



GRADO EN INGENIERÍA TELEMÁTICA

Curso Académico 2020/2021

Trabajo Fin de Grado

Integracion del Robot Lego Ev3 a la plataforma de Kibotics

Autor : Daniel Pulido Millanes

Tutor : Dr. José María Cañas Plaza

Índice general

Lista de figuras	5
Lista de tablas	7
1. Introducción	1
1.1. Tecnologías web	1
1.1.1. HTTP	2
1.1.2. Tecnologías en cliente	4
1.1.3. Tecnologías en servidor	4
1.2. Robótica	5
1.2.1. Aplicaciones robóticas	5
1.2.2. Software en robótica	8
1.3. Robótica educativa	9

Índice de figuras

1.1.	Aplicación <i>web</i> dedicada a la venta <i>on-line</i>	2
1.2.	Aplicación <i>web</i> dedicada a la distribución de contenidos	2
1.3.	Comunicación cliente-servidor en HTTP	3
1.5.	Robot médico <i>Da Vinci</i>	7
1.6.	Vehículo <i>Tesla</i> con imágenes de sus sensores	7
1.7.	Robots de logística de <i>Amazon</i>	8
1.8.	Robot <i>Curiosity</i> de la <i>NASA</i>	8
1.9.	Interfaz gráfica de <i>Scratch</i>	11
1.11.	Interfaz gráfica de <i>Kodu</i>	12
1.12.	Interfaz gráfica de <i>Snap!</i>	12
1.13.	Interfaz gráfica de <i>AppInventor</i>	13
1.14.	Ejemplo de <i>robot</i> montado con <i>Mindstorms</i>	13
1.15.	mBot de la plataforma <i>Makeblock</i>	14

Índice de cuadros

Capítulo 1

Introducción

En este capítulo se introducen los conceptos básicos en los que se apoya este proyecto. Se explican y se ponen en contexto las tecnologías web y el estado actual de la robótica y su expansión hasta llegar al punto en el que se encuentra, haciendo especial mención a la robótica educativa en la que se centra el proyecto.

1.1. Tecnologías web

Las tecnologías web han ido evolucionando a lo largo de los últimos años. Se basan fundamentalmente en un modelo cliente-servidor. Actualmente lo más común son las *aplicaciones web*, que son herramientas ejecutadas mediante un navegador *web* en las que los datos son procesados y almacenados en un servidor. Dentro de estas aplicaciones *web* exitosas se pueden encontrar sistemas de correo electrónico (*Gmail* o *Outlook*), tiendas *online* (*Amazon*), distribución de contenidos (*Netflix* o *Spotify*) o *wikis* (*Wikipedia*).

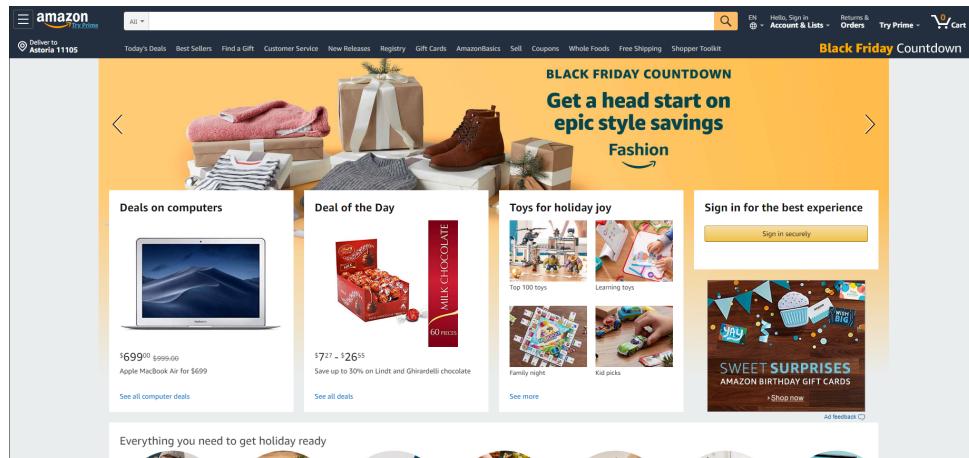


Figura 1.1: Aplicación web dedicada a la venta *on-line*

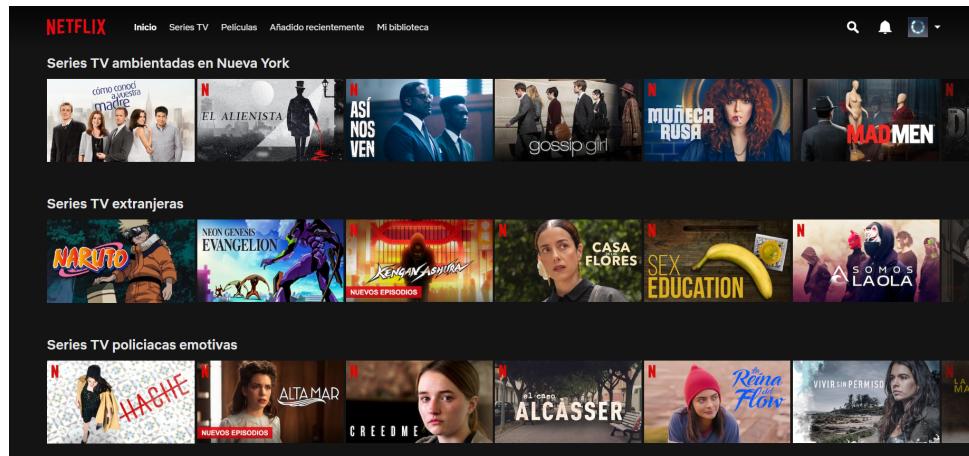


Figura 1.2: Aplicación web dedicada a la distribución de contenidos

1.1.1. HTTP

HiperText Transfer Protocol (HTTP) es el protocolo de nivel de aplicación utilizado para transferir recursos hipermedia entre ordenadores y sigue el esquema petición-respuesta entre cliente y servidor (Figura 1.3). En esta comunicación el cliente abre una conexión *TCP* con el servidor y envía un mensaje de petición *HTTP* y, por parte del servidor, responde al cliente con un mensaje *HTTP* y cierra la conexión *TCP*. El protocolo *HTTP* no mantiene estado. Es decir, el servidor trata cada petición de manera aislada y no almacena información sobre peticiones realizadas por un mismo cliente. Las peticiones están definidas por el protocolo y tienen métodos concretos:

- GET: Sigue una solicitud de un recurso al servidor especificando su *URL*.
- HEAD: Método similar a GET con la diferencia de que únicamente solicita las cabeceras y no descarga el recurso completo.
- POST: Envía datos al servidor, normalmente un recurso específico que provoca un cambio de estado.
- PUT: Actualiza información sobre un recurso del servidor.
- DELETE: Elimina en el servidor un recurso.

Aunque estos son los principales métodos, el protocolo tiene flexibilidad para ir añadiendo nuevos e incorporar funcionalidad. El número de métodos ha ido aumentando con las nuevas versiones.

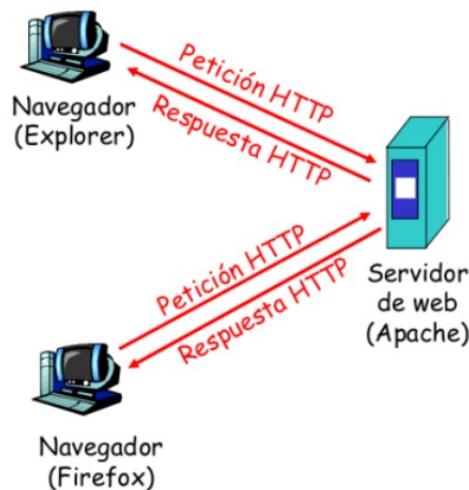


Figura 1.3: Comunicación cliente-servidor en HTTP

La versión actual del protocolo es del 2 de abril de 2019 (*HTTP/2.4.38*) pero la primera versión es del 1991, que es cuando aparece un estándar para la publicación de páginas web mediante un lenguaje de marcas de hipertexto: *HTML*.

1.1.2. Tecnologías en cliente

Es la parte encargada de dar forma a la interfaz de usuario y de establecer la comunicación con el servidor. Una pieza importante del cliente es el navegador, ya que es el encargado de leer e interpretar la información recibida. Entre los navegadores *web* más empleados se encuentran *Firefox*, *Google Chrome* u *Opera*[?]. Las tecnologías que hacen posible esa comunicación son:

- **HTML.** Estándar más utilizado para el desarrollo de páginas web. Sigue el modelo de objetos *Document Object Model (DOM)*, que define la manera en la que se comunican dichos objetos con una interfaz para representar documentos. El *DOM* permite el acceso dinámico a través de la programación con lenguajes como *JavaScript* y define como se comunican objetos y elementos con el navegador *web*. Actualmente los navegadores usan la versión *HTML5*, que incluye muchas mejoras respecto a su predecesor: *canvas*, *web-sockets*, *WebRTC*, vídeo, audio, etc. Este lenguaje indica la estructura de una página web, para editar el estilo y presentación visual hay que hacer uso de otros elementos como *CSS*.
- **Cascading Style Sheets (CSS).** Lenguaje de diseño gráfico para definir y crear la presentación de un documento escrito en un lenguaje de marcado. De esta forma, se puede separar información y datos (en los documentos *HTML*) y todo lo relativo al diseño y presentación (en documentos *CSS*). Actualmente los navegadores usan la versión *CSS3*.
- **JavaScript.** Lenguaje destinado a programar la lógica, dinamismo e interacción con el usuario. Es el más común y extendido en el desarrollo de páginas. Debido a su importancia en el proyecto, se explicará más a fondo en próximos capítulos.

1.1.3. Tecnologías en servidor

Son las encargadas de dar forma al servidor web de manera que permiten el acceso a herramientas como base de datos, conexiones de red o recursos compartidos. O, dicho de otra forma, se ocupan de realizar las tareas necesarias para hacer posible crear una aplicación que visualizará el cliente. Las más utilizadas son:

- *Node.js*: es una forma de ejecutar *JavaScript* en el servidor. Proporciona un entorno de ejecución del lado del servidor que compila y ejecuta a gran velocidad. Esto es debido a

que compila en código máquina nativo en lugar de interpretarlo o ejecutarlo.

- *Django*: entorno web de alto nivel programado en *python* diseñado para realizar aplicaciones de cualquier complejidad. Es seguro, rápido y escalable. Además, incluye una interfaz para acceder a bases de datos lo que facilita las consultas al no tener que manejar *SQL* y realizarlas con filtros de *Python*.
- *Spring*: herramienta basada en *Java* cuya finalidad es simplificar el desarrollo de aplicaciones ya que facilita la configuración de la aplicación y el despliegue en servidor. De esta manera, *Spring* está pensado para aplicaciones web, servicios *REST*, análisis de datos e integración de sistemas.

1.2. Robótica

La robótica es una rama tecnológica encargada del diseño y construcción de aparatos que realizan operaciones y trabajos en sustitución de la mano de obra humana.

Un robot es un sistema autónomo programable capaz de realizar tareas de ayuda al ser humano y con aplicaciones en campos diversos como la medicina, el hogar, las fábricas, etc.

Los robots se componen de sensores, controladores y actuadores.

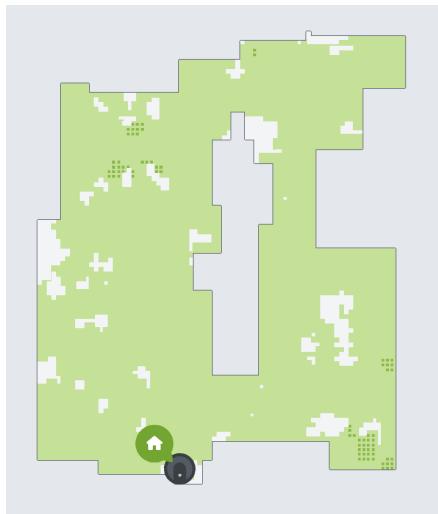
- **Sensores**: son los encargados de recoger información del entorno. En este grupo se encuentran láseres, cámaras, ultrasonidos u odómetros. Estos dispositivos equivalen a los sentidos humanos.
- **Controladores**: analizan los datos recogidos por los sensores y elaboran una respuesta que va a ser enviada a los actuadores. En los seres humanos equivale al cerebro.
- **Actuadores**: se encargan de transformar energía eléctrica, hidráulica o neumática en mecánica. Son los que interactúan con el entorno y equivalen a los músculos humanos.

1.2.1. Aplicaciones robóticas

La robótica es ya una realidad. Los robots no son utilizados únicamente por empresas para labores industriales, si no que se pueden encontrar en hogares y en la vida cotidiana de las personas. Esto es debido al aumento de eficiencia que generan, reducción de costes o el control

de errores que aportan. Algunas ejemplos de *robots* empleados en la actualidad, tanto en ámbito doméstico como en industrial, son los siguientes:

- Aspiradora robótica *Roomba*, robot autónomo que, en los modelos más avanzados, reconocen el entorno, trazan un mapa de la casa y vacían su depósito automáticamente.



(a) Mapa generado por un robot *Roomba*



(b) Aspirador robótico *Roomba*

- Robot médico *Da Vinci*, que permite al cirujano operar a través de una consola mejorando así su precisión y reduciendo riesgos en operaciones quirúrgicas.



Figura 1.5: Robot médico *Da Vinci*

- Vehículos autónomos *Tesla*, que mediante cámaras y sensores de ultra-sonidos analizan el entorno para garantizar una conducción autónoma segura.



Figura 1.6: Vehículo *Tesla* con imágenes de sus sensores

- Robots de logística de *Amazon*, que se encargan, de manera autónoma, de localizar la estantería donde se encuentra un paquete solicitado y la desplaza por todo el centro logístico al lugar de destino.

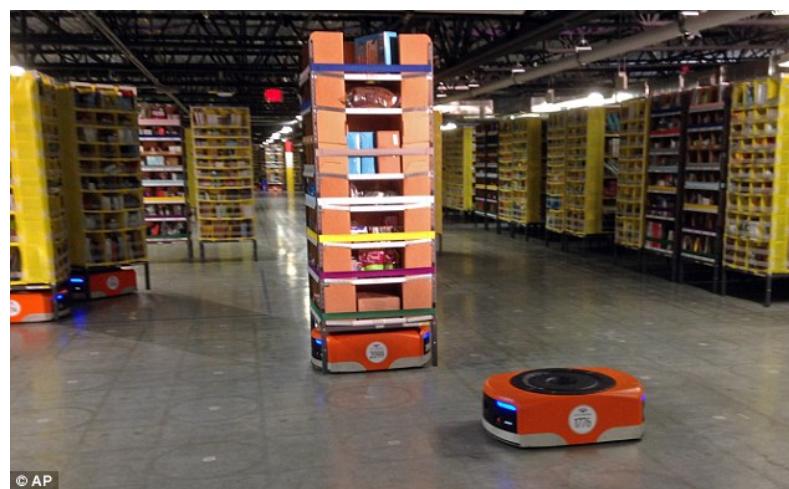


Figura 1.7: Robots de logística de *Amazon*

- Robot *Curiosity* de la *NASA*, empleado para la exploración de la superficie del planeta Marte.



Figura 1.8: Robot *Curiosity* de la NASA

1.2.2. Software en robótica

Para dotar de inteligencia autónoma a los robots se requiere desarrollar sistemas complejos, aplicaciones e infraestructuras. Por ejemplo, hace años, el desarrollo de *software* se realizaba adoptando soluciones “*ad-hoc*” dotando a cada robot de un diseño específico y con sensores y actuadores concretos. Esto implicaba que había que implementar todo el *software* para un nuevo robot debido a que no se podía aplicar el desarrollado anteriormente. En la actualidad, existen plataformas que permiten el desarrollo de aplicaciones robóticas de forma eficiente y genérica permitiendo así reutilizar aplicaciones creadas en otros robots.

Para dotar de inteligencia a un robot se desarrolla un *software* que habitualmente se programa con ayuda de herramientas, como los *middleware* robóticos. El uso de estas herramientas permiten introducir una capa de abstracción con los *drivers* y el *hardware* del robot, reduciendo la complejidad y los conocimientos necesarios para realizar desarrollos. Algunas de estos *middleware* son:

- ***Robot Operating System (ROS)*[?]**. Plataforma de *software* libre para el desarrollo de *software* de robots. Provee servicios estándar de un sistema operativo como la abstracción de *hardware*, control de dispositivos de bajo nivel, mecanismos de intercambio de mensajes entre procesos y una serie de herramientas utilizadas en robótica.
- ***ORCA*[?]**. Plataforma de *software* libre dedicada al desarrollo de aplicaciones robóticas.

Dedicada principalmente al desarrollo de componentes, que se pueden ejecutar de manera independiente o se pueden unir para formar sistemas robóticos de distintas complejidades. Permite reutilizar código y emplear componentes robóticos ya creados.

- **OROCOS[?]** Proyecto de *software* libre dedicado al control de robots y máquinas. Está orientado a componentes y permite añadir funcionalidad de manera sencilla y sin recopilar todo el código. Incluye paquetes complementarios como filtros de *Bayes*, librerías de control dinámico y cinemático o visión.

1.3. Robótica educativa

La robótica con fines educativos está empezando a adquirir importancia en la enseñanza porque su aprendizaje está disponible para estudiantes de cualquier nivel. Este método de enseñanza intenta despertar el interés de los alumnos porque, gracias a la innovación que posibilita la tecnología, añade un componente atractivo e integrador a las asignaturas tradicionales. En 2015 la comunidad de Madrid introdujo la asignatura de robótica en los planes docentes de Enseñanza Secundaria con la asignatura “Tecnología, Programación y Robótica”[?] y en el curso 2020-2021 se empezará a implantar en Educación Primaria la asignatura “Programación y Robótica”[?].

El método más empleado para este tipo de enseñanza es la educación *STEM* (*Science, Technology, Engineering and Mathematics*). Este tipo de educación promueve una cultura de pensamiento científico, así como la adquisición de conocimientos tecnológicos aplicables a situaciones reales y permite desarrollar competencias para la resolución de problemas y un pensamiento creativo y crítico.

La enseñanza en centros escolares se realiza en gran parte mediante plataformas como la creada por LEGO o placas Arduino que simplifican el aprendizaje y resulta motivadora para el alumno porque obtiene resultados vistosos y se le puede dar una aplicación real.

Resulta importante acercar la robótica a alumnos en edades muy tempranas, partiendo de nociones básicas, dada la importancia que está adquiriendo la disciplina y la fuerte presencia

que tiene en la mayor parte de los sectores laborales. Además, la capacidad de programar es una parte importante en la sociedad actual ya que, con esta habilidad, se aprenden estrategias para resolver problemas, diseñar proyectos y comunicar ideas.

Una manera de acercar estas nociones básicas son los lenguajes de programación visual. Se trata de lenguajes potentes y muy intuitivos que abstraen al alumno de la complejidad que implica la sintaxis y aportan un entorno visual haciendo que el software sea bien aceptado por estudiantes de corta edad. Además permiten a los usuarios programar manipulando elementos gráficos en lugar de hacerlo textualmente. Los lenguajes de programación visual más importantes son:

- **Scratch[?]:** proyecto liderado por el *MIT*, es utilizado por estudiantes y docentes para programar animaciones, juegos e interacciones fácilmente gracias a su interfaz visual. Se trata de un lenguaje donde los programas se construyen ensamblando bloques gráficos y cada bloque es el equivalente a una función o método de cualquier lenguaje de programación. Actualmente está en su versión *Scratch 3.0* y en la figura 1.9 se puede ver un ejemplo de su interfaz gráfica.

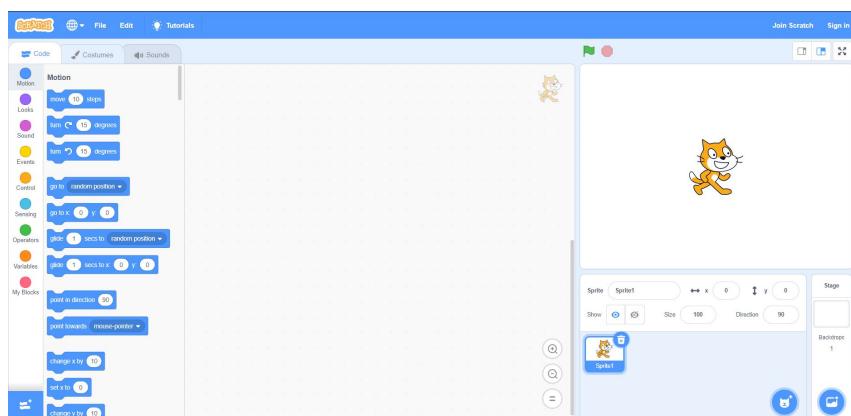
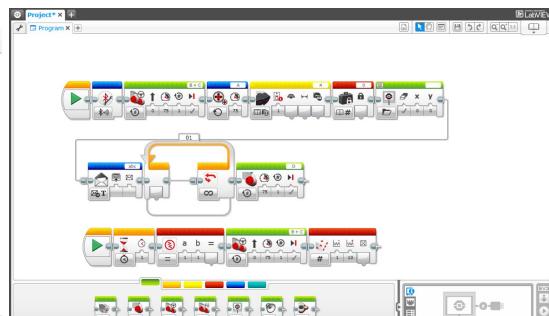


Figura 1.9: Interfaz gráfica de Scratch

- **LEGO[?]:** dispone de una amplia gama de robots programables y cada uno de ellos tiene un sistema gráfico. En la figura 1.14 se pueden ver dos ejemplos de distintos software para distintos robots.



(a) Interfaz gráfica de LEGO WeDo 2.0



(b) Interfaz gráfica de LEGO MINDSTORMS EV3

- **Kodu[?]:** lenguaje de programación visual creado por Microsoft para desarrollar videojuegos. Diseñado para ser muy accesible y agradable para cualquier usuario siendo un lenguaje basado en reglas, condiciones y acciones prescindiendo de muchas convenciones de programación como bucles, subrutinas o variables simbólicas. Permite a los más jóvenes ser creadores de sus propios videojuegos.



Figura 1.11: Interfaz gráfica de Kodu

- **Snap![?]:** basado en *Scratch*, sigue su facilidad para aprender a programar pero su uso se concentra en edades algo más avanzadas. Accesible desde cualquier navegador al estar programado en *JavaScript*.

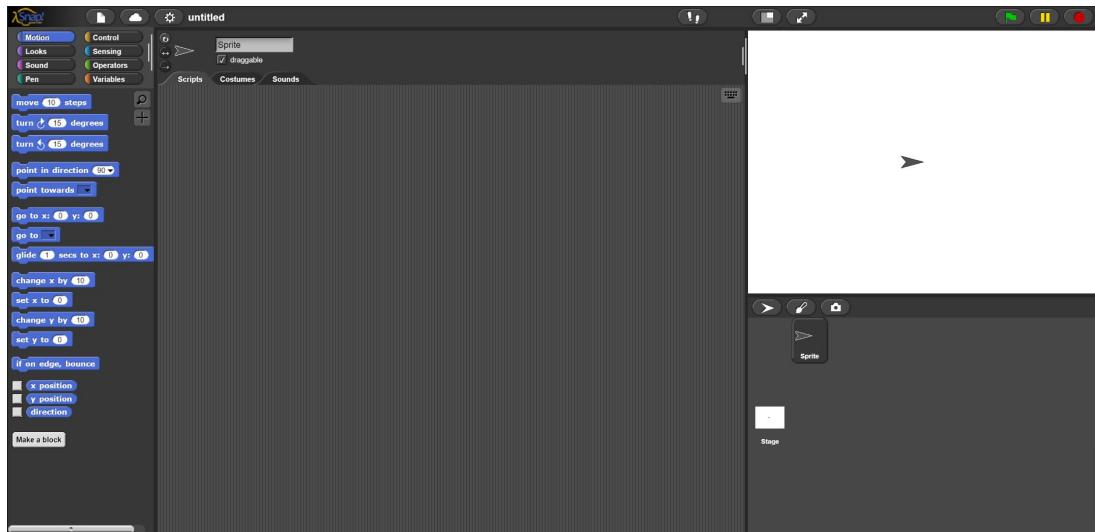


Figura 1.12: Interfaz gráfica de Snap!

- **AppInventor[?]:** Entorno de desarrollo de software creado por *Google* en colaboración con el *MIT* destinado a la elaboración de aplicaciones *Android*. El usuario puede desarrollar de forma visual y con ayuda de bloques una aplicación que cubre un gran número de necesidades básicas en un dispositivo móvil.

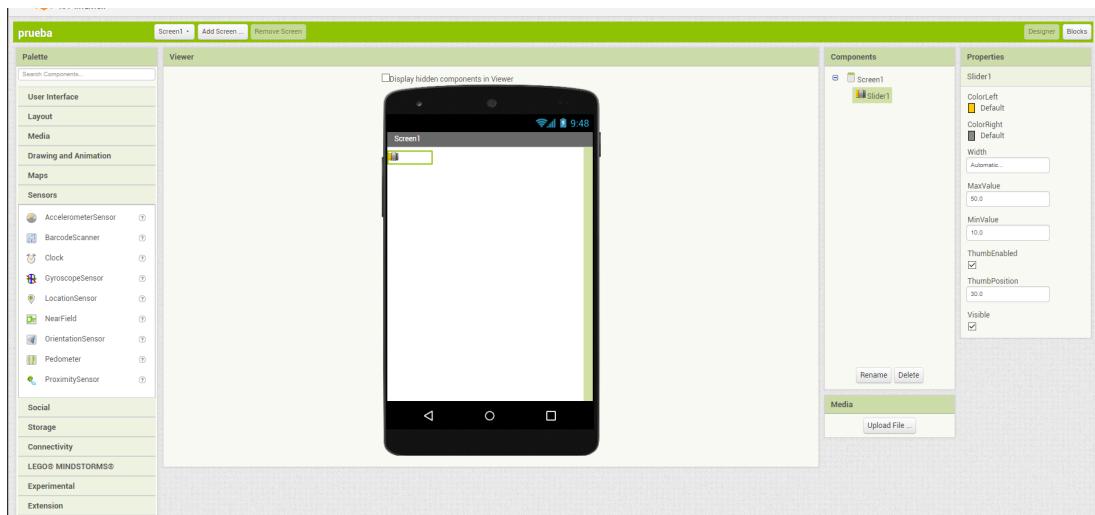


Figura 1.13: Interfaz gráfica de AppInventor

También es habitual en robótica educativa el uso de plataformas hardware, que incorporan los elementos básicos para la construcción de un robot. Las más empleadas son:

- **LEGO mindstorms**: plataforma dedicada a construir *robots* con piezas de LEGO. También incluye un microprocesador para ser programado, sensores (infrarrojos, táctiles y de color) y motores.



Figura 1.14: Ejemplo de *robot* montado con *Mindstorms*

- **Makeblock[?]**: plataforma de construcción robótica muy accesible que permite la creación de *robots* con *kits* dirigidos a todo tipo de público. El *robot* que se construye con estos *kits* es el *mBot*:



Figura 1.15: mBot de la plataforma *Makeblock*

- **Arduino[?]**: Plataforma de creación electrónica. Permite crear microordenadores a los que dar multitud de usos con una sola placa, que dispone de entradas para periféricos y un microcontrolador.

