esc

EMBEDDED SYSTEMS CONFERENCE ‘15

Accelerate your next connected device prototype

21 July 2015

UBM

Accelerate Your Next Connected Device Prototype:

A look at ~~three~~ four different prototyping environments

Matt Liberty

Jetperch LLC

matt@jetperch.com

https://github.com/mliberty1/mcu_proto

M2M

Machine-to-machine (1968)

IoT

Internet of Things (1999), Internet of Everything

WoT

Web of Things (2007) – application layer for IoT

Skynet?

Terminator (1984), self-awareness in 1997

Prototyping

Device that emulates the final product’s functionality;
used to validate the product concept

Product Features of Growing Importance

- Internet connected
- Low cost
- Low power consumption
- Rapid feedback cycle: build/try/learn
- Rapid time to market

Connected Device Technologies

- WiFi (802.11)
- 802.15.4: Zigbee, 6LoWPAN (RFC 6282)
- Bluetooth SMART (formerly low energy)
- Bluetooth
- Ethernet and PoE
- WAN: 2G/3G/4G, LoRa and more

Prototyping Programming Environments

- Processing/Wiring (C++ like): Arduino, Energia
- Lua, eLua and Squirrel
- JavaScript: Espuirino, KinomaJS, WeIO
- Python and MicroPython: WiPy
- C/C++: mbed, Linux, FreeRTOS and many other RTOS's

Hardware

- Raspberry Pi & Beaglebone
- Atheros AR9331: Arduino Yun, WeIO, Black Swift, Onion
- TI CC3200 & CC3100
- ESP8266 (see hackaday, Arduino IDE port, nodemcu) ($6!)
- EMW3165 ($7.95)
- Electric Imp
- Particle (formerly Spark): Core, Photon, Electron
- Intel Edison
- … and many more …

Platforms

- Electric Imp (Imp001, Imp002, Imp003)
- Particle (Core, Photon, Electron)
- Thingsee
- TinkerForge
- SmartThings
- WICED (Broadcom)
- Cosino
- littleBits

Protocols

- MQTT
- ZeroMQ
- Thread (6LoWPAN on 802.15.4)
- Protocol Buffers
- HTTP & Websockets, often with JSON
- CoAP (RFC 7252)

Prototyping Options Summary

- No clear winning combination
- Many, many options
- My (somewhat arbitrary) criteria
  - Microcontroller
  - WiFi

https://github.com/mliberty1/mcu_proto

Example 1

Electric Imp

Credit: Electric Imp

Electric Imp

Credit: Electric Imp

- Programmed in Squirrel, comparable to Lua [cheatsheet]
- https://electricimp.com/docs/
- https://electricimp.com/docs/gettingstarted/quickstartguide/
- Imp001 with April breakout board
- Moto X: Failed BlinkUp first few times, eventually worked
- Working device running code in under 30 minutes

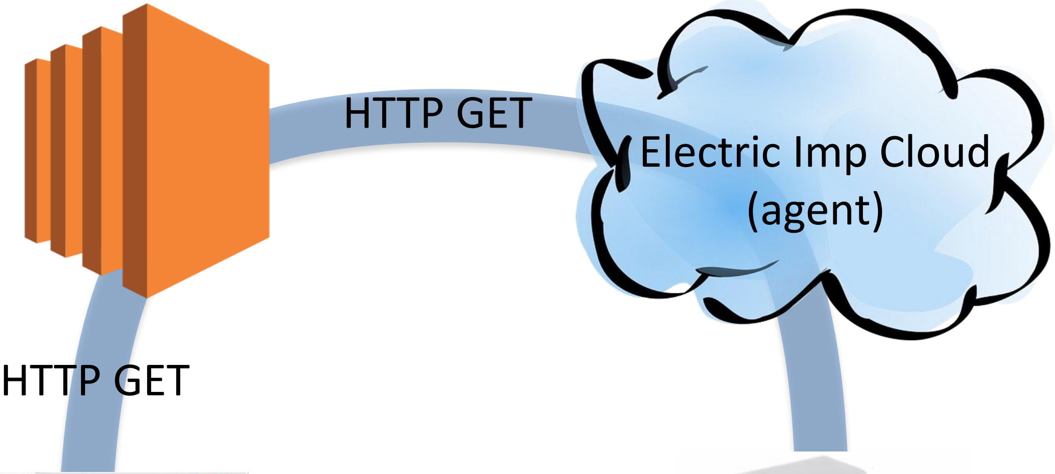
Electric Imp

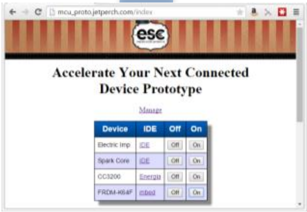
Credit: Electric Imp

- Online IDE and toolchain
- Event based architecture with devices, agents and apps
- Devices connect to agents
- Agents run on Electric Imp’s servers and talk to their device
- Apps talk to the agents using HTTP
- Imp API simplifies communication and message serialization

Architecture

mcu_proto.jetperch.com
Amazon EC2 instance




mcu_proto.jetperch.com/index

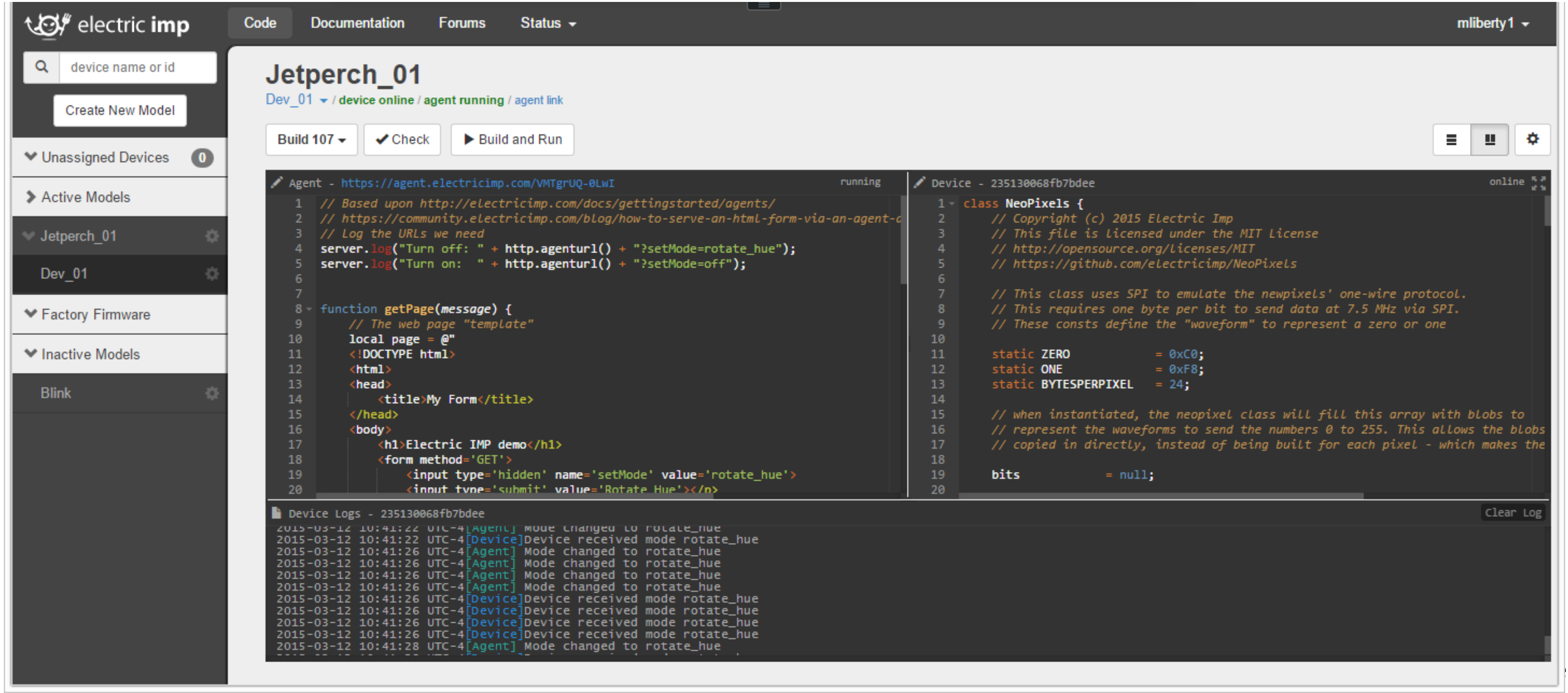
Accelerate Your Next Connected Device Prototype

Manage

Device	IDE	Off	On
Electric Imp	IDE	Off	On
Spark Core	IDE	Off	On
CC3200	Energia	Off	On
FRDM-K64F	mbed	Off	On

Electric Imp
(device)

Electric Imp Demo



```
Agent - https://agent.electricimp.com/VMTgrUQ-0LwI
// Based upon http://electricimp.com/docs/gettingstarted/agents/
// https://community.electricimp.com/blog/how-to-serve-an-html-form-via-an-agent-c
// Log the URLs we need
server.log("Turn off: " + http.agenturl() + "?setMode=rotate_hue");
server.log("Turn on:  " + http.agenturl() + "?setMode=off");


function getPage(message) {
    // The web page "template"
    local page = @"
    <!DOCTYPE html>
    <html>
    <head>
        <title>My Form</title>
    </head>
    <body>
        <h1>Electric IMP demo</h1>
        <form method='GET'>
            <input type='hidden' name='setMode' value='rotate_hue'>
            <input type='submit' value='Rotate Hue'></p>
```

```
Device - 235130068fb7bdee
class NeoPixels {
    // Copyright (c) 2015 Electric Imp
    // This file is licensed under the MIT License
    // http://opensource.org/licenses/MIT
    // https://github.com/electricimp/NeoPixels

    // This class uses SPI to emulate the newpixels' one-wire protocol.
    // This requires one byte per bit to send data at 7.5 MHz via SPI.
    // These consts define the "waveform" to represent a zero or one

    static ZERO             = 0xC0;
    static ONE              = 0xF8;
    static BYTESPERPIXEL    = 24;

    // when instantiated, the neopixel class will fill this array with blobs to
    // represent the waveforms to send the numbers 0 to 255. This allows the blobs
    // copied in directly, instead of being built for each pixel - which makes the

    bits            = null;

```

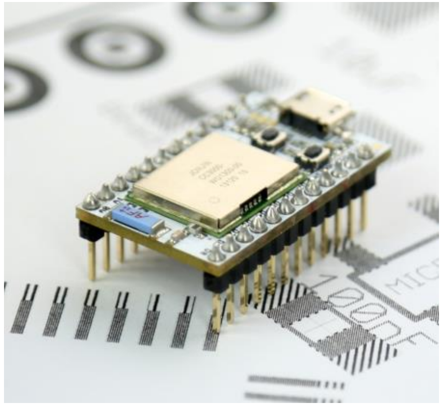
```
Device Logs - 235130068fb7bdee
2015-03-12 10:41:22 UTC-4[Agent] Mode changed to rotate_hue
2015-03-12 10:41:22 UTC-4[Device]Device received mode rotate_hue
2015-03-12 10:41:26 UTC-4[Agent] Mode changed to rotate_hue
2015-03-12 10:41:26 UTC-4[Agent] Mode changed to rotate_hue
2015-03-12 10:41:26 UTC-4[Agent] Mode changed to rotate_hue
2015-03-12 10:41:26 UTC-4[Agent] Mode changed to rotate_hue
2015-03-12 10:41:26 UTC-4[Device]Device received mode rotate_hue
2015-03-12 10:41:26 UTC-4[Device]Device received mode rotate_hue
2015-03-12 10:41:26 UTC-4[Device]Device received mode rotate_hue
2015-03-12 10:41:26 UTC-4[Device]Device received mode rotate_hue
2015-03-12 10:41:28 UTC-4[Agent] Mode changed to rotate_hue
```

Electric Imp: Thoughts

- Squirrel language is easy to learn and use
- Fast iteration time
- Well designed API which reduces complexity
- Great logging included
- Closed ecosystem
- Great documentation and examples

Example 2

Particle Core



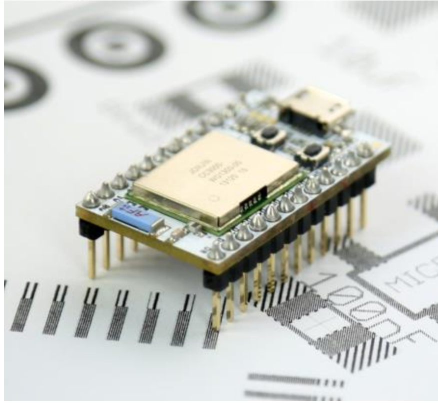
Credit: particle.io

Particle

Credit: particle.io

- Programmed in Wiring (C++ ish), same as Arduino
- Core: Cortex-M3 (STM32F103) with TI CC3000
- Photon (May): Cortex-M3 (STM32F205) with BCM43362
- Developed using node.js

Particle Core

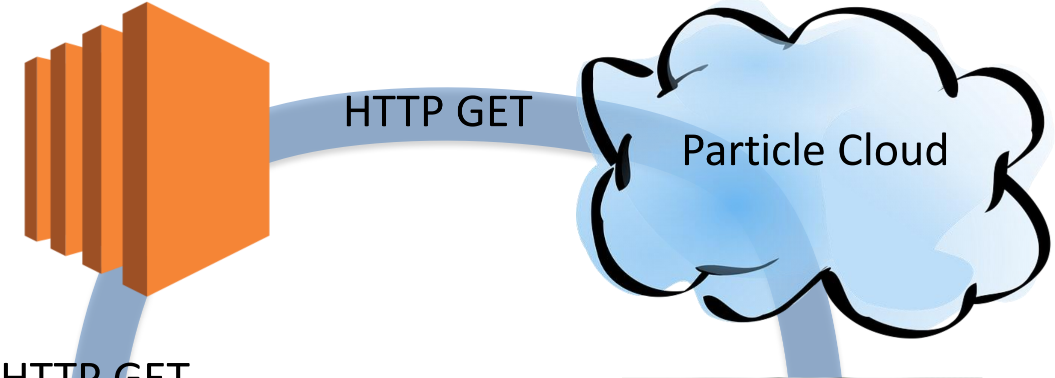


Credit: particle.io

- Android app on Moto X failed
- However, the CLI worked great
- Had a networked controlled LED in under 1 hour using website example

Architecture

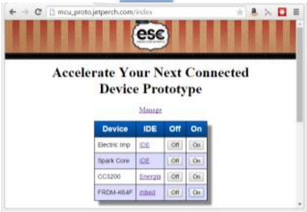
mcu_proto.jetperch.com
Amazon EC2 instance



HTTP GET

Particle Cloud

HTTP GET

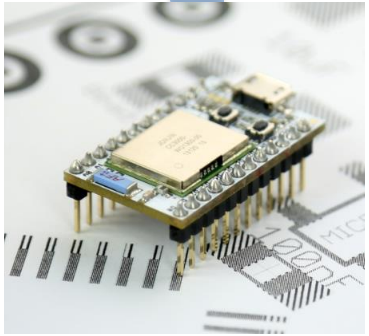


mcu_proto.jetperch.com/index

Accelerate Your Next Connected Device Prototype

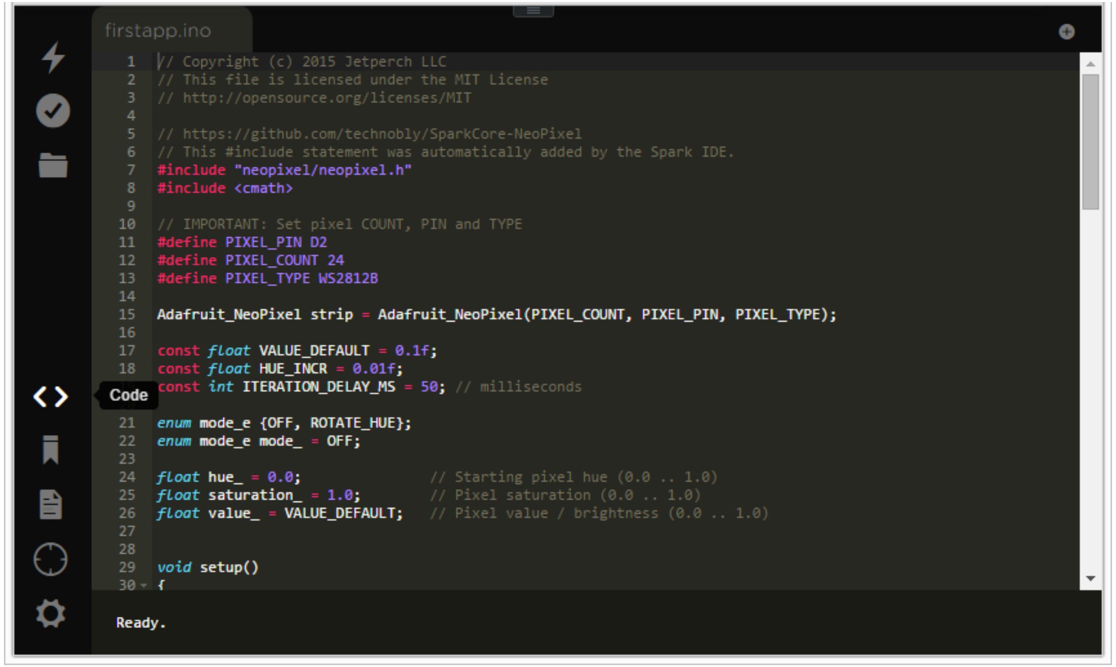
Manage

Device	IDE	Off	On
Electric Imp	IDE	Off	On
Spark Core	IDE	Off	On
CC3200	Energia	Off	On
FRDM-K64F	mbed	Off	On



ParticleCore
(device)

Particle Demo



```
// Copyright (c) 2015 Jetperch LLC
// This file is licensed under the MIT License
// http://opensource.org/licenses/MIT

// https://github.com/technobly/SparkCore-NeoPixel
// This #include statement was automatically added by the Spark IDE.
#include "neopixel/neopixel.h"
#include <cmath>

// IMPORTANT: Set pixel COUNT, PIN and TYPE
#define PIXEL_PIN D2
#define PIXEL_COUNT 24
#define PIXEL_TYPE WS2812B

Adafruit_NeoPixel strip = Adafruit_NeoPixel(PIXEL_COUNT, PIXEL_PIN, PIXEL_TYPE);

const float VALUE_DEFAULT = 0.1f;
const float HUE_INCR = 0.01f;
const int ITERATION_DELAY_MS = 50; // milliseconds

enum mode_e {OFF, ROTATE_HUE};
enum mode_e mode_ = OFF;

float hue_ = 0.0;               // Starting pixel hue (0.0 .. 1.0)
float saturation_ = 1.0;        // Pixel saturation (0.0 .. 1.0)
float value_ = VALUE_DEFAULT;   // Pixel value / brightness (0.0 .. 1.0)


void setup()
{
```

Ready.

Particle Thoughts

- Slow iteration time (>30 seconds): flash firmware, reboot, reconnect to WiFi
- Documentation and examples are not as good as Electric Imp
- Support for both cloud and local development
- Open source with active community

Example 3

CC3200 with Energia

Credit: ti.com

CC3200 with Energia

- Energia is Wiring (of Arduino fame) for CC3200
- CC3200 is an SoC with a CC3100 Simplink WiFi and Cortex M4
- Just a device, not a full IoT framework

Credit: ti.com

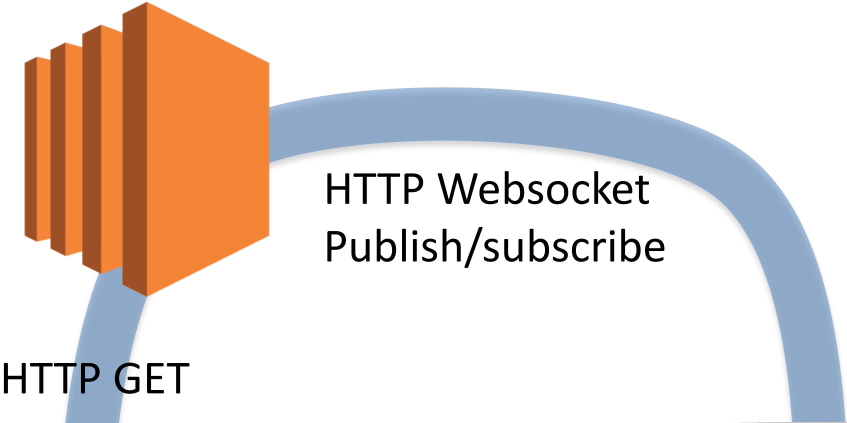
CC3200 with Energia

- Setup
  - See http://energia.nu/pin-maps/guide_cc3200launchpad/
  - See http://energia.nu/cc3200guide/
  - See http://processors.wiki.ti.com/index.php/CC31xx_%26_CC32xx
  - Installed SDK with FTDI drivers
  - Installed UniFlash & programmed latest service pack
  - Unzip Energia, configured board, serial port, & downloaded examples
- Up an running with basic WiFi examples in a under 2 hours

CC3200 with Energia

- Need server: use Amazon EC2 instance running Ubuntu server
- Use websockets (alternatives include MQTT, HTTP/AJAX)
- Implement server using Python3
  - CherryPy: Web framework for python (not recommended, use flask)
  - ws4py: Websockets implementation that supports CherryPy
  - Jinja2: Templating engine

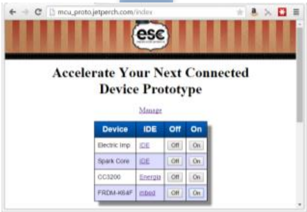
Architecture



mcu_proto.jetperch.com
Amazon EC2 instance

HTTP Websocket
Publish/subscribe

HTTP GET



mcu_proto.jetperch.com/index

Accelerate Your Next Connected Device Prototype

Manage

Device	IDE	Off	On
Electric Imp	IDE	Off	On
Spark Core	IDE	Off	On
CC3200	Energia	Off	On
FRDM-K64F	mbed	Off	On

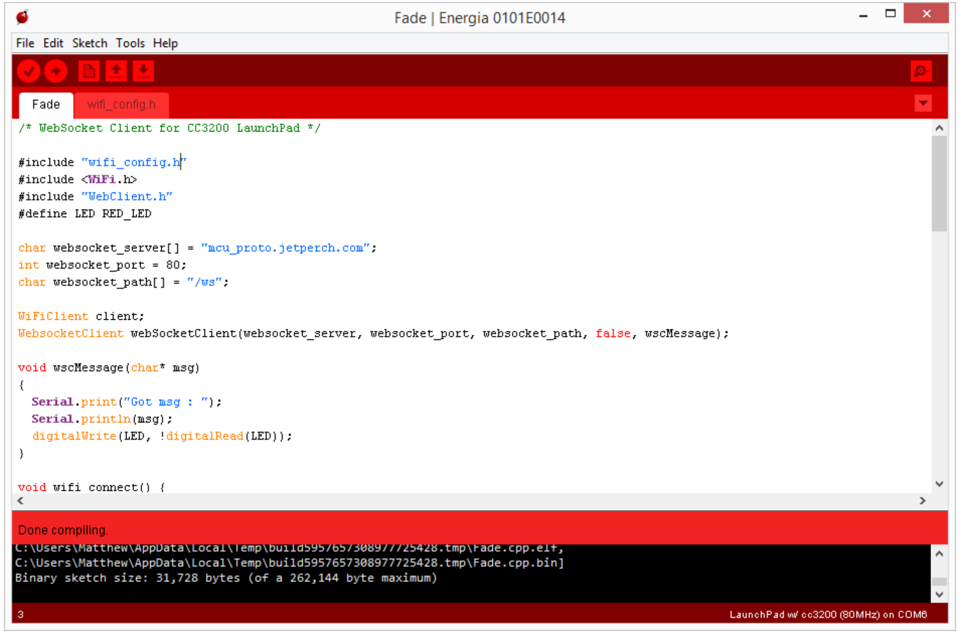
CC3200
(device)

Amazon EC2 Instance

- Configure and start a Linux or Windows server in minutes
- Use SSH/SCP/SFTP to control
- If you have never started a virtual internet instance, take the time (EC2 has a free tier for the first year)

See server code at
https://github.com/mliberty1/mcu_proto/blob/master/server/server.py

CC3200 Demo



```
/* WebSocket Client for CC3200 LaunchPad */

#include "wifi_config.h"
#include <WiFi.h>
#include "WebClient.h"
#define LED RED_LED

char websocket_server[] = "mcu_proto.jetperch.com";
int websocket_port = 80;
char websocket_path[] = "/ws";

WiFiClient client;
WebsocketClient webSocketClient(websocket_server, websocket_port, websocket_path, false, wscMessage);

void wscMessage(char* msg)
{
  Serial.print("Got msg : ");
  Serial.println(msg);
  digitalWrite(LED, !digitalRead(LED));
}

void wifi_connect() {
```

Done compiling.

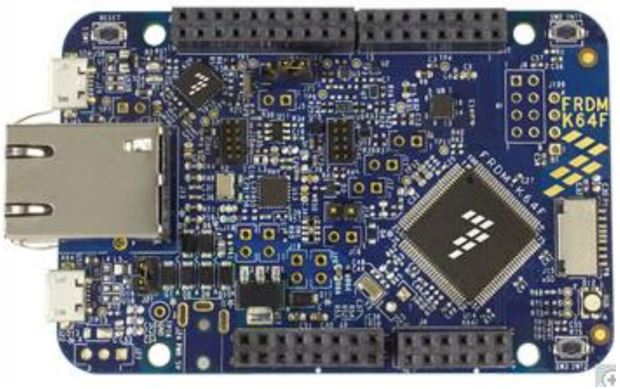
```
C:\Users\Matthew\AppData\Local\Temp\build5957657308977725428.tmp\Fade.cpp.elf,
C:\Users\Matthew\AppData\Local\Temp\build5957657308977725428.tmp\Fade.cpp.bin]
Binary sketch size: 31,728 bytes (of a 262,144 byte maximum)
```

CC3200 Thoughts

- Took longer to get working
- Had to worry about both server and device
- More flexibility: could run server locally for latency-sensitive applications

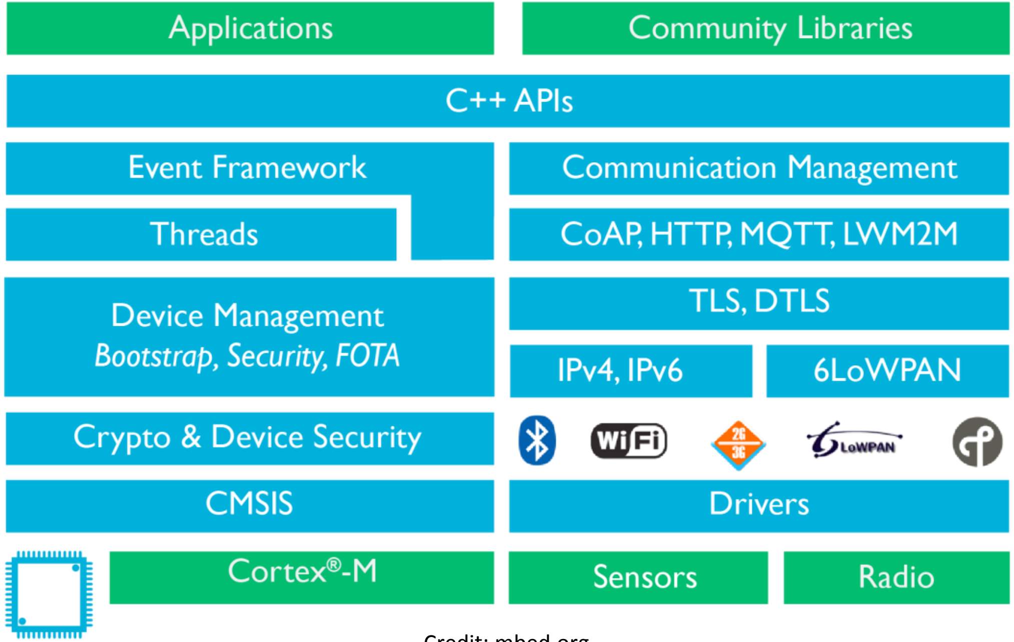
Example 4

FRDM-K64F with mbed



Credit: freescale.com

mbed



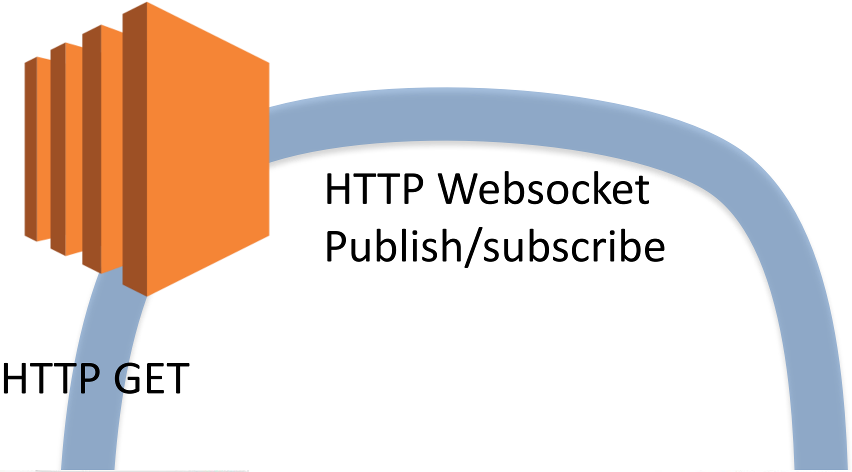
Credit: mbed.org

FRDM-K64F with mbed

- Getting started with mbed
- Used online compiler, but offline toolchain available
- Example WebSocketClient already existed: network connected blinking LED through EC2 server in under 2 hours.

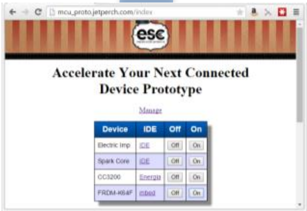
Architecture

mcu_proto.jetperch.com
Amazon EC2 instance



HTTP Websocket
Publish/subscribe

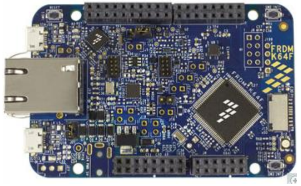
HTTP GET



Accelerate Your Next Connected Device Prototype

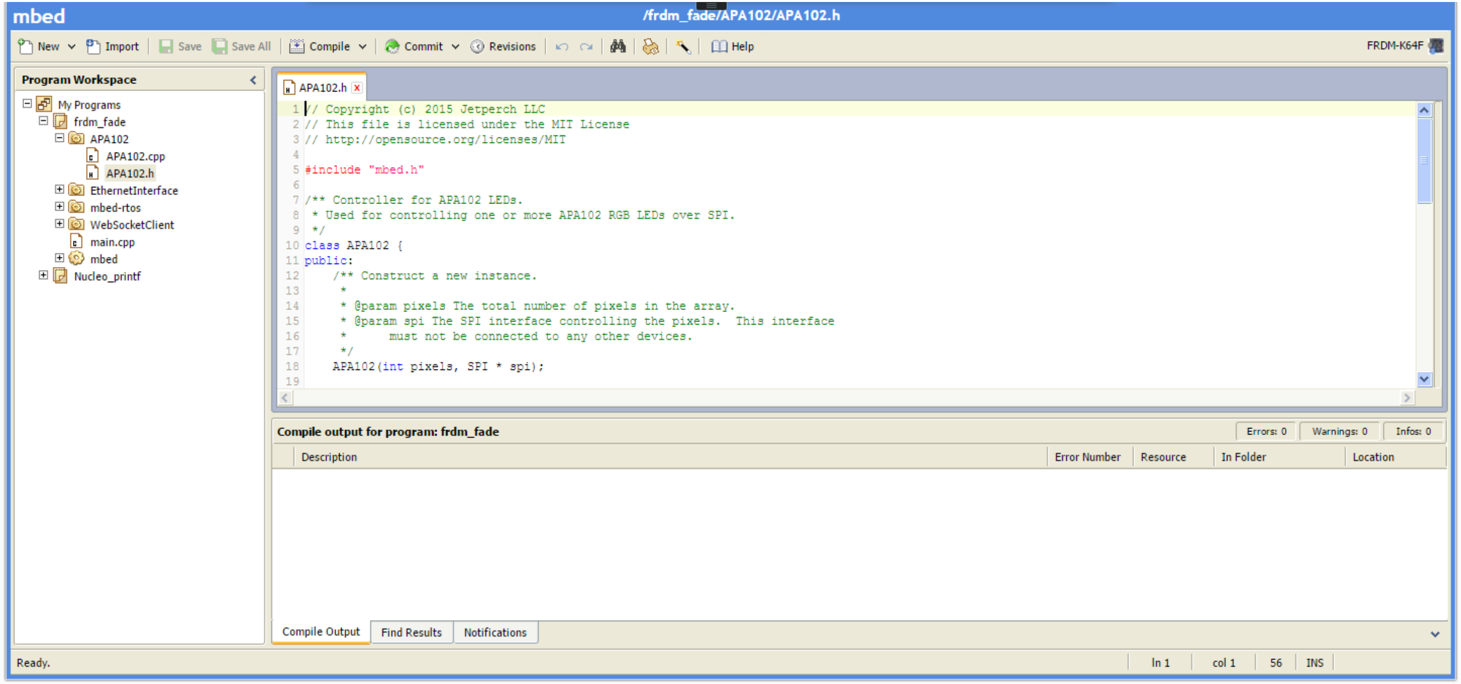
Manage

Device	IDE	Off	On
Electric Imp	IDE	Off	On
Spark Core	IDE	Off	On
CC3200	Energia	Off	On
FRDM-K64F	mbed	Off	On



FRDM-K64F
(device)

FRDM-K64F with mbed demo



```
// Copyright (c) 2015 Jetperch LLC
// This file is licensed under the MIT License
// http://opensource.org/licenses/MIT

#include "mbed.h"

/** Controller for APA102 LEDs.
 * Used for controlling one or more APA102 RGB LEDs over SPI.
 */
class APA102 {
public:
    /** Construct a new instance.
     *
     * @param pixels The total number of pixels in the array.
     * @param spi The SPI interface controlling the pixels.  This interface
     *      must not be connected to any other devices.
     */
    APA102(int pixels, SPI * spi);

```

Compile output for program: frdm_fade

Errors: 0 | Warnings: 0 | Infos: 0

Description	Error Number	Resource	In Folder	Location

Ready. ln 1 col 1 56 INS

FRDM-K64F Thoughts

- mbed is still under very active development and still seems to have rough edges
  - Should see great strides with ARM backing
  - Change browser download directory to the mbed USB mass-storage path for easy online compile, download & reset and
- Broad library with many user-contributed modules
- Had to worry about both server and device
- More flexibility: could run server locally for latency-sensitive applications

Example Summary

Example	Language	Online Compiler	Offline Compiler	Breadth	Ease of use
Electric Imp	Squirrel	x		Excellent	Excellent
Particle Core	Wiring (C++)	x	x	Great	Good
CC3200 & Energia	Wiring (C++)	x	x	Fair	Good
FRDM & mbed	C++	x	x	Good	Good

Conclusions

- Prototyping connected devices can be quick and painless
  - Many solutions (too many?), some end-to-end
  - Writing your own end-to-end service for prototyping is not difficult
- Converting connected prototypes to products is not trivial
  - Security
  - Device management, in-field upgrades, etc.
  - Reliability and ease of use

Accelerate Your Next Connected Device Prototype:

A look at three different prototyping environments

Matt Liberty

Jetperch LLC

matt@jetperch.com

Code at https://github.com/mliberty1/mcu_proto

Grand finale: Go to http://mcu_proto.jetperch.com

References

- http://tech.co/prototype-hardware-startups-2015-02
- https://leanpub.com/iot-javascript

Other platforms

- AirBoard (small Arduino + Bluetooth + WiFi + XBee)
- OpenWRT
- Printoo: Flexible BT Smart Arduino
- LightBlue Bean: Arduino Bluetooth SMART
- Fritzing
- NodeUSB (under development, Lua on ESP8266)
- DigiStump Acorn/Oak

Cloud Services

- thethings.io
- Node-RED
- IFTTT