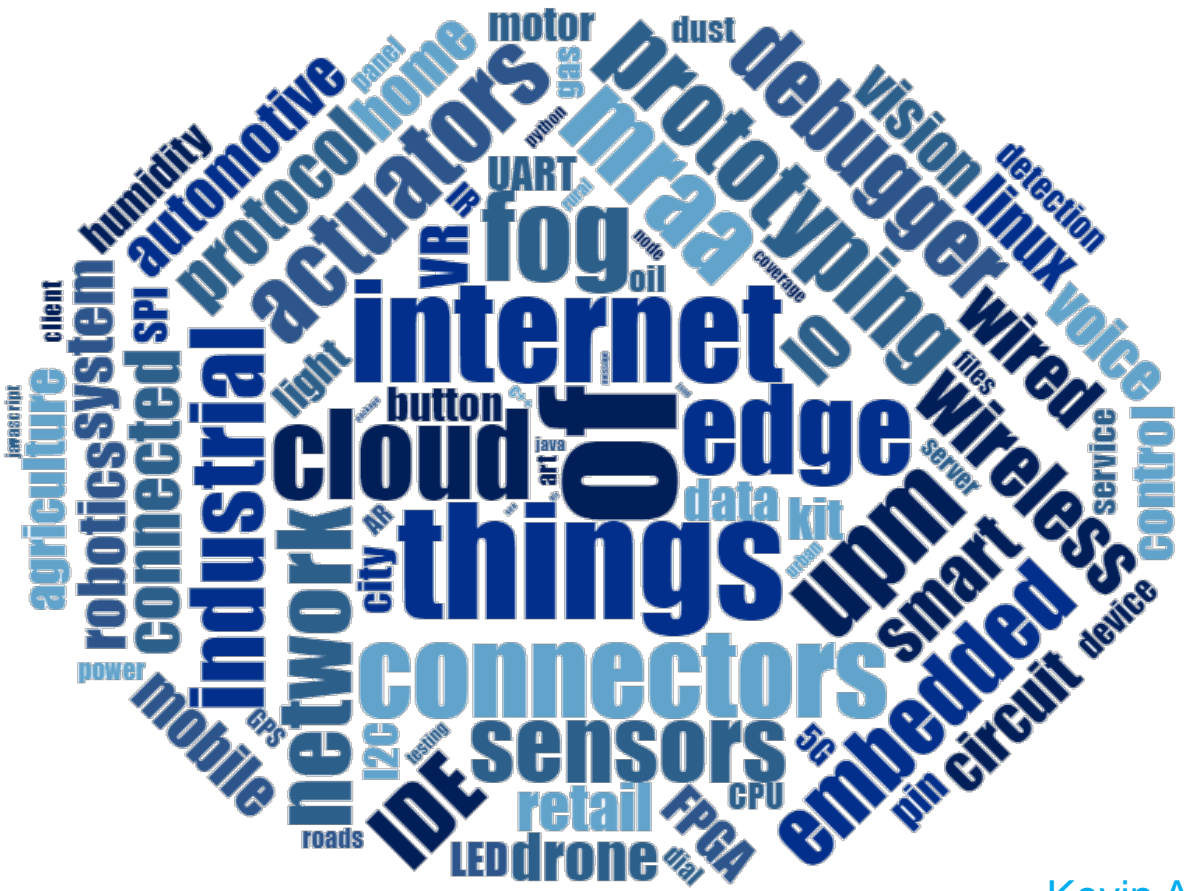
An Introduction to The Internet of Things

where and how to start

November 2017

Mihai Tudor Panu <mihai.tudor.panu@intel.com>



EST. 1999

Kevin Ashton, P&G

Agenda

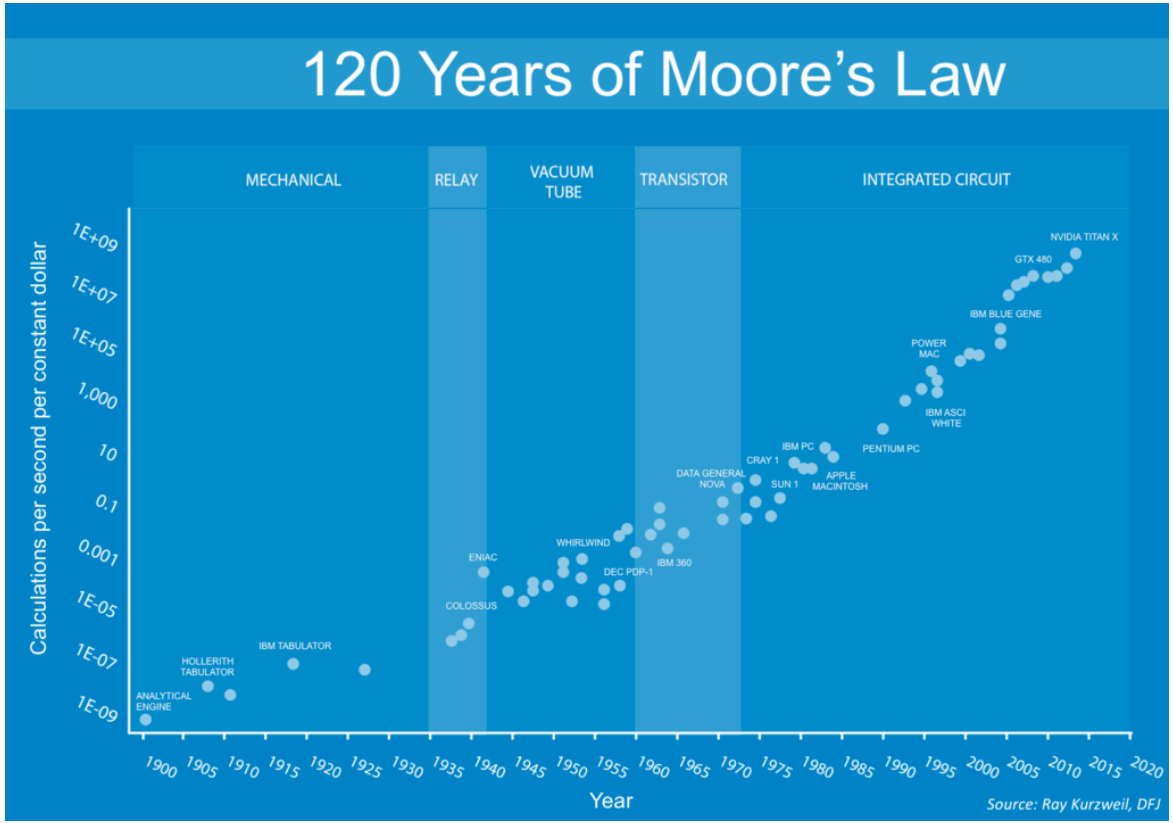
- High level key concepts surrounding IoT
- Easy to use examples and fields they cover
- Developing end-to-end solutions
  - Retail, Automotive, Home & City, Industrial, Computer Vision
- Hardware platforms and tools
- Software to get you going

Concepts

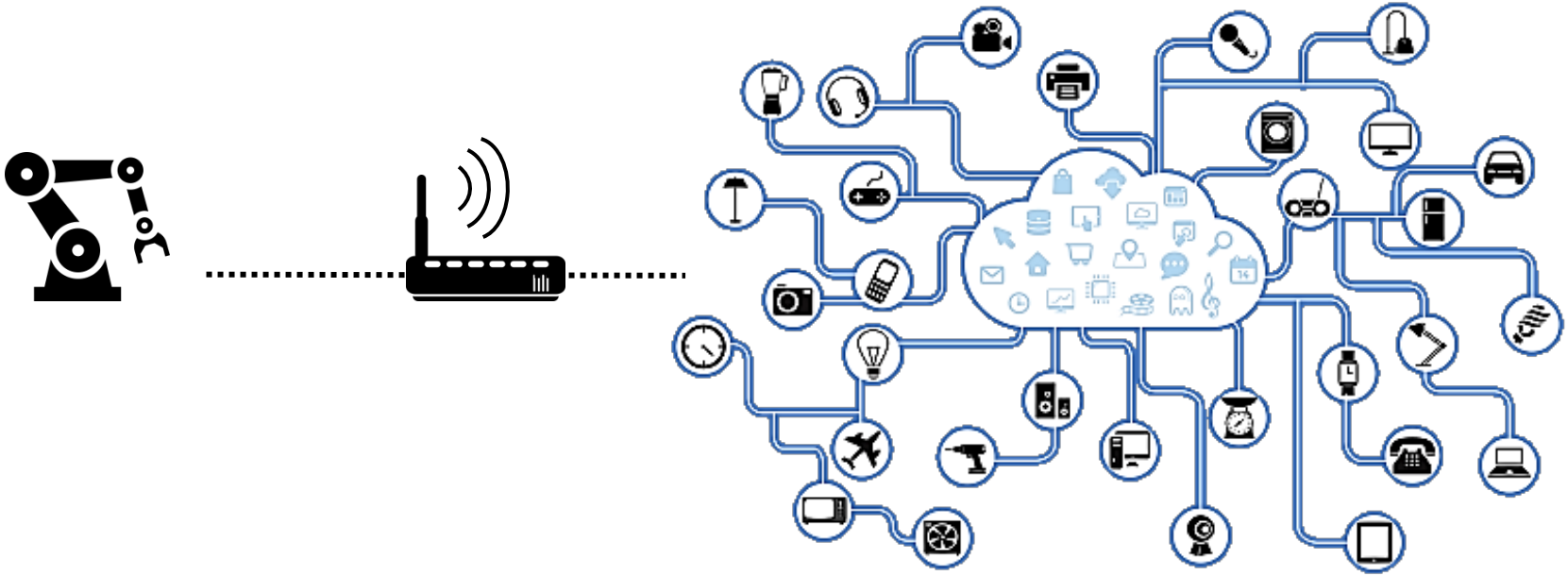
The IoT Enabler: Moore's Law

"The number of transistors in integrated circuits will double every 2 years."

Gordon Moore,
Intel cofounder



Typical IoT Solution Layout



Edge ⇔ Gateway ⇔ Cloud

More Powerful Edge – Fog

Microservers:

- Can connect to sensors directly and function as an edge devices too
- Plenty of compute power and storage gives more options for data aggregation and processing
- Ability to run and use private cloud solutions locally or distributed across sites
- Enhanced security due to dedicated hardware (e.g. crypto, TPM)
- Custom accelerator cards for advanced algorithms and data processing
- Great fit for Vision, AI, Machine Learning and Big Data

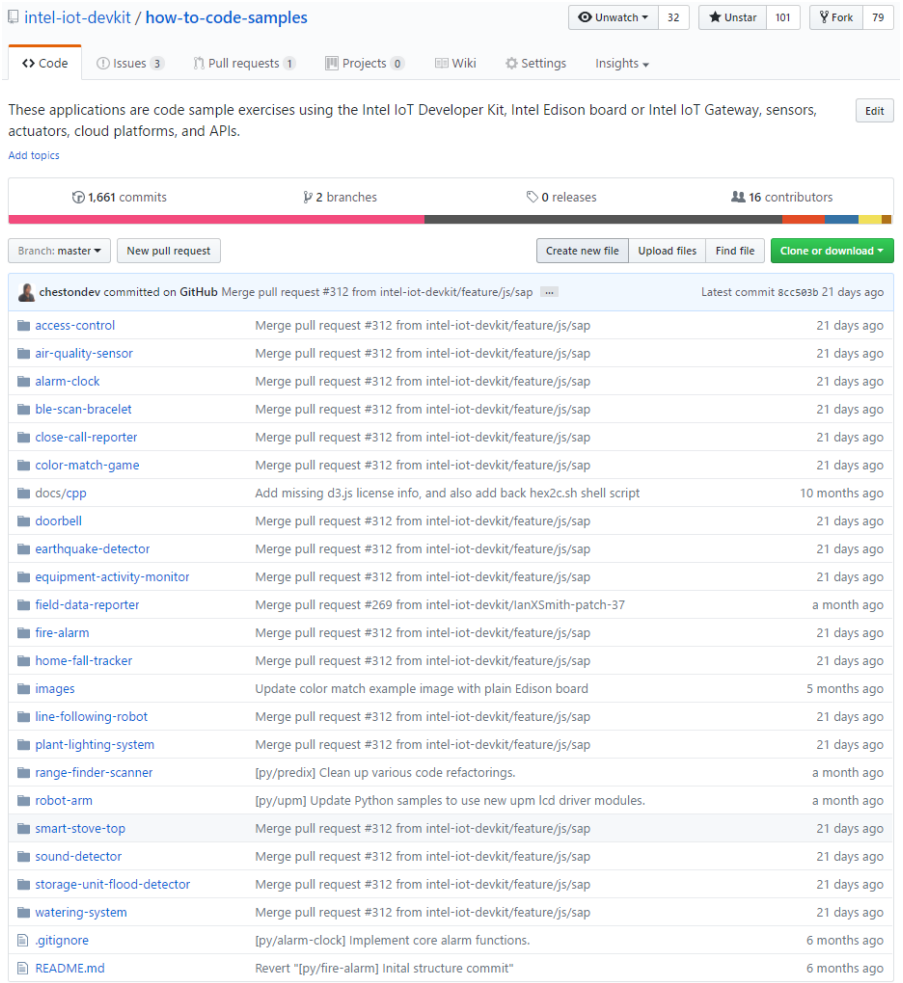
Example applications

How-to Code Samples

18 Complete how-to code samples in multiple programming languages:

- Myriad of IoT starter applications to explore
- Compatible with SeeedStudio, DFRobot sensor kits and many others
- Will run on any board with minimal code changes
- Learn how to use cloud services: AWS*, Bluemix*, Azure*, M2X*, Predix*, SAP*
- Different technologies for sending data: MQTT, REST

http://github.com/intel-iot-devkit/how-to-code-samples



How-to Code Samples

Where do they fit:

- Healthcare and assistance
  - Home Fall Tracker
- Security
  - Access Control
- Agriculture
  - Plant Lighting System
  - Watering System
- Robotics
  - Robot Arm
  - Line Following Robot
- Smart home & city
  - Alarm Clock
  - Doorbell
  - Earthquake Detector
  - Fire Alarm
  - Smart Stove Top
  - Storage Unit Flood Detector
  - Air Quality
- Industrial
  - Equipment Activity Monitor
  - Sound Detector
- Wearables
  - BLE Bracelet

http://github.com/intel-iot-devkit/how-to-code-samples

Developing End to end solutions

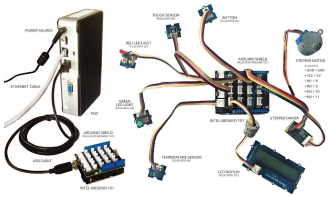
Intelligent Vending Machine

This retail vertical project monitors the inventory, product sales, and maintenance of a vending machine. The gateway gathers data from a temperature sensor, stepper motor, coil switch, and a product-purchasing application for edge data analytics. The prototype was created using the **Intel® IoT Developer Kit, Intel® IoT Gateway Software Suite, Grove* IoT Commercial Developer Kit, Intel® System Studio IoT Edition, and Microsoft* Azure*** cloud services and then deployed to an **Intel® IoT Gateway** using industrial sensors and a miniature scale vending machine.

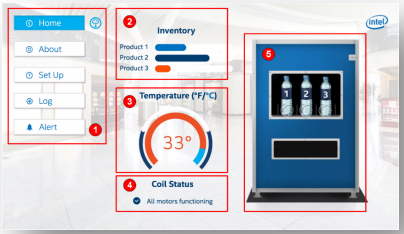
Articles:

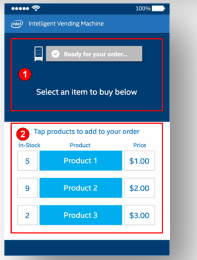
- "The Making Of" Story
- "How To" Build This Solution
- "Code" Available on GitHub

Prototype



Product





intel Software DEVELOPER ZONE https://software.intel.com/en-us/articles/path-to-product-the-making-of-an-intelligent-vending-machine

Transportation

This connected transportation project monitors the status of a refrigerated trailer. The gateway gathers data from a temperature and magnetic sensor for edge data analytics and monitoring. The prototype was created using the **Intel® IoT Developer Kit, Intel® IoT Gateway Software Suite, Grove* IoT Commercial Developer Kit, and Intel® System Studio IoT Edition**, and then deployed to an **Intel® IoT Gateway** using industrial sensors, **Intel® XDK** and a miniature scale truck trailer.

Articles:

- The "Making Of" Story
- How To Build This Solution
- Code Available on GitHub

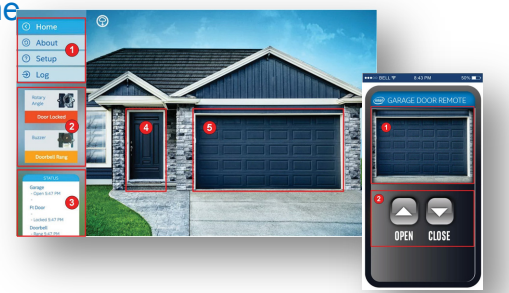
Prototype

Product

https://software.intel.com/en-us/articles/iot-path-to-product-the-making-of-a-connected-transportation-solution

Smart Home

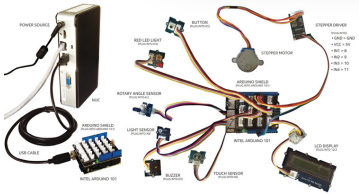
This smart home vertical project monitors the status of a home's front door and garage door for increased security. The gateway gathers data from a doorbell, door lock, stepper motors, and a garage door application for edge data analytics. The prototype was built using the **Intel® IoT Developer Kit, Intel® IoT Gateway Software Suite, Grove* IoT Commercial Developer Kit, Intel® XDK, and IBM Bluemix*** and then deployed to an **Intel® IoT Gateway** using industrial sensors and a miniature scale home.



Articles:

- The "Making Of" Story
- How To Build This Solution
- Code Available on GitHub

Prototype



Product

Intel Software DEVELOPER ZONE https://software.intel.com/en-us/articles/iot-path-to-product-smart-home

Environment Monitor

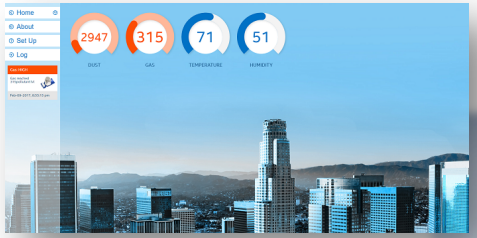
This smart building vertical project monitors air quality and pollutant levels in the surrounding environment. The gateway gathers data from a temperature and humidity sensor, a gas sensor, and a dust particle sensor. The solution was created using the **Intel® IoT Developer Kit, Grove* IoT Commercial Developer Kit, Intel® System Studio IoT Edition, and Amazon* AWS*** cloud services and specialized sensors. It was deployed to a portable enclosure.

Articles:

- "The Making Of" Story
- "How To" Build This Solution
- "Code" Available on GitHub

Explore

Deploy



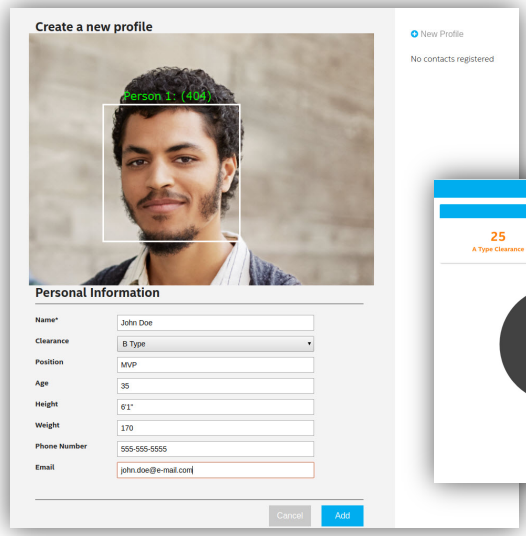
intel Software DEVELOPER ZONE https://software.intel.com/en-us/articles/iot-reference-implementation-making-of-an-environment-monitor-solution

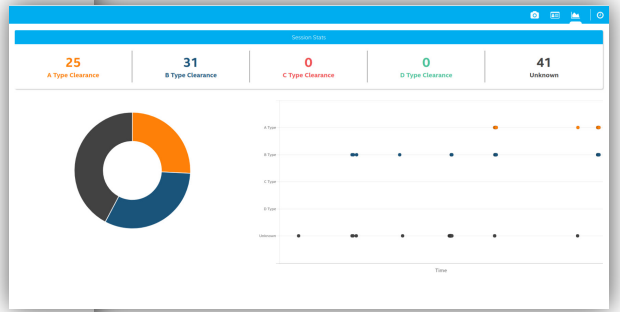
Face Access Control

The Face Access Control application uses facial recognition as the basis of a control system for granting physical access. The application detects and registers the image of a person's face into a database, recognizes known users entering a designated area and grants access if a person's face matches an image in the database.

Articles:

- "How To" Build This Solution
- "Code" Available on GitHub





intel Software DEVELOPER ZONE https://software.intel.com/en-us/articles/iot-reference-implementation-face-access-control

Hardware platforms

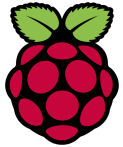
So Many Boards

Typical options:

- MCU
- FPGA
- CPU

101, UNO, MEGA, Leonardo, Yun, …

ARDUINO

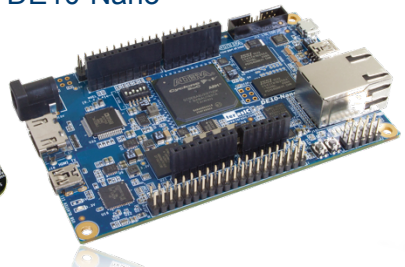


Raspberry Pi & Compatibles

Beaglebone

terasIC
www.terasic.com

DE10-Nano

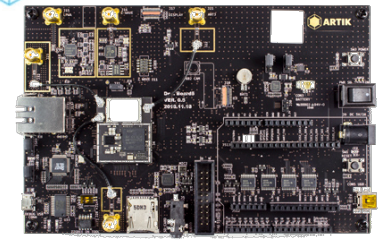


STM32 - Nucleo

LinkIt™ Smart 7688

LinkIt™
by MediaTek

Samsung ARTIK™



AAEON

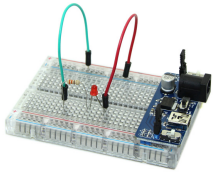
UP and UP Squared

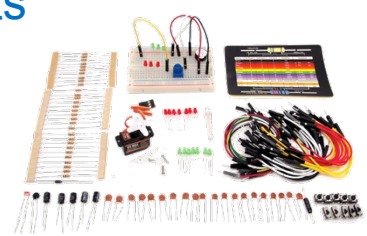
LATTEPANDA

What to Get?

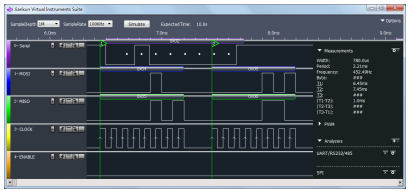
- Does your board come with a power supply?

- Breadboards, wires, cables, circuit parts





- Monitor the signals with a logic analyzer or oscilloscope



- Sensors, actuators, peripherals

Where to Get it?

- Prototyping – Platforms and Sensors

- Parts for tying it all together

DEVELOPER KITS

How-to Code Samples

- UP Squared IoT Grove Development Kit with GrovePi+ shield
- UP Squared is RPi compatible
- Full kit listing:

What's in the box
UP Squared board (Celeron N3350, 2GB RAM / 32GB eMMC)
Power supply 5V @ 6A
EU/US Power cord
16GB USB memory stick
Micro USB3.0 serial cable
Grove interface board
Grove LCD RGB backlight
Grove light sensor v1.2
Grove button
Grove temperature and humidity sensor
Grove green LED

http://up-shop.org

Reference Implementations

- Start with a prototype, then transition seamlessly to an industrial grade solution that can be taken to market
- Prototypes built with the **Grove* IoT Commercial Kit**, uses **Intel® NUC** as Gateway running **Intel® Gateway Software Suite** and **Arduino 101** as a sensor hub
- What's in the kit?

1 x
Intel® NUC Kit DE3815TYKE with 4G Memory

Sensors:
•1 x Grove* - Button
•1 x Grove* – Sound Sensor v1.2
•1 x Grove* – Touch Sensor
•1 x Grove* – Light Sensor v1.2
•1 x Grove* – Temperature Sensor v1.1
•1 x Grove* – Rotary Angle Sensor(P)
•1 x Grove* – Piezo Vibration Sensor

Others:
•1 x Grove* Base Shield v2
•14 x 26AWG Grove* Cable
•1 x B to A Type USB Cable – 0.5m

Actuators:
•1 x Grove* – LCD RGB Backlight
•1 x Grove* – Buzzer
•1 x Grove* – Red LED
•1 x Grove* – Green LED
•1 x Grove* – Blue LED
•1 x Gear Stepper Motor with Driver

SOFTWARE TO GET YOU GOING

mraa.io/demo

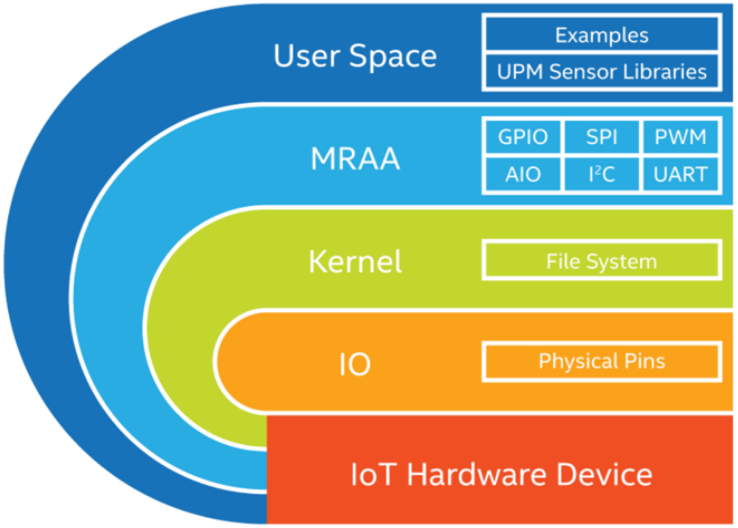
Before You Begin

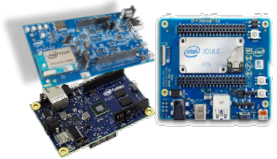
- Learn how to deploy OS images to devices:
  - Typical media: SD card or USB flash drive
  - Rufus, Win32 Disk Imager, dd (Linux)
- Familiarize yourself with tools that allow remote connections and transfers
  - Serial connections with Putty
  - SCP file transfers
  - VNC works great for desktop access on more powerful IoT platforms
- Find out how to view system logs and install new software on the target
- Create a Github account and start using git to get samples
- Bonus: Visit Docker and embrace virtualization

Hardware Abstraction (MRAA)

- Standard IO Interface for all IoT Developer Kits
- Supports Intel & non-Intel (community) platforms
- Abstraction APIs:
  - GPIO
  - Analog (AIO)
  - PWM
  - SPI
  - I²C
  - UART

Typical stack on UNIX systems:



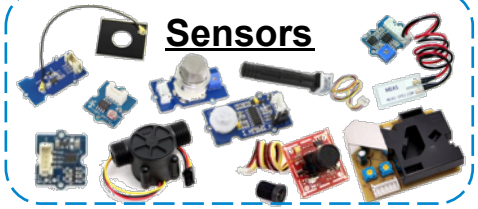


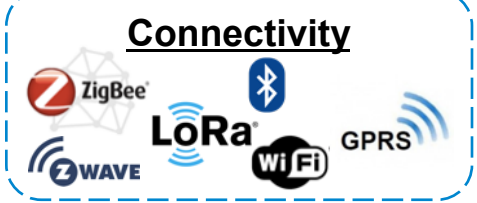
Legacy Intel® Boards

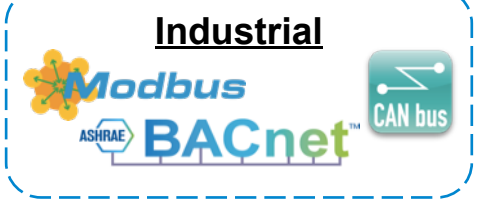
UP & UP2
MinnowBoard

Intel® IoT Gateways
& More

Standardized Sensor APIs (UPM)







Light | Temp | Gyro | Gas | Humidity | Accel | Other …

UPM C++ APIs

UPM C generic interfaces

GPIO | i2c | SPI | UART | PWM

MRAA C/C++ APIs

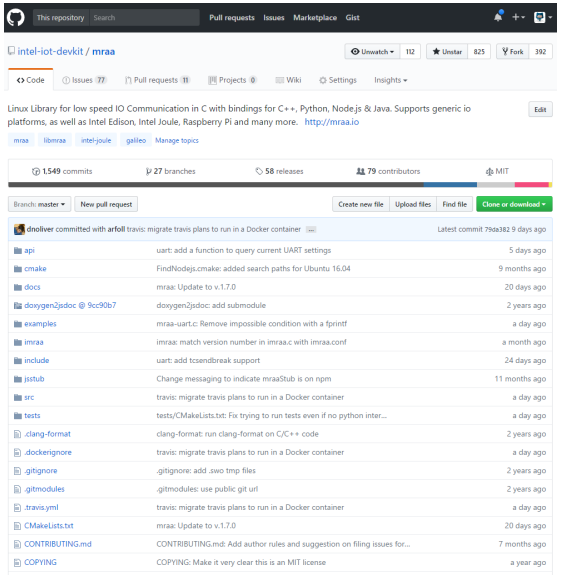
iio

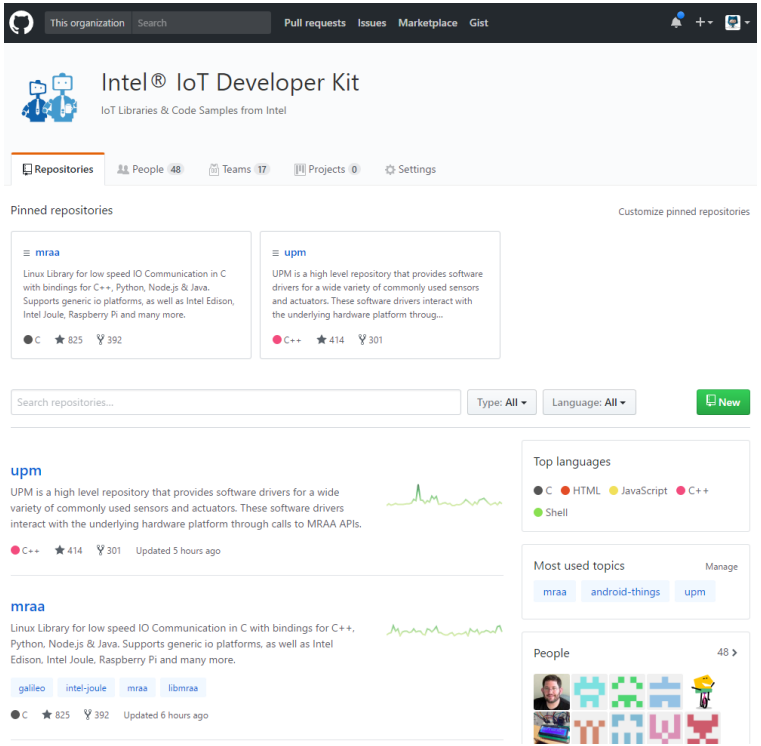
Libmodbus
Bacnet
Other…

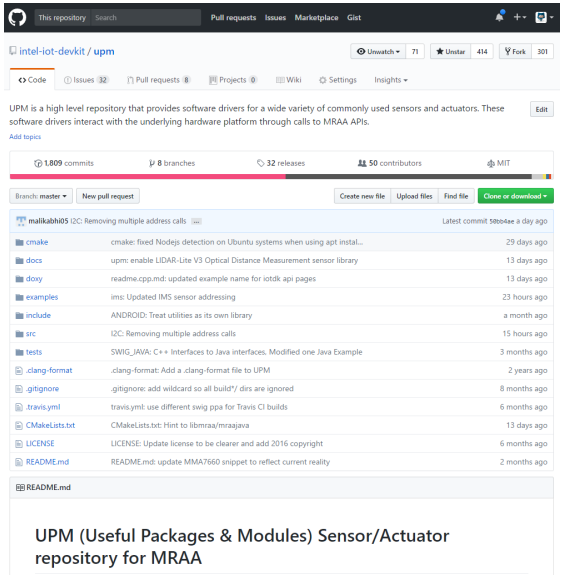
Linux Kernel

GitHub Repositories

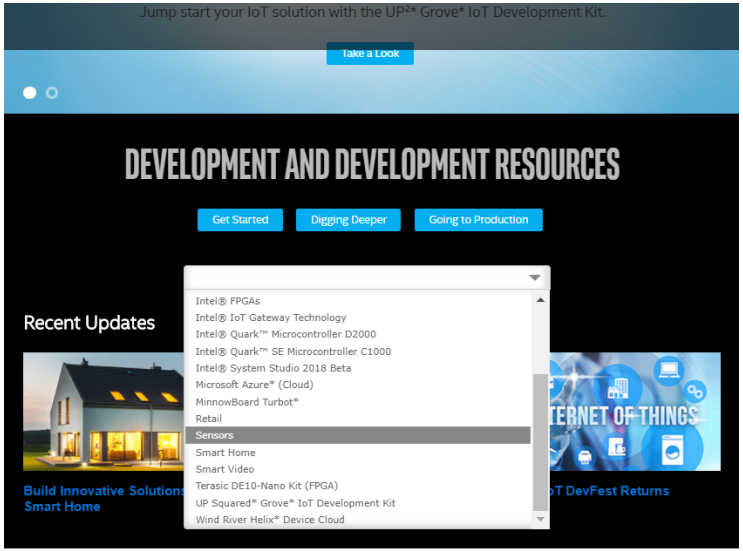
github.com/intel-iot-devkit

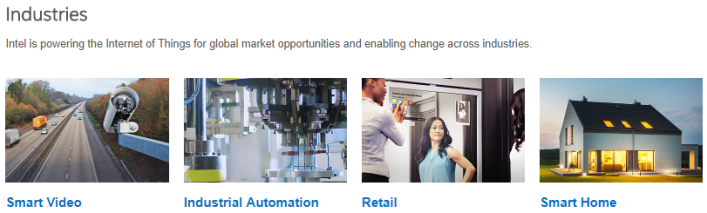




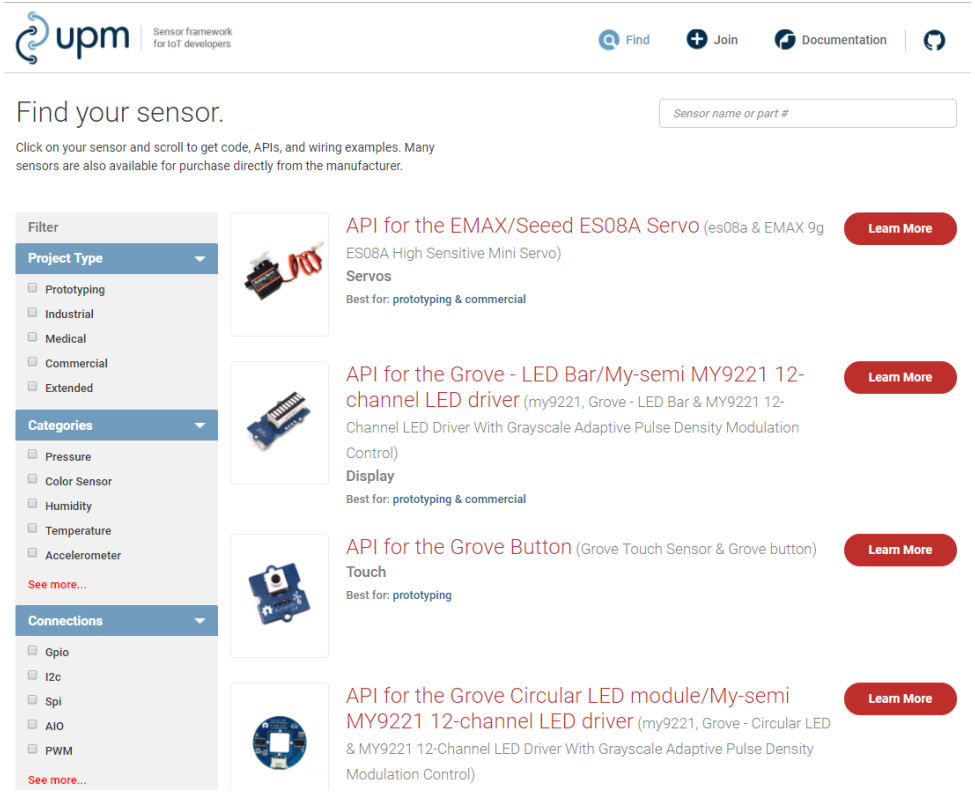


Intel Developer Zone - IoT





https://software.intel.com/en-us/iot



https://upm.mraa.io

Compatibility and Other Tools

Multiple OS support

Zephyr · androidthings · ubilinux · ubuntu · yocto PROJECT · WIND · archlinux · openSUSE

Multiple language support

C · C++ · Java · node.js · python · Lua

Integrated Development Environments (IDEs)

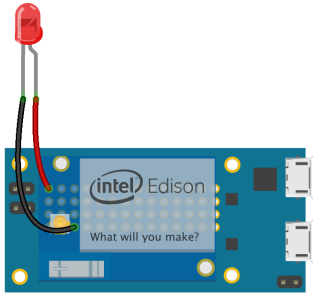
Build confidence

Easy to use, fun to learn

Create and debug full solutions

MRAA sample code for led blink

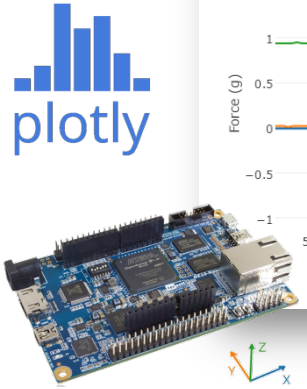
```
int
main(int argc, char** argv)
{
    if (argc < 2) {
        printf("Provide an int arg if you want to flash on something other than %d\n", DEFAULT_IOPIN);
        iopin = DEFAULT_IOPIN;
    } else {
        iopin = strtol(argv[1], NULL, 10);
    }

    signal(SIGINT, sig_handler);

    //! [Interesting]
    mraa::Gpio* gpio = new mraa::Gpio(iopin);
    if (gpio == NULL) {
        return mraa::ERROR_UNSPECIFIED;
    }
    mraa::Result response = gpio->dir(mraa::DIR_OUT);
    if (response != mraa::SUCCESS) {
        mraa::printError(response);
        return 1;
    }

    while (running == 0) {
        response = gpio->write(1);
        sleep(1);
        response = gpio->write(0);
        sleep(1);
    }
    delete gpio;
    return response;
    //! [Interesting]
}
```

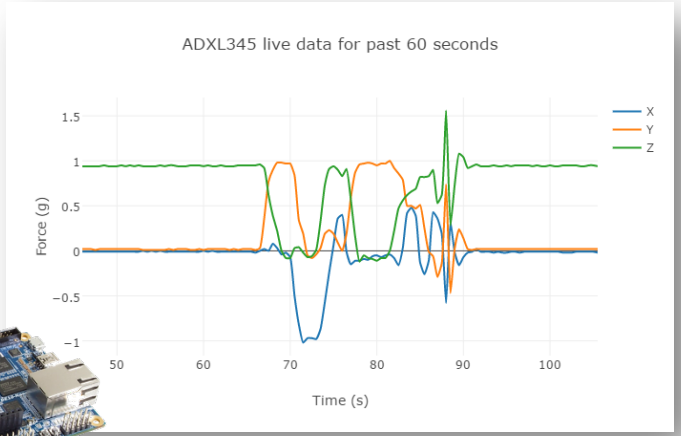



```
"use strict";

const mraa = require('mraa'); //require mraa
console.log('MRAA Version: ' + mraa.getVersion()); //write the mraa version to the console

let myLed = new mraa.Gpio(13); //LED hooked up to digital pin 13 (or built in pin on Galileo Gen1 & Gen2)
myLed.dir(mraa.DIR_OUT); //set the gpio direction to output
let ledState = true; //Boolean to hold the state of Led

function periodicActivity() {
    myLed.write(ledState ? 1 : 0); //if ledState is true then write a '1' (high) otherwise write a '0' (low)
    ledState = !ledState; //invert the ledState
}

setInterval(periodicActivity, 1000); //call the periodicActivity function every second
```

UPM sample code for ADXL345

```
int
main(int argc, char **argv)
{
//! [Interesting]
    int16_t *raw;
    float *acc;

    // Note: Sensor only works at 3.3V on the Intel Edison with Arduino breakout
    upm::Adxl345* accel = new upm::Adxl345(0);

    while(true){
        accel->update(); // Update the data
        raw = accel->getRawValues(); // Read raw sensor data
        acc = accel->getAcceleration(); // Read acceleration (g)
        fprintf(stdout, "Current scale: 0x%2xg\n", accel->getScale());
        fprintf(stdout, "Raw: %6d %6d %6d\n", raw[0], raw[1], raw[2]);
        fprintf(stdout, "AccX: %5.2f g\n", acc[0]);
        fprintf(stdout, "AccY: %5.2f g\n", acc[1]);
        fprintf(stdout, "AccZ: %5.2f g\n", acc[2]);
        sleep(1);
    }
//! [Interesting]
    return 0;
}
```






```
// Load accelerometer
var adxl345 = require('jsupm_adxl345');

// Instantiate on I2C bus
var adxl = new adxl345.Adxl345(0);

setInterval(function()
{
    adxl.update(); // Update the data
    var raw = adxl.getRawValues(); // Read raw sensor data
    var force = adxl.getAcceleration(); // Read acceleration force (g)
    var rawvalues = raw.getitem(0) + " " + raw.getitem(1) + " " + raw.getitem(2);
    console.log("Raw Values: " + rawvalues);
    console.log("ForceX: " + force.getitem(0).toFixed(2) + " g");
    console.log("ForceY: " + force.getitem(1).toFixed(2) + " g");
    console.log("ForceZ: " + force.getitem(2).toFixed(2) + " g");
}, 1000);
```

THANK YOU
Questions?


intel
experience
what's inside™