Actas de las Jenui, vol. 4. 2019.
Páginas: 39-46

# ¿Puedo programar mi móvil? Pero si acabo de llegar

Sergio Barrachina Mir, Germán Fabregat Llueca
Departamento de Ingeniería y Ciencia de los Computadores
Universitat Jaume I
12071 Castellón de la Plana
barrachi@uji.es, fabregat@uji.es

## Resumen

Por sorprendente que parezca, cada vez que preguntamos a nuestros estudiantes recién matriculados en el Grado en Ingeniería Informática si han programado alguna vez, la respuesta mayoritaria es que no. Los pocos que han estudiado formación profesional en informática suelen tener alguna noción, pero la mayor parte de los que han estudiado bachillerato, ninguna.

Esta falta de competencias básicas de programación supone una desventaja en aquellas asignaturas relacionadas con esta materia. En nuestro grado, esta desventaja es especialmente evidente en la asignatura Estructura de computadores, de primer curso y primer semestre que, sin ser una asignatura de programación al uso, tiene por objeto que el estudiante adquiera competencias relacionadas con la arquitectura de un computador y, por tanto, con la programación en lenguaje ensamblador.

Para suplir esta falta de base, se ha impartido un taller de programación para móviles con MIT App Inventor. Este taller ha tenido una gran aceptación, ha sido muy bien valorado por los estudiantes y consideramos que ha contribuido a mejorar los resultados de Estructura de computadores.

## Abstract

Surprisingly, every time we ask the newly enrolled students in the Degree in Computer Engineering whether they have ever programmed, the majority answer is no. The few that have done a computer science vocational training module usually have some notion, but most of those who have done high school, none.

This lack of basic programming skills is a disadvantage in those courses related to this matter. In our degree, this disadvantage is especially evident in the Computers Structure course, taught on the first year at the first semester. Although it is not a usual programming course, it requires the student to acquire skills related to computer architecture, and, therefore, to programming in assembly language.

To address this lack of previous knowledge, a workshop on mobile programming has been taught using MIT App Inventor. This workshop has had a great acceptance, has been very well evaluated by the students, and we believe that has contributed to improve their results on the Computers Structure course.

## Palabras clave

Primer curso, Conceptos básicos de programación, Programación de móviles, MIT App Inventor

## 1. Motivación

La asignatura Estructura de computadores de primer curso, primer semestre, se imparte en los grados en Ingeniería Informática y Matemática Computacional de la Universitat Jaume I. Esta asignatura no exige conocimientos previos de programación. De hecho, teniendo en cuenta esta situación, en cursos previos se replanteó la asignatura y se desarrollaron dos recursos docentes [1, 2] orientados a facilitar y motivar su estudio. Aunque estos cambios han mejorado sus resultados, los estudiantes sin unas competencias básicas de programación, aún tienen que realizar un mayor esfuerzo para seguirla satisfactoriamente. Esto es debido a que la asignatura detalla el funcionamiento a bajo nivel del computador, es decir, muestra cómo un computador es capaz de ejecutar programas en código máquina. Por tanto, el estudiante sin unos conocimientos básicos de programación —tipos de datos, estructuras de datos, estructuras de control, codificación, etc.—, no podrá apoyarse en esta base para estudiar los mecanismos hardware que hacen posible su aplicación.

Por otro lado, y pese a las declaraciones realizadas desde AENUI-CODDII [3] en favor de la inclusión de asignaturas específicas de ciencia y tecnología informática en los estudios básicos de la enseñanza secundaria y bachillerato, las asignaturas de informática siguen siendo actualmente optativas en estas enseñanzas y, además, solo una parte de estas asignaturas se centra

en la programación. Esto conlleva que en la práctica, muchos de los estudiantes recién ingresados en el grado en Ingeniería Informática, pese a haber optado por esta titulación, no posean competencias básicas de programación.

Así pues, y con el objeto de paliar esta situación, se ha ofertado un taller intensivo a los estudiantes de primero, justo al comienzo del semestre, con el objetivo de que puedan desarrollar una serie de competencias básicas de programación de una forma práctica e intuitiva, orientadas especialmente a mejorar sus resultados en la asignatura Estructura de computadores.

## 2. Objetivo docente

El objetivo docente del taller propuesto es mejorar los resultados académicos de los estudiantes de primer curso, especialmente en la asignatura Estructura de computadores. Como ya se ha comentado, al impartirse esta asignatura en el primer semestre, e introducir aquellos conceptos de arquitectura de computadores que permiten la ejecución de programas, depende especialmente de que los estudiantes posean competencias básicas en programación.

## 3. Selección del entorno de programación

Puesto que el objetivo del taller es que los estudiantes adquieran competencias básicas de programación al principio del curso de una forma práctica e intuitiva, y en unas pocas sesiones, es sumamente importante seleccionar correctamente el entorno de programación.

Para introducir rápidamente y de una forma gráfica los conceptos básicos de programación, la primera decisión que se tomó al respecto fue la de recurrir a un entorno visual de programación. De entre estos entornos, evaluamos utilizar Scratch o MIT App Inventor.

Scratch [4] es de sobra conocido y, pese a estar orientado principalmente a usuarios entre 8 y 16 años, su uso como herramienta de introducción a la programación es valorada por encima de la de otros entornos visuales por estudiantes universitarios [8]. Además, se ha utilizado con éxito tanto en etapas previas, como para iniciar a la programación en la propia universidad, extendiéndolo en ocasiones para la programación de dispositivos hardware: robots *Lego Mindstorms NXT* [5], hardware ad-hoc del proyecto educativo SUCRE4Kids [9]...

MIT App Inventor[1], por su parte, es un entorno visual de programación, que presenta una interfaz muy similar a la de Scratch, pero que está orientado al desarrollo intuitivo de aplicaciones totalmente funcionales para móviles y tabletas. Es decir, se programa con la misma facilidad que con Scratch, pero además, los estudiantes obtienen un producto tangible: una aplicación para su móvil;[2] que pueden mostrar e instalar en los móviles de sus amigos y familiares, e incluso publicar en la tienda en línea Google Play. Esta orientación a la programación móvil, al desarrollo de un producto real que se puede mostrar, supone una motivación importante, y habitualmente ha inclinado la balanza en favor de MIT App inventor, frente a Scratch, como un entorno de introducción a la programación para estudiantes de grado [7, 6, 10].

En el caso que nos ocupa, desarrollar competencias básicas de programación que sean de utilidad para la asignatura Estructura de computadores, MIT App Inventor posee una ventaja añadida: permite acceder a los sensores del móvil (cámara, temporizador, GPS...), por lo que se puede también utilizar para introducir fácilmente y de una forma aplicada, conceptos de entrada/salida que posteriormente se desarrollarán en la asignatura.

Por todo lo anterior, se decidió realizar el taller usando la plataforma MIT App Inventor.

## 4. Programación móvil con MIT App Inventor

El taller propuesto, Programación móvil con MIT App Inventor, está disponible en la siguiente dirección web para cualquier profesor que esté interesado en ofertarlo o que quiera recomendarlo a sus estudiantes como herramienta de auto-aprendizaje:
`http://lorca.act.uji.es/curso/mit-app-inventor/`

Está orientado a estudiantes recién ingresados en el grado en Ingeniería Informática, especialmente a aquellos que no hayan recibido formación previa en programación, y una vez completado, el estudiante debería ser capaz de:

- Diferenciar entre los conceptos de valor y variable.
- Reconocer y saber para qué se pueden utilizar las diferentes estructuras condicionales.
- Reconocer y saber para qué se pueden utilizar las diferentes estructuras iterativas.
- Reconocer algunos de los objetos proporcionados por *MIT App Inventor*, acceder a sus propiedades y llamar a sus funciones.

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

[2] Actualmente MIT App Inventor permite realizar aplicaciones solo para Android, pero está previsto soportar también a iOS. En cualquier caso, ya existen entornos basados en MIT App Inventor que sí permiten publicar aplicaciones también para iOS, p.e., Thunkable.

- Diferenciar entre propiedades y funciones de un objeto.
- Encapsular partes de un código utilizando funciones.
- Interactuar con la entrada/salida de un dispositivo móvil (cámara, acelerómetro, GPS, NFC...).
- Diseñar interfaces de usuario simples.
- Realizar programas que se ejecuten como respuesta a un evento.
- Desarrollar aplicaciones sencillas para móviles utilizando *MIT App Inventor*.

La duración recomendada para la realización del taller es de 10 horas, repartidas en 4 sesiones de 2 horas y media cada una.

Siguiendo las recomendaciones para cursos introductorios de MIT App Inventor,[3] en la primera sesión se presenta el entorno de programación, se accede a la aplicación, se configuran correctamente los móviles de los estudiantes y se realizan de una forma guiada, los cuatro tutoriales básicos proporcionados por MIT App Inventor, que desarrollan de forma gradual: I) una aplicación que habla, II) una extensión de la anterior aplicación que habla cuando se agita el móvil, III) una aplicación que permite mover una bola en pantalla utilizando el dedo, y IV) una aplicación de dibujo que permite tomar fotos.

Para las siguientes sesiones, en lugar de recurrir a algunos de los tutoriales del repositorio de MIT App Inventor,[4] se optó por desarrollar proyectos propios que persiguieran los objetivos formativos del taller e introdujeran conceptos de arquitectura de computadores.

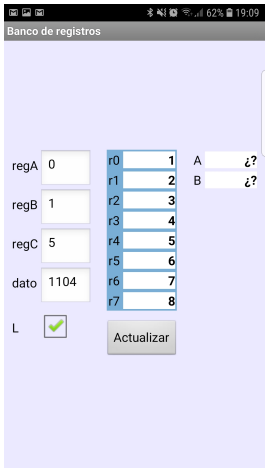



Figura 1: Captura de pantalla de la aplicación móvil desarrollada en el proyecto «Simulador de un banco de registros»

En concreto, se han creado los siguientes tutoriales:

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

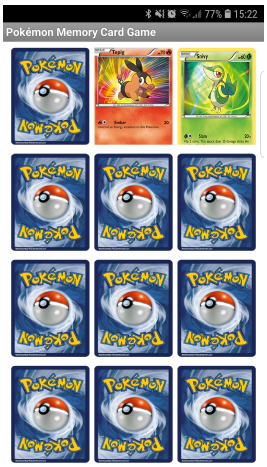



Figura 2: Captura de pantalla de la aplicación móvil desarrollada en el proyecto «Juego de memorización»

**Simulador de una ALU** Consiste en el desarrollo de una aplicación móvil que simula una Unidad Aritmético Lógica (ALU) capaz de realizar una operación seleccionable de entre cuatro: suma, resta, multiplicación o división, sobre dos operandos numéricos y mostrar el resultado obtenido.

**Simulador de un banco de registros** Se propone el desarrollo de un simulador gráfico de un banco de 8 registros, que permite la lectura simultánea de dos de sus registros y la escritura de un valor numérico en uno de ellos. Al final de este proyecto, se propone como ampliación añadir el simulador de ALU anterior para operar con el contenido de los registros.

**rgbFoto** Muestra el desarrollo de una aplicación móvil que permite seleccionar un color mediante sus componentes RGBA, tomar una foto y virarla al color previamente seleccionado.

**Juego de memorización** Consiste en el desarrollo de un juego de memorización en el que hay que localizar 6 parejas de cartas pudiendo levantar como máximo dos cada vez.

**El Tiempo** Proyecto en el que se desarrolla una aplicación que obtiene y visualiza datos meteorológicos de la Agencia Estatal de Meteorología (AEMET).

A modo de ejemplo, las Figuras 1 y 2 muestran las capturas de pantalla de las aplicaciones realizadas tras completar los proyectos «Simulador de un banco de registros» y «Juego de memorización», respectivamente.

Estos tutoriales siguen la misma estructura que los tutoriales básicos de MIT App Inventor, comienzan describiendo la aplicación que se quiere realizar y los elementos de MIT App Inventor que se van a utilizar. A continuación, proponen el desarrollo de la aplicación de tal forma que el estudiante comience realizando una

primera versión de la aplicación en el menor tiempo posible y mediante sucesivos pasos vaya refinándola hasta la versión final. En cada paso se proporciona un enlace con su solución, por si algún estudiante tuviera algún problema con su versión. Además, en muchos de los pasos se proponen posibles extensiones, para que los estudiantes que vayan más adelantados puedan realizarlas.

# 5. Resultados

El taller se ofertó a los 164 estudiantes matriculados en la asignatura Estructura de computadores, especificando que estaba especialmente indicado para aquellos estudiantes que no tuvieran conocimientos de programación. Se matricularon 46 estudiantes, y lo completaron 42 (el 91 %).

En cuanto a la valoración del taller por parte de los estudiantes, se les pasó un formulario estándar de satisfacción[5] en el que se les preguntaba sobre el nivel de esfuerzo requerido, los conocimientos adquiridos, las habilidades y dedicación del profesor, el contenido del taller, que aspectos consideraban útiles del taller y sugerencias de mejora.

Los cuatro primeros aspectos se valoraron por medio de preguntas que los estudiantes contestaban utilizando una escala tipo Likert de cinco puntos y los dos últimos mediante preguntas de respuesta abierta. En los cuatro primeros aspectos se obtuvieron respuestas mayoritariamente positivas. A modo de muestra, se detallan a continuación las preguntas correspondientes al contenido del taller:

- P1: Los objetivos del taller estaban claros.
- P2: El contenido del taller estaba bien organizado y planificado.
- P3: La carga de trabajo del taller fue la adecuada.
- P4: Los alumnos pudieron participar activamente en el taller.

Como se puede observar en la Figura 3, la mayor parte de los estudiantes estaba de acuerdo o totalmente de acuerdo con cada una de estas preguntas.

En cuanto a la primera de las preguntas de respuesta abierta, ¿qué aspectos de este taller te resultaron más útiles?, los estudiantes han incidido en: que esté orientada a programar para móviles, el haber aprendido a diseñar y a programar, el haber visto cómo relacionar conceptos mientras piensan cómo construir un programa, y el haber adquirido conocimientos básicos de programación.

De la segunda pregunta ¿cómo mejorarías este taller?, conviene destacar las siguientes respuestas: más horas para poder realizar más prácticas y para tocar más temas, y desarrollar más juegos.

Por otro lado, antes de comenzar el taller se pasó un cuestionario con 15 preguntas tipo test muy sencillas sobre conceptos básicos de programación. Este cuestionario se volvió a pasar una vez completado el taller. Como se puede observar en la Figura 4, las notas obtenidas después del taller (mediana: $8{,}0$) fueron mejores que las obtenidas al principio (mediana: $6{,}67$).

Para comprobar si la variación positiva entre las medias de los resultados obtenidos antes y después del taller se puede considerar significativa, se ha realizado la prueba T de Student para muestras relacionadas[6] sobre estas calificaciones, obteniendo un t-estadístico de $9{,}26$ y un p-valor de $0{,}84 \cdot 10^{-11}$, por lo que se puede concluir que es muy probable que la valoración positiva que se ha constatado sea significativa.

A continuación se comparan los resultados académicos obtenidos por los estudiantes en las cinco asignaturas de primer curso, primer semestre, en función de si han aprovechado el taller o no. Para ello, se ha partido de los listados de calificaciones de estas asignaturas y de un listado adicional que combina los resultados de las cuatro asignaturas que no son Inglés. En este último listado, formado únicamente por los estudiantes que se han presentado a estas cuatro asignaturas, se ha calculado la nota de cada estudiante como la media de sus notas en esas asignaturas.

Para realizar este estudio, se han dividido los estudiantes en dos grupos: los estudiantes que han aprovechado el taller y el resto (los que no lo han aprovechado o no lo han hecho). Se ha considerado que un estudiante ha aprovechado el taller si en el cuestionario básico de programación, ha obtenido una nota igual o superior a la mediana de las notas de ese cuestionario (un 8). De los 41 estudiantes que han completado el taller, 30 cumplen esta condición (el 73 %).

El Cuadro 1 muestra para cada listado de notas, distinguiendo entre los estudiantes que han aprovechado el taller y el resto: el número de estudiantes, la mediana de las notas, el porcentaje de aprobados sobre presentados y los resultados del test de normalidad de Shapiro-Wilk[7].

Como se puede observar, el número de estudiantes varía ligeramente en cada asignatura en función de cuántos se han presentado. Es interesante destacar que aunque se han presentado alrededor de 120 estudiantes a cada una de las asignaturas, solo 83 estudiantes se han presentado a las cuatro que no son Inglés.

En el Cuadro 1 también se puede constatar que las

[5] Como formulario de satisfacción se utilizó la plantilla «Valoración del curso» de la aplicación de formularios de Google.

[6] Para realizar la prueba T de Student para muestras relacionadas, se ha utilizado la función `ttest_rel` del módulo `stats` de la biblioteca SciPy de Python.

[7] Para realizar la prueba de normalidad de Shapiro-Wilk se ha utilizado la función `shapiro` del módulo `stats` de la biblioteca SciPy de Python.

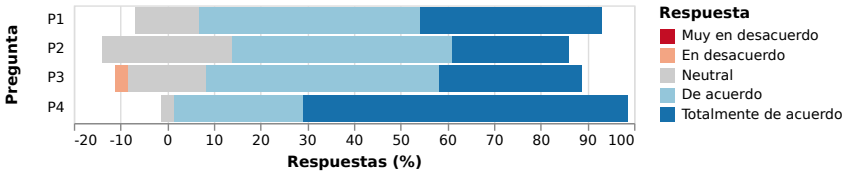



Figura 3: Tanto por ciento de estudiantes que ha seleccionado cada una de las cinco posibles respuestas, centrando las respuestas neutrales —en gris— sobre el 0, para cada una de las cuatro preguntas realizadas sobre los contenidos del taller

| Asignatura | Taller MIT | N | Mediana | Dif | Aprobados | Dif | Shapiro-Wilk W | Shapiro-Wilk p-valor |
|---|---|---|---|---|---|---|---|---|
| Estructura de computadores | Sí | 30 | 7,87 | 1,43 | 86,67 | −1,18 | 0,98 | 0,053 |
| | No | 107 | 6,44 | | 87,85 | | 0,92 | 0,020 |
| Informática básica | Sí | 26 | 6,90 | 1,50 | 73,08 | 16,09 | 0,96 | 0,003 |
| | No | 93 | 5,40 | | 56,99 | | 0,94 | 0,120 |
| Inglés | Sí | 28 | 6,90 | 0,18 | 92,86 | 8,34 | 0,98 | 0,157 |
| | No | 84 | 6,72 | | 84,52 | | 0,94 | 0,085 |
| Matemáticas | Sí | 27 | 5,80 | 1,41 | 55,56 | 11,23 | 0,96 | 0,005 |
| | No | 97 | 4,39 | | 44,33 | | 0,93 | 0,056 |
| Programación | Sí | 26 | 7,35 | 0,85 | 76,92 | 12,92 | 0,93 | 0,000 |
| | No | 100 | 6,50 | | 64,00 | | 0,90 | 0,014 |
| Todas salvo «Inglés» | Sí | 23 | 7,04 | 1,15 | 82,61 | 20,94 | 0,98 | 0,423 |
| | No | 60 | 5,88 | | 61,67 | | 0,97 | 0,746 |

Cuadro 1: Número de estudiantes, mediana de las notas, porcentaje de aprobados sobre presentados y resultados del test Shapiro-Wilk para cada uno de los listados de calificaciones

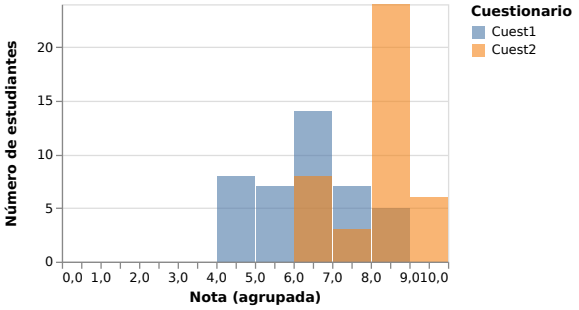



Figura 4: Resultados del cuestionario sobre conocimientos básicos de programación, antes y después de realizar el taller

medianas de las calificaciones obtenidas por los estudiantes que han aprovechado el taller son superiores a las de los que no lo han hecho. Conviene señalar que justamente en la asignatura Inglés, que no debería verse influida por si se ha aprovechado o no el taller, es donde se observa una menor diferencia entre las medianas de los dos grupos de estudiantes.

Por otro lado, los tantos por ciento de aprobados de los estudiantes que han aprovechado el taller también son mejores, salvo en el caso de la asignatura Estructura de computadores, que es ligeramente inferior (86,67 % frente a 87,85 %). No obstante, los porcentajes de aprobados de Estructura de computadores han sido este curso más elevados que los del curso pasado (que fueron del 76 %) y que los del resto de asignaturas de este curso, a excepción de Inglés, que habitualmente suele obtener resultados similares. Por último, es interesante observar que el tanto por ciento de aprobados sobre presentados que se obtiene al combinar las calificaciones de los estudiantes presentados a las asignaturas de matemáticas e informática es un 21 % más alto para el grupo de los estudiantes que han aprovechado el taller.

Para poder valorar con más detalle cómo se han distribuido las calificaciones de cada asignatura, estas se han representado por medio de gráficas de cajas y bigotes (véase la Figura 5), que agrupan las calificaciones por sus cuartiles. Como se puede ver, en todos los casos, salvo para Inglés, los estudiantes que han aprovechado el taller parten de notas más altas y obtienen mayoritariamente calificaciones más altas que sus compañeros que no lo han hecho. Esta diferencia en cómo se han distribuido las calificaciones es especialmente sig-

nificativa en las asignaturas Estructura de computadores e Informática básica, donde más de la mitad de los estudiantes que han aprovechado el taller obtienen mejores notas que las tres cuartas partes de los estudiantes que no lo han hecho.

Por otro lado, para visualizar el tanto por ciento de estudiantes que ha obtenido una determinada calificación en cada asignatura, en la Figura 6 se muestran los histogramas normalizados de las calificaciones obtenidas por los estudiantes que han aprovechado el taller y los que no, para todas las asignaturas de primer curso, primer semestre, y para la combinación de todas estas asignaturas salvo Inglés. En el caso de las asignaturas Estructura de computadores e Informática básica, la mayor diferencia porcentual se concentra en las calificaciones altas: entre 8 y 10 para la primera; y entre 7 y 8, pero especialmente, entre 9 y 10, para la segunda. En el caso de Inglés, se observan dos histogramas muy similares para ambos grupos de estudiantes. Para Matemáticas y Programación, pese a que se observan diferencias porcentuales en notas altas, también hay picos en notas bajas, aunque más acusados en Matemáticas que en Programación. Por último, al combinar las notas de las asignaturas que no son Inglés, se puede observar cómo el histograma correspondiente a los que han aprovechado el taller es similar al principio al de los que no, pero desplazado a la derecha, presentando, además, diferencias porcentuales elevadas en las calificaciones entre 6 y 8 y entre 9 y 10.

Para poder evaluar mediante pruebas estadísticas si la diferencia entre las medias de los distintos grupos de calificaciones que se muestran en la Figura 6 son estadísticamente significativas, es necesario comprobar previamente si las muestras que se quieren comparar provienen de poblaciones normalmente distribuidas. Esta comprobación debe realizarse puesto que tanto la prueba T de Student como la prueba T de Welch requieren que las dos poblaciones que se vayan a comparar estén normalmente distribuidas.

Como se puede observar en el Cuadro 1, las únicas calificaciones que superan el test de normalidad de Shapiro-Wilk (esto es, que obtienen un p-valor $\geq 0{,}05$) son las de la asignatura Inglés (con p-valores de $0{,}157$ y $0{,}085$, para los que han aprovechado el taller y los que no, respectivamente), y las que combinan los resultados de las asignaturas que no son Inglés (con p-valores de $0{,}423$ y $0{,}746$, respectivamente). En los otros casos, en al menos uno de los dos grupos de estudiantes, se obtienen p-valores inferiores a $0{,}05$, por lo que se debe rechazar la hipótesis nula de que esas muestras correspondan a una distribución normal (y por tanto, se deba aceptar que probablemente no estén normalmente distribuidas).

Aplicando la prueba T de Welch[8] sobre las calificaciones a las que puede aplicarse: las de Inglés y las calificaciones agregadas de todas las asignaturas salvo Inglés, se han obtenido los resultados que se muestran en el Cuadro 2. Aplicando el nivel de significación anterior, $\alpha = 0{,}05$, se puede concluir: I) que en el caso de la asignatura Inglés (p-valor $= 0{,}165 > 0{,}05$), no hay una diferencia estadísticamente significativa entre haber realizado el taller o no, que era lo esperado; y II) que en el caso de las calificaciones agrupadas (p-valor $= 0{,}038 < 0{,}05$ y t-estadístico $= 1{,}83 > 0$), es posible rechazar la hipótesis nula y aceptar la hipótesis alternativa de que las calificaciones obtenidas si se aprovecha el taller son significativamente diferentes (y mayores) a las que se obtienen en caso contrario.

| Asignatura | Prueba T de Welch t-estadístico | p-valor |
|---|---|---|
| Inglés | 1,00 | 0,161 |
| Todas salvo Inglés | 1,83 | 0,038 |

Cuadro 2: Resultados de la prueba T de Welch sobre las calificaciones de Inglés y las calificaciones agregadas de todas las asignaturas salvo Inglés

# 6. Conclusiones y trabajo futuro

Se ha impartido un taller de programación móvil con MIT App Inventor orientado a estudiantes de primer curso para que fueran capaces de desarrollar un conjunto de competencias básicas en programación que mejoraran sus resultados académicos, especialmente en la asignatura Estructura de computadores.

El taller ha tenido una buena acogida entre los estudiantes. La gran mayoría de los inscritos lo han completado y lo han valorado muy positivamente. Además, resultados obtenidos en un cuestionario sobre conocimientos básicos de programación realizado al principio y al final del taller, evidencian una mejora significativa.

Comparando las calificaciones obtenidas en las asignaturas de primer curso por los estudiantes que han aprovechado el taller con las del resto de estudiantes, se puede apreciar un aumento en el porcentaje de estudiantes que han obtenido una calificación más alta, especialmente en las asignaturas más relacionadas con programación.

Como trabajo futuro se pretende reeditar el taller siguiendo las recomendaciones realizadas por los estudiantes y obtener más información sobre el efecto del taller sobre su desempeño académico.

[8] Para realizar la prueba T de Welch para muestras independientes, se ha utilizado la función `ttest_ind` indicando que las varianzas de las dos muestras no tienen por qué ser similares, del módulo `stats` de la biblioteca SciPy de Python.

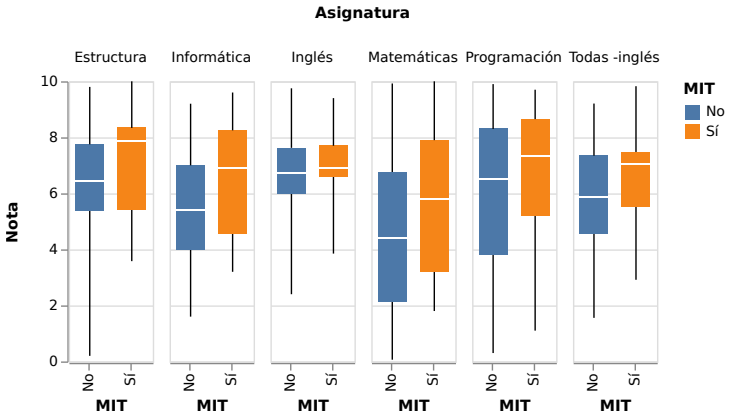
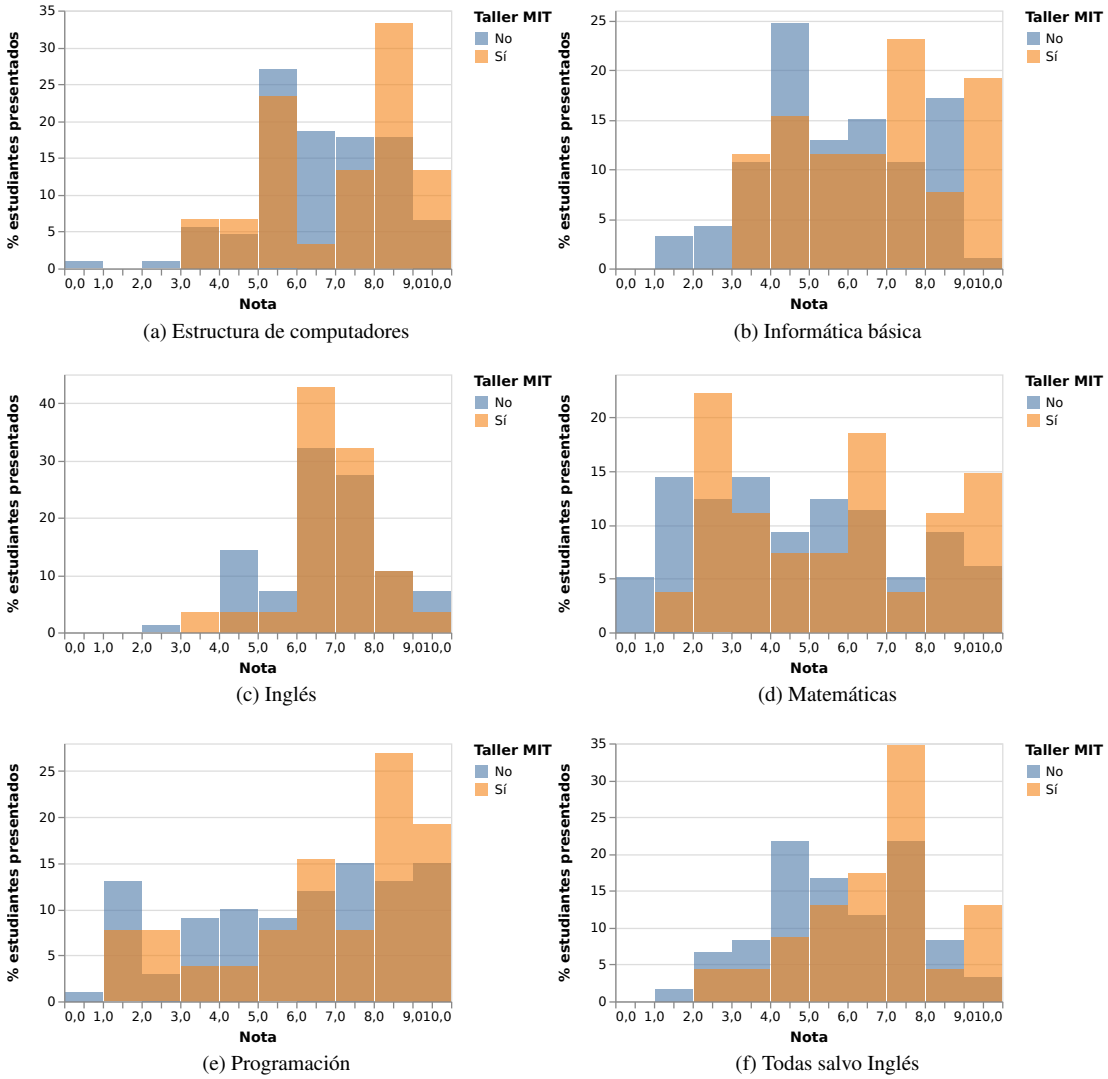



Figura 5: Distribución de las calificaciones de los estudiantes que han aprovechado el taller (naranja) y de los que no (azul), para cada asignatura y para la combinación de todas las asignaturas menos Inglés




(a) Estructura de computadores

(b) Informática básica

(c) Inglés

(d) Matemáticas

(e) Programación

(f) Todas salvo Inglés

Figura 6: Histogramas normalizados de las calificaciones obtenidas en las distintas asignaturas de primer curso, primer semestre, por los estudiantes que han aprovechado el taller (en naranja) y el resto de estudiantes (en azul)

## Agradecimientos

Nos gustaría agradecer en primer lugar el trabajo realizado por los desarrolladores de MIT App Inventor y por la comunidad educativa constituida alrededor de esta herramienta. Consideramos que es una excelente plataforma para la difusión de las STEM y su publicación en abierto permitirá que mucha gente la utilice, con las ventajas que eso conllevará para la sociedad en su conjunto.

También nos gustaría dar las gracias a nuestros compañeros de primer curso: Juan Carlos Amengual Argudo, Fernando Javier Hernando Carrillo, José Luis Llopis Borrás, Ana María Lluch Peris, Marina Murillo Arcila, y María Luisa Renau Renau, que nos han cedido amablemente las calificaciones de las asignaturas de las que son responsables para que pudiéramos contrastar sus resultados.

Por último, agradecemos al Departamento de Ingeniería y Ciencia de los Computadores el apoyo recibido para poder llevar a cabo esta iniciativa y por la financiación del taller.

## Referencias

[1] Sergio Barrachina, Germán Fabregat, Carlos Fernández y Germán León: *Utilizando ARMSim y QtARMSim para la docencia de Arquitectura de Computadores*. ReVisión, 8(3):2, 2015.

[2] Sergio Barrachina Mir, Germán Fabregat Llueca y José Vicente Martí Avilés: *Utilizando Arduino DUE en la docencia de la entrada/salida*. En *Actas de las XXI Jornadas de la Enseñanza Universitaria de la Informática*, páginas 58–65. Universitat Oberta La Salle, 2015.

[3] Xavi Canaleta, Fermín Sánchez, Inés Jacob, Ángel Velázquez y Merche Marques: *Declaración AENUI-CODDII por la inclusión de asignaturas específicas deficiencia y tecnología informática en los estudios básicos de la enseñanza secundaria y bachillerato*. Actas de XX las Jornadas sobre Enseñanza Universitaria de la Informática, páginas 229–236, 2014.

[4] John Maloney, Mitchel Resnick, Natalie Rusk, Brian Silverman y Evelyn Eastmond: *The Scratch programming language and environment*. ACM Transactions on Computing Education (TOCE), 10(4):16:1–16:15, 2010.

[5] Roberto Muñoz, Thiago S Barcelos, Rodolfo Villarroel, Marta Barría, Carlos Becerra, Rene Noel y Ismar Frango Silveira: *Uso de Scratch y Lego Mindstorms como apoyo a la docencia en Fundamentos de programación*. En *Actas de las XXI Jornadas de la Enseñanza Universitaria de la Informática*, páginas 248–254. Universitat Oberta La Salle, 2015.

[6] Stavros A Nikou y Anastasios A Economides: *Transition in student motivation during a Scratch and an App Inventor course*. En *2014 IEEE Global Engineering Education Conference (EDUCON)*, páginas 1042–1045. IEEE, 2014.

[7] Stamatios Papadakis, Michail Kalogiannakis, Vasileios Orfanakis y Nicholas Zaranis: *Novice programming environments. Scratch & App Inventor: a first comparison*. En *Proceedings of the 2014 Workshop on Interaction Design in Educational Environments*, páginas 1–7. ACM, 2014.

[8] Sonia Pamplona Roche y Nelson Medinilla Martínez: *Evaluación de entornos de programación para el aprendizaje*. Actas de las XVII Jornadas sobre Enseñanza Universitaria de la Informática, páginas 83–90, 2011.

[9] Sergio Trilles y Carlos 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, 3(0):303–310, 2018, ISSN 2531-0607.

[10] David Wolber: *App Inventor and real-world motivation*. En *Proceedings of the 42nd ACM technical symposium on Computer science education*, páginas 601–606. ACM, 2011.