decode

D4.3 Provisional Hardware Platform

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no. 732546

Project no. 732546

DECODE

DEcentralised Citizens Owned Data Ecosystem

D4.3 Provisional Hardware Platform

Version Number: V1.0

Lead beneficiary: Arduino

Due Date: August 31

Author(s): Arduino/BCMI-Labs: Ernesto E. Lopez, Kristoffer Engdahl

Editors and reviewers: Tom Demeyer (Waag), George Danezis, Bano Shenar and Alberto Sonnino (UCL)

Dissemination level:		
PU	Public	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Approved by: Francesca Bria (Chief Technology and Digital Innovation Officer, Barcelona City Hall)

Date: 28/08/2017

This report is currently awaiting approval from the EC and cannot be not considered to be a final version.

Contents

1. Abbreviations

AIO – Asynchronous I/O

BCN – Barcelona

BLE – Bluetooth Low Energy

GbE – Gigabit Ethernet

ICT – Information and Communication Technology

I/O- Input/Output

IoT – Internet of Things

IP – Internet Protocol (Address)

JVM – Java Virtual Machine

MVP – Minimum Viable Product

NDA – Non Disclosure Agreement

OS - Operative System

OSHWA – Open Source Hardware Association

PC – Personal Computer

RFID – Radio Frequency Identification

SBC – Single Board Computer

SDK – Software development kit

SoC- System on Chip

VNC- Virtual Network Computing

WP – Work Package

2. Executive summary

The DECODE HUB is the hardware component of the DECODE architecture. The HUB is any device on which the DECODE OS is installed. It provides connectivity to IoT devices, connects to other DECODE nodes and supports the DECODE OS. As any other component of the DECODE architecture, the HUB needs to follow the values of openness, security, scalability, deployability and flexibility. The goal of this document is to propose a commercial single board computer that can be used as a provisional platform to test DECODE and that is consistent with the aforementioned values.

The DECODE project follows a lean methodology and therefore this document does not contain final specifications but rather guidelines that will feed into the pilot's implementation and the initial round of testing and experiments. Alterations to these specifications might take place following the first round of testing during an ongoing iterative and lean process which will not be complete until the final deployment of the DECODE platform and pilots.

In the inception of DECODE, it was suggested to implement the DECODE HUB as a box based on a custom hardware. A specifically designed hardware platform might be a better option in terms of security, transparency and flexibility and can be adapted to match all the requirements and specifications of DECODE. However, it presents a problem when it comes to the deployment and adoption of DECODE. For that reason DECODE will not run exclusively on a custom embedded device. The DECODE OS developed by Dyne is designed to run on a variety of computing architectures, including several embedded ARM platforms, desktop PC, laptops and servers, and therefore it becomes the most easy to adopt solution. However a provisional platform will be presented as an alternative HUB for the pilots that require extra control of the execution environment or the ability to connect to IoT sensors in a controlled way as in the case of the Making Sense pilot. The physical design of a custom made DECODE HUB is yet to be determined at the time of this deliverable.

After reviewing several single board computers (SBCs) it was agreed with Dyne that a suitable provisional hardware platform would be the A20-OLinuXIno-LIME2[1], an open source SBC designed by Olimex[2], an open source hardware company. The A20-OLinuXIno-LIME2 is a working target of the DECODE OS and we have run some tests to use as a benchmark for future comparisons.

[1] https://www.olimex.com/Products/OLinuXino/A20/A20-OLinuXIno-LIME2/open-source-hardware

[2]https://www.olimex.com/

3. Introduction

The DECODE architecture, in order to decentralize the data management, communicate with IoT devices and provide ownership of the data, needs the physical presence of a HUB on the local area network. The HUB has a set of specifications that are required in order to run the DECODE OS and the other elements of the DECODE NODE and it takes into consideration the IOT-A reference model (defined by IOT-A project, FP7 257521). Apart from the technical requirements, the HUB must be consistent with core values of the project as well.

The outcome of this deliverable will be the presentation of a base DECODE HUB physical appliance able to run a base installation of the DECODE OS and the DECODE NODE interface application and smart rule language engine. This document will provide the reader with the specifications required for the selection of an embedded hardware platform suitable for DECODE. It will also show the technical features of the selected single board computer that will be used in a provisional manner to run some of the tests of the pilots, and that will act as a proof of concept of the DECODE HUB. It will also serve as a starting point to develop a security-oriented DECODE-customized hardware in the future. The provisional platform will not aim to cover all the hardware security features that might be desired for protecting sensitive data but will comply with the open hardware specifications for “maker” technology.

The following sections will describe the HUB technical requirements and will continue with a case study of the different available single board computers in the market. After the selection of the provisional platform, a selection of tests results will be presented in order to proof the basic performance of the embedded platform.

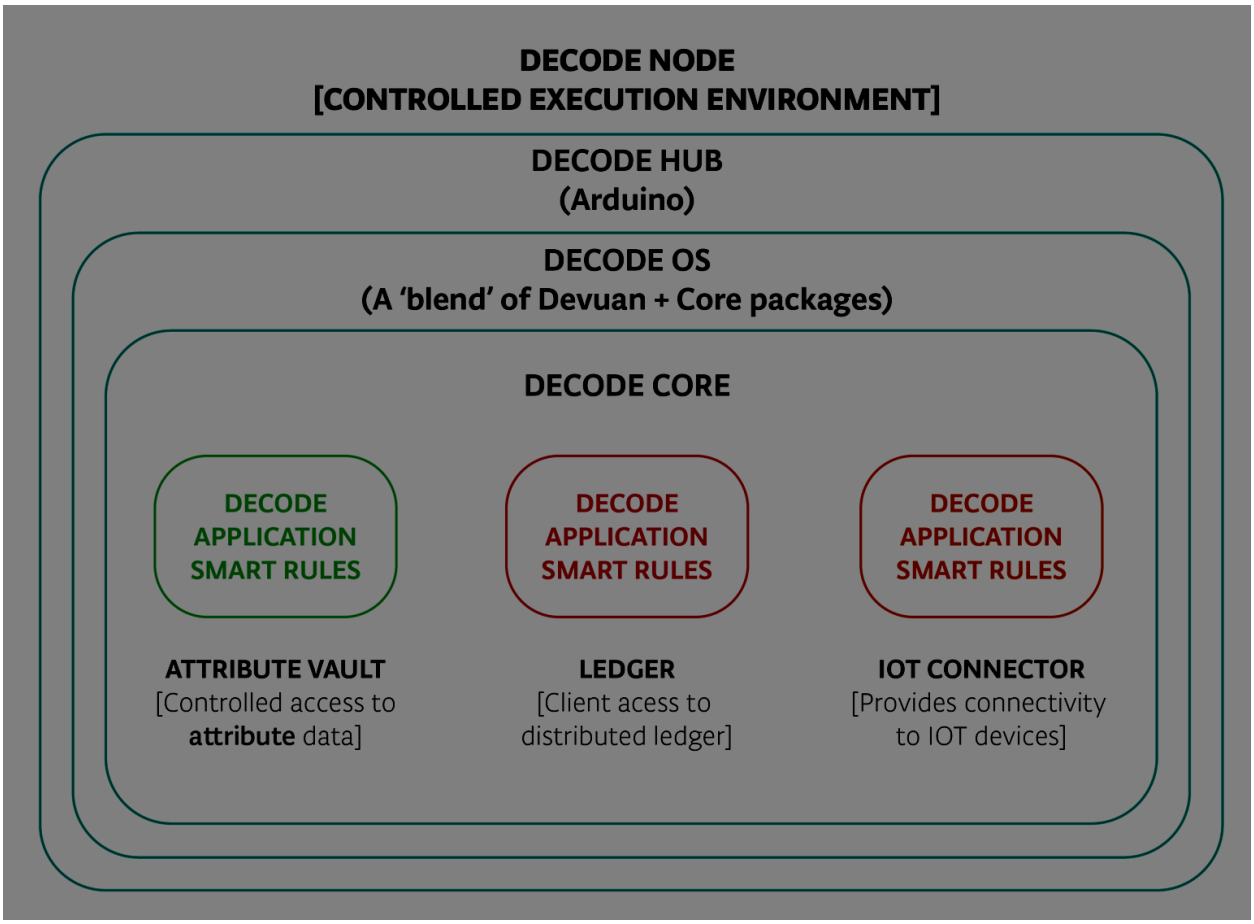


Illustration 1: DECODE NODE architecture

The current state of the architecture suggests the need of different types of NODEs. A "full" NODE will contain all the modules defined in the NODE architecture, and might function as a dedicated server.

The full definition of the hardware architecture and the different types of NODEs is yet to be defined at the time of this deliverable. For this reason, the specifications for the HUB presented in this document may change in the future. For a detailed description of the complete NODE architecture please refer to the forthcoming DECODE white paper.

4. DECODE HUB requirements

The following section will describe the hardware specifications that are required for the consistency of DECODE and that will be used to select the provisional hardware platform.

4.1 Capability of running DECODE OS

The processor of the HUB needs to be able to run the DECODE OS. The DECODE OS is based on Devuan and is developed by Dyne who published the first release on April 2017. To assure compatibility the processor selected has to be a target of the DECODE OS SDK. The DECODE OS SDK is modeled after the Devuan SDK and supports all its architecture targets.

- ARM architecture targets:
- Raspberry PI 3 (raspi3 arm64)
- Raspberry PI 2 (raspi3 and raspi2)
- Raspberry PI 1 (raspi1 and raspi0)
- Chromebook Acer (chromeacer)
- Chromebook Veyron (chromeveyron)

All ARM SoCs from Allwinner Technology (Processor chip manufacturer), for which the best known products are the sunxi SoC series, such as the A10 (sun4i), A13 (sun5i) and A20 (sun7i) chips, which are very successful in the low-budget tablet market.

The list of current working targets from the sunxi SoC series are:

- Cubieboard 2
- Cubietruck
- Cubieboard4
- Cubieboard
- Cubietruck plus
- A10-OLinuXino-Lime
- A20-A20-OLinuXIno-LIME2
- A20-OLinuXino-Lime
- A20-OLinuXino-MICRO
- Bananapi
- Bananapro
- CHIP
- CHIP pro
- Lamobo R1

Orangepi 2

- Orangepi
- Orangepi lite
- Orangepi mini
- Orangepi plus
- Orangepi zero
- q8 a33 tablet 1024×600
- q8 a33 tablet 800×480

Dyne aims to increase the list of supported architectures in the future and they will pay particular attention to ARM64 targets due to their relevance as base architecture for the DECODE project.

For more information regarding the DECODE OS and the SDK, please refer to Deliverable 4.1 or the white paper.

Additionally to the capability to run the DECODE OS, the DECODE HUB needs to host enough computing power to act as local data broker, encrypt data and potentially run local storage of historical information. ARM processors have a good compromise of efficiency and computing power. They are commonly used in mobile devices and many chip manufacturers fabricate them in single, dual and quad core versions.

4.2 Transparency

One of the goals of the DECODE platform is to create a level playing field that enables developers from all backgrounds to contribute to the society by implementing innovative applications and opening up new economical, technological and social values based on the new infrastructure that DECODE will provide. In order to facilitate the participation of these developers, all the elements of the DECODE architecture should be open source. For that reason the HUB should be open source and compliant with the Open Source Hardware Association (OSHWA). Schematics, design files and documentation should be available for designers to build upon.

DECODE will encourage hardware designers to join their efforts in creating a more secure and open hardware.

Illustration 2: Open Source Hardware Logo

4.3 Deployability and availability

Deployability is a key factor to ensure DECODEs adoption and success. Therefore the HUB must be easily available. The most available potential HUB is any commercial computer. According to Barcelona data-sheet 2017[3] the Household ICT penetration is 88.3 in 2015 (% on population 16 to 74 years with a computer at home). Despite most of commercial computers not being open source this is a viable alternative for promoting an early adoption. For scenarios where an embedded device with a small form factor is required, low cost open source single board computers (SBCs) are a good alternative for the DECODE HUB.

[3]https://bcnroc.ajuntament.barcelona.cat/jspui/bitstream/11703/101684/1/Barcelona_Datasheet_2017.pdf.pdf

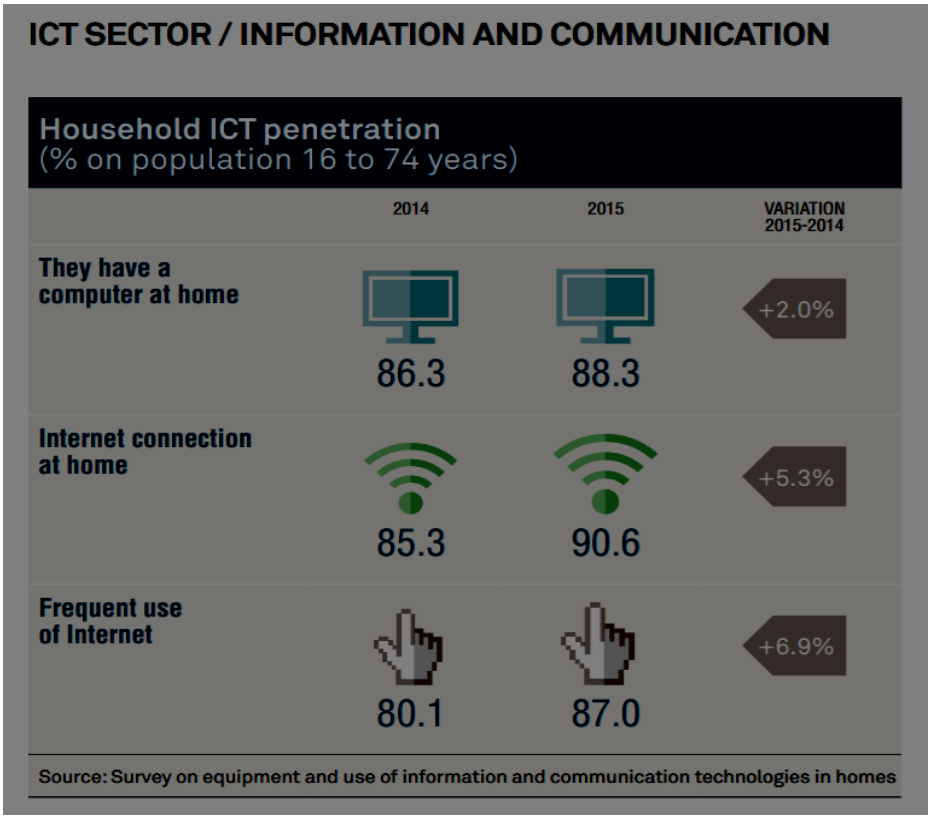


Illustration 3: Household ICT penetration BCN 2014/2015. City council Barcelona 2017

Embedded devices usually have a price between 25 and 150 euros and they often come with a small form factor which make them portable and easily installable. This embedded devices often require very low power consumption to run and most of them are based on very efficient ARM based processors that allows them to operate without the need of active cooling systems like fans or liquid coolers.

4.4 Hardware security

The need for privacy and confidentiality might differ according to the use cases of the platform. For instance, noise quality data might be less critical than health monitoring data or communication with the healthcare sector where a higher level of not only confidentiality but also essentially privacy might be required. This implies encryption in many cases. To build trust in the platform, the DECODE system needs to respond to the need for security.

Software-based security means that access conditions can be hacked and that logs can be tampered with. In addition, the data itself cannot be considered secure if left unprotected on a regular PC.

For this reason the DECODE platform user should consider the adoption of hardware-based security as well when the level of confidentiality required is higher. This measures should be made easy to interact with and effortless for the end-user.

Some of the hardware security features that the HUB could provide are:

Multiple factor authentication

The security protection provided by a single authentication method, e.g. a password, have proved to be weak. To make access conditions more secure, the system can ask for a multiple factor authentication. This could be the combination of a password with some token that the user have (like an RFID or BLE device) or biometric data like fingerprint readers or voice recognition.

Secure processors

Secure microcontrollers with built-in cryptographic engines and secure boot loader can guard against threats such as cryptanalysis intrusions, physical tampering, and reverse engineering. These secure microcontrollers are equipped with silicon-level anti-tampering features that allow to make them tamper resistant and to provide tamper evidence. The security keys, used to run cryptographic algorithms, need to be stored in a secure memory managed by a secure microcontroller and should only be accessible by the secure microcontroller and not from the outside of the memory.

Tamper avoidance

Anti-intrusion sensors can be incorporated into the electrical design to ensure someone tampering physically with the device would not have access to any sensitive data.

Side channel attacks protection

Side channel attacks consist in attacks based on information gained from the physical implementation of a system, rather than brute force or weaknesses in the algorithms. For example, timing information, power consumption, electromagnetic leaks or even sound can provide an extra source of information, which can be exploited to break the system.

The first step to protect against this types of threats is the reduction of the electromagnetic and sound radiations. Other measures include random calculations and delays introduced between normal operations and try to balance the power consumption of different data values.

Access attempt detection

Any attempt to access the system data remotely is detected. If the access is illegitimate (unauthenticated, coming from an unknown IP address, etc.), access is denied and in some cases data can be erased.

4.5 Connectivity

Decentralized transactions will be continuously being carried by the NODE. As a result, the NODE will be using the network resources in an extensive manner. To reduce the latency in the communication and to avoid bandwidth bottlenecks, a high speed Ethernet port should be part of the specifications. Many commercial prototyping platforms comes with an Ethernet port supporting 100Mb Ethernet or Gigabit Ethernet (GbE).

4.6 Expansion capabilities

To facilitate the development of IoT systems a flexible hardware design is necessary. For prototyping, it is useful to have a modular design that can be adapted to the maximum possible extent. The support of add-on sensors, actuators and a variety of wireless transceivers (e.g. Zigbee and/or LoRaWAN) to augment generic hardware is desirable. The prototyping platform should come with connectors and stackable headers that allow the access to GPIO pins and serial communication ports from the processor or microcontroller. However there is a trade-off between security and expansion. For scenarios where security is a priority, the HUB should be physically inaccessible and destroyed when accessed. In this case, IoT connectivity will be through embedded BLE and WiFi.

4.7 Processor availability

Some processor vendors do not provide datasheets of their products without the sign of a non-disclosure agreement (NDA). Moreover some of those vendors do not accept a NDA without a business case that assure the purchase of a minimal batch that can go up to thousands of units.

To facilitate the prototyping process and the future collaboration of other designers, the chosen processor should be available in small quantities and off-the-shelf without the need of signing NDAs.

4.8 Low cost/ low power

Efficiency and low cost are desired features for an IoT system. Depending on the particular set up of the IoT devices, these might need to be powered from batteries and have a small form factor to make them easy to install in any environment. Big and more powerful devices often have higher costs and require a higher energy demand making them not suitable for IoT.

5. Boards comparison and selection

5.1 Board list

Based on the HUB technical requirements, a selection of popular SBCs were listed as candidates for the provisional hardware platform. The features taken into consideration for the selection of these boards are, an open source hardware design (schematics minimum, some of them includes design files), Gigabit Ethernet port, availability of GPIO pins and serial communication and capability of running an operative system. Other features such as availability of processor, price and support were also considered.

Down below is a list[4] [5] of commercial single board computers that are close to the requirements mentioned before. This list is not exhaustive and only represents a subset of boards that were taken into consideration. Most of the boards runs on a dual core or quad core ARM based processors.

BOARD	CPU	CORE	W/LAN	GPIO	OS	PRICE($)
A20-A20-OLinuXIno-LIME2	Allwinner A20	2x A7 @ 1GHz	WiFi(opt) GbE	160	Linux/ Android	47+8
Banana Pi BPI-M64	Allwinner A64	4x A53 @ 1.2GHz	WiFi,BT GbE	40	Linux/ Android	74
Banana Pro	Allwinner A20	2x A7 @ 1GHz	WiFi,BT GbE	40	Linux/ Android	48
Cubieboard 4	Allwinner A80	4xA15@2GHz	WiFi,BT GbE	67	Linux/ Android	120

[4] http://linuxgizmos.com/ringing-in-2017-with-90-hacker-friendly-single-board-computers/

[5] https://www.board-db.org/

		4xA7@1.3GHz				
Firefly FirePrime	Rockchip RK3128	4x A7 @ 1.3GHz	WiFi,BT GbE	84	Linux/ Android	79
Firefly RK3288	Rockchip RK3288	4x A17@ 1.8GHz	WiFi,BT GbE	84	Linux/ Android	129
HummingBoard-Gate	NXP i.MX6	1x/2x/4x A9 @ up to 1.2GHz	WiFi(opt) GbE	36	Linux/ Android	83
Inforce 6410Plus	Qualcomm Snapdragon	4x A15 @ 1.7GHz	WiFi,BT GbE	12	Linux/ Android	143
LinkSprite Arches	Allwinner A80	4xA15@ 2GHz 4xA7@1.3GHz	WiFi,BT GbE	32	Linux/ Android	95
MinnowBoard Turbot	Intel Atom E3826	2x x86 @ 1.46GHz	WiFi(opt) GbE	40	Linux/ Android	140
NanoPi M2	Samsung S5P4418	4x A9 @ 1.4GHz	WiFi(opt) GbE	40	Linux/ Android	25
Odroid-C2	Amlogic S905	4x A53 @ 1.5GHz	WiFi(opt) GbE	40	Linux, Android	40+8
Odroid-XU4	Samsung Exynos5422	4xA15@ 2GHz 4xA7@1.4GHz	WiFi(opt) GbE	40	Linux/ Android	74

OrangePiPlus2/Plus2E	Allwinner H3	4x A7 @ 1.6GHz)	WiFi, GbE	40	Linux/ Android	49
Pine A64	Allwinner A64	4x A53 @ 1.2GHz	WiFi(opt) GbE	40	Linux/ Android	29
Udoo X86	Intel Braswell	4x Braswell @ 2.56GHz	WiFi(opt) GbE	14	Linux/ Android	125
Wandboard	NXP i.MX6	1x, 2x, or 4x A9 @1GHz	WiFi(opt) GbE	10	Linux/ Android	79

5.2 Provisional Hardware Platform

After reviewing the specifications of the embedded platforms and in agreement with Dyne, the chosen board for the provisional hardware platform is the **A20-OlinuXIno-LIME2**[6]. This board is designed by an open-source hardware company called Olimex[7], based in Bulgaria. The platform supports most of the features that we were looking for the provisional hardware: The A20-OLinuXIno-LIME2 is a target of the DECODE OS, It is completely open source (all schematics and board design files are available to the customer under the Creative Commons Attribution-ShareAlike 3.0 Unported License), they provide the design files in KiCad (an open source PCB design tool), It supports 1000MB Ethernet, contains a SATA port and has expansion capabilities. The A20-OLinuXIno-LIME2 contains an Allwinner A20 chip that host a dual core ARM A7 processor that can be obtained in small quantities with no need of NDAs. The processor Allwinner A20 is full supported by the community from linux-sunxi 3.4 kernel and later.

However this decision might change during the curse of the project if a newer more suitable option is found.

[6] https://www.olimex.com/Products/OLinuXino/A20/A20-OLinuXIno-LIME2/open-source-hardware

[7] https://www.olimex.com/

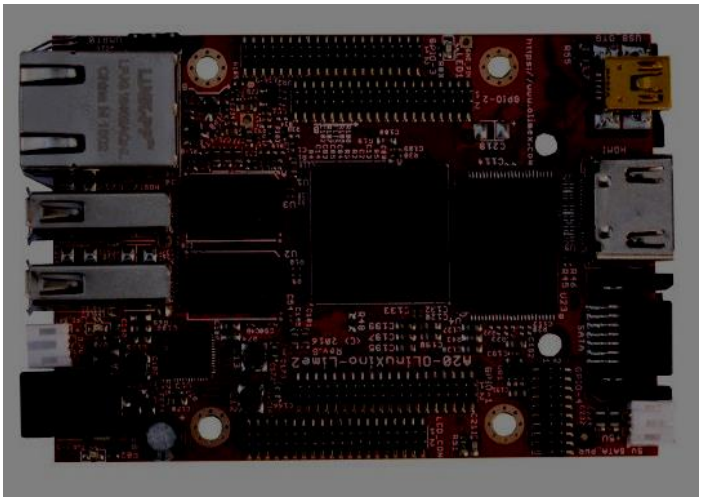


Illustration 4: A20-OLinuXino-LIME2 Single Board Computer

The full list of technical specifications from the user manual[8] are listed below:

- A20 Cortex-A7 dual-core ARM Cortex-A7 CPU and dual-core Mali 400 GPU
- 1GB DDR3 RAM memory
- 1000MBit native Ethernet
- 4GB NAND FLASH memory (available only on the 4GB version of the board)
- 4GB eMMC memory (available only on the eMMC version of the board)
- SATA connector with 5V SATA power jack
- Native HDMI
- Capable of Full HD (1080p) video playback
- 2x USB Low-Full-High-Speed hosts with power control and current limiter
- USB-OTG with power control and current limiter
- LiPo Battery connector with battery-charging capabilities
- LCD row of pins (0.05" step)
- 160 GPIOs on four GPIO rows of pins (0.05" step)
- MicroSD card connector
- DEBUG-UART connector for console debug with USB-SERIAL-CABLE-F
- Status LED, power LED and battery charge status LED
- 2KB EEPROM for MAC address storage and more
- 2 BUTTONS with ANDROID functionality + RESET button
- 2 mount holes
- 5V input power supply, noise immune design
- PCB dimensions: (84 × 60)mm ~ (3.3 × 2.4)"
- Storage: micro SD

[8] https://www.olimex.com/Products/OLinuXino/A20/A20-OLinuXIno-LIME2/resources/A20-OLinuXino-LIME2-UM.pdf

- Price: 55 Eur

5.3 Other alternatives:

The Raspberry pi 3 is a SBC developed by the Raspberry Pi Foundation. The Raspberry is a well designed board, has a lot of software support coming from a large community and is very popular for fast prototyping. Adding sensors to this board often requires minimal effort in installation and configuration. This SBC was used in the Prototype 1: Air quality App as specified in the Deliverable 1.1. There are however 3 main reasons that prevented the Raspberry Pi 3 from being selected as a first candidate. The first one is the lack of GbE, this SBC only supports 100Mb Ethernet. The second reason is the difficulty of acquiring the Broadcom BCM2837 processor. Broadcom have licenses around the BCM2835 that do not comply with the Open-source prerequisites for DECODE. NDAs are required to acquire the SoC datasheet and they do not allow signing them without providing a business model and estimation of how many chips are going to be sold. In addition to this, Raspberry Pi do not provide design files of their products, they do however provide schematics of their designs.

The Odroid family of boards are known to be cheap, and powerful single board computers. However at the time of this deliverable, none of them were Devuan targets. This presents a problem if we wanted to start testing the DECODE OS on an embedded platform since we would need to wait for Devuan support. In addition to this, ODROID do not provide design files of their products, they do however provide schematics of their designs.

The trade-off between power consumption and performance was also an aspect to consider in the selection of the provisional platform. Quad-core SBCs provides greater performance than dual-cores or single cores but they consume more power, decreasing the battery life of the device. In addition, the performance of the processor can be severely compromised due to overheat. For most Quad-core SBCs extra hardware is required to keep the temperature of the processor low.

The A64-OlinuXino board is another board from Olimex with an All-winner A64 processor (1.2 GHz Quad-Core ARM Cortex-A53, 64-bit). The board looks promising but it is not yet available at the time. It might be tested after this deliverable once is available.

Many of the boards seems to offer an excellent set of hardware features. However in practice many of them lacks a good software support or proper documentation, making them poor alternatives for DECODE.

6. Hardware setup and system installation

The following section will describe the steps carried to install Devuan and set up the tests. We are adding these steps to make our set up reproducible for other partners and easy to debug.

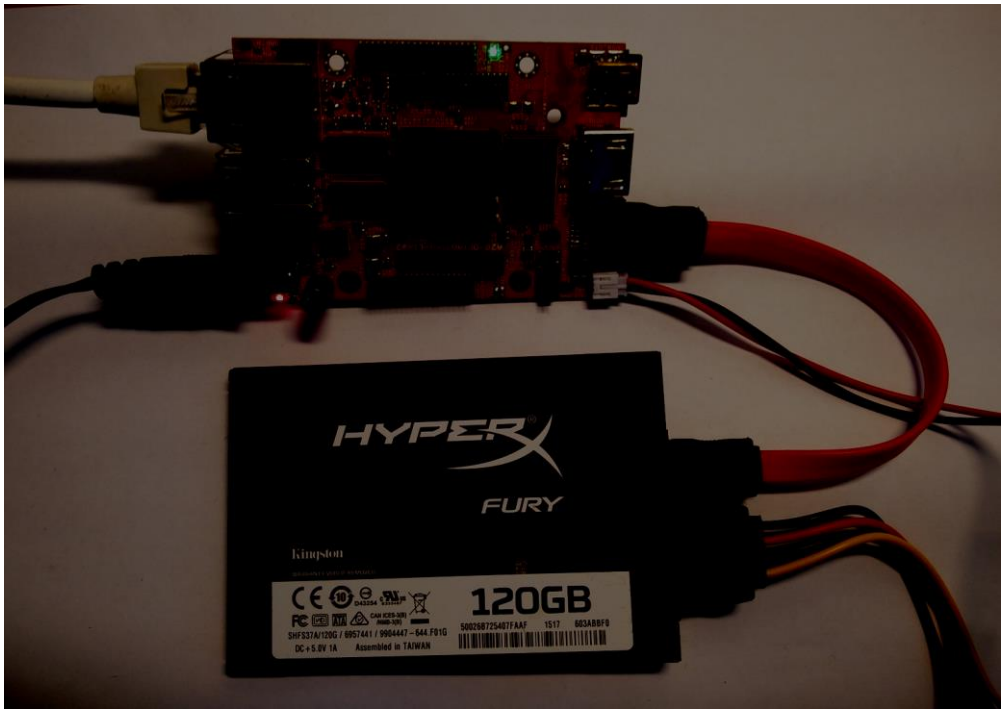


Illustration 5: Hardware setup of the A20-OLinUXino-LIME2

6.1 Flashing system iso to SD-card

Download the image you want:

```
$ curl -O https://files.devuan.org/devuan_jessie/embedded/devuan_jessie_1.0.0_armhf_sunxi.img.xz
```

Download the shasums and the signature:

```
$ curl -O https://files.devuan.org/devuan_jessie/embedded/SHA256SUMS
$ curl -O https://files.devuan.org/devuan_jessie/embedded/SHA256SUMS.asc
```

Verify the signature:

```
$ sha256sum -c SHA256SUMS
```

```
$ devuan_jessie_1.0.0_armhf_sunxi.img.xz: OK  ←This line is output from the command above
```

Use dd to write the raw disk image to a SD card

```
$ xzcat devuan_jessie_1.0.0_armhf_sunxi.img.xz | sudo dd of=/dev/mmcblk0 bs=2M
```

6.2 Flash the U-boot Blob

Download the u-boot blob and flash it to your memory card

```
$ curl -O https://files.devuan.org/devuan_jessie/embedded/u-boot/A20-A20-OLinuXIno-LIME2_defconfig.bin
$ sudo dd if=20-A20-OLinuXIno-LIME2_defconfig.bin of=/dev/mmcblk0 bs=1024 seek=8 && sync
```

After flashing the memory card, insert it in the Olinuxino board and plug in the power cable . Use the main admin username "root" and the password "toor", to gain root access.

6.3Extra configurations

Add a user to the system and give admin rights (sudo).

Add a non-root user:

```
$ mkuser username
```

Give the new user sudo priviledges:

```
$ usermod -a -G sudo username
```

SATA Installation: Read/Write permissions for the non-root user.

```
$ cd /media/your_external_drive
$ sudo chmod -R -v 777 *
```

Setting up a graphical user interface

Installing the lxde graphical user interface

```
$ apt-get update
$ apt-get install --no-install-recommends lxde
$ apt-get install lightdm
$ reboot
```

Finally, reboot the card and login with credentials of the new user you just made. This will now be done in a graphical instead of command line.

- Note on Graphics Drivers

The mainline kernel works fine for a basic Linux desktop system running on top of a simple frame buffer. This is good enough for the users who do not require 3D graphics or video playback acceleration. If 3D graphics are needed the Linux kernel needs to replaced. We did not explore this further, since 3D graphics and/or video playback seems redundant in the current state of the project.

6.4 Java Installation

JVM is one of the options considered as the base for the DECODE core. In order to run the JVM performance tests, Java installation is required. We used the default java-packages available in the Devuan/Debian repository: default-jre & default-jdk. These are installed with this command:

```
$ sudo apt-get install default-jre default-jdk
```

The following information about Java on the system was compiled by SPECjvm2008:

JVM-info (Java virtual machine)	
Vendor	Oracle Corporation
Vendor URL	http://java.oracle.com/
JVM name	OpenJDK Zero VM
JVM version	24.131-b00 interpreted mode
JVM available	n/a
Java Specification	1.7
JVM command line startup	
JVM launcher startup	default
Additional JVM tuning	
JVM class path	SPECjvm2008.jar
JVM boot class path	/usr/lib/jvm/java-7-openjdk-armhf/jre/lib/resources.jar: /usr/lib/jvm/java-7-openjdk-armhf/jre/lib/rt.jar: /usr/lib/jvm/java-7-openjdk-armhf/jre/lib/sunrsasign.jar: /usr/lib/jvm/java-7-openjdk-armhf/jre/lib/jsse.jar: /usr/lib/jvm/java-7-openjdk-armhf/jre/lib/jce.jar:

	/usr/lib/jvm/java-7-openjdk-armhf/jre/lib/charsets.jar: /usr/lib/jvm/java-7-openjdk-armhf/jre/lib/rhino.jar: /usr/lib/jvm/java-7-openjdk-armhf/jre/lib/jfr.jar: /usr/lib/jvm/java-7-openjdk-armhf/jre/classes

6.5 VNC Installation

Virtual Network Computing (VNC) is a graphical desktop sharing system that uses the Remote Frame Buffer protocol (RFB) to remotely control another computer. It transmits the keyboard and mouse events from one computer to another, relaying the graphical screen updates back in the other direction, over a network.[9] VNC or a VNC-like protocol could be used to build an interface for interacting with the node on a user level. VNC can also be used as a way of testing the board performance on heavy network loads.

We installed TightVNC on top of Devuan for it to run a VNC server. This worked correctly when using a SSH-tunnel on the VNC client. We did this so that port 22 was used instead of port 5901 (which is usually blocked by firewalls).

We did not try to use VNC through an external IP or domain. This can be further explored if needed.

Starting the VNC server.

```
vncserver -depth 16 -geometry 1024x768 :1 -localhost
```

Connecting to the server

You need to setup a ssh-tunnel on VNC client computer, like this:

```
$ ssh -L 5901:127.0.0.1:5901 -N -f -l vnc 172.16.12.182
```

View/accessing the VNC-server from a VNC client

Use your favourite VNC viewer to open a connection to: localhost:5901. We used the VNC viewer Vinagre for connecting to the board. We were running Gnome3 on top of Ubuntu on the client computer. No major lags were noticed during the test.

[9] *Richardson, T.; Stafford-Fraser, Q.; Wood, K. R.; Hopper, A. (1998). "Virtual network computing" (PDF). IEEE Internet Computing. **2**: 33. doi:10.1109/4236.656066.*

Adding COPY/PASTE functionality

Its handy to be able to copy and paste between the server and client. To do this: first install autocutsel:

```
$ sudo apt-get install autocutsel
```

Open your /home/username/.vnc/xstartup and add **autocutsel -fork** like this:

```
$ nano /home/username/.vnc/xstartup
```

It should look like this:

```
#!/bin/sh
export XKL_XMODMAP_DISABLE=1

autocutsel -fork

exec ck-launch-session startlxde        #for an lxde session
#exec ck-launch-session gnome-session   #for a gnome-session
#exec ck-launch-session openbox-session #for an openbox desktop
#exec ck-launch-session startxfce4      #for an xfce desktop
#exec ck-launch-session icewm           #for an icewm desktop
#exec ck-launch-session startkde        #for a kde desktop
exec pcmanfm --desktop
```

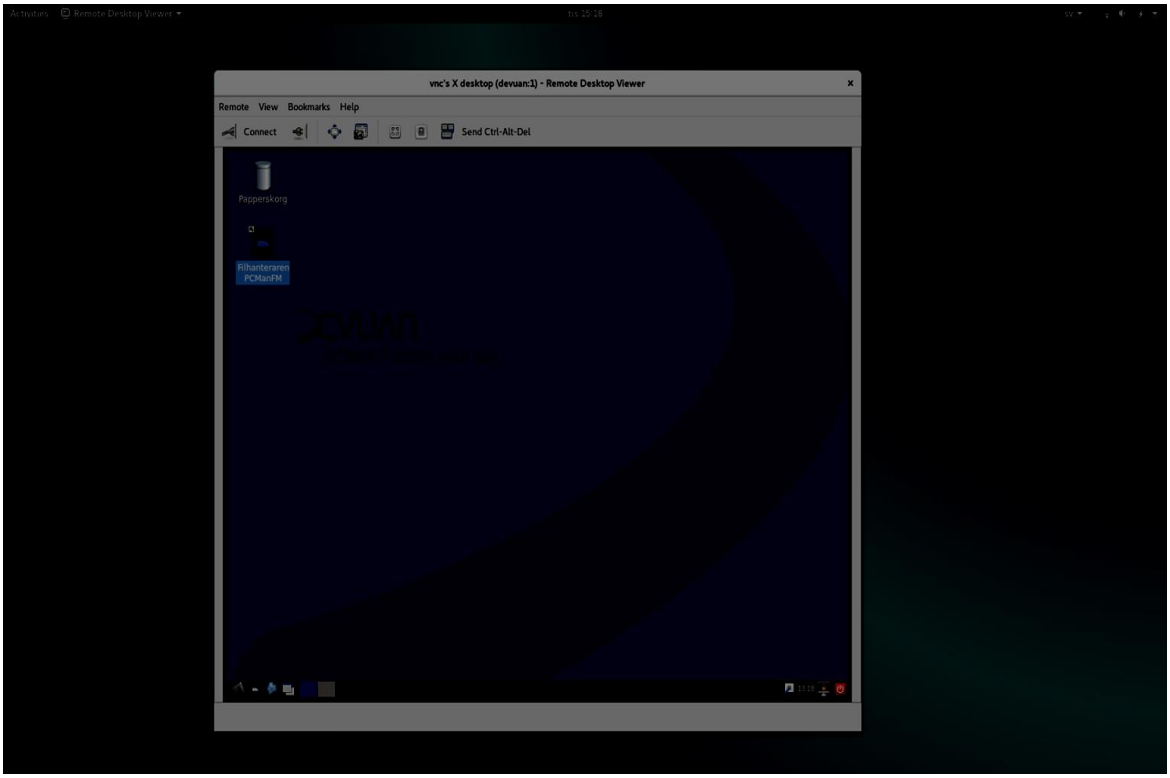


Illustration 6: A screen-shot from a VNC client computer showing the Desktop GUI running on the board.

7. Benchmarking the board

The following section will be dedicated to explain the first round of performance tests that were carried on the A20-OLinuXIno-LIME2 board. These tests will serve as a baseline benchmark to compare it with other single board computers. in the future, more realistic testing will be made once the definition of the architecture is more concrete. These tests will be presented in the deliverable *4.5-Benchmark and comparison of one-board computers and hardware specifications*.

7.1 Spec JVM2008

SPECjvm2008 is a benchmark suite for measuring the performance of a Java Runtime Environment (JRE). It contains several benchmarks focusing on core java functionality. The SPECjvm2008 workload mimics a variety of common general purpose application computations, measuring basic Java performance on a wide variety of both client and server systems.

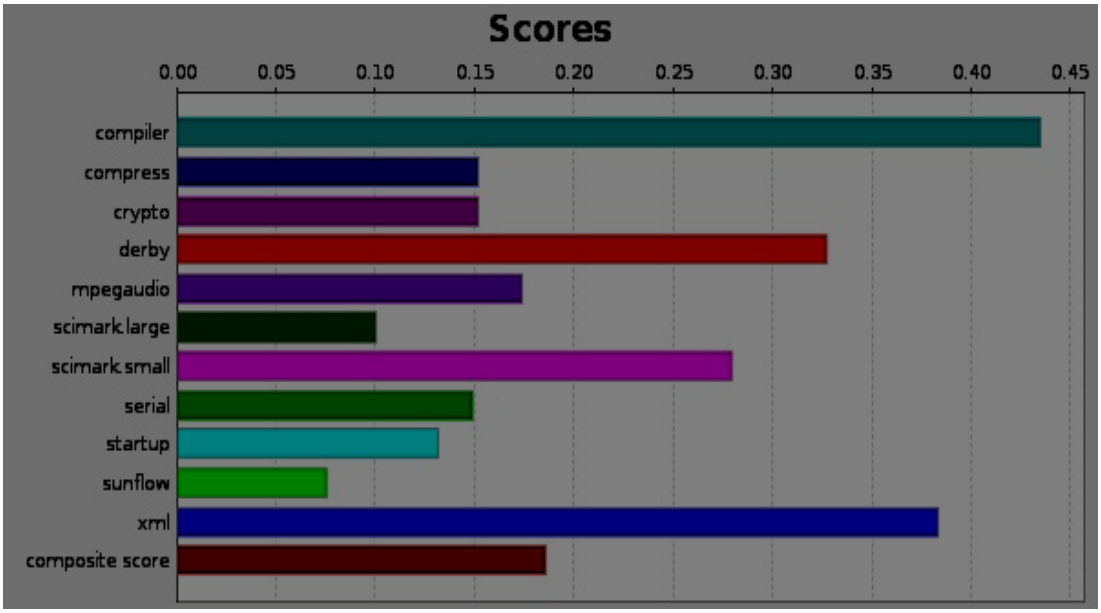
SPEC also finds user experience of Java important, and the suite therefore includes startup benchmarks and has a required run category called base, which must be run without any tuning of the JVM to improve the out of the box performance.[10]

We ran the whole SPECjvm test on the board under the run category called base, which must be run without any tuning of the JVM to improve the out of the box performance. These are the results:

Benchmark	Operation per minute (Ops/m)
compiler	0.43
compress	0.15
crypto	0.15
derby	0.33
mpegaudio	0.17
scimark.large	0.1
scimark.small	0.28
serial	0.15

[10]Standard Performance Evaluation Board: https://www.spec.org/jvm2008/

startup	0.13
sunflow	0.08
xml	0.38



The non-composite results from the cryptography part of the test:

Benchmark name	Configuration	Iteration	Expected run time (millis)	Actual run time (millis)	Operations	ops/m
crypto.aes	number of threads: 2 warmup time: 120000 run time: 240s	warm up	120000	120000	0.15	0.08
		iteration 1	240000	240000	0.31	0.08
crypto.rsa	number of threads: 2 warmup time: 120000 run time: 240s	warm up	120000	120000	0.59	0.29
		iteration 1	240000	240000	1.17	0.29
crypto.signverify	number of threads: 2	warm up	120000	120000	0.31	0.15

	warmup time: 120000 run time: 240s					
		iteration 1	240000	240000	0.62	0.15

7.2 Open Benchmarking / Phoronix Test Suite

Phoronix Test Suite (PTS) is a free, open-source benchmark software for Linux and other operating systems which is developed by Phoronix Media with cooperation from an undisclosed number of hardware and software vendors. The Phoronix Test Suite has been endorsed by sites such as Linux.com, LinuxPlanet and has been called "the best benchmarking platform" by Softpedia [11].

The following information about the system was compiled by the Phoronix Test Suite:

Hardware information
PROCESSOR: ARMv7 rev 4 @ 0.96GHz (2 Cores) Core Count: 2 Scaling Driver: cpufreq-dt performance GRAPHICS: simple framebuffer OpenGL: 2.1 Mesa 10.3.2 Display Driver: modesetting 0.9.0 Screen: 1920x1080 Motherboard: Allwinner sun7i (A20) Family Olimex A20-OLinuXIno-LIME2 Memory: 1024MB Disks: 120GB KINGSTON SHFS37A + 32GB SD card File-System: ext4 Mount Options: data=ordered errors=remount-ro noatime rw

[11] http://news.softpedia.com/news/The-Best-Benchmarking-Platform-Phoronix-Test-Suite-87396.shtml

Disk Scheduler: CFQ Operating system: Devuan 1.0 Kernel: 4.11.2 (armv7l) Desktop: LXDE 0.7.2 Display Server: X Server 1.16.4 Compiler: GCC 4.9.2

We ran the following disk-related benchmarks:

Disk-related benchmarks	
io-stress: Random Write The basic idea behind AIO is to allow a process to initiate a number of I/O operations without having to block or wait for any to complete. At some later time, or after being notified of I/O completion, the process can retrieve the results of the I/O.[12] The aio-stress is a basic utility for testing the Linux kernel AIO api. It can benchmark and help validate AIO changes for files and block devices, testing both O_DIRECT and buffered io.[13]	2.19 MB/s
sqlite: Timed SQLite Insertions SQLite is an embedded SQL database engine. Unlike most other SQL databases, SQLite does not have a separate server process. SQLite reads and writes directly to ordinary disk files.[14]	97.32 seconds
fs-mark: 1000 Files, 1MB Size The fs_mark benchmark tests synchronous write workloads. It can vary the number of files, directory depth, etc. It has detailed timings for reads, writes, unlinks and fsyncs that make it good for simulating mail servers and other setups.	16.57 Files/s
tiobench: 64MB Random Read - 32 Threads	133.28 File/s

[12]IBM, Ihttps://www.ibm.com/developerworks/library/l-async/index.html

[13]Oracle,https://oss.oracle.com/~mason/aio-stress/

[14]Sqlite, https://www.sqlite.org/about.html

tiotest is a file system benchmark especially designed to test I/O performance with multiple running threads.[15]	
tiobench: 64MB Random Write - 32 Threads	5.98 MB/s
compilebench: Test: Compile Compilebench tries to age a filesystem by simulating some of the disk IO common in creating, compiling, patching, stating and reading kernel trees. It indirectly measures how well filesystems can maintain directory locality as the disk fills up and directories age. Thanks to Matt Mackall for the idea of simulating kernel compiles to achieve this.[16]	3.23 MB/s
compilebench: Test: Initial Create	4.36 MB/s
compilebench: Test: Read Compiled Tree	15.90 MB/s
postmark: Disk Transaction Performance Postmark is a single threaded benchmark, but Auto-pilot can automatically run several concurrent processes and analyze the results. Postmark generates a small workload by default: only 500 files are created and 500 transactions performed. Auto-pilot increases the workload to a pool of 20,000 files, and performs 200,000 transactions by default.[17]	84 Transactions/s
compress-gzip: 2GB File Compression This test measures the time needed to compress a file using Gzip compression.[18]	125.22 seconds
apache: Static Web PageThis is a test of ab, which is the Apache benchmark program. This test profile measures how many requests per second a given system can sustain when carrying out 1,000,000 requests with 100 requests being carried out concurrently.[19]	657.09 Request/s

[15]https://linux.die.net/man/1/tiotest

[16]https://oss.oracle.com/~mason/compilebench/

[17]http://www.filesystems.org/docs/auto-pilot/Postmark.html

[18]https://openbenchmarking.org/test/pts/compress-gzip

[19]https://openbenchmarking.org/test/pts/apache

8. Conclusion

This document covers the hardware requirements for the DECODE HUB and provides a candidate platform to use in the pilots.

Despite an open custom made hardware for DECODE which would be ideal in terms of increasing security and transparency, this approach will put a barrier for the adoption of DECODE since every user would require this particular hardware to run DECODE. In order to assure the adoption of DECODE, the DECODE OS will be prepared to run in any commercial computer.

However not all the cases can run on a regular PC or laptop. An embedded platform is needed for IoT use cases like the Making Sense pilot. The selected one board computer, the A20-OLinuXIno-LIME2 from Olimex, seems to have all the prerequisites to act as a provisional hardware platform for the IoT scenario. The preliminary tests shows a successful load of the Devuan OS, SATA drive configuration, VNC application and JAVA performance.

The initial benchmarking presented in this document will be used as a baseline and will be expanded with more realistic testing once the architecture definition is more concrete.

This provisional hardware platform will cover the elementary requirements to run the tests of the pilots but do not provide any type of hardware-based security. For critical applications were confidentiality level is high, it is recommended to adopt a platform that take this into consideration.

During the life of the project this option might change if a newer more suitable option comes out to the market. We will pay particular attention to processors supporting the 64-bit ArmV8 architecture since they seem to offer a good trade off for performance and efficiency and the DECODE OS is ready to target it.

Future work

Arduino will work in close collaboration with Dyne to establish test benchmarks related to DECODE computational tasks to evaluate different one-board hardware solutions. This work will be presented in Deliverable 4.5: *Benchmark and comparison of one-board computers and hardware specifications,* with due date in December 2017.