nabto

connect – simple and secure

PLATFORM FOR

THE INTERNET OF THINGS

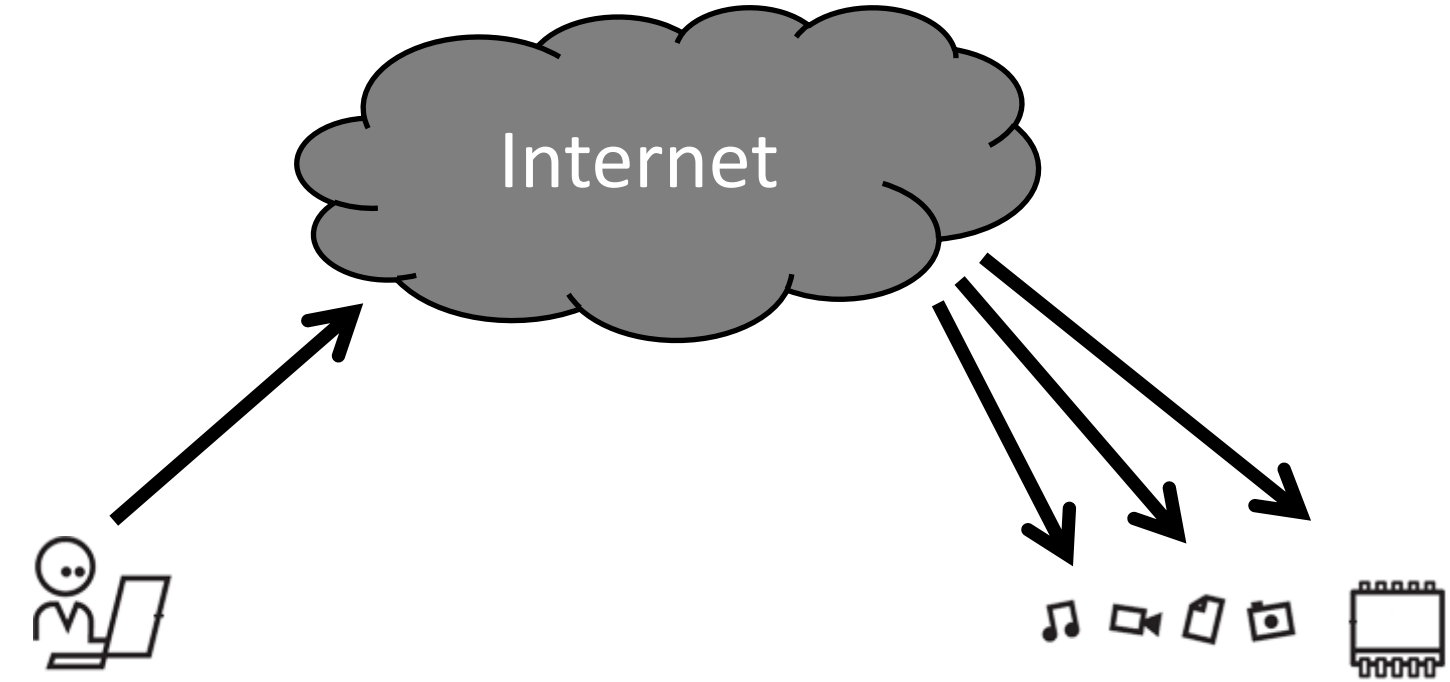
AGENDA

nabto

- What is Nabto
- Architecture
- Devices as web servers
- Programming a micro web server
- Client API
- Live examples

PROBLEM?

nabto



- Firewalls and security
- Server bottlenecks
- Complex development
- HTML design for different purpose

CONNECT – SIMPLE AND SECURE

nabto

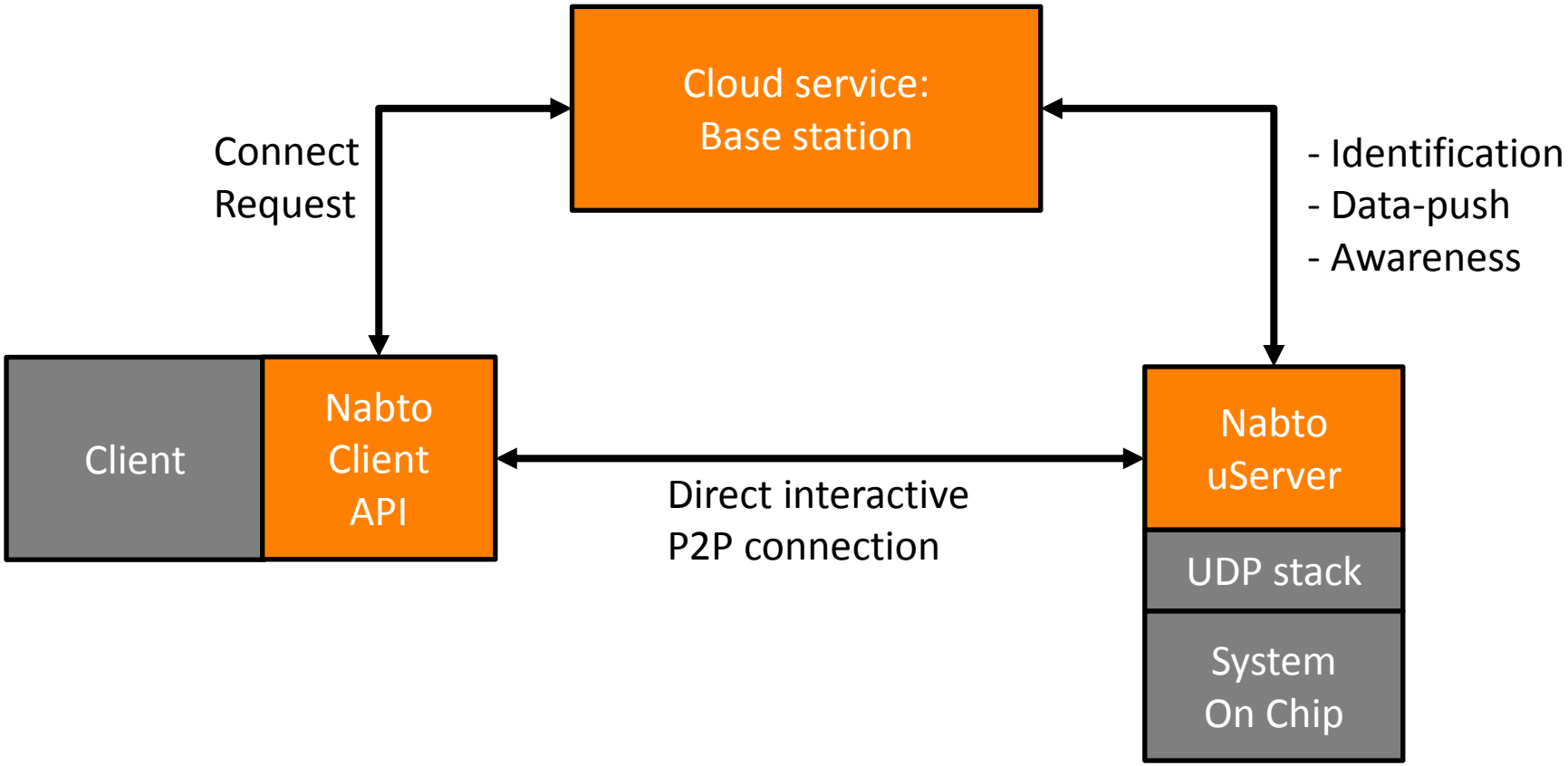
Nabto's vision: Develop a communication platform for the "Internet of things". The platform solves the following issues:

- Connect: **Remote (browser) access**
  - Increase end-user value of your product by offering remote access
- Simple: **Simple platform**
  - Lower development and maintenance cost of your firmware
- Secure: **Maximal security**
  - Nabto is build upon state-of-the-art security

NABTO – THE PRODUCT

nabto

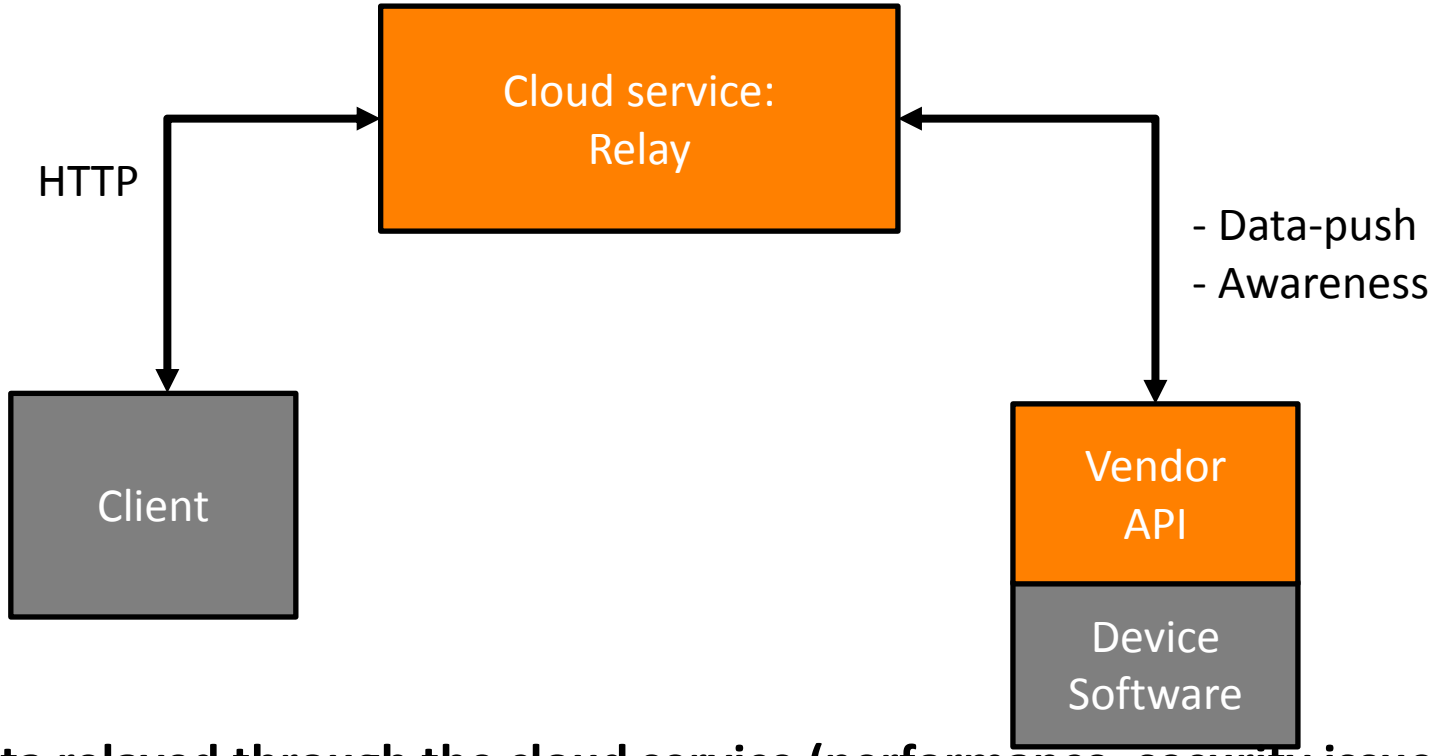
A software platform for the ”Internet Of Things”
With a peer-to-peer interactive client pull focus



EXISTING DESIGN

nabto

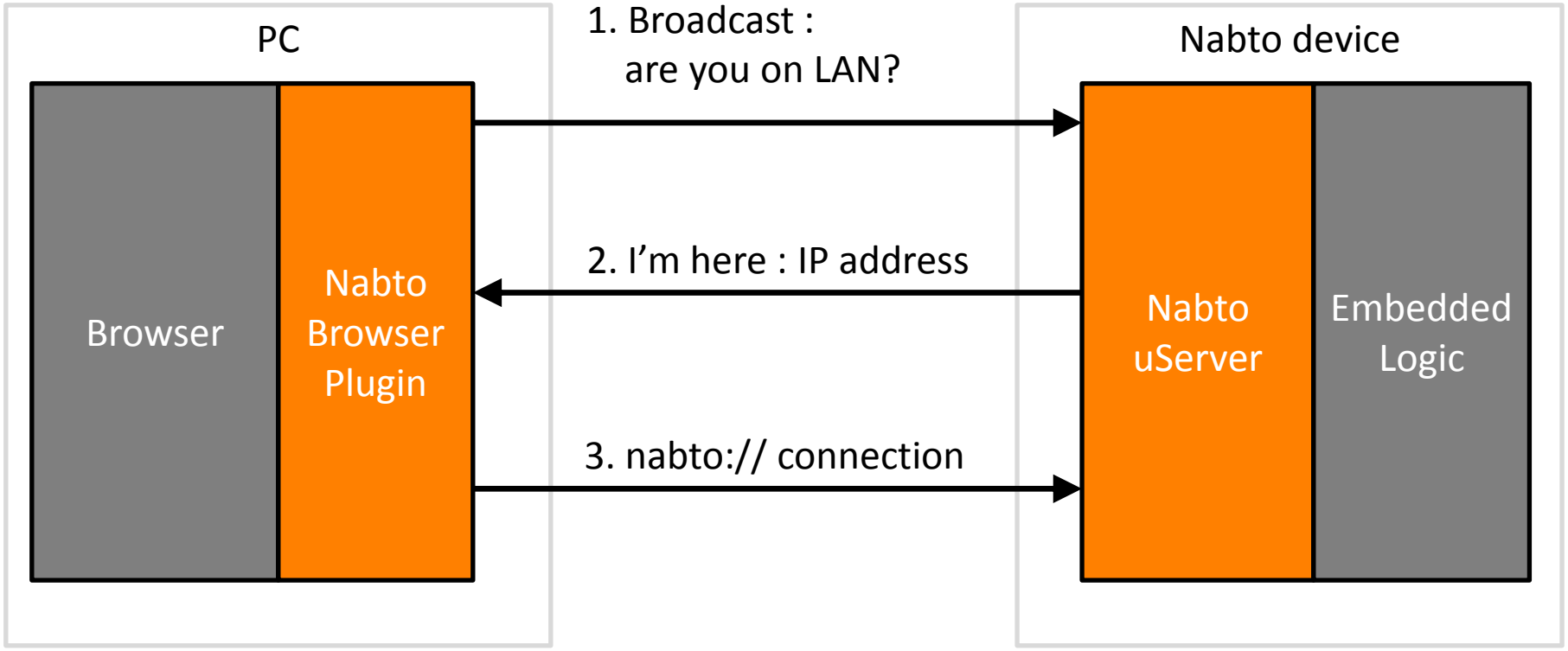
Compared to existing design



Notice:

- **All data relayed through the cloud service (performance, security issues)**
- **No internet = no interface and access**
- **All HMI computing is central**
- **Latency is suboptimal**
- **HMI computing is not (necessarily) encapsulated modular**

NO INTERNET – NO PROBLEM



NB: LAN can be just a net-cable from Laptop to device

WEB SERVERS

nabto

What they got

Standard Internet-platform:

Expensive

Complex

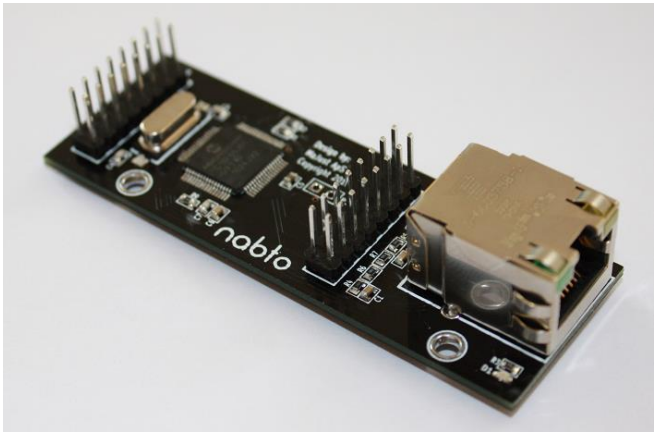
NB: HTTP was created a CERN for another problem type

What they need

Nabto platform:

Inexpensive

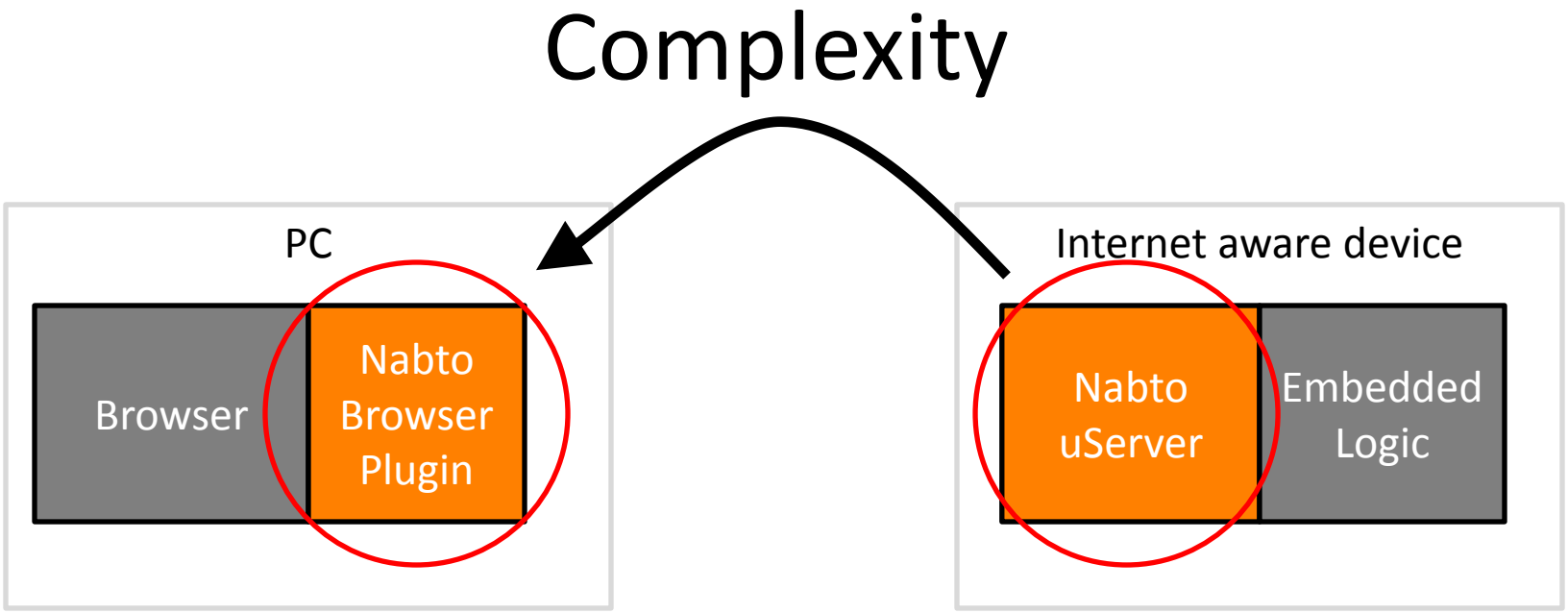
Simple



Ardunio, ColdFire, FreeRTOS, GainSpan, Microchip, RTX4100, ...

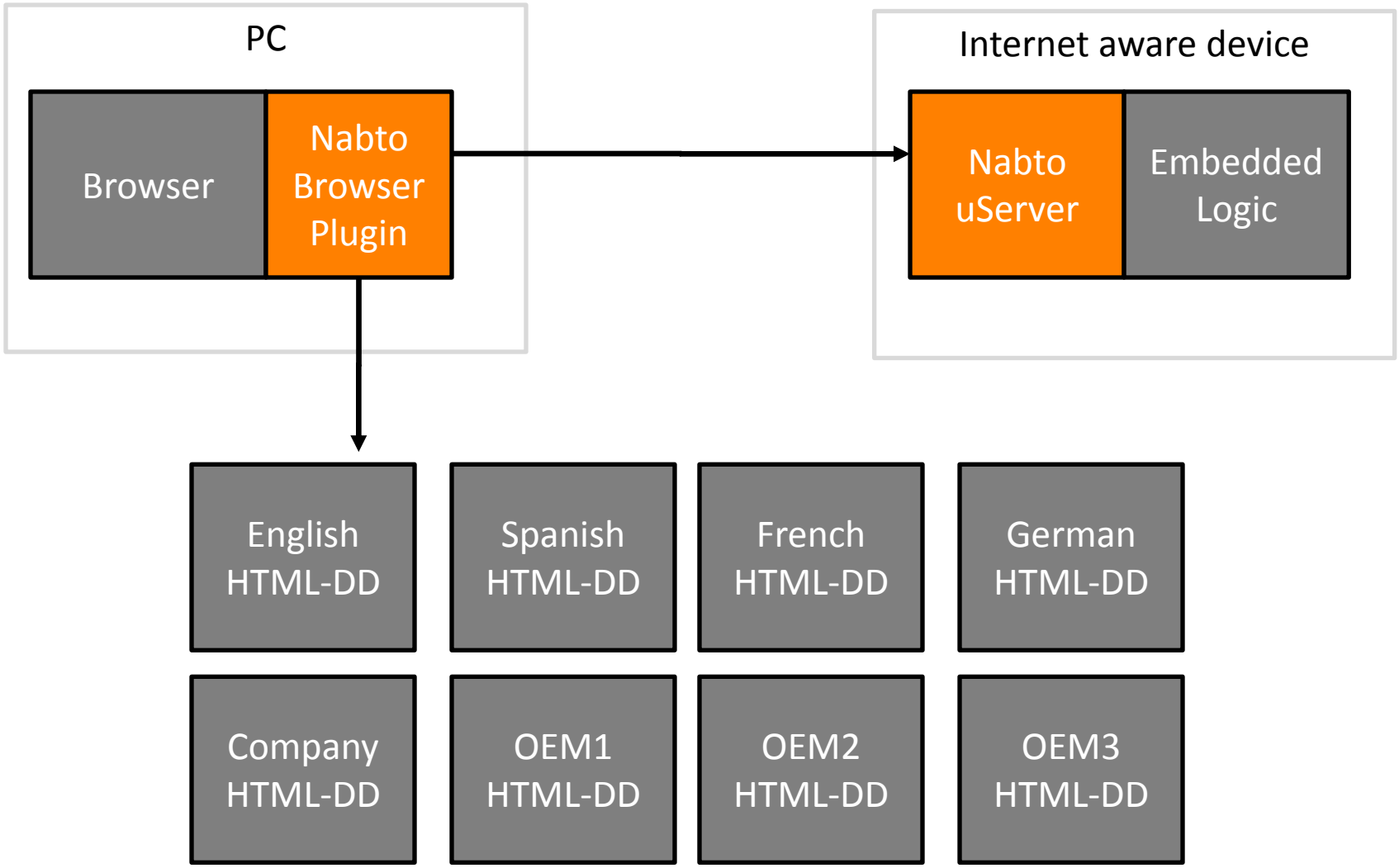
MOVE THE COMPLEXITY TO THE CLIENT

nabto



Complexity is moved from **Device** to **Client platform** by installing a protocol plugin on the client

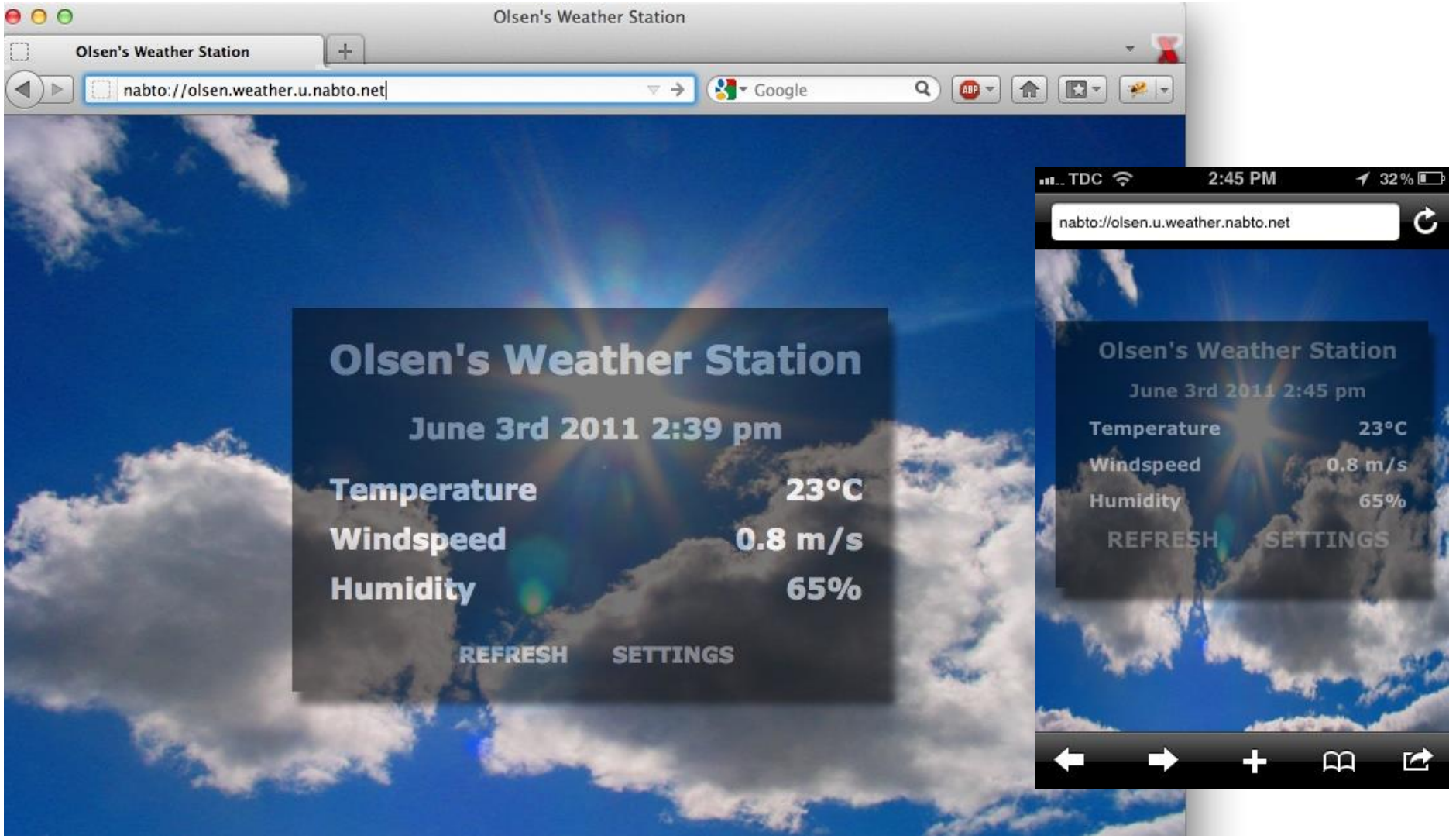
HTML Device Driver



MICRO WEB SERVER EXAMPLE

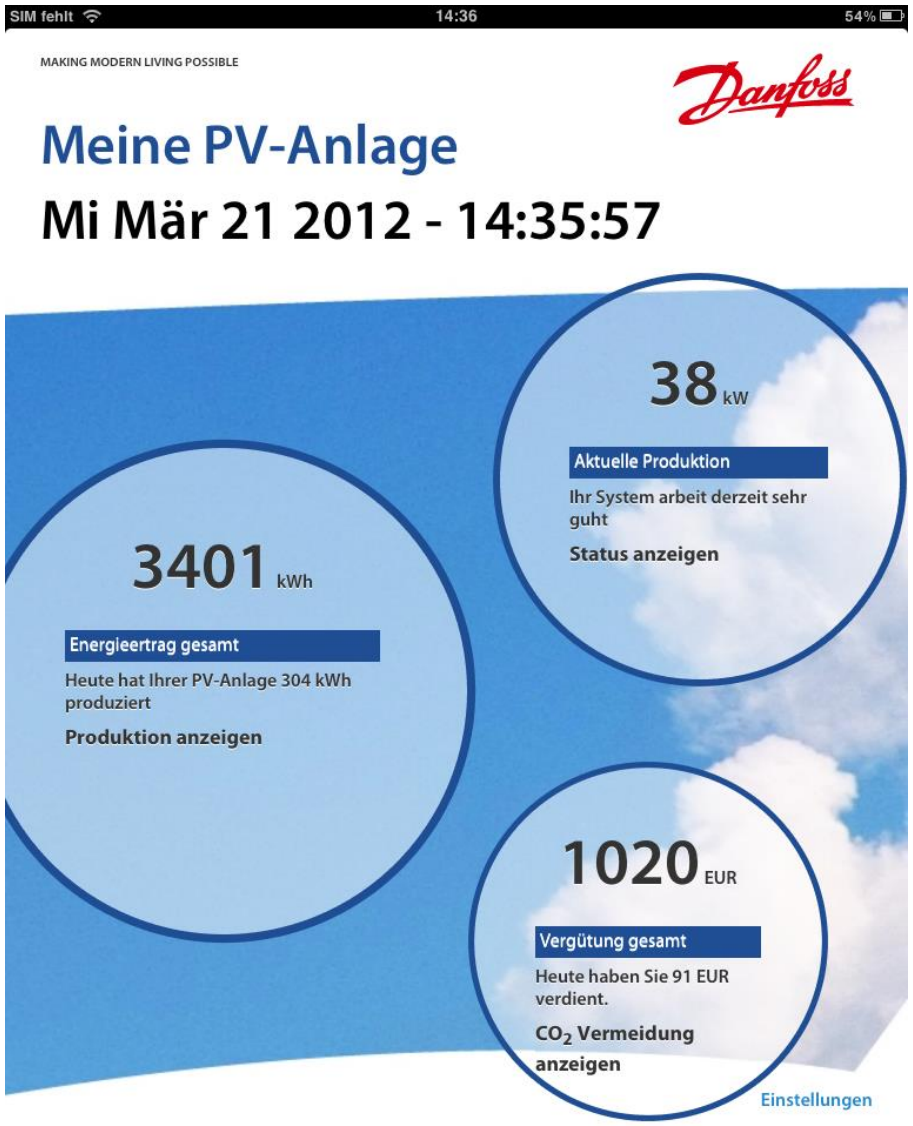
nabto

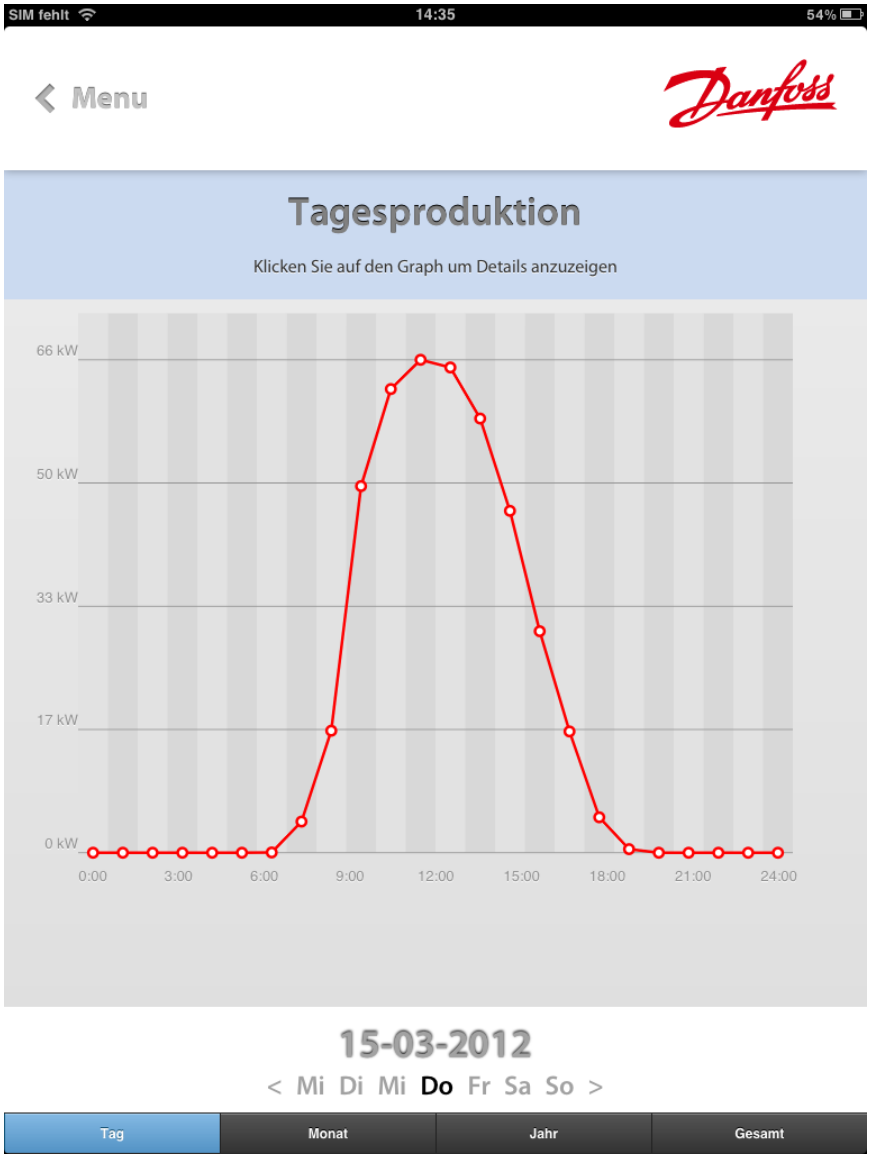
- Web server running on 8 bit Atmel AVR CPU with 2 kB of RAM, 32 kB flash



Real Life Example: Danfoss

nabto



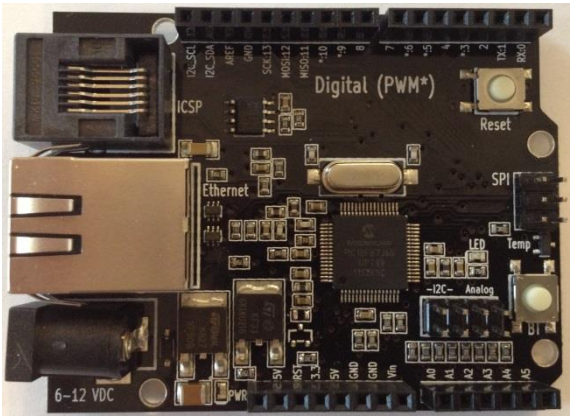


Freescale MCF52255 – 64kb RAM 512kb flash

NABDUINO

nabto

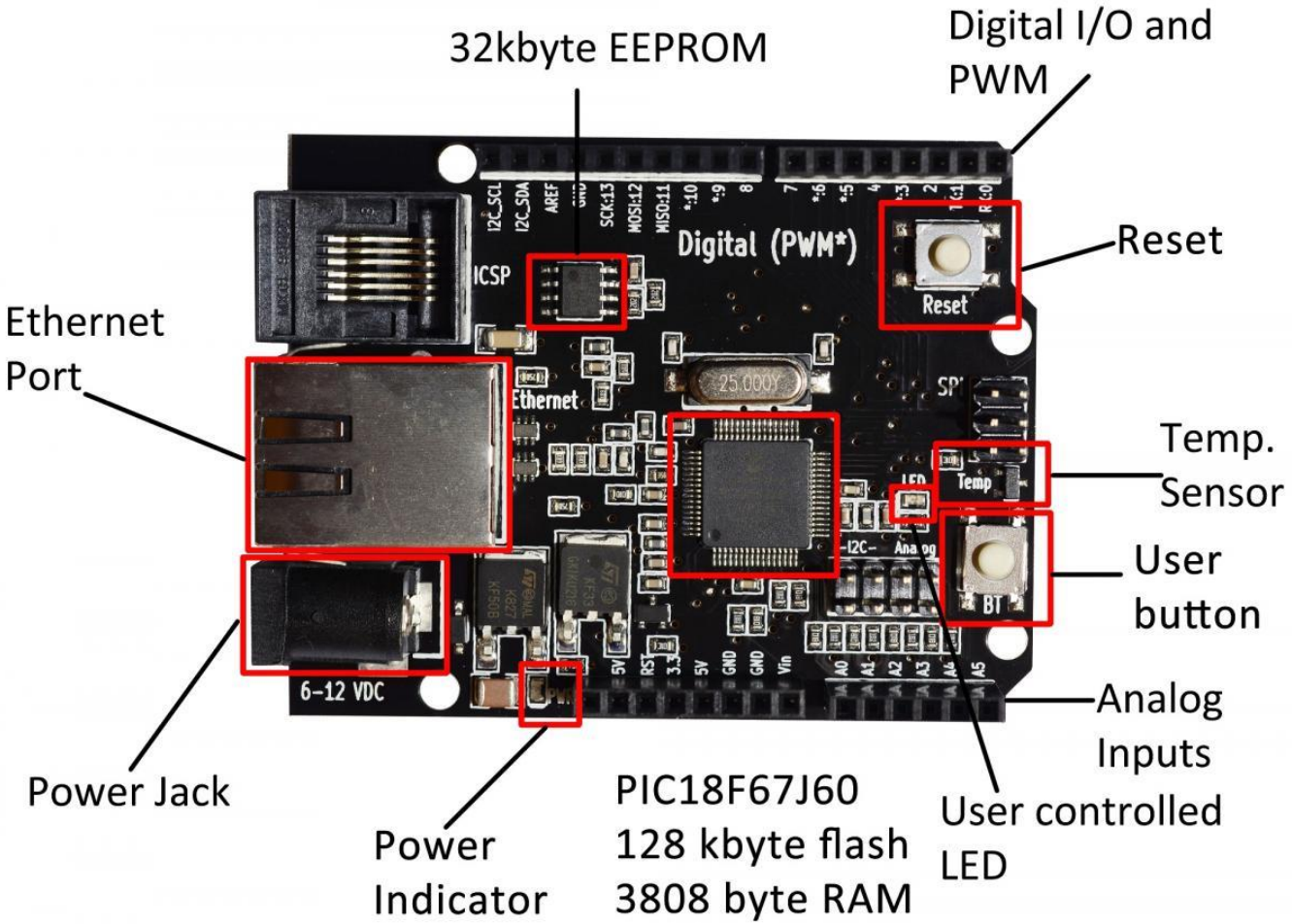
http://nabduino.com



NABDUINO BOARD

nabto

DEMO - http://demo2.nabduino.net



nabto

nabto

connect - simple and safe

15 minutes break

DEVELOPMENT TOOLS

nabto

http://nabduino.com/download

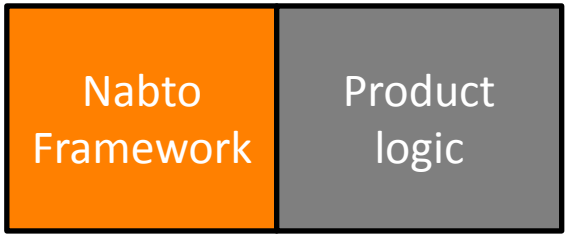
- Download and install MPLAP X IDE
- Download and install MPLAB C18 compiler
  - Don’t download the XC compilers (when asked to)
  - Only build bootloader_release with free compiler
- Goto Microchip Application Libraries
  - Download and install Microchip Libraries for Applications
- Download Nabto Updater
  - Copy NabtoUpdater.exe to location for easy drop’ing
- Download and unzip the Starterkit
  - cd unabto_starterkit\unabto\external
  - mklink /J Microchip "C:\microchip_solutions_v2013-02-15\Microchip"

DELIVERY

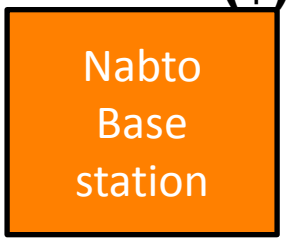
nabto

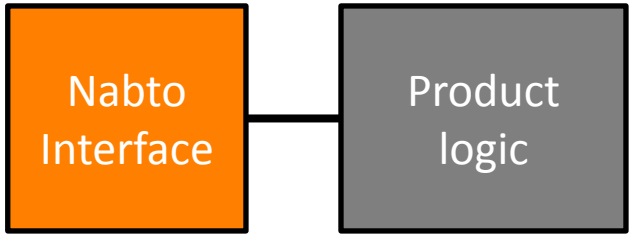
Closed source - Client API

Source available - Starter Kit ($)



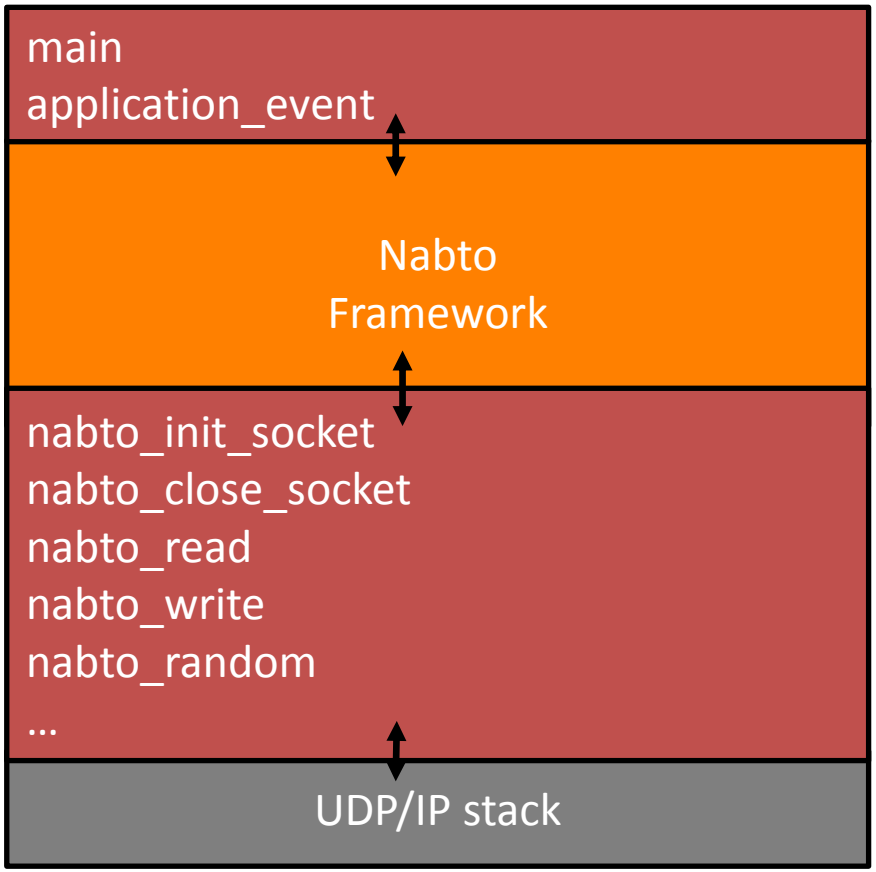
Closed source ($)





Chip and Hardware modules ($)

1. Implement Functions



Application Interface

Device platform

Platform Interface

#define NABTO_DEVICE NABTO_DEVICE_PIC18

- **Implement 14 platform functions (or use an existing implementation)**
- **Implement the application event handler (see later)**
- **Implement the main loop (loop or event driven)**

2. Implement Main Loop

nabto

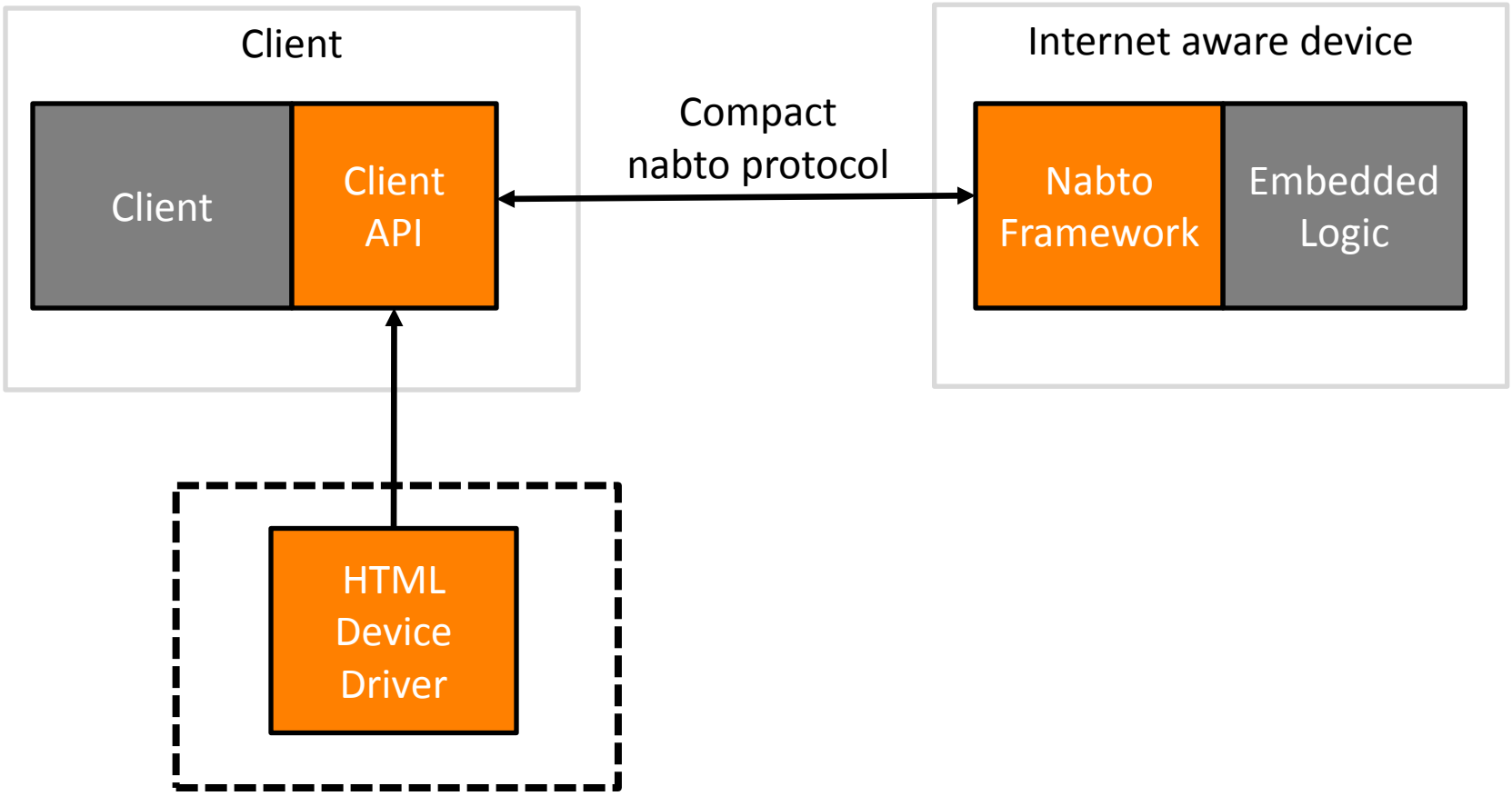
```
int main() {
    nabto_main_context_t nmc;
    nabto_init_default_values(&(nmc.nms));
    nmc.nms.id = "foo.u.nabto.net";
    nabto_main_init(&nmc);
    while (true) {
        nabto_main_tick(&nmc);
        sleep_ms(10);
    }
    nabto_main_close(&nmc);
    return 0;
}
```

3. Implement Event Handler

nabto

```
application_event_result_t application_event(
    application_request_t* req, buffer_read_t* ibuf, buffer_write_t* obuf) {
    switch (req->query_id) {
        case 0x0a: {
            uint16_t sensor_id;
            uint8_t filter;
            uint16_t temperature;
            buffer_read_uint16(ibuf, &sensor_id);
            buffer_read_uint8(ibuf, &filter);
            temperature = readTemperature(sensor_id, filter);
            buffer_write_uint16(obuf, temperature);
            return AER_REQ_RESPONSE_READY;
        }
    }
    return AER_REQ_INV_QUERY_ID; }
```

4. IMPLEMENT HTML-DD



"HTML-Device driver" encapsulates GUI and defines specific data transport interface of the device

HTML Device Driver

nabto

Just a simple .zip file

- /static
  - Static content = jpg, png, css, javascript, etc.
- /nabto
  - HTML templates (old style)
- /nabto/unabto_queries.xml
  - Mapping : Request URL -> Binary format
  - Mapping : Response -> JSON response

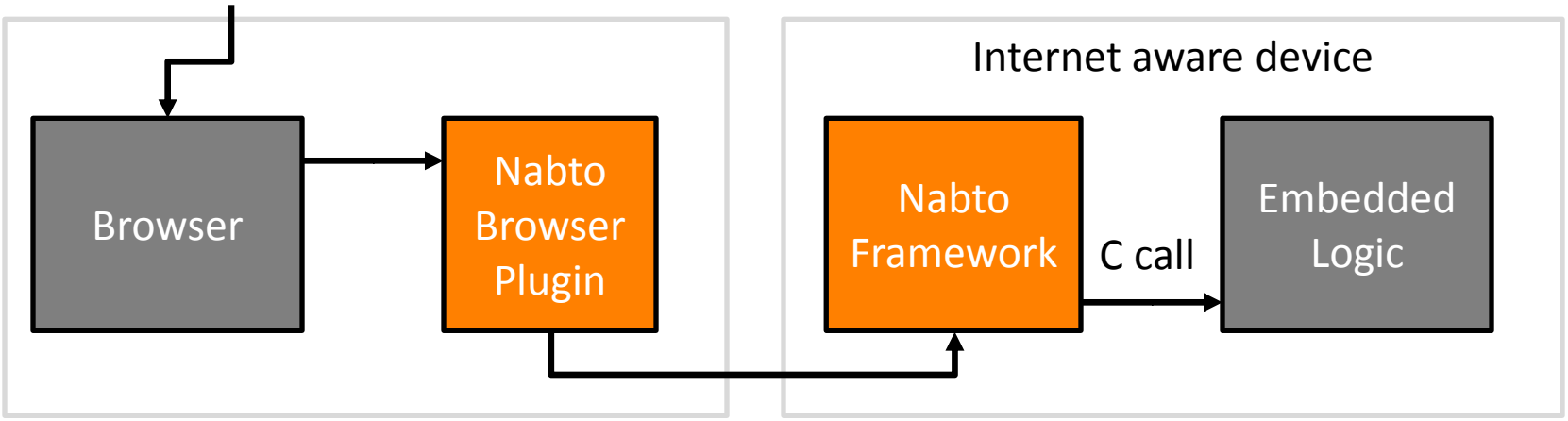
EXAMPLE : GETTEMPERATURE

nabto

```
<query name="getTemperature" id="0x0a">
    <request>
        <parameter name="sensorId" type="uint16"/>
        <parameter name="filter" type="uint8" default="0"/>
    </request>
    <response format=“json">
        <parameter name="temperature" type=“uint16"/>
    </response>
</query>
```

EXAMPLE : GETTEMPERATURE REQUEST

User input - via a nice menu
nabto://5924.nabduino.net/getTemperature?sensorId=3

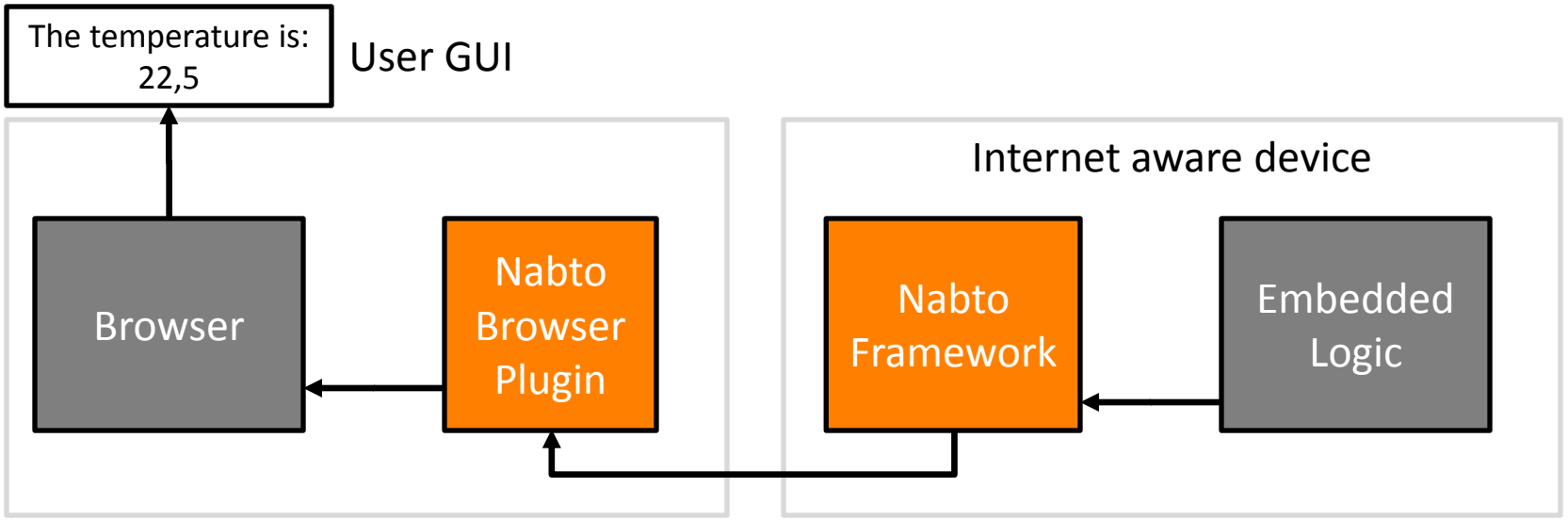


application_event(0x0a, &Buf[1], 3)

Request buffer: | 0x0a \| 0x00 0x03 \| 0x00 |

Buf[0] : GetTemperature request identifier
Buf[1,2] : Sensor identification
Buf[3] : Filter identification
(see former slide for XML definition)

EXAMPLE : GETTEMPERATURE RESPONSE



Respone in JSON format
Handled by JavaScript:

Response buffer: 0x16 0x80

application_event() function
returns response buffer

```
function queryDevice(request) {
   jQuery.getJSON(request, null, function(response) {
      var response = response["response"];
      $(”button”).val(response[”temperature"])
   });
}
```

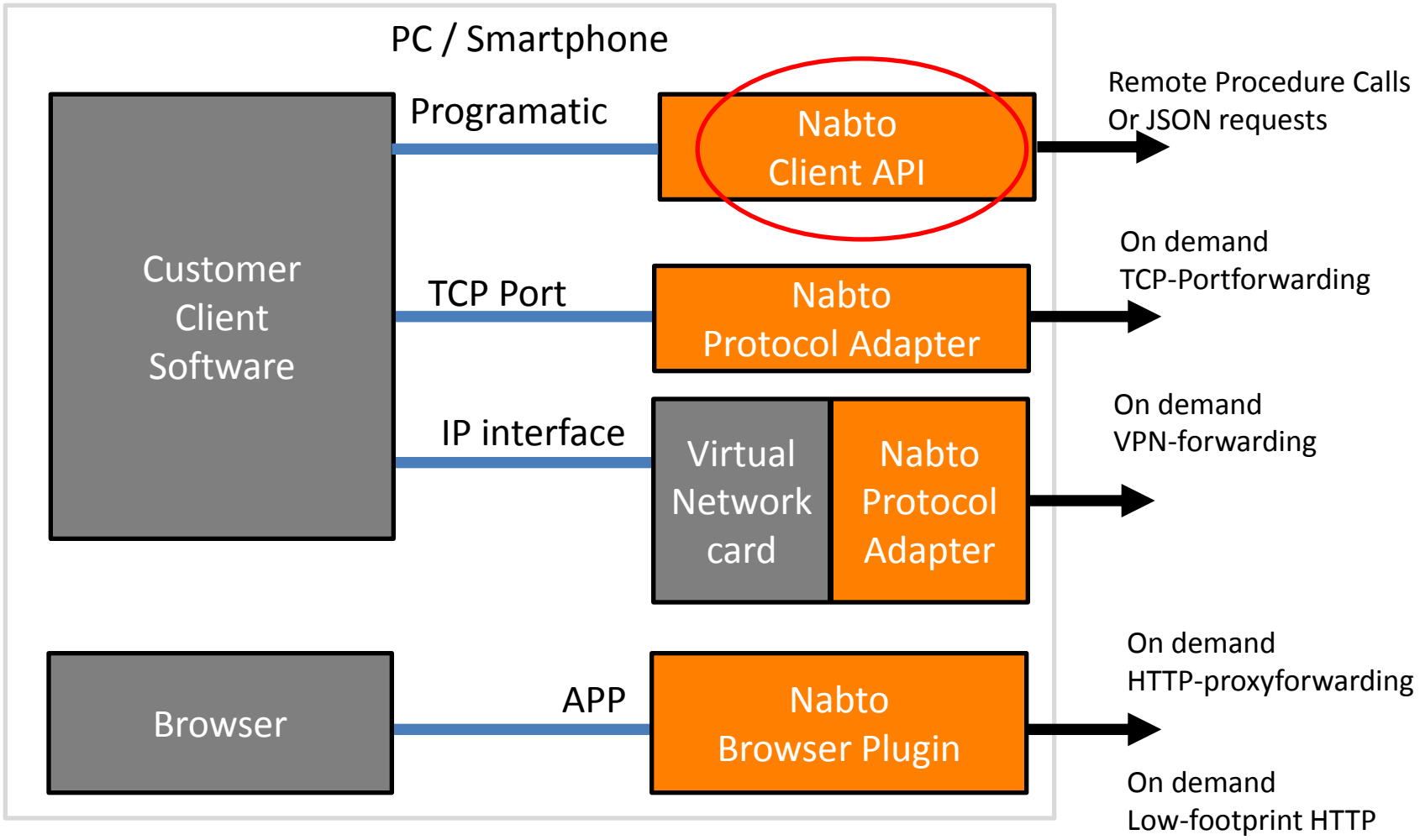
Building and Burning

nabto

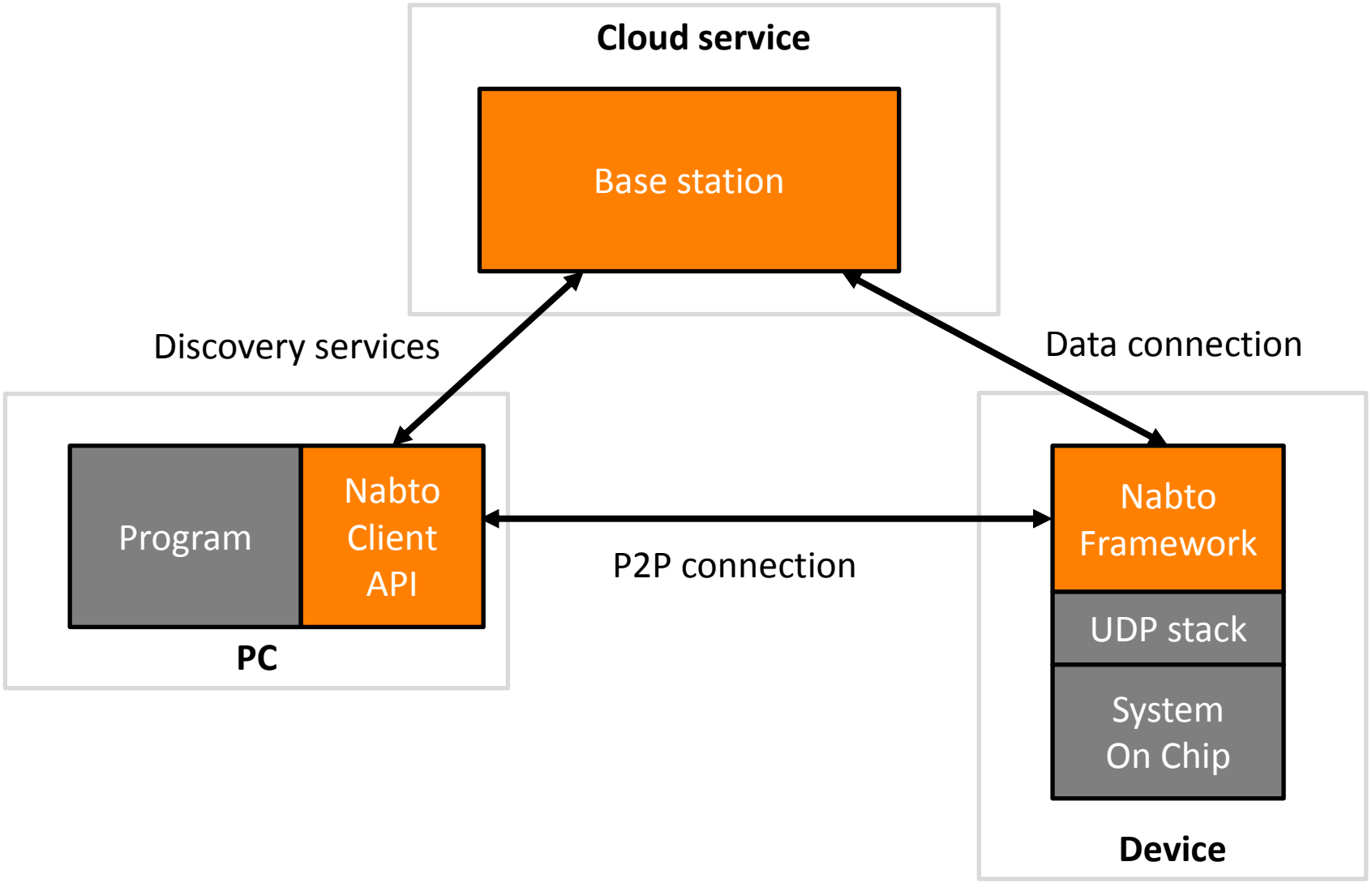
- Start MPLAB.X IDE and open project
  - unabto_starterkit/unabto/demo/nabduino/MPLAB.X
- Set Nabduino as main project (in Projects window)
- Edit src/application.c (application_event)
- Build main project
- Open an Explorer
  - Go to dist/bootloader_release/production in MPLAB.X
- Drag MPLAB.X.production.hex to NabtoUpdater.exe
- Reset the nabduino board

- Documentation:
  - unabto_starterkit/doc/starter_kit/unabto_starterkit.html

CLIENT API


PC / Smartphone
Customer Client Software
Programatic
Nabto Client API
Remote Procedure Calls Or JSON requests
TCP Port
Nabto Protocol Adapter
On demand TCP-Portforwarding
IP interface
Virtual Network card
Nabto Protocol Adapter
On demand VPN-forwarding
Browser
APP
Nabto Browser Plugin
On demand HTTP-proxyforwarding
On demand Low-footprint HTTP

NABTO – CLIENTAPI - PROGRAM CONNECTION



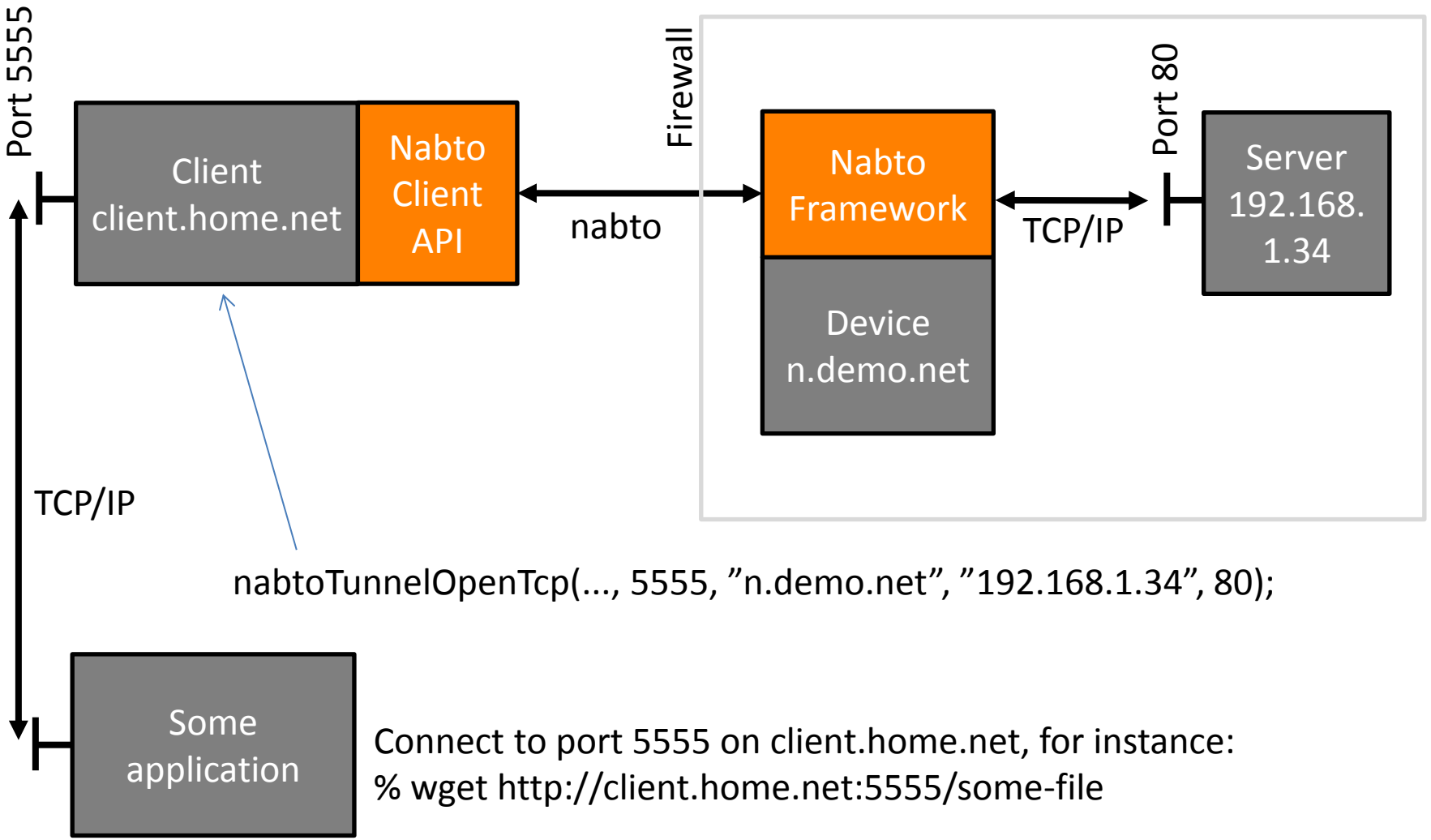
CLIENT API

nabto

- Configuration and initialization API
    - nabtoStartup
    - nabtoShutdown
- Session API
    - nabtoOpenSession
    - nabtoFetchUrl
    - nabtoAsyncFetch
    - nabtoCloseSession
- Streaming API
    - nabtoStreamOpen
    - nabtoStreamRead
    - nabtoStreamWrite
    - nabtoStreamClose
- Tunnel API (next release)
    - nabtoTunnelOpenTcp
    - nabtoTunnelInfo
    - nabtoTunnleClose

TUNNELING

nabto



Connect to port 5555 on client.home.net, for instance:
% wget http://client.home.net:5555/some-file

nabto

connect – simple and secure

THANK YOU FOR LISTENING