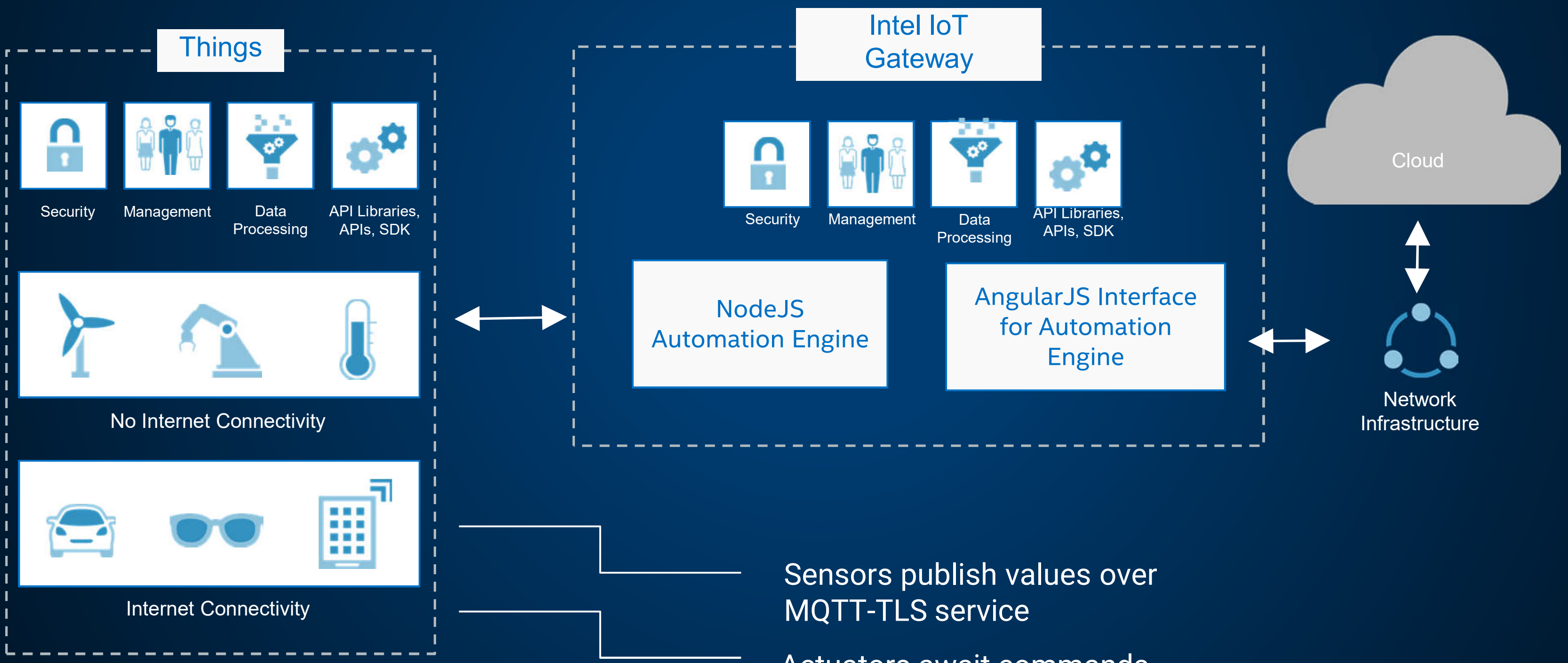


CREATING A RULE BASED SYSTEM

Lamdba Functions, Serverless Architecture

DEPLOYMENT OF AUTOMATION ENGINE

Closed loop management is the ability of an IoT Edge Network to monitor sensors and automatically trigger actions based on predefined rules.



Things

Security

Management

Data Processing

API Libraries, APIs, SDK

No Internet Connectivity

Internet Connectivity

Intel IoT Gateway

Security

Management

Data Processing

API Libraries, APIs, SDK

NodeJS Automation Engine

AngularJS Interface for Automation Engine

Cloud

Network Infrastructure

Sensors publish values over MQTT-TLS service

Actuators await commands via RESTful HTTPS

REQUIREMENTS OF AUTOMATION ENGINE

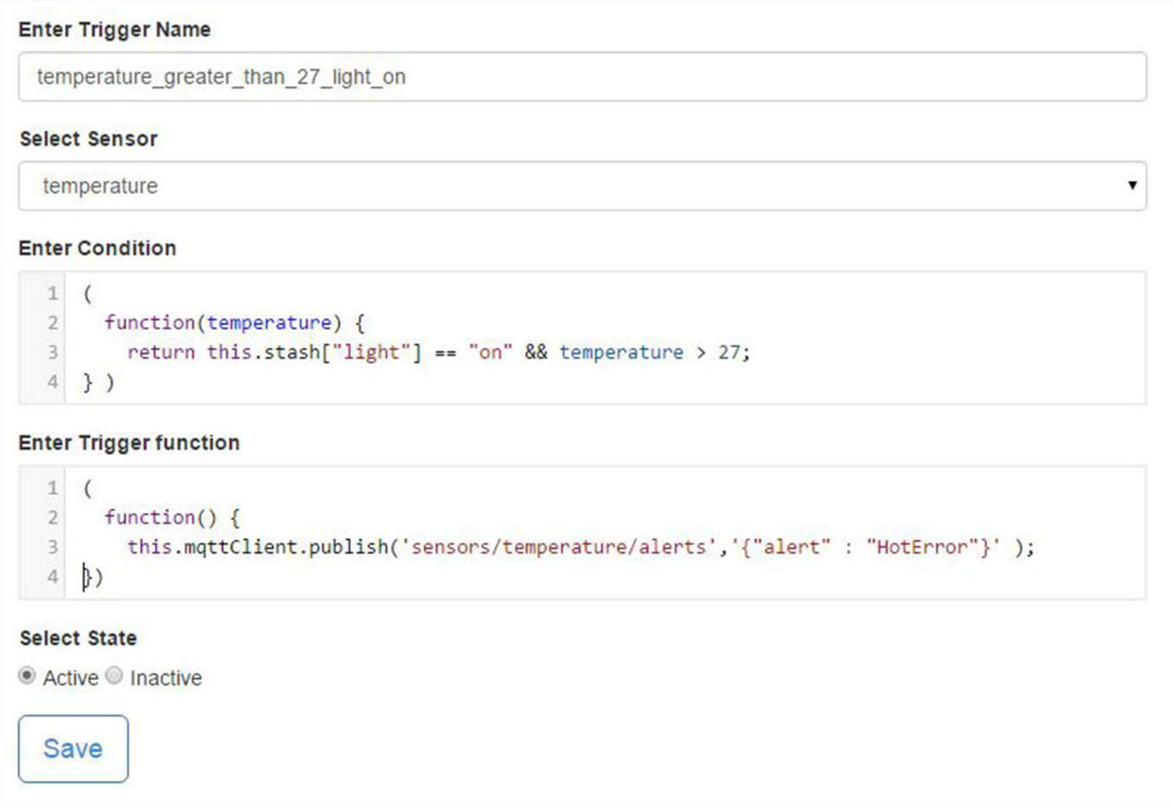
- Accessibility to a store which contains the business rules and policy actions for the IoT edge network. This is probably a database, but could be simple like a file or complex like a blockchain.
- Subscribes to all sensor data traffic on the edge network
- As sensor data arrives, all automation rules associated with that sensor have their predicate functions evaluated.
- If the predicate function evaluates to True then the action function is run.

What is a Trigger?

A trigger is a predicate (condition) function and an action function that implements the rule based business logic on the edge network.

The predicate function returns a true or false, a “Go” or “No Go”

In our implementation, the stash always holds the last sensor reading from each sensor.



Enter Trigger Name

temperature_greater_than_27_light_on

Select Sensor

temperature ▾

Enter Condition

```
(
  function(temperature) {
    return this.stash["light"] == "on" && temperature > 27;
} )
```

Enter Trigger function

```
(
  function() {
    this.mqttClient.publish('sensors/temperature/alerts','{"alert" : "HotError"}' );
})
```

Select State

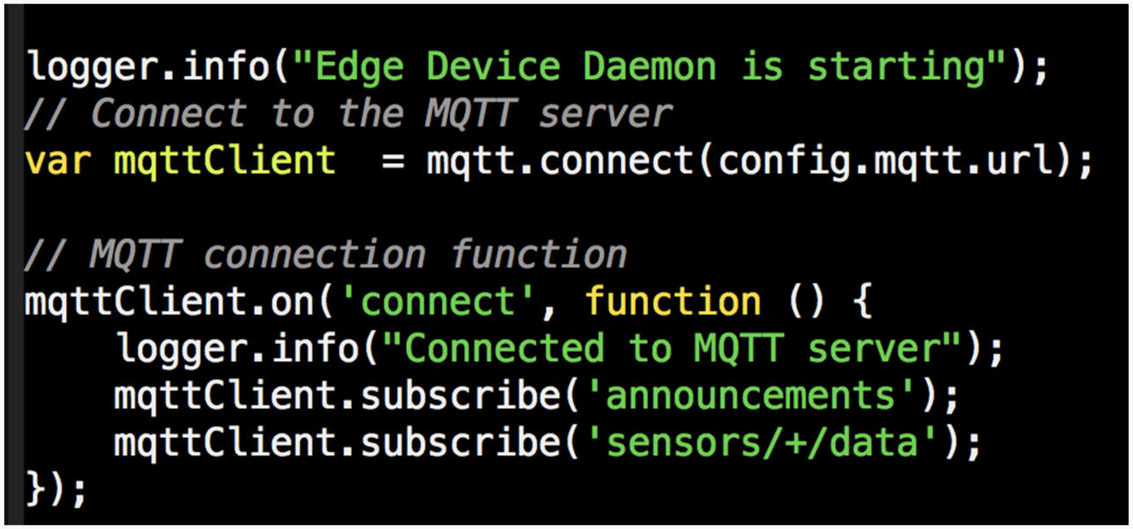
● Active ○ Inactive

Device Triggers Actions

There are many possible events that an action function could trigger.

- Write Data to the LCD screen
- Restart the device
- Execute an application
- Run a script
- Request data item from the device
- Send an SMS messages to an administrator.
- Send time synchronization
- Send an alert to the cloud

Creating a Connection to the MQTT Broker



```
logger.info("Edge Device Daemon is starting");
// Connect to the MQTT server
var mqttClient  = mqtt.connect(config.mqtt.url);

// MQTT connection function
mqttClient.on('connect', function () {
    logger.info("Connected to MQTT server");
    mqttClient.subscribe('announcements');
    mqttClient.subscribe('sensors/+/data');
});
```

NodeJS has a NPM package for MQTT.

var mqtt = require('mqtt') will import the package

MQTT has an event driven loop that allows you to customize your response to different events.

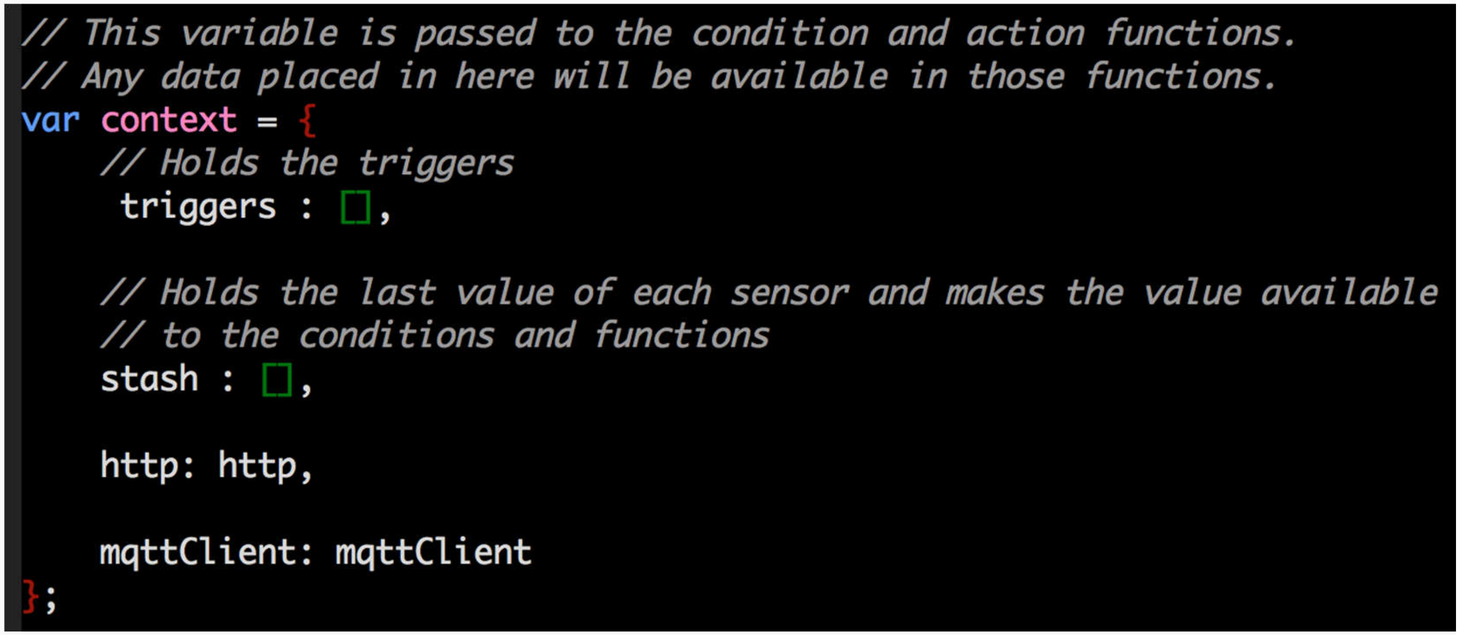
On the Connect Event, we log that we connected and subscribe to the "announcement" topic

the "+" matches any string. We are subscribing to all sensors that

The Context Object

This object holds variables and libraries that will be made accessible to the condition and trigger functions.

If you want to use HTTP or MQTT in a trigger define them in the context object.



```
// This variable is passed to the condition and action functions.
// Any data placed in here will be available in those functions.
var context = {
    // Holds the triggers
     triggers : [],

    // Holds the last value of each sensor and makes the value available
    // to the conditions and functions
    stash : [],

    http: http,

    mqttClient: mqttClient
};
```

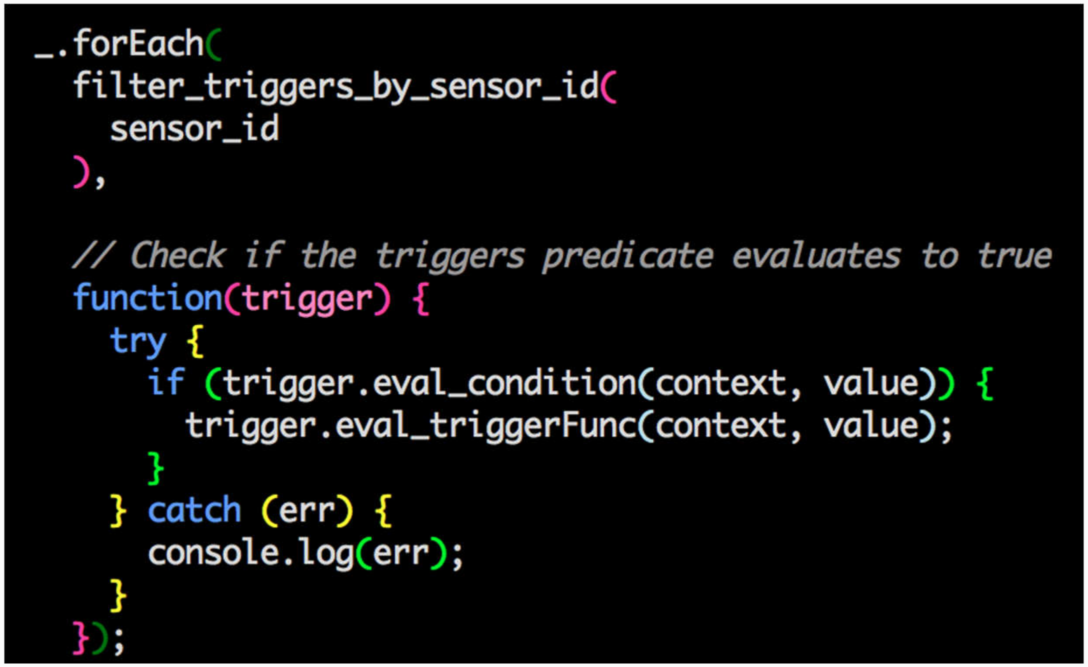
Rules Engines

For each incoming piece of data

filter the list of rules by the sensor that send the incoming data

and evaluate the condition function with the context

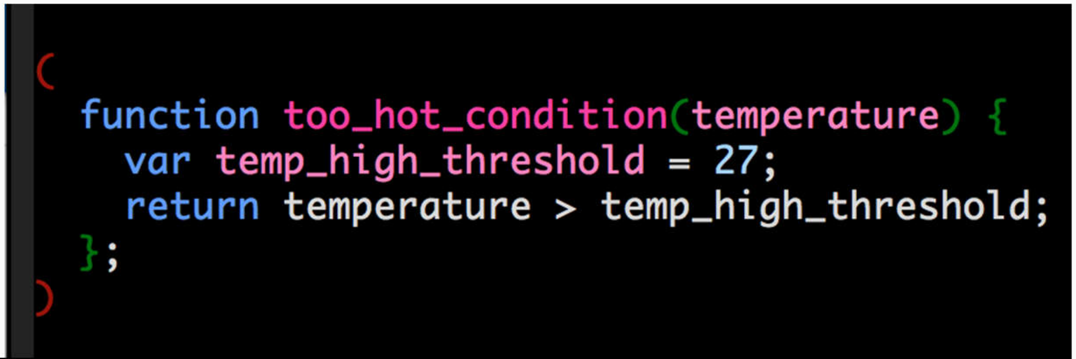
if it returns true then execute the trigger function



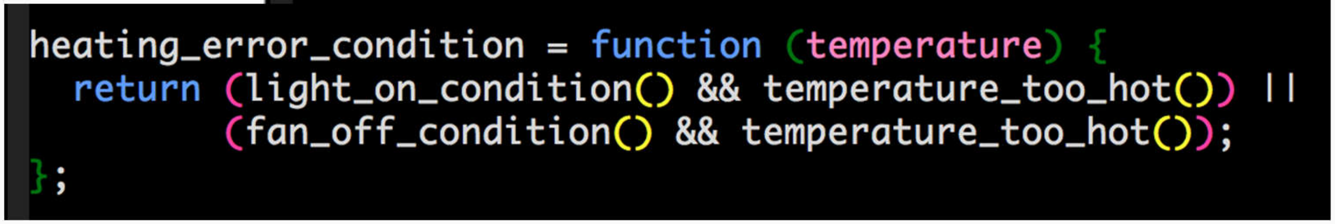
```
_.forEach(
  filter_triggers_by_sensor_id(
    sensor_id
  ),

  // Check if the triggers predicate evaluates to true
  function(trigger) {
    try {
      if (trigger.eval_condition(context, value)) {
        trigger.eval_triggerFunc(context, value);
      }
    } catch (err) {
      console.log(err);
    }
  });
```

Condition Functions

Conditions Functions, also called predicate functions, always return TRUE or FALSE.



```
(
  function too_hot_condition(temperature) {
    var temp_high_threshold = 27;
    return temperature > temp_high_threshold;
  };
)
```



```
heating_error_condition = function (temperature) {
  return (light_on_condition() && temperature_too_hot()) ||
         (fan_off_condition() && temperature_too_hot());
};
```

Example MQTT Trigger Function

Example Trigger Function

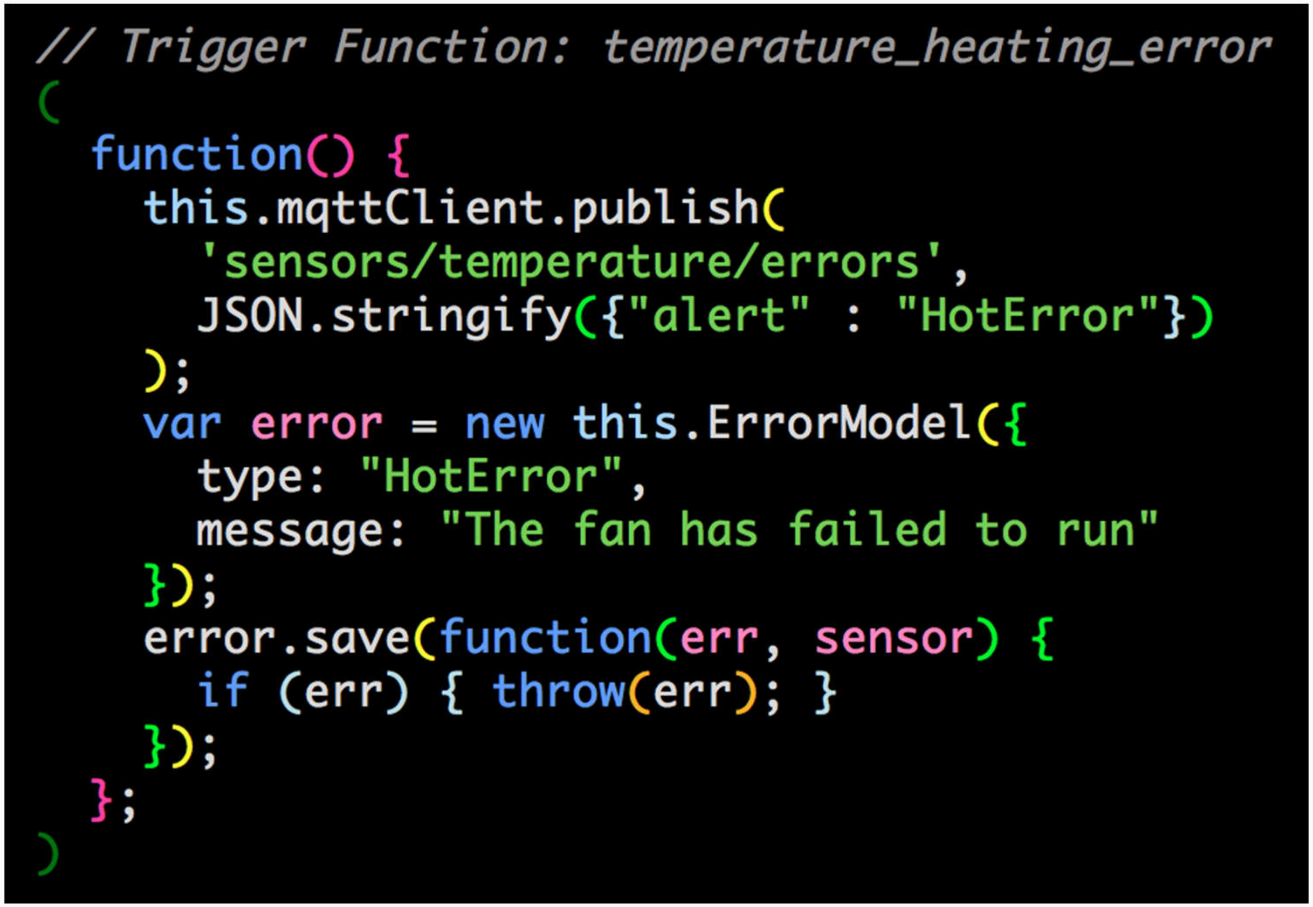
Publish an JSON error report to the MQTT topic **sensors/temperature/errors**

mqttClient and **ErrorModel** are part of the Context Object and now are accessible under the **this** object

Create a new MongoDB document called **error**.

Save error to the database

Be sure to put the parentheses around the entire function



```
// Trigger Function: temperature_heating_error
(
  function() {
    this.mqttClient.publish(
      'sensors/temperature/errors',
      JSON.stringify({"alert" : "HotError"})
    );
    var error = new this.ErrorModel({
      type: "HotError",
      message: "The fan has failed to run"
    });
    error.save(function(err, sensor) {
      if (err) { throw(err); }
    });
  };
)
```

Example HTTP Trigger Function

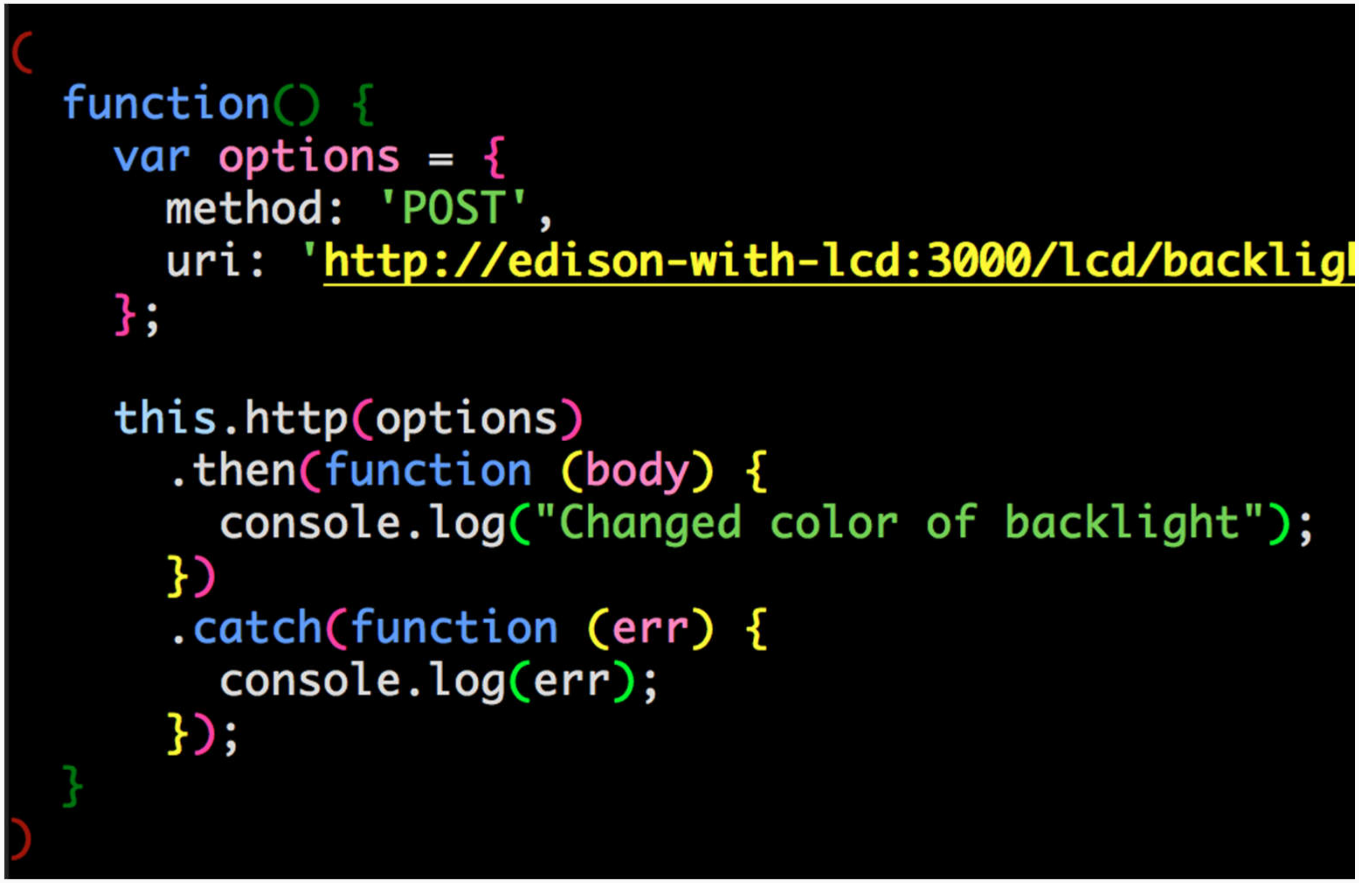
Example Trigger Function

Create an HTTP POST request for the Intel Edison with the LCD.

Send the HTTP request with the **this.http** object.

Write two functions to handle the success and error conditions.

Be sure to put the parentheses around the entire function



```
(
  function() {
    var options = {
      method: 'POST',
      uri: 'http://edison-with-lcd:3000/lcd/backlig
    };

    this.http(options)
      .then(function (body) {
        console.log("Changed color of backlight");
      })
      .catch(function (err) {
        console.log(err);
      });
  }
)
```

intel®