H2020-ICT-2016-1 - 731736640 - 31/03/2017

eCraft2Learn

Digital Fabrication and Maker Movement in Education
Making Computer – supported Artefacts from Scratch

Deliverable D4.1 (ver5)

Architecture Analysis and Design

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No 731345.

PROJECT DESCRIPTION

Acronym:	**eCraft2Learn**
Title:	Digital Fabrication and Maker Movement in Education: Making Computer-supported Artefacts from Scratch
Coordinator:	University of Eastern Finland
Reference:	731345
Type:	RIA
Program:	HORIZON 2020
Theme:	Technologies for Learning and Skills
Start:	01. January, 2017
Duration:	24 months
Website:	http://www.project.ecraft2learn.eu/
E-Mail:	office@ecraft2learn.eu

Consortium:

University of Eastern Finland, Finland, (UEF), Coordinator

Edumotiva, Greece (EDUMOTIVA)

Mälardalen University of Sweden, Sweden (MDH)

Zentrum für Soziale Innovation, Austria, (ZSI)

The University of Oxford, United Kingdom, (UOXF)

Synyo GmbH, Austria, (SYNYO)

University of Dundee, Scotland, (UNIVDUN)

University of Padua, Italy, (UNIPD)

Technopolis City of Athens, Greece (TECHNOPOLIS)

Evothings, Sweden (EVOTHINGS)

Arduino, Sweden (ARD)

Ultimaker, United Kingdom (ULTIMAKER)

DELIVERABLE DESCRIPTION

Number:	**D4.1**
Title:	**Architecture Analysis and Design**
Lead beneficiary:	**UNIVDUN**
Work package:	WP4
Dissemination level:	Public (PU)
Type	Report (R)
Due date:	31.03.2017
Submission date:	31.03.2017
Authors:	**Andrea Alessandrini,** UNIVDUN
Contributors:	**Afshin Ameri,** MDH
	Baran Curuklu, MDH
	Marie Ehrndal, ARD
	Margit Hofer, ZSI
	Bernhard Jaeger, SYNYO
	Alex Jonsson, EVOTHINGS
	Adam Linson, UNIVDUN
	Emanuele Menegatti, UNIPD
	Michele Moro, UNIPD
Reviewers:	**Ken Kahn,** UOXF
	Calkin Suero Montero, UEF

Version Control

Version	Date	Person in charge (Organization)	Changes	Quality Assurance
1	5.03.2017	Andrea Alessandrini (UNIVDUN)	Early Draft Version	UNIVDUN, MDH, UOXF
2	10.03.17	Andrea Alessandrini (UNIVDUN)	First Draft Version -integrating partner comments	UNIVDUN, MDH, UOXF, SYNYO, ZSI, ARD
3	15.03.2017	Andrea Alessandrini (UNIVDUN)	Second Draft Version -added parts -integrating partner comments	UNIVDUN, MDH, UOXF, SYNYO, ZSI, UNIPD
4	17.03.2017	Andrea Alessandrini (UNIVDUN)	- Draft for Internal review - Proof read	UOXF
5	28.03.2017	Andrea Alessandrini (UNIVDUN)	-integrating UOXF comments - Proof read	UEF

Acknowledgement: This project has received funding from the European Union's Horizon 2020 Research and Innovation Action under Grant Agreement No 731345.

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

EXECUTIVE SUMMARY

This report presents information about the vision and early definition for the eCraft2Learn technological environment with a focus on the software components and early design requirements. The document gives an overview of technologies for programming, making, and printing artefacts. The document presents the eCraft2Learn technological and educational objectives. It shows the methods used to review and evaluate the eCraft2Learn technologies. The report also introduces activity scenarios and early design specifications. The description of technologies, systems and products to consider for satisfying the eCraft2Learn objectives is given, followed by an analysis of the programming languages and environments. The document concludes with a proposed initial design concept for the unified user interface and the eCraft2Learn architecture.

1 INTRODUCTION

1.1. ABOUT THIS DELIVERABLE

This deliverable, D4.1, provides the vision and early definition for the eCraft2Learn technological environment with a focus on the software components, as well as design requirements for future work toward the implementation of this system. The outcomes of this deliverable take into account the pedagogical outputs, through the work carried out in parallel in Task 3.1, without limiting the bottom-up manner of investigating possible solutions. This approach is important, considering that the proposed solutions ought to be valid in various pedagogical settings including informal ones, such as after-school activities.

1.2. ORGANISATION OF THIS DOCUMENT

The document presents an overview of technologies for programming, making, and printing artefacts. The document first introduces the eCraft2Learn technological and educational objectives. It then presents the methods used to review and evaluate the eCraft2Learn technologies. Next, it presents activity scenarios and design specifications for the eCraft2Learn unified interface. Following a description of technologies, systems and products to consider for fulfilling the eCraft2Learn objectives, the document continues with an analysis of the programming languages and environments, based on a SWOT approach (assessing Strengths / Weaknesses / Opportunities / Threats). The document concludes with a proposed initial design concept for the unified user interface and the eCraft2Learn architecture.

1.3. THE LEARNING CONTEXT AND THE ECRAFT2LEARN TECHNOLOGICAL ENVIRONMENT

The system that will be developed aims to provide technological support to formal and informal pedagogical approaches in real-life scenarios (e.g. curricular activities, such as classroom projects, and extracurricular activities, such as after-school workshops). The prime age group the project aims to reach is 13-17 year olds. The educational model we use is the five-stage eCraft2Learn craft- and project-based pedagogy: ideate - plan - create - program - share.

In this context two different user groups are identified: learners and teachers/coaches. In the context of this work, these two groups are equally important, although the main goal of the eCraft2Learn project is to positively affect the learning processes of learners. Teachers can also become learners, especially with respect to their evolving role from a traditional teacher

to a coach (please see D3.1). A learner can also lead or coach his/her learner group. In this case, the leader may interact with group members in ways similar to a formal teacher, although typically, a peer leader would not require the same set of analytical tools (e.g., for formal assessment) that teachers use. For this reason, different roles may be associated with a system identity, such that the same user may have different rights based on their momentary role (e.g. Sally may teach in a school classroom where she uses the system for formal assessment of her students; she may also teach informally after school, in which case she uses the same system account identity, but in a coaching role that does not require giving marks).

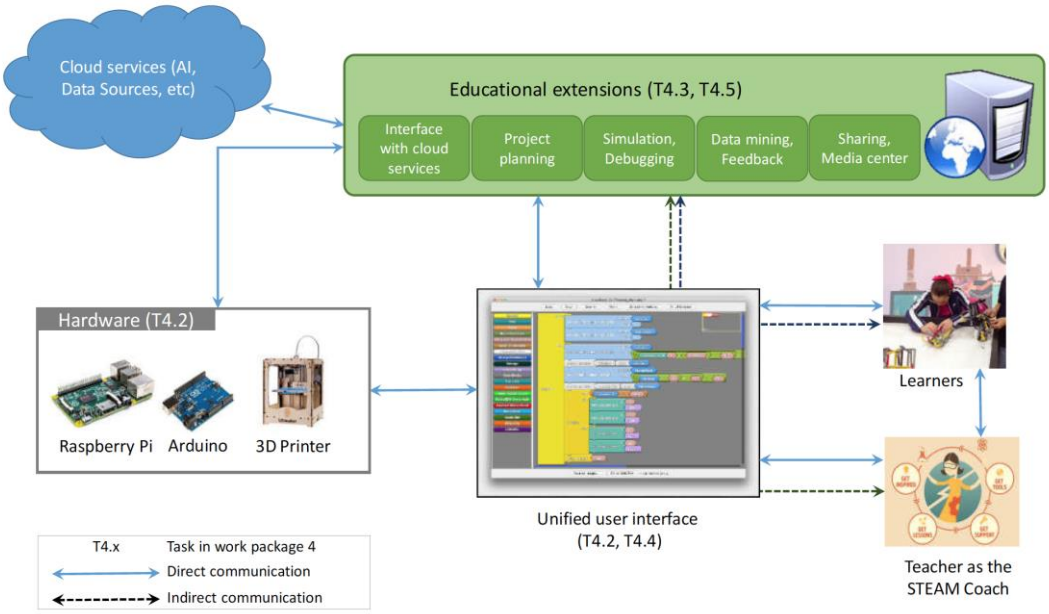


Figure 1: eCraft2Learn technological core

To this end, the eCraft2Learn technological core connects new technologies with existing ones that together contribute to the learning experience. The framework that defines the appropriate usage of these technologies is the five-stage eCraft2Learn craft- and project-based pedagogy (ideate - plan - create - program - share).

Stage 1 - *Ideate*. This stage allows learners to investigate their ideas in a manner that is not rigidly controlled (roughly equivalent to what is commonly referred to as “brainstorming”). Although there may be a clear goal of the project, the activities in this stage should make use of exploratory, open-ended searching and browsing. It is plausible to assume that the activities will be carried out both in the classroom and in other locations, including outside. Typical activities may include exploring the world physically (e.g., taking pictures, exploring

situations outside the classroom, etc.) or virtually (e.g., through online community discussion). These explorations will conclude by drawing from them a specific challenge in relation to the overall educational objective (e.g. a biology lesson on DNA might lead from viewing animations to building an interactive DNA model). This process can be guided by the STEAM coach based on the context and specific needs of the learners.

Stage 2 - *Plan*. In this stage, the learners begin the process of planning how a specific project will be carried out. Once the challenge has been defined, the learners will start to collect information to make a project plan. This may include getting feedback from the STEAM coach on their project plan, and also negotiating the roles for group members based on skill levels (e.g. previous coding experience), interests (e.g. favouring scientific or artistic aspects), and physical resource sharing constraints (e.g. a shared 3D printer). Here, learners must systematically collect, classify, and store project material that is of interest (e.g. gathering from a shared repository pre-existing 3D models, circuit diagrams, code, etc. that could be adapted to a new project).

Stage 3 - *Create* and Stage 4 - *Program*. In creation, learners embark on the co-design and co-creation of their computer-supported artefact solutions through the application of DIY technologies. The visualisation and simulation of designs are important parts of Stage 3. In Stage 4, programming, the learners will develop computer programs to add functionality to their artefacts. This stage includes software debugging and integrated SW/HW simulation.

Depending on the specific project, creation and programming can proceed linearly, or the two stages may iteratively cycle (back and forth) until a stable design with the desired functionality is achieved. Working in this manner to evolve and incrementally improve a design is a common engineering method. Some of the possible relationships between hardware and software stages are indicated in Figure 2, below.

Stage 5 - *Share*. Learners will be encouraged to share and showcase their projects and implementation ideas, through the open (online) community or through eCraft2Learn dissemination events. In return, they will receive feedback from artists, designers, and engineers worldwide. Learners will also participate in eCraft2Learn dissemination events, where they will showcase their projects to the community in general.

1.4. METHODOLOGY AND APPROACH

In this section, we describe the operationalisation of Task 4.1 by the WP4 consortium partners, with a focus on the process of screening different possible resources for our collection of programmes, clustered into several categories:

a) Learning environments
b) Programming languages
c) Programming interfaces/environments
d) Artificial Intelligence services (cloud APIs)
e) 3D modelling tools/environments.

These categories are used by educators and learners, which suggests that they are promising candidates for inclusion in the eCraft2Learn unified user interface (UUI).

Following the raw data gathered from several sources and threads, we refined our choice during collocated and online meetings with the input of the WP4.1 partners and (later) the entire eCraft2Learn consortium.

Resources were evaluated using a triangulation of methodologies, leading to the establishment of the (preliminary) eCraft2Learn architectural design. The evaluation task was approached from different perspectives, and encompassed a number of qualitative methods for deciding on appropriate tools. Following a SWOT analysis of each potential resource, we conducted a more in-depth assessment of alternatives. We gathered insights from interviews, demonstrations, and observations, in relation to the detailed activity scenarios elaborated in section two of this document.

A SWOT analysis (Strengths - Weaknesses - Opportunities - Threats) is a strategic analytical tool that combines an assessment of strengths and weaknesses (e.g. of an organisation, a product design, etc.) with an assessment of opportunities and threats posed by the target of the analysis (e.g. a design for a very small mobile phone may reflect a strength, given its light weight and portability, while it may also pose a threat, for instance, of being easily lost). This approach was applied to each of the technologies, grouped into different categories.

Our aim was to facilitate and provide justification for making decisions regarding which possible elements will be included in the eCraft2Learn interface architecture, with due consideration of the full potential of the eCraft2Learn ecosystem aimed at the end-users.

Thus, the eCraft2Learn consortium partners identified different criteria relevant to our evaluation of available technologies:

a) General criteria

- Usability
- Current standing (e.g. under active development or abandoned)
- Cost
- Customisability / extensibility (e.g. look-and-feel, functionality enhancement via plugins)
- Licence (free and open source vs. proprietary)
- Developer and user community support (size and activity of communities)
- Ease-of-implementation in schooling environments and informal educational settings.

b) Technical criteria

- Compatibility with widely deployed educational systems (e.g. Moodle)
- Administrator rights requirements (e.g. for software installation)
- Features (functionalities)
- Platforms (OS, mobile, etc.)

c) Pedagogical criteria

- Learning curve for usability (intuitive or requires training)
- Suitability for target age group (e.g. interface specifically designed for 3-5 year olds not suitable or 13-17 year olds)
- Allows group work
- Supports sharing.

The objective of the SWOT analysis was to identify the best:

- programming languages
- programming interfaces/environments
- artificial intelligence services (cloud APIs)
- 3D-modelling tools/environments

for the eCraft2Learn platform, based on a number of factors including an understanding of different user needs in real-world scenarios. Thus, the WP4 partners analysis focused on developing an architecture that would enable learning-community collaboration, ensure integration with existing platforms, and be user friendly.

After the collection of each partner's final analysis, we were able to create a map that outlined the preferences for and/or against specific programmes. During several meetings, these results were discussed in detail, until reaching a final decision on the above elements that shall be implemented in the eCraft2Learn architecture (eCraft2Learn preliminary technologies and tools "toolkit").

The collected data were matched against activity scenarios (scenarios that show how a particular activity is done in a specific context). The scenarios were discussed in detail with the project consortium. The activity scenarios, together with the SWOT analysis, led us to create design concepts to satisfy preliminary requirements and integrate with a preliminary system architecture.

2 SCENARIOS

To understand the different (technical) needs as well as steps to take in a learning scenario with eCraft2Learn, two scenarios were assembled. They provide some initial insight into the complex interrelation between pedagogy, technology, and environment, and they foster understanding about how the technology will be embedded.

Activities in the eCraft2Learn ecosystem are developed within five stages of that consider the features of personalised and adaptive learning within flexible and open learning scenarios (elaborating on the basic concepts of 'ideate - plan - create - program - share'):

1. Ideation - Exploring the world
2. Planning a project
3. Designing and building computer-supported artefact
4. Programming the built computer-supported artefact
5. Showcasing.

These stages can be mapped onto meaningful scenarios that are dependent on the age group and learning goals of the specific context. The stages can be tied, for example, to a theater robotics project, where activities start by selecting a story or theater chapter to portray, then by designing and implementing characters and their actions, and finally culminating in a performance. Steps are not necessarily strictly followed in linear order, since when learners freely find their personalised ways through a project, some back and forth between steps is expected (e.g. when implementing a chosen design is not working out, in some cases, a

learner may discover an innovative breakthrough, while in other cases, a new design may be sought).

2.1. First Scenario: Secondary school science class - Biology lesson

Susan, a secondary school science teacher, has a class of 30 fifteen-year-olds, and she will start to teach them about DNA next week. She is eager to combine hands-on projects with pedagogical ideas she learned about in her professional development courses. For her DNA lesson, she plans to use 3D modelling, 3D printing, computer programming, and assembly instructions for electronic components and circuits. Her idea is to let the students build 3D-modelling 'wireframes' of a DNA sequence.

Susan first creates groups according to her students' diverse abilities and skills, for instance, making sure students who already have programming experience are not all concentrated into one group.

A group of students, Paul, Kelly, and Julian, started to work on their project on their 3D wireframe model of DNA. Once they were done, Susan helped them with the 3D printing of the model's pieces. The teacher's help was important, since there were a lot of errors to resolve at the beginning of the process. First, they had to download drivers for the correct 3D printer hardware. Then, once in a while, the printing process would fail, and they would have to discard the spoiled material and start the sequence again. User errors would also occur, such as miscalculations of scale between the software model and the printed result.

Awhile later, Kelly and Julian wondered what would happen if they modify the model shape. Kelly changed a parameter of one of the DNA double-helix 'rungs', and the group started to see interesting results. The teacher helped them print some additional pieces to transform their physical model.

The students then assembled the printed pieces into a full model. The 3D-printed model gave them a much different sense than the 3D computer images, because they could hold the model with their hands, rotate it directly, and compare their own model with other students' models.

Susan continued to discuss concepts from the lesson plan throughout the process, and she now felt that the students understood more about DNA than they would have from just reading a textbook and taking a test.

The following day, she had each group present their model to the rest of the class, to discuss what they learned about how molecules form the famous double-helix structure. To prepare for the presentation, the teacher asked them to 'animate' their physical models with technological enhancements. Susan used the guidebook included with the STEAM 'packet' to explain different ways they could transform their physical models with Arduino circuits. Each group selected a project and began to work on it. To design the circuits, the students were facilitated by paper template circuits, and example code which was ready to use and easy to modify.

During the presentation Paul, Kelly, and Julian's group took turns explaining why they wanted to animate their physical models using LEDs in a particular way, and teaching their classmates how they did it. They explained how to highlight the different proteins that connected the DNA strands by assigning different coloured LEDs to each. They showed that it was difficult to see that certain protein sequences were repeated. By lighting up the coloured LEDs, everyone could easily see the patterns. Protein sequences become even more evident when the LEDs associated with them flashed at the same time. They then explained to the class how they programmed the Arduino for their project, including how they solved a tricky problem: creating a flashing light sequence to highlight repetitive structures. This peer learning process continued with each group presentation.

In just one week, Susan was able to foster student interest in DNA, while students learned how to use 3D models and printing, and how to program small circuits using basic programming elements such as sequences and loops. The students also gained experience in collaborating with each other, and in using software and hardware technology to realise ideas that began in their own imagination. Susan felt empowered by these tools, which facilitated the transformation of classroom roles and activities, and ultimately helped her achieve a progressive pedagogical approach in her classroom.

2.2. SECOND SCENARIO: MR. JONES, THE SUBSTITUTE MUSIC TEACHER

A substitute teacher for music class, Mr Jones, has 10 students, who are around 16 years old. He has been asked to teach them about how music is made in a recording studio. He starts by

playing a recording for them, and explains some of the different technologies that are used to make it. However, he notices that many of the students are getting bored. He asks: “Does anyone know how to make a drum machine?” The students laugh, joking about how they can download one from an app store with their mobile phones. They are expecting to be reprimanded, but they are instead surprised by Mr Jones’ reply: ‘How about we all learn how to make a drum machine ourselves, without our mobiles?’ The students are very intrigued and do not really believe this is possible, but they are willing to give it a chance.

Mr Jones has brought with him an eCraft2Learn briefcase with 4 project kits. The kits contain electronic components and simple instructions on how use the components, with STEAM project examples. He asks the students to form groups, and loans each group a kit. The students are interested, but some are worried this task might be too difficult for them, since they do not know about technology like Arduinos. He tells them not to worry, and asks them to take out a sheet of paper. ‘Your sheet of paper will become the buttons for your drum machine!’ They think he must be joking, but they now feel very comfortable working on the project, since instead of a complicated circuit, they are focused on a piece of paper. He asks them to draw lines on the paper, dividing it up into ‘buttons’, in any arrangement they would like.

He then asks them to take out of their project kits a few different coloured wires, and a handful of small sensors. They are instructed to connect each sensor to a different coloured wire, and to connect the free end of the wire to the series of pins on the Arduino board. Finally, he asks them to tape down each wire onto the piece of paper, so that there is one sensor in each square they’ve drawn. He also invites them to connect the small speaker from their kits to the Arduino audio output connector.

The students are following along, but they seem to be losing interest. Sensing this, Mr Jones decides to take an intermediary step. There are also LEDs in the kit, and he asks them to connect the LEDs to wires, and the wires to the other set of pins on the Arduino board. He knows from a previous project that there is software pre-loaded onto the Arduinos that connects the input and output pins. He now invites the students to “play” the squares on the paper, which trigger the lights, and the students are immediately engaged, for a moment. They quickly tire of making lights flash, but they still want to know more about how the Arduino works. And they really want to make the drum machine that was promised to them!

At this point, Mr Jones tells them to plug the USB connector on their Arduinos into their classroom workstations, which have the Arduino coding environment on them. They load the software from the device onto their screens, and he explains to them what each line of code does, and what makes the LEDs light up. He then shows them how to add new lines of code that trigger a drum sound when the light is triggered. For the remaining time, the students play collaborative rhythms using drum sounds made by their paper and Arduino drum machines. Some of the students even get the idea that they can go back into the software and replace the drum sounds with sounds of their own voices. Now they are enjoying making music together, while having learned about music technology through an exploratory, hands-on approach.

2.3. SPECIFICATION FROM SCENARIOS

The eCraft2Learn technical specifications have been developed from a scenario-based design approach (Carroll, 2000). Figure 2 (below) depicts aspects of the roles, functions, and technologies derived from the first scenario (above), and generalisable to other cases. A tabular view of this information is presented in Appendix D.

To briefly summarise the contents of Figure 2 (and Appendix D), we can imagine a process by which a teacher selects a biology lesson from among different subject areas available from the eCraft2Learn unified interface. Within this lesson, there is a further project library for selecting specific projects that relate to the topic of the lesson (e.g. within the biology subject area, there are projects for the topic of DNA). The teacher can select one or more projects based on classroom resources and student skill levels. Typically, the teacher would divide learners into groups, to create a balance of skills and interest within each group, and to help facilitate resource allocation. The makeup of each group and the available resources would indicate to the teacher which project templates could be assigned.

Once a group of learners has been assigned a project template, the group can engage in the five-stage eCraft2Learn process: Ideate, Plan, Create, Program, Share. To Ideate (stage 1), learners might arrive at the idea to embed coloured LEDs in a 3D printed DNA model, to highlight biological patterns of protein arrangement (see scenario 1, above). This idea would then require a Plan (stage 2) for using the different software and hardware technologies that would result in a design for a circuit and 3D model. They could then move onto Create (stage 3), in which they would print and assemble their models and components, while also taking care to Program (stage 4), by writing the software code that will provide the LED functionality

(e.g. flashing colours). Once their model is assembled with their electronics, and their code is running, they will have completed the project from the template. Finally, they can Share (stage 5) their design, code, photos, notes, and more, with their immediate classroom peers, the wider online community, and also with the teacher, who can use the shared materials as a part of a learner assessment.

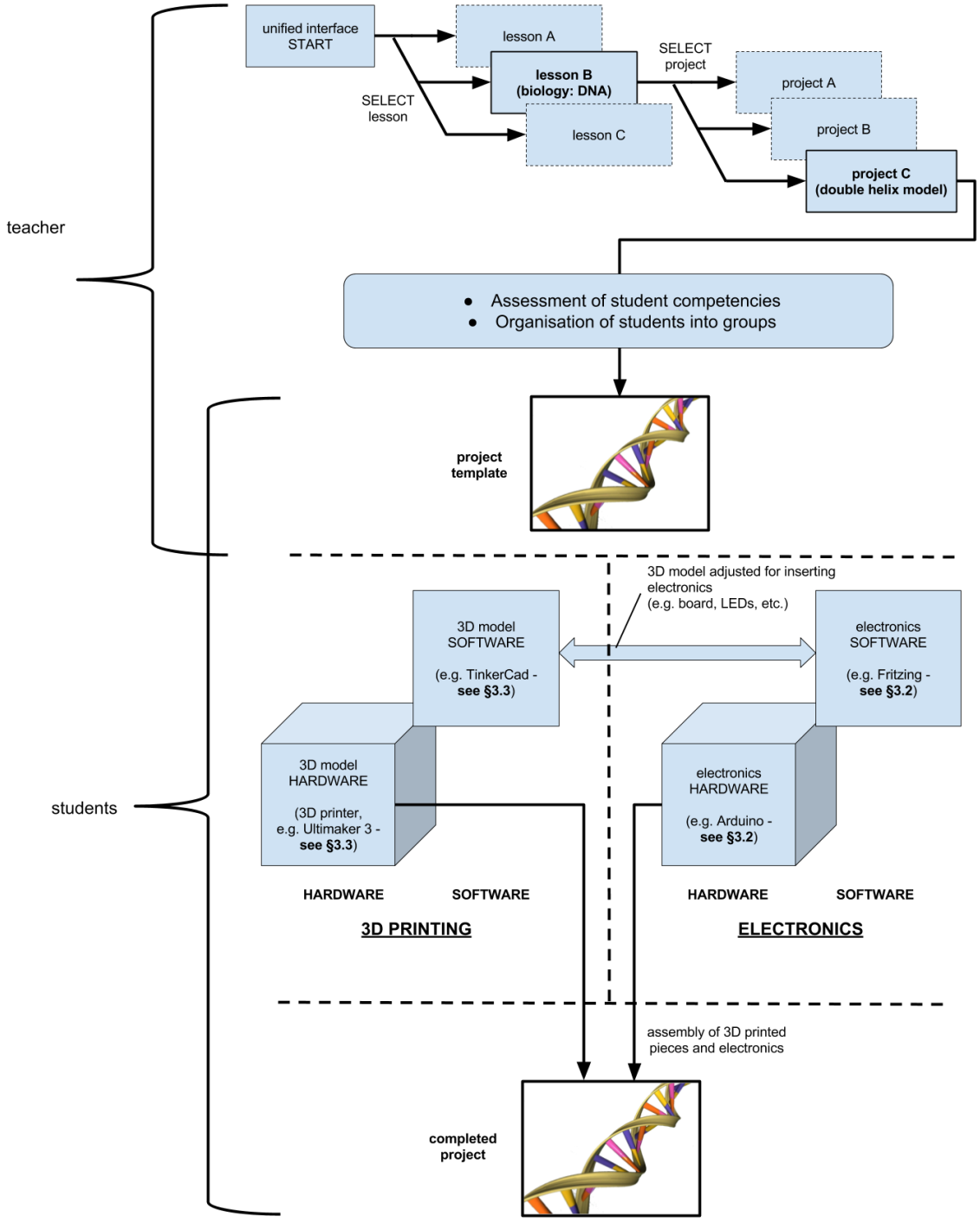


Figure 2: DNA Scenario 1: users activity flow – systems' diagram
(see Appendix D for associated table)

3 SERVICES, SYSTEMS AND PRODUCTS

eCraft2Learn will develop a learning ecosystem that will enhance craft- and project-based pedagogies through using, combining, and modifying successfully implemented technical platforms such as Arduino- and Raspberry Pi-based electronics, cloud-based 3D-printer simulators, and maker community-generated content.

In the following sections, we give an overview of systems, services, and technologies available that need to be considered for our project. We review tools for ideation, planning, and collaborative project management (3.1), programming, scripting, and markup languages (3.2), physical computing tools (3.3), 3D printing tools (3.4), and DIY technologies (3.5).

3.1. TOOLS FOR IDEATION, PLANNING, AND COLLABORATIVE PROJECT MANAGEMENT

Here, we review tools for the collaborative management for learning environments that have potential for integration with the eCraft2Learn technological ecosystem.

3.1.1. MOODLE[1]

Moodle is a VLE (virtual learning environment) for desktop and mobile that supports the activities of administrators, teachers, students, and parents. It includes standard features such as calendar and grade systems. Moodle was originally developed to help educators create online courses with a focus on interaction and collaborative construction of content, and it is in continual evolution.

The Moodle environment is based on a plugin architecture. Plugins are a flexible tool set, allowing Moodle users to extend the features of the site. There are hundreds of plugins for Moodle, extending the features of Moodle's core functionality. Each plugin is maintained in the Moodle plugins directory. Graphical themes for Moodle can be installed to change the look and functionality of a Moodle site or of an individual course.

Moodle has been translated into over 100 different languages and is accessible in many countries worldwide. Institutions can add as many Moodle servers as needed without having to pay license fees. Opensource.com (from RedHat Linux) reports that Moodle will always be an open source project. It runs without modification on Unix, Linux, FreeBSD, Windows, OS X, NetWare and any other systems that support PHP and a database, including webhost providers. Also, a Moodle mobile app is available in Google Play, App Store (iOS), and the Windows Phone Store. Users can download and install Moodle on a Web server, such as Apache HTTP Server, and a number of database management systems, such as PostgreSQL, are supported. Pre-built combinations of Moodle with a Web server and database are available for Microsoft Windows and Macintosh. Other automated installation approaches

1 The Moodle section is adapted, with significant modification, from https://en.wikipedia.org/wiki/Moodle, accessed 16-March-2017. See also https://docs.moodle.org/32/en/Main_page.

exist, such as installing a Debian package, deploying a ready-to-use TurnKey Moodle appliance, using the Bitnami installer, or using a "one-click install" service such as Installatron.

Moodle's adopted universal e-learning standards include SCORM and LTI. Sharable Content Object Reference Model (SCORM) is a collection of e-learning standards and specifications that define communications between client side content and a server side learning management system, as well as how externally authored content should be packaged in order to integrate with the LMS effectively. Learning Tools Interoperability (LTI) is a standard way of integrating rich learning applications (often remotely hosted and provided through third-party services) with educational platforms. Moodle uses the External Tool activity to act as an 'LTI consumer' as standard, and will act as an 'LTI provider' using a plugin.

Figure 3: Moodle 'weekly outline' overview with resource links

3.1.2. BLACKBOARD2

Blackboard Learn (previously the Blackboard Learning Management System), is a virtual learning environment and course management system developed by Blackboard Inc. It is Web-based server software that features course management, customizable open architecture, and scalable design that allows integration with student information systems

2 The Blackboard section includes material from https://en.wikipedia.org/wiki/Blackboard_Learn, accessed 16-March-2017.

and authentication protocols. It may be installed on local servers or hosted by Blackboard ASP Solutions. Its main purposes are to add online elements to courses traditionally delivered face-to-face and to develop completely online courses with few or no face-to-face meetings.

Blackboard is proprietary software, and has poor Linux compatibility and support. It has been identified as having a number of further limitations, including a traditional, hierarchical education system structure implicit in its design, and service provider issues with glitches, outages, and associated high costs to use and maintain. Its functionality has substantial overlap with Moodle, including communication tools such as chat and threaded discussions, and content tools, such as calendars, grading, and media libraries.

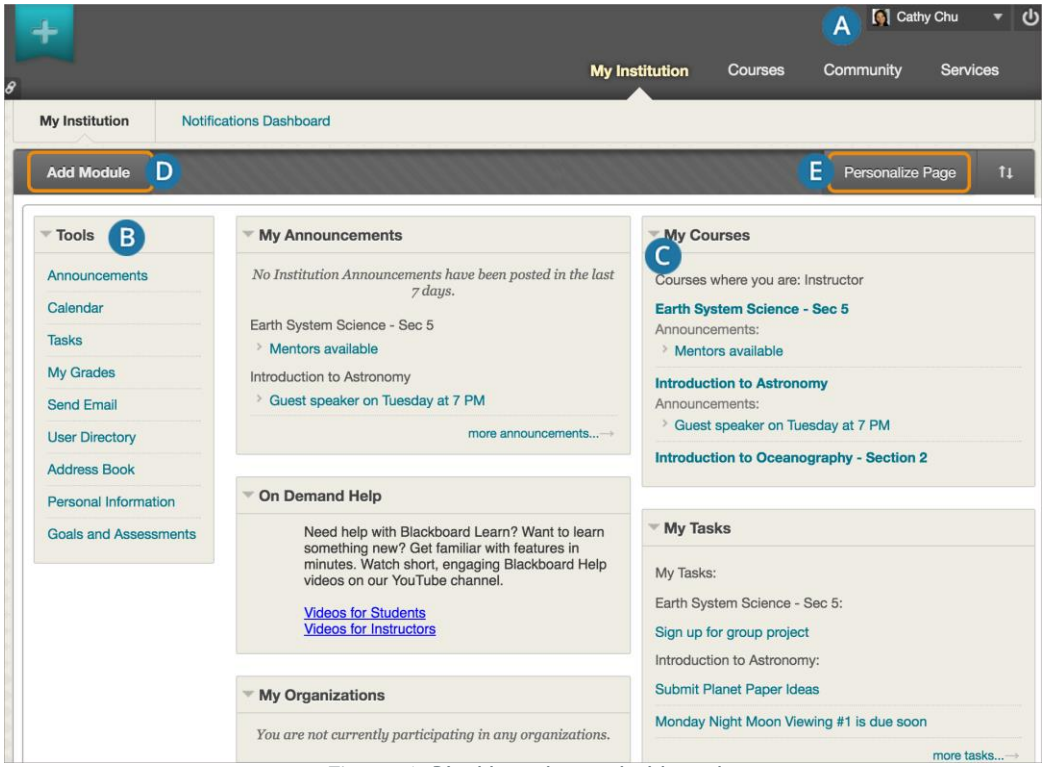


Figure 4: Blackboard user dashboard

3.1.3. ILIAS[3]

ILIAS offers a flexible, open source, SCORM-compliant environment for learning and working online with integrated tools. ILIAS goes far beyond the idea of learning being confined to courses as a lot of other LMS do. ILIAS can rather be seen as a type of library providing learning and working materials and contents at any location of the repository. This offers the possibility to run ILIAS not as a locked warehouse but as an open knowledge platform where content might be made available for non-registered users too.

ILIAS offers a lot of features to design and run online-courses, create learning content, offer assessments and exercises, run surveys and support communication and cooperation among users. These include desktop functionalities, such as note-taking, bookmark management, learning resource access, and progress monitoring. It also has multiple modes of delivering

3 The ILIAS section includes material from https://en.wikipedia.org/wiki/ILIAS, accessed 16-March-2017.

and sharing content, via categories, folders, groups, and courses. Contents can include text, images or videos to the page, which can be moved, copied or linked into other parts of the system. It also has real-time interactive chat, podcasting, and group forums, as well as full-featured administration and management capabilities, such as control over roles and permissions, course management and grading, course-wide progress tracking, announcement channels, and designing, conducting, and evaluating user assessments and exams.

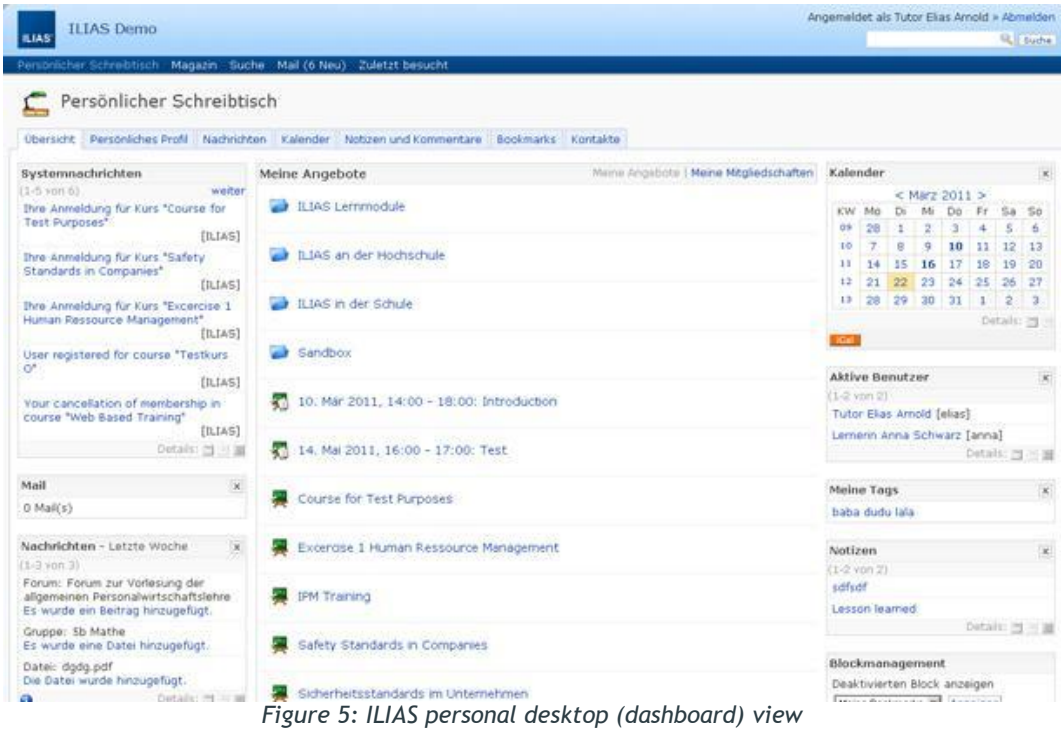
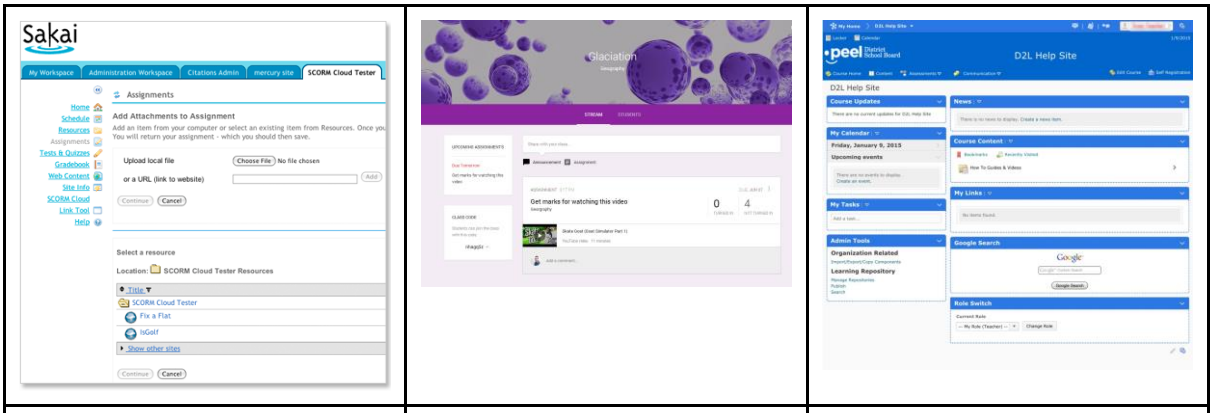


Figure 5: ILIAS personal desktop (dashboard) view

3.1.4. OTHER E-LEARNING PLATFORMS (GOOGLE CLASSROOM, BRIGHTSPACE, SAKAI, ETC.)

A number of e-learning platforms are available, including Google Classroom (free/closed), Brightspace (pay/closed), Sakai (free/open), and others. These typically offer features similar to Moodle (which most seem to be directly based on), with additional features and drawbacks. For instance, Google Classroom integrates with other Google products for email (GMail), file storage (Google Drive), etc., to provide the full set of functionality available through Moodle. While it has the support of Google, Inc., the company has been criticised for its user data gathering (not only in general, but specifically for Classroom). Brightspace is far less widely adopted than Moodle, cost-intensive, and lacks many of the customization options found in other platforms. Sakai provides a strong offering, again, substantially overlapping with Moodle's feature set, but primarily aimed at higher education institutions.



Sakai interface	Google Classroom interface	Brightspace interface

Figure 6: other e-learning platform interfaces
(l-to-r: Sakai, Google Classroom, Brightspace)

3.2. PROGRAMMING, SCRIPTING, AND MARKUP LANGUAGES

In this section we give an overview of programming, scripting, and markup languages that relate to the objectives of our project.

3.2.1. LOGO[4]

Logo is an educational programming language specific aimed at introducing children to programming and mathematics, developed by Feurzeig and Papert in the 1960s (Papert, 1980). The language was the first targeting young children (4-12) as primary users. It is known for its turtle "character", which can be a hardware robot or a software avatar that receives user commands for movement and drawing to produce physical drawings or on-screen graphics. The language uses commands like FORWARD 70 (move the turtle/cursor 70 steps) or RIGHT 50 (rotate 50 degrees).

Today, Logo has several off-shoots (e.g. NetLogo, StarLogo, etc.). NetLogo was initially aimed at high school and undergraduate education. It contains several hundred well-designed pedagogic sample models. Nowadays a significant portion of its users are researchers (typically in social and life sciences.[5] NetLogo is open source, and is the most popular agent-based modelling tool, and it continues to be actively developed. There is a web-based version and it also runs on Windows, Mac, and Linux systems.

3.2.2. PYTHON

Python is a text-based general-purpose programming language that supports multiple programming paradigms and styles (object-oriented, procedural, etc.). The language runs on multiple platforms and is open source, with a large, active community. Most of the software for the One Laptop per Child XO is written in Python. There are several cases where Python is used in primary and high schools, and it is now frequently taught in first-year computer science university courses.

4 http://el.media.mit.edu/logo-foundation/
5 http://ccl.northwestern.edu/netlogo/references.shtml

3.2.3. JAVASCRIPT

JavaScript is a scripting language commonly used in web development. JavaScript is frequently used as a client-side scripting language that provides dynamic functionality for websites. When combined with HTML5 (described below), it can be used to create standardised interactions with computer hardware (e.g. microphone, camera, hard drive, etc.). For this reason, the combination of JavaScript and HTML5 is often used to develop powerful web-based applications.

3.2.4. HTML5

HTML5 is a markup language, for formatting documents to be viewed in web browsers. It also interoperates with a number of JavaScript APIs, to provide extended functionality for content delivery (e.g. video), real-time communication (e.g. chat), offline processing, content storage, and more. It is an open standard published by the World Wide Web Consortium (W3C). It is lightweight, easy to learn and understand, and is widely supported and adopted.

3.2.5. EDEN AND CONSTRUIT![6]

EDEN is the Engine for DEfinitive Notations, developed exclusively at the University of Warwick. It is designed for constructing and exploring a wide variety of models. There are a number of implementations of the EDEN interpreter, including one in JavaScript (JS-EDEN). In contrast to the products of traditional programming, the models connect with personal experience through a different approach to software development -- a kind of 'alternative computing' that can be used for educational technology.

The CONSTRUIT! Project (based on EDEN) uses principles and tools to enable educators and learners to collaborate in creating live interactive resources that capture personal understandings of a phenomenon (which they term ‘construals’). Construals serve as personal, shareable ‘working models’ or understandings. According to the project founders, their approach is designed to be more accessible than conventional programming but more expressive and powerful than conventional uses of ICT. Their stated aim is to facilitate the online development of open educational resources that can be flexibly modified by educators and learners to support the integration of instruction and construction. This makes it well suited for potential integration into the eCraft2Learn ecosystem.

3.2.6. PROCESSING[7]

Processing has promoted software literacy, particularly within the visual arts, and visual literacy within technology. Initially created to serve as a software sketchbook and to teach programming fundamentals within a visual context, Processing has evolved into a development tool for professionals. The Processing software is free and open source, and runs on Mac, Windows, and GNU/Linux platforms.

---

6 The EDEN section uses material from https://www2.warwick.ac.uk/fac/sci/dcs/research/em/, http://construit.org/, and http://construit.org/project/.

7 The Processing section uses material from https://processing.org/overview/, accessed 17-March-2017.

Processing continues to be an alternative to proprietary software tools with restrictive and expensive licenses, making it accessible to schools and individual students. An active community of contributors develop and share programs, code, libraries, and tools to extend the possibilities of the software. The Processing community has written more than a hundred libraries to facilitate computer vision, data visualization, music composition, networking, 3D file exporting, and programming electronics.

Figure 7: Processing IDE

3.2.7. TOONTALK REBORN

ToonTalk is a programming environment that is web based, with very rich features. It uses modern web technologies such as HTML5 and Javascript, it supports interactive webpage displays and web services, including speech recognition and AI features. The user interface, however, might need to be improved if it is to be included in the eCraft2Learn ecosystem.

3.2.8. OTHER PROGRAMMING LANGUAGES, ENVIRONMENTS, ETC.

Our analysis also considered a number of other technologies for inclusion in the eCraft2Learn ecosystem. Those summarised here (and also found in the SWOT table appendices) were ruled out relatively early in the evaluation process, since they did not meet the targeted needs of the eCraft2Learn project. For this reason, it is beyond the scope of this document to provide further details of their specifications.

In brief, they can be divided into two main categories: (1) general programming languages and tools that could be coupled with educational materials (traditional “learn how to code” approach), in which we evaluated C/C++, Pascal, Ruby, Lua, and Google Go; and (2) programming languages and tools that are specifically designed or oriented toward formal educational settings or informal learning contexts, in which we evaluated Alice, Oz, Swift, Greenfoot, Kodu, and Agentsheets/Agentcube. (We also evaluated Scratch and Snap!, which are discussed in a dedicated section below.) In addition, we considered the benefits and drawbacks of proprietary languages and development environments (e.g. Macromedia Flash / ActionScript), and also general open standards such as XML (a markup rather than a programming language, but frequently used in combination with programming).

3.3. PHYSICAL COMPUTING TOOLS

Physical computing is about creating a conversation between the physical and the digital world. Physical Computing tools play a fundamental role in sustaining a meaningful, flexible, and efficient iterative making and learning process. In recent years, many prototyping platforms have become available for designers to construct interactive prototypes rapidly in a way that was unimaginable a few years ago. The Arduino platform (Mellis, Banzi, Cuartielles, & Igoe, 2007) is widely used for this purpose. It is based on free and open source principles and is low-cost (related toolkits include Phidgets (Greenberg & Fitchett, 2001) and LittleBits (Bdeir, 2009)). Most of the tools reviewed below relate to the Arduino platform.

3.3.1. ARDUINO[8]

The **Arduino** development platform consists of the open-source coding environment and the core library. Arduino microcontrollers are programmed using a dialect of the C and C++ programming languages. The specialised language uses clearly described functions like *digitalRead()* and *analogWrite()*, which seem to be more easily grasped than advanced low-level microcontroller programming. The integrated development environment (IDE) runs on many OS platforms and also has a web-based variant. Historically, Arduino relates to Processing (described above), and also enjoys its own large, active international community of contributors. It is well suited to our 13-17 user group, due its vast educational support, use in a wide range of Maker projects, and its ability to be used by all skill levels.

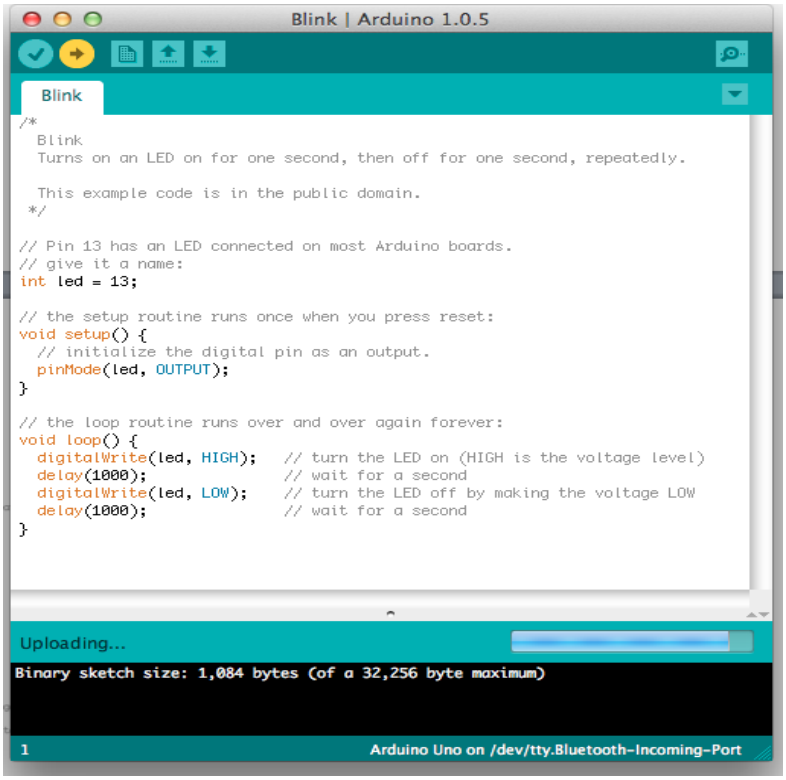


Figure 8: Code example and Arduino IDE

8 www.arduino.ac

3.3.2. ARDUINO CREATE[9]

Arduino Create enables Arduino programming directly from the web browser. It requires a login and downloadable plugin (Arduino Create Agent) installation on the local computer. The interface has easy access to a wealth of materials including libraries, code examples, and more, which are automatically pre-selected and suggested based on what Arduino board is specified as the project basis.

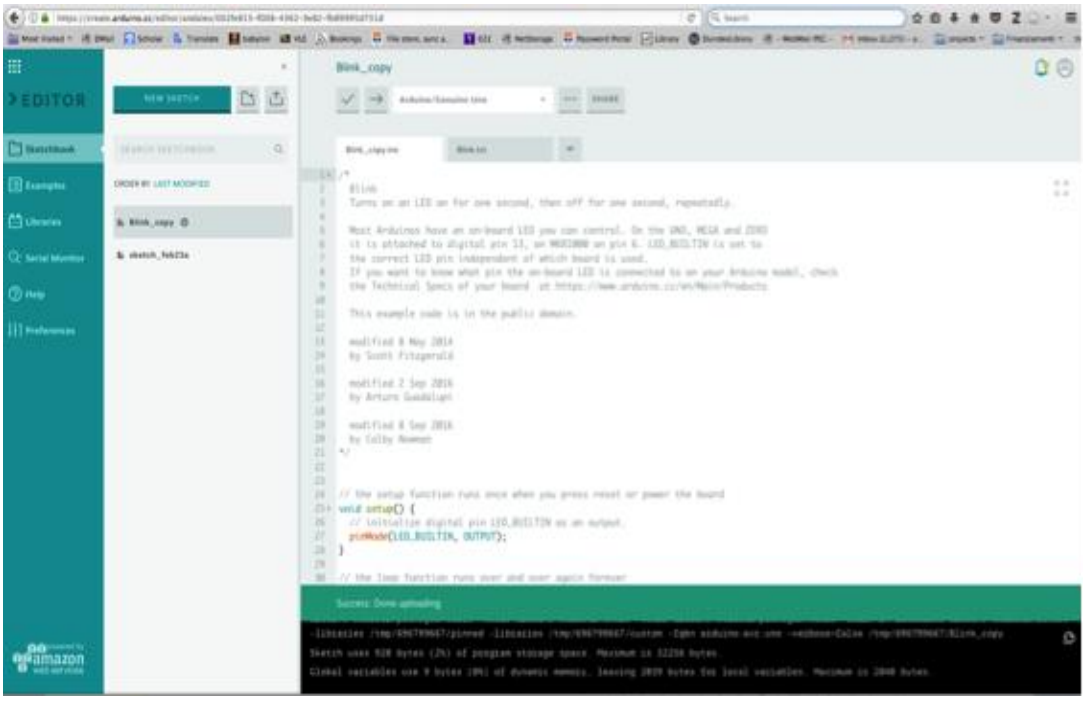


Figure 9: The online Arduino Create interface

3.3.3. SCRATCH FOR ARDUINO[10]

The Scratch for Arduino (S4A) offers a visual paradigm for programming Arduinos that is based on the Scratch environment. The environment use component "blocks" grouped by function: digital/analogue, I/O, motor control, and streaming. S4A uses a Firmata-type protocol to interface with the code and the Scratch program. The Firmata library implements the Firmata protocol for communicating with software on the host computer. A version of Scratch is installed on Raspberry Pis with similar functionality.

9 https://create.arduino.cc/
10 http://s4a.cat/

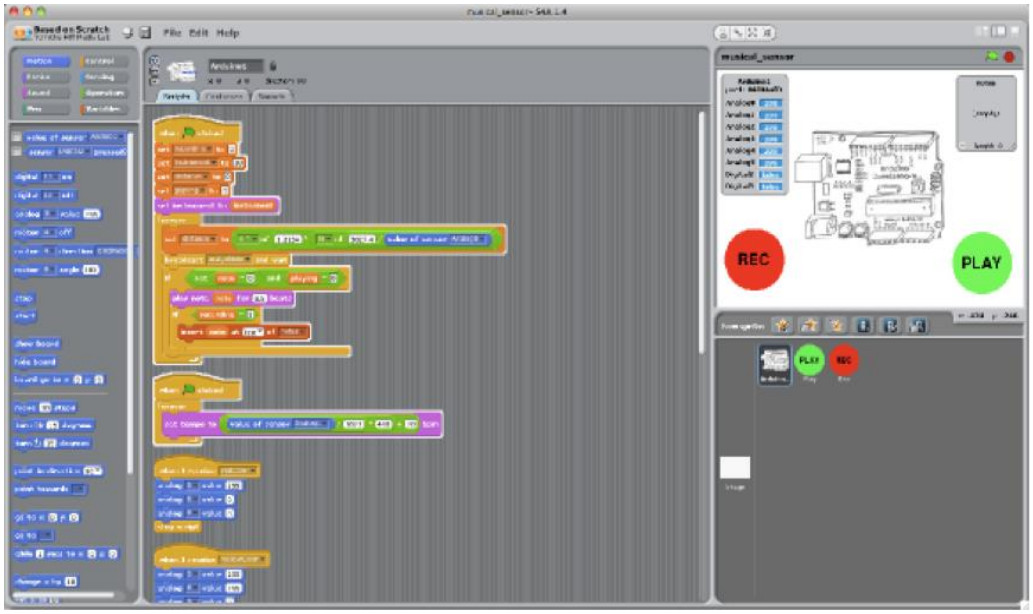


Figure 10: Scratch for Arduino

3.3.4. SNAP! FOR ARDUINO[11]

Snap4Arduino is a variation of the Snap! blocks-based programming language that lets the user program a wide range of Arduino circuit boards. (S4A uses standard Firmata firmware, which needs to have been loaded onto the Arduino board. It also supports the Linino library.) A web-based version is currently in the works. As with the desktop version, you need to have Firmata loaded into your Arduino board. It uses the Google Chrome browser with a special browser extension (Snap4Arduino.crx). (Beyond Arduino, NodeSnap[12] extends Snap! to run on Raspberry Pis.)

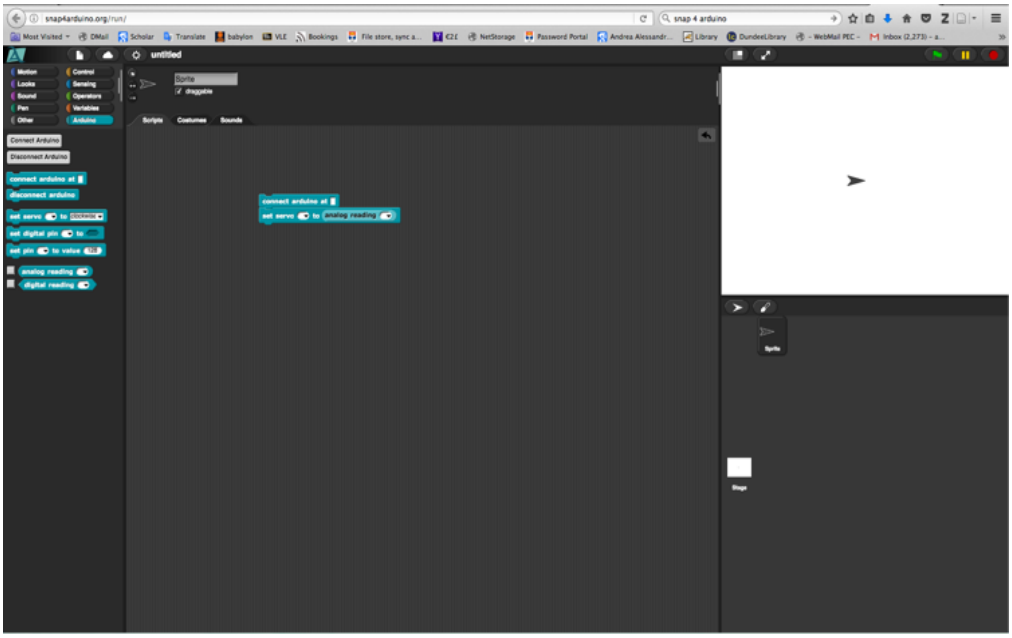


Figure 11: Snap4Arduino

11 http://snap4arduino.org/index.html
12 https://github.com/rasplay/nodesnap

3.3.5. ARDUBLOCK[13]

ArduBlock is a visual programming language for Arduino. ArduBlock uses a drag-and-drop programming paradigm. The ArduBlock converts the visual blocks into textual code via the Arduino IDE. In this way, programming syntax is hidden by visual graphics, with in-built affordances for connecting smaller units (similar to Scratch). ArduBlock is open source and works on Windows, Mac, and Linux platforms. It is not a standalone program, as it requires the Arduino IDE.

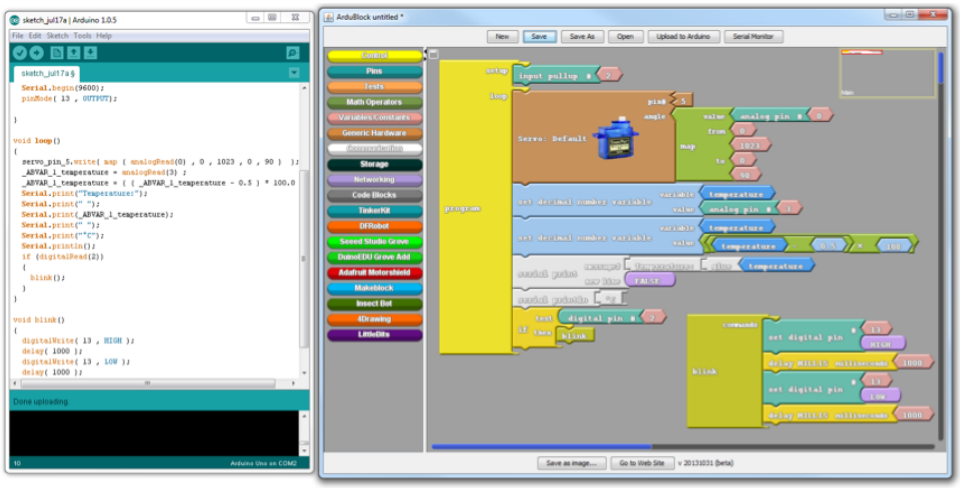


Figure 12: ArduBlock

3.3.6. MINIBLOQ[14]

Minibloq is similar to ArduBlock, a graphical programming environment. In contrast to ArduBlock, however, Minibloq is a standalone program (it does not require the Arduino IDE). Minibloq generates Arduino-ready code instantaneously, as its graphical blocks are dragged into a special interface pane.

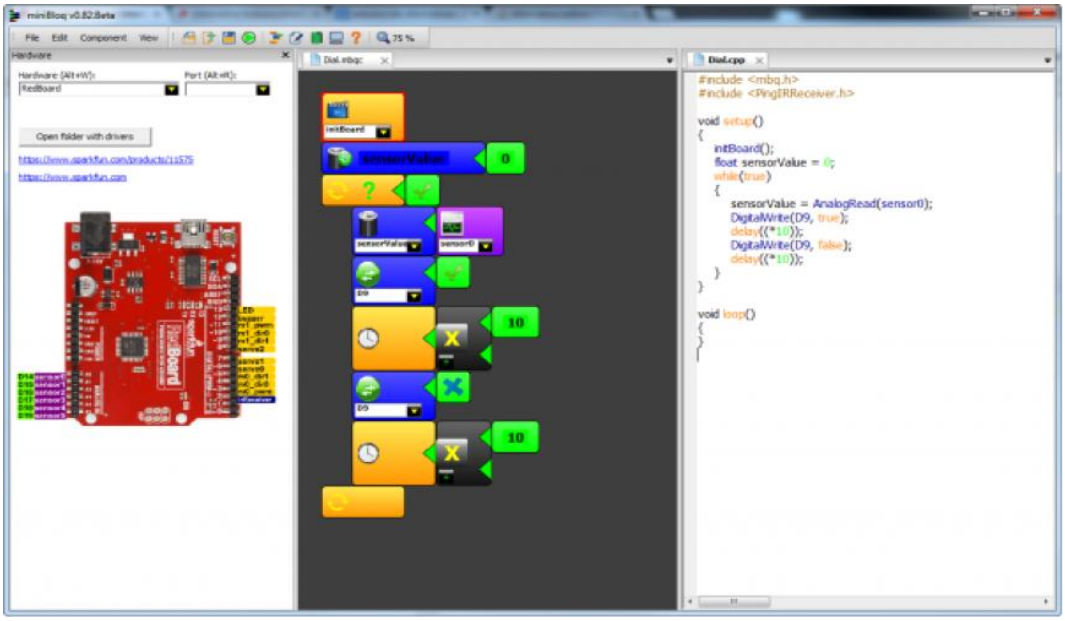


Figure 13: Minibloq

13 http://blog.ardublock.com/
14 http://blog.minibloq.org/

3.3.7. MODKIT[15]

The main Modkit environment is Modkit Micro (accessible via Modkit.io), a successful visual programming browser-based implementation. At the moment the environment supports the main Arduino boards like the Uno, LilyPad, etc. The interface can easily switch between blocks (graphical) or source code (textual) programming styles. Currently, it lacks strong community support.

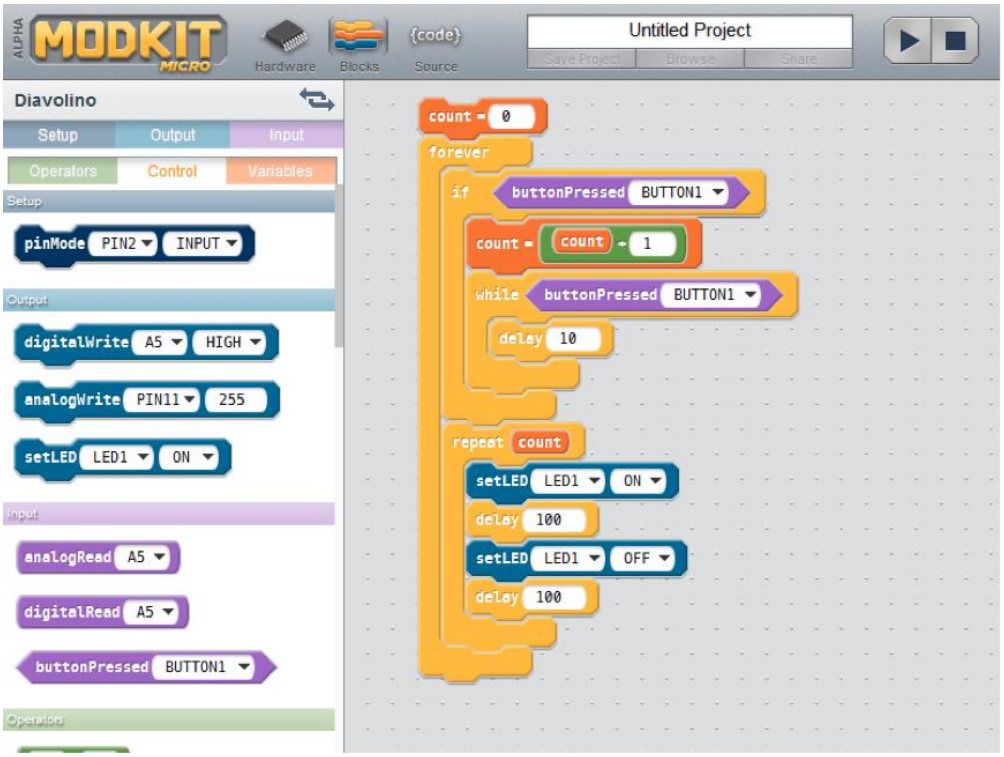


Figure 14: Modkit

3.3.8. BLOCKLY FOR ARDUINO[16]

Blockly is a HTML5-based block interface for Arduino programming. Blocks generate textual code in real-time, including Arduino code and XML.

15 http://www.modkit.com

16 https://ardublockly.embeddedlog.com/demo/index.html
http://blokkencode.ingegno.be/index_en.html

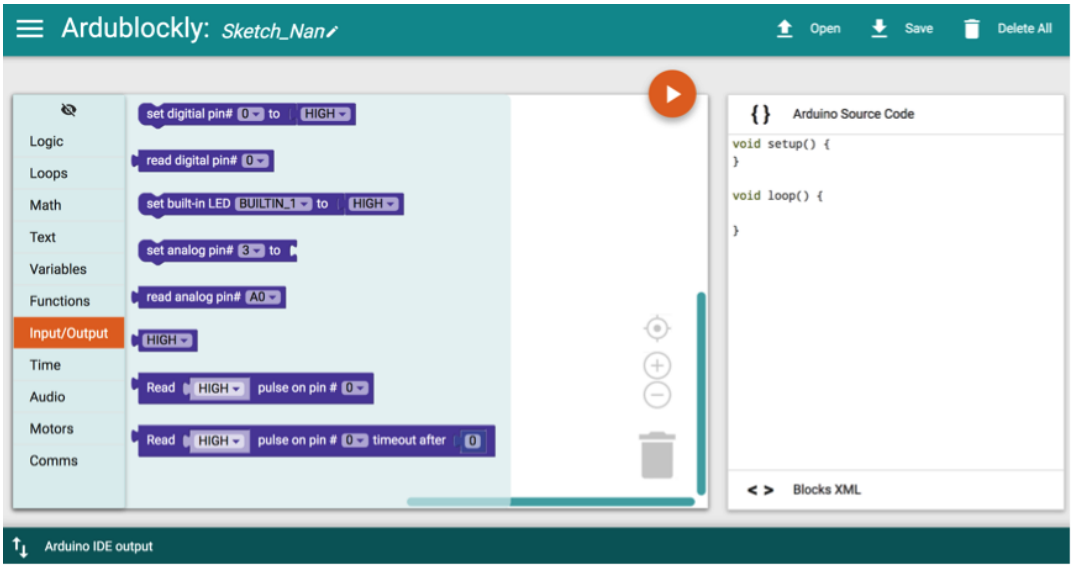


Figure 15: Blockly

3.3.9. BITBLOQ[17]

Bitbloq is a visual electronics and block-based programming interface that toggles between visual and textual modes.

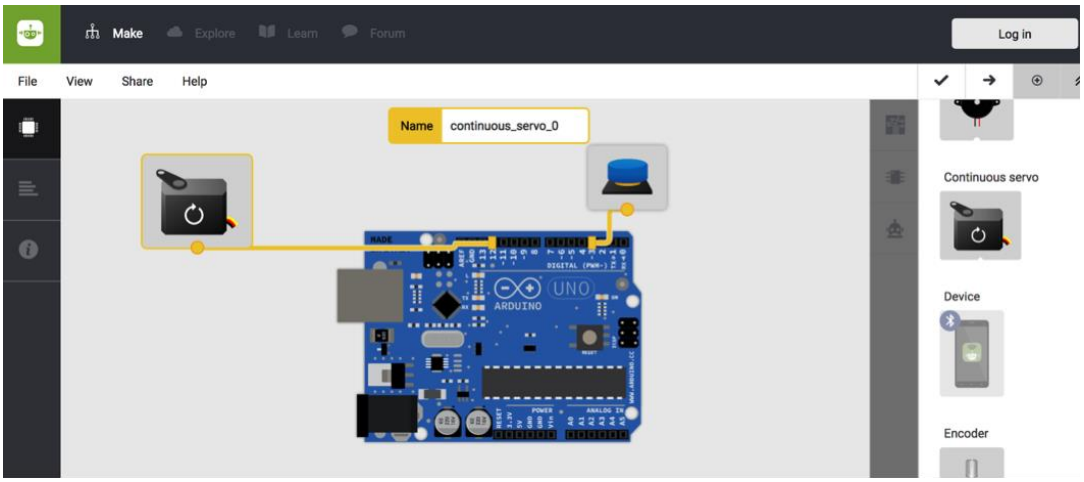


Figure 16: Bitbloq

3.3.10. FRITZING[18]

Fritzing is an open-source hardware initiative that makes electronics accessible to a wide audience. It offers a software tool, a community website, and additional services such as custom circuit board (PCB) printing. As with Processing and Arduino, the Fritzing community fosters a creative ecosystem that allows users to document their prototypes, share them with others, teach electronics in a classroom, and layout and manufacture professional PCBs.

17 http://bitbloq.bq.com/#/
18 http://fritzing.org/home/

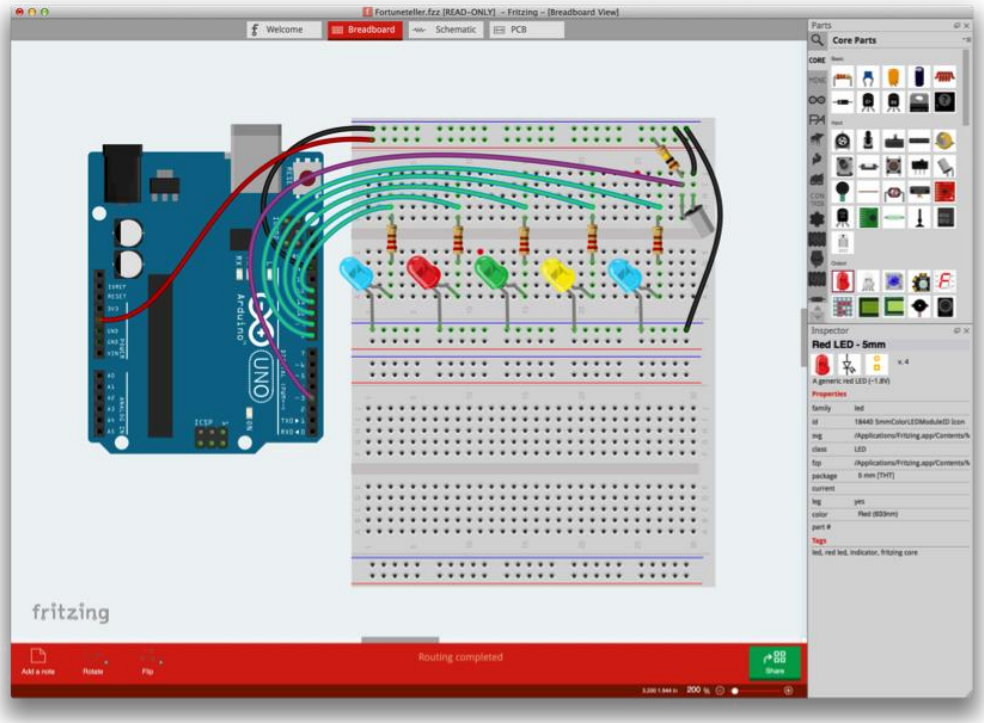


Figure 17: Fritzing

3.3.11. INTERFACING WITH ARDUINO HARDWARE (TRANSFERRING SOFTWARE TO HARDWARE)

Some of the above tools are for designing Arduino circuits, while others are for developing code to run on the Arduino hardware. Code that runs on Arduino hardware is called a 'sketch' (in their terminology) and a sketch must be transferred from the development machine (e.g. a notebook or workstation computer) to the Arduino hardware. Transferring sketches from a host to an Arduino was previously a rather cumbersome process (legacy approach). However, two new standard interfaces for writing sketches to Arduinos are now available for wired and wireless internet connected devices: Arduino Ethernet Shield 2 [19] (wired) and Arduino WiFi Shield [20] (wireless). These technologies (hardware modules that connect to Arduino circuit boards) facilitate a simple process of loading sketches (developed with the above referenced software) onto Arduino hardware via a local internet (or intranet) connection.

Other approaches to transferring sketches onto Arduino hardware include the use of specialised tools, for instance, browser-based tools that are commonly tied to specific web browsers (e.g. Google Chrome, Mozilla Firefox), some of which require administration privileges on the local machine where the sketch is developed. Variants include serial port connections via a Google Chrome app API, a Chrome-only Bluetooth API, and the jUART extension for Firefox. Another browser-based approach is based on the Firmata protocol,

19 www.arduino.cc/en/Main/ArduinoEthernetShield
20 www.arduino.cc/en/Main/ArduinoWiFiShield

which has some cross-browser, cross-platform support and also interfaces with Snap4Arduino and Scratch4Arduino. Another option we considered is Wyliodrin. It strengths relies on its web based platform that is also compatible with several embedded hardware systems such as Arduino and the Raspberry Pi. Another important opportunity/feature of this programming environment is that devices could be driven wirelessly.

3.3.12. CONNECTING PROJECTS WITH AI CLOUD SERVICES

It is possible for learners engaged with maker technologies to develop projects that connect with established AI technologies for providing, e.g. speech recognition services or machine vision recognition capabilities. The main providers of AI interfaces compatible with this scenario include Google AI Cloud, IBM Watson Services (vision, speech, and data API), Amazon Alexa Voice Service API, and Microsoft Cognitive Services. Some of the software described above, such as Snap!, can interface with these AI Cloud services so that learners can develop simple projects that provide advanced speech and vision recognition technology. Generally, these AI Cloud services are provided at no cost for low volume usage (typical of maker projects), however, free registration is required in order to use the services (registration provides an alphanumeric 'key' that must be included in the maker software that issues the request to the cloud service). Alternatively, if a single key were used for the entirety of the eCraft2Learn ecosystem including all deployments, no further individual registrations would be required. However, the remote request volume would likely exceed the free usage quota and there would be an associated service fee charged by the provider.

3.4. 3D PRINTING

3D printing is a technology that can output 3D objects by selectively adding or removing material under control of a computer. The objects are normally designed using 3D-modelling software and/or obtained from sources such as 3D scanners. In order to "print" the 3D object, the computer slices the 3D software model into thin layers and sends the information to the 3D printer. The 3D printer in turn adds (or removes) material, layer by layer, in order to produce the final shape of the object.

A range of different types of 3D printers are currently available, from industrial versions that can print metal objects to smaller, consumer-grade desktop versions that normally print plastic objects using resin as source material. Since the goal of this project is to use 3D printing in school environments, in this document, we will only focus on consumer-grade desktop 3D printers. Also, a more detailed description and overview of 3D printing technologies will be later presented in D4.2, in which we will focus on the process of 3D printing and 3D modelling.

3.4.1. 3D PRINTING WORKFLOW

The first step in printing a 3D model is to create one. It is also possible to create a 3D model using 3D scanners or downloading 3D virtual objects from different sources (e.g. 3D Warehouse). A variety of 3D modelling/design programs are available today. Some are explicitly designed for industrial engineering needs and some are designed primarily for use by hobbyists and students.

In order to 3D print a model, the software model needs to be converted to a specific file format that the printer can process. The most commonly used format for 3D printers is the STL file format. The process of generating an STL file from the 3D model includes a "repairing" step where the model is checked for errors and printability. It should be noted that not all 3D software models can be printed by all printers, especially due to structural constraints of the desired object, and limitations of the printing apparatus.

After repairing the virtual 3D model, a slicer then decomposes it into thin layers that the printer can handle. This information is included in a specific language (called G-Code) which tells the printer how to print the 3D object, for example, telling the printer to move the output nozzle to a specific location, activate the material injection, and so on.

The 3D printing process itself can range from several minutes to several hours or even days, depending on the size of the object and the material being used. The final 3D printed object normally needs to go through a final finishing step, for removing support scaffolds, sanding, colouring, etc.

3.4.2. 3D MODELLING SOFTWARE

The aforementioned 3D printing process contains key components for the eCraft2Learn project: 3D modelling software and the 3D printer. Deliverable 4.2 will focus more on the 3D printer hardware, so in this section, we focus only on the 3D modelling software. The aim is to find a 3D modelling software that is:

- easy to use (moderate learning curve)
- 3D printer friendly (can export STL files or communicate directly with 3D printers)
- free and open source.

In the following, a list of possible 3D modelling software that can be used in this project is presented.

3.4.2.1. TINKERCAD[21]

TinkerCad is a member of a family of software from Autodesk called 123D, which are all freely available to users. The family includes many applications such as 123D Design, 123D Make, 123D Sculpt, etc. Unfortunately most of the members of this family will soon be discontinued or shut down. Autodesk aims to replace most of these with a single application called Fusion 360 (we will discuss Fusion 360 separately). One of the few parts of 123D that will not be shut down is TinkerCad.

TinkerCad is a Web-based, solid modelling tool aimed for anyone new to 3D modelling. It has a simple and easy to use interface. The user designs 3D objects using basic shapes (cube, cylinder, etc.) to create 3D objects. It has a multi-language interface with support for 15 languages at the time of writing (Finnish and Greek are not currently supported). The users can share their designs through Thingiverse, a community for 3D printable resources. There are also a lot of online resources and videos on how to use TinkerCad for different design purposes.

21 www.tinkercad.com

On a more advanced level, TinkerCad allows the users to employ Autodesk's Shape Script API to define shapes programmatically. These shapes can be added to the interface as basic building blocks of more complex shapes or used directly.

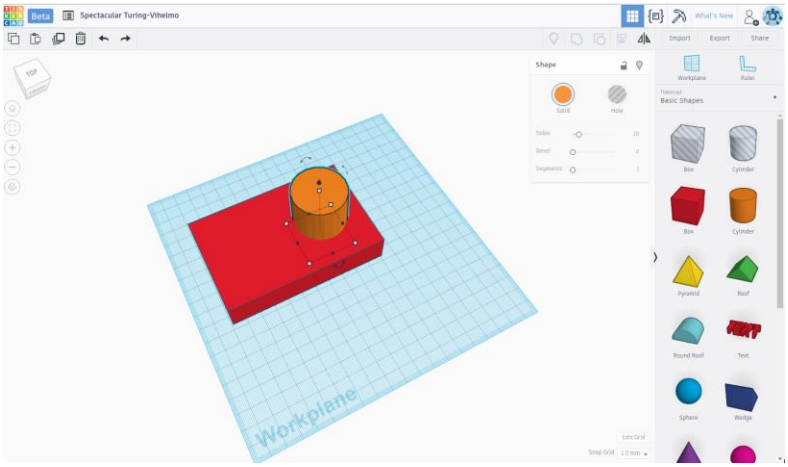


Figure 18: A screenshot of TinkerCad. On the right panel the basic shapes are presented that can be used as basic building blocks for other shapes. Each shape can also be defined as a "hole", which basically means it will be subtracted from the original shape. For example in the above design, it is possible to define the orange cylinder as a "hole", which in turn creates a cylindrical hole in the red cube. Please also notice the icon on the top right that directly connects TinkerCad with Thingiverse.

3.4.2.2. BLOCKSCAD[22]

BlocksCad is a Web-based 3D modelling tool that uses code blocks to define and render a 3D shape. This can be a very useful tool to learn 3D concepts such as translation, rotation, etc. It uses the same code block logic as other code block languages (Scratch!, Snap, etc.). This approach helps the learner to focus on 3D modelling concepts, while using a familiar block-based programming paradigm. Using BlocksCad, the learner can shape basic 3D elements and adjust them to specific project needs. BlocksCad can also generate .STL files, for sharing with other 3D platforms and/or outputting to 3D printers.

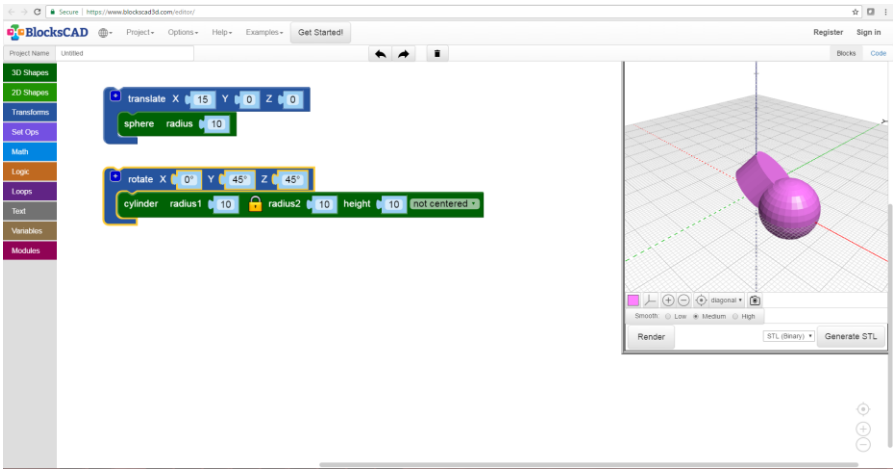


Figure 19: A screenshot of BlockCAD. The 3D model on the right is rendered from the block code developed on the left side.

22 www.blockscad3d.com

3.4.2.3. OpenSCAD[23]

OpenSCAD uses a 3D programming approach in which the user writes code that will be rendered to a 3D model. In contrast to BlocksCAD, OpenSCAD facilitates the development of more complex shapes. However, achieving such complexity requires more programming knowledge. OpenSCAD is an open source application for Linux, Windows and Mac OS platforms. It is possible to use external 3D editors for OpenSCAD, and OpenSCAD may also be used in command-line mode. OpenSCAD can also export and import .STL files, which is useful for 3D printing and sharing purposes.

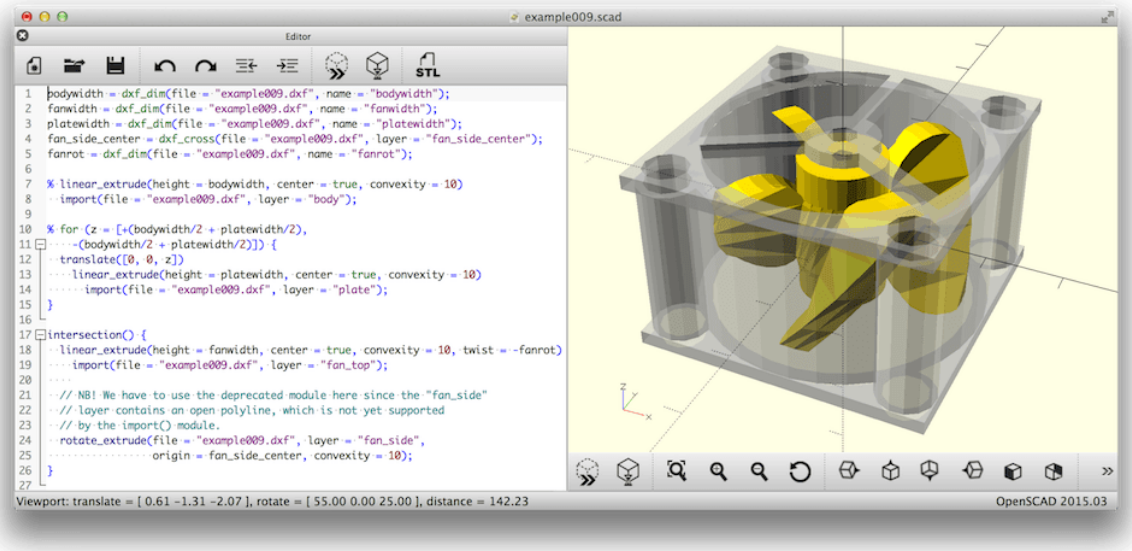


Figure 20: A screenshot of OpenSCAD. The 3D model on the right is rendered from the code on the left.

3.4.2.4. Fusion 360[24]

Fusion3D is Autodesk's new 3D modelling platform, to replace the 123D family of products. It offers free access to teachers, learners, hobbyists, and startups. Fusion 360 is a desktop application that can be used for CAD, CAm and CAE on Linux, Windows and Mac. The interface is designed for professional users, but is relatively easy to learn. Autodesk provides a lot of instructional material for Fusion 360, and related third-party online resources are also available. Fusion 360 also has built-in functions for directly interacting with 3D printers.

23 www.openscad.org
24 http://pixologic.com/sculptris

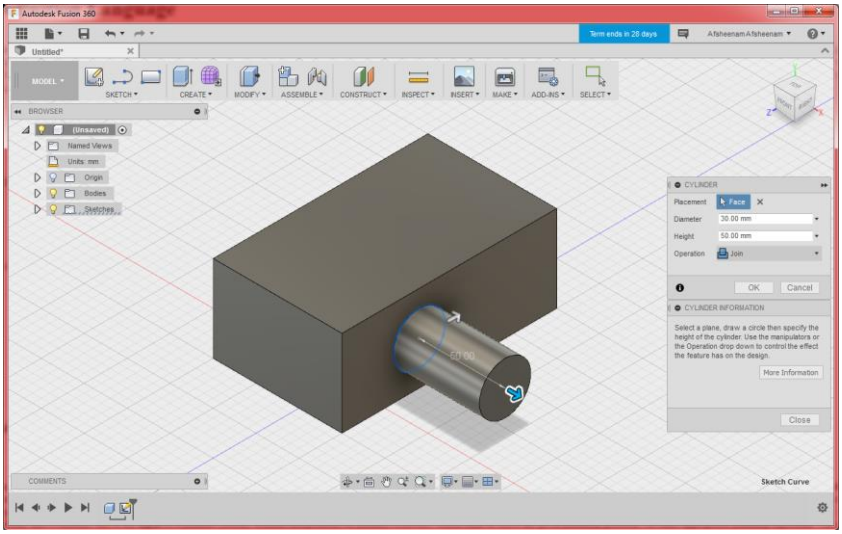


Figure 21: Fusion 360

3.4.2.5. SKETCHUP[25]

SketchUp is a 3D modelling application for Windows and Mac OS. It comes in two versions: SketchUp Make and SketchUp Pro. The Make version is aimed at high school students and hobbyists and is free to use. SketchUp uses 3dwarehouse as its 3D model sharing platform, a service which makes possible the repair and export of 3D models as .STL files for printing and/or sharing.

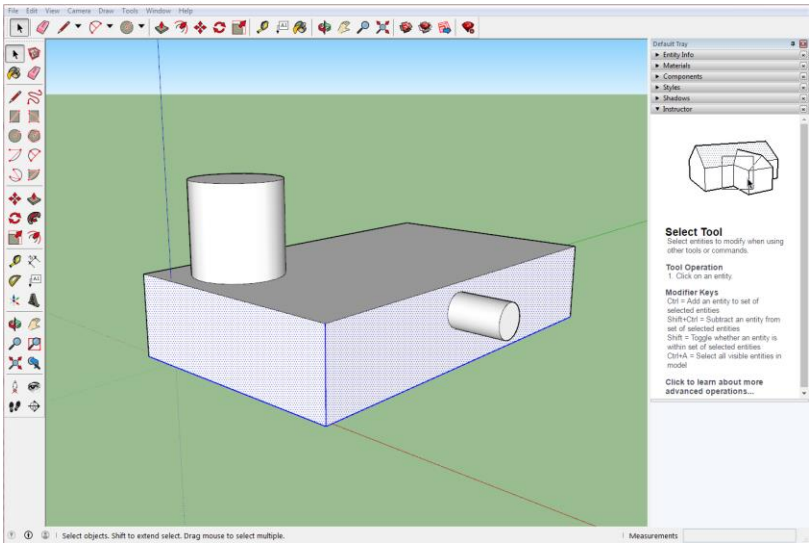


Figure 22: SketchUP

3.4.2.6. SCULPTRIS[26]

In contrast to most 3D modelling software focused on engineering and production, Sculptris allows the user to design objects by forming virtual clay into the final desired form. This approach might be

25 www.sketchup.com
26 http://pixologic.com/sculptris

of special interest to art students and teachers. Support is offered to learners through the community of Sculptris users, and other online sources and tutorials are also available.

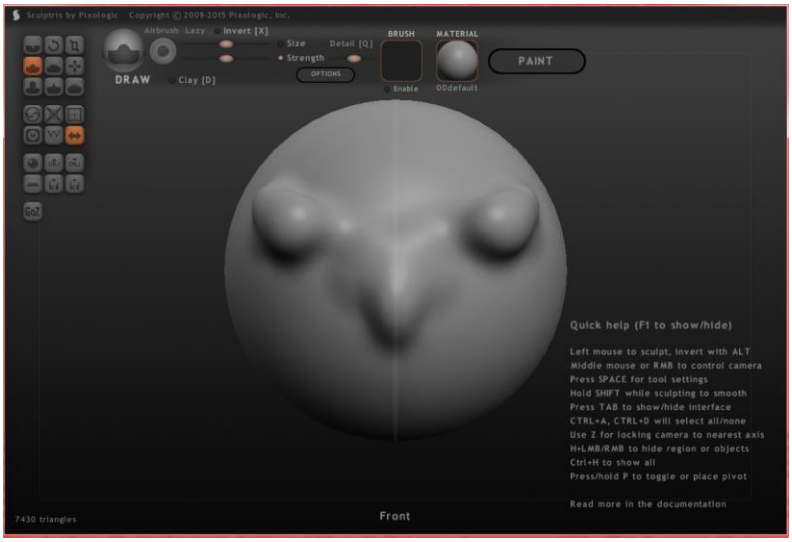


Figure 23: Sculptris

3.4.2.7. FREECAD[27]

FreeCAD is a free and open source parametric 3D modelling software. Users and developers can also develop Python scripts for FreeCAD to achieve different goals, e.g. to build customised models, extend FreeCAD functionality, or embed it in another application. FreeCAD can also be used in command-line mode. FreeCAD has a strong online community that offers support to learners.

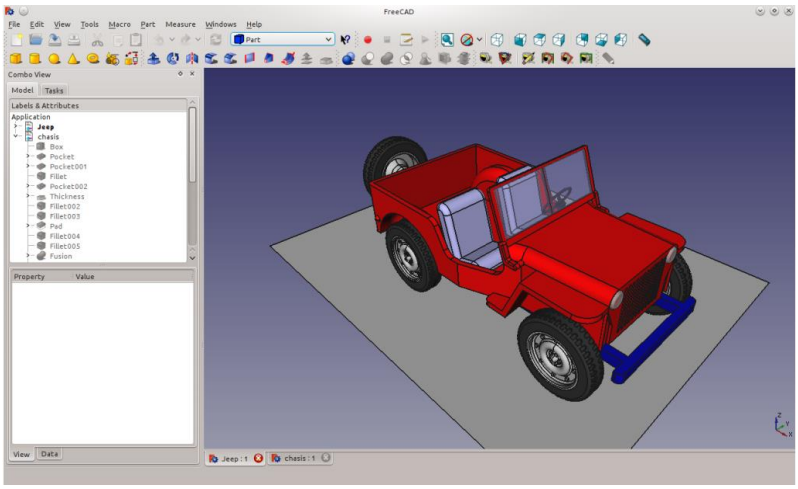


Figure 24: FreeCAD

3.5. DIY TECHNOLOGIES

The acronym DIY is short for Do-It-Yourself, which in a broad cultural sense, refers to the ability of lay persons (rather than specialised professionals or experts) to build, modify, or repair various entities such as electronic and mechanical devices. In our context, the DIY principle is specifically related to

27 www.freecadweb.org

the setup of electronics-based systems using household or hobbyist hardware. Such projects can be further extended with 3D printing (described above), which allows individuals to design and print physical components to augment household elements and commercial hobbyist kits.

eCraft2Learn is strongly inspired by the Constructionist learning by making philosophy and method, which is based on using easily accessible technologies. In contrast to hardware which is expensive, complex, or provided as a black box, the goal to educate future generations of empowered and aware citizens implies a significant change in how to learn, by means of motivating and rewarding practical experiences(Fourie & Meyer, 2015). Currently available DIY electronics, with its associated concepts of 'open source', 'open innovation' and 'peer learning', reduces the barrier to 'learning by making'.

This section provides a brief introduction to the main DIY technologies we plan to adopt in the eCraft2Learn project. Arduino and Raspberry PI are of major interest, while the other approaches listed here could be easily integrated into the ecosystem we are developing. Combining a DIY approach with robotics projects offers a strong emotional impact to learners, as they explore the psychological and sociological aspects of the human-robot interactions.

3.5.1. Arduino[28]

Arduino is a platform comprised of open-source hardware and software to promote the design and realization of low-cost digital devices able to interact with the surrounding environment through sensors and actuators (the software IDE is described in detail above). The success of this initiative, started in Italy in 2003, is due to the effort to standardise crucial aspects such as pin layout, board resources, connectivity and expandability. The hardware reference designs are provided under a Creative Commons Attribution Share-Alike 2.5 license, and the source code for the IDE under the GNU General Public License v. 2.

Several producers of Arduino boards are presently on the market, together with producers of Arduino-compatible products (these usually avoid directly using the Arduino name). Arduino boards (at the time of writing, 17 official types with additional format variations) use various microprocessors and controllers, although the software tools described above make it relatively easy to program standard applications for these boards. The layout of pins permits interfacing basic boards with expansion boards (shields), typically stacked in a vertical arrangement to remain compact. Several producers sell starter kits that include one or more Arduino boards, sensors, motors, LEDs, resistors, wires, and other components to help learners tinker with these components and to build meaningful devices.

28 www.arduino.cc

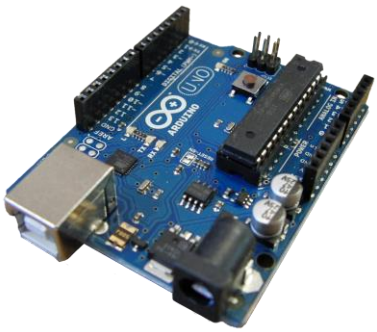


Figure 25: An Arduino board

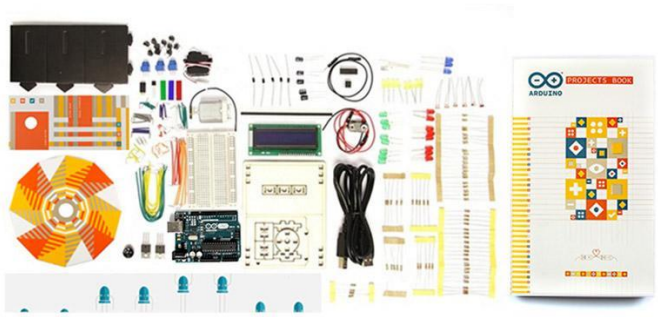


Figure 26: An Arduino starter kit

3.5.2. ARDUINO-BASED ROBOTS

Some Arduino kits include components for building sensor-based mini-robots: a rigid structure, controllable motors with or without encoders, sensors suitable to realise usual robotic behaviors such as line following, obstacle avoidance, etc. We include here also those kits that use Arduino-compatible hardware (see for example mBot www.makeblock.cc/mbot/). These kits offer the same tinkering possibilities of other simpler starter kits but they have a higher ceiling, in that the learner can eventually create autonomous robots that interact with the physical environment through sensors. As with other Arduino projects, the robots do not need to maintain a permanent connection with the PC used to program them. Other programming environments provide similar features but using a block-based language (e.g. mBlock). Some examples of Arduino robotic kits are listed here:

- Elegoo (www.elegoo.com)
- BYOR (http://byor.scuoladirobotica.it/en/homebyor.html)
- Arduino Robot (https://www.arduino.cc/en/main/Robot)
- Elecfreaks Robot Starter Kit (http://www.elecfreaks.com/estore/freakscar-robot-starter-kit-741.html)
- Multiplo Robot Kit (https://github.com/multiplo)
- Parallax Robotic shield kit (https://www.parallax.com/product/130-35000).

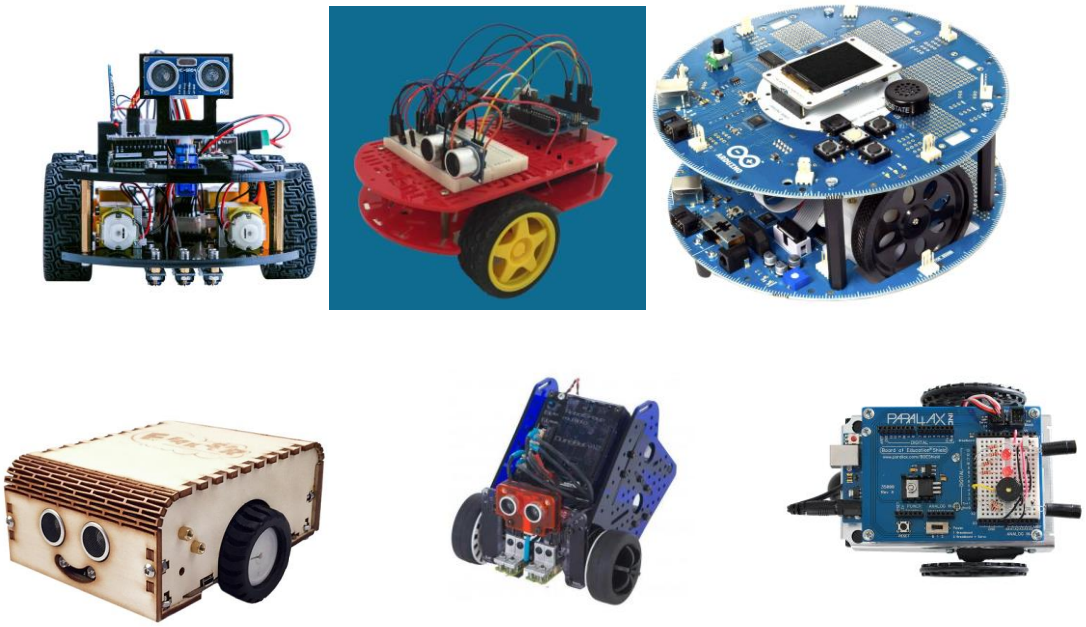


Figure 27: Arduino-based robots:
(l-to-r Elegoo, BYOR, Arduino Robot, Elecfreaks, Robot Starter kit, Multiplo, Parallax Robotic shield kit)

3.5.3. RASPBERRY PI[29]

The Raspberry Pi (RPi) is a small single-board computer produced by the (UK) Raspberry Pi Foundation. One of the aims of the foundation is to offer a low-cost entry point into computing, which serves to promote the teaching and learning of basic computer science to students and individuals, including in developing countries. It has also become a platform for the Maker movement due to its low cost, as well as its native free and open-source Linux OS.

Raspberry Pi's have an additional benefit, beyond their ability to be used in specific Maker projects. In particular, because they are fully functioning computers with desktop capabilities, they can be used as a host to run most of the software described in this document, including but not limited to web-based applications. For example, an Arduino can be programmed by running the Arduino IDE on an RPi system that interfaces with the Arduino.

Given the relative low cost of the RPi, it is promising for inclusion into the eCraft2Learn ecosystem, and may be used in place of a traditional desktop computer. This can eliminate the need for computers in schools that may be lacking in number, or that may require special administrative oversight. Since the standard RPi OS is a GNU/Linux distribution (called "Raspbian"), it can also facilitate computer literacy through the GNU principles of transparent computing aimed at learners and makers.

The current main product line includes the Raspberry Pi 3 Model B that is bundled with on-board WiFi, Bluetooth and USB boot capabilities. These features are also present in the smaller and cheaper Raspberry Pi Zero board. Peripherals are not provided together with the basic board but several bundles include accessories for building a complete system with the power of a small PC. All models feature a Broadcom System-On-Chip (SOC) that integrates an ARM-compliant CPU and a GPU. Most

29 www.raspberrypi.org

boards have USB ports, HDMI and composite video output, a phone jack and they include also GPIO pins supporting general protocols like I2C. The B-models have an Ethernet port, and some also have 802.11n WiFi and Bluetooth.

The OS and applications are stored on an SD card in either the SDHC or MicroSDHC sizes. Its native OS is Raspbian, a Debian-based Linux distribution, but other possibilities are also available (Ubuntu, Windows 10 IoT Core, Risc OS and others). RPi supports Python and Scratch, along with most any software that runs on Linux. The current RPi 3 integrates a Broadcom BCM2837 SoC, 1.2 GHz 64-bit quad-core, ARM Cortex-A53 architecture, 512 KB shared L2 cache, 1 GB of RAM. The video controller can emit standard modern TV resolutions (HD and Full HD), together with higher or lower monitor resolutions. RPi boards also support the HAT (Hardware Attached on Top) principle for expansion boards.

Figure 28: Raspberry Pi 3 board

Figure 29: Shield stacked on a Raspberry board (HAT)

3.5.4. RASPBERRY PI-BASED ROBOTS

As with the Arduino, there are also robotic kits based on the Raspberry Pi board. The relative capabilities of the electronics (significantly more than Arduinos) and the flexibility of a full desktop-level software environment (also more powerful than Arduinos) can facilitate the development of much more elaborate and intensive applications. Examples of kits:

- GoPiGo (https://www.dexterindustries.com/gopigo/)
- BrickPi Starter bundle (https://www.dexterindustries.com/brickpi/)
- CoderBot (http://www.coderbot.org/en/index.html).

3.5.5. WEMOS30

Apart from Arduinos and RPis, other platforms are also on the market. Wemos is one of the series of boards based on the ESP-8266 chip, a low-cost, Wi-Fi powered component with a full TCP-IP stack microcontroller capabilities, particularly suitable for building Internet-of-Things (IoT) structures. Some interesting features of this architecture are: 32-bit RISC CPU, 64 KB instruction, 96 KB data, 16 GPIO pins, one 10-bit ADC, integrated UART. The ESP-8266EX chip equips, for example, the Wemos D1 mini

30 www.wemos.cc

Pro board, which includes a 16 MB flash memory, a micro-USB socket and a ceramic antenna. One relevant aspect is the possibility to program the board using the standard Arduino IDE. Wemos provides also a Raspberry Pi reference card to combine the two cards' features through the HAT principle. Wemos also produces some extension shields (e.g. a motor shield, depicted below).

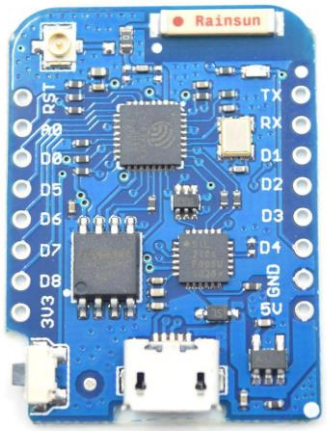
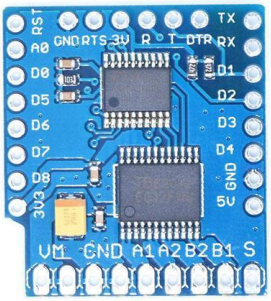


Figure 30: Wemos D1 mini Pro



VM: Motor power supply +
GND: Motor power supply -
A1, A2: Motor A
B2, B1: Motor B
S: Standby control port(in IO mode)

Figure 31: A Wemos shield

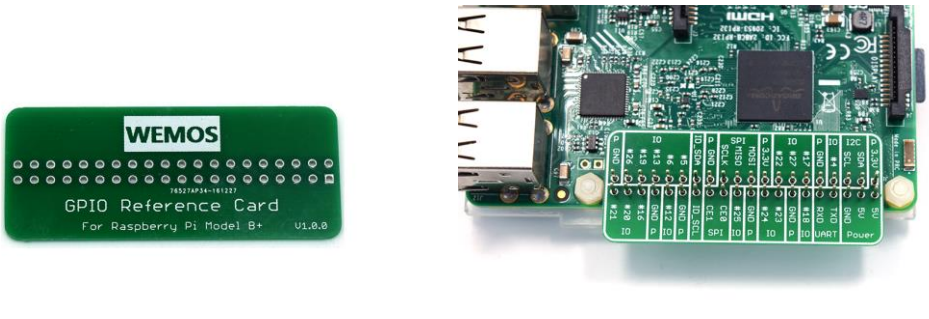


Figure 32: Wemos reference card for Raspberry Pi

3.5.6. OTHER EDUCATIONAL ROBOTS

While Wemos does not specifically target robotics or educational applications, it is possible to use their hardware in these settings. There are other pre-packaged educational robotic kits (beyond those mentioned above) that could be relevant to the eCraft2Learn project, for example:

- Lego Mindstorms EV3 (www.mindstorms.lego.com): This robot is usually programmed using its native EV3-G environment, but its architecture permits other approaches. An integrated SD socket makes it possible to boot different OSs and therefore to support different programming environment. For instance, an experimental Scratch extension called EV3+Scratch connects the EV3 reality to Scratch.
- Finch (www.finchrobot.com): This is robot kit is designed for engaging learners in computer science. Equipped with accelerometers, motors, buzzer, LED, temperature and obstacle

sensors, and drawing capabilities, it supports a dozen programming languages and connects with multiple development environments.

Figure 33: Lego Mindstorms EV3

Figure 34: Finch

4 FRAMEWORK, ARCHITECTURE, AND UNIFIED USER INTERFACE (UUI)

4.1. FRAMEWORK

One long-term goal of this project is to create a technical solution to be used in educational environments. Flexibility and adaptability are key aspects of the proposed ecosystem. The diagram below shows an overview of the framework.

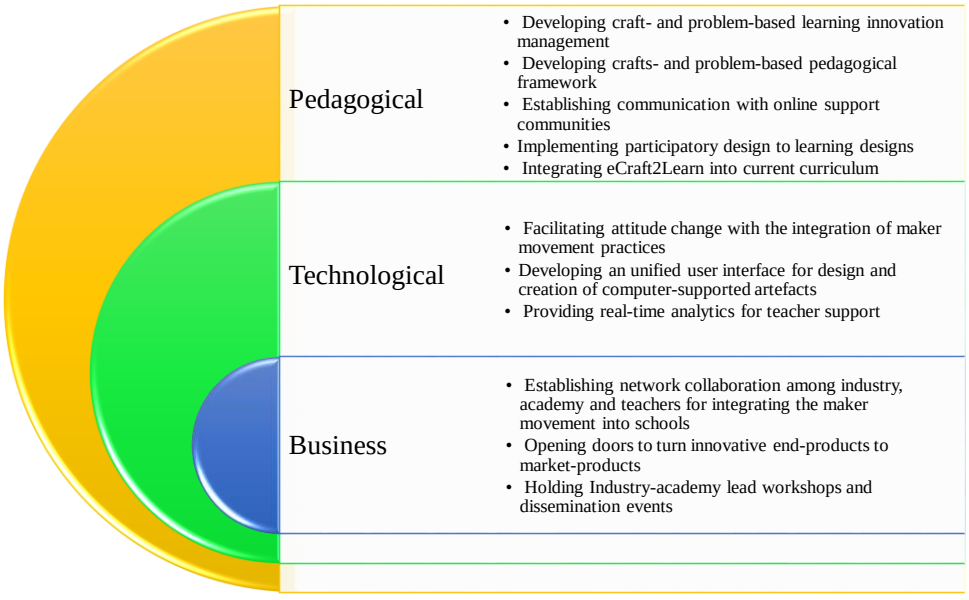


Figure 35: eCraft2Learn objectives (PO1-5, TO1-3, BO1-3) and their interrelations

The pedagogical (PO1-5) and technical (TO1-3) objectives assume that the proposed interface solution is evaluated across the lifetime of the project. Thus, there is a need for collecting data about what the users do while taking part in the activities, to contribute to the evaluation.

4.2. ARCHITECTURE AND UNIFIED USER INTERFACE (UUI)

The proposed interface will allow minor as well as major modification during the lifetime of the eCraft2Learn project. One of the challenges of the project is to allow the whole system to adapt to specific user categories (learner, teacher/coach, administrator). In order to test the proposed architecture, and its components, an initial generic user interface has been designed. This is termed the unified user interface (UUI).

The UUI is organised as a dashboard where users might easily add or remove components (see diagram below). The UUI features a central section where learning contents are available to learners. The teacher can choose, delete, and create icons (or "cards") from the learning objects repository, which is linked to several virtual learning management systems (e.g. Moodle, etc.). On the right side of the UUI, the students can access default management and planning tools. The planning tools are interchangeable and other options could be chosen based on learner and teacher needs.

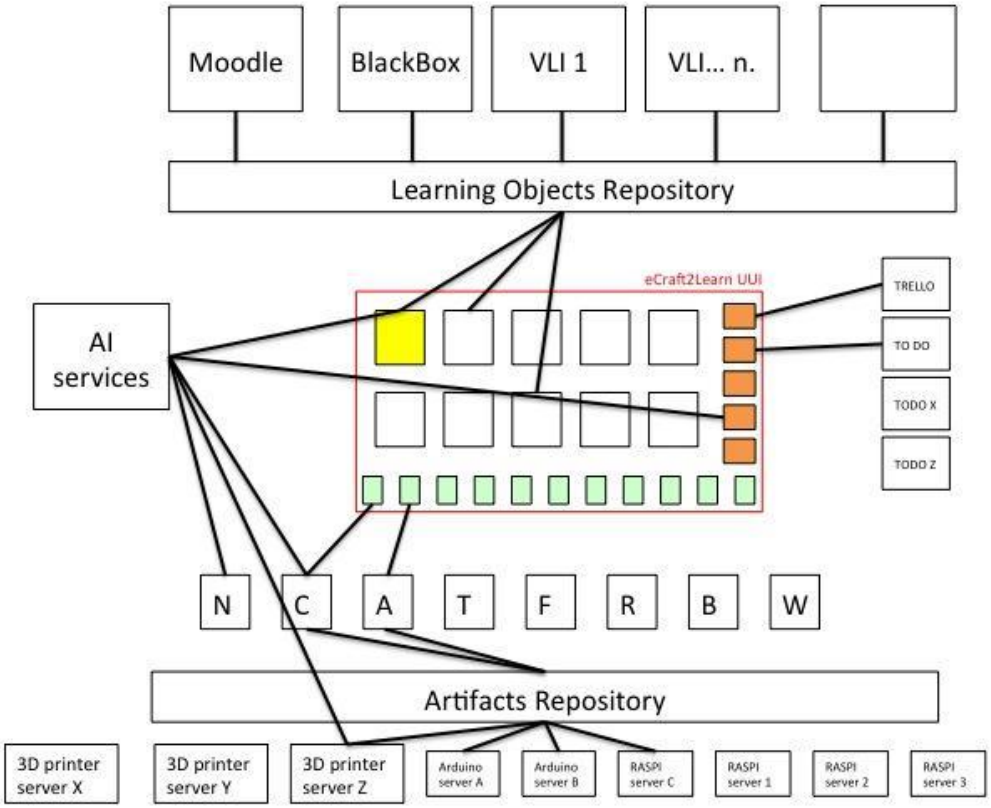


Figure 36: eCraft2Learn UUI architecture

On the bottom of the UUI we have our toolkit section. Interchangeable small icons bring the students to tools such TinkerCad, Arduino Create, Fritzing, and others (described above). The web-based

interface would enable learners and teachers to directly program an Arduino board and/or send 3D creations to a 3D printer.

The UUI will also feature an artefacts/project repository, where learners and teachers can access other creations and share their own creations. The artificial intelligence layer will connect with the UUI architecture layer to collect data and adaptively generate suggestions for the users.

Our UUI also relates to the five stages of our educational approach (ideate - plan - create - program - share). In the above diagram, in the central (main) area, we have tools for ideation and exploration (stage 1). On the right side of the UUI, we have the planning and managing (stage 2). And on the bottom, we have creating and programming (stage 3 and 4). Sharing and showcasing (stage 5) is also found on the right side of the interface, together with the planning options.

5 SUMMARY AND CONCLUSIONS

5.1. SUMMARY OF THE TECHNOLOGIES

We identified a wide range of technologies that have the potential to be included in the eCraft2Learn ecosystem. Appendix C presents a tabular overview of all of them. The table lists each technology and allows for a comparison based on simplicity, costs, attached communities, and capabilities for interfacing with other tools/systems.

We focused here on software, and also indicated some basic hardware possibilities (WP4.2 will explore in greater detail the hardware and robotics technology platforms). We looked at virtual learning environments (VLE), programming languages, programming environments, artificial intelligence services, and modelling tools/computer aided design (CAD).

We considered VLE systems such as Moodle and ILIAS, which enjoy wide deployment and large user and developer communities. We highlighted programming languages and related technologies, including Arduino, Processing, SCRATCH, Snap!, JavaScript and HTML5. We also assessed Artificial Intelligence Services that may be bundled with projects (IBM Watson, Google AI Cloud, Microsoft Cognitive Services, and Amazon AWS).

Members of the eCraft2Learn consortium also conducted a SWOT analysis of programming environments. After an initial screening of the programming environments, and a further assessment in relation to the objectives of the eCraft2Learn unified user interface (UUI), we narrowed the selection. The programming environments under consideration included: AgentSheets & AgentCube; ToonTalk Reborn; Squeak/Smalltalk; Alice; Greenfoot; Kodu; Minibloq; Modkit; Microsoft Small Basic; Mama; Snap!; ArduBlock; Makeblock and mBlock; Wyliodrin; Visualino; Bitbloq; Blockly for Arduino; and Netlabtoolkit.

5.2. CONSORTIUM RECOMMENDATIONS

Based on the considerations discussed in the previous sections, and the joint analysis of the technologies by the consortium members (summarised in the Appendices), the following recommendations were agreed upon for inclusion in eCraft2Learn: Arduino and Raspberry Pi for hardware circuit building, SCRATCH and Snap! for developing code to run on the hardware, Processing and Python for learning and understanding code, and HTML5/Javascript for the basis that will support the UUI. This would allow the UUI to remain transparent to the teachers and learners, who may also learn how it is constructed, and eventually contribute extensions to it through a sharing community.

With their strong focus on education and learning, the Fritzing and Arduino Create development environments were identified as especially well matched to the proposed eCraft2Learn ecosystem. TinkerCad and SketchUp for 3D modelling were also identified well-matched to our eCraft2Learn objectives.

The selection of appropriate programming languages and environment for the eCraft2Learn platform required a number of discrete steps: collecting resources, understanding the broader ecosystem that includes the learner and teacher perspectives to elaborate requirements, adding selection criteria related to broader concerns of administration and development, and finally mapping the selections to the overall project objectives.

The proposed programming languages and environments were analysed in terms of how well they enabled communication and collaboration in STEAM education, how user-friendly their approach was, and, in addition, how well they fit to both formal and informal education. The solutions proposed were favoured for their simplicity of use, cost effectiveness, adaptability to curricula and time restrictions, and openness for creativity and collaboration by learners and teachers.

6 REFERENCES

Bdeir, A. (2009). Electronics as material: littleBits. In *Proceedings of the 3rd International Conference on Tangible and Embedded Interaction* (pp. 397–400). Cambridge, United Kingdom: ACM.

Carroll, J. M. (2000). *Making use: scenario-based design of human-computer interactions*. MIT Press Cambridge, MA, USA.

Fourie, I., & Meyer, A. (2015). What to make of makerspaces: Tools and DIY only or is there an interconnected information resources space? *Library Hi Tech*, *33*(4), 519–525.

Greenberg, S., & Fitchett, C. (2001). Phidgets: easy development of physical interfaces through physical widgets. In *Proceedings of the 14th annual ACM symposium on User interface software and technology* (pp. 209–218). ACM New York, NY, USA.

Mellis, D., Banzi, M., Cuartielles, D., & Igoe, T. (2007). Arduino: An Open Electronic Prototyping Platform. In *CHI: ACM Conference on Human Factors in Computing Systems*.

Papert, S. (1980). *Mindstorms*. Basic Books New York.

7 APPENDICES

APPENDIX A - PROGRAMMING LANGUAGE TABLE

Table 1: Programming Language

Programming Language	Partner	Include (Y/N)	Strength	Weakness	Opportunity	Threats
Basic	All	N	Easy to learn	Limited for today's needs	Used by MS	Closed
Logo	All	N	Scratch's basis	Old	Grounded all educational languages and tools	none
	UNIPD	N	Pedagogically sound and effective	Already 'implied' by other choices (Scratch, Snap)	-	-
Pascal>Delphi	All	N	Elegant language	Not easy to learn	Used in scientific field	Complicated solutions, may lead to too simple project, and lack of motivation
Python	UNIVDUN	Y	Intermediate step between HW and WEB application	Require basic programming skills	fast growing	None
	UNIPD	Y	Of interest as intermediate language /tool	Textual, not very easy	Current relevant interest and wider and wider knowledge	-
	ARD	Y	Relatively easy (compared to other languages) to understand/learn, forgiving	Harder than visual programming	It is widely used, and the knowledge would be instantly applicable	-
	EDUMOTIVA	Y	Easy, good support, useful, can be used on Arduino and Raspberry Pi, lots of scientific libraries	Textual, not visual on its original programming environment	Widely spread	-
	EVOTHINGS	Y	Easy to learn, many hw platforms available	Interpreted language, performance issues	-	-
Ruby	UNIVDUN	Y/N	Scripting for Sketchup	Not easy to learn/textual	Widely spread	None

Programming Language	Partner	Include (Y/N)	Strength	Weakness	Opportunity	Threats
Visual Basic	UNIVDUN	N	Powerful language	Exclusive Microsoft	Widely used	Connected to Microsoft strategies
C/C++ (Arduino)	UNIVDUN	Y	Very powerful language	Not easy to learn	Suitable both for Hw and Sw	None specific
	UNIPD	Y	Strictly related to Arduino	Textual (but it can be used as an intermediate language of other graphical languages)	Code downloadable onto an Arduino board to be run autonomously	-
	ARD	Y (biased)	Relatively easy (simple functions, etc.)	Textual(Not as easy to learn as visual programming tools)	Widely used, large and community that can support the learning process	-
newLISP	All	N	advanced language	Complex to learn	Used in several area	none
Processing	UNIVDUN	Y	Visual applications development	For advance use, basic programming skills required	Widely used for education	MIT licence
	UNIPD	Y	A computer running a Processing application can interact with an Arduino board	-	Processing is a flexible software sketchbook and a language for learning how to code within the context of the visual arts	-
	ARD	Y	Easy to setup I/O communication between Processing and I/O board(such as Arduino)	Textual - relatively easy to learn, but does require basic knowledge about programming	Visual output that suitable for showcasing purposes	-
Oz	All	N	Advanced language	Not so well known	Education oriented language	Complicated solutions, may lead to too simple project, and lack of motivation
JavaScript[1]	UNIVDUN	Y	SNAP! is implemented with JS	Not easy to learn	Extend the ToonTalk Reborn language. Overcame limitation of Server	none
	UNIPD	Y	Snap! is developed in Javascript, thus new blocks can be programmed in JS and added to the language	Textual, only partly Java inspired	Extensions to Snap!	-
	ARD	Y		JavaScript is textual, and not as easy to learn	JavaScript is widely used, and directly applicable knowledge	-

Programming Language	Partner	Include (Y/N)	Strength	Weakness	Opportunity	Threats
	EDUMOTIVA	Y	Widespread in web development, cross platform from PC, mobiles, smart TVs. Can use a block type of programming like blockly for students which is javascript based	Not easy to learn. Need also knowledge of HTML, css. Problems with old web browsers	Widely used	-
	EVOTHINGS	Y	Largest language on earth	Versioning on older browsers	JS is finding its way also into embedded systems	-
HTML5	UNIVDUN	Y	Pages for Web	Some old browsers don't support it	Widely spread	none
	ARD	Y	Easy to learn	Doesn't have much in common with programming languages(it's a markup language)	Directly applicable knowledge	Needs javascript and css to impress anyone, and can therefore be difficult to teach
	EDUMOTIVA	Y	Easy to learn, widespread. cross platform from PC, mobiles, smart TVs.	Need of javascript, css. Problems with old web browsers	-	-
XML	UNIPD	not sure	Snap! saves programs in XML, so it is relevant as a portable intermediate language	-	a portable intermediate language	-
	ARD	not sure	Easy to learn	Can't be used standalone - needs other languages to become meaningful (little in common with other languages)	Directly applicable knowledge	-
JS-Eden	All	Y	Powerful and recent	Still early dev stage	Java Script based	none
Flash (Action Script[2])	All	N	Interesting tool	Old	Widely spread	Adobe licence
	ARD	N	Relatively easy to learn	-	-	Becoming obsolete
	UEF	N	-	Obsolete	-	Obsolete
	EDUMOTIVA	N	Use modern technologies	Not easy to learn. No wide documentation	-	Obsolete, use custom language not wide spread

Programming Language	Partner	Include (Y/N)	Strength	Weakness	Opportunity	Threats
Swift	All	N	New elegant language	Textual and complex to learn	none	none
	UEF	N	-	Vendor lock	none	-
	UNIVDUN	Y	Modern	-	Open Source	Apple's baby...
	EVOTHIGS	N	-	Mostly mobile platforms	None	-
	UNIPD	Y	Efficient, implemented in standard C	-	Good for embedded system, e.g. Raspberry PI	None
LUA	ARD	Maybe	Relatively easy to learn	-	none	Not sure how widely it is used?
	UNIVDUN	N	Free and Open Source	Game scripting language	Embedded into game engines	Obsolete
	EVOTHINGS	Y	Very easy to learn, also for smaller children	Not very widespread	Make gamers interested in robotics	-

APPENDIX B - PROGRAMMING ENVIRONMENTS TABLE

Table 2: Programming Environments

Programming Environment	Partner	Include (Y/N)	Strength	Weakness	Opportunity	Threats
AgentSheets & AgentCube	All	N	Easy to use	AgentSheets is very old	-	Obsolete
	ARD	N	engaging through game design	No 'undo', and not very user friendly	None	-
ToonTalk Reborn	ARD	Y	Web based, very feature rich	User interface can be improved	Using speech recognition and other modern tech	-
Squeak/Smalltalk	All	N	Powerful tool	Quite old		Obsolete
Alice	All	N	Easy to use	Abstractions of programming concepts	Widely spread	Obsolete
Greenfoot	All	N	Very simple to use	Only for simple things	Widely used	None
Kodu	All	N	Nice interface		Connected to game programming	Microsoft Proprietary
Minibloq	UNIVDUN	N	Clear graphical interface	early development	Blocks translated instantaneously into text code	Sustainability
	EDUMOTIVA	Maybe	Stand-alone program, which does not require Arduino, installed. Apart from images has also code	Less known comparing to scratch	None	further analysis needed
	ARD	Maybe	You can see the board, block language and textual programming	-	-	further analysis needed
Modkit	UNIVDUN	N	Polished interface	$ 399.99 USD school licence	Scratch family	Proprietary
	EDUMOTIVA	N	-	It costs, it is not a sustainable solution	-	-
	ARD	N	Looks like a well thought through and ok looking environment	Cost and not being open source	-	further analysis needed
Microsoft Small Basic	UNIVDUN	N	Easy language	Basic language	Basic is widely used	Microsoft proprietary
Mama	UNIVDUN	N	Multilanguage	Complex interface	3D programming	Sustainability
	ARD	N	-	Windows only, hard to engage kids of today with dated 3D graphics	-	-
Snap!	EDUMOTIVA	Y	Easy to use interface Pure JavaScript implementation. It runs in every modern browser.	lacks the community that the Scratch website has.	Can create a eCraft2Learn version	-
	ARD	Y	Easy to use, available in different forms, open source	-	further analysis needed	-
ArduBlock	UNIVDUN	Maybe	Typo proof	Arduino "plug-in" Not simultaneous block > text	Arduino world	Sustainability
	ARD	Maybe	Clear hierarchies in the block code	Not simultaneous block and text, not sure if available for web?	-	-
Makeblock and mBlock	UNIVDUN	N	Interesting programming paradigm diverging from Scratch	Kickstarter project	None	Premature

Programming Environment	Partner	Include (Y/N)	Strength	Weakness	Opportunity	Threats
	ARD	N	Seems easy to use, and looks good	Not open source, and seems hard to integrate into anything else	-	-
Wyliodrin	UNIPD	Y	Web based platform compatible with several embedded systems, Arduino and RPI included	-	Devices can be driven in a wireless form	-
Visualino	All	Maybe	Visual programming paradigm	Early development stage	Arduino World	Sustainability
Bitbloq	All	Maybe	Visual environment	Not widely spread	Arduino World	Sustainability
Blockly for Arduino	ARD	Y	Block language for programming	Only Arduino	Simple, seems made for integration with other software	
RoboMind	UNIVDUN	Y	Visualization and text Standard interface design	Pay for account (£4.50)	Programming real robots (e.g. NXT)	None
OPEN Roberta	UNIVDUN	Y	Looks good Nice environment	-	Multi-platform	None
	ARD	Y	User friendly, great tutorial	Only robots? Might be a gender issue		Gender bias
netlabtoolkit	UNIVDUN	N	Nice and real time control feedback	Early development stage	Cloud based Internet of Things	Sustainability
	ARD	Maybe	Multi-platform, seems easy to use(haven't tried it)	In development(possibly buggy)	-	-
Fritzing	UNIVDUN	Maybe	Good interface design	Simulator environment	Arduino world	None
	ARD	Maybe	Easy to understand, no need to learn electronics symbols and schematics	-	further analysis needed	-
Arduino Create	UNIVDUN	Y	Completely web based	Installing plug-in	Large Arduino community	Dependent to Browsers
	ARD	Maybe	Web based	Textual	-	-

APPENDIX C - TECHNOLOGIES OVERVIEW TABLE

Table 3: Technologies Overview

Categories	Technologies	Considered	Open Source/ permissive licence	Fee based (company charges for use)
Virtual Learning Environments	Moodle	x	x	
	Blackboard			x
	ILIAS	x	x	
	Google Classroom			
	Brightspace			x
	Sakai		x	
Programming languages	BASIC			
	Logo	x	x	
	Pascal > DELPHI			x
	Python	x	x	
	Ruby		x	
	Visual Basic			x
	C (family) Arduino	x	x	
	newLISP		x	
	Processing	x	x	
	SCRATCH SCRATCH For Arduino	x	x	

Categories	Technologies	Considered	Open Source/ permissive licence	Fee based (company charges for use)
	Snap!	x	x	
	Oz		x	
	JavaScript JSON [XML alternative] JS-Eden	x		
	HTML5	x	x	
	Flash			x
	Swifts		x	
	Go		x	
	LUA		x	
Programming environments	AgentSheets & AgentCubes			
	ToonTalk & ToonTalk Reborn	x	Some versions	
	Squeak/Smalltalk			
	Alice			Some
	NetLogo		x	
	Greenfoot		x	
	Kodu			x
	RoboMind		some	
	Minibloq		x	

Categories	Technologies	Considered	Open Source/ permissive licence	Fee based (company charges for use)
	Modkit			x
	Microsoft Small Basic			x
	Mama			x
	ArduBlock		x	
	Makeblock and mBlock			
	Wyliodrin		some	some
	Visualino			
	Bitbloq			
	Blockly for Arduino	x		
	OPEN Roberta			
	NTK			
	Fritzing	x	x	
	Arduino Create	x	x	
Artificial Intelligence Services	IBM Watson	x		In some cases
	Microsoft Cognitive Services	x		In some cases
	Google AI Cloud	x		In some cases
	Amazon Alexa Voice Service	x		In some cases
modelling tools (CAD)	TinkerCad	x		
	BlocksCAD	x	x	

Categories	Technologies	Considered	Open Source/ permissive licence	Fee based (company charges for use)
	OpenSCAD		x	
	Fusion 360		x	Just for Pro version
	Blokify		x	
	SketchUp	x	Some versions	x
	Sculptris		x	
	Makers Empire			x
	3DMTP (3D Model To Print)			x
	Art of Illusion		x	
	3DSlash		x	
	Doodle3D			x
	FreeCAD		x	

APPENDIX D – SCENARIO ROLES / FUNCTIONS / TECHNOLOGIES

Table 4: Roles, functions and technologies

Specifications	Teacher	Students
Create activity connect lesson plan materials with selections from project library	x	
VLE project library search	x	
VLE create a new project	x	
Connect to school resource management for equipment availability / booking	x	
Organise students into groups use interactive student assessment software to create balanced groups with clear skill levels	x	
Fill out web form with skills, preferences		x
Accept/modify auto-generated groups	x	
Select project(s) use group skill level information to assign specific projects or project elements to groups/individuals	x	
Browse suggested projects auto-matched to group/individual skill levels	x	
Select project with multiple sub-projects	x	
Assign (sub-)projects to groups/individuals	x	
Open project view with links to materials, software		x
Design 3D model (software) use interactive software interface to select model templates, add/resize elements, format for printing		x
Select software app for designing model		x
Import model from project library		x

Specifications	Teacher	Students
Open tool menu with model elements		x
Add elements to model		x
Resize/modify elements and connections		x
Export/share model to printing format		x
Select software app for print formatting		x
Import model from repository (from newly added model or existing/shared model)		x
Configure model for printing (size, resolution, etc.)		x
Select printer for output	x	x
Verify/Dispatch print job	x	x
Print / assemble 3D model (hardware) observe printing process, and as needed: cancel/restart, make adjustments in software, re-calibrate hardware		x
Monitor print queue	x	x
Pause/Cancel print job	x	x
Add/replace raw printing material	x	
Collect and assemble printed pieces		x
Design electronic circuit (software) use interactive software to select components (e.g. LEDs), assign to pins, develop code-based instructions for electronic behaviour (e.g. colour cycle, blink, sensor response, etc.)		x
Select software app for circuit design		x
Select software app for coding / compiling / hardware transfer		x
Browse circuit design repository	x	x

Specifications	Teacher	Students
Browse code repository	x	x
Create new circuit design	x	x
Create new code	x	x
Import circuit design from repository		x
Import code from repository		x
Modify imported circuit		x
Modify imported code		x
Export/share circuit design	x	x
Export/share code	x	x
Print reference schematic of circuit design	x	x
Transfer code to hardware (e.g. Arduino)	x	x
Build electronic circuit (hardware) follow software-generated diagram for connecting chips, wires, components		x
Select board / components	x	x
Access station for assembly		x
Test hardware functionality		x
Test (loaded) software functionality		x
Modify component/wiring connections	x	x
Initiate overwrite of running code with new code	x	x
Assemble 3D printout with electronics build		x

Specifications	Teacher	Students
integrate 3D printout with electronic circuit (e.g. insert in simple enclosure, or run wires through channels for distributed components such as LED ‘eyes’)		
Access station for assembly		x
Embed board, wiring and components		x
Join internal wiring/components / Repair damaged connections		x
Test embedded system functionality (monitor/modify moving gears, joints, etc.)		x
Assess project(s) observe physical project result, discuss project process, discuss broader lesson plan	x	
Compare 3D model and 3D printout	x	
Compare circuit diagram and assembled circuit	x	
Observe (raw) code and running code	x	
Discuss work process and craftsmanship	x	x
Discuss relationship of project and lesson	x	x
Assign score/mark	x	