Lesson # 2
Shark Patrol

Core concepts of this lesson:
- **Loops**
- **Variables**
- **Receiving and comparing sensor values**
- **Time of Flight principle**

Other concepts used:
- **Conditionals**
- **Programmatically moving drones**

Required equipment / preparation:
- **Masking tape**
- **Large movable object to represent a shark. A large inflatable shark would be ideal (wheeled trolley or a cardboard box tied to a string will also work)**
- **Drones + Drone Chargers & Spare Batteries**
- **Laptops with Python IDLE and the CODE4FUN Tello Library installed**

Lesson Scenario

Your team just launched a tech startup in Byron Bay, NSW. You are approached by a local council, whose members are concerned about recent shark attacks. Your task is to develop a **prototype** of a Shark Patrol System for shark beach patrol using a set of drones.

Your simulation should be done in a school hall or a large indoor area. The challenge will be finding the ideal patrol route for the drone and using the built in sensors on the drone alongside Python programming to make sure no sharks get through the allocated patrol area.

This lesson plan is based on this story published by the Washington Post:

Shark-detecting drones to patrol Australian beaches

Show this article & video to the students and discuss it. Talk about possible challenges we will face, given we will need to simulate and showcase this system in an indoor environment, and given our drones do not have downward facing cameras which can actually see sharks.

Table of Contents

Lesson Overview

Explain to the teams that we are going to use a large movable object to simulate a shark. An inflatable shark would be ideal, but a wheeled trolley or just a big cardboard box would also work.

Given our drones have infrared sensors which measure distance to the ground, we can use them to detect ‘sharks’. Students will need to code their drones so that they would fly around a given sector in a chosen pattern constantly checking the distance to the ground. They will also need to code a conditional statement using infrared sensor to detect ‘the shark’. If the distance between the drone and the ground gets suddenly shorter (as if the drone has just flown over a moving trolley or an inflatable shark), the drone will stop in place and do a flip - which will signal the students that the drone has just detected ‘the shark’.

A very exciting activity could be done at the end to test the students’ code: when all drones are able to patrol their sectors and detect the ‘shark’, the sectors can be rearranged into a row, one sector after another, simulating an ocean beach. With the sectors arranged this way, the teacher may drag an inflatable shark, or roll a wheeled trolley or other movable object through the row of patrol sectors. Each drone should detect the ‘shark’ as it passes through their sector and do a flip. Ideally, all drones should do a flip similar to a domino effect, one after the other, as the shark passes through the zones.

How to read this lesson plan:

Note box

This block is used for explaining the concepts used in the lesson. These explanations should be discussed with the students as a group. They generally start with a prompt to ask the students and should be fairly interactive.

Exercise

These blocks are used for additional exercises that students should do, either together as a group, or individually.

This is an informative note that generally gives the reader some hint as to how to teach this lesson, for example common mistakes or things they need to look out for. This is mainly aimed at the teacher, not the students.

Extension Tasks
This block is for extra tasks that can be given to early finishers.

Section 1: Classroom Setup

Setting up the flying area:

Mark out squares on the ground with the masking tape (one per team) these squares should be roughly 3 meters by 3 meters. These areas will represent the sections of ocean that each team will be defending from sharks.

Find an object that will represent a shark: an inflatable shark would be ideal,  but a wheeled trolley, or a wheeled table, or a large cardboard box attached to a string will work.

Be sure to space out the patrolling areas to ensure no drone collisions.

Assign each team of students a drone to work with and have them set their drones inside of the marked squares. Ideal ratio would be 2-3 students per drone. Make sure each team has at least one laptop with Python IDLE and CODE4FUN Tello Library installed.

It is recommended to keep the batteries charging while not in use, to minimise waiting during the lesson

Setting up the coding environment:

Provide each team with one computer and have them open the Python environment and import the CODE4FUN Tello library.

```
from tello import *

start() # Start sending commands to the drone
```

Section 2: Using the TOF sensor

What is a TOF sensor?

The TOF (time of flight) sensor is a way to check the distance between the drone and the ground underneath it.

It works by sending out an infrared signal and timing how long it takes to bounce off the ground (or another object) and return back to the infrared sensor on the drone. Because we know the speed that light travels, we can use the time it took to calculate the distance between the drone and the object.

The **get_tof()** function included in the Tello library returns the distance in millimeters from the drone to the ground (or object underneath the drone).

Let's try printing the value returned from the **get_tof()** function to get the height of the drone in millimeters. We will store this value in a variable called 'result' and then print that variable.

What is a variable?

A variable is something that holds a value. It's called a variable because the value it holds might change. Variables are an important part of programming, and will be used a lot throughout this course. Here is an example of creating a variable, and using that variable in a print statement:

```
message = "Hello!"

print(message)
```

Result:

Hello!

In the above example, we create a variable called message and then print out that variable. Variables must have a name that starts with a letter, or an underscore. It's a good idea to name your variables so that they hint at what the variable contains. Here, we've chosen the word message to represent a variable containing something we're going to print.

Variables can hold many things, including numbers and text. We can make changes to a variable as well:

```
num = 5

print(num)

num = num + 2  # Add 2 to the number

print(num)
```

Result:

5

7

At the end of the above example, our `num` variable now contains the number 7.

```
from tello import *

start()

# Get the value and store it in a variable called result
result = get_tof()

print(result) # Print the value of the result variable
```

Result:

100

Our sensor is not absolutely accurate, therefore when the drone in on the ground the result should be between 0 and 100. Next, let's program our drone to take off and see how this value changes when

the drone is in the air:

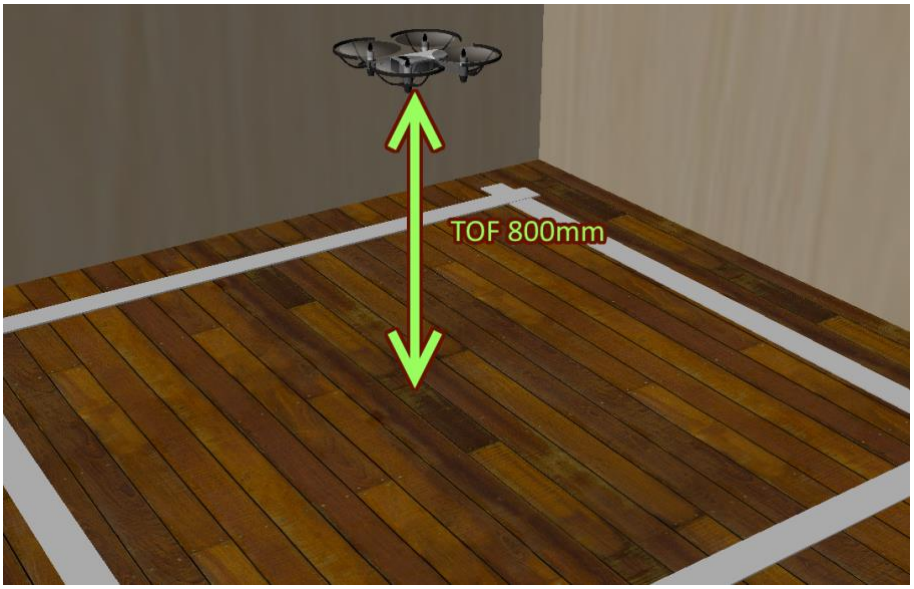
```
from tello import *

start()

takeoff() # Tell the drone to take off

result = get_tof()

print(result)

land() # Land the drone
```

Result:

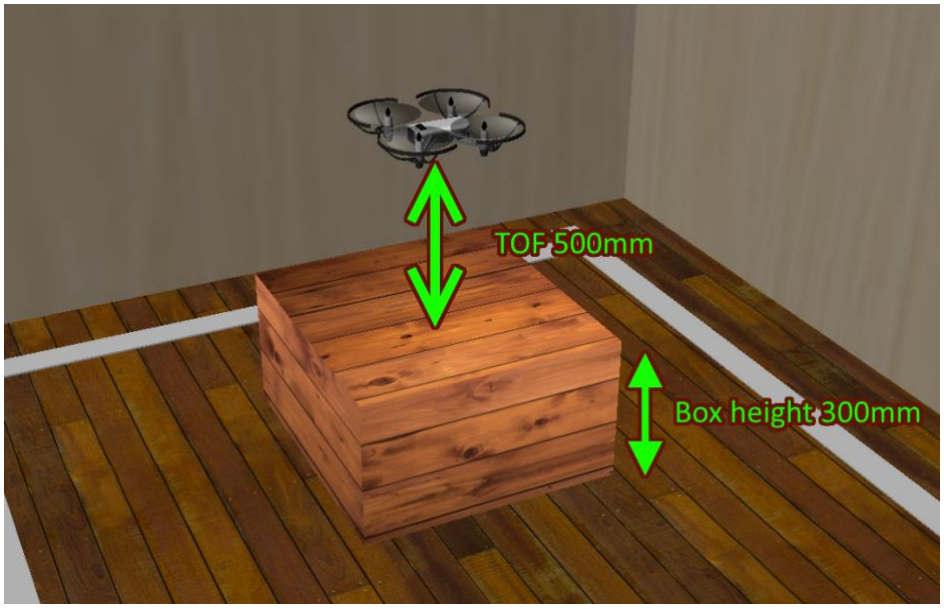
783

The height when the drone is in the air should be around 800 mm.

Later on, we're going to be comparing the TOF sensor value to this initial value, so it's ok if the value here is different to 800.



The way we're going to tell if there is a shark beneath us is if that number is smaller than some threshold, for example we could say if that number was less than 600mm and it's normally 800mm, then it's fair to assume that we have flown over something.



In order to do this, we need to establish a baseline value that we can use to compare to the current value. We'll store the TOF sensor value right after takeoff in a variable called **initial_tof**, so we know what the TOF value is when there is nothing underneath the drone. Later on we can compare this value to the current TOF.

```
from tello import *

start()

takeoff() # Tell the drone to take off

initial_tof = get_tof() # Store the initial TOF just after takeoff

land() # Land the drone
```

Section 3: Loops

We want the drones to continuously check the distance to the ground (or shark), not just once. In order to do this, we're going to use a loop.

What is a loop?

Loops are used to run a portion of code many times, in this case our portion of code that checks the distance to the ground. The kind of loop we're going to look at is called a "for loop":

```
for i in range(5):
  print(i)
```

Result:

```
0
1
2
3
4
```

The print() function here will run 5 times, once for each value of the variable i from 0 up to (but not including) 5.

For this lesson, we're not using the value of the variable, we just want our code to run multiple times.

Using a **for loop**, we can program the drone to check the height many times:

```
from tello import *

start()

takeoff()

initial_tof = get_tof()

for i in range(100): # Repeat 100 times
  result = get_tof()
  print(result)

land()
```

Whilst this code is running, try moving an object or your hand underneath the drone and watch as the distance printed out changes.

Section 4: Testing for the shark

Now that the drone is checking the height multiple times, we can use the value we get from it to determine whether or not it's over an object (inflatable shark or a trolley) .

To do that, we'll use an **if statement** to check if the current TOF is at least 100 mm less than our initial TOF, and if it is - then the drone should do a flip to show us it's spotted a shark. Here is the code:

```
...
for i in range(100):
  result = get_tof()
  # If the TOF is less than our initial value by 100mm
  if result <= initial_tof - 100:
    flip_forward() # Do a forward flip
  print(result)
...
```

Test the above code: let the drone takeoff, and then put a hand or any large object underneath it - the drone should now flip when detecting an object:

Section 5: Moving and Searching

Whilst the drone is moving forward, it cannot also be scanning the ground, as it can only process **one command at a time**. In order for the drone to search an area, it needs to repeatedly move a small amount, then check the height.

Let's program our drone to move forward 3 meters. We will need to use **forward ()** command for it. Let's move 20 cm before we check the height. If we want to fly 3 meters overall, we will need to repeat this 15 times. Let us add this to our code:

```
...

for i in range(15): # Change this from initial 100 to 15
  forward(20) # Move forward by 20 cm, so overall our drone will move 15 X 20 = 300
cm, or 3 meters
  result = get_tof()
  if result <= initial_tof - 100:
    flip_forward()
  print(result)

...
```

With this code, the drone will move 20 centimeters at a time, checking the height each time, until it has done it 15 times. This will take the drone 3 meters from its starting position.

Section 6: More complex movement

Students should use their knowledge from previous lessons to move the drones in patterns within their patrolling area.

Students can experiment with different shapes for the drone to fly in, and see which ones work best.

Here are some examples:

Circle Movement:

In addition to moving 20cm each loop, drones may also be programmed to turn some amount, in order to move in a circle.

```
...

for i in range(15): # Repeat 15 times
  forward(20) # Move forward by 20 cm
  clockwise(24) # Turn clockwise by 24 (360/15) degrees
  result = get_tof()
  if result <= initial_tof - 100:
    flip_forward()
  print(result)

...
```

Square Movement:

To move in a square shape, a **nested loop** could be used to repeat the existing code 4 times, turning 90 degrees each time:

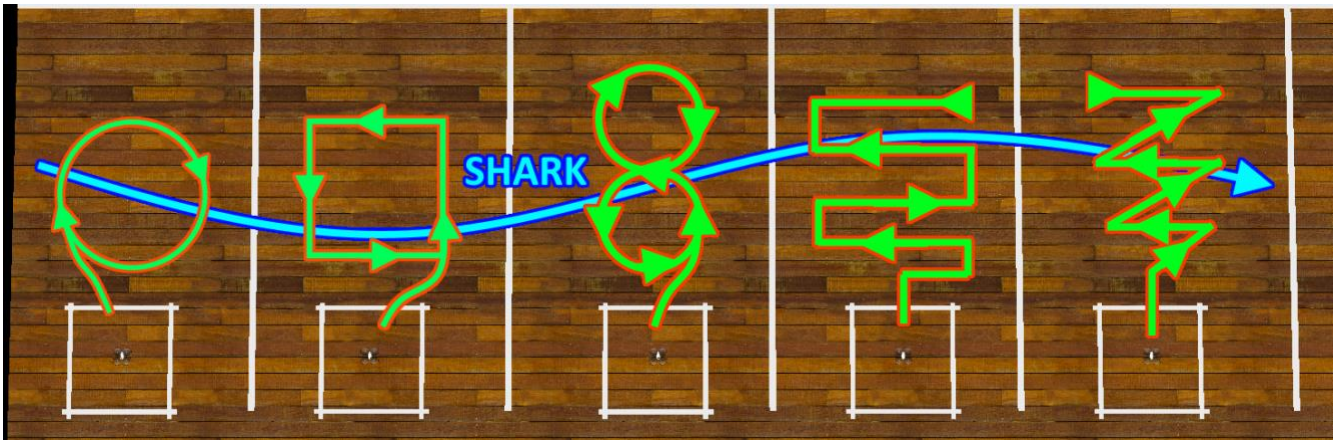
```
...

for j in range(4): # Repeat 4 times
  clockwise(90) # Turn 90 degrees
  for i in range(15): # Repeat 15 times
    forward(20) # Move forward by 20 cm
    result = get_tof()
    if result <= initial_tof - 100:
      flip_forward()
    print(result)

...
```


If students are familiar with using Python's Turtle library, it may be helpful to visualise the motion before trying it with the drones.

A Turtle simulation of these movements can be seen here: https://repl.it/@thefryscorer/Drone-

simulation-using-Turtle

Section 7: Testing the drones with a simulated shark

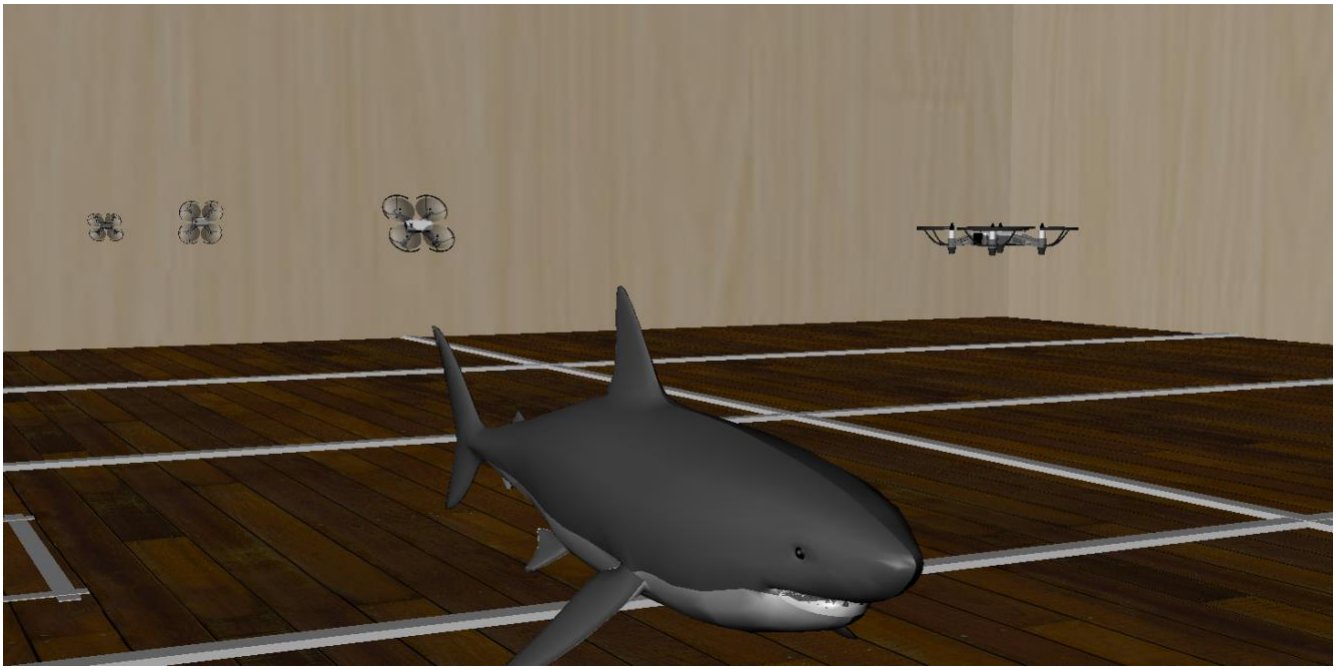


Now that each drone is capable of patrolling a sector, checking the distance between the drone and the ground and an object beneath it, we can test the effectiveness of each team's drone by simulating detecting a shark as it swims across a section of the ocean.

The sectors should be rearranged into a row (see the image above), one sector after another, with a gap in between to reduce the chance of drones colliding. With the sectors rearranged this way, each drone can patrol one section of this "ocean".

A teacher may use an inflatable shark or just a cardboard box attached to a rope, or a wheeled trolley, to simulate a shark moving through this 'ocean area'. The shark should move through each sector in turn, and ideally each drone will detect the passing shark and do a flip to signal that it has detected it.

The shark should move slowly, in order to give each drone enough time to detect it, as the TOF sensor is not capable of detecting fast moving objects, and the drones are only able to detect the shark when stationary (ie, just after the forward command finishes).



Conclusion

Outcomes of this lesson:

- Understanding of the **Time Of Flight** principle used to detect the distance between the drone and an object
- How **loops** can be used to repeat code multiple times
- How **conditional statements** can be used to accomplish tasks
- How **variables** can be used to store a value

Real life applications:

Drones have been used on Australian beaches to detect sharks, and can give advanced warning to swimmers, surfers and lifeguards. The drones used have special software developed to detect sharks among other sea animals and swimmers. This software uses Artificial Intelligence (AI) developed using thousands of images captured by drones, and can distinguish between sharks, surfers, dolphins, whales, boats and many other objects. This software is reported to have a 90% accuracy rate, which is much higher than the accuracy of a skilled lifeguard.

Group discussion topics

- Why are drones used to detect sharks? What do they do better than humans?
  - Artificial intelligence can analyze pictures of sharks quicker and more accurately than human beings.
  - Drones can fly over water and access areas that are difficult for humans to access. Helicopters and other aircrafts can be expensive to maintain.

- Are drones better than other methods for dealing with the threat of shark attacks? For example is it better to use drones than shark nets which ensnare sharks and kill them?
    - Shark nets also kill many other marine animals and sea life.
    - In NSW, the Shark Meshing Program currently protects 51 beaches.
    - Read more about NSW Shark Management program.

- What else could a drone be used to detect?
    - Drones could also be used to detect swimmers in trouble, bushfires, other animals and much more.

Extension Tasks

Challenge students with the following question:

- Which flying pattern would be the most efficient for drones in terms of patrolling the ocean and detecting sharks?

Ask students to program drones to fly in different patterns: spiral, zig-zagging, figure- 8, etc

Ask students to develop a computer simulation program testing the efficiency of different patterns. Use this Scratch project as an example:
https://scratch.mit.edu/projects/259007466/#player

Complete Code (square pattern)

```
from tello import *

start()
takeoff()

initial_tof = get_tof()

for j in range(4):
  clockwise(90)
  for i in range(15):
    forward(20)
    result = get_tof()
    if result <= initial_tof - 100:
      flip_forward()
    print(result)

land()
```