Control autónomo de robots de aplicación agrícola con Plexil.

Conference Paper · November 2016

CITATION
1

READS
288

2 authors:

Héctor Fabio Cadavid
Netherlands eScience Center
36 PUBLICATIONS **149** CITATIONS
SEE PROFILE

Javier Chaparro
Escuela Colombiana de Ingeniería
16 PUBLICATIONS **115** CITATIONS
SEE PROFILE

AADECA 2016 - Semana del Control Automático - 25º Congreso Argentino de Control Automático
1 al 3 de Noviembre de 2016 - Buenos Aires, Argentina.

CONTROL AUTÓNOMO DE ROBOTS DE APLICACIÓN AGRÍCOLA CON PLEXIL

Héctor Fabio Cadavid Rengifo [*]
Javier Alberto Chaparro Preciado [*]

[*] *Escuela Colombiana de Ingeniería -*
hector.cadavid@escuelaing.edu.co,
javier.chaparro@escuelaing.edu.co

Resumen: Aunque el campo de la robótica no ha tenido aún gran impacto en países como Colombia, las posibilidades que ofrece para ayudar a resolver algunos de sus problemas, como el daño ambiental que está causando la agricultura artesanal al medio ambiente, siguen siendo enormes. Este artículo presenta los resultados de un proyecto de investigación que busca validar posibilidad de incorporar la tecnología de control Plexil -orientada a sistemas de misión crítica- en la constricción de soluciones robóticas robustas de aplicación agrícola. Se presentan los resultados de un caso de estudio real y los productos derivados del mismo.

Palabras Claves: Robótica, Agricultura, Plexil, Automatización, Programación Síncrona

1. INTRODUCCIÓN

Se ha reportado, desde hace ya varios años, que en Colombia la aplicación rudimentaria de pesticidas y fertilizantes está causando un severa carga ambiental, poniendo en riesgo -aparte de la sostenibilidad del sector agropecuario- recursos tan vitales como el agua (González *et al.* (2012)). En el mundo, esta misma situación ha impulsado el desarrollo de nuevas técnicas de gestión agrícola en las que -con apoyo de la tecnología-, se busca maximizar la efectividad de los insumos agrícolas, a la vez que se minimice el impacto de los mismos. Estas técnicas, que se enmarcan en el concepto de *Agricultura de Precisión*, van desde la aplicación controlada de pesticidas(Jones *et al.* (2016)), el control genético de las plantas (Ruckelshausen *et al.* (2009)) hasta la cosecha automática en plantaciones (Kondo *et al.* (2005)).

Las técnicas de Agricultura de Precisión soportadas por robot autónomos presentan grandes retos en diferentes aspectos, tales como la mecánica, la sensórica y el control autónomo. Aunque los dos primeros son fundamentales -ya que de éstos depende la adecuada percepción y acción sobre el terreno por parte del robot-, el no contar con un control autónomo robusto, puede llevar a comportamientos incorrectos o impredecibles. Esto puede comprometer no solo la correcta ejecución de la tarea (es decir, que se cumpla al pie de la letra con su especificación), sino la integridad física de los componentes del robot, lo que puede acarrear pérdidas económicas significativas.

Considerando lo anterior, el presente artículo muestra los resultados de un trabajo de investigación de los programas de Ingeniería de Sistemas e Ingeniería Electrónica de la Escuela Colombiana de Ingeniería, en el que se propone un modelo de control robusto para robots de aplicación agrícola basados en el paradigma de programación síncrono/reactivo. Para tal fin, se evaluó la viabilidad y los beneficios de incorporar Plexil, un lenguaje experimental desarrollado por la NASA para sistemas de misión crítica, en la capa de control de un prototipo robótico de Agricultura de Precisión.

El artículo está estructurado de la siguiente manera: En la sección 2 se da una breve reseña de la tecnología Plexil, y de las características de los lenguajes síncronos. En la sección 3 se

describe cómo se hizo la integración del entorno Plexil con el prototipo robótico y el entorno de programación. En la sección 4 se ilustra el caso de estudio con el que se evaluó el lenguaje Plexil en un problema de Agricultura de Precisión. En la sección 5 se discuten las oportunidades que brindan proyectos como el acá descrito, a la luz de otros proyectos relacionados (orientados a simplificar y mejorar el desarrollo de robots autónomos). Finalmente, en la sección 6 se dan las conclusiones y la proyección que tiene este trabajo hacia el futuro.

2. PLEXIL Y EL EJECUTIVO UNIVERSAL

PLEXIL (*Plan Execution Interchange Language*) es un lenguaje síncrono/reactivo desarrollado por la NASA para soportar operaciones autónomas en misiones no tripuladas. Los programas en PLEXIL, llamados 'planes', especifican qué acciones deben ser ejecutadas por un sistema autónomo bien sea como parte de su operación convencional, o como respuesta a eventos dados por el entorno (por ejemplo, aquellos generados por los sensores del robot). El componente de software que ejecuta los planes Plexil -normalmente embebido en el robot- es conocido como Ejecutivo Universal.

Los planes Plexil consisten en un conjunto jerárquico de nodos que representan una descomposición de tareas. Los nodos hoja representan operaciones primitivas (asignación de variables, ejecución de comandos, etc), mientras que los nodos intermedios definen una jerarquía de control de sus descendientes (e.g., si son ejecutados secuencial o concurrentemente).

Cada nodo está equipado con un conjunto de condiciones que activan su ejecución (las cuales pueden hacer referencia a valores de las variables locales, o eventos externos), y pueden -en cualquier momento- ofrecer información acerca de su estado de ejecución: *inactive, waiting, executing, iterationend, failing, finishing* o *finished* o -en caso de que el nodo haya terminado su tarea-, el estado de terminación del mismo: *success, skipped,* o *failure.*

Cuando los eventos son reportados por el ambiente externo (sensores, temporizadores, etc), los nodos activados por dichos eventos son ejecutados concurrentemente, actualizando variables locales en el proceso -lo cual puede, a su vez, conducir al cambio de estado de otros nodos-. Sin embargo, a pesar de que varios eventos podrían ser reportados simultáneamente, dado que la semántica de Plexil -y por ende el Ejecutivo Universal- son diseñados considerando la hipótesis de sincronía (Potop-Butucaru *et al.* (2005)), todas las operaciones que se realicen en paralelo estarán perfectamente sincronizadas, por lo que no se estarán sujetas al *interleaving.*

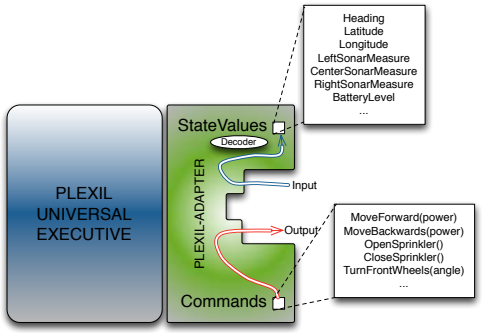

Fig. 1. Adaptador de propósito general, con una interfaz de I/O abstracta.

3. ARQUITECTURA DE CONTROL DEL ROBOT

Los planes Plexil interactúan con su entorno de dos maneras. Primero, con la recepción de eventos tales como el nivel de batería, la actualización de la posición relativa o las mediciones de los sensores de proximidad, y segundo, mediante la solicitud de acciones (o comandos), tales como girar las ruedas, encender el motor o detenerse. Sin embargo, dado que el Ejecutivo Universal de Plexil está diseñado para ser independiente de la plataforma en la que se use, es necesario desarrollar un adaptador le permita a éste saber:

- Cómo identificar eventos -para activar Nodos del plan- a partir de los datos transmitidos por los sensores del robot.
- Cómo convertir los comandos invocados por el plan Plexil en peticiones a los microcontroladores de los motores y demás actuadores.

Dicho adaptador debe entonces definir, por un lado, qué variables ofrece el entorno del robot (es decir, qué sensores hay disponibles), y por el otro, qué comandos se pueden invocar.

Con el fin de desacoplar el adaptador desarrollado y el entorno 'real' con el cual éste interactuaría -teniendo en mente que éste podría ser el robot real o un entorno simulado-, se planteó el modelo de la figura 1. En este, aunque la totalidad de la lógica de la lectura de señales y la lógica de invocación de acciones asociadas a comandos está implementada, el origen y destino -de las señales y de los comandos respectivamente- queda abierto a cualquier entorno.

Para efectos del proyecto, dos diferentes interfaces de entrada/salida fueron desarrolladas:

- Interfaz que maneja la entrada y la salida a partir de tuberías (pipes) UNIX. Esta interfaz permite probar la ejecución de planes fuera del entorno del robot real, usando

la entrada (STDIN) y salida (STDOUT) estándar del sistema operativo para el envío de peticiones y la recepción de eventos del un plan Plexil. Sobre la base de esta interfaz, se desarrolló de un entorno de validación y prototipado rápido, cuya arquitectura se presenta en la figura 2, y cuya descripción más detallada puede consultarse en (Cadavid and Chaparro (2016)).

- Interfaz que maneja la entrada y la salida usando los parámetros de conexión GPIO de la board Intel Edison y el protocolo RS-232, el cual es usado por el prototipo robótico real, tal como se muestra en la figura 3.

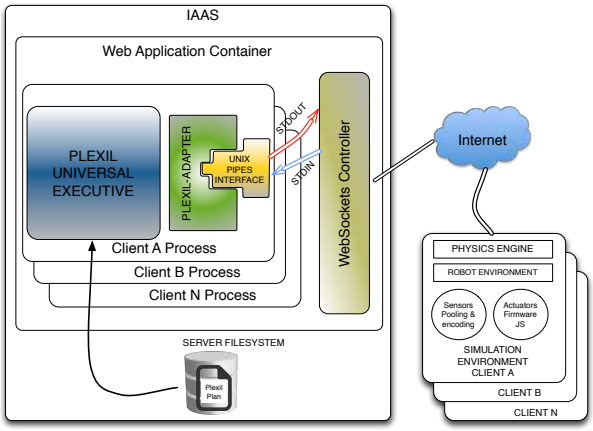

Fig. 2. Esquema de alto nivel con la arquitectura del entorno de prototipado y simulación. En lugar del adaptador I/O de tipo RS-232, se utiliza el adaptador de tubos UNIX conectado con un simulador corriendo en un cliente Web a través de WebSockets.

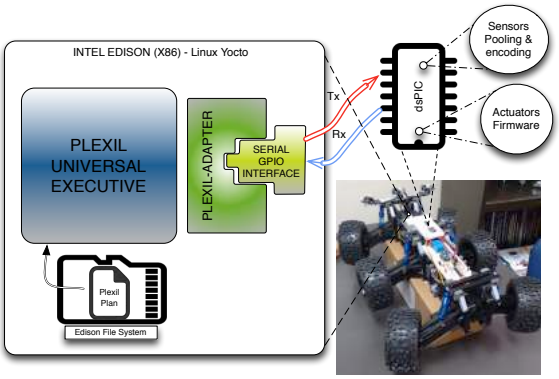

Fig. 3. El adaptador de Plexil usando la versión de la Interfaz de comunicaciones que permite la conexión con el *firmware* del robot, a través del protocolo RS232.

4. CASO DE ESTUDIO

Las aplicaciones avanzadas de Agricultura de Precisión orientadas a la dosificación controlada de pesticidas y fertilizantes requieren no solo de sensores que permitan al robot ubicarse en su entorno, sino sensores que le permitan identificar en tiempo real las características del suelo (tales como su acidez y su conductividad eléctrica) para la toma de decisiones.

Aunque este tipo de aplicaciones se salen del alcance del proyecto -por el tipo de sensores requeridos-, como prueba de concepto se plantea una aplicación que, aunque menos sofisticada, cumple con el propósito de validar tanto el modelo de integración del control basado en Plexil, como el comportamiento del robot con los planes de automatización planteados.

4.1 Escenario

Como escenario se plantea un espacio llano, donde los únicos obstáculos son dos muros laterales. Como objetivo, se plantea lograr que el robot aplique de manera uniforme -haciendo aspersiones equidistantes en cada línea de aplicación- una determinada sustancia en un espacio cerrado.

La tarea planteada en este caso de estudio considera como restricciones y condiciones:

(1) Como el terreno y la alineación imperfecta de las ruedas del robot no garantizan un desplazamiento continuo en línea recta, el robot debe estar en capacidad de corregir continuamente el rumbo, apoyado por las lecturas del campo magnético terrestre notificadas por la IMU.
(2) Aunque no habrá obstáculos intermedios, el robot debe detectar los muros laterales para saber cuando maniobrar e iniciar la aspersión en la siguiente línea horizontal.
(3) Al detectarse los muros laterales, el robot debe realizar un giro lo más cerrado posible y recuperar la trayectoria horizontal lo antes posible para iniciar una nueva secuencia de aspersiones en el sentido contrario.
(4) Dado que el robot aún no cuenta con un mecanismo de geo-posicionamiento preciso, y dado que el terreno (que es plano) permite lograr rápidamente una velocidad constante, para la prueba de concepto la dosificación equidistante se basará en mediciones del tiempo.
(5) Por la restricción anterior, se debe garantizar que no se abra el dosificador mientras el robot esté maniobrando para ubicarse en la siguiente línea de aplicación.

4.2 Plan Plexil de automatización

La figura 4 muestra, esquemáticamente, el plan Plexil planteado para la el escenario y la tarea antes descritos. Como se observa en la misma, del nodo raíz del plan descienden (con una jerarquía de ejecución concurrente) los nodos:

- DoserTime: Nodo que, periódicamente, detiene el robot, abre el dispensador momentáneamente, y luego reactiva el movimiento.
- DirectionLeftFix/DirectionRightFix: Nodos que hacen cambios momentáneos de dirección cuando se detecta una pérdida en el rumbo (variable 'heading'), siempre que el robot no esté en una maniobra de prevención de colisión (CollisionHandler.state).
- HeadingUpdate: Jerarquía de nodos concurrentes encargados de calcular la dirección. Cada uno de estos nodos está asociado a una de las diferentes condiciones de transformación (planteadas en Caruso (1995)) de lecturas de campo magnético a un valor de dirección, en grados respecto al norte magnético:
  (a) $(y > 0) \rightarrow heading = 90 - atan(\frac{x}{y}) * \frac{180}{\pi}$
  (b) $(y < 0) \rightarrow heading = 270 - atan(\frac{x}{y}) * \frac{180}{\pi}$
  (c) $(y = 0, x < 0) \rightarrow heading = 180$
  (d) $(y = 0, x > 0) \rightarrow heading = 0.0$
- ColisionHandler: Jerarquia de nodos secuenciales con los pasos para realizar el giro cuando se detecta el muro lateral. Según la dirección que lleve el robot, se activará uno u otro nodo con la respectiva secuencia.

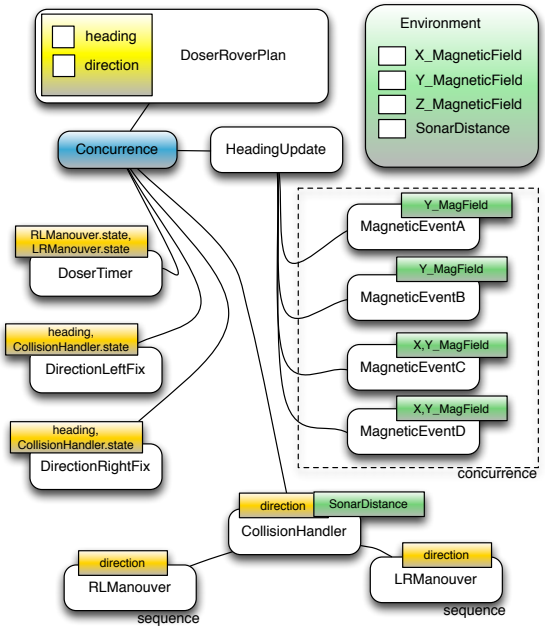

Fig. 4. Representación esquemática del plan planteado para el caso de estudio. Sobre cada nodo se indica de qué variables (en amarillo) o de qué eventos del entorno (verde) depende su condición de inicio. Código completo disponible en: https://git.io/vPrYN

4.3 Resultados preliminares

El Plan Plexil descrito anteriormente fue implementado y validado, a través de varias iteraciones, haciendo uso del entorno de prototipado y simulación descrito anteriormente, tal como se muestra en la figura 5. Como métricas para la validación del comportamiento de plan (utilizada a través de dichas iteraciones), se utilizó -tomando el terreno como un plano cartesiano- el promedio de las desviaciones estándar de las posiciones en Y de cada punto aplicado (dv), y el promedio de las desviaciones estándar de las distancias en X entre puntos adyacentes (dh), donde m es el número de líneas aplicadas, y n el número de puntos aplicados en cada línea :

(a) $dv = \frac{\sum_{i=1}^{m} \sigma_{yi}}{m}$

(b) $dh = \frac{\sum_{i=1}^{m} \sigma_{\Delta xi}}{m}, \sigma_{\Delta xi} = \sqrt{\frac{\sum_{i=1}^{n} (\Delta x - \mu_{\Delta x}))^2}{n}}$

Como se observa en la figura 6, a través del ajuste del algoritmo y sus parámetros (velocidad, pausas entre cambios de dirección, ángulos de giro, etc), se logró llegar a una versión del mismo que -en la simulación- logró unos promedios para las desviaciones estándar que van entre dos y tres unidades (3 a 9 centímetros, en un terreno de 20 metros de ancho, teniendo en cuenta la escala de la simulación). Por otro lado, aunque se comprobó que el robot real ejecutaba los comandos en los mismos tiempos que la simulación, está en proceso el desarrollo el desarrollo del espacio físico que permita probar, medir y comparar -respecto a su simulación- éste y otros planes de automatización, lo cual dará elementos para ajustar tanto la estratega de dichos planes como el modelo de simulación.

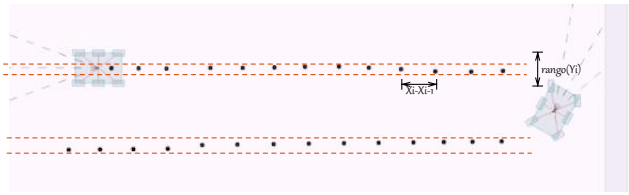

Fig. 5. Comportamiento del plan evaluado a través del entorno de prototipado rápido y simulación desarrollado como parte del proyecto.

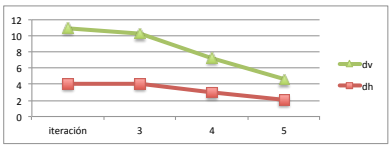

Fig. 6. Evolución del desempeño del plan a través de ajustes a los parámetros del mismo.

5. TRABAJOS RELACIONADOS Y DISCUSIÓN

5.1 Frameworks para la automatización de robots

En los últimos años se han desarrollado frameworks para el desarrollo de sistemas robóticos, tales como PLAYER (Collett *et al.* (2005)), CARMEN (Montemerlo *et al.* (2007)), y ROS (Quigley *et al.* (2009)) -siendo este último el de mayor acogida-, los cuales permiten un proceso de

desarrollo más simple y menos propenso a errores mediante la estandarización de las comunicaciones con sus componentes electro-mecánicos.

Aunque dichos frameworks permiten la integración con entornos de simulación como Gazebo (Koenig and Howard (2004)) o Webots (Michel (2004)) para validar el comportamiento esperado del robot frente a un conjunto de escenarios, difícilmente, al estar soportados por lenguajes de programación imperativos/asíncronos (basados en hilos o en eventos) podrán ser verificados formalmente. Mas allá de esto, un problema de la programación asíncrona convencional es que deja en manos del desarrollador tareas críticas como la identificación de posibles condiciones de carrera, y la implementación de mecanismos correctos y eficientes de sincronización que las prevengan. Por otro lado, y como consecuencia de lo anterior, las labores de desarrollo, y depuración de este tipo de programas de control puede resultar en exceso complejas, lo que puede conducir a que los mismos evolucionen en un proceso de ensayo-error.

Para ilustrar lo anterior, vale la pena comparar la aproximación a una parte del caso antes descrito, con un modelo de programación basado hilos y eventos asíncronos como el usado normalmente en ROS.

Por ejemplo, la aplicación del pesticida requeriría, por un lado, un hilo que periódicamente aplique el pesticida, salvo cuando esté en curso la maniobra de cambio de línea:

```
...
ros::Rate r(10);
while (should_continue)
{
  if (!manouvering){
    stop();
    applyPesticide();
    start();
  }
  ros::spinOnce();
  r.sleep();
}
...
```

Por otro lado, lo anterior requeriría definir un *callback* que se ejecute cuando se dé el evento de una medición de los sensores de proximidad que implique una colisión cercana:

```
...
void collisionEventCallback(msg)
{
  val=evalEvent(msg);
  if (val < MIN){
    manouvering=true;
  }
  op1()
  op2()
  //inverse direction after manoeuvring
  direction = direction * -1;
}
...
```

Este programa, como se sabe, presenta una condición de carrera con la variable 'manouvering', pues se puede dar el caso de que aún con el evento de que el sensor de distancia arroje una medida menor a MIN, el hilo que aplica el pesticida consulte el estado de la variable justo antes de ser modificada.

Al comparar el segmento del plan Plexil que maneja este aspecto (aspersión y maniobra de cambio de ruta), se puede ilustrar por qué en muchos casos la programación sincronía de más alto nivel puede resultar -además de más robusta-, más clara y simple:

```
DoserRoverPlan:{
  Concurrence{
  ...
    DoserTimer{
      Skip RLManouver.state=EXECUTING
                || LRManouver.state=EXECUTING
        Repeat true;
        Wait (delay), tolerance;
        stop();
        SynchronousCommand applyPesticide();
        start();
    }
    RLManouver:{
      Start direction==-1 && collisionEvent;
      //Left to right manouver
    }
    LRManouver:{
      Start direction==1 && collisionEvent;
      //Right to left manouver
    }
  }
}
```

Como se observa, para este mismo escenario se pueden definir dependencias a nivel de estados de los nodos (y no solo de variables compartidas), de manera que -en este caso-, el nodo de dosificación siempre permanecerá en espera mientras cualquiera de los nodos que realiza las maniobras de cambio de línea estén activos.

5.2 Aplicaciones de Plexil en otros contextos

Además de las aplicaciones experimentales documentadas de Plexil hechas por parte de la NASA, tales como el control del robot de exploración K10, y el manejo de algunas tareas de automatización de la Estación Espacial Internacional (Vandi Verma (2006)), la literatura presenta casos de adaptación de Plexil a modelos de control y automatización de robots muy especializados, tales como el *Planning Domain Definition Language*/PDDL (Munoz *et al.* (2010)) o los basados en sistemas multiagente (Ziafati (2014)). Por otro lado, y gracias a que Plexil desde un principio fue diseñado teniendo en mente facilitar la verificación y validación formal, en la literatura se encuentra desde modelos para validación mecánica (Dowek *et al.* (2010); Brat *et al.* (2008)) y automática (Rocha *et al.* (2012)) de planes escritos en este lenguaje hasta frameworks y entornos de validación que los incorpora (Rocha *et al.* (2012); Biatek *et al.* (2014)).

Todos estos avances que desde la informática teórica se han dado al rededor de lenguajes como Plexil, justifican aún más la oportunidad de incorporar este tipo de tecnologías en el sector académico e industrial de la robótica.

6. CONCLUSIONES Y TRABAJO FUTURO

A través de este proyecto se comprobó la viabilidad de integrar entornos de control de alto nivel como Plexil, hasta ahora tratados casi exclusivamente de forma teórica, en aplicaciones industriales de alto impacto para Colombia. Esto presenta un panorama muy promisorio para futuros proyectos de investigación, en donde se podría contribuir a potenciar la industria de la robótica nacional mediante el trabajo interdisciplinario desde la Ingeniería Electrónica, la Ingeniería de Software y la Informática Teórica.

Como trabajo futuro, se plantea contribuir a proyectos de Software Libre muy ampliamente difundidos como ROS, con la integración de las herramientas acá presentadas dentro de su entorno. Con esto, se buscará tener un impacto sobre una comunidad de profesionales y entusiastas de la robótica que pueda también retroalimentar este y futuros proyectos en el área.

Los autores agradecen el soporte de la Escuela Colombiana de Ingeniería en el desarrollo de este proyecto, al cual apoyó con recursos de la convocatoria Interna de Investigación 2014/2015.

REFERENCIAS

Biatek, Jason, Michael W Whalen, Mats PE Heimdahl, Sanjai Rayadurgam and Michael R Lowry (2014). Analysis and testing of plexil plans. In: *Proceedings of the 2nd FME Workshop on Formal Methods in Software Engineering*. ACM. pp. 52–58.

Brat, G, M Gheorghiu, D Giannakopoulou and C Pasareanu (2008). Verification of plans and procedures. In: *Aerospace Conference, 2008 IEEE*. IEEE. pp. 1–8.

Cadavid, H.F. and J.A. Chaparro (2016). Hardware and software architecture for plexil-based, simulation supported, robot automation. In: *Proceedings of the IEEE Colombian Conference on Robotics and Automation - CCRA 2016*.

Caruso, Mike (1995). Compass heading using magnetometers. *Honeywell Application Note AN*.

Collett, Toby HJ, Bruce A MacDonald and Brian P Gerkey (2005). Player 2.0: Toward a practical robot programming framework. In: *Proceedings of the Australasian Conference on Robotics and Automation (ACRA 2005)*. p. 145.

Dowek, Gilles, César Munoz and Camilo Rocha (2010). Rewriting logic semantics of a plan execution language. *arXiv preprint arXiv:1002.2872*.

González, Juan Pablo, Brigid Pacheco, Fernanda Viasus and Karen Ayala (2012). Movilidad de pesticidas en aguas superficiales empleados en agricultura y riesgos para la salud humana en la zona centro del departamento de boyacá-colombia. *L'esprit Ingénieux*.

Jones, Austin, Usman Ali and Magnus Egerstedt (2016). Optimal pesticide scheduling in precision agriculture. In: *2016 ACM/IEEE 7th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE. pp. 1–8.

Koenig, Nathan and Andrew Howard (2004). Design and use paradigms for gazebo, an open-source multi-robot simulator. In: *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*. Vol. 3. IEEE. pp. 2149–2154.

Kondo, Naoshi, Kazunori Ninomiya, Shigehiko Hayashi, Tomohiko Ota and Kotaro Kubota (2005). A new challenge of robot for harvesting strawberry grown on table top culture. In: *2005 ASAE Annual Meeting*. American Society of Agricultural and Biological Engineers. p. 1.

Michel, Olivier (2004). Webotstm: Professional mobile robot simulation. *arXiv preprint cs/0412052*.

Montemerlo, Michael, N Roy, S Thrun, D Haehnel, C Stachniss and J Glover (2007). Carmen, robot navigation toolkit. *Retrieved June*.

Munoz, Pablo, Maria D R-Moreno and Bonifacio Castano (2010). Integrating a PDDL-based planner and a PLEXIL-executor into the ptinto robot. In: *Trends in Applied Intelligent Systems*. pp. 72–81. Springer.

Potop-Butucaru, Dumitru, Robert de Simone and Jean-Pierre Talpin (2005). The synchronous hypothesis and synchronous languages. *The embedded systems handbook* pp. 1–21.

Quigley, Morgan, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler and Andrew Y Ng (2009). Ros: an open-source robot operating system. In: *ICRA workshop on open source software*. Vol. 3. Kobe, Japan. p. 5.

Rocha, Camilo, Hector Cadavid, Cesar Munoz and Radu Siminiceanu (2012). A formal interactive verification environment for the plan execution interchange language. In: *Integrated Formal Methods*. Springer. pp. 343–357.

Ruckelshausen, A, P Biber, M Dorna, H Gremmes, R Klose, A Linz, F Rahe, R Resch, M Thiel, D Trautz et al. (2009). Bonirob–an autonomous field robot platform for individual plant phenotyping. *Precision agriculture* **9**(841), 1.

Vandi Verma, Ari Jonsson, Corina Pasareanu Michael Iatauro (2006). Universal executive and PLEXIL: Engine and language for robust spacecraft control and operations. In: *Proceedings of the American Institute of Aeronautics and Astronautics Space Conference (2006)*.

Ziafati, Pouyan (2014). Plexil-like plan execution control in agent programming. *Plexil-Like Plan Execution Control in Agent Programming*.