Development of a wireless communication platform for multiple-mobile robots using ROS

Conference Paper · July 2019

DOI: 10.1109/CEIT.2018.8751845

4 authors:

Pipit Anggraeni
Politeknik Manufaktur Bandung
28 PUBLICATIONS 80 CITATIONS
SEE PROFILE

Mariem Mrabet
University of Tunis El Manar
1 PUBLICATION 16 CITATIONS
SEE PROFILE

Michael Defoort
Université Polytechnique Hauts-de-France
215 PUBLICATIONS 5,740 CITATIONS
SEE PROFILE

Mohamed Djemai
University Polytechnic Hauts-de-France
292 PUBLICATIONS 4,048 CITATIONS
SEE PROFILE

Development of a wireless communication platform for multiple-mobile robots using ROS

P. Anggraeni
LAMIH UMR CNRS 8201
Polytechnic University Hauts-de-France
59313 Valenciennes, France
pipit.anggraeni@etu.uphf.fr

M. Mrabet
Mechatronics Dept
ENICARTHAGE
2035 Tunis, Tunisia
mariem.mrabet94@gmail.com

M. Defoort and M. Djemai
LAMIH UMR CNRS 8201
Polytechnic University Hauts-de-France
59313 Valenciennes, France
michael.defoort@uphf.fr
mohamed.djemai@uphf.fr

***Abstract*—The MiniLab Enova mobile robot is presented in this paper. The main aim is to implement multi-master systems for managing a wireless communication network of multiple robots. A communication control scheme for multiple MiniLab Enova mobile robots was developed using ROS (Robot Operating System). ROS multi-master system is build from two or more ROS networks, each one with its own roscore node. For multiple robots, there are namespace conflict with ROS topics in a single master system. This limitation can be overcome using multiple masters. With this framework, it is possible to control the movement of multiple robots via a wireless communication network using only one monitoring computer.**

***Keywords*—mobile robot, wireless communication, ROS(Robot Operating System) , multi-master system**

I. Introduction

Mobile robots are classified by the environment in which they travel such as: land or home robots are usually referred to as Unmanned Ground Vehicles (UGVs) [1] [2], aerial robots are usually referred to as Unmanned Aerial Vehicles (UAVs) [3] [4], and underwater robots are usually called Autonomous Underwater Vehicles (AUVs) [5] [6]. In industrial applications, Automatic Guided Vehicle (AGVs) [7] [8] [9] are the most frequently used mobile robots to deliver or transport materials around a manufacturing facility or warehouse.

Modern manufacturing facilities more and more frequently lead to a highly decentralized system with self-organized modules that provide flexibility and improve adaptability to achieve better performance and efficiency. Modern factories are altering rapidly by optimizing adaptations for changes in market fluctuations such as shorter production time, mass customization and increased delivery speed. One of the main advantages of smart factories are robustness due to decentralization of the control system to avoid single point of failure problems.

Multiple robot systems for modern factories have received a lot of interest both from a theoretical and simulation views [10]. In [11], a new method was proposed to design a tracking controller for one nonholonomic mobile robot such that the tracking errors converge to zero for any arbitrary initial tracking error in a fixed-time. For discrete time system, [12] a decentralized model predictive protocol which uses the difference between two consecutive inputs is derived such that the consensus problem for multiple mobile robots is solved.

It is clear that for multiple cooperative robots, the communication architecture and topology play a vital role in distributed network since they dictate how robots will share information among each other. In [13], a wireless communication for mobile robots was proposed based on parallel processing and data transfer for various network topologies. However, it is not possible to add a new node during network. Robot Operating System (ROS) as robotic standard programming provides an efficient solution to solve this problem.

In practice, for the communication aspects, the Robot Operating System (ROS) [14] has been implemented. ROS offers a wide range of controllers for various hardware platforms. Drivers for several robots, sensor devices and a well-defined structure for communication. For the communication between agents, it is possible to share information for any other agent which is interested in (Publisher/Subscriber) or arrange a point to point information exchange (Service/Client). For the communication between two or more ROS networks, ROS multi-master systems are used.

ROS multi-master system comprise two or more ROS networks where each network has its own roscore nodes. Since each robot uses the same software architecture and has the same nodes, issue some problems due to same node name arises in the overall network. This issue can be solved by using different namespaces for each robot. With this framework, it is possible to control the movement of multiple robots via wireless communication network using only one monitoring computer. This solution using multi-master systems is called `multimaster_fkie`.

This paper focuses on the implementation of a decentralized control architecture based on multi-master systems using multiple MiniLab Enova Robot while managing the wireless communication network. This configuration can be easily implemented using common ROS tools.

This paper is structured as follows. Section 2 presents a general description of the MiniLab Enova Robot platform. In Section 3, a ROS overview and `multimaster_fkie` are discussed. Section 4 presents the framework of the decentralized communication architecture. Section 5 introduces the use cases and the experiments done for the validation of such an architecture. Finally, conclusions are summarized and the future work is discussed in the last section.

II. MINILAB ENOVA ROBOT PLATFORM

The MiniLab robot Enova ROBOTICS, as shown in Fig. 1, is a medium-sized mobile robot that offers the best trade-off between robustness and economic competitiveness. The control architecture of the Mini-Lab robot is an open-source based on the Robot Operating System (ROS). ROS is a flexible framework for writing robotic software. It includes several contributed libraries and packages as localization, mapping, planning, perception, etc.

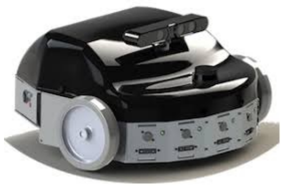


Fig. 1. MiniLab Enova Robot

A. MiniLab Hardware Components

Mini-Lab Enova Robot is equipped with a wide variety of parts and sensors that can become useful in navigation tasks, such 5 ultrasonic sensors, 5 infra-red (IR) distance sensors which provide the distance from obstacles to the robot, orbbec camera, encoders with a resolution of 16 bits, LED indicators, buttons and two DC motors. The maximum speed of the robot is 1.5m/s in both forward and backward movement. The maximum slope angle is 10^o. Each wheel has a driving motor mounted on his axis. The wheels have been chosen to provide more accurate odometry localization. The robot can navigate autonomously or teleoperated using its camera, which transmits video in real time.

MiniLab is optimized for indoor applications, with dimensions of 0.409m x 0.364m x 0.231m in width x length x height and weighing 11.5kg. It also has a load capacity of 3kg. Fig. 2 shows the dimensions of MiniLab robot.

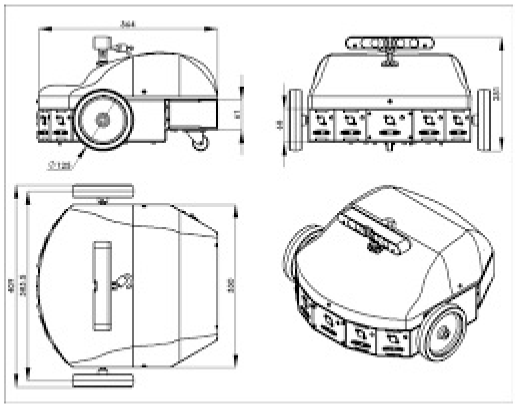


Fig. 2. Dimensions of MiniLab Enova Robot

The processing on the MiniLab is performed by a Intel Atom X86. Each robot has its own wireless router to connect the MiniLab to the network. It allows to design a communication network for multiple robots. The MiniLab is powered by 12V battery for 4 hours of autonomy.

Some hardware descriptions of the MiniLab Enova robot will be explained in the following.

1) Intel Atom N2800: Intel Atom is the brand name for a line of ultra-low-voltage x86-64 microprocessors by Intel Corporation. Atom is mainly used in netbooks, nettops, embedded applications ranging from health care to advanced robotics, and mobile Internet devices (MIDs). The line was originally designed in 45 nm complementary metaloxidesemiconductor (CMOS) technology and subsequent models, codenamed Cedar, used a 32 nm process. Fig. 3 depicts the processor board of Intel Atom N2800.

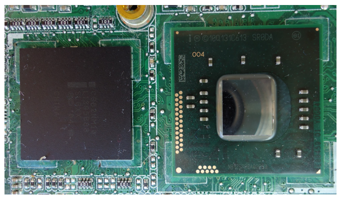


Fig. 3. Intel Atom N2800

2) TL-WR802N Wireless Routers: Each MiniLab Enova mobile robot is equipped with one TL-WR802N wireless router as shown in Fig. 4. This device performs the functions of a router and also includes functions of wireless access point. It is used to provide access to Internet or a private computer network. This router has 300Mbps wireless data rate with several operation modes such as router, repeater, client, access point and hotspot. Powered through a micro USB port by an external power adapter or USB connection to a computer it provides flexibility for any situation.

Fig. 4. TL-WR802N Wireless routers

3) Roboteq Controller: Fitting into a very compact 73x73mm enclosure, Roboteqs SDC2130 controller is designed to convert commands received from an RC radio, Analog Joystick, wireless modem, PC (via RS232 or USB) or microcomputer into high voltage and high current output for driving one or two DC motors. A CAN bus interface allows up to 127 controllers to communicate at up to 1Mbit/s on a single twisted pair. Fig. 5 illustrates the Roboteq controller.

Fig. 5. Roboteq SDC2130 controller

The controller features a high-performance 32-bit microcomputer and quadrature encoder inputs to perform advanced motion control algorithms in open loop or closed loop (speed or position). The SDC21xx features several analog, pulse and digital I/Os which can be remapped as command or feedback inputs, limit switches, or many other functions.

For mobile robotic applications, the controller two motor channels can either be operated independently or mixed to set the direction and rotation of a vehicle by coordinating the motion of each motor.

III. Robot Operating System

Robot Operating System (ROS) is a framework that is widely used in robotics. The philosophy is to make a piece of software that could work in other robots by making little changes in the code. What we get with this idea is to create functionalities that can be shared and used in other robots without much effort so that we do not reinvent the wheel.

ROS was originally developed in 2007 by the Stanford Artificial Intelligence Laboratory (SAIL) with the support of the Stanford AI Robot project. As of 2008, development continues primarily at Willow Garage, a robotics research institute, with more than 20 institutions collaborating within a federated development model. A lot of research institutions have started to develop projects in ROS by adding hardware and sharing their code samples. Also, the companies have started to adapt their products to be used in ROS.

The sensors and actuators used in robotics have also been adapted to be used with ROS. Every day an increasing number of devices are supported by this framework. ROS provides standard operating system facilities such as hardware abstraction, low-level device control, implementation of commonly used functionalities, message passing between processes, and package management. It is based on graph architecture with a centralized topology where processing takes place in nodes that may receive or post, such as multiplex sensor, control, state, planning, actuator, and so on. The library is geared towards a Unix-like system (Ubuntu Linux is listed as supported while other variants such as Fedora and Mac OS X are considered experimental).

ROS multi-master system [15] is built from two or more ROS networks, each one with its own roscore node. For this purpose we need a package called `multimaster_fkie` [16]. This package can be easily installed as shown below

```
$sudo apt-get install ros-indigo-multimaster-
fkie
```

This package offers a set of nodes to establish and manage a multi-master network. This requires no or minimal configuration. The changes are automatically detected and synchronized. The `multimaster_fkie` allows two substantial nodes: the `master_discovery` and the `master_sync` nodes to run simultaneously.

The main features of `master_discovery` is to send multicast messages periodically to the ROS network to make the other `roscore` environments aware of its presence. It also detects the changes in the local network and informs all `roscore` about these changes.

The other nodes called `master_sync` enable us to select which hosts, topics and services which should be synchronized or ignored between different `roscore`. It also helps to register and update information of topics and services to the local `roscore`.

IV. Decentralized communication architecture

In general, MiniLab Enova mobile robot has its own ROS network (either wired, wireless or a combination of both), where a single roscore manages all the communications between all the ROS nodes, either in a single computer or multiple computers in the same network (depending on the complexity and computational needs of the system). See Fig. 6 for a generic hardware setup of such configuration.

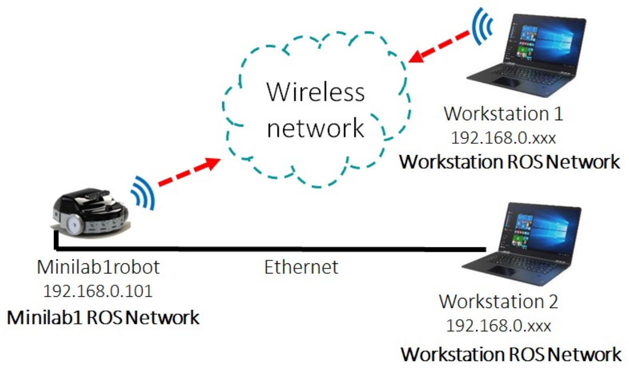


Fig. 6. MiniLab robot setup for single ROS system

In multi-agent systems, multiple robots need to exchange information. Using ROS, there is two possible solutions which can create a single large ROS network managed by a single roscore node or create a configuration to allow information to be exchanged between ROS sub-systems.

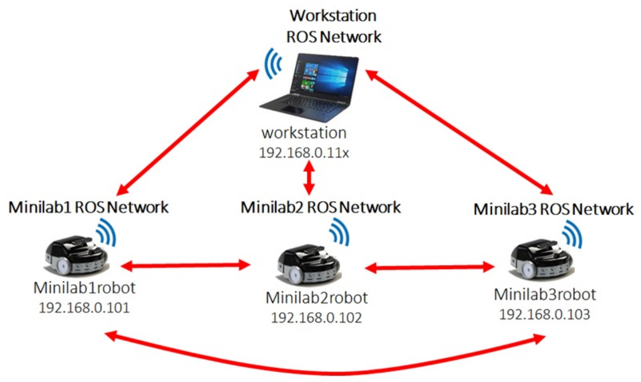


Fig. 7. MiniLab robot runs its own master managing local communication.

Fig. 7 illustrate the configuration of 3 robots to allow information to be exchanged between ROS sub-systems. With this distributed approach, every robot executes its own master. Local inter-process communication is handled by the local master while global communication between robots is handled by a special communication package. Robots may communicate directly forming an ad hoc wireless network and do not rely on a central controller.

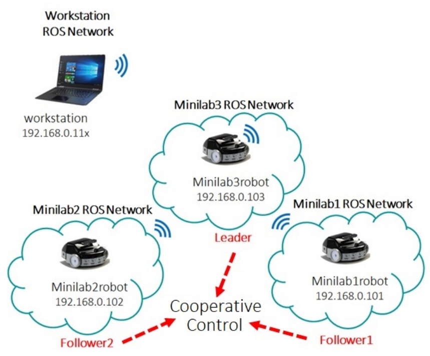


Fig. 8. Multiple robots wireless decentralized communication architecture

See Fig. 8, it is example of possible hardware configuration for multiple robots wireless decentralized communication architecture. Note that in Fig. 8, the multi-agent system consists of one leader and two followers which are connected to the leader wireless network and one monitoring workstation which is also connected to the leader.

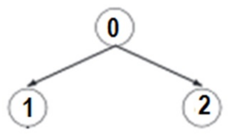


Fig. 9. Network topology for the illustrative example.

V. MiniLab as Platform for Cooperative Control of Multi-Agent Systems

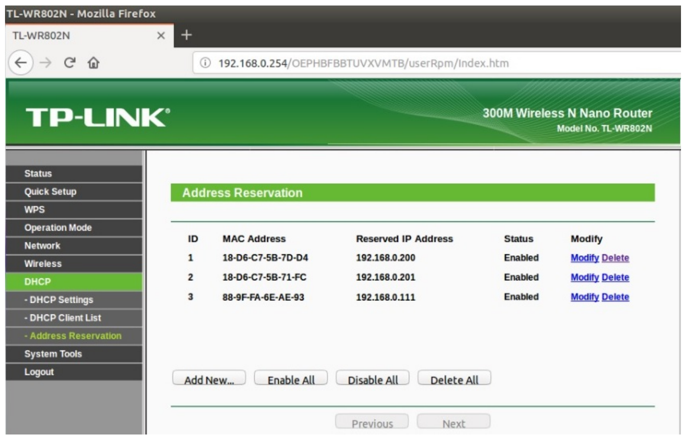
In this section, we will present the procedure for the implementation of the decentralized configuration explained Fig. 8 and Fig. 9. First, we need to configure address reservation and verify this the reservation from PC workstation. To have a proper configuration of the network, we need to modify the */etc/hosts* file from each robots and PC. Afterward we can start `multimaster_fkie` package for our topology. So that we can try our cooperative control protocol on this ROS network configuration.

A. Routers Address Reservation

We need to specify a dedicated IP address for all robots and workstation in the ROS Network, such that all robots and workstation will always receive the same IP address each time when it connects to the DHCP (Dynamic Host Configuration Protocol Server) server. Since each MiniLab Enova robot requires a permanent IP address, we need to configure the address reservation of the router.

First, we open the web browser and in the address bar, type in: http://192.168.0.254. Then, type the username and password in the login page. Then, you should click on DHCP−>Address Reservation on the left side and click Add New button. The MAC and IP address should be given and the status should be selected as enabled. This configuration is shown in Fig. 10.

Note that the MAC address is the MAC address of the device where you want to reserve the IP address. For our case, we need to know the MAC address of each router in the robot and PC workstation. The IP address of the robots are reserved with 192.168.0.2xx and for the workstation 192.168.0.11x



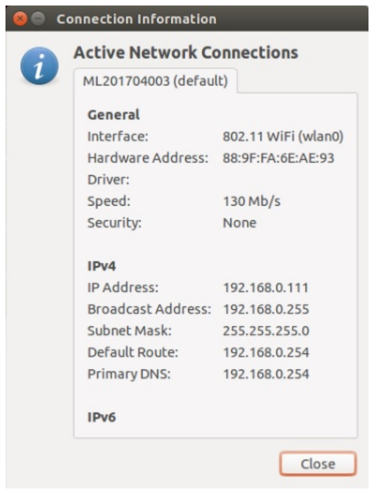
Address Reservation

ID	MAC Address	Reserved IP Address	Status	Modify
1	18-D6-C7-5B-7D-D4	192.168.0.200	Enabled	Modify Delete
2	18-D6-C7-5B-71-FC	192.168.0.201	Enabled	Modify Delete
3	88-9F-FA-6E-AE-93	192.168.0.111	Enabled	Modify Delete

Fig. 10. TP-Link Address Reservation

B. Wireless connection of the leader

To verify the reservation of the IP address for the PC workstation, we have just to connect to the wifi of the leader and we can see from connection information of the PC workstation as shown in Fig. 11.



Active Network Connections

ML201704003 (default)

General

Interface:	802.11 WiFi (wlan0)
Hardware Address:	88:9F:FA:6E:AE:93
Driver:	
Speed:	130 Mb/s
Security:	None

IPv4

IP Address:	192.168.0.111
Broadcast Address:	192.168.0.255
Subnet Mask:	255.255.255.0
Default Route:	192.168.0.254
Primary DNS:	192.168.0.254

IPv6

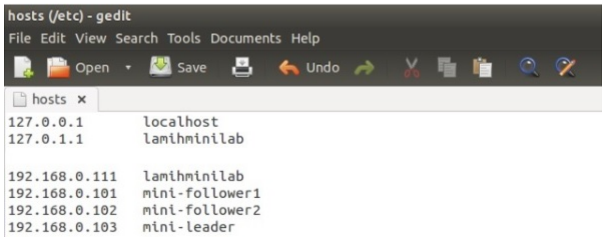
Fig. 11. Wireless connection to the leader

C. Host name and IP address binding

For each robot and workstation, it is necessary to modify the */etc/hosts* file using text editor. Fig. 12 shows the contents of this file for the workstation for the example presented in Fig. 8.

Fig. 13 shows the contents of this file for each robot for the example presented in Fig. 8.

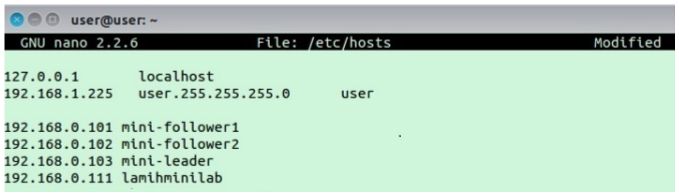
To verify the appropriate configuration of the network, before launching any ROS nodes, it is interesting to ping all other robots from the workstation, using both the host name and the IP address. If all the pings return a reply, the network has been properly configured.



```
127.0.0.1       localhost
127.0.1.1       lamihminilab

192.168.0.111   lamihminilab
192.168.0.101   mini-follower1
192.168.0.102   mini-follower2
192.168.0.103   mini-leader
```

Fig. 12. Contents of the */etc/hosts* file for the workstation for the example presented in Fig.8



```
127.0.0.1       localhost
192.168.1.225   user.255.255.255.0      user

192.168.0.101 mini-follower1
192.168.0.102 mini-follower2
192.168.0.103 mini-leader
192.168.0.111 lamihminilab
```

Fig. 13. Contents of the /etc/hosts file for each robot for the example presented in Fig. 8

Using the current console, the following command should be executed, where hostname or IP address must be the IP address or host name of each robot and the workstation.

```
$export ROS_MASTER_URI=http://<host_ip>:11311
```

`$ROS_MASTER_URI` is required setting that tells nodes where they can locate the master.

For all robots and the workstation, we have to verify if the multicast feature (group communication) is temporary enabled using the following command:

```
$sudo sh -c "echo 0 /proc/sys/net/ipv4/icmp_
echo_ignore_broadcasts"
```

If this command returns 0, the multicast feature is enabled. However, when all robots and workstation restart, this configuration will be lost.

Now, the ROS network has been configured as multi master. The next step is to discover and synchronize each robot and the workstation.

D. Technical check

It is time to launch the `multimaster_fkie` nodes and take a first look at how it works. First of all, it is needed to launch a local roscore on each computer

```
$roscore
```

Then, the master_discovery node should be launched in each computer, passing as an argument the mcast_group parameter to specify the multicast address to be used.

```
$rosrun master_discovery_fkie master_
discovery_mcast_group:=224.0.0.1
```

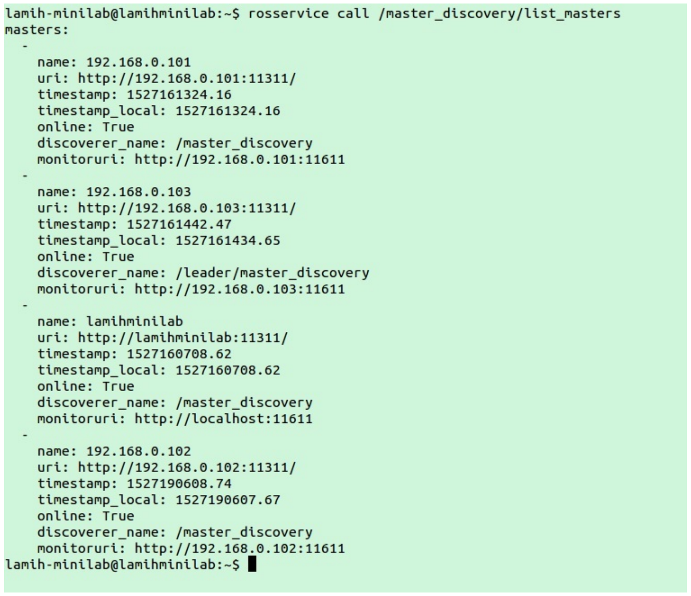
Afterwards, the master_sync node should be launched in each computer. Without any additional parameter, all topics and services on all computers will be synchronized with all others.

```
$rosrun master_sync_fkie master_sync
```

When both nodes are running on all robots and workstation, the following command will list all the available ROS masters on the common network.

```
$rosservice call /master_discovery/list_masters
```

Fig. 14 shows the information reported by this command for the example presented in Fig. 8.



```
lamih-minilab@lamihminilab:~$ rosservice call /master_discovery/list_masters
masters:
  -
    name: 192.168.0.101
    uri: http://192.168.0.101:11311/
    timestamp: 1527161324.16
    timestamp_local: 1527161324.16
    online: True
    discoverer_name: /master_discovery
    monitoruri: http://192.168.0.101:11611
  -
    name: 192.168.0.103
    uri: http://192.168.0.103:11311/
    timestamp: 1527161442.47
    timestamp_local: 1527161434.65
    online: True
    discoverer_name: /leader/master_discovery
    monitoruri: http://192.168.0.103:11611
  -
    name: lamihminilab
    uri: http://lamihminilab:11311/
    timestamp: 1527160708.62
    timestamp_local: 1527160708.62
    online: True
    discoverer_name: /master_discovery
    monitoruri: http://localhost:11611
  -
    name: 192.168.0.102
    uri: http://192.168.0.102:11311/
    timestamp: 1527190608.74
    timestamp_local: 1527190607.67
    online: True
    discoverer_name: /master_discovery
    monitoruri: http://192.168.0.102:11611
lamih-minilab@lamihminilab:~$ █
```

Fig. 14. Information reported by the *rosservice call* command for the example presented in Fig. 8

VI. EXPERIMENTAL RESULTS FOR THE CONSENSUS ALGORITHM FOR SECOND-ORDER MAS

This subsection focusses on the consensus problem for second order MAS using MiniLab Enova robots. In fig. 15 the experimental setup used for the implementation of the consensus protocol is presented.

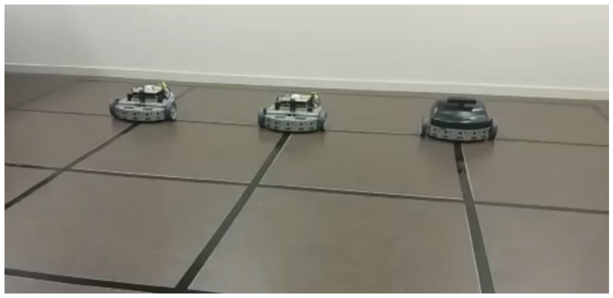


Fig. 15. Experimental setup for the illustration of the consensus protocol.

Consider a multi-agent system consisting of a leader labeled by 0, and two followers, labeled by $i \in 1, 2.$. The leader dynamics is given by the following second-order system

$$\begin{cases} \dot{x}_{1,0} & = & x_{2,0} \\ \dot{x}_{2,0} & = & f(t, x_{1,0}, x_{2,0}) \end{cases} \quad (1)$$

where $x_0 = [x_{1,0}, x_{2,0}]^T \in \mathbb{R}^2$ is the leader state and $f(t, x_{1,0}, x_{2,0})$ is its inherent dynamics. The dynamics of the ith follower is as follows

$$\begin{cases} \dot{x}_{1,i} = x_{2,i} \\ \dot{x}_{2,i} = f(t, x_{1,i}, x_{2,i}) + u_i \end{cases} \tag{2}$$

where $x_i = [x_{1,i}, x_{2,i}]^T \in \mathbb{R}^2$ (resp. $u_i \in \mathbb{R}$ is the state (resp. control input) of the ith followers $(i = 1, 2)$.

We apply the consensus algorithm proposed in [17]

$$u_i = -\sum_{j=1}^{2} a_{ij}[(x_{1,i} - x_{1,j}) + \gamma(x_{2,i} - x_{2,j})], \qquad i = 1, 2. \tag{3}$$

where a_{ij} is the (i, j) entry of the adjacency matrix $\mathcal{A}_n \in R^{2\times 2}$ and γ is a positive scalar. When $x_{1,i}$ and $x_{2,i}$ denote, respectively, the position and velocity of the ith follower, (3) represents the acceleration of that agent. With (3), the consensus is achieved or reached by the team of agents if, for all $x_{1,i}(0)$ and $x_{2,i}(0)$ and all $i, j = 1, 2.$, $x_{1,0}(t) - x_{1,i}(t) \to 0$ and $x_{2,0}(t) - x_{2,i}(t) \to 0$, as $t \to \infty$.

We use a directed graph for the communication topology between agents as shown in Fig. 16,

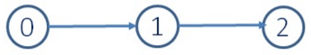


Fig. 16. Communication topology

Fig. 17 shows the evolution of the position of each agent using consensus algorithm (3) under the interaction topology given by Fig. 16

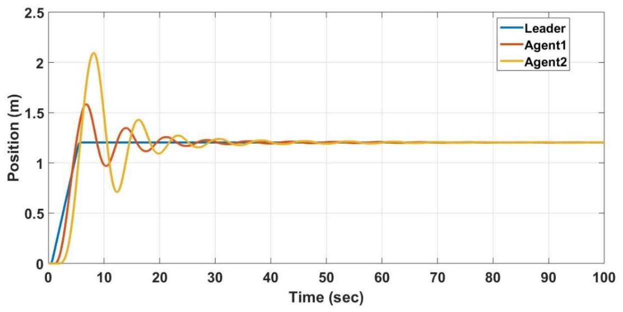


Fig. 17. Robot position for the considered experimental results using consensus protocol (3).

VII. Conclusion

The multi-agent system framework was presented using Robot Operating System (ROS). It is clear from these preliminary results that the proposed multi-master system works properly. Using this scheme, it is possible that multiple robots communicate through a wireless network. Also, it is possible to use this low cost configuration to use only one workstation for monitoring and control of multiple robots.

Acknowledgment

This work has been supported by International Campus on Safety and Intermodality in Transportation, the European Community, the Regional Delegation for research and Technology, the Nord-Pas-de-Calais Region, the Ministry of Higher Education and Research and the National Center for Scientific Research under the project ELSAT VUMOPE, the UVHC BI-CFNes.

References

[1] M. Defoort, T. Floquet, A. Kokosy, and W. Perruquetti, ”Sliding-mode formation control for cooperative autonomous mobile robots,” IEEE Transactions on Industrial Electronics, 55(11), pp.3944-3953, 2008.

[2] M. Defoort, J. Palos, A. Kokosy, T. Floquet, and W. Perruquetti, ”Performance-based reactive navigation for non-holonomic mobile robots,” Robotica, 27(2), pp.281-290, 2009.

[3] M. Orsag, C. Korpela, and P. Oh, ”Modeling and control of MM-UAV: Mobile manipulating unmanned aerial vehicle,” Journal of Intelligent and Robotic Systems, 69(1-4), pp.227-240, 2013.

[4] V. Grabe, M. Riedel, H.H. Bulthoff, P.R. Giordano, and A. Franchi, ”The TeleKyb framework for a modular and extendible ROS-based quadrotor control,” In (ECMR) European Conference Mobile Robots on (pp. 19-25). IEEE, September, 2013.

[5] K. DeMarco, M.E. West, and T.R. Collins, ”An implementation of ROS on the Yellowfin autonomous underwater vehicle (AUV),” In OCEANS 2011 IEEE, (pp. 1-7), September, 2011.

[6] M. Cashmore, M. Fox, T. Larkworthy, D. Long, and D. Magazzeni, ”AUV mission control via temporal planning,” In Robotics and Automation (ICRA) IEEE International Conference on (pp. 6535-6541), May, 2014.

[7] D. Herrero-Perez, and H. Matinez-Barbera, ”Decentralized coordination of autonomous agvs in flexible manufacturing systems,” In Intelligent Robots and Systems, September, 2008. IROS 2008 IEEE/RSJ International Conference on (pp. 3674-3679), 2008.

[8] U.A. Umar, M.K.A. Ariffin, M.K.A., N. Ismail, and S.H. Tang, ”Hybrid multiobjective genetic algorithms for integrated dynamic scheduling and routing of jobs and automated-guided vehicle (AGV) in flexible manufacturing systems (FMS) environment,” The International Journal of Advanced Manufacturing Technology, 81(9-12), pp.2123-2141, 2015.

[9] G. Demesure, M. Defoort, A. Bekrar, D. Trentesaux, and M. Djemai, ”Decentralized Motion Planning and Scheduling of AGVs in an FMS,” IEEE Transactions on Industrial Informatics, 14(4), pp.1744-1752, 2018.

[10] L. Sabattini, E. Cardarelli, V. Digani, C. Secchi, C. Fantuzzi, and K. Fuerstenberg, ”Advanced sensing and control techniques for multi AGV systems in shared industrial environments,” Emerging Technologies and Factory Automation (ETFA) IEEE 20th Conference on (pp. 1-7) IEEE, September, 2015.

[11] P. Anggraeni, M. Defoort, M. Djemai, and Z. Zuo, ”Fixed-Time Tracking Control of Chained-form Nonholonomic System with External Disturbances,” International Conference on Control Engineering and Information Technology (CEIT) 2017.

[12] P. Anggraeni, M. Defoort, M. Djemai, A. Subiantoro, and A. Muis, ”Consensus of Double Integrator Multi-Agent Systems Using Decentralized Model Predictive Control,” International Conference on Control Engineering and Information Technology (CEIT) 2017.

[13] V. Srovnal Jr, Z. Machacek, and V. Srovnal, ”Wireless communication for mobile robotics and industrial embedded devices,” In Networks, 2009. ICN’09. Eighth International Conference on (pp. 253-258) IEEE, March, 2009.

[14] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, ... and A.Y. Ng, ”ROS: an open-source Robot Operating System,” ICRA workshop on open source software Vol. 3, No. 3.2, p. 5, 2009.

[15] http://wiki.ros.org/multimaster

[16] http://wiki.ros.org/multimaster_fkie

[17] W. Ren, and R.W. Beard, ”Distributed consensus in multi-vehicle cooperative control,” London: Springer London, 2008.