Introduction to Programming Using Mobile Phones and MIT App Inventor

Sergio Barrachina Mir and Germán Fabregat Llueca

***Abstract*—At the beginning of each year, we ask our new undergraduate students in Computer Engineering if they have ever developed a computer program. Surprisingly, the most frequent answer is no. The few students who have attended a Computer Science training module usually have some basic programming notions; however, most of our students coming straight from high school have never programmed. This lack of basic programming skills represents a major drawback when taking programming-related courses. This is especially true for the course on Computer Organization, taught during the first semester of the first year, as one of its main objectives is to explain the processor architecture, and therefore a great part of it revolves around programming in assembly language.**

To tackle this lack of basic programming skills, a workshop on mobile application programming using MIT App Inventor is offered to freshmen. This workshop is highly welcomed and positively received by the students, and we believe that it has contributed to improving their performance on courses related to programming, and in particular, on the Computer Organization course.

***Index Terms*—Programming basic concepts, mobile programming, MIT App Inventor**

I. Introduction

THE *Computer organization* course is taught at Jaume I University in the first semester of the first year of the Computer Engineering and Computational Mathematics degrees. Prior programming skills are not a prerequisite for the course. In fact, in view of this, the course was redesigned in the last years and two teaching resources [1][2] were developed to support and motivate students in learning its material. Even though these changes have improved success rates and performance on the course, students without basic programming skills still struggle; this is because the course details the low-level functioning of computers, i.e., it is shown how a computer executes programs in machine code. Thus a student without a basic knowledge of programming concepts—such as data types and structures, control structures, data encoding, etc.—will not be able to build on this base to understand the hardware mechanisms that make programming possible.

On the other hand, despite the statements made since the AENUI-CODDII [3] declaration in favor of including specific science and information technology classes in basic schooling at the secondary level and in the Spanish Baccalaureate, computer science classes continue to be elective at these levels; furthermore, only part of these classes teach programming. This implies that, in practice, many of the students enrolling in Computer Engineering degrees do not possess basic programming skills, despite having chosen this specialization.

To remedy this situation, we have been offering an intensive workshop to first year students at the beginning of their first semester, aiming to help them develop a set of basic programming skills in a practical and intuitive way, oriented especially towards improving the learning and teaching process on the *Computer organization* course.

We discussed this workshop and its results in the presentation “Can I program my phone? But I just got here!” published in the proceedings of the conference *XXV Jornadas sobre Enseñanza Universitaria de la Informática* (JENUI 2019) [4]. The presentation was selected as one of the top two at JENUI 2019 and suggested to be published in IEEE-RITA. This article elaborates on that presentation by detailing the experimental design, describing the suggested student projects in more detail, and indicating in which way each project contributes to achieving the learning objectives of the workshop.

S. B. and G. F. Authors are with the Department of Computer Science and Engineering, Jaume I University, 12071–Castellón de la Plana, Spain (e-mail: barrachi@uji.es and fabregat@uji.es).

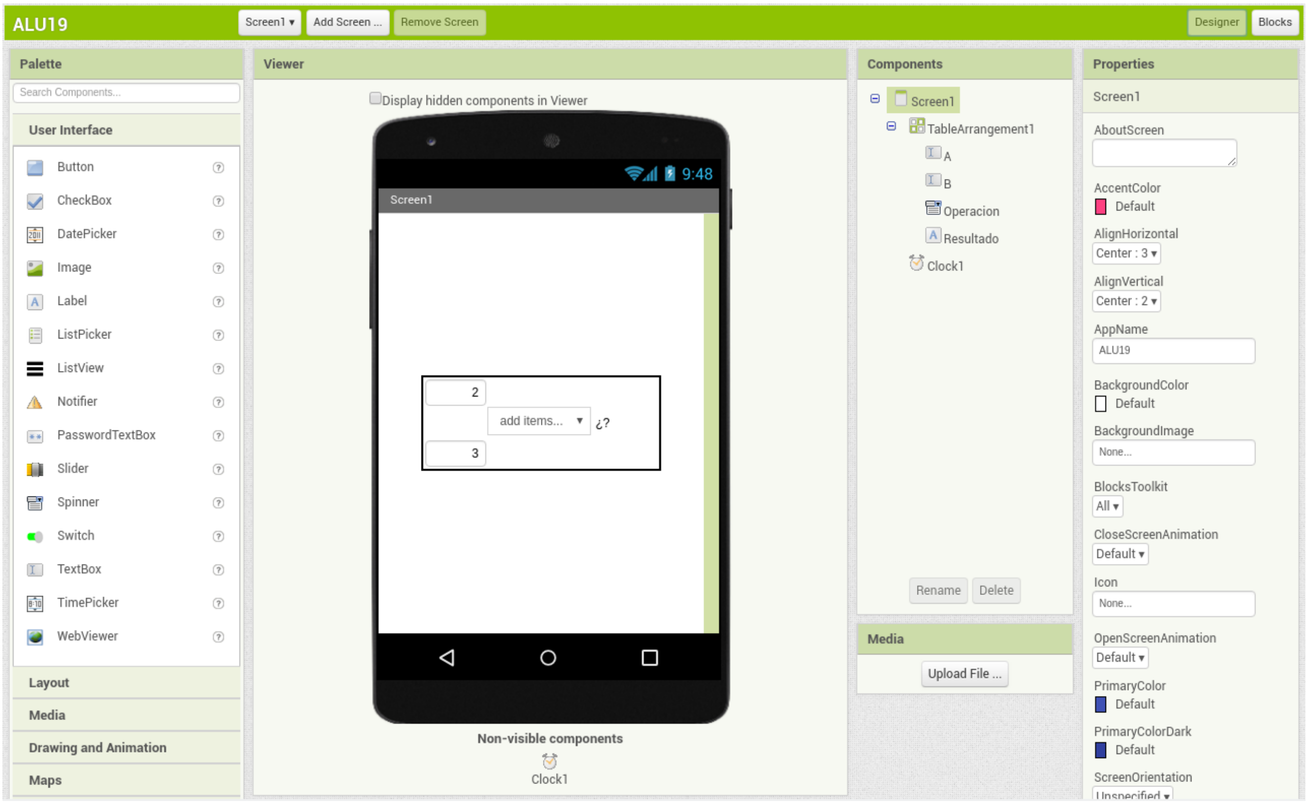
II. Learning Objectives

The learning objective of the proposed workshop is that first-year students master a set of basic programming skills allowing them to improve their outcomes in related subjects, especially on the *Computer organization* course. As discussed above, this first-semester course introduces the Computer organization concepts needed to understand how the processor executes programs; consequently, the better a student understands programming concepts, the easier it should be to meet many of the learning objectives of this course.

III. Experimental Design

To be able to reliably confirm that the students' academic results after having taken the workshop are correlated with their participation rather than being determined by other factors, the participants should have been selected in a randomized way from among all first-year students.

However, since we assumed that the workshop would help students improve their learning in the courses most related to programming and that it would be especially beneficial to those with little or no knowledge of programming, we opted for offering the workshop to all students, stating its learning objectives and that it would be especially interesting for those with little or no programming knowledge. In so doing, we offered all students an opportunity to participate. For the same reason, the workshop was offered free of charge and on a



Screen1
AboutScreen
AccentColor: Default
AlignHorizontal: Center : 3
AlignVertical: Center : 2
AppName: ALU19
BackgroundColor: Default
BackgroundImage: None...
BlocksToolkit: All
CloseScreenAnimation: Default
Icon: None...
OpenScreenAnimation: Default
PrimaryColor: Default
PrimaryColorDark: Default
ScreenOrientation: Unspecified

Fig. 1: Graphical interface editor of MIT App Inventor

schedule on which all students could attend, regardless of the group in which they were enrolled.

Due to the fact that the admission process was not randomized, in comparing the results of students having taken the workshop with those of students who did not, it should be taken into account that any differences might in truth be caused by other factors not considered in this study.

On the other hand, the students were informed that their academic results would be used in aggregate form for this study. However, to prevent this knowledge from affecting their results, they were informed of this only after the assessment stage of the various courses had been completed.

IV. Choosing a Programming Environment

Since the goal of the workshop is for students to acquire basic programming skills at the beginning of the academic year in a practical and intuitive way, and in only a few sessions, it is very important to choose the programming environment most suited to accomplish this.

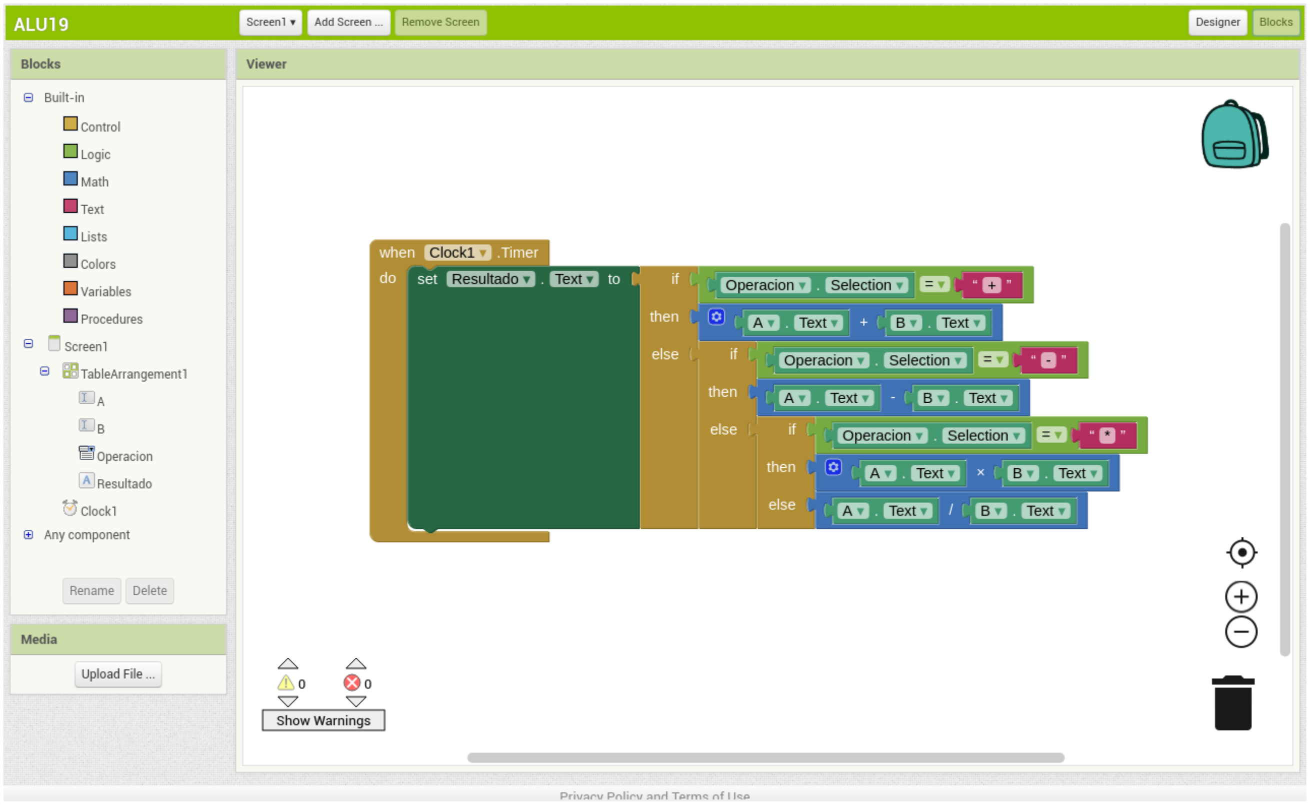
In order to introduce the basic programming concepts in a fast-paced and visual way, the first decision we made was to resort to a visual programming environment. Among the available environments, we considered using either Scratch or MIT App Inventor.

Scratch [5] is a rather well-known programming environment which allows users to create applications by dragging and dropping blocks instead of writing code. It was developed to help young people, especially those between 8 and 16 years of age, learn to program. Furthermore, it has been successfully used both at earlier educational stages and in introductory programming courses at the university level [6], [7], [8]. Another feature making Scratch interesting is that its programming environment can be easily extended. There are, among others, extensions enabling the user to use Scratch to program various hardware devices: *Lego Mindstorms NXT* [9] robots or the *ad-hoc* hardware of the educational project SUCRE4Kids [10].

MIT App Inventor[1], on the other hand, is a visual programming environment featuring an interface very similar to that of Scratch, but which is oriented towards the intuitive development of fully functional applications for mobile phones and tablets. MIT App Inventor allows the user to visually define both the elements of the application's graphical user interface (see Fig. 1) and its code (see Fig. 2). Thus it allows the user to develop programs with the same facility as does Scratch, but in addition the students obtain a tangible product: a mobile application[2] which they can demonstrate and install on their friends' and family members' phones, or even publish in the Google Play online shop. This orientation towards mobile app programming and the development of a real, demonstrable product provides an important motivation and has typically tipped the scales in favour of MIT App Inventor rather than Scratch as an introductory programming environment for undergraduate students [11][12][13].

[1]http://appinventor.mit.edu/explore/about-us.html

[2]Even though MIT App Inventor currently only supports the development of applications for Android, a version for iOS is in development. In any case, there are environments based on MIT App Inventor that support publishing applications for iOS, e.g., Thunkable (https://thunkable.com/).



```
when Clock1 .Timer
do  set Resultado . Text to
      if    Operacion . Selection = "+"
      then  A . Text + B . Text
      else  if    Operacion . Selection = "-"
            then  A . Text - B . Text
            else  if    Operacion . Selection = "*"
                  then  A . Text × B . Text
                  else  A . Text / B . Text
```

Fig. 2: Code editor of MIT App Inventor

With respect to our goal of developing basic programming skills useful for the *Computer organization* course, MIT App Inventor possesses an additional advantage: it allows its apps to interact with the mobile phone's sensors (camera, timer, GPS...), so that it can be used for introducing the concepts of input/output in an easy and practical way; these concepts are subsequently elaborated in the coursework.

For all the above reasons, we decided to use the MIT App Inventor platform in teaching this workshop.

V. Mobile App Development with MIT App Inventor

The workshop "Mobile app development with MIT App Inventor" is available at the following URL to any teacher who wishes to teach it or to recommend it to students as a self-study tool: <http://lorca.act.uji.es/curso/mit-app-inventor/>.

It is directed at students recently enrolled in a Computer Engineering degree, especially those with no prior education in programming. Upon completion, the student should be able to do the following:

LO1 Distinguish values from variables.
LO2 Recognize various conditional structures and know some of their uses.
LO3 Recognize various iterative structures and know some of their uses.
LO4 Recognize some of the objects provided by MIT App Inventor.
LO5 Distinguish between object properties and functions.
LO6 Encapsulate blocks of code using functions.
LO7 Interact with the input/output devices of a mobile phone (timer and camera).
LO8 Design simple user interfaces.
LO9 Recognize various types of events (button click, alarm, screen initialization and camera response) and link them to parts of the code.
LO10 Develop simple mobile apps using MIT App Inventor.

Given that the workshop must be short in duration in order to be taught as a whole at the beginning of the academic year without overloading the students, one has to bear in mind that the above objectives will be met at a very basic level. Based on this consideration, we recommend a workshop duration of 10 hours, split for example into 4 sessions of 2.5 hours each, paced at one session per week.

Following MIT App Inventor recommendations for introductory courses[3], in the first session the programming environment is presented, the application is accessed, the students' mobile phones are set up correctly, and the four basic tutorials provided by MIT App Inventor are completed in a guided way. The latter show how to develop the following apps step-by-step: i) an application playing speech sounds, ii) an extension of the former, an application playing speech sounds when the user shakes the phone, iii) an application allowing the user to drag a ball across the screen with a finger, and iv) a drawing application that allows the user to take photos.

In the following sessions, rather than drawing on the MIT App Inventor's library of tutorials[4], we have opted for

[3]http://appinventor.mit.edu/explore/teach.html
[4]http://explore.appinventor.mit.edu/ai2/tutorials

TABLE I
RELATIONSHIP BETWEEN THE PROJECTS AND THE LEARNING OBJECTIVES OF THE WORKSHOP

	1	2	3	4	5	6	7	8	9	10
ALU simulator	●	●		●			●	●	●	
Registers bank	●	●	●	●	●	●	●	●	●	
RGB photo	●		●	●	●	●	●	●	●	●
Memory game	●	●	●	●	●	●	●	●	●	●
Weather app	●	●	●	●	●	●	●	●	●	●

developing our own projects which promote the workshop's learning objectives by introducing computer architecture concepts in addition to basic programming notions.

Each of the projects proposes the incremental, step-wise development of a specific application. The projects have been organized in such a way that the student obtains a functional version as soon as possible, even if it represents a very basic and incomplete version of the final application. With each subsequent step the student enhances the project further, but here again the idea is to arrive at a new functional version as soon as possible, so that the student can test the extensions just implemented. In this way, the various components and blocks of code needed to develop the application can be introduced incrementally.

Furthermore, two kinds of measures have been taken to accommodate diversity. On the one hand, at the end of each step we provide a link allowing the student to download a correct version of the project after completion of the given step. In this way, if a student does not manage to complete a given step on his/her own, he/she can always continue with the next one while building on a correct version. On the other hand, at the end of many of the steps, possible further extensions related to the one just realized are suggested. In this way, if a student is far ahead of the others, he/she can always explore the suggested extensions on his/her own.

The projects were designed to tackle the learning objectives of the workshop relevant to the *Computer organization* course (LO1, LO2, LO3, LO6, LO7 and LO9) in an incremental manner and to introduce course-related concepts: computer components and information access (*Simulator of an ALU and of a registers bank*), encoding and input/output (*RGB photo*), complex data structures (*Memory game*) and interruptions (*Weather app*). The relationship between the various projects and the learning objectives is shown in Table I.

A. ALU Simulator

The first project proposed develops a mobile app which simulates an arithmetic logic unit (ALU) (see Fig. 3) which carries out an arithmetic operation—the user has a choice between addition, subtraction, multiplication and division—on two numeric operands, and displays the result. The development of this project encompasses the following stages: i) an adder, which given two inputs updates its output on button press; ii) an ALU which allows the user to select the desired operation and updates the result on button press; and finally, iii) an ALU which periodically updates its result as a function of its input values and the operation selected.

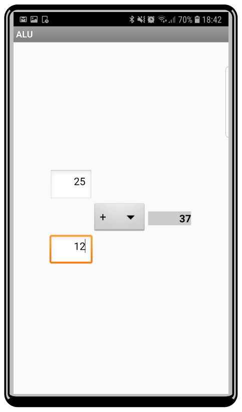
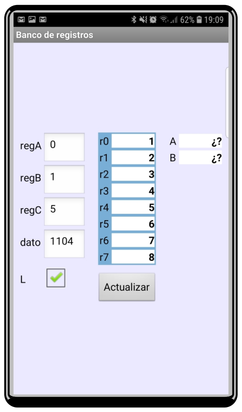


Fig. 3: ALU simulator


Banco de registros

regA 0
regB 1
regC 5
dato 1104
L ✓

r0	1
r1	2
r2	3
r3	4
r4	5
r5	6
r6	7
r7	8

A ¿?
B ¿?

Actualizar

Fig. 4: Registers bank simulator

This project first demonstrates the use of the graphic layout elements, tags, drop-down lists and the mobile phone timer.

Concerning programming, the project demonstrates how to modify object properties, read the selected element of a drop-down list, the use of conditional control structures and how to respond to a timer event.

B. Registers Bank Simulator

The second project proposes developing a graphic simulator of a bank of 8 registers (see Fig. 4) which can simultaneously read out two of its registers and write a numerical value into one of them. The extension suggested at the end of this project is to integrate the above-mentioned ALU simulator with the registers bank so the former operates on the register contents.

The first step of this project consists in creating the graphical interface showing the register contents, whose values are given as list elements. In the second step the student uses code to initialize this list. The third step implements the read function and the fourth the write function. The final step consists in carrying out a read or write operation depending on whether a corresponding box is checked.

Regarding the graphical interface, this project introduces the use of boxes and a more complex use of the graphic layout elements, including the personalization of background colors.

In terms of programming skills, we introduce the list as a data structure with its basic operations: declaration, initializa-

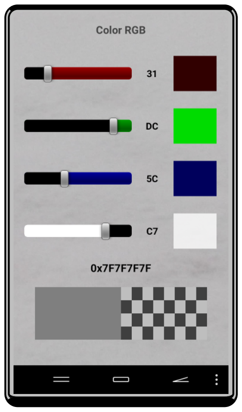


Fig. 5: RGB photo

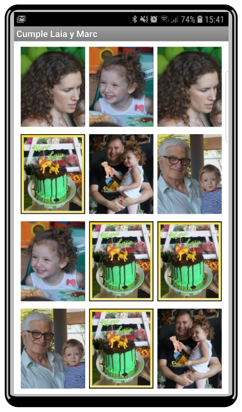


Fig. 6: Memory game

tion, adding elements, and reading and writing them; global variables; iterative control structures and procedures.

C. RGB Photo

The third project consists in the development of a mobile application that allows the user to select a color using RGBA components, then take a photo and shift its hue along the color axis previously defined (see Fig. 5). The value of each RGBA color component can be modified using a sliding control and is shown as a hexadecimal number. The selected color is displayed in three ways: as a hexadecimal number; by varying the hue of a box without taking transparency into account; and by varying the hue of a box in which a selected level of transparency is applied, which allows an underlying chessboard pattern to become visible to a corresponding degree.

This project comprises four steps: i) define the graphical interface; ii) select the values of color components using sliding controls; iii) construct the color based on its components; and iv) take a photo and modify its hue based on the selected color.

In this project we introduce the use of sliding controls and access to the mobile phone's camera for taking photos.

In terms of programming skills, this project expands on the use of procedures and control structures, and introduces aspects of representation—the hexadecimal representation of numbers—and information encoding—the conversion of real numbers (position of the slider) into integers and the construction of a color value (4 bytes) based on its components (1 byte each).

D. Memory Game

This project consists in the development of a memory game in which 6 pairs of identical cards have to be matched while only two cards can be turned at any one time (see Fig. 6).

This project is divided into the following steps. The first step consists in creating the graphical interface of the game, loading the images corresponding to the front and back of the cards and adding the code for turning up to three cards—which is actually implemented using buttons. In the second step we show how sounds can be added to the game and how to make the app play a sound whenever a card is turned. The third step demonstrates how to use a procedure to turn a specific card based on a parameter value—instead of repeating the same code multiple times for turning each card. The fourth step illustrates how a data structure—a list of lists—can capture the information required for the logic of the game and simplify the above procedure call. The fifth and final step specifies the algorithm that has to be implemented to complete the game—up to this point, cards could be turned, but the rules of the game did not yet apply. The final step also includes playing sounds when turning identical vs. different cards, and at the end of the game—when all cards have been paired. At the end the project includes an additional optional step in which possible extensions are specified.

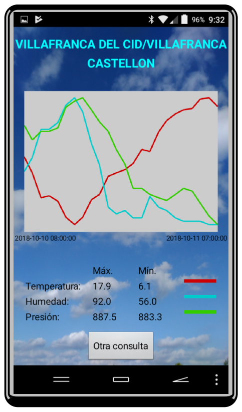
This project does not introduce any new interface elements, although it demonstrates how buttons can be used to simulate other types of elements, in this case cards, simply by changing their background image.

The following programming concepts are introduced: i) a more complex data structure (a list of lists) and how to access its elements; ii) the use of local variables in a procedure; and, finally, iii) how to implement an algorithm that is more complex than the ones seen up to this point.

E. Weather App

This project proposes the development of an application that downloads and visualizes meteorological data from the Spanish State Meteorological Agency (Agencia Estatal de Meteorología, AEMET) (see Fig. 7). This project is the most complex one and demonstrates how to obtain information from a web service and work with asynchronous responses. Furthermore, unlike the previous projects, which used only a single screen, this project allows the user to switch between a set of screens, each of which requests or shows a specific type of information.

The project is organized into the following steps. In the first step, we describe the OpenData API used by AEMET, the JSON format and the MIT App Inventor extension which allows the developer to work with JSON data. In the second step, we describe the web component and how to use it to send web requests and react once the response to a particular request is received. In the third step, we demonstrate how to use a


VILLAFRANCA DEL CID/VILLAFRANCA CASTELLON

	Máx.	Mín.	
Temperatura:	17.9	6.1	🟥
Humedad:	92.0	56.0	🟦
Presión:	887.5	883.3	🟩

Fig. 7: Weather app

canvas to represent the received data graphically. In the fourth step, we demonstrate how to transfer data between the different screens of the application. In the final step we introduce error management mechanisms which allow the application to handle both expected and unexpected errors in the most adequate manner possible.

Concerning the interface, in this project students see for the first time how to create an application consisting of several screens and how to use canvasses for plotting graphs.

In terms of programming skills, the students see how to transfer information across several screens of an application, how to use components to receive web information in an asynchronous manner, and how to manage the possible errors that may occur while the application is running.

VI. Results

The workshop was offered to the 164 students enrolled in the *Computer organization* course, while specifying that it would be especially useful to those with no prior knowledge of programming. Of these, 46 students enrolled and 41 (90%) completed the workshop. However, one has to bear in mind that the fact that the students themselves chose to participate in the workshop could have influenced their motivation to complete it, which would explain the high completion rate.

The students' attitude during the workshop was markedly positive; they participated actively and completed the proposed exercises. Furthermore, even though additional hints were included after each of the steps, the students did not fall back on them, except for one group on one occasion. Even though they knew this support was available, the students preferred to find the solutions on their own. We should bear in mind that the high motivation and willingness to learn demonstrated in the workshop might also have influenced their academic results.

The students assessed the workshop using a standard student satisfaction survey[5] in which we asked them about the level of effort required, the knowledge acquired, the expertise and dedication of the teacher, the workshop content, which aspects they considered useful and suggestions for improvement. The first four aspects were evaluated using questions which the students answered on a five-point Likert scale, while open questions were used for the last two. We received mostly positive responses concerning the first four aspects. For example, below we list the questions regarding the workshop content:

Q1: The goals of the workshop were clear.
Q2: The workshop content was organized and planned well.
Q3: The workload was adequate.
Q4: The students could actively participate in the workshop.

As can be seen in Fig. 8, the majority of the students agreed or strongly agreed with each of these statements.

In response to the first open question ("Which aspects of the workshop did you find most useful?"), the students mentioned the following points: its focus on mobile app programming; having learned to design and program applications; having seen how to put concepts in relation while thinking about how to build a program; and having acquired basic programming skills.

The following answers to the second open question ("How would you improve this workshop?") stand out: more hours to be able to complete more exercises and touch on more subjects; and developing more games.

On the other hand, before the workshop we asked the students to answer a quiz featuring 15 very easy questions about basic programming concepts. The students took the quiz again after completing the workshop. As can be seen in Fig. 9, the grades obtained after the workshop (median: 8.0) were better than those obtained initially (median: 6.67).

In order to verify whether the positive variation between the mean results obtained before and after the workshop can be considered significant, we carried out Student's repeated-measures t-test[6] on the grades obtained, obtaining a t value of 9.26 and a p-value of $0.84 \cdot 10^{-11}$; from this we conclude that the observed improvement in grades is significant with a high probability.

We also checked whether the academic results obtained by the students on the five courses taught in the first semester of the first year differed depending on whether the students had benefited from the workshop or not. To this end, we drew on listings of grades the students had earned on these courses and an additional listing combining the results of all courses except *English*. This last listing included only students who had attended all four courses and assigned to each student the average grade earned on them.

For this study we divided the students into two groups: those who had benefited from the workshop and all others (who had either not benefited or not participated). A student was considered to have benefited from the workshop if he/she scored at or above the median grade (an 8) on the basic programming concepts quiz. Out of the 41 students who completed the workshop, 30 (73%) met this condition.

Table II shows the following information for each course, distinguishing between students who benefited from the workshop and all others: the number of students, the median grade,

[5] We used the "Class Evaluation" template in the Google Forms application to evaluate student satisfaction.

[6] We carried out Student's repeated-measures t-test using the `ttest_rel` function from the `stats` module in the SciPy Python library.

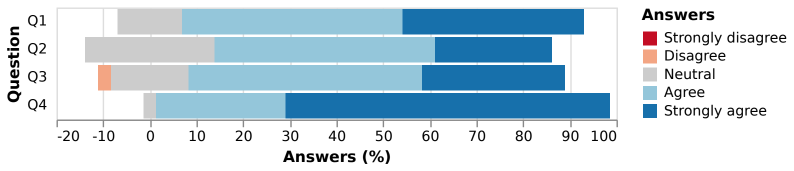


Fig. 8: Percentage of students who selected each of the five possible answers, with neutral responses (in gray) centered at 0, for each of the four questions asked about the workshop content

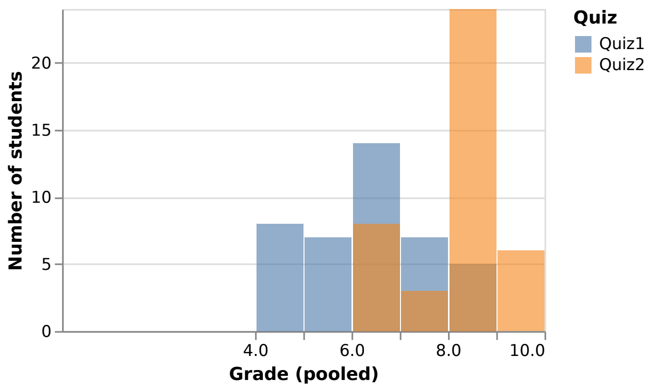


Fig. 9: Results obtained on the basic programming concepts quiz, before and after completing the workshop

the pass rate and the result of the Shapiro-Wilk test for normality[7].

As is evident, the number of students varies slightly across courses depending on how many students attended. It is interesting to note that although around 120 students attended each course, only 83 attended all courses except *English*.

Table II also confirms that the median grades earned by students who benefited from the workshop are higher than those of students who did not. We should point out that we observe a smaller difference between the median grades of the two student groups precisely in the *English* course, on which performance is not expected to be influenced by benefiting from the workshop.

On the other hand, the pass rate among students who have benefited from the workshop is also higher in all subjects except *Computer organization*, where it is slightly lower (86.67% vs. 87.85%). Nevertheless, the pass rate in this course is higher this year than it was last year (76%) and higher than the pass rates on all other courses this year, except for *English*, on which results resemble those of past years. We still regard the workshop as having had a positive impact on the *Computer organization* course, despite the slightly lower pass rate, given the following considerations: i) the pass rate in both groups is above 85%; ii) the median grade of students who have benefited from the workshop is higher than that of students who did not, and iii) in the grade distribution of the two groups, presented below, it can be seen that students who have benefited from the workshop show better results.

[7]We carried out the Shapiro-Wilk test for normality using the `shapiro` function from the `stats` module in the SciPy Python library.

Finally, it is interesting to observe that the combined pass rate in the mathematics and computing courses is 21% higher for the group of students who benefited from the workshop.

To evaluate the grade distributions of each course in more detail, we plotted them as boxes representing the middle two quartiles and bars representing the lowest and highest quartiles (see Fig. 10). It can be seen that for all courses except *English*, the grades of students who benefited from the workshop both start out higher and in their majority remain higher than those of their classmates who did not. This difference in grade distribution is especially significant in the *Computer organization* and *Basic computing* courses, where more than half of the students who benefited from the workshop earned higher grades than three quarters of those who did not.

On the other hand, in order to visualize the percentage of students who earned a specific grade in each course, Fig. 11 shows normalized histograms of the grades earned by students who have benefited from the workshop and those who did not for each course taught in the first semester of the first year, and for the combination of all courses except *English*. On the *Computer organization* and *Basic computing* courses, the greatest percentage difference is found for the high grades: between 8 and 10 for the former; and between 7 and 8, but especially between 9 and 10, for the latter. For *English*, the histograms are very similar for both groups of students. For *Mathematics* and *Programming*, even though percentage differences can be seen in the high grades, there are also spikes in the low grades, although they are more pronounced in *Mathematics* than in *Programming*. Finally, when combining the grades in all courses except *English*, we can see how the histogram representing the students who have benefited from the workshop looks similar to that representing the other students but is shifted towards the right, and further shows high percentage differences for grades between 6 and 8, and between 9 and 10.

To gauge whether the differences between the means of the various groups of grades shown in Fig. 11 are statistically significant, either Student's or Welch's t-test can be used. To make the choice we must first ascertain whether the samples to be compared come from normally distributed populations.

As can be seen in Table II, the only grades passing the Shapiro-Wilk test for normality (i.e., yield a p-value ≥ 0.05) are those for *English* (with p-values of 0.157 and 0.085 for students who benefited from the workshop and those who did not, respectively) and those for all courses except *English* combined (with p-values of 0.423 and 0.746, respectively). In all other cases, p-values lower than 0.05 are found for at least one of the two student groups; thus the null hypothesis—

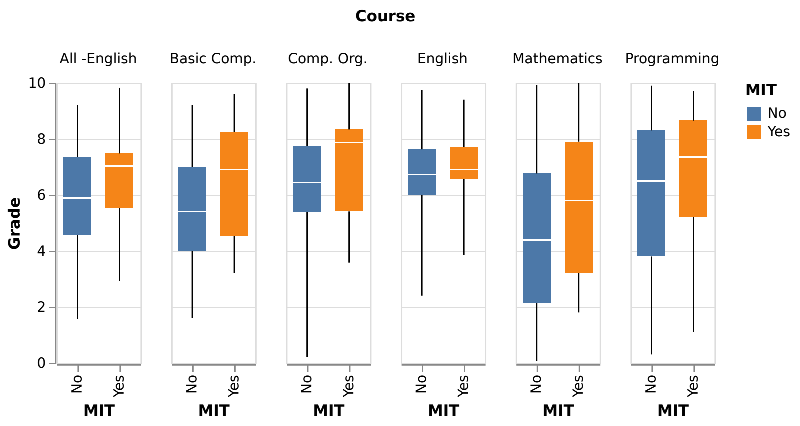
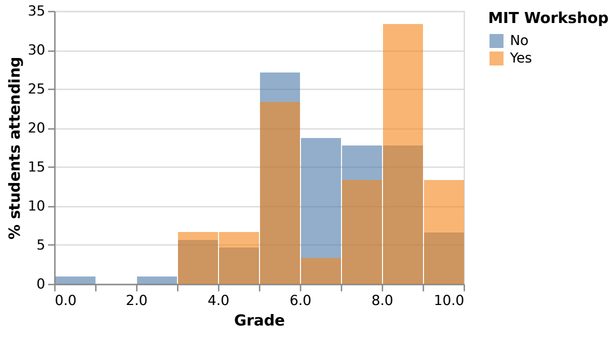
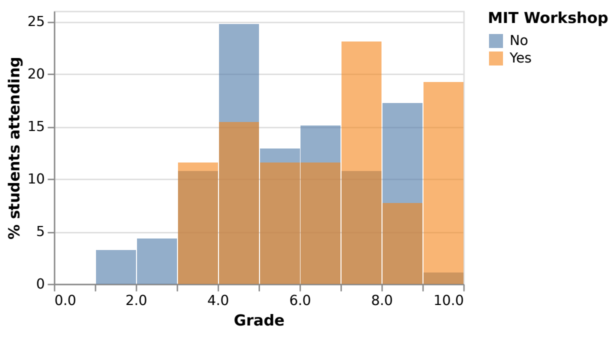


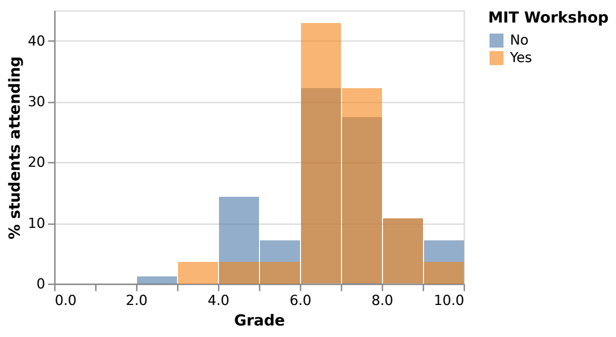
Fig. 10: Grade distribution of students who benefited from the workshop (orange) and those who did not (blue), for each individual course and for all courses combined except *English*



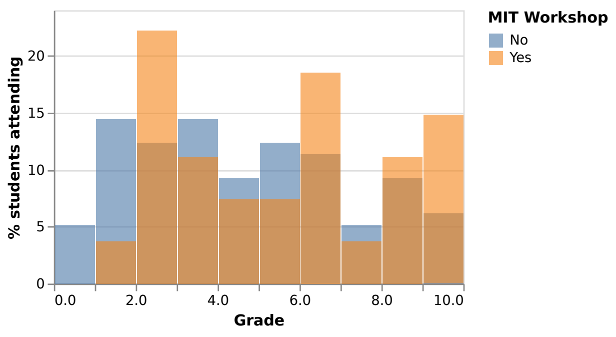
(a) Computer organization



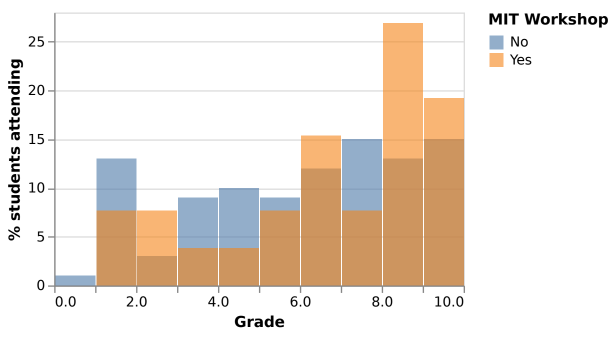
(b) Basic computing



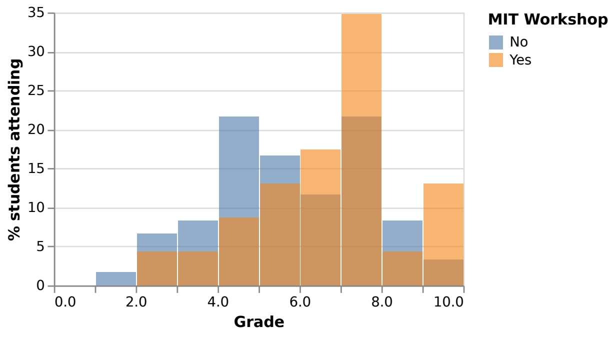
(c) English



(d) Mathematics



(e) Programming



(f) All except English

Fig. 11: Normalized histograms of the grades obtained by students who have benefited from the workshop (orange) and all other students (blue) on various courses taught in the first semester of the first year

TABLE II
NUMBER OF STUDENTS, MEDIAN GRADE AND ITS 95% CONFIDENCE INTERVAL, PASS RATE AND SHAPIRO-WILK TEST RESULTS FOR EACH GRADE LISTING

Course	MIT workshop	N	Median	95% CI	Dif	% Passed	Dif	Shapiro-Wilk W	Shapiro-Wilk p-value
Computer organization	Yes	30	7.87	[7.22, 8.52]	1.43	86.67	−1.18	0.98	0.053
	No	107	6.44	[6.11, 6.77]		87.85		0.92	0.020
Basic computing	Yes	26	6.90	[6.11, 7.69]	1.50	73.08	16.09	0.96	0.003
	No	93	5.40	[5.00, 5.80]		56.99		0.94	0.120
English	Yes	28	6.90	[6.48, 7.32]	0.18	92.86	8.34	0.98	0.157
	No	84	6.72	[6.42, 7.03]		84.52		0.94	0.085
Mathematics	Yes	27	5.80	[4.78, 6.81]	1.41	55.56	11.23	0.96	0.005
	No	97	4.39	[3.86, 4.92]		44.33		0.93	0.056
Programming	Yes	26	7.35	[6.34, 8.36]	0.85	76.92	12.92	0.93	0.000
	No	100	6.50	[5.96, 7.04]		64.00		0.90	0.014
All computer science courses	Yes	24	7.30	[6.51, 8.10]	0.83	79.17	12.99	0.95	0.012
	No	68	6.47	[6.01, 6.93]		66.18		0.93	0.086
All except English	Yes	23	7.04	[6.32, 7.76]	1.15	82.61	20.94	0.98	0.423
	No	60	5.88	[5.43, 6.34]		61.67		0.97	0.746

which assumes that the samples fit a normal distribution—has to be rejected (in other words, the samples are probably not normally distributed).

Once we have verified that the *English* grades as well as the grade averages over all other courses represent normally distributed populations, the next step is to check whether the variances of the populations corresponding to students benefited from the workshop and all other students are equal. If the variances are equal, Student's t-test should be used; otherwise, Welch's t-test is the appropriate one.

Applying the Levene test[8] to the *English* results of students who benefited from the workshop and those who did not yield a W-statistic of 2.55 and a p-value of 0.12 (> 0.05); thus it can be assumed that the population variances are similar. Applying the same test to the grade averages over all courses except *English* earned by students who benefited from the workshop and those who did not yield a W-statistic of 0.56 and a p-value of 0.46 (> 0.05); thus also in this case population variances can be assumed to be similar.

Applying Student's t-test[9] to the relevant grades, i.e., those earned in *English* and in the combination of all other courses, yields the results shown in Table III. In applying the above threshold of significance ($\alpha = 0.05$) we can conclude the following: i) for the *English* course (p-value $= 0.187 > 0.05$) there is no statistically significant difference between workshop participants and other students, as expected; and ii) for the combination of all other courses (p-value $= 0.038 < 0.05$ and t $= 1.8 > 0$) we can reject the null hypothesis and accept the alternative hypothesis, which states that the grades earned by students who benefited from the workshop are significantly higher than those earned by the other students.

We should nevertheless remember that the students themselves chose to participate in the workshop. Thus we have to bear in mind that the higher grades earned by students who benefited from the workshop as compared to their classmates might in truth be owing to other factors not investigated in this study.

[8]We carried out the Levene test using the `levene` function from the `stats` module in the SciPy Python library.

[9]We carried out Student's t-test for independent samples using the `ttest_ind` function (from the `stats` module in the SciPy Python library) indicating that the population variances are equal.

TABLE III
RESULTS OF STUDENT'S T-TEST FOR GRADES IN ENGLISH AND THE COMBINED GRADES IN ALL COURSES EXCEPT ENGLISH, AND THE 95% CONFIDENCE INTERVAL OF THE DIFFERENCE BETWEEN THE MEDIAN GRADES OF WORKSHOP PARTICIPANTS AND ALL OTHER STUDENTS

Course	Student's t-test t	Student's t-test p-value	95% CI
English	0.89	0.187	[−0.32, 0.84]
All except *English*	1.80	0.038	[−0.08, 1.67]

VII. CONCLUSIONS AND FUTURE WORK

We have created a mobile application programming workshop using MIT App Inventor for first year undergraduate students with the objective of helping students develop a set of basic programming skills that improve their learning process, especially on the *Computer organization* course. The workshop follows the general philosophy of the MIT App Inventor tutorials, which is to allow students to see tangible results of their work as soon as possible. To this end we propose five projects in increasing order of difficulty, organized into a sequence of steps, which allows the necessary skills to be acquired and tested in a gradual way.

The workshop was favorably received by the students. Almost all students who enrolled completed the workshop and evaluations were very positive. In addition, the students were presented with a quiz about basic programming concepts before and after the workshop, and the results show significant improvements in knowledge.

On the other hand, when comparing the grades earned in our first-year courses by students who have benefited from the workshop with those earned by all other students, we observed a greater percentage of students from the former group obtaining higher grades, especially in programming-related courses.

In terms of future work, we intend to modify the workshop by incorporating some of the changes recommended by the students: better allocate the time dedicated to the various projects; substitute one of the current projects with a game; and succinctly explain the resources offered by MIT App Inventor at the beginning of the workshop. We will also take advantage of the new editions to gather more information about its effect on the academic performance of its participants.

Acknowledgments

First of all we would like to thank the developers of MIT App Inventor and the creative community that has taken shape around this tool. We consider it an excellent platform for STEM dissemination, which we think will benefit our society.

We would also like to thank our colleagues who kindly made available the grades of the courses for which they are responsible: Juan Carlos Amengual Argudo, Fernando Javier Hernando Carrillo, José Luis Llopis Borrás, Ana María Lluch Peris, Marina Murillo Arcila, and María Luisa Renau Renau.

Finally, we would also like to acknowledge the support received from the Engineering and Computer Science Department of Jaume I University.

References

[1] S. Barrachina, G. Fabregat, C. Fernández, and G. León, "Utilizando ARMSim y QtARMSim para la docencia de Arquitectura de Computadores," *ReVisión*, vol. 8, no. 3, p. 2, 2015.

[2] S. Barrachina, G. Fabregat, and J. V. Martí, "Utilizando Arduino DUE en la docencia de la entrada/salida," in *Actas de las XXI Jornadas de la Enseñanza Universitaria de la Informática*. Universitat Oberta La Salle, 2015, pp. 58–65.

[3] X. Canaleta, F. Sánchez, I. Jacob, Á. Velázquez, and M. Marques, "Declaración AENUI-CODDII por la inclusión de asignaturas específicas de ciencia y tecnología informática en los estudios básicos de la enseñanza secundaria y bachillerato," in *Actas de las XX Jornadas sobre Enseñanza Universitaria de la Informática*, july 2014, pp. 229–236.

[4] S. Barrachina Mir and G. Fabregat Llueca, "¿Puedo programar mi móvil? Pero si acabo de llegar," *Actas de las Jornadas sobre Enseñanza Universitaria de la Informática*, vol. 4, 2019.

[5] J. Maloney, M. Resnick, N. Rusk, B. Silverman, and E. Eastmond, "The Scratch programming language and environment," *ACM Transactions on Computing Education (TOCE)*, vol. 10, no. 4, pp. 16:1–16:15, 2010.

[6] D. Topalli and N. E. Cagiltay, "Improving programming skills in engineering education through problem-based game projects with Scratch," *Computers & Education*, vol. 120, pp. 64–74, 2018.

[10] S. Trilles and C. Granell, "SUCRE4Kids: El fomento del pensamiento computacional a través de la interacción social y tangible," *Actas de las XXIV Jornadas sobre Enseñanza Universitaria de la Informática*, vol. 3, no. 0, pp. 303–310, 2018.

[7] D. Ozoran, N. Cagiltay, and D. Topalli, "Using Scratch in introduction to programming course for engineering students," in *2nd International Engineering Education Conference (IEEC2012)*, vol. 2, 2012, pp. 125–132.

[8] S. P. Roche and N. M. Martínez, "Evaluación de entornos de programación para el aprendizaje," in *Actas de las XVII Jornadas sobre Enseñanza Universitaria de la Informática*, july 2011, pp. 83–90.

[9] R. Muñoz, T. S. Barcelos, R. Villarroel, M. Barría, C. Becerra, R. Noel, and I. Frango Silveira, "Uso de Scratch y Lego Mindstorms como apoyo a la docencia en fundamentos de programación," in *Actas de las XXI Jornadas de la Enseñanza Universitaria de la Informática*. Universitat Oberta La Salle, 2015, pp. 248–254.

[11] S. Papadakis, M. Kalogiannakis, V. Orfanakis, and N. Zaranis, "Novice programming environments. Scratch & App Inventor: a first comparison," in *Proceedings of the 2014 Workshop on Interaction Design in Educational Environments*. ACM, 2014, pp. 1–7.

[12] S. A. Nikou and A. A. Economides, "Transition in student motivation during a Scratch and an App Inventor course," in *2014 IEEE Global Engineering Education Conference (EDUCON)*. IEEE, 2014, pp. 1042–1045.

[13] D. Wolber, "App Inventor and real-world motivation," in *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education*. ACM, 2011, pp. 601–606.

Sergio Barrachina Mir graduated in Telecommunications Engineering (major in Electronics) from the Technical University of Valencia in 1995 and earned his PhD in Computer Engineering at Jaume I University in 2003, where he is Associate Professor in Computer Architecture and Technology since 2012.

He has been teaching mainly first- and second-year courses of the former Computing degree and the current Computer Engineering and Computational Mathematics degrees.

He is member of the High Performance Computing & Architectures (HPC&A) research group, with which he has participated in numerous projects related to high-performance computing and architectures.

Germán Fabregat Llueca graduated in Physics (major in Electricity, Electronics and Computing) from Valencia University in 1989 and earned his PhD in the same discipline in 1996. He has been teaching at Jaume I University since 1991 and is Associate Professor in Computer Architecture and Technology since 2001.

His work focuses on fault tolerance, embedded systems and sensor networks, with a special interest in industrial applications. His teaching work additionally features the continuous development of teaching support applications as well as practice equipment, simulators, systems programming environments, etc.