Más allá de Jupyter: usando Google Colab para la Programación de Robots

Enric Cervera, Raul Marin, Josep Marin
Universitat Jaume-I de Castelló, ecervera@uji.es, rmarin@uji.es, josepmaringarces@gmail.com

Resumen

En los últimos años el entorno interactivo de programación Jupyter se ha popularizado enormemente como herramienta para la enseñanza en muchos campos científicos y técnicos, incluyendo la robótica. Jupyter permite integrar materiales documentales con ejemplos de código ejecutables y modificables. En este trabajo proponemos el uso de Google Colab, una herramienta basada en Jupyter, para mejorar la experiencia del usuario y programador, tanto del profesor que prepara el material como del estudiante que lo usa. Estas mejoras se basan principalmente en la integración de Google Drive y GitHub, lo que permite el almacenamiento y distribución de las plantillas iniciales de los notebooks desde la nube, y las copias de seguridad de los notebooks modificados por los estudiantes en sus propias cuentas de Google Drive. Todo ello sin necesidad de almacenar los notebooks en los ordenadores locales donde se ejecuta Jupyter, sean robots reales u ordenadores donde se ejecutan simulaciones. Además de la descripción del workflow usando Colab, presentaremos ejemplos concretos de uso en simulador y un robot real. También, a modo de aplicación de la tecnología, se comparte la experiencia obtenida en el uso de esta plataforma con estudiantes de primer curso de ingenierías, así como con la formación inicial de un equipo interesado en la competición de robots CEABOT.

Palabras clave: Educational Robotics, Robot Competitions, Programming

 https://doi.org/10.17979/spudc.9788497498418.0662

1 Introducción

Jupyter es un entorno de computación interactivo basado en páginas web. Extiende la programación basada en consola e integra los procesos de desarrollo y ejecución de código con la documentación del mismo y su contexto [8].

En sus inicios Jupyter se desarrolló a partir del entorno Interactive Python [10] así como de ideas del entorno propietario Mathematica [13]. En la actualidad se ha convertido en un entorno muy popular tanto en docencia como en investigación en muchas disciplinas, especialmente data science, pero también en robótica [12].

Los cuadernos Jupyter se han usado para el desarrollo de simulaciones en la nube para RoboCup [4]. Siendo ROS un estándar de facto para la programación de robots [11], la interfaz de Jupyter con el código ROS se presentó en [3], mostrando las ventajas de los cuadernos interactivos frente a las páginas web estáticas de los tutoriales de ROS. Este enfoque se amplió a una programación general de robots [2], y para el uso de Jupyter en el desarrollo y prueba de código de robótica de código abierto [5]. Otras propuestas de entornos Jupyter con ROS para la educación en robótica se han presentado en [1, 6, 7].

En este artículo proponemos el uso de Google Colab [9] para la educación en robótica. Colab se basa también en Jupyter notebooks, pero los integra con Google Drive y GitHub. Dicha integración simplifica en gran medida la implementación y administración de cuadernos y permite que el usuario los almacene en su cuenta en la nube. Además, los cuadernos no necesitan guardarse localmente en el robot o en el ordenador del simulador.

En las siguientes secciones, describiremos las diferencias entre el enfoque estándar de Jupyter y Colab, y mostraremos ejemplos completos en el simulador de Webots, TurtleBot 3, y robots susceptibles de ser usados en la competición de robótica del Comité Español de Automática (i.e. CEABOT). De hecho, el paper presenta, a modo de conclusión, una planificación del uso de esta herramienta para la ayuda a la formación de un equipo participante en las pruebas CEABOT. Finalmente, presentamos las conclusiones y futuras líneas de desarrollo.

2 Arquitectura Jupyter Notebooks

En la configuración más habitual, el cliente utiliza un navegador de Internet (Chrome, Firefox, Safari, Edge) para conectarse al servidor de Jupyter (Fig. 1). El servidor ejecuta el código del notebook accediendo a través de una API de Python al simulador o al robot. Si se está ejecutando un simulador, tanto éste como el navegador cliente pueden ejecutarse en el mismo ordenador o puesto de trabajo. Si se usa un robot, lo habitual es que Jupyter se ejecute en el ordenador del robot, que estará conectado mediante WiFi al ordenador del cliente.

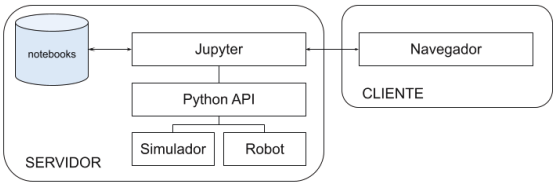

Figura 1: Configuración habitual de Jupyter.

En esta configuración los notebooks están almacenados en el servidor de Jupyter, y se modifican libremente por el usuario. Al final de la sesión es necesario restaurar los notebooks al estado original para realizar una nueva sesión. Una posibilidad es descargar los notebooks de un repositorio en GitHub, y antes de cada sesión descartar los cambios realizados mediante el comando git restore.

Si el usuario desea conservar una copia de sus notebooks, debe descargarlos explícitamente en el ordenador cliente, y de allí subirlos a la nube o copiarlos en un dispositivo externo.

Es posible configurar un entorno multiusuario mediante JupyterHub, con el consiguiente coste de administrarlo para mantener la seguridad y privacidad de los usuarios. Nuestro objetivo es delegar esta administración a la nube de Drive y GitHub mediante el uso de Colab.

3 Arquitectura Google Colab

Colab, también conocido como “Colaboratory” es básicamente un servidor de Jupyter con una interfaz adaptada para su integración con Google Drive y GitHub. Además, permite ejecutar los notebooks tanto en la nube como en un entorno local. Esta segunda opción es la que nos va a permitir ejecutar el código sobre el simulador o el robot.

En la Fig. 2 se muestra la configuración de Google Colab conectado al entorno local del servidor.

https://doi.org/10.17979/spudc.9788497498418.0662

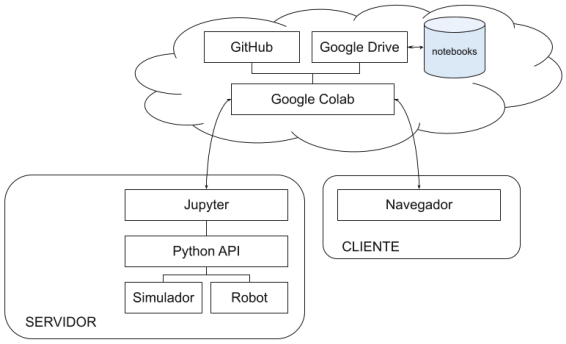

Figura 2: Configuración de Jupyter con Google Colab.

El usuario usa el navegador de Internet para conectarse a la web de Colab, donde selecciona la ejecución en un entorno local. Si el servidor ejecuta un simulador en el mismo ordenador que el cliente, la conexión es directa. Si el servidor se ejecuta en otro ordenador o en un robot, es necesario redireccionar el puerto 8888 del ordenador cliente al servidor mediante el comando ssh:

```
ssh -N <usuario>@<IP_SERVIDOR>
  -L 8888:<IP_CLIENTE>:8888
```

De esta manera Colab se conecta con Jupyter en el servidor. No obstante, los notebooks están almacenados en la nube y no se guarda copia local en el servidor.

Los notebooks originales se distribuyen a través de un repositorio GitHub. Un botón en el notebook permite abrirlos directamente en Colab, y ejecutarlos. El usuario puede almacenar los notebooks modificados en su propia cuenta de Google Drive, y compartirlos también con otros usuarios.

Al final de la sesión, no es necesario ningún tipo de mantenimiento en el servidor, pues no se ha almacenado ningún fichero localmente.

El propietario o administrador del repositorio GitHub puede subir modificaciones directamente al repositorio después de haberlas ejecutado en Colab para comprobar su correcto funcionamiento.

4 Experimento 1: Simulador

A continuación se muestra un ejemplo de ejecución de un notebook en Colab sobre el simulador Webots. El notebook está almacenado en un repositorio de GitHub (Fig. 3).

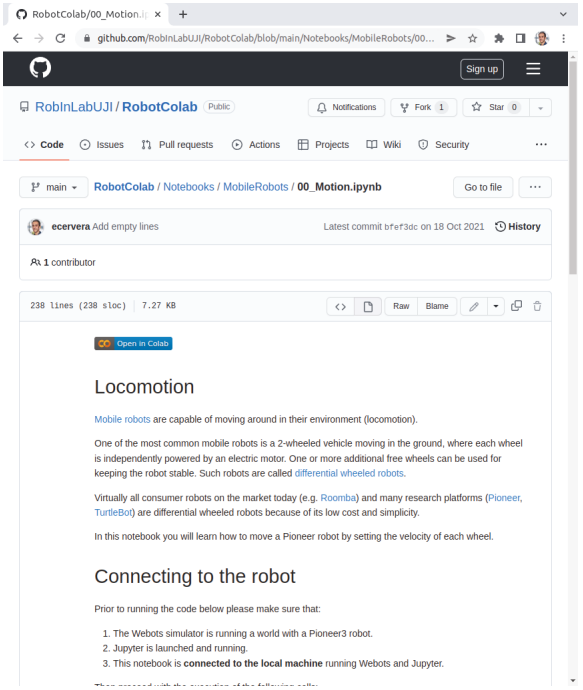

Figura 3: Notebook almacenado en GitHub.

Este notebook se ejecuta en un servidor Jupyter local a través de Colab. Para ello hay que lanzar Jupyter desde un terminal, y abrir la aplicación Webots con la simulación del robot.

El notebook incorpora un botón etiquetado “Open in Colab” que abre Colab para editar y ejecutar el código del notebook. Es necesario seleccionar la opción “Conectar a un entorno de ejecución local” para conectarse al servidor Jupyter lanzado localmente. Si no es así, el notebook se ejecutará en la nube de Colab y obviamente no se podrá conectar al simulador.

Una vez lanzado, la configuración del escritorio aparece en la Fig. 4, donde se muestra la ventana del simulador, la terminal donde se ejecuta Jupyter, y la ventana del navegador del cliente con el código en ejecución.

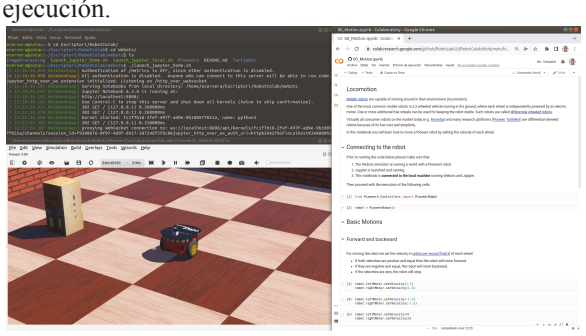

Figura 4: Ejecución del notebook en Colab, conectado al simulador Webots local.

El código de este ejemplo crea un objeto `PioneerRobot` que permite ejecutar órdenes de

https://doi.org/10.17979/spudc.9788497498418.0662

movimiento en el robot simulado, indicando los valores de las velocidades de los motores izquierdo y derecho del robot.

5 Experimento 2: TurtleBot

En este experimento se ha validado la ejecución del notebook en la plataforma robótica real TurtleBot. Inicialmente se ha conectado el puerto 8888 del ordenador del usuario con el mismo puerto del robot donde se ejecuta Jupyter, que ha de ser accesible a través de la red. En la Fig. 5 se muestra un ejemplo de notebook almacenado en un repositorio de GitHub.

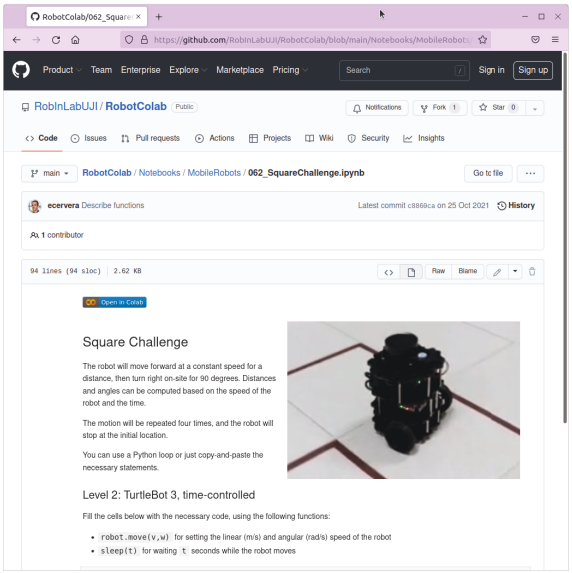

Figura 5: Notebook almacenado en un repositorio de GitHub.

Mediante un clic en el botón “Open in Colab” se abre el notebook en Google Colab. A continuación se selecciona la opción “Conectar a un entorno de ejecución local”, que gracias a la redirección de puertos se conectará al Jupyter del robot.

El notebook puede entonces modificarse y ejecutarse en el robot. El estudiante podrá almacenar el notebook modificado en su cuenta de Google Drive y compartirlo con otros usuarios. La Fig. 6 muestra el navegador ejecutando el notebook, y una imagen del robot en funcionamiento.

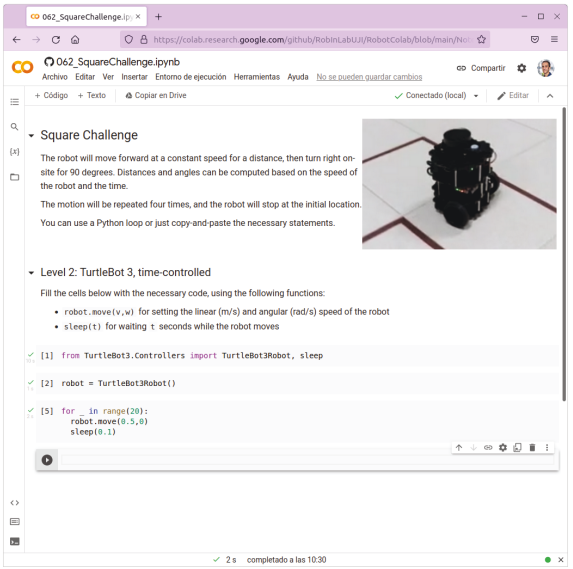

Figura 6: Notebook en ejecución e imagen del robot en funcionamiento.

El código del notebook es un ejemplo sencillo de movimiento. Tras crear el objeto robot, se invoca repetidamente el método move con una pequeña pausa. Esto es necesario porque el controlador del robot incorpora un temporizador que detiene automáticamente el movimiento si no llega un nuevo comando de velocidad dentro del tiempo fijado.

Este experimento ha demostrado la flexibilidad en la experiencia educativa del estudiante, que puede guardar sus programas en el sistema Colab, y decidir su ejecución en simulación o en el robot TurtleBot según convenga.

6 Experimento 3: CEABOT

En este experimento se ha procurado extender el uso del sistema descrito anteriormente a la formación de equipo del concuso CEABOT, con el objetivo de reforzar y acompañar estudiantes y futuros investigadores del área de la robótica en su proceso de formación, fomentando el trabajo en equipo y la compartición de conocimientos. Esta competición

https://doi.org/10.17979/spudc.9788497498418.0662

incluye una fase de formación, donde los futuros equipos reciben la información básica necesaria para iniciarse en la programación de pequeños robots móviles humanoides. Así mismo, después de esta formación común de todos los equipos las correspondientes universidades inician sus procesos de entrenamiento. Precisamente, en la Universidad Jaume I de Castellón existe un taller de competición CEABOT, donde se acompaña a los estudiantes y se les da soporte y orientación académica en el desarrollo de sus proyectos. Se trata de una formación a lo largo de toda la carrera, con lo cual es necesario dar esta formación de forma gradual, procurando vincularla en la medida de lo posible a la formación que reciben en los correspondientes grados universitarios.

Actualmente el equipo CEABOT UJI está formado por estudiantes de primer curso del grado en Inteligencia Robótica e Ingeniería en Informática, los cuales han recibido una formación básica de programación en Python y todavía no tienen la formación necesaria para bordar todas las pruebas CEABOT de forma satisfactoria. Por ello, es necesario crear un plan de formación, experiencia compartida en esta sección.

Durante la edición de la competición CEABOT 2022 se proponen las siguiente pruebas[1]:

1. Prueba Libre: Permite a los estudiantes desarrollar un proyecto que sea aplicable total o parcialmente a un robot humanoide, dando importancia a la aplicación social e industrial.

2. Prueba Escaleras: Prueba de programación de la controladora del robot humanoide, pudiendo desarrollar sensores propios, con el objetivo de reconocer las escaleras y recorrerlas en ambos sentidos (ver Figura 7).

3. Visión: Prueba que en la edición del 2022 plantea el uso de marcadores libres (e.g. ArUco) para seguir un robot líder. En esta prueba se permiten dos modalidades, el seguidor con ruedas y el humanoide, así como la ejecución del algoritmo de control en el propio robot o en un ordenador externo (ver Figura 8). Al permitir el uso de robots móviles con ruedas, esta prueba es un buen reto para iniciar a los estudiantes en el control, y concentrarse en la solución del problema, evitando las complicaciones de la locomoción.

4. Sumo: Prueba de percepción y acción robótica con el objetivo de reconocer al oponente y derribarlo “cordialmente” (ver Figura 9).

Figure 7: Prueba escaleras en la competición CEABOT

Figure 8: Prueba Visión

[1]Página web del CeaBot con las pruebas mencionadas previamente, descritas a mayor detalle: http://ceabot.es/pruebas-ediciones-presenciales

https://doi.org/10.17979/spudc.9788497498418.0662

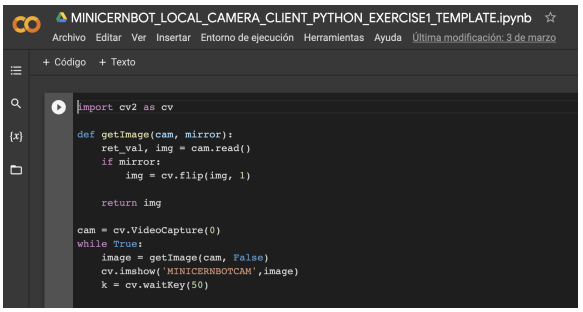

Figure 9: Plantilla Notebook Colab para iniciar la formación en la prueba de visión

Figure 10: Prueba Sumo en la competición CEABOT

En esta sección se resume el resultado de la reflexión académica contrastada de forma práctica, con el objetivo de iniciar la formación de un nuevo equipo CEABOT en la Universidad Jaume I de Castellón, considerando la alineación de los estudios con las pruebas a resolver. Esta formación se realizará de forma gradual a lo largo de los siguientes cursos, permitiendo a los estudiantes afrontar nuevos retos y pruebas de forma equilibrada (ver Figura 11).

1. FASE 1: Prueba visión con robot móvil TurtleBot 3 y arquitectura Colab. Durante el primer curso de ingeniería los estudiantes reciben formación básica en Python, a través del sistema Colab, entre otros. De esta forma, en esta fase el equipo CEABOT utiliza la misma tecnología para programar el algoritmo de seguimiento del líder, utilizando Python básico y la librería OpenCV.

2. FASE 2: Pruebas sumo y escaleras en bucle abierto. En esta fase educativa el equipo CEABOT recibe una formación básica de programación del robot humanoide Bioloid con RobotTask Plus/ Motion. De esta forma comprende el uso de secuencias de movimientos ya programados y la definición de nuevos, así como su combinación ordenada a través de un lenguaje de programación.

3. FASE 3: Pruebas sumo y escaleras en bucle cerrado. Una vez recibida la formación básica de electrónica, sensores y actuadores, los estudiantes estarán capacitados para programar algoritmos básicos de control en bucle cerrado, permitiendo resolver las pruebas de sumo y las escaleras con un mayor grado de autonomía.

4. FASE 4: Integración electrónica de ordenador embebido con cámara integrada en el robot humanoide. En esta fase educativa los estudiantes aprovechan los conocimientos recibidos en las asignaturas de electrónica, sensores y actuadores, y sistemas operativos para integrar una Raspberry Pi en el robot humanoide, añadiendo capacidad de programación de algoritmos de control por visión.

5. FASE 5: Integración informática del sistema Colab+Github+Jupyter en Raspberry Pi del humanoide Bioloid y resolución de la prueba de visión. En esta fase se cierra el bucle de aprendizaje que ha permitido iniciarse en la programación básica en Python con el robot móvil TurtleBot, permitiendo alcanzar resultados similares en un robot humanoide, y allanando así la curva de aprendizaje.

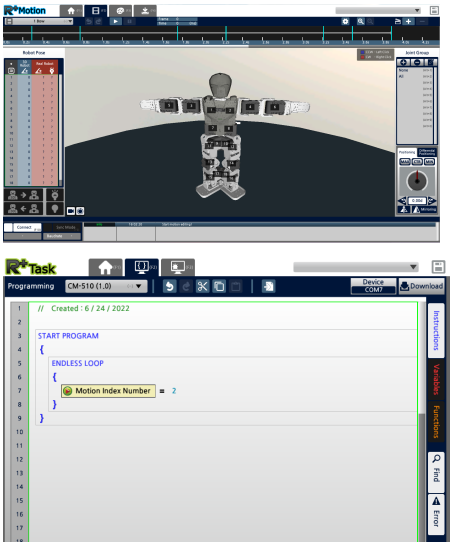

Figura 11: Programación básica robot humanoide Bioloid con RobotMotion (arriba) y RobotTask (abajo)

https://doi.org/10.17979/spudc.9788497498418.0662

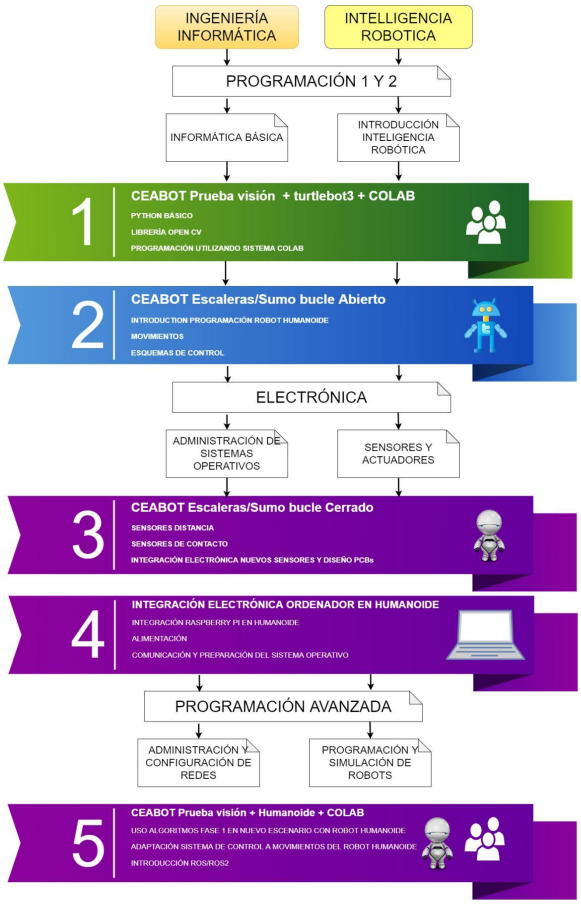

Figura 12: Fases plan formación equipo CEABOT

7 Conclusiones

Hemos presentado el uso de Google Colab para la programación de robots, tanto en simulación como en robots reales. Colab integra el servidor Jupyter con los servicios en la nube de GitHub y Google Drive. Con ello se simplifica la distribución y el mantenimiento de los notebooks, y se facilita que los usuarios almacenen el trabajo realizado en sus propias cuentas en la nube.

Además, no es necesario almacenar localmente los notebooks en los servidores del robot o del simulador, con lo que se evita que los usuarios encuentren copias modificadas por usuarios anteriores.

La configuración de Colab es muy sencilla pues se basa en la conexión a un entorno de ejecución local, requiriendo únicamente la redirección de un puerto hacia el ordenador del robot o del simulador, si éste es distinto del ordenador de trabajo del usuario.

Los ejemplos del simulador Webots y del robot TurtleBot 3 muestran el uso de los notebooks en entornos reales. En el futuro planeamos extender el uso de Colab a otros simuladores y plataformas, y estandarizar la interfaz para todos aquellos sistemas basados en ROS 1 y ROS 2.

Así mismo, el artículo presenta el uso de esta forma de aprendizaje en la formación de un nuevo equipo CEABOT, de estudiantes de primer curso del grado de ingeniería. El esquema de aprendizaje presenta una estructura por fase vinculada a las asignaturas que los estudiantes cursan durante los primeros cursos.

Agradecimientos

Este trabajo ha sido financiado por el Departamento de Ingeniería y Ciencia de los Computadores de la Universitat Jaume I de Castelló, y los proyectos PID2020-115332RB-C31 (COOPERAMOS), IDIFEDER/2018/013 (GV), UJI-B2021-30 (AUDAZ), H2020-Peacetolero-NFRP-2019-2020-04, UJI-B2021-27, UJI-B2021-42, y PROMETEO/2020/034 (GV).

English summary

Beyond Jupyter: Using Google Colab for Robot Programming

Abstract

In recent years, the Jupyter interactive programming environment has become extremely popular as a teaching tool in many scientific and technical fields, including robotics. Jupyter allows you to integrate documentary materials with executable and modifiable code examples. In this work we propose the use of Google Colab, a Jupyter-based tool, to improve the user and programmer experience, both for the teacher who prepares the material and for the student who uses it. These enhancements are primarily based on the integration of Google Drive and GitHub, allowing storage and distribution of initial notebook templates from the cloud, and backups of student-modified notebooks to their own Google Drive accounts. All this without the need to store the notebooks on the local computers where Jupyter is executed, be they real robots or computers where simulations are executed. In addition to the description of the workflow using Colab, we will present concrete examples of use in a simulator and a real robot. Also, as an application of the technology, the experience obtained in the use of this platform is shared with first-year engineering students, as well as with the initial training of a team interested in the CEABOT robot competition.

 https://doi.org/10.17979/spudc.9788497498418.0662

Keywords: Educational Robotics, Robot Competitions, Programming

Referencias

[1] Cañas, J. M., Perdices, E., García-Pérez, L., & Fernández-Conde, J. (2020). A ROS-based open tool for intelligent robotics education. Applied Sciences, 10(21), 7419.

[2] Casañ, G. A., & Cervera, E. (2018). The Experience of the Robot Programming Network Initiative. Journal of Robotics, 2018.

[3] Cervera, E. (2018, March). Interactive ROS Tutorials with Jupyter Notebooks. In TRROS@ ERF (pp. 1-11).

[4] Cervera, E., Casañ, G., & Tellez, R. (2017, July). Cloud simulations for RoboCup. In Robot World Cup (pp. 180-189). Springer, Cham.

[5] Cervera, E., & Del Pobil, A. P. (2019). Roslab: sharing ros code interactively with docker and jupyterlab. IEEE Robotics & Automation Magazine, 26(3), 64-69.

[6] Erdoğmuş, A. K., & Yayan, U. (2021, August). Virtual Robotic Laboratory Compatible Mobile Robots for Education and Research. In 2021 International Conference on INnovations in Intelligent SysTems and Applications (INISTA) (pp. 1-6). IEEE.

[7] Fischer, T., Vollprecht, W., Traversaro, S., Yen, S., Herrero, C., & Milford, M. (2021). A RoboStack Tutorial: Using the Robot Operating System Alongside the Conda and Jupyter Data Science Ecosystems. arXiv preprint arXiv:2104.12910.

[8] Kluyver, T.; Ragan-Kelley, B.; Pérez, F.; Granger, B.E.; Bussonnier, M.; Frederic, J.; Kelley, K.; Hamrick, J.B.; Grout, J.; Corlay, S.; et al. (2016). Jupyter Notebooks-a Publishing Format for Reproducible Computational Workflows; IOS Press: Clifton, NJ, USA; pp. 87–90.

[9] Nelson, M. J., & Hoover, A. K. (2020, June). Notes on using Google Colaboratory in AI education. In Proceedings of the 2020 ACM conference on innovation and Technology in Computer Science Education (pp. 533-534).

[10] Pérez, F., & Granger, B. E. (2007). IPython: a system for interactive scientific computing. Computing in science & engineering, 9(3), 21-29.

[11] Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R., & Ng, A. Y. (2009, May). ROS: an open-source Robot Operating System. In ICRA workshop on open source software (Vol. 3, No. 3.2, p. 5).

[12] Ruiz-Sarmiento, J. R., Baltanas, S. F., & Gonzalez-Jimenez, J. (2021). Jupyter Notebooks in Undergraduate Mobile Robotics Courses: Educational Tool and Case Study. Applied Sciences, 11(3), 917.

[13] Wolfram, S. (1991). Mathematica: a system for doing mathematics by computer. Addison Wesley Longman Publishing Co., Inc..