A Leader-Follower Communication Protocol for Multi-Agent Robotic Systems

Lubna Najjar
Computer Engineering Department
An-Najah National University
Nablus, Palestine
lubna_najjar@yahoo.co.uk

Noor Johari
Computer Engineering Department
An-Najah National University
Nablus, Palestine
noor-johari@outlook.com

Manar Qamhieh
Computer Engineering Department
An-Najah National University
Nablus, Palestine
m.qamhieh@najah.edu

***Abstract*—The implementation of multi-agent robotic systems is becoming a trending technology due to its useful applications that require the coordination between multiple robots without external interference from humans. Such systems require the combination of successful communication, artificial intelligence and self-organization. In this paper, a new leader-follower communication protocol is proposed, implemented upon wireless communication to establish the coordination between multiple robots. The communication protocol specifies the message format between a leader robot and its followers in the system while taking into consideration some practical considerations like obstacle avoidance and communication loss scenarios. The correctness of the proposed communication protocol is tested on two robotic cars using Arduino microcontroller boards and nrf24l01 wireless module.**

***Index Terms*—leader-follower, multi-agent robots, communication protocol, communication networks, Arduino robots.**

I. Introduction

Daily life requirements are becoming more and more demanding to the point where human efforts alone are not enough to keep up with these rapid demands. Hence, a trend towards robotics and automated embedded systems has emerged in the last decade as a solution to human demands in many life sectors such as agriculture, environmental monitoring, military tasks and many other applications.

The evolution of robotics has already started to penetrate humans daily life [1], [2]. There are many robotic applications ranging from small systems such as nanorobotics [3] that are used mainly in medical treatments, to vast systems that are used for agriculture, manufacturing, transportation and aerospace applications [4], [5].

One trending example of robotics is multi-agent or swarm robots which are defined as a group of collaborative and coordinated robots that are programmed to perform certain tasks without the necessity for human interaction or commanding. Inspired from the behaviour of some animals, insects and birds, swarm robots usually follow a leader-follower relationship, in which one unit of the group is designated as a leader that issues controls to the rest of units to follow. In order to implement successful leader-follower robots, some techniques should be implemented, such as Artificial Intelligence (AI) logic, efficient network communication and high-quality hardware.

The aim of this paper is to design a multi-agent robotic system that consists of at least two robots based on leader-follower approach. The main contribution of this paper is the innovative design of the communication protocol that defines the format of exchanged messages between the leader and its followers which is implemented over a wireless transmission channel. The proposed communication protocol provides a two-way communication between leader and followers while taking into consideration some practical issues regarding sensing the environment for obstacles and communication problems. Two car robots were designed to test the correctness of the communication protocol. The hardware specification of robots is designed mainly using Arduino microcontroller boards, nrf24l01 wireless module as a communication channel between robots and a set of ultra-sonic sensors to detect obstacles nearby in the environment.

This paper is organized as follows; Section II summarizes the major researches done previously related to our leader-follower approach. Section III describes in details our leader-follower communication protocol and Section IV discusses some practical considerations such as communication problem and obstacles. Then Section V describes briefly the hardware implementation of two actual robot cars designed to test the communication protocol. Finally, Section VI provides a conclusion of the paper and some possible future work.

II. Related Work

The idea of robot design began in the 1960s, but it is not until the 1980s when robots were used in industrial operations [6]. They were used as a solution to many technical problems in important sectors such as manufacturing, agriculture, transportation and others [7]. Hence, robotic applications usually need a high level of artificial intelligence in order to be successful specially when these applications involve critical tasks such as in scientific and medical domains [8].

One example of robotic applications is swarm robots or multi-agent robots [9]–[12] which has been recently one of the most trending research topics in automation control and robotics. A multi-agent system consists of a number of autonomous agents (robots) with coordinated behaviour in order to perform collective tasks. The coordination between robots requires a particular agent acting as a leader to steer

978-1-5386-7942-5/19/$31.00 ©2019 IEEE

the other agents of the system (followers). Challenges such as tracking control of multiple mobile robots, self-organization techniques and environment sensing features require a sophisticated leader-follower communication protocol. There are many possible implementations for such protocol like using cameras [13], proximity sensors [14] and absolute location sensing using GPS [15].

One interesting technique for leader-follower communication uses wireless transmission systems. The idea is to use low-priced easy-to-use wireless communication modules between the leader and its followers to establish a reliable coordination through formatted message exchange. For example, a leader-follower architecture that uses multiple communication protocols was implemented in [16] in order to control several follower devices with artificial intelligence. Likewise, *Rosner et al.* [17] provided a leader-follower system that assumes fault tolerant communication. While *Dimargonas et al.* [18] proposed a leader-follower cooperative architecture where multiple leaders are involved in a decentralized heterogeneous system with feedback control feature.

This review of related researches demonstrates that leader-follower systems are not limited to one type of architecture and can be applied in swarm robots and multi-agent systems. In the following sections, a new generic leader-follower protocol over wireless communication is presented between one leader and multiple follower agents with environment sensing and feedback features that can be easily integrated in swarm robotic systems.

III. Leader-Follower Communication Protocol

In this section, the generalized communication protocol is explained based on the leader-follower architecture of a multi-agent robotic system. This communication protocol is implemented over wireless transmission between agents and it defines the messaging format exchanged between the leader and followers that ensures correct transmission, automated control, reception feedback and environment sensing features.

In the first step of the communication protocol, the leader agent has to establish a reliable communication with its followers in the system in order to control their actions and movements. Then, the leader should be aware of any arising problems in the environment such as communication loss or obstacle detection and avoidance techniques. The detailed format of the proposed communication protocol is discussed in the following subsections and the hardware implementation of the system is provided later on in Section V.

A. Leader Message Format: Request Message

The leader agent can initiate the communication with the followers by broadcasting a request message over wireless transmission. The leader message has many purposes:

- Send a command[1] to a specific follower.
- Broadcast a command to all followers in range.
- Solve problems related to the leader such as communication loss with followers or obstacle avoidance.

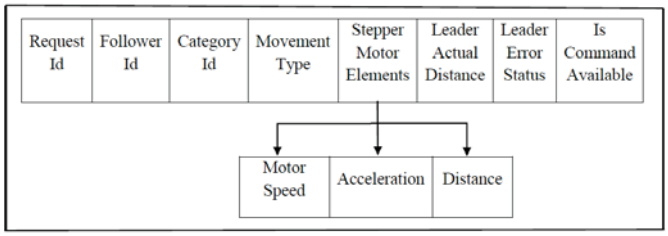

Fig. 1. Leader Request Format

The leader message format is shown in Figure 1 and its fields are explained as follows:

- Request ID: A unique ID to identify each request sent by the leader.
- Follower ID: The ID of a specific follower to whom the request is sent. A special broadcast ID is applied here for all followers to receive the message. This field can be expanded to support an unlimited number of followers.
- Category ID: An ID to define the type of request. Until now, two categories were implemented as follows:
  - Movement Control (ID = 0).
  - Check Movement Possibility (ID = 1).
- Movement Type: The type of movement that the follower has to execute (move forward, backward, left, right and stop).
- Stepper Motor Elements[2]: Specific movement data required for the follower to keep up with the leader.
  - Motor speed.
  - Motor acceleration.
  - Distance: The distance for the follower to move.
- Leader Actual Distance: The actual distance moved by the leader. This data is used only when the follower encounters an obstacle. Otherwise, this field is unused.
- Leader Error Status: An error ID to indicate the status of the leader system:
  - No error (ID = 0).
  - Obstacle error (ID = 1).
  - Communication error (ID = 2).
- Is Command Available: This flag is used when the follower faces an obstacle and the leader is required to communicate with the follower to check which command the follower can execute (more details in Section IV). Otherwise, this field is unused.

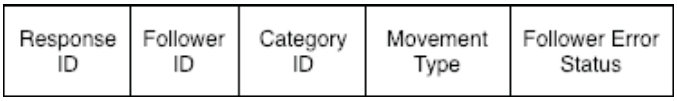

Fig. 2. Follower Response Format

[1]Implemented commands so far are: move forward, backward, turn right, left and stop.

[2]Stepper motors are used in the hardware design of the wheels of robots.

B. Follower Message Format: Response Message

A follower agent usually communicates with the leader's request by sending a feedback response message or a trap message to report a local problem at follower's side.

The format of the response message is shown in Figure 2 and its fields are explained as follows:

- Response ID: Message's ID that corresponds to the request ID (from leader's request message).
- Follower ID: The ID of the follower that sends this message.
- Category ID: Same field from leader request message.
- Movement Type: The type of movement that the follower should execute.
- Follower Error Status: An error ID to indicate the status of the follower system:
  - No error (ID = 0).
  - Obstacle error (ID = 1).

It is worth mentioning that all fields of leader and follower messages have variable length and can be generalized and expanded to include any future categories and features. Hence, the architecture of this proposed communication protocol is general and scalable.

IV. Practical Considerations and Cases

After describing the format of exchanged messages of the leader-follower communication protocol, this section explains the application of the protocol while taking into consideration some practical cases that may face the agents of the system. These cases include the ideal case where no errors encounter the leader and the followers. The second case considers the problem of obstacle avoidance on the leader side and how artificial intelligence is used to allow the leader to steer its followers into avoiding any obstacles. Respectively, the third case considers the obstacle avoidance on the follower side. Finally the communication loss case is considered when the leader loses communication with one of its followers.

A. The Ideal Case

In this case and after the leader establishes a successful connection with the follower, it starts sending commands to the follower so as to control its movement. After each control request from the leader, the follower has to respond with a feedback message once it completes the requested command. There are no problems considered in this case, hence the follower must mimic the movement of the leader. The ideal case scenario is demonstrated in Figure 3.

In the ideal case, the leader stores the generated request in a queue based on the *Request ID*. Then it sends the request message to the follower and it increments the Request ID afterwards so as to ensure a unique request ID. Finally the command is executed by the leader.

The follower receives the leader request, parses it, and stores each field of the request in local variables in a queue based on the *Request ID*. Then the follower starts executing the command, and it sends a feedback response once the execution

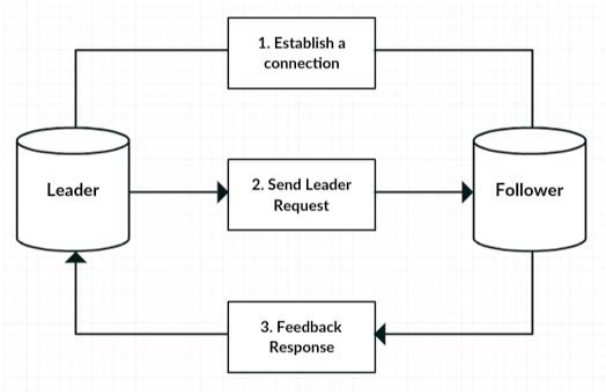

Fig. 3. Ideal Case

is done. In the ideal case, it is assumed that there are no problems in the execution of the command and, hence, the *Follower Error Status* is set to 0 in the response message.

B. Leader Obstacle-Avoidance Case

This case considers that the leader faces an obstacle while a command is being executed. Hence, the leader should use artificial intelligence so as to find an alternative path to avoid the obstacle. At the same time, the leader should inform the follower of the new commands. A flow diagram of this case scenario is shown in Figure 4.

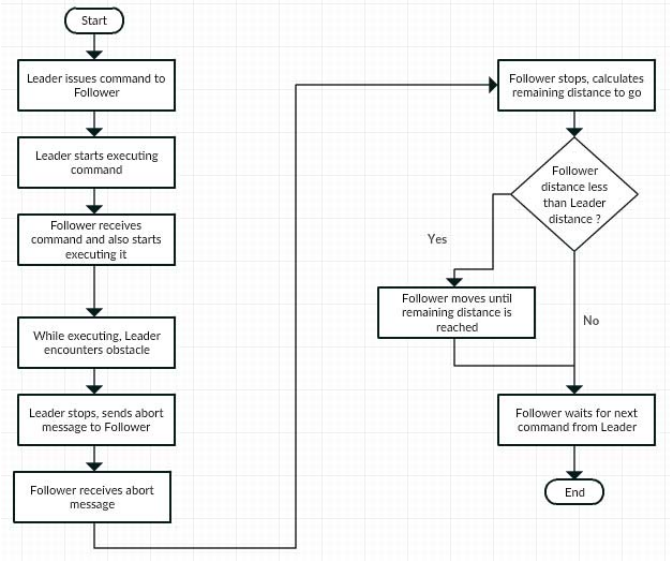

Fig. 4. Leader Obstacle-Avoidance Case

While the leader is executing a movement command, it constantly checks for obstacles by using ultrasonic sensors (hardware design to be explained later) which were set to detect obstacles at 30 cm distance. When an obstacle is detected, the leader immediately stops its current task, creates a request message whose *Leader Error Status* is set to 1, *Leader Actual Distance* is set to the actual distance moved by the leader before aborting and stop command is set as

Movement Type. Then the leader sends this message to the follower.

As for the follower and during its execution of the received command, it performs a continuous check for abort messages from the leader. When an abort message is received, the follower immediately stops the current command, parses the message and stores the fields in local variables. The follower then uses the data from *Leader Actual Distance* to calculate how much distance was not executed by the follower due to the abort message. Afterwards, the follower continues to move until the *Leader Actual Distance* is reached, then the follower stops and waits for the next command from the leader.

C. Follower Obstacle-Avoidance Case

This case deals with obstacle avoidance performed by the follower. It happens when the follower is executing a command received from the leader, and it faces an obstacle. Hence, the follower has to inform and guide the leader in avoiding the obstacle by using an artificial intelligence algorithm. The flow diagram of this scenario is described in Figure 5.

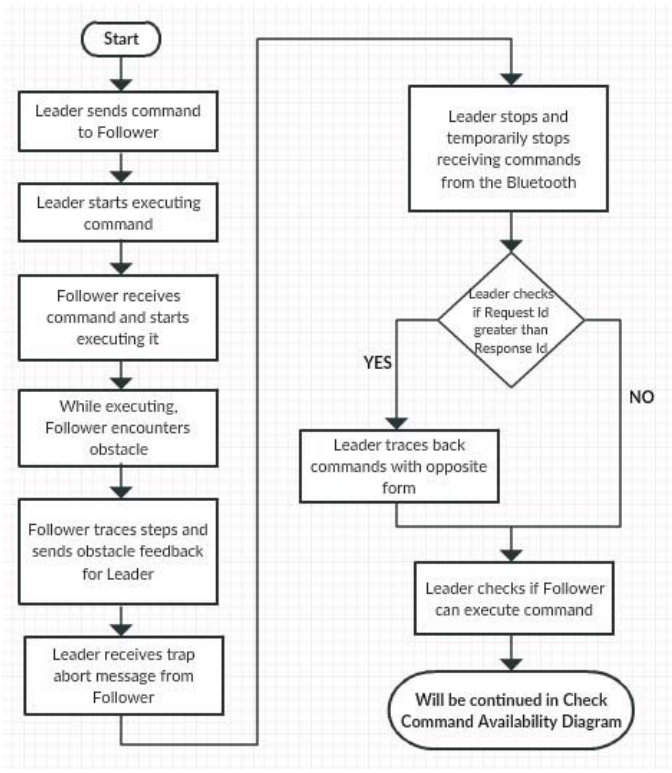

Fig. 5. Follower Obstacle-Avoidance Case

As mentioned previously in Subsection IV-A, when the follower finishes performing a command, a feedback is generated and sent back to the leader. It should be mentioned that the follower does not send this feedback response unless the command has been completed fully and successfully without any obstacles or problems. However, if an obstacle is detected at follower's side, it responds with the same format of the Feedback Response Protocol, with the only difference being that the *Follower Error Status* is equal to 1, making this response a trap command. The follower also traces the command steps that had been executed, before the obstacle was confronted.

Whenever the leader receives a feedback response, the leader parses this message and checks the value of the *Follower Error Status*. If it is equal to 1, the leader immediately stops movement and retrieves these commands from the queue according to the current Request Id and movement type fields.

Next, the leader starts communicating to avoid the obstacle by checking with the follower which movement type is currently available to move. Each command has a different priority based on the command that initiated this scenario. This was implemented in order to be able to avoid the obstacle while ensuring that the leader does not get far off its initial target path. The different commands along with their corresponding priorities are shown in Table I.

Command	Priority Degree		
	1	2	3
Forward	Right	Left	Backward
Backward	Right	Left	Forward
Right	Left	Forward	Backward
Left	Right	Forward	Backward

TABLE I
PRIORITY DEGREE TABLE

According to command priorities, an algorithm was introduced in this research in order to detect and avoid obstacles effectively and intelligently. This process is fully automated and does not require human interference. Through continuous communication between the leader and follower, a correct path will be chosen to avoid obstacles and return to the leaders target path. This algorithm is summarized in Figure 6.

The automated communication between the leader and the follower is implemented with the leader transmitting the same *Leader Request message* that was introduced previously in Subsection III-A, but with slight changes such as setting the *Category Id* to 1 and enabling the *IsCommandAvailable* field to verify that the command requested by leader is acceptable to be performed by the follower. Then the follower responds with the regular Feedback Response Format mentioned previously and sets the *Follower Error Status* according to whether the received command can be executed or not. When the leader receives this feedback, the availability of performing this command can be known by checking the *Follower Error Status*. If the follower responds that the command can be performed, both the leader and follower execute the command.

However, if the follower responds that the command cannot be executed, due to another obstacle being in the path, the leader reruns this process by proposing the next command based on the priority order in Table I. When all options are explored, the process is retried, and it will keep getting retried until a suitable obstacle-free path is found.

D. Communication Loss Case

According to the proposed communication protocol architecture, the leader agent should maintain a valid communica-

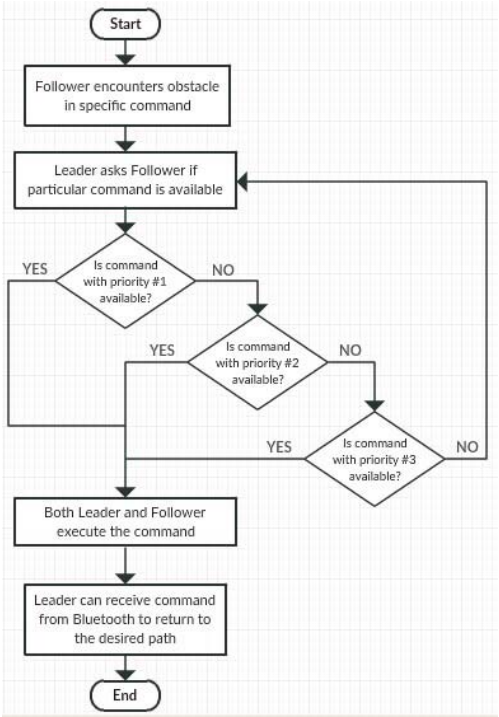

Fig. 6. Check Command Availability

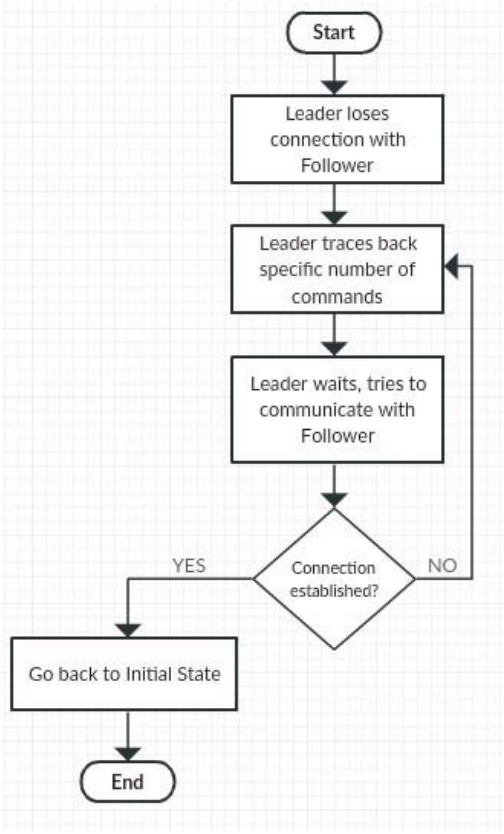

Fig. 7. Communication Loss Case

tion between all of its followers at all times. Hence, all robot agents of the system move together to perform a certain task or operation. So, when the leader sends a request and fails to receive a feedback response message from a follower, this is an indication that there is a communication loss problem due to a hardware malfunctioning at the follower's side. The following algorithm steps are performed:

- The leader broadcasts a request message with *Leader Error Status* set to 2 (communication loss error flag).
- Any follower receives this message immediately aborts its current command.
- The leader traces a specific number of commands so as to ensure the correctness of the system.
- The leader waits for a certain amount of time and then tries to reconnect with the follower.
  - If the connection is regained successfully, the system resumes normally starting from the initial state before the connection was lost, and the leader clears the *Leader Error Status* to 0, indicating that there are no communication problem anymore.
  - If the connection is still down, the leader traces commands back until it reaches the last successful command executed by the malfunctioning follower. If the connection problem persists, the leader reports the problem to the administrator of the system.

The flow diagram of the communication loss case is shown in Figure 7.

V. Experimental Hardware Design

After explaining the leader-follower communication protocol and algorithm, a multi-agent hardware robotic system is designed to test and prove the correctness of the architecture. The system consists of two robotic cars, one of them serves as the leader agent and the other serves as a follower. Extra follower robots can be added to the system simply without changing the current hardware design. There is no difference in the hardware design between the leader and the follower robotic cars, but they differ only in their software so as to implement the leader-follower communication protocol described earlier in Section III.

Each robotic car consists of the following modules:

- Controller Module: An Arduino MEGA-2560 development board is used as the core controller of the car and it is programmed using Arduino Software. This module is responsible for interfacing and controlling the other modules in the system.
- Motion Module: It consists of four stepper motors with L298N H-Bridge Motor Drivers for the control of the wheels. Stepper motors were chosen so as to guarantee accurate wheel movement for better coordination between the leader and followers.
- Communication Module: A nRF24L01 wireless transceiver is used in each car to establish a wireless connection between the leader and followers. The nRF24L01 transceiver is a low-cost module that

communicates with the controller module through SPI protocol.

- Sensory Module: Each car is equipped with ultrasonic sensors for environment sensing and for obstacle detection. The cars were programmed to detect objects at distances equal to 30cm.
- Power Module: Each car is equipped with a battery of 9v and 1.8A which provides enough power for all hardware modules.

Figure 8 shows the designed robotic car.

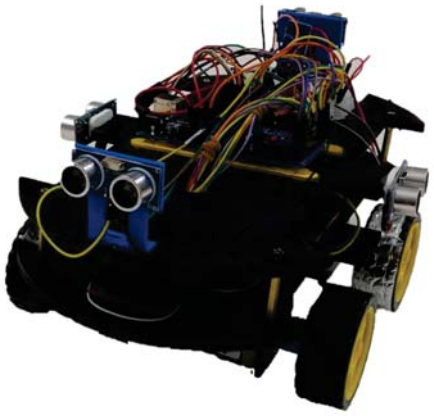

Fig. 8. Robotic Car

By using the designed robotic cars, the communication protocol was tested successfully. The leader agent was programmed to move to a specific target location remotely using a mobile application and the follower agent followed the leader's movements. Both cars were able to arrive at the destination successfully. The practical application scenarios described in Section IV were tested as well. A demo of the experiment, which includes four videos of the cases described in Section IV is available at [19].

The first video shows the ideal case where the leader-follower protocol is executed without errors. The second video shows the leader encountering various obstacles and how it deals with this scenario. The leader stops its movement and instructs the follower to stop instantly as well. The third video presents the follower facing an obstacle and by communicating with the leader, both agents coordinate their movement so as to avoid this obstacle without disorienting from their target destination. The final video shows the follower's battery being disconnected, and how the leader detects communication loss problem and traces back to the follower promptly so as to move apart from each other.

VI. Conclusion and Future Work

The importance of leader-follower systems in the world of robotics is inarguable. They can be implemented in many domains such as transportation, agriculture, aerospace systems and much more so as to facilitate human life in the future.

The main objective of this research was to build a system that includes a powerful and generic communication protocol that allows a fully automated coordination between a leader device and any number of follower devices that are part of robotic system. Although the system was tested for movement controls so far, it was designed in a generic way so that new controls and operations can be added easily in the future.

In the future, a more sophisticated environment sensing can be improved by using image processing so as to develop a more intelligent avoidance protocol. Also, it is possible to use other modules for positioning and geo-locationing so as to improve the placement of robots such as GPS modules and electronic compasses.

References

[1] H. Choset, "Coverage for robotics–a survey of recent results," *Annals of mathematics and artificial intelligence*, vol. 31, no. 1-4, pp. 113–126, 2001.

[2] M. Lungarella, G. Metta, R. Pfeifer, and G. Sandini, "Developmental robotics: a survey," *Connection science*, vol. 15, no. 4, pp. 151–190, 2003.

[3] R. A. Freitas, "Current status of nanomedicine and medical nanorobotics," *Journal of computational and theoretical nanoscience*, vol. 2, no. 1, pp. 1–25, 2005.

[4] B. Siciliano and O. Khatib, *Springer handbook of robotics*. Springer, 2016.

[5] J. Billingsley, A. Visala, and M. Dunn, "Robotics in agriculture and forestry," in *Springer handbook of robotics*. Springer, 2008, pp. 1065–1077.

[6] M. W. Spong, S. Hutchinson, M. Vidyasagar *et al.*, *Robot modeling and control*. Wiley New York, 2006, vol. 3.

[7] J. J. Craig, *Introduction to robotics: mechanics and control*. Pearson/Prentice Hall Upper Saddle River, NJ, USA:, 2005, vol. 3.

[8] S. J. Russell and P. Norvig, *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,, 2016.

[9] X. Ge, Q.-L. Han, D. Ding, X.-M. Zhang, and B. Ning, "A survey on recent advances in distributed sampled-data cooperative control of multi-agent systems," *Neurocomputing*, vol. 275, pp. 1684–1701, 2018.

[10] J. Qin, Q. Ma, Y. Shi, and L. Wang, "Recent advances in consensus of multi-agent systems: A brief survey," *IEEE Transactions on Industrial Electronics*, vol. 64, no. 6, pp. 4972–4983, 2017.

[11] B. Liu, T. Chu, L. Wang, and G. Xie, "Controllability of a leader–follower dynamic network with switching topology," *IEEE Transactions on Automatic Control*, vol. 53, no. 4, pp. 1009–1013, 2008.

[12] E. Şahin, "Swarm robotics: From sources of inspiration to domains of application," in *International workshop on swarm robotics*. Springer, 2004, pp. 10–20.

[13] N. Moshtagh, N. Michael, A. Jadbabaie, and K. Daniilidis, "Vision-based, distributed control laws for motion coordination of nonholonomic robots," *IEEE Transactions on Robotics*, vol. 25, no. 4, pp. 851–860, 2009.

[14] J. F. Roberts, T. S. Stirling, J.-C. Zufferey, and D. Floreano, "2.5 d infrared range and bearing system for collective robotics," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*. IEEE, 2009, pp. 3659–3664.

[15] C. Zhang, N. Noguchi, and L. Yang, "Leader–follower system using two robot tractors to improve work efficiency," *Computers and Electronics in Agriculture*, vol. 121, pp. 269–281, 2016.

[16] L. E. Rosner, K. Rajaiah, K. D. Pedersen, J. Krisciunas, M. Culler, V. Kertesz, and J. A. Wolf, "Fault tolerant communication monitor for a master/slave system," Oct. 2 2001, uS Patent 6,298,376.

[17] A. W. Blackett, B. J. Gilbert, and M. A. Hancock, "Method and system for master slave protocol communication in an intelligent electronic device," Sep. 14 2004, uS Patent 6,792,337.

[18] D. V. Dimarogonas, P. Tsiotras, and K. J. Kyriakopoulos, "Leader–follower cooperative attitude control of multiple rigid bodies," *Systems & Control Letters*, vol. 58, no. 6, pp. 429–435, 2009.

[19] Experiment demo. https : / / drive . google . com / open ? id = 1hO6QT1uzxGrjWEvVbYoB8CMi40LGQILe. Accessed: 2019-01-17.