International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 3 (2017) pp. 311-316
© Research India Publications. http://www.ripublication.com

Bringing ROS to Agriculture Automation: Hardware Abstraction of Agriculture Machinery

Sami Salama Hussen Hajjaj[1] and Khairul Salleh Mohamed Sahari [2]

Centre for Advanced Mechatronics and Robotics Universiti Tenaga Nasional (UNITEN), 43000 Kajang, Selangor, Malaysia.

[1]ORCID 0000-0003-1198-4802, [2]ORCID 0000-0002-3838-501X

Abstract
Concerns over food security in recent years have led to increased interest in agriculture automation, researchers around the world are looking for new and innovative ways to automate agriculture and improve its efficiency. Some researchers attempt to automate existing agriculture machinery, while others attempt to introduce robotics into agriculture, with both approaches have merit and potential. However, agriculture automation brings about vast engineering challenges; networking over wide open fields, navigation over rough soil, localization in a field of greenery, image processing hampered by loss of illumination due to weather, etc. and due to these challenges, agriculture automation remain impractical and uneconomical. The Robot Operating System (ROS) offers potential solutions to these challenges, and when combined with other open-source technologies, such as android or IoT, the potential is even higher. But before these benefits could be realized, agriculture machinery and robots must be made ROS-ready. This paper outlines and discusses the hardware/software steps needed to achieve exactly that. First by detailing the processing of making existing agriculture machinery ROS-ready, then by detailing the process of modifying ROS-ready robots to perform Agriculture tasks. The procedure is discussed in details with real examples, and references to ROS's resources and material. Taking another step closer towards implementing practical agriculture automation

Keywords: Agriculture automation, Agriculture robotics, Robot Operating System (ROS), Hardware abstraction, Internet of Things, IoT, Technology convergence, Robot software, Robotics, android, Open-Source technologies

INTRODUCTION

In recent years, concerns over food security have led to increased interest in agriculture automation; global spending on agriculture robotics is rising exponentially, with organizations worldwide working on agriculture automation and on developing agriculture robots [1–5]. But in the age of Internet of Things (IoT) and Technology Integration, new and exciting solutions arise [6, 7].

This paper is part of an ongoing project; the AgribotWorld project, which aims at implementing practical agriculture automation, by utilizing open-source technologies and platforms.

The Robot Operating System (ROS) is a leading open-source plat-form for robot software development. It has experienced tremendous growth in recent years due to its many great features; such as Software Sharing, Distributed Computing, Hardware support, and Collaboration with other platforms [8–10]. ROS provides a library of reliable and free software for robot operations; such as navigation, control, sensor data processing, and others. ROS also provides mechanisms for extending and customizing this software library. Furthermore, ROS supports more than 170 various types of off-the-shelve robots, as well as more than 100 types of sensors, making ROS ideal for hardware customization [9]. Through Distributed Computing, ROS software can be spread across a group of networked machines, which opens the door for Group Robotics, Cloud Robotics, Human Robot Interaction (HRI), and much more [11].

CHALLENGE: ROS-AGRICULTURE

Automated agriculture machines would need to operate in a unique environment; in open fields, on rough terrains, under the elements, and in collaboration with human workers. This environment brings about many agriculture-specific challenges, such as networking over vast open fields, navigation on rough terrain, localization in field of open greenery, effects of weather on sensor data acquisition, establishing effective HRI, software integration, and others [12]. Using conventional methods, the technological and monitory costs of tackling these challenges could be too prohibitive to many organizations, rendering the whole idea of agriculture automation impractical [13].

ROS could provide reliable and cost-effective solutions to the challenges of agriculture automation, and when integrated with other open-source platforms, such as android, IoT, and other technologies, the potential is even higher [14, 15]. But before this potential is realized, agriculture machinery and agriculture robots must be made ROS-ready, in other words, ROS must be incorporated in agriculture automation.

This paper outlines the detailed process of incorporating ROS in agriculture automation, be it for existing agriculture machinery, or for new agriculture robots. Taking another step closer towards implementing practical agriculture automation.

REVIEW: AGRICULTURE AUTOMATION

Agriculture automation can be achieved either by automating agriculture machinery, or by introducing new automated systems, namely robotics, into agriculture operations.

Automating agriculture machinery

This option might be appealing for many reasons; workers and manufacturers are already familiar with existing technologies, in-vestments already committed to current machinery so investing in new ones could be difficult, especially when current machines have higher capacity (power, speed, maximum weight, etc.) over the smaller mobile robots. But challenges also arise; such as in-stalling the needed sensors and actuators to automate these ma-chines, establishing a wireless network over vast open fields, and implementing an effective HRI framework between these ma-chines and agriculture workers, and other challenges [13]. Some researchers recognized these challenges, and developed complete systems that attempted to automate current agriculture operations and machinery [16], such as the RHEA project shown in figure 1 below.

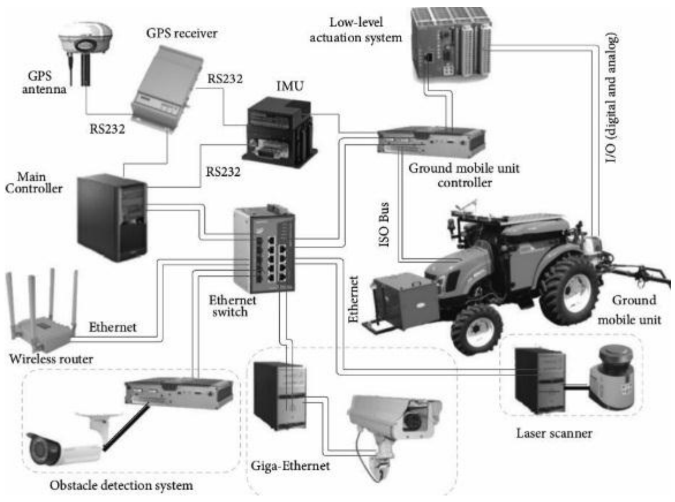


Figure 1: Automating agriculture machinery [16]

The figure shows the hardware components needed to automate just one tractor, which demonstrates another challenge: Soft-ware Integration. Components shown in the figure are written in languages and in different platforms, combining them together would require implementing the principles of IoT and Technology Convergence; such as standardization of data types, unification of user input, inter-platform data exchange, and more [17].

Introducing Robotics in Agriculture

This option is also appealing, considering the amazing developments in robotics in the last 20 years, researchers are experimenting robots of various types to perform generic agriculture tasks, such as harvesting, picking, herding, and others [4]. But with that, many agriculture-specific challenges also arise; navigation on rough terrains and in open fields, localization in fields of greenery, effects of daytime illumination on sensor data, and others. Some researchers tackled these issues and introduced many solutions; utilization of GPS and Sonar in navigation in open fields, incorporating the effects of slip (in rough soil) in control algorithms of mobile robots, utilizing Neural Networks to overcome the effects of weather on image processing (and other sensor data acquisition), and much more [12].

Incorporating ROS in Agriculture.

Despite the progress mentioned above, agriculture automation re-main far from being operational; there are no reports of implementing agriculture automation in large scale generic operations, with most solutions mentioned above, while very promising, are still being experimental at best [13].

ROS provides reliable and cost-effective solutions to the challenges of agriculture automation; ROS provides library of robot software for robot operations, networking, HRI, software integration, and others. To utilize these features, ROS needs to be incorporated in agriculture automation. This can be achieved either by extending ROS support to agriculture machinery, such as the tractor shown in figure 1, or by modifying ROS-ready robots to become ROS-ready agriculture robots. Although the "R" in ROS is for Robot, ROS can support any electromechanical system that provides feedback. This includes sensors, actuators, robots, or automated agriculture machinery. So, henceforth, these terms are collectively referred to as the ROS-ready Machine.

REVIEW: THE ROS-READY MACHINE

In ROS, software is divided into Packages, each containing smaller units of software, called nodes, each perform a specific task, as shown in figure 2. The flow of data is managed by the ROS Master, a core ROS package that acts like traffic police; it identifies every node by its unique name, functionality, and the type of data it exchanges with other nodes. Every ROS node or package follows the same set of standards, such as package structure, topic names, message type definitions, etc. This standardization facilitated software integration for ROS; it allowed ROS packages to interact with one another smoothly and effectively.

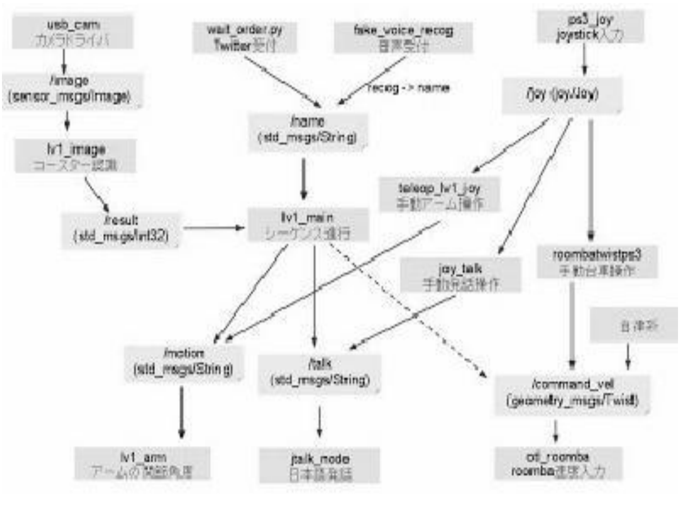


Figure 2: A typical ROS package, [18]

This also facilitated ROS's great features, such as Software Sharing, Distributed Computing, Hardware Support, and Collaboration with other platforms. This also facilitated ROS's exponential growth of the last few years.

Therefore, a ROS-ready machine can be thought of as a mega ROS node; it publishes its data, such as current state, sensor data, and odemetry, and receives control commands, all in accordance with ROS standards and definitions.
The process of making a machine ROS-ready can be thought of as the process of developing the required ROS software that would turn the machine into a ROS node, interacting with other nodes and packages, this process is outlined and discussed next.

MAKING A MACHINE ROS-READY

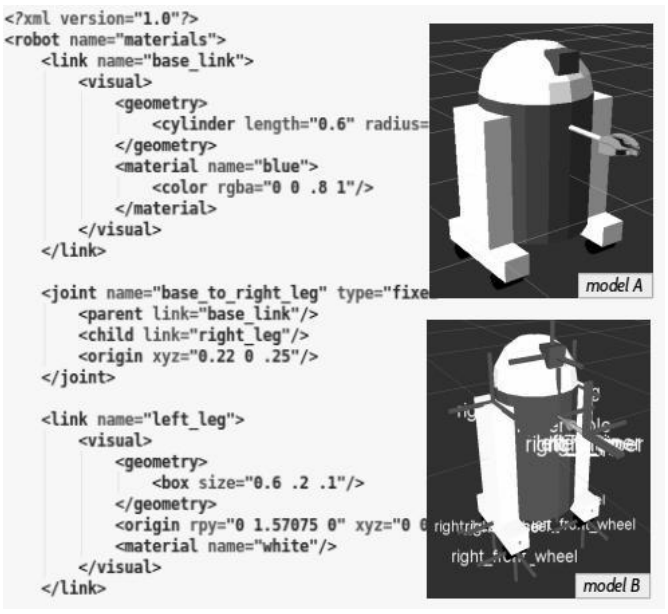
For a machine to interact with other ROS packages and nodes, the following information is needed:

- The Description model
- Sensory data
- Odometry information
- Control interface

Using a collection of ROS package, the above information can be developed and shared with other ROS packages, allowing the machine to publish its current state (model description, sensor data, and odometry) to other ROS packages, and receive control commands from them.

The Description Model

The description model is a file that describes the geometry, physics, and relative relationships of the various components of the robot/machine [19]. This is achieved through the Unified Robot Description Format, or simply the URDF model of the robot/machine, as shown in figure 3.



```
<?xml version="1.0"?>
<robot name="materials">
    <link name="base_link">
        <visual>
            <geometry>
                <cylinder length="0.6" radius=
            </geometry>
            <material name="blue">
                <color rgba="0 0 .8 1"/>
            </material>
        </visual>
    </link>

    <joint name="base_to_right_leg" type="fixe
        <parent link="base_link"/>
        <child link="right_leg"/>
        <origin xyz="0.22 0 .25"/>
    </joint>

    <link name="left_leg">
        <visual>
            <geometry>
                <box size="0.6 .2 .1"/>
            </geometry>
            <origin rpy="0 1.57075 0" xyz="0 0
            <material name="white"/>
        </visual>
    </link>
```

Figure 3: The URDF model of a machine, [19]

The URDF is an XML file that describes the components of the machine; links, joints, grippers, sensors, etc. as well as their geometries, surface structures, colors, masses, inertias, joint/link types, motion types and ranges, etc. [19]. This XML file can then be visualized using ROS's rviz package, as shown in figure 3, or any other open-source visualization platform, such as moveit! or gazebo, which ROS supports and is integrated smoothly with.

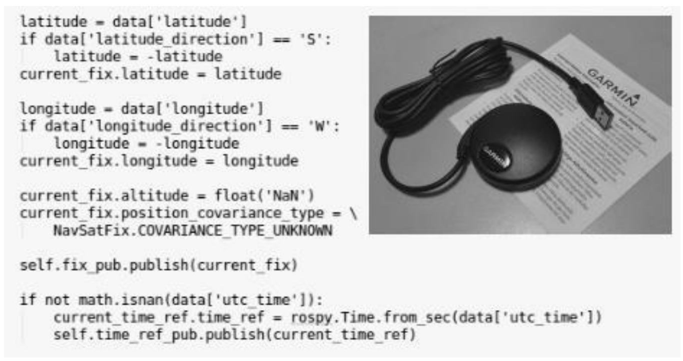
The TF Tree

The tf tree describes the various frames of the robot; such as the base frame, joint frames, arm frame, sensor frame, etc. and their relative relationships, as shown on Model B in figure 3. The tf tree is needed to perform the complex robot kinematics operations; by combining the tf tree with the URDF model of the machine, all kinematics operations, forward or inverse, are then performed by ROS's tf package under the hood, saving developers from the burden of performing these complex operations manually. The detailed process of creating the URDF model and the tf Tree is described in this hands-on tutorial [19].

Sensor Data

To utilize sensor data, sensors need to produce this data as per ROS standards. ROS-enabled sensors capture raw sensor data, re-package it a sensor_msgs/Data message types, then publish these messages to other ROS packages. This allows other ROS packages to capture these sensor data, and utilize it in visualization, data processing, and autonomous control algorithms of the robot/machine This is how ROS is able to support more than 100 sensors of various types; for each of these sensors, a driver pack-age is developed and ready for use with the sensor.

Garmin's GPS sensor is a good example, it is a stand-alone USB receiver that can be added to a robot or a machine, it is ROS-enabled via the nmea navsat driver package, as shown in figure 4 below.



```
latitude = data['latitude']
if data['latitude_direction'] == 'S':
    latitude = -latitude
current_fix.latitude = latitude

longitude = data['longitude']
if data['longitude_direction'] == 'W':
    longitude = -longitude
current_fix.longitude = longitude

current_fix.altitude = float('NaN')
current_fix.position_covariance_type = \
    NavSatFix.COVARIANCE_TYPE_UNKNOWN

self.fix_pub.publish(current_fix)

if not math.isnan(data['utc_time']):
    current_time_ref.time_ref = rospy.Time.from_sec(data['utc_time'])
    self.time_ref_pub.publish(current_time_ref)
```

The process of creating driver packages for any sensors is dis-cussed in details in this hands-on tutorial [20].

Odometry

Odometry is a fundamental component of robot navigation and control. Through odometry, the robot can estimate its current actual position and velocity, and use this information to correct its movements. Without odometry, the machine would have to de-pend entirely on dead-reckoning, which is not reliable and would lead to errors, failures, and accidents. There are different ways of obtaining odometry data; it could be feedback from a controller, captured directly through dedicated odometers, calculated from sensor data, or a combination of these sources. These sources provide different types of odometry information; some are direct values, some are interpretations of sensor data, and others are post-processed odometry data. This data needs to be processed, fused, and then fed back to the controller for navigation and control. Thankfully, in ROS, the process is stream-lined, as it can be seen in figure 5.

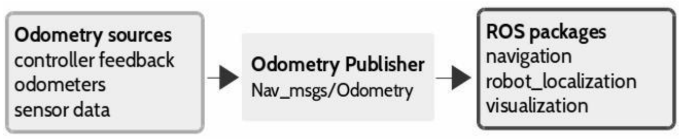


Figure 5: Odometry in ROS

In ROS, Odometry data is packaged in the nav_msgs/Odometry message type. So, all odometry sources must publish their data as per this message type. This includes odometers, sensors, and controller feedbacks. This is then captured by the Odometry publisher package, which fuses this information and publishes a cohesive set of odometry data for other ROS packages, where it could be used for control, visualization, etc.

As discussed earlier, ROS-enabled sensors are made so by developing the required driver packages that publish their data to ROS, part of this data is odometry data as well, and ROS sup-ports nearly 20 pose estimators and odometers. The process of capturing odometry data for a custom machine is discussed in details in this tutorial [21][1].

The robot state publisher

The robot state publisher is a ROS package combines all ma-chine information defined so far (description model, tf tree, sensor data, and odometry), and publishes them all as one set of data, that defines the current state of the machine.
This information is then published to other ROS package, allowing them to capture the robot state, and utilize this information for their functionalities, as shown in figure 6.

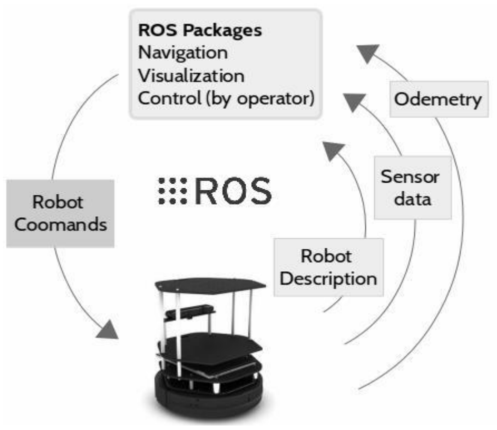


Figure 6: a ROS-enabled machine

[1] Assuming Odometry sources (controller feedback, odometers, sensors) were already setup as discussed in previous section (Sensor data)

As seen in the figure, the machine publishes out state to other ROS packages, who utilize this data in their algorithms. At the same time, the machine also receives control and motion commands from these packages, all in accordance with ROS standards, making the machine a ROS-ready machine.

Controller interface

A ROS controller is the opposite of a ROS sensor driver; it is a Subscriber that captures incoming control messages, extract data from them, and uses these values to actuate the robot/machine.
The first step in creating this controller is to define the required ROS message type containing control commands, such as the geometry msgs/Twist, which defines the desired linear and angular speeds of the mobile robot. Other information include the required ROS topic, and how the data is extracted from the messages, and used to actuate the robot/machine. This allows other ROS packages intending to control the machine to structure their publishers according to this subscriber; such as its topic, message types, and expected data, as shown in fig 6.

The ros control pacakge

Controlling machines to perform elaborate tasks (such as agriculture) is a multi-layered process; from actuator/motor control, to motion and joint movements control, and to finally task (such as agriculture) control. Developing such a multi layered control algorithm from scratch could be a challenging endeavour.

Through ROS's ros control package, shown in figure 7, the process is greatly streamlined and simplified.

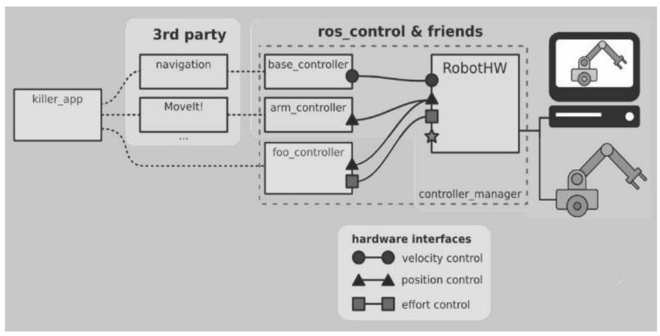


Figure 7: The ros control package, [22]

ros- control provides a management system that combines several ROS packages that perform varying levels of machine control; such as Robot HW for hardware actuation, base controller for mobile robot control, arm controller for robot arm control, and navigation and moveit! for complex motion planning. As shown in the white boxes in figure 7.

Furthermore, the package allows developers to add their custom controllers (for agriculture) to this structure, such as the foo controller, highlighted in a blue boxe in figure 7.
The ros control package showcases the real power of ROS and its features of Software sharing and Re-use; developers need

only to create their custom controllers (for agriculture) by extending and building on ROS's existing control structure, rather than building the whole control structure from scratch.

MODIFYING A ROS-READY ROBOT

Modifying a ROS-ready robot to perform agriculture tasks involves three steps; adding the necessary hardware to enable the robot to perform the agriculture tasks, updating is ROS pack-ages to reflect these changes, and developing the required custom packages to allow it to perform these tasks.

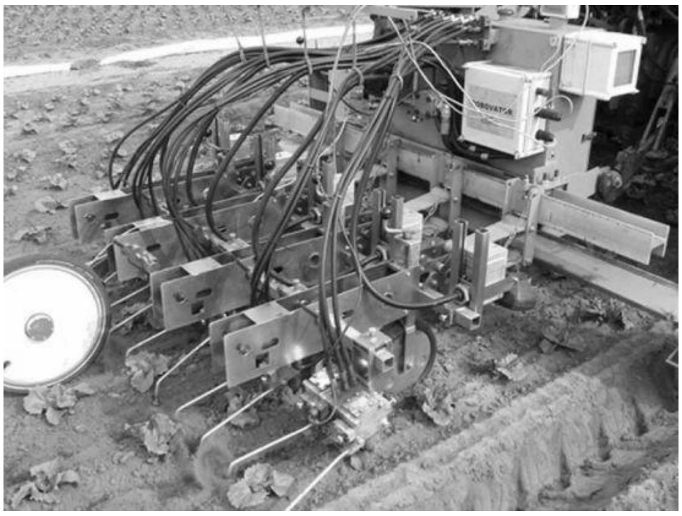


Figure 8: A robotic Weeder, [4]

For example, the robot shown in figure 8 requires hooks, shovels, actuators, sensors, and other components to be added so the robot can perform this task. GPS sensors could be added for outdoor navigation. Manipulators and grippers could be added for picking, harvesting, or other similar tasks. These changes required updates in the robot description model, sensor drivers, and controller interfaces, as shown in table 1.

Table 1: Hardware changes of ROS-ready machine

Hardware changes	ROS software changes
adding rigid link	robot model
adding new sensor	sensor driver & robot model
adding new actuator	controller interface & odom

For example, if a mobile robot is to be modified to become a fruit picker, a robotic arm needs to be installed, along with its actuators, sensors, and controller. This would require extensive ROS updates. First, the URDF model is updated to include the description of the arm, using ROS's Arm/Hand description models. This would also integrate the arm's global pose with that of the moving base of the robot. Next step is to add the required drivers for all new sensors. If the sensors were ROS-ready, then their drivers can be downloaded and installed, as shown in fig 4, but if the sensor is not ROS-ready, then a new ROS driver needs to be developed and installed. Next, the robot control package is updated to allow it to receive ROS control commands. Finally, the necessary ROS packages needed to control the arm, to pick fruits needs to be developed, allowing the mobile robot to become an agriculture robot, all in accordance with ROS standards.

DISCUSSION

Incorporating agriculture machinery into ROS is challenging; a number of sensors and actuators would need to be added to the machine to automate it. These additions would require extensive ROS software updates and additions; such as robot/machine description models, tf trees, sensor drivers, and controller inter-faces. It is then recommended to use ROS-ready hardware when possible, to reduce the burden of software developments, since their this software is available at ROS repositories.

On the other hand, modifying ROS-ready robot also poses some challenges; robots might have smaller capacities than existing agriculture machines, they might run of power quickly, and might require extensive training and maintenance. All of which would require attention from the developers. It is then recommended that the role of robots in the agriculture operation be clearly identified and planned from the beginning; such a way that robots would compliment agriculture operations rather than attempt to change it. This would make introduction of agriculture robots manageable by the developers.

But once these issues are addressed, then the agriculture ma-chine/robot becomes fully ROS-ready, allowing these machines utilize ROS's features and functionalities; such as navigation and motion planning, visualization, HRI, networking, and much more. Custom ROS software can be developed for agriculture tasks, allowing robots to scan the fields, pick fruits, work the land, gather data, weed the field, and much more.

CONCLUSIONS

The Robot Operating System (ROS), along with other open source technologies, has the potential to provide practical solutions for agriculture automation.

The first step in realizing this potential is to incorporate ROS in agriculture automation, which can be achieved making existing agriculture machines ROS-ready, and/or by introducing ROS-ready robots into agriculture.

Making an agriculture machine ROS-ready involves the following steps; developing its description model, ROS sensors drivers, and ROS controller interfaces. Also, it involves creating custom ROS packages to allow the robot to perform the required agriculture tasks.

Introducing ROS-ready robots into agriculture involves a similar process; to modify the robot description model, to update sensor drivers and controller interfaces, and to create custom ROS packages to allow the robot to perform the needed agriculture tasks.

Once completed, then these ROS-ready machines can utilize the features of ROS and other open-source technologies and features; such as software for navigation and motion planning, networking over vast open fields, HRI with agriculture workers, visualization and monitoring tools, and much more.

This allows developers to develop and implement practical agriculture automation, taking another step towards improving agriculture efficiency and output, leading to improved food security for our future generations.

ACKNOWLEDGEMENTS
The authors would like to thank the College of Engineering, the Innovative & Research Management Centre (iRMC), and the Centre for Advanced Mechatronics and Robotics (CAMaRo), UNITEN, for their continued support of this work.

BIOGRAPHY
Sami Hajjaj is a researcher with a keen interest in Robotics, robot software, and related areas. Active since 2012, and is currently a faculty member of the College of Engineering, UNITEN.

REFERENCES

[1] The United States Department of Agriculture. USDA agriculture projections to 2024. Technical report, Office of Chief Economist, 2015.
[2] The Food and Agriculture Organization of the UN, (FAO). The state of food insecurity in the world 2015. meeting the 2015 international hunger targets: taking stock of uneven progress. Technical report, WFO, 2015
[3] David Lane and Rob Buckingham. Ras 2020 national strategy for robot and automation. Technical report, Robot and Automation Systems, SIG 2014.
[4] Manoj Sahi. The rise of agricultural robots. https://www.tractica.com/automation-robotics/the rise of agricultural robots/, 2015. Accessed=2016-06-21.
[5] Frank Tobe. Are agriculture robots ready? https://www.therobotreport.com/news/ ag-in-transition-from-precision-ag-to-full, 2016. Accessed=2016-06-21.
[6] Muhammad Younas, Irfan Awan, and Antonio Pescape. Internet of things and cloud services. Future Generation Computer Systems, 56:605 – 606, 2016.
[7] Rahul. Iot applications with examples. http://internetofthingswiki.com/ iot-applications-examples/541/, January 2016. Accessed: 2016-06-12.
[8] J. Boren and S. Cousins. New Exponential Growth of ROS [ROS Topics]. IEEE Robotics Automation Magazine, 18(1):19–20, March 2011.
[9] Jason M. O'Kane. A Gentle Introduction to ROS. Independently published, October 2013.
[10] The ROS Organization. Browsing ROS packages for indigo.http://www.ros.org/browse/list.php Accessed=2016-06-15.
[11] Sami Hajjaj and Khairul Salleh. Establishing a remote ROS network via Port Forwarding: a detailed procedure. Int J of Advanced Robotic Systems, 2016. Under Reveiw.
[12] Sami Salama Hussen Hajjaj and Khairul Salleh Mohamed Sahari. Review of Research in the Area of Agriculture Mo-bile Robots, pages 107–117. Springer Singapore, Singa-pore, 2014.
[13] Sami Hajjaj and Khairul Salleh. Review of agriculture robotics: Practicality and Feasibility. In 2016 IEEE Inter. Symp. On Robotics and Intelligent Sensors (IRIS) (IEEE-IRIS2016), Tokyo, Japan, 2016.
[14] The ROS Organization. ROS hardware support: Sensors. http://wiki.ros.org/Sensors, 2016. Accessed=2016-06-15.
[15] The ROS Organization. ROS hardware support: Robots. http://wiki.ros.org/Robots, 2016. Accessed =2016-06-15.
[16] Luis Emmi, Mariano Gonzalez de Soto, Gonzalo Pajares, and Pablo Gonzalez de Santos. New trends in robotics for agriculture: Integration and assessment of a real fleet of robots. The Scientific World Journal, 2014:21, March 2014. Article ID 404059.
[17] R. Eaton, J. Katupitiya, K. W. Siew, and K. S. Dang. Preci-sion guidance of agricultural tractors for autonomous farm-ing. In Systems Conference, 2008 2nd Annual IEEE, pages 1–8, April 2008.
[18] The Kitemas LV1 project team. The otl-ros-pkg, from takashi ogura's robot's ros pacakges. http://www.ros. org/news/robots/, 2010. Accessed=28-11-2016.
[19] Ioan Sucan et al. The URDF pacakge for robot models. http://wiki.ros.org/urdf, 2016. Accessed=15/11/2016.
[20] Amber Elliot et al. Adding a new ROS sensor driver. http://wiki.ros.org/turtlebot/Tutorials/ indigo/Adding%20New%203D%20Sensor, 2016. Accessed=10/12/2016.
[21] Issac Saito Publishing odometry data over ROS for robotshttp://wiki.ros.org/navigation/Tutorials/RobotSe tup/Odom 2016. Accessed=20-10-2016.
[22] Adolfo Rodriguez. ros control, an overview. http://roscon.ros.org/2014/wp-content/uploads/2014 /07/ros_control_an_overview.pdf 2014. Accessed=5-6-2016.