ELKOMIKA: Jurnal Teknik Energi Elektrik, Teknik Telekomunikasi, & Teknik Elektronika

ISSN(p): 2338-8323 | ISSN(e): 2459-9638 | Vol. 11 | No. 1 | Halaman 1 - 15
DOI : http://dx.doi.org/10.26760/elkomika.v11i1.1 Januari 2023

Design of Deliberative and Reactive Hybrid Control System for Autonomous Stuff-Delivery Robot Rover

EDDY WIJANTO

Program Studi Teknik Elektro, Universitas Kristen Krida Wacana, Indonesia
Email: eddy.wiyanto@ukrida.ac.id

Received 22 September 2022 **|** *Revised* 26 Oktober 2022 **|** *Accepted* 7 November 2022

ABSTRAK

Saat ini, robot otonom memiliki peranan signifikan di berbagai bidang. Robot otonom meningkatkan efektivitas dan produktivitas sekaligus menurunkan risiko dan tingkat kesalahan secara signifikan. Dua paradigma dapat digunakan saat merancang robot otonom, yaitu paradigma reaktif dan deliberatif. Paper ini merancang sistem kontrol hybrid untuk menggabungkan elemen terbaik dari sistem kontrol deliberatif dan reaktif untuk robot pengiriman otonom. Model tersebut dibuktikan dengan misi untuk memindahkan tiga objek berwarna, dari posisinya ke tujuan. Dari perancangan dan pengujian, sistem hybrid dapat meminimalisir kelemahan dari masing-masing sistem, sehingga proses pengiriman barang dapat tercapai sesuai dengan rencana dan dapat bereaksi terhadap kondisi yang sebelumnya tidak diketahui dalam perencanaan, seperti hambatan. Penggunaan sistem hybrid membuka kemungkinan untuk merancang sebuah mobile robot otonom yang dapat beroperasi di lingkungan apapun.

***Kata kunci**: deliberative, reactive, hybrid, autonomous, robot rover*

ABSTRACT

Recently, autonomous robots have become increasingly significant in numerous fields. Robots with autonomy increase effectiveness and productivity while significantly lowering risk and error rates. Two paradigms can be used when creating an autonomous robot, namely reactive and deliberative paradigm. This paper proposes a hybrid control system to combine the best elements of deliberative and reactive control system for an autonomous delivery robot rover. The model was proved with a mission to move the three colored objects, from their respective positions to the goal cells. From the design and testing the hybrid systems can minimize the weaknesses of each system, so that the stuff-delivery process can be achieved in accordance with the plan and can react to conditions that were not previously known in planning, such as the obstacle. The use of a hybrid system opens up the possibility of designing an autonomous mobile robot that can operate in any environment.

***Keywords**: deliberative, reactive, hybrid, autonomous, robot rover*

1. INTRODUCTION

One of the most crucial needs for any robotics application is robot autonomy. While designing and creating autonomous robots, engineers must overcome many challenges, and fully robot autonomy is still a work in progress **(Asokan, 2016)**. Because autonomous robots are complex systems, many different software components must interact and work together. Since robots are getting more human-like, they are developing into significant systems that must adhere to safety standards, including logical, temporal, and real-time constraints **(Panigrahi, et al, 2021)**. An important research area of autonomous mobile robotics is the creation of companions that live in our environment and perform tasks that help us in our daily lives **(Ingrand, et al, 2017)**.

Two paradigms, namely the deliberative and reactive paradigms, can be used in the design of autonomous robots **(Murphy, 2000)**. The fundamental feature of the reactive paradigm is all activities are carried out through behaviors **(Wang, et al, 2017)**. The direct mapping of sensory inputs to a series of motor activities that are then used to carry out a task resulting in behaviors. The reactive paradigm no longer includes the plan element **(Tobaruela, et al, 2017)**. There is no planning carried out based on a global map or model of the environment in reactive architectures, which are made up of a collection of reactive behaviors. Sensing and acting go hand in hand in these reactive activities. The reactive paradigm has advantages such as being applicable to robots with limited and inexpensive hardware resources, low complexity, goal convergence, ease of adaptation to changing conditions, and the ability to safely navigate a robot through completely unknown environments with unpredictable moving obstacles **(Murphy, 2000)**.

Behavior-based robots have become well-known in a number of fields in recent years. Applications for behavior-based robotics have expanded in fields like demining, search and rescue, office automation, and health care, where they are progressively replacing people in labor-intensive and risky jobs **(Hassani, et al, 2018)**. The reactive paradigm and a variety of related subjects, including its history, guiding principles, practical applications, and ongoing research, were covered in **(Lazzeri, et al, 2018)**. In **(Savage, et al, 2021)**, three genetically evolved reactive obstacle-avoidance behaviors for mobile robots were reported. These behaviors were selected using genetic algorithms to enable mobile robots to react to obstacle most effectively as they move toward their target. Three approaches—a traditional method based on potential fields, finite state machines (FSM), and probabilistic finite state machines (PFSM) based on hidden Markov models (HMM)—were examined **(Savage, et al, 2021)**. A mobile robot must also sense its surroundings, perceive its working environment, plot a trajectory, and respond properly depending on the information as an intelligent system. Robotic control architectures define the combination of these capabilities that should be used to create and advance autonomous navigation **(Alatise, et al, 2020)**.

Experimenting with decision-making models using a deliberative architecture requires the availability of engineering tools that enable rapid development, deployment, and evaluation **(Ingrand, et al, 2017)**. Faced with various open environments and performing various tasks and interactions, autonomous robots require distinct considerations to complete their missions **(Gascueña, et al, 2015)**. Further research in **(Avram, et al, 2022)** present a deliberative framework aimed at coordinating the automation of behavioral processes of robotic platforms intended for operation in harsh environments. Between sensing and acting, deliberative architectures have a planning stage. No action can be taken without planning, which is based on a map of the environment that the sensors have collected **(Ibáñez, et al, 2021)**.

Many intralogistics companies such as manufacturing, warehouses, cross-docking, terminals, and hospitals are now using autonomous mobile robots. Autonomous operation under dynamic conditions is possible **(Cho, et al, 2017)**. Unlike automated guided vehicle (AGV) systems that rely on a central entity for all AGV planning, routing, and dispatch decisions, autonomous mobile robots can interact and negotiate independently with other resources, such as machines and systems **(Fragapane, et al, 2021)**. Our previous work found that a negotiator can be implemented to realize a fully autonomous mobile robot based on the behavior of the reactive system **(Wijanto, 2022)**.

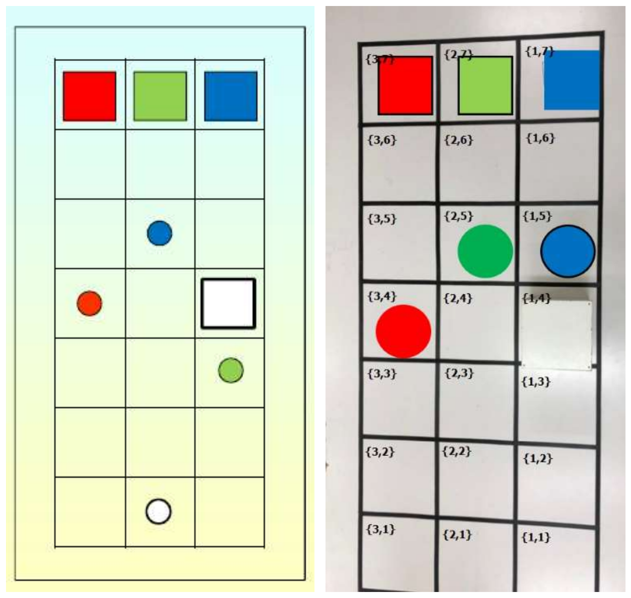
Understanding how to integrate decision-making capabilities into mobile robots is essential for designing robots that can manifest high-level intelligence in real-world settings. In hybrid designs, reactive behaviors are combined with deliberative motion plans, as in the other two models. When deliberative systems compared with reactive architectures, it finds that the former work in a more predictable manner, depend more heavily on an accurate and comprehensive representation of the world, and produce robot trajectory optimizations. Reactive architectures, on the other hand, may function without a model of the world and are computationally considerably simpler. They also respond more quickly to dynamic changes in the environment. By enabling a high-level deliberative planner to tailor the robot's behaviors to the task at hand, known or projected external variables, and available robotic resources, this gives a reactive robot more flexibility.

In this paper, a hybrid control system is designed to combine the best elements of the previous two models, i.e., deliberative and reactive control system. Further, this hybrid system is designed for autonomous stuff-delivery robot rover. The implementation of the hybrid system brings a potential to design a fully autonomous mobile robot in any conditions, such as different number and position of stuff object, different number and position of obstacles, and different home position.

The remainder of this paper is organized as follows. Section 2 describes the methods used in this research, including design of the world model and proves of the a* algorithm, the control system structure, the structure of the robot, the design of the layer structure, and the design of the reactive system. Section 3 presents the results and discussion. Finally, last section conclude the work along with the proposed future work.

2. METHODS

To prove that a deliberative and reactive hybrid control system can work well for an autonomous delivery rover robot, a mission was conducted. In this mission, hybrid control system-based robot are designed to move the three colored objects, i.e., red, green, blue, from their respective positions to the goal cells, respectively. World or search area in the form of 7 x 3 grid arena. Positions of the object are randomly generated offline as the knowledge of the world. One randomly placed prior to the run box shall be unknown to the system. In completing its mission, robot searchs objects by using A* algorithm. The robot shall back to the home or start point after finishing the task. Figure 1(a) depicts the search area of the mission while Figure 1(b) shows the layout of the search area with objects and obstacle that used in the mission.



(a) **(b)**

Figure 1. (a) Design of The Search Area in The Mission, (b) Layout of The Search Area with Objects and Obstacle that Used in The Mission

2.1 *World Model Design*

The World Model used in this mission is a 7 x 3 square-based grid area, where each grid is marked with the black line. Each grid is a square of 20 x 20cm. The three colored objects, i.e., red, green, blue, is a cylinder with diameter of 3cm. The obstacle is a block with a size of 15 x 15 x 30cm. From Figure 1, circles represent the objects while rectangles represent the goal position for each object. The state of each grid is indicated by an index, starting from indexes {1.1} to {3.7}. The start state of the robot is {2,1} and the goal state is {1,7}, {2,7}, and {3,7}. The position of the object is generated offline randomly where in this mission the object's position is indexed {3,4}, {1,5}, and {2,5}. The pair of state positions and goal states of each object were listed in Table 1.

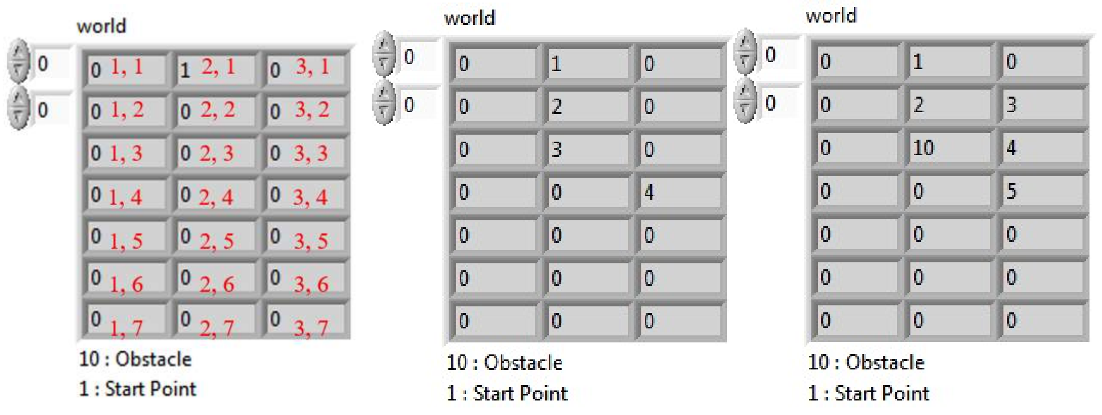
Table 1. State of The Object and Goal

Object Color	Position State	Goal State
Red	{3,4}	{3,7}
Green	{2,5}	{2,7}
Blue	{1,5}	{1,7}

2.2 Proves of The A* Algorithm

In A* algorithm, the function will use the Euclidean Distance as a function and the cost of being taken as the *G* function. In this mission, the *G* value uses an encoder value in the degree where the cost for straight motion is 640° while the cost for diagonal movements is 870°. By knowing the value of *G* and the value of *H*, the A* algorithm can be used for path generating. Before the A* algorithm is implemented into the robot, it first proved in the LabVIEW simulation by using the two-dimentional array world to display the generated path. The initial world map used is depicted in Figure 2(a).

In order to see whether the A* algorithm can work properly, an experiment is conducted where the start state is {2,1} and the goal state is {3,4}. The obstacle is not given in the search area. Based on the theory of A* algorithm and the value of *F*, the path that should be generated is {2,1}, {2,2}, {2,3}, {3,4}. The simulation results of A* algorithm are showed in Figure 2(b). From the simulation results, it can be seen that the A* algorithm program can produce the right path. The second experiment uses the same start state and goal state but an obstacle was placed in the search area, in which the obstacle position is {2,3}. Based on the theory of A* algorithm and the value of *F*, the path that should be generated is {2,1}, {2,2}, {3,2}, {3,3}, {3,4}. The result proves that the A* algorithm program can produce the path correctly by avoiding the obstacle as shown in Figure 2(c).



0 1, 1	1 2, 1	0 3, 1
0 1, 2	0 2, 2	0 3, 2
0 1, 3	0 2, 3	0 3, 3
0 1, 4	0 2, 4	0 3, 4
0 1, 5	0 2, 5	0 3, 5
0 1, 6	0 2, 6	0 3, 6
0 1, 7	0 2, 7	0 3, 7

10 : Obstacle
1 : Start Point

0	1	0
0	2	0
0	3	0
0	0	4
0	0	0
0	0	0
0	0	0

10 : Obstacle
1 : Start Point

0	1	0
0	2	3
0	10	4
0	0	5
0	0	0
0	0	0
0	0	0

10 : Obstacle
1 : Start Point

(a) **(b)** **(c)**

Figure 2. (a) Initial World Map for LabVIEW A* Simulation, (b) LabVIEW A* Simulation without Obstacle, (c) LabVIEW A* Simulation with Obstacle

2.3 Hybrid Control System Structure

The control system structure used in this mission is designed based on the Autonomous Robot Architecture (AuRA) structure. Figure 3 presents the hybrid control system structure implemented in the mission where the structure can be divided into deliberative layer for the planner and reactive layer for the sensory system and actuating system. The planner layer is divided into three subsystems as follows:

1. Mission Planner

   Mission planner responsible for setup the state, including start state, goal state, and rules for navigators. The mission planner also setup the sub-goals and determine the rules for using A* algorithm. The mission planner determines the first object to be taken between the three existing objects. To determine this first object, the mission planner calculates the cost from the start state to the sub-goal state and the cost of the sub-goal state to the goal state. This cost calculation is done for all three objects to be compared where the object with the smallest cost will be selected as the first object to be taken. The same procedure is carried out by a mission planner to determine the second and third objects. If an obstacle is detected, the mission planner will activate the rule for path changes, which will be done through A* algorithm by the navigator. When the robot has completed retrieval of an object, the mission planner will activate the rule to generate the next path, which is done by the navigator using the A* algorithm. If the robot has completed its mission, the mission planner will activates the rule to generate the path back to home or start position, which is done by the navigator using A* algorithm.

2. Navigator
   Navigator was implemented for path generating using the A* algorithm. Navigator performs path generating based on start state information, goal state, and rule from mission planner. The Navigator will do the path updating based on the information or requests from the mission planner. The actions list generated by the navigator will be informed to the pilot.
3. Pilot
   The pilot generates behavior based on the actions list generated by the navigator. The pilot also carry out monitoring action to monitor the movement and position produced, whether it still matches the path or the actions list provided by the navigator. If there is an obstacle or a condition occurs that does not allow the robot to continue its mission, the pilot will send information to the navigator in order to request the new path or new actions list.

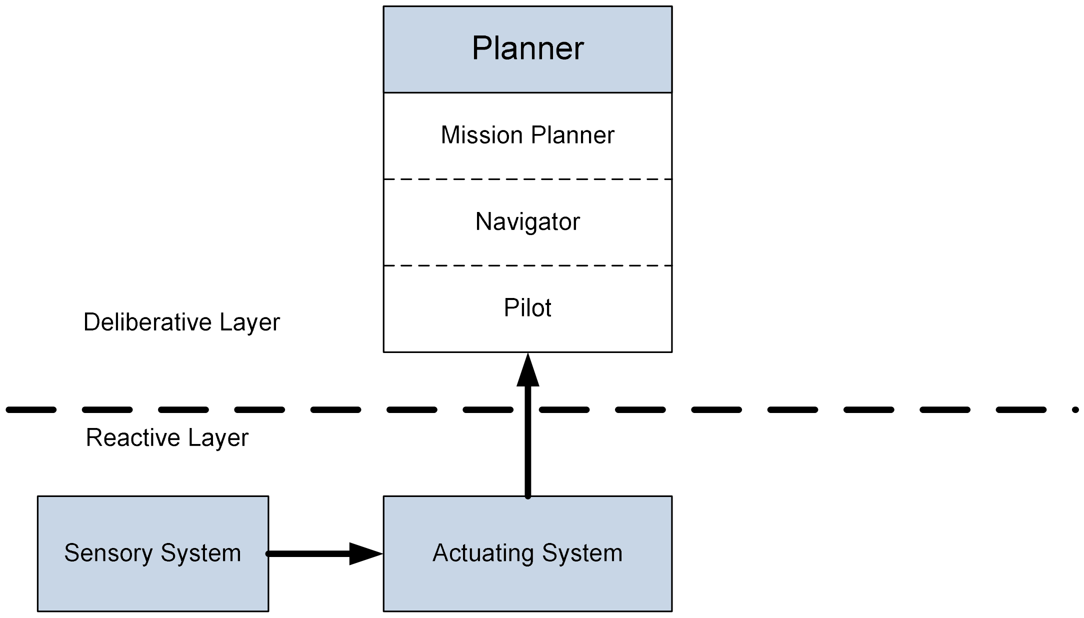


Figure 3. Hybrid Control System Structure

Reactive system generates effective motion. Reactive systems produce movements based on the behavior information from the pilot. The reactive system also responsible to organize or negotiate between existing behaviors. The sensory system produces information in the form of sense results towards the surrounding environment. The reading value from the sensor is used as the basis for motion control, such as effective movement control, movement control when there is an obstacle, and other motion control. The actuating system used as a driver on the motor system based on the driving commands.

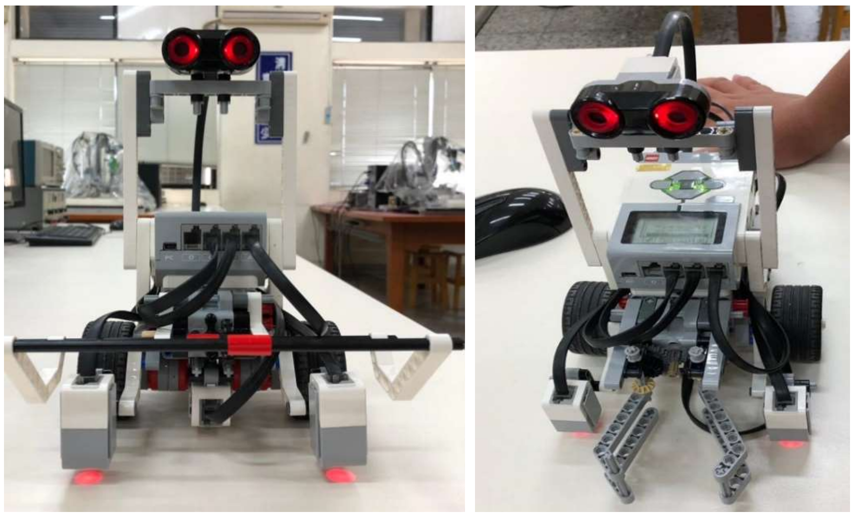
2.4 Robot Structure

The robot used in the mission is embedded robot EV3 from Mindstorm. The robot structure used in this mission is depicted in Figure 4(a) and 4(b) for the first and second design of the robot, respectively. The first design has a weaknesses, especially in the gripper, where in some conditions, the object can be released from the gripper. For this reason, redesign of the gripper is done to ensure the object stays inside the gripper until the release or open grip process.

The value of the motor power for the gripper also have to be given properly so that the object will not separated from the gripper.

The sensors used in the robot are as follows:

1. Encoder Sensor
   The encoder will produce a value which is the distance of movement. Based on the value of the encoder, the robot will produce accurate movements from one grid to another.
2. Gyro Sensor
   Gyro Sensor measures the robot's rotational motion and changes in its orientation. Gyro sensor ensures the robot moves in the proper direction so that it can minimize the movement errors that occurs.
3. Ultrasonic Sensor
   Ultrasonic Sensor is used to detect the obstacle. Ultrasonic sensors are installed on the front of the robot in which the detection range is more focused and not too wide so it is easier to detect the obstacle.
4. Light Sensor
   In this robot, two light sensors were used. Light sensors are used to detect black lines on the grid when doing position correction or positioning so that the robot's direction of movement is accurate and further reduces the accumulated errors of robot movements.



(a) **(b)**

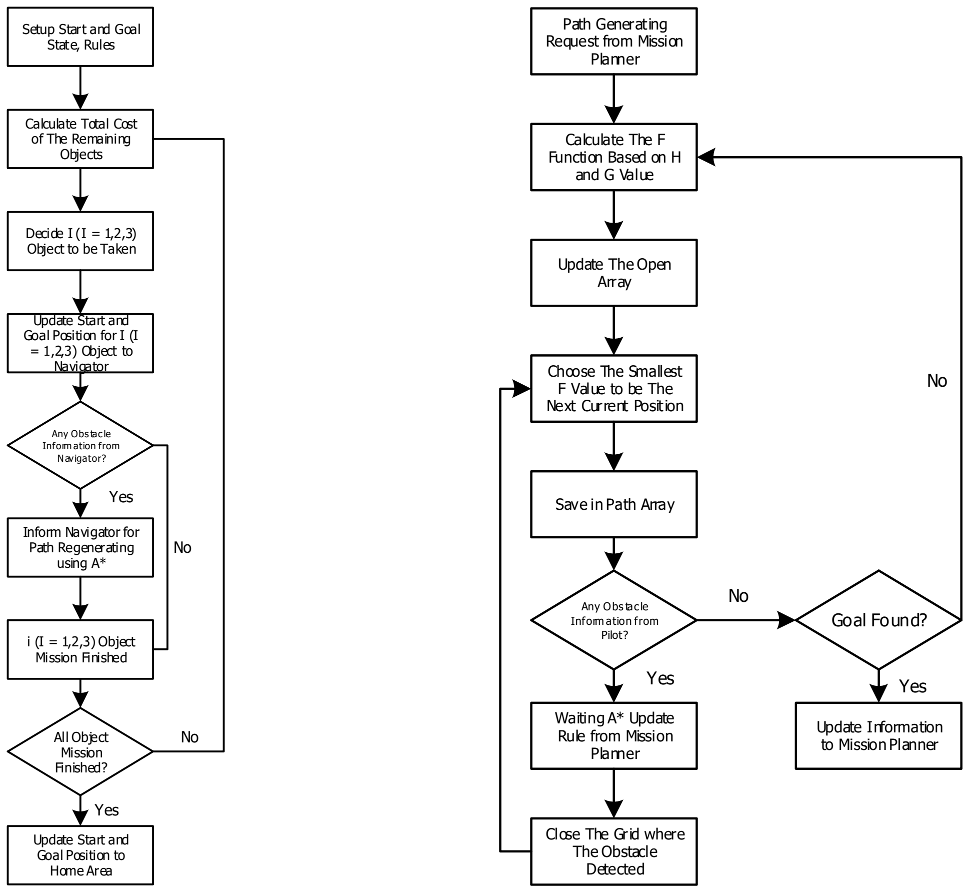
Figure 4. (a). First Design of The Robot (b). Second Design of The Robot

2.5 Design of Layer Structure

The layer structure used in the mission are as follows:

1. Mission Planner
   Before starting the mission, mission planner sets up the start and goal state, as well as the rules that will be used. To determine the order of retrieval of objects, the mission planner calculates costs for all remaining objects. Objects with minimum cost will be taken first among the remaining objects. Further, the mission planner will updates the start position along with the position of the object's goal and informs the navigator. The same procedure will be carried out by the mission planner until all objects have been taken. If mission planner receives information from the navigator about an obstacle, the mission planner will activates the rule, further the navigator performs paths regenerating using A*

algorithm. If all objects have been moved, the mission planner will updates the starting position and goal to the home area. Figure 5(a) presents the flowchart of mission planner program structure.


Setup Start and Goal State, Rules
Calculate Total Cost of The Remaining Objects
Decide I (I = 1,2,3) Object to be Taken
Update Start and Goal Position for I (I = 1,2,3) Object to Navigator
Any Obstacle Infomation from Navigator?
Yes
Inform Navigator for Path Regenerating using A*
No
i (I = 1,2,3) Object Mission Finished
All Object Mission Finished?
No
Yes
Update Start and Goal Position to Home Area
Path Generating Request from Mission Planner
Calculate The F Function Based on H and G Value
Update The Open Array
Choose The Smallest F Value to be The Next Current Position
No
Save in Path Array
Any Obstacle Infomation from Pilot?
No
Goal Found?
Yes
Yes
Waiting A* Update Rule from Mission Planner
Update Information to Mission Planner
Close The Grid where The Obstacle Detected

(a) **(b)**

Figure 5. (a) Mission Planner Program Structure (b) Navigation Program Structure

2. Navigator
   When the navigator receives request for paths generating from the mission planner, the navigator will measures *F* value based on the *H* and *G* values of the eight grids around the current position. The Navigator will updates the open array. Further, the navigator will select the grid with the smallest *F* value as the next current position. When the navigator receives information from the pilot regarding an obstacle, the navigator will informs the mission planner and waits for the path replanning request from the mission planner. After receiving the path replanning request, the navigator will performs a path generating using A* algorithm, by entering the grid where the obstacle is found in the close list. When a goal has been found, the navigator will updates the information to the mission planner and waits for the information on the start and the next goal state from the mission planner. Figure 5(b) displays the flowchart of navigation program structure.
3. Pilot
   After the pilot receives the path list or action list from the navigator, the pilot checks the state position. Further, the pilot will generates the behavior by giving an angle

value for the gyro and the cost value for the encoder. In the process, the pilot updates the state position to the navigator. When an obstacle is detected, the pilot will sends the state position of the obstacle to the navigator and waits for the next path or action list. If the state has reached the end of the path or action list, the pilot will stop the behavior and waits for the path information or the next action list from the navigator. Flowchart of pilot program structure is shown in Figure 6.

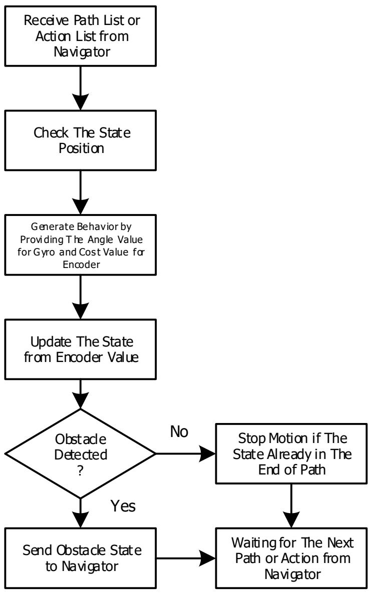

Receive Path List or Action List from Navigator
Check The State Position
Generate Behavior by Providing The Angle Value for Gyro and Cost Value for Encoder
Update The State from Encoder Value
Obstacle Detected ?
No
Stop Motion if The State Already in The End of Path
Yes
Send Obstacle State to Navigator
Waiting for The Next Path or Action from Navigator

Figure 6. Pilot Program Structure

2.6 Design of Reactive System

The structure of the reactive system used in the mission is demonstrated in Figure 7 where *S* refers to suppress and *I* refers to inhibit.

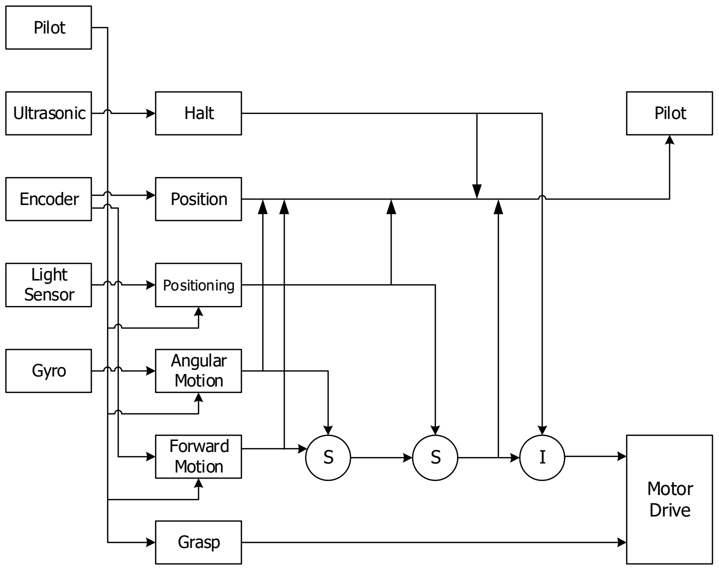


Figure 7. Reactive System

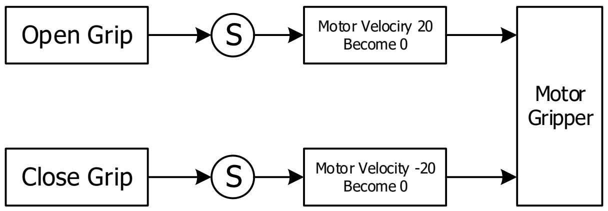


Figure 8. Grasp Behavior

The negotiators used in this reactive system are as follows:

1. When the angular motion behavior is executed, it suppresses forward motion
2. When positioning behavior is executed, it suppresses other motion behavior
3. When the obstacle is detected, the halt behavior will be executed where this behavior will inhibits the entire reactive system until the next behavior generating from the pilot.

The velocity used to drive the gripper motor is 20 where the motor velocity decreases from 20 to 0 to open the grip and from -20 to 0 to close the grip. The grasp behavior is depicted in Figure 8.

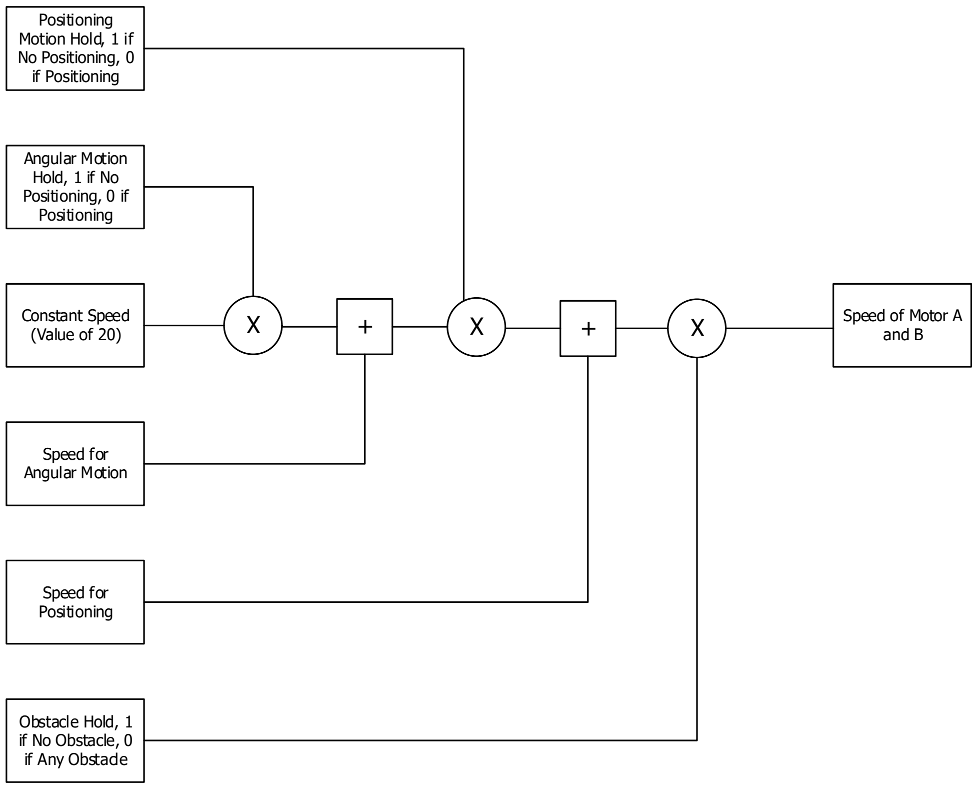

Positioning Motion Hold, 1 if No Positioning, 0 if Positioning
Angular Motion Hold, 1 if No Positioning, 0 if Positioning
Constant Speed (Value of 20)
X
+
X
+
X
Speed of Motor A and B
Speed for Angular Motion
Speed for Positioning
Obstacle Hold, 1 if No Obstacle, 0 if Any Obstacle

Figure 9. Forward Motion Behavior

In the forward motion behavior, the motor speed used is 20. Several things that affect the forward motion behavior, including:

1. When an obstacle is detected, the halt behavior will works and gives a multiplier value of 0 to the motor speed in forward motion

2. When the angular motion behavior works, this behavior will produces a multiplier value of 0 to the motor speed in forward motion
3. When the positioning behavior works, this behavior will produces a multiplier value of 0 to the motor speed in forward motion

Figure 9 presents the forward motion behavior.

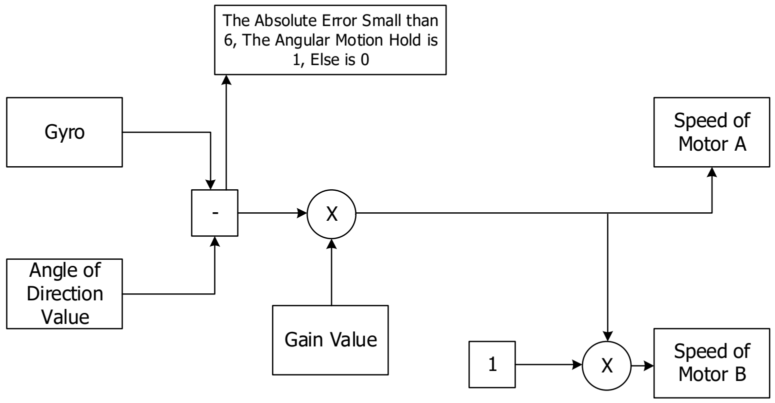


Figure 10. Angular Motion Behavior

The mechanism of the positioning behavior used in the mission is the robot will move straight until both light sensors detect the black line, then the gyro and encoder will be reset. The robot will backoff until the encoder value reaches 200. In order to ensure the robot position in the middle of the grid, positioning is also done by angular turn 90 degree counter clockwise, then the robot will move straight until both light sensors detect the black line. Further, the gyro and encoder will be reset. The robot will backoff until the encoder value reaches 200. The robot then makes angular motion at 90 degrees clockwise and the gyro sensor is reset. The positioning behavior is conducted at start condition and after getting the object. The positioning behavior is displayed in Figure 11.

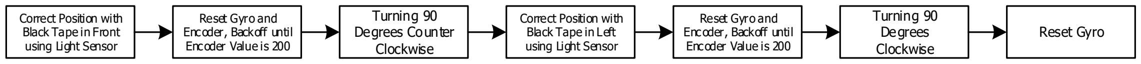

Correct Position with Black Tape in Front using Light Sensor
Reset Gyro and Encoder, Backoff until Encoder Value is 200
Turning 90 Degrees Counter Clockwise
Correct Position with Black Tape in Left using Light Sensor
Reset Gyro and Encoder, Backoff until Encoder Value is 200
Turning 90 Degrees Clockwise
Reset Gyro

Figure 11. Angular Motion Behavior

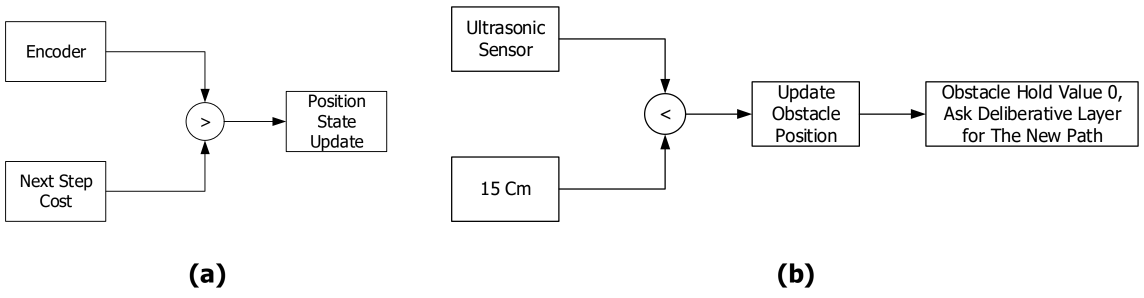


Figure 12. (a) Position Behavior, (b) Halt Behavior

Position Behavior is the part that plays a role in providing position state updates to the pilot. When the encoder reading value is greater than the next cost, the position state will be updated. Figure 12(a) presents the position behavior. If the value of the ultrasonic sensor is less than 15 cm, which indicates an obstacle, the halt behavior will be activated. Halt behavior will stop or inhibit all existing reactive systems. The obstacle position is updated to the pilot, then the pilot will informs the navigator and waits for the new path list. The halt behavior is depicted in Figure 12(b).

3. RESULTS AND DISCUSSION

In order to prove that the hybrid deliberative and reactive control system proposed in this paper can be implemented for autonomous stuff-delivery robot, we performed 100 test runs on the physical robots used in the mission with five different scenario and world model, in which 10 test runs for every scenario. For each scenario, we randomly created obstacle. The robot completely finished the delivery process for each scenario and test runs with a little time difference according to the scenario. For the clarity of analysis, in this section we used the world model defined in the Section 2. Table 2 summarize the state of each step while Table 3 reveals the object collection time.

Table 2. State of The Robot

Mission	Starting State	Route State	Goal State
Go to The First Sub-Goal	{2,1}	{2,1}, {2,2}, {2,3}, {2,4}, {2,5}	{2,5}
Go to The First Goal	{2,5}	{2,5}, {2,6}, {2,7}	{2,7}
Go to The Second Sub-Goal	{2,7}	{2,7}, {2,6}, {3,6}, {3,5}, {3,4}	{3,4}
Go to The Second Goal	{3,4}	{3,4}, {3,5}, {3,6}, {3,7}	{3,7}
Go to The Third Sub-Goal	{3,7}	{3,7}, {3,6}, {2,6}, {1,6}, {1,5}	{1,5}
Go to The Third Goal	{1,5}	{1,5}, {1,6}, {1,7}	{1,7}
Avoid The Obstacle	{1,7}	{1,7}, {1,6}, {2,6}, {2,5}	{2,5}
Go Home	{2,5}	{2,5}, {2,4}, {2,3}, {2,2}, {2,1}	{2,1}

Table 3. Object Collection Time

Scenario	Average Time (Second)
First	238
Second	227
Third	232
Fourth	218
Fifth	245

The results observed in the testing of the mission are as follows:

1. Go to The First Object (First Sub-Goal)
   This part leaves us with three objects for which the mission planner must first calculate the object cost. To determine the order in which objects should be retrieved, mission planner calculates the cost of all remaining objects. From Figure 2(b), we can see that the position of the blue object is {1,5}, the position of the green object is {2,5}, and the position of the red object is {3,4}. Among the remaining objects, the object with the lowest cost is retrieved first. The mission planner works and can determine the object acquisition order based on the calculation of the function value *F* where the minimum *F* value between the start or home position {2,1} and the three objects is the *F* value. of the Green object at position {2.5}. In this section, the navigator is working correctly and can perform path generation using the A* algorithm with route states {2,1}, {2,2}, {2,3}, {2,4}, {2.5}. Pilots also work well and can generate behavior based on paths or action lists generated by the navigator, which is positioning, forward motion, and grasp behavior.

2. Go to The First Goal
   After moving to the first sub-goal {2,5} and grabbing the green object, the robot has to carry the green object to the first goal {2,7}, where the pilot updates the start and end positions. Then, motions are generated based on paths that are forward motion and release behavior, with rouse states at {2,5}, {2,6}, {2,7}. This part shows that the pilot can update the start and end positions and perform the proper behavior.
3. Go to The Second Object (Second Sub-Goal)
   This part leaves us with two objects where the mission planner have to calculate the object cost between the blue and red objects. From Figure 2(b), we can found that the position of the blue object is {1,5} and the position of the red object is {3,4}. Among the remaining objects, the object with the lowest cost is retrieved first. The mission planner works and can determine the object acquisition order based on the calculation of *F* value where the smallest *F* value between the last position from the previous step at {2,7} and the both objects is the red object in position {3,4}. This section also shows that the navigator is functioning properly and can do the path generating by using A* algorithm with the route state of {2,7}, {2,6}, {3,6}, {3,5}, {3,4}. Pilots have also worked properly and can do the behavior generating based on paths or action lists generated by the navigator, which is positioning, forward motion, and grasp behavior. It proves that the communication between mission planner, navigator, and pilot has worked well in which after the first object has been taken and placed in the goal, the pilot provided information to the navigator then forwarded it to the mission planner.
4. Go to The Second Goal
   After reaching the second sub-goal {3,4} and grabbing the red object, the robot must move it to the second goal {3,7}, where the pilot need to update the start and goal positions and generate the forward motion and release behavior based on paths with the route state of {3,4}, {3,5}, {3,6}, {3,7}. This section shows that the pilot was successful and that it was able to update the start and goal locations.
5. Go to The Third Object (Third Sub-Goal)
   There is only one object left in this part. This section shows that the navigator is working properly and can do the path generating by using A* algorithm between the last position from the previous step at {3,7} and the blue objects at position {1,5}, with the route state of {3,7}, {3,6}, {2,6}, {1,6}, {1,5}. Pilots also work properly and can generate behavior based on paths or action lists generated by the navigator, which is positioning, forward motion, and grasp behavior. This proves that the communication between mission planners, navigators, and pilots functioned well. After the second object was picked up and placed on target, the pilot provided information to the navigator and further forwarded it to the mission planner.
6. Go to The Third Goal
   After reaching the third sub-goal {1,5} and grabbing the blue object, the robot should move the object to the third goal {1,7}. The pilot needs to update the start and end positions and perform the behavior generating based on paths, which is forward motion and release behavior, with the route state of {1,5}, {1,6}, {1,7}. This part presents that the pilot has functioned well and can updated the start and goal positions.
7. Avoid The Obstacle
   After completing the mission, the robot must return home, which is from position at {1,7} to home position at {2,1}. In this scenario there is an obstacle at {1,4}. This section proved the reactive mechanism has functioned well, when an obstacle is detected, the pilot can provide information to the mission planner through the navigator. The mission planner uses the A* algorithm to provide route regeneration information to the navigator. The navigator can generate a new path to avoid the presence of obstacles, and the pilot can follow the navigator's new path using the new route states of {1,7}, {1,6}, {2,6},

{2,5}. To avoid diagonal movements near the obstacle, the mission planner gives a rule for navigator in the path generating in which when the obstacle is detected, (*X*), (*X*-1), and (*X*+1) positions are entered into the close list and both will return to the open list after the robot does not detect an obstacle again, where *X* is the position state of the obstacle.

8. Go Home
   After avoiding the obstacle, in this section, the mission planner can update the start and goal position to the home area, which is the last position from previous step at {2,5} to the home position at {2,1}. The Navigator successfully generated an action list based on information of start and home position by using the A* algorithm with the route state of {2,5}, {2,4}, {2,3}, {2,2}, {2,1}. The pilot has been able to produce behavior according to the action list from the navigator, which is forward and positioning behavior.

Further, from Table 3, it can be seen that the average time to finish each of five scenarios implemented in the mission, there is only a slight time difference. The differences come from the position of the obstacles that effect the cost function of each sub goal and goal itself. Overall, from the experiment of five scenarios, the hybrid system can finish the mission successfully, without being affected by the position of the obstacle. These results prove that the hybrid system can conduct the correct plan and in the same time can react to the unknown environment situation according to the obstacle in the mission.

4. CONCLUSIONS

From the design and testing of the Deliberative and Reactive Hybrid Control System for Autonomous Stuff-Delivery Robot Rover, it can be concluded that hybrid systems which are a combination of deliberative systems and reactive systems can minimize the weaknesses of each system. With this hybrid system, the stuff-delivery process can be achieved in accordance with the plan and can react to conditions that were not previously known in planning, such as the obstacle. From the design and testing, it has been proven that the A* algorithm can be implemented in the embedded systems, such as the EV3 Mindstorm Robot with some adjustments, for example, using one dimentional array and cluster. Robots can perform a search mechanism through paths generating using the A* algorithm. Further works will focused on more complex world scenario and increasing the efficiency of the hybrid control system.

REFERENCES

Alatise, M., Hancke, G. (2020). A Review on Challenges of Autonomous Mobile Robot and Sensor Fusion Methods. *IEEE Access*, *8*, 39830-39846.

Asokan, T. (2016). Autonomy for Robots: Design and Developmental Challenges (Keynote Address). *Procedia Technology*, *23*, 4-6.

Avram, O., Baraldo, S., Valente, A. (2022). Generalized Behavior Framework for Mobile Robots Teaming with Humans in Harsh Environments. *Front Robot AI*, *9*, 898366.

Cho, J.-H., Kim, Y.-T. (2017). Design of Autonomous Logistics Transportation Robot System with Fork-Type Lifter. *International Journal of Fuzzy Logic and Intelligent Systems*, *17*(3), 177-186.

Fragapane, G.D., Koster, R., Sgarbossa, F., Strandhagen, J.O. (2021). Planning and control of autonomous mobile robots for intralogistics: Literature review and research agenda. *European Journal of Operational Research*, *294*(5), 405-426.

Gascueña, Manuel, J., et al. (2015). Deliberative Control Components for Eldercare Robot Team Cooperation. *J. Intell. Fuzzy Syst.*, *28*(1), 17-28.

Hassani, I., Maalej, I., Rekik, C. (2018). Robot Path Planning with Avoiding Obstacles in Known Environment using Free Segments and Turning Points Algorithm. *Mathematical Problems in Engineering*, *2018*(6), 1-13.

Ibáñez, J.R.S., Perez-del-Pulgar, C., Garcia, A. (2021). Path Planning for Autonomous Mobile Robots: A Review. *Sensors*, *21*(23), 7898.

Ingrand, F., Ghallab, M. (2017). Deliberation for Autonomous Robots: A Survey. *Artificial Intelligence*, *247*(4), 10-44.

Lazzeri, N., Mazzei, D., Cominelli, L., Cisternino, A., Rossi, D.D. (2018). Designing The Mind of a Social Robot. *Applied Sciences*, *8*(2), 302.

Murphy, R. (2000). Introduction to AI Robotics. London: MIT Press.

Panigrahi, P., Bisoy, S. (2021). Localization Strategies for Autonomous Mobile Robots: A Review. *Journal of King Saud University-Computer and Information Sciences*, *34*(XXIII).

Savage, J., Muñoz, S., Contreras, L., Matamoros, M., Negrete, M., Rivera, C., Steinbabuer, G., Fuentes, O., Okada, H. (2021). Generating Reactive Robots' Behaviors using Genetic Algorithms. *International Conference on Agents and Artificial Intelligence* (pp. 698-707).

Tobaruela, T.A., Rodríguez, A. (2017). Reactive Navigation in Extremely Dense and Highly Intricate Environments. *Plos One*, *12*(12), e0189008.

Wang, X., Zhang, J. (2017). RPL: A Robot Programming Language Based on Reactive Agent. *International Conference on Electrical, Automation and Mechanical Engineering,* (pp. 250-255).

Wijanto, E. (2022). Design of Behavior-Based Reactive System for Autonomous Stuff-Collecting Mobile Robot. *Techne*, *21*(1), 101-116.