A NEW PROGRAMMING LANGUAGE FOR THE EMBEDDED IOT

Christian Weichel
Bosch Connected Devices and Solutions GmbH
Eclipse PAX
Scaling over quantity

![Graph showing cumulative effort over pieces for Breadboard, Custom PCBA, DFM, Industrialization & Certification, XDK, and Others.]
Eclipse PAX
Scaling over quantity

Cumulated Effort

Breadboard
Custom PCBA
DFM
Industrialization & Certification
Others

XDK

Pieces

1
10
100
1000
10000
Eclipse PAX
Scaling through integration
### Eclipse PAX

We need a way to program that

<table>
<thead>
<tr>
<th>Lowers barriers</th>
<th>Scales to production</th>
<th>Scales to low-power HW</th>
</tr>
</thead>
<tbody>
<tr>
<td>➤ Familiar and easy to learn</td>
<td>➤ Afford a lot of control → go really low level</td>
<td>➤ Very low runtime impact</td>
</tr>
<tr>
<td>➤ “Feels” good → user centred design</td>
<td>➤ Based on proven infrastructure</td>
<td>➤ No garbage collection</td>
</tr>
<tr>
<td>➤ Convenience matters → high abstraction</td>
<td>(LLVM on bare metal, e.g. Rust + Zinc or Taylor Swift are still years out)</td>
<td>➤ Preference for static memory allocation</td>
</tr>
<tr>
<td>➤ Doesn’t annoy power users → high ceiling</td>
<td>➤ Enable trust in software → allow inspection in a world people know</td>
<td></td>
</tr>
<tr>
<td>➤ Enable IoT development for new user groups</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Bosch Connected Devices and Solutions GmbH 2017. All rights reserved, also regarding any disposal, exploitation, reproduction, editing, distribution, as well as in the event of application for industrial property rights.
Eclipse PAX
What about existing languages

- LUA
- Python
- mRuby
- Javascript

Widespread adoption but expensive at runtime and far removed the system
Eclipse PAX
What about existing languages and platforms

<table>
<thead>
<tr>
<th>Rust</th>
<th>C++</th>
<th>Arduino</th>
<th>mbed OS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Far away from being production ready on bare metal</td>
<td>High “accidental complexity”</td>
<td>Based on C++ / removed from other industry-grade platforms</td>
<td></td>
</tr>
</tbody>
</table>
Eclipse PAX
Inspired by research


Powerful language (C#) and brilliant IDE support make IoT prototyping more accessible.

People understand what they can see.

- Direct interaction with sensor data and devices.

Programming as Domain Specific Language (DSL).
Eclipse PAX
Transpiles to C
## Eclipse PAX

### Base Language
- Imperative with future aspirations for functional elements
- Event driven
- Exceptions try/catch instead of return
- Extension methods provide OO feeling
- String interpolation built into the language

### Type System
- Statically typed with generics and optional types
- Type inference
- Generic Types
- Static Heapless Memory Management through Data Structure Size Inference

### Model-Driven
- Declarative Setup of platform-defined system resources
- Direct Access to System Resources such as sensors, connectivity and GPIO
- Generic Extensible Platform Support not specific to XDK
- Built-in Library Mgmt

### Transpiler
- Transpiles to C code
- Traceability between PAX and C code (thanks to Xtext > 2.12)
- Variable names, function names and comments are carried over
Eclipse PAX

Imperative and Statically Typed

- Language design inspired to TypeScript, Swift, Rust, Scala
- Supports all classic control structures
- Immutable and mutable variables
- Static typing → all expressions have a type at compile time
  - Generics are supported for types and functions

```typescript
function <T : integer> sumEven(list : iterable<T>) : T {
    let immutableValue = 2;
    var result = 0;

    for(var x in list) {
        if(x % immutableValue == 0) {
            result += x;
        }
    }

    return result;
}
```
Extension Methods

- First parameter can be written on left side during function invocation
- Provides an “object oriented feeling”
  - It helps that functions are polymorphic
- Standard library is implemented this way

```javascript
function mean(self : iterable<float>) : float {
    var result = 0.0;
    for(var x in self) {
        result += x;
    }
    return self.sum() / self.length();
}

let values = [0.0, 1.0, 2.0, 3.0];
var meanValueA = mean(values);

var meanValueB = values.mean();
```
Eclipse PAX
Declarative Setup

- We need to configure the system we run on
  - Sensors, Connectivity and GPIO

- All of those define “configuration items”
  - Accelerometer Range
  - Bluetooth Advertising Interval
  - SPI clock speed

- All of those can offer events
  - Accelerometer Activity
  - Bluetooth Connection Incoming
  - SPI Slave Message Received
Eclipse PAX
Declarative Setup of Sensors

All setup starts with the `setup` keyword

```
setup accelerometer {
    range = Range_8g;
    activity_threshold = 200;
}
```

Followed by the resource we want to configure

Sensors offer platform-defined `configuration` items
Eclipse PAX
Declarative Setup of Connectivity

Connectivity (and GPIO) can be named

```
setup devNetwork : WLAN {
    ssid = "BCDS_DevNet";
    psk = "MySuperSecretPassword";
}
```

```
setup backend : LWM2M {
    transport = devNetwork;
    server = "10.0.0.1";

    var shockDetected =
        property(url="/1/2", init=false);
}
```

*Signals* configure things like LWM2M properties, BLE characteristics, REST APIs or GPIO pins
Eclipse PAX
Declarative Setup of GPIO

Every system resource has their own signals

```plaintext
setup gpio {
    var externalLed = digitalOut(pin=A4, driveStrength=High);
    var battery = analogIn(pin=B2);
}

setup mySensor : I2C {
    sda = Pins.A1;
    scl = Pins.A2;

    var creg1 = register(address=0x0A, init=0x00 as uint16_t);
    var value = register(address=0xAB, init=0x00 as uint32_t);
}
```

I2C registers mapping, including data type
Direct Access

Access sensor values as if they were variables

every 100 milliseconds {
    if(accelerometer.magnitude > 5000) {
        // TODO: do something
    }
}

Eclipse PAX
Event Driven

every system.startup {
  // system just started
}

every 100 milliseconds {
  // TODO: do something every 100ms
}

every accelerometer.activity {
  // our device was just moved
}
Eclipse PAX
Event Driven

- Time
  - 5 seconds
  - 100 milliseconds

- Sensors
  - accelerometer.activity
  - light.bright

- Connectivity
  - backend.connected
  - smartphone.cfgchar.written

- GPIO
  - gpio.myPin.fallingEdge
Eclipse PAX
Where do we go from here?

Available today at http://xdk.io
Eclipse PAX

Where do we go from here?
Eclipse PAX

Is a new programming language for the embedded IoT
Enables high-level features transpiled to C
Is not limited to the XDK
Want to play with this stuff?

http://xdk.io