A Control of Multiple Drones for Automatic Collision Avoidance

Korrakot Surakul*, Sunantha Sodsee*, and Sucha Smanchat*

Abstract

Unmanned Aerial Vehicles (UAV) are often used in hazardous missions. Many models, sizes and types are available for different usages. A well-known type of UAV is “drone”, which is controlled by a remote controller via radio wave. These drones have internal memory and use battery for energy source. At present, commercial off-the-shelf drones are subject to human manual control, which can only control one drone at a time. In order to enable autonomous control of multiple drones, the detection of objects around the drones and collision avoidance mechanism are necessary.

This paper presents a control of multiple drones for automatic collision avoidance by using a detection device that is composed of ultrasonic sensors and an embedded control device to detect objects in four directions. A collision avoidance algorithm is designed based on the object detection to enable automatic avoidance. The result has shown that the detection device and the collision avoidance algorithm can work with an accuracy greater than 90%.

Keyword: Drone, Detection Device, Collision Avoidance.

1. INTRODUCTION

Drones, a well-known type of Unmanned Aerial Vehicles (UAV), have often been used in situations where human access is not viable [1]. Until recently, as the prices get cheaper, drones have become a new mean of entertainment commercially available off-the-shelf. Drones are controlled by a remote controller via radio wave. Commercial off-the-shelf drones are usually controlled manually and only one drone can be controlled at a time.

At present, one of the popular drone low-cost drones is AR.Drone 2.0 manufactured by Parrot. It supports Wi-Fi (IEEE 802.11) standard to communicate with the drone through wireless network infrastructure [2]. As the default setting, AR.Drone 2.0 must be controlled by software application developed by Parrot. It has Graphic User Interface (GUI) that users use to control only a single drone. In this setting, drone cannot fly autonomously and cannot avoid collision by themselves.

To address this problem, this research focuses on the development of a technique to control multiple drones with an automatic collision avoidance mechanism. An external detection device is designed that applies Arduino UNO cooperated with ultrasonic sensors to detect objects around drone in 4 directions. Based on the detected objects, drone is able to reroute its path by using a collision avoidance algorithm.

The rest of the paper is organized as follows. Section 2 explains the background technologies. Section 3 describes the design of the detection device and collision avoidance algorithm. Section 4 presents the test case designs and the evaluation results of the proposed technique. Section 5 concludes this paper.

2. BACKGROUND TECHNOLOGY

We describe in this section the background technologies used in this research including Quad-Rotor, Ultrasonic Sensor, Embedded Device and Node.js.

* *Faculty of Information Technology at King Mongkut's University of Technology North Bangkok.*

2.1 Quad-Rotor

Quad-Rotor [3] is a type of drone with four propellers for flying according to aerodynamics principle. Each propeller is driven by a brushless motor that is robust, light-weighted and requires low power to operate. Quad-Rotor is also installed with sensors such as accelerometer, pressure sensor, gyroscope sensor, magnetometer sensor. Embedded intelligence, which is installed on drone, receives control data from remote controller and translate them into motor control signal by Pulse Width Modulation Signal (PWM). An example of Quad-Rotor is depicted in Figure 1.

Figure 1. *Quad-Rotor [3]*

2.2 Ultrasonic Sensor

Ultrasonic is an oscillating sound pressure wave with a frequency greater than 20 kHz. It is used to detect objects and measure distance by sending out signal and detecting the signal reflected back from the objects [4]. This sensor is low cost, light-weighted and has low power consumption. An example of an ultrasonic sensor is shown in Figure 2.

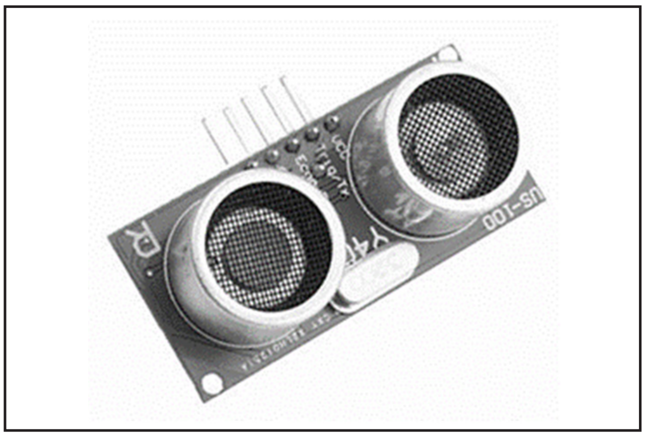


Figure 2. *Ultrasonic Sensor [5]*

2.3 Embedded Device

Embedded device [6] is a control device analogous to a computer. It has a processing unit, memory and communication ports to external devices. Transceivers for analog and digital signals can be manipulated through pin interface. Developers can program to interface with different sensors such as, IMU sensor, temperature sensor, infrared sensor. It also features low power consumption, small-sized and a flash memory unit that can store instruction programing. An example of Arduino embedded device is shown in Figure 3.

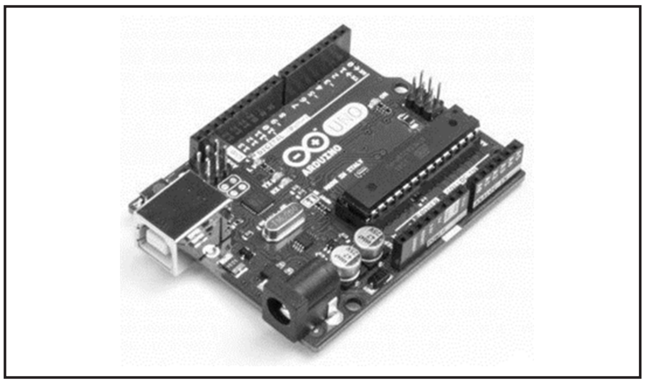


Figure 3. *Embedded Device [7]*

2.4 Node.js

Node.js [8] is an open source software tool that is used on many operating systems. Node.js is based on JavaScript syntax using JavaScript Engine Version 8. It has a fast processing capability due to its nature of asynchronous processing. It is an event-driven language that processes multiple tasks at the same time without sequential order of operations.

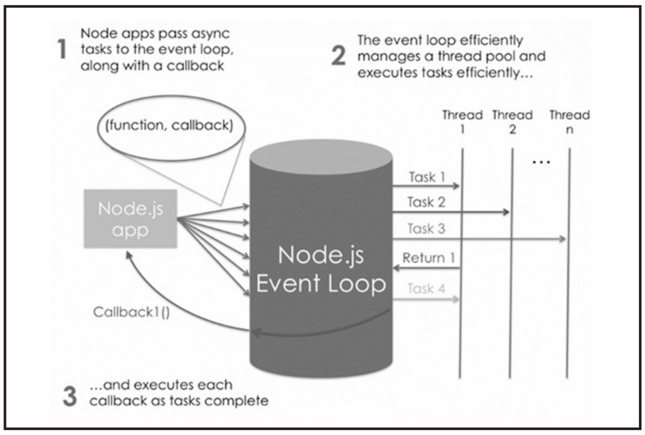

1 Node apps pass async tasks to the event loop, along with a callback
2 The event loop efficiently manages a thread pool and executes tasks efficiently...
Thread 1
Thread 2
Thread n
(function, callback)
Node.js app
Node.js Event Loop
Task 1
Task 2
Task 3
Return 1
Task 4
Callback1()
3 ...and executes each callback as tasks complete

Figure 4. *Node.js Processes [8]*

For example in Figure 4, the process can be explained as followed. Step 1: an application invokes an event with callback function. Step 2: the Event Loop receives the event and spawn a thread to process the event without synchronization. Step 3: the event that has finished processing emits a callback signal to the invoking application.

2.5 UAV Collision Avoidance

A control of multiple UAVs for cooperative search was proposed in [9]. It features collision avoidance and the consideration of UAV communication range constraints. Path planning algorithm is used to convert waypoint path into Yable trajectories that defined in rectangle area. Neural network is used for dynamic navigation of UAVs by choosing a leader. The leader UAV then leads other UAVs by cooperative search algorithm, which maintains the distances between drones for controlling collision avoidance when drones move close to each other. This technique, however, is only implemented in simulation.

Another approach to control of cooperative and non-cooperative multiple drones was proposed in [10]. This technique uses drone trajectory prediction and planning, which are dependent on the velocity profiles characterized by UAV type, atmosphere, limitation of sensors and the control model of UAV. This work uses particle filter to help predicting the trajectory of UAV and uses a detection algorithm based on rectangular grid to calculate the distance from the other drones or objects for collision avoidance.

3. METHODOLOGY

This section explains the conceptual framework for a control of multiple drones for automatic collision avoidance.

3.1 Conceptual Framework

A control of drone for autonomous movement can be developed based on software development kit and existing library called "AR.Drone Autonomy". However, this library cannot control drone efficiently because it cannot deal with the noises in the sensory data emitted from the sensors on the drone. Also, to control multiple drones, the problem of collision between drones must be addressed. The proposed control of multiple drones for automatic collision avoidance is divided into two parts: detection mechanism, which is to be installed on the drone, and collision avoidance algorithm, which is implemented on a drone controller system (e.g. on a PC), as depicted in Figure 5.

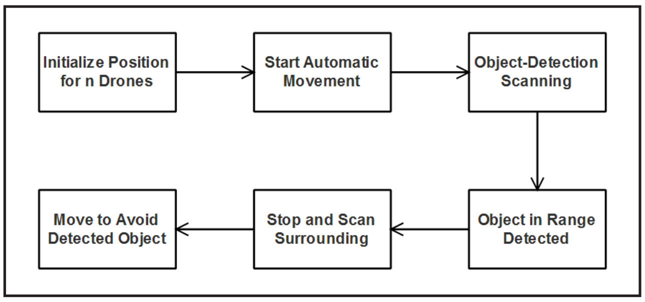

Initialize Position for n Drones
Start Automatic Movement
Object-Detection Scanning
Object in Range Detected
Stop and Scan Surrounding
Move to Avoid Detected Object

Figure 5. *Overview Process*

3.2 Detection Mechanism

For the detection mechanism, a detection device is designed to detect objects around the drone in four directions. The sensor signals are then processed to determine an object detection.

3.2.1 Detection Device

For the detection device, we use the Arduino UNO as the embedded controller. Four ultrasonic sensors are installed on four directions of the detection device. In order to enable the communication between the Arduino UNO and the drone controller system (i.e. the PC acting as the drone controller system), Wi-Fi modules can be used. In this research, we design two detection devices by using two different Wi-Fi modules to transfer detection state data to the controller system.

TONYLABS CC3000 shown in Figure 6 is a Wi-Fi shield module, weighting 20 grams. The PCB circuit design of the detection device that uses CC3000 is depicted in Figure 7. Each of the four ultrasonic sensors is installed in each direction: Node A is sensor 1; Node B is sensor 2; Node C is sensors 3; and Node D is sensor 4.

ESP-8266 shown in Figure 8 is a light-weighted mini module, weighting 1.5 grams. The PCB circuit design with this module is depicted in Figure 9. This module can be installed directly on PCB together with the 4 ultrasonic sensors. The evaluation of the performance of both modules will be explained in a subsequent section.

These PCBs use the power from Arduino UNO, which draws the current from 5VDC USB port as the main source. It can draw the maximum current at 500 mA.

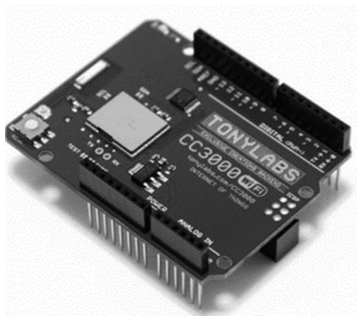


Figure 6. *TONYLABS CC3000 [11]*

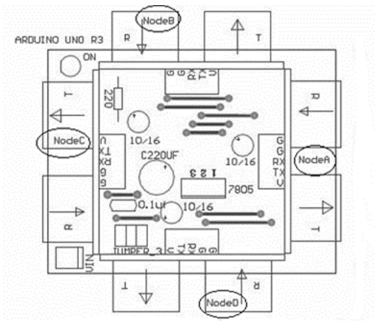


Figure 7. *CC3000 Circuit Design*

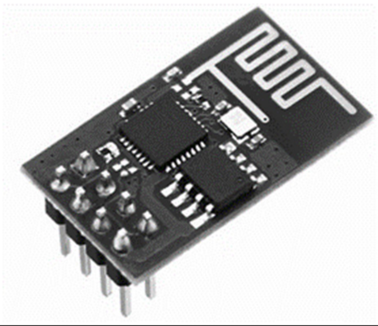


Figure 8. *ESPRESSIF ESP-8266 [12]*

3.2.2 Object Detection Process

The object detection process consists of three steps involving the detection and the assessment of such detection. The output of this process is the detection state, which is composed of four parameters representing the four directions.

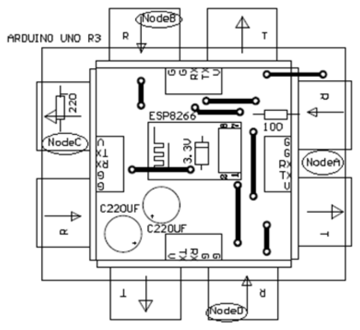


Figure 9. *ESP-8266 Circuit Design*

These parameters are: 'F' for frontal direction, 'R' for right direction, 'B' for back direction and 'L' for left direction

Step 1: Arduino UNO controls the ultrasonic sensors to send out sonic waves to scan for objects every 50 milliseconds. A sensor that detects an object will send an interrupt message through the interface of Arduino UNO. The Arduino use the ultrasonic data, which is the delay of the echo signal, to calculate the distance of the object.

Step 2: the calculated distance is compare with an alert threshold, which is the distance that the drone should be alert of a collision. If the calculated distance is less than or equal to the alert threshold, the detection state for that direction is set to '1'; otherwise the detection state is set to '0'. For example, if an object is detected within the alert threshold at the front of the drone, the value of the parameter 'F' will be set to '1'.

Step 3: Arduino UNO sends detection state to drone controller system via the Wi-Fi module.

These three steps are repeated indefinitely until the drone is shutdown.

3.3 Collision Avoidance Algorithm

When the drone controller system receives a detection in any direction, it will conduct the collision avoidance following these steps.

Step 1: Upon receiving a detection in any direction from detection device (i.e. at least one parameter is set to '1'), drone controller system stops the drone movement and orders it to hover.

Step 2: Drone controller system waits for additional detection state from the detection device to validate detected objects and their directions.

Step 3: Based in the directions of the detected objects, controller reroutes the drone movement for collision avoidance according to the patterns specified in Table 1.

Drone movement is expressed as 3D axis (X, Y, Z) respectively representing pitch, roll, and yaw angle. These movements have a directional marking as '+' and '-'. The movement +X and -X mean moving forward and backward, respectively. The movement +Y and -Y mean moving right and left, respectively. The movement +Z and -Z means moving up and down, respectively.

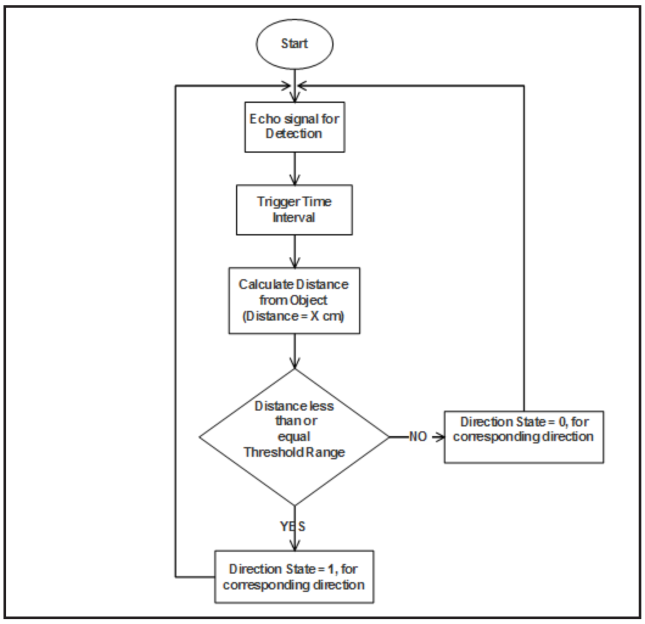

Start
Echo signal for Detection
Trigger Time Interval
Calculate Distance from Object (Distance = X cm)
Distance less than or equal Threshold Range
NO
Direction State = 0, for corresponding direction
YES
Direction State = 1, for corresponding direction

Figure 10. *Object Detection Processes*

The patterns in Table 1 can be explained as follows.

Pattern 1: No object is detected (no. 1)

Patterns 2 and 3: An object is detected in one direction and the drone has moved on the X-axis. It will avoid the object in the clockwise direction.

Patterns 4 and 5: An object is detected in one direction and the drone has moved on the Y-axis. It will avoid the object in the counterclockwise direction.

Patterns 6 to 9: Objects are detected in two adjacent directions (e.g. a corner). The drone will move on the Y-axis opposing the detected direction.

Patterns 10 and 11: are situations where objects move closer to the drone from two opposite sides. These two cases should not happen because the drone should have performed the avoidance when detecting one of the objects before reaching this situation. These extreme cases are hard to characterize. Thus, at this stage, we set the drone to move up on the Z-axis when these patterns occurs.

Patterns 12 to 15: Objects are detected in three directions (e.g. a dead end). The drone will move in the opposite direction of its previous movement (i.e. the only direction without detected objects).

Pattern 16: Objects are detected in all direction. This case is defined as practically impossible in our scope.

Table 1. *Patterns for Collision Avoidance*

No	Normal Move	Detection Status	Avoidance Move
1	Any direction	F : 0 | R : 0 | B : 0 | L : 0	No detection
2	+X	F : 1 | R : 0 | B : 0 | L : 0	+Y
3	-X	F : 0 | R : 0 | B : 1 | L : 0	-Y
4	+Y	F : 0 | R : 1 | B : 0 | L : 0	-X
5	-Y	F : 0 | R : 0 | B : 0 | L : 1	+X
6	+X	F : 1 | R : 1 | B : 0 | L : 0	-Y
7	+X	F : 1 | R : 0 | B : 0 | L : 1	+Y
8	-X	F : 0 | R : 1 | B : 1 | L : 0	-Y
9	-X	F : 0 | R : 0 | B : 1 | L : 1	+Y
10	Any direction	F : 0 | R : 1 | B : 0 | L : 1	+Z
11	Any direction	F : 1 | R : 0 | B : 1 | L : 0	+Z
12	+X	F : 1 | R : 1 | B : 0 | L : 1	-X
13	-X	F : 0 | R : 1 | B : 1 | L : 1	+X
14	+Y	F : 1 | R : 1 | B : 1 | L : 0	-Y
15	-Y	F : 1 | R : 0 | B : 1 | L : 1	+Y
16	Any direction	F : 1 | R : 1 | B : 1 | L : 1	-

4. EVALUATION

In this section mention to result of protocol, simulation and collision avoidance testing. Our experiments employ the AR.Drone 2.0 manufactured by Parrot as shown in Figure 11. It has a processor 1GHz 32 bit ARM Cortex A8 Processor, a digital signal processor 800 MHz Video DSP TMS320 DMC64x, a HD Camera 720p 30FPS, a battery Li-Po capacity 1500 mAh and support Wi-Fi.

AR Drone 2.0 has the hulls for indoor and outdoor uses. The indoor hull weights 450 grams and the outdoor hull weights 380 grams. AR.Drone 2.0 has an embedded Linux operating system, which is installed on internal memory and has a software development kit for application development.

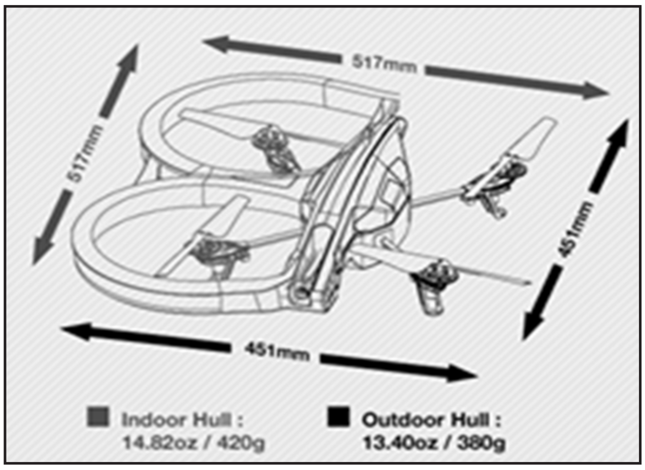


Figure 11. *AR.Drone 2.0 [2]*

4.1 Protocol Testing

We first test the drone communication with the drone controller. In this test, the drone movement is programmed using the AR.Drone Autonomy library, which bases on Node.js. The steps of the test are as follows.

Step 1: Initialize the drone to hover.

Step 2: Order the drone to fly to the coordinate (X: 0.5, Y: 0.5) and go back to the origin. (X: 0, Y: 0)

The result of the protocol testing is shown in Figure 12 and 13. Figure 12 depicts the hovering of the drone at Step 1. The coordinate plot shows that, when hovering, the drone has a slight but acceptable stagger.

In Step 2, the drone moves to the coordinate (X: 0.5, Y: 0.5) and comes back as shown in Figure 13. The plot shows that the drone can go to the designated coordinate successfully. However, there is a slight deviation of 10 centimeters on X-axis and 20 centimeters on Y-axis when returning to the origin position.

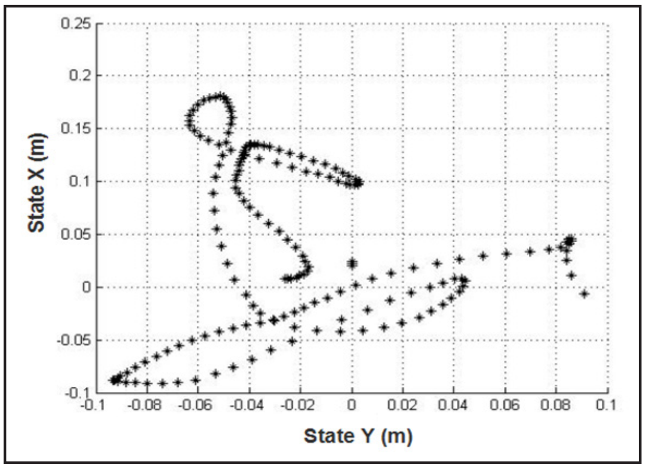


Figure 12. *Movement plot during initial hovering*

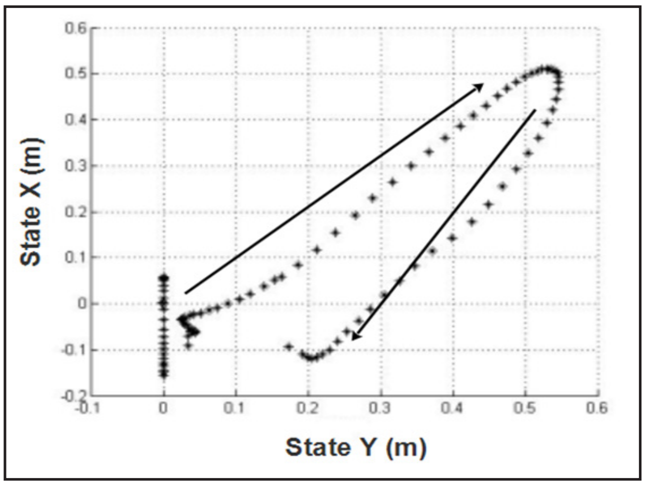


Figure 13. *Go to (X: 0.5, Y:0.5) and return to origin*

4.2 Detection and Avoidance Simulation Test

To reduce the risk before an actual flight, we test collision avoidance algorithm using a drone simulator. As mentioned earlier, we design two detection devices. The first device uses TONYLABS CC3000 module to send the detection state data over HTTP protocol. The other device uses ESPRESSIF ESP-8266 module to send the data over UDP protocol. In the test, the devices are placed on a table and are connected to the drone control system via USB port. Then we put an object inside its alert threshold, which is set to 10 centimeters, to see whether the drone simulator can react as expected. No actual drone is involved in this test. The test process is as follows.

Step 1: Order the drone simulator to go to the coordinate (X: 1, Y: 0) at the velocity 0.05 meter/second.

Step 2: When the drone simulator is moving, places a flat object near the detection device within the alert threshold to simulate an object being detected.

Step 3: Check and record whether the drone simulator reports movement according to the patterns in Table 1 to determine the success rate for each pattern. Each pattern is tested 10 times for detection and avoidance simulation testing.

Step 4: The first three steps are repeated for the second device.

The success rates of this test is shown in Table 2. The result shows that the both detection devices and the collision avoidance algorithm work as expected. An example of the detection state data is shown in Figure 14.

In addition, we compare the performance of the two detection devices in terms of power consumption, latency and time to connect to Wi-Fi network. Power consumption is calculated by the electric current measured by a digital multimeter. The transmission latency of data packet is tracked in the program. The time to connect to Wi-Fi network is measured from switching on the device until it obtains an IP address.

Table 3 shows that the detection device with ESP-8266 module has a lower power consumption than the one with CC3000 module. Table 4 shows that the detection device with CC3000 module has low data latency than the one with ESP-8266 module. Table 5 shows that the detection device with ESP-8266 module requires less time to connect to Wi-Fi than the one with CC3000 module. Note that although CC3000 uses TCP protocol, we disable the acknowledgement mechanism in our program to suit real time control. Thus, the detection device with CC3000 communicates in a way similar to UDP (which is used by ESP-8266).

From this device performance comparison, it can be concluded that although both devices can work similarly, the ESP-8266 module is more suitable for being used on the drone because it weighs only 1.5 grams and consumes less power.

Table 2. *Success rate of the simulation test*

No	Detection Status	Module	
		CC3000	ESP-8266
2	F : 1 | R : 0 | B : 0 | L : 0	100 %	100 %
3	F : 0 | R : 0 | B : 1 | L : 0	100 %	100 %
4	F : 0 | R : 1 | B : 0 | L : 0	100 %	100 %
5	F : 0 | R : 0 | B : 0 | L : 1	100 %	100 %
6	F : 1 | R : 1 | B : 0 | L : 0	100 %	100 %
7	F : 1 | R : 0 | B : 0 | L : 1	100 %	100 %
8	F : 0 | R : 1 | B : 1 | L : 0	100 %	100 %
9	F : 0 | R : 0 | B : 1 | L : 1	100 %	100 %
12	F : 1 | R : 1 | B : 0 | L : 1	100 %	100 %
13	F | 0 | R : 1 | B : 1 | L : 1	100 %	100 %
14	F : 1 | R : 0 | B : 1 | L : 1	100 %	100 %
15	F : 1 | R : 0 | B : 1 | L : 1	100 %	100 %
	Total	100 %	100 %

Table 3. *Power consumption between two modules.*

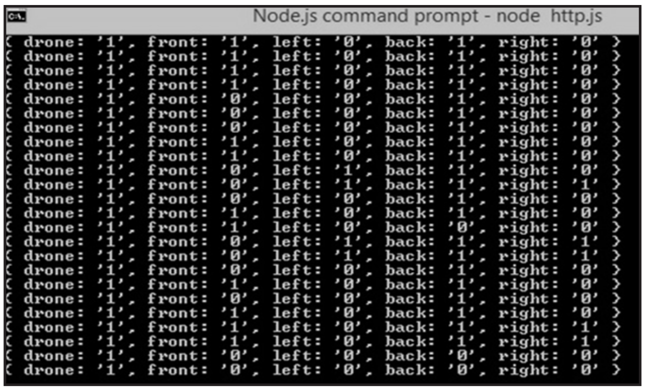
Module	Min (Watt)	Max (Watt)	Mean (Watt)
CC3000	0.74	1.05	1.03
ESP-8266	0.45	0.79	0.54

Table 4. *Latency between two modules.*

Module	Min (ms)	Max (ms)	Mean (ms)
CC3000	30	122	50
ESP-8266	205	317	220

Table 5. *Time to connect to Wi-Fi between two modules.*

Module	Min (s)	Max (s)	Mean (s)
CC3000	30	60	40
ESP-8266	2	5	3



```
Node.js command prompt - node  http.js
{ drone: '1', front: '1', left: '0', back: '1', right: '0' }
{ drone: '1', front: '1', left: '0', back: '1', right: '0' }
{ drone: '1', front: '1', left: '0', back: '1', right: '0' }
{ drone: '1', front: '1', left: '0', back: '1', right: '0' }
{ drone: '1', front: '0', left: '0', back: '1', right: '0' }
{ drone: '1', front: '0', left: '0', back: '1', right: '0' }
{ drone: '1', front: '0', left: '0', back: '1', right: '0' }
{ drone: '1', front: '1', left: '0', back: '1', right: '0' }
{ drone: '1', front: '1', left: '0', back: '1', right: '0' }
{ drone: '1', front: '0', left: '1', back: '1', right: '0' }
{ drone: '1', front: '0', left: '1', back: '1', right: '1' }
{ drone: '1', front: '0', left: '0', back: '1', right: '0' }
{ drone: '1', front: '1', left: '0', back: '1', right: '0' }
{ drone: '1', front: '1', left: '0', back: '0', right: '0' }
{ drone: '1', front: '0', left: '1', back: '1', right: '1' }
{ drone: '1', front: '0', left: '1', back: '1', right: '1' }
{ drone: '1', front: '0', left: '0', back: '1', right: '0' }
{ drone: '1', front: '1', left: '0', back: '1', right: '0' }
{ drone: '1', front: '0', left: '0', back: '1', right: '0' }
{ drone: '1', front: '1', left: '0', back: '1', right: '0' }
{ drone: '1', front: '1', left: '0', back: '1', right: '1' }
{ drone: '1', front: '1', left: '0', back: '1', right: '1' }
{ drone: '1', front: '0', left: '0', back: '0', right: '0' }
{ drone: '1', front: '0', left: '0', back: '0', right: '0' }
```

Figure 14. *Detection Result of detection device.*

From figure 14 shows windows console of detection status which sent to controller. The state "0" is no detected and "1" is detected.

4.3 Collision avoidance result

This section present the experiments on collision avoidance mechanism and the results.

4.3.1 Single drone with separated device

This test uses one drone with a separated detection device. We also use a flat object to simulate object detection similar to the previous experiment. The alert threshold is set to 10 centimeters The test process is as follows.

Step 1: Order the drone to fly to the coordinate (X: 0.7, Y: 0) at the velocity 0.05 meter/second.

Step 2: While the drone is flying, places a flat object near the detection device within the alert threshold to simulate an object being detected.

Step 3: Check and record whether the drone moves according to the patterns in Table 1 to determine the success rate for each pattern. Each pattern is tested 10 times.

The success rates of this test is shown in Table 6. The result shows that the collision avoidance mechanism works well above 80% up to 100% success rate with the average of 93.33 %. The examples of flight paths of patterns 2, 4, 5 and 8 are shown in Figure 15 through 18, respectively.

The error in this test is caused by Wi-Fi signal interference as shown in Figure 19. The drone communication infrastructure relies on navigation data and such interference affects the correctness of these data.

Table 6. *Accuracy of single drone collision avoidance.*

No	Detection Status	Success Rate
2	F : 1 | R : 0 | B : 0 | L : 0	80 %
3	F : 0 | R : 0 | B : 1 | L : 0	80 %
4	F : 0 | R : 1 | B : 0 | L : 0	100 %
5	F : 0 | R : 0 | B : 0 | L : 1	80 %
6	F : 1 | R : 1 | B : 0 | L : 0	100 %
7	F : 1 | R : 0 | B : 0 | L : 1	100 %
8	F : 0 | R : 1 | B : 1 | L : 0	100 %
9	F : 0 | R : 0 | B : 1 | L : 1	100 %
12	F : 1 | R : 1 | B : 0 | L : 1	100 %
13	F : 0 | R : 1 | B : 1 | L : 1	90 %
14	F : 1 | R : 1 | B : 1 | L : 0	90 %
15	F : 1 | R : 0 | B : 1 | L : 1	100 %
	Mean	93.33 %

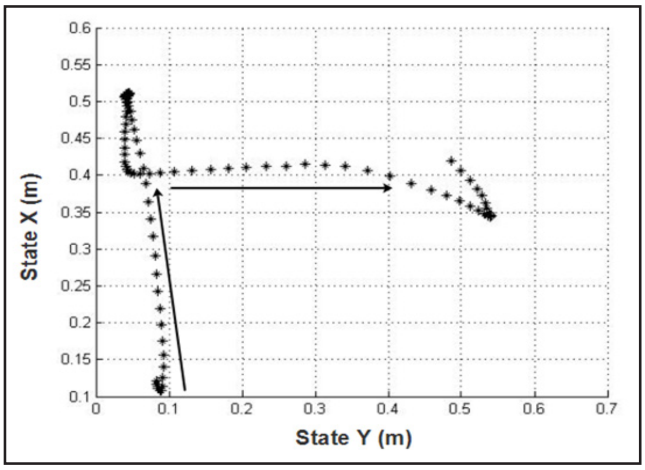


Figure 15. *Result of avoidance pattern 2*

4.3.2 Two drones with separated devices

This test is conduct by placing the two detection devices to face each other. A flat object is again used to simulate the object detection. The alert threshold is set to 10 centimeters The test process is as follows.

Step 1: Place the two drones on the ground with the first drone on the left side and the second drone on the right side with a distance of 2.4 meter apart.

Step 2: Order the first drone to fly to the coordinate (X: 0, Y: 1.2) at the velocity 0.05 meter/second and order the second drone to fly to the coordinate(X: 0, Y: -1.2) at the same velocity. These orders will make both drones move toward each other.

Step 3: When the drones are flying, places a flat object between the two detection devices to simulate an object being detected by both devices

Step 4: Check and record whether the drone can perform the avoidance. This test is repeated 10 times.

The flight paths of the two drones when hovering at the beginning of the test is shown in Figure 20. The two drones have a slight stagger during the hovering.

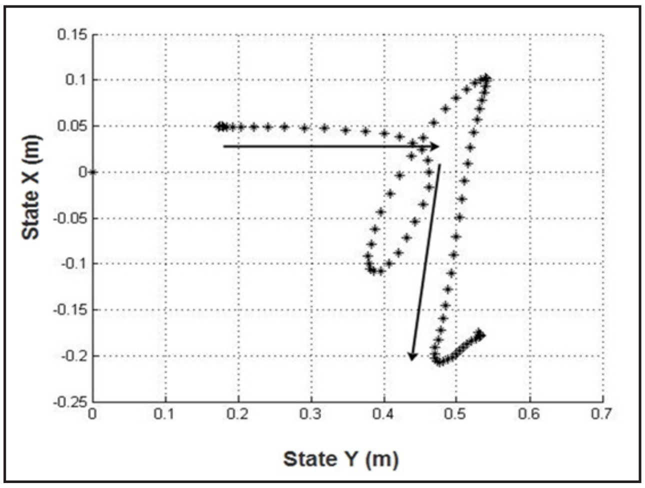


Figure 16. *Result of avoidance pattern 4*

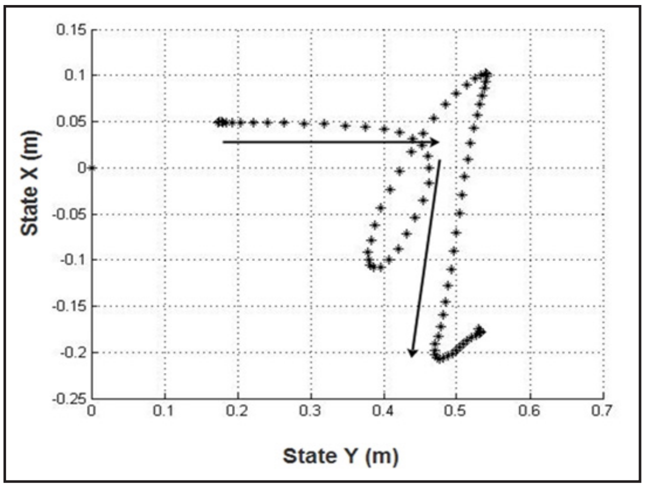


Figure 17. *Result of avoidance pattern 5*

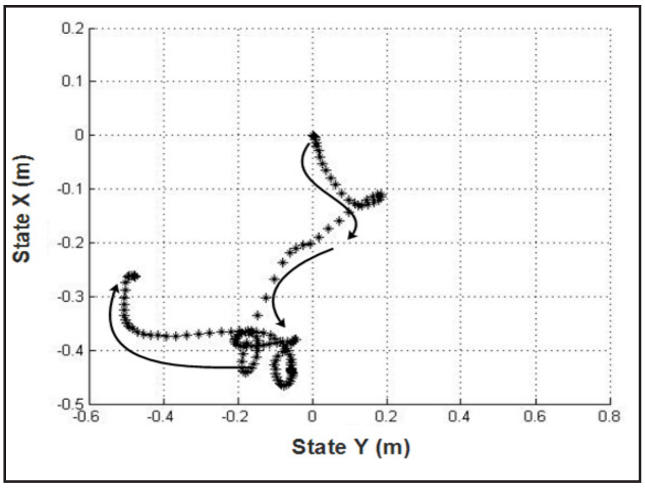


Figure 18. *Result of avoidance pattern 8*

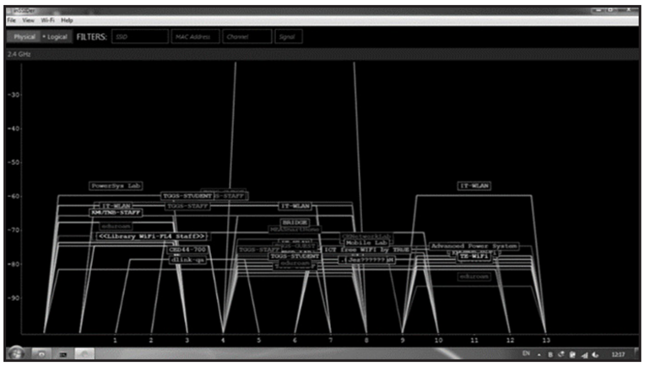


Figure 19. *Interfering Wi-Fi networks in the test area*

4.3.2 Two drones with separated devices

This test is conduct by placing the two detection devices to face each other. A flat object is again used to simulate the object detection. The alert threshold is set to 10 centimeters The test process is as follows.

Step 1: Place the two drones on the ground with the first drone on the left side and the second drone on the right side with a distance of 2.4 meter apart.

Step 2: Order the first drone to fly to the coordinate (X: 0, Y: 1.2) at the velocity 0.05 meter/second and order the second drone to fly to the coordinate(X: 0, Y: -1.2) at the same velocity. These orders will make both drones move toward each other.

Step 3: When the drones are flying, places a flat object between the two detection devices to simulate an object being detected by both devices

Step 4: Check and record whether the drone can perform the avoidance. This test is repeated 10 times.

The flight paths of the two drones when hovering at the beginning of the test is shown in Figure 20. The two drones have a slight stagger during the hovering.

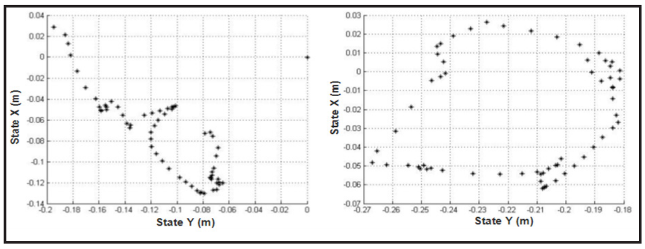


Figure 20. *Flight paths during the hovering of the drones*

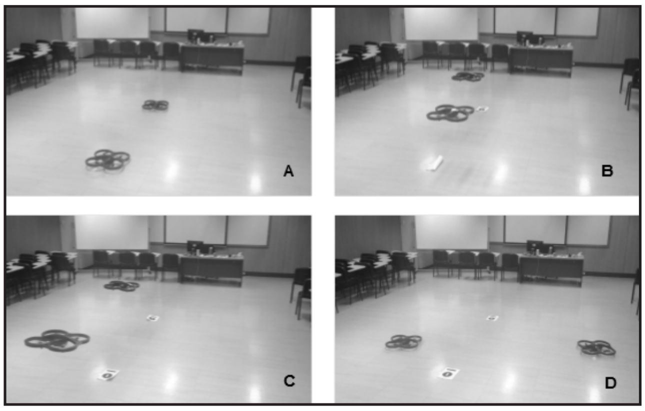


Figure 21. *Flight paths of the two drones before and after the avoidance.*

From the result, the drones successfully perform the collision avoidance 9 out of 10 times. The error of the flight path is caused by the noise in drone sensors and signal interference. As a result, the drones sometimes move out of their path to the destinations in some cases. This error may be solved by modifying the drone operating system. However, the drones used in this research do not allow such modification. This issue is therefore left for future work. The video screenshots in Figure 21 depict the following events from the test.

A. The initial placement of the drones.

B. The two drones move toward each other.

C. The two drones avoid the collision.

D. The two drones land on the ground and stop working.

4.4 Drone payloads test result

Drones have a limitation on payload depending on their factory design. To determine the maximum payload weight, we test the stability of the drone when adding a weight load between 10-100 grams, increasing 10 grams at a time. Each weight load is tested 10 times by ordering the drone to perform a simple movement such as flying in a straight path. If the drone sways too much from its path then we consider that the drone loses its stability.

The result of payload test is shown in Table 7. When the payload increases, the drone loses its stability more often. We suggest that the drone should have at least 70% success rate in maintaining its stability, which reflects a maximum of 40 grams payload.

Table 7. *Success rate of maintaining balance at different payloads.*

No	Payloads (grams)	Stability
1	10	90%
2	20	90%
3	30	80%
4	40	70%
5	50	60%
6	60	60%
7	70	60%
8	80	50%
9	90	50%
10	100	40%

5. CONCLUSION

Commercial off-the-shelf drones have a limitation of human manual control and cannot move automatic by themselves. We propose a mechanism to control multiple drones through a control system with collision avoidance.

The proposed mechanism includes a detection device and a collision avoidance algorithm. The detection device consists of embedded device cooperated with 4 ultrasonic sensors in order to detected objects in 4 directions. The device send detection state to controller to resolve any potential collision.

From our experiment, the detection device work as expected. When tested with a single drone, the collision avoidance success rate is at 93.33 %. With two drones, the success rate is at 90 %.

This research has the limitations of hardware and software. AR.Drone 2.0 uses low cost sensors, which have a high degree of noise. In addition, the drone lose its balances when only a small weight is attached to it. Also, the software development kit of AR.Drone 2.0 does not allow modification of low-level mechanism. Thus, we cannot adjust the sensors to make the controller more efficient.

In our experiment, we have found that the drone velocity of 0.05 m/s is optimal for the alert threshold and ultrasonic sensor range. This issue can be further investigate to determine the relationship between the velocity and the alert threshold in order for the drone to be more adaptive to different situations.

In the future, modification of the detection device may be necessary to reduce its weight. Also, customized drone can be built to allow for sensor adjustment and for drone-to-drone communication to enable fully automatic navigation.

6. REFERENCES

[1] N. A. S. Museum. *Military Unmanned Aerial Vehicle,* 2012.

[2] N. B. Stephane Piskorski, Pierre Eline, Frederic D'Haeyer, *AR.Drone Developer Guide,* 2012.

[3] T. J. Paul Gerin Fahlstrom, Gleason John, *Introduction to UAV Systems,* Fourth Edition ed.: Wiley & Sons, Ltd 2012.

[4] B. P. Power. *Ultrasound in Electrochemistry: From Versatile Laboratory Tool to Engineering Solution:* John Wiley & Sons, 2012.

[5] *Ultrasonic US-100.* Available online at http://www.bananarobotics.com

[6] S. Heath, *Embedded systems design:* Newnes, 2003.

[7] *Arduino UNO Revision 3.0.* Available online at http://www.arduino.cc

[8] *Node.js.* Available online at http://www.nodejs.org

[9] R. W. Beard and T. W. McLain. "Multiple UAV cooperative search under collision avoidance and limited range communication constraints." *In Decision and Control, 2003. Proceedings. 42nd IEEE Conference on,* Vol. 1, pp. 25-30, 2003.

[10] D. Alejo, R. Conde, J. A. Cobano and A. Ollero. "Multi-UAV collision avoidance with separation assurance under uncertainties." *In Mechatronics, 2009. ICM 2009. IEEE International Conference on,* pp. 1-6, 2009.

[11] *TONYLABS CC3000.* Available online at http://www.tonlabs.com

[12] *ESPRESSIF ESP-8266.* Available online at http://www.espressif.com