[Isla] Hi, I'm Isla and you're watching coding@home, a regular educational program that you can catch right here on 10 Peach during Term Two and will be available online.

Our mission is to show you just how fun and awesome coding can be.

Last week, we talked about our target audience and how we could design a solution that they might respond to.

We also researched cybersecurity and chat bots and we spoke to a software engineer about how they approach their coding project and tailored solutions for their own clients.

Every day we're solving problems, making decisions.

Perhaps your problem, is that you're hungry, or you're deciding what to watch on TV, and maybe yousolve that problem by making something to eat, or by channel surfing.

Now, your friend calls you and mentions that they're hungry.

"Hey", you say to them, "Have I got a solution for you!"

And you tell them how to make some Vegemite toast.

"You get some bread, spread some butter and Vegemite on it."

"Not too much though."

"Oh wait and you have to toast the bread first."

Your friend's Vegemite toast may not turn out very well.

Sequences are important.

If we want to solve a problem many times over or in exactly the same way as we did last time, we need to record each step and the order we did each step.

In cooking that's called a recipe, in coding it's called an algorithm.

An algorithm is a precise ordered series of steps that can be represented in words or by a diagram. and in this episode you'll be learning how writing an algorithm is the next step for your project which you can use as your entry for the Premier's Coding Challenge.

Did you know just about any problem, can be solved with an algorithm?

After all, an algorithm is simply an ordered series of steps that helps to explain how to do something.

I have an example that can help us learn more about algorithms.

Look at this simple map.

It shows a house, a school and a road.

This is Terri's house.

We can write an algorithm to help Terri travel from home to school.

Let's start by looking at the route Terri needs to travel to get to school.

Leave the house, turn right and walk to the corner.

Safely cross the road, go in the school gate.

This doesn't have to be exact at this stage.

We just want to start with some simple steps and we can add more steps in later, if we need to.

We can call this our, Travel to School algorithm.

Next, we'll use a series of shapes and arrows to turn our steps into a flow diagram.

Let's start with an oval and write what we're trying to do inside it.

Then we write the next step in a rounded rectangle and then connect the oval and the rectangle with an arrow showing the direction of flow.

Now we can go ahead and add all of the steps in the same way.

When we get to the end of our steps, we use a red oval to show we're at the end of our diagram.

Let's see how our whole Travel to School algorithm looks as a flow diagram.

Leave the house. Turn to the right and walk to the corner.

Safely cross the road. Go in the school gate. And You have arrived.

We can turn any algorithm into a flow diagram.

You can see that the arrows flow from one step to another keeping the steps in order.

This is pretty good for our first flow diagram.

Let's evaluate it and see how well it works.

If we look at each of the steps, it's easy to see that some of the steps are more complex than some of the others.

The first step is pretty simple.

All you have to do is Leave the house.

The second step can be broken down into two steps.

Turn to the right and Walk to the corner.

The next step is a little bit more complex - Safely crossing the road.

This step requires a little more thought and I think a decision needs to be made, before we can move on to the next step.

Let's look a little closer and see if we can find where we need to make a decision.

Depending on the road, there might be some help for you to cross.

Crossing lights, pedestrian crossing, a crossing guard.

Sometimes there's nothing at all.

Terri needs to make a safety check and cross the road when it is safe.

We can represent this decision-making part of the process with a diamond box.

This diamond box allows us to make a decision.

There are two answers to this question - Yes and No.

IF the answer is No, we need to do another road safety check.

And we need to repeat this process until the road is safe to cross.

If the answer is Yes, we can move on to our next step.

You can see the Road Safety Check block, and our new decision block.

When we add decisions to our algorithm, we call it branching because it makes our flow diagram kind of look like the branches of a tree.

Before we added the diamond block our algorithm was a straight process from,

'Travel to school' to 'You have arrived'.

The path that we take along our algorithm is dependent on the decisions that we make.

You can see that the 'No' arrow loops back to the Road Safety Check.

When we repeat part of our algorithm, it's called repetition.

Some people also describe it as creating a loop in the algorithm.

So let's look at our whole flow diagram.

Using branches and repetition allows us to make decisions and repeat parts of the algorithm if we need to.

Let's recap.

An algorithm is an ordered series of steps that can be used to help us solve problems.

Algorithms can be represented as a flow diagram.

Branching is when we make a decision in an algorithm.

Repetition is when we repeat parts of the algorithm.

The path we follow through the algorithm is dependent on each of the decisions that we make.

Later in this episode we'll meet some special guests who use algorithms every day in their work with drones.

And you'll get some practice writing your own algorithms with an educational drone specialist.

[Isla] Welcome back to coding@home!

Today we're talking algorithms, what they are and how we can use them in our coding project.

We've already made our own basic algorithm helping Terri to get to school.

One particular piece of tech that makes use of algorithms in their coding are drones.

In order to stay in the air, take off and land safely and avoid collisions, drones need to follow algorithms.

So let's hear from some drone industry professionals on just how they go about using algorithms up in the air.

[Stacey] Hey everyone, it's Stacey here.

And today we're taking coding@home to new heights. That's right.

We're flying high to get a new perspective on algorithms.

[Stacey] Dr. Catherine Ball - It's lovely to meet you.

Now you're obviously very passionate about this technology.

How did you decide that a career in drones was for you?

[Catherine] So being an environmental scientist using drones to capture imagery of turtles on offshore islands gave us real insight into how those animals were behaving without us being there, standing amongst them and disturbing how they might naturally behave.

So we were able to fly long-range Australian drones for hundreds of kilometres.

And that was when I realized that drones were something quite special when it came to monitoring and looking after planet Earth.

[Stacey] And you're the founder of World of Drones and Robotics Congress, why was creating this so important to you?

[Catherine] Well, all drones are robots, but not all robots are drones and one of the things I realized working in the industry, if we really wanted it to become business as usual, to keep people safe and get better data and to create new enterprise and new jobs, we actually needed to get drones to a place where robotics were in terms of how their businesses work.

So I created a business conference based here in Queensland, that was going to be able to accelerate the drone industry so that everyone can use them in their day-to-day work.

[Stacey] So how are drones helping people especially here in Queensland?

[Catherine] Well, Queensland is the number one place in the world for this type of drone technology.

So Google chose to use Google Wing's first commercial services at Logan so you can get bread and coffee and medicines delivered, which is fantastic.

Then if you think about our Rural and Regional Health Services and the

Royal Flying Doctors, being able to get medicines to people rather than taking the whole plane - you could just send a drone.

Then if you think about the Great Barrier Reef, you know, which is very important to Queensland, you can actually have drones that are swimming under the water that can take imagery of the corals that can actually kill crown-of-thorns starfish.

So drones really can be used for pretty much anything you put your mind to.

[Stacey] Drones are pretty complex and use a lot of branches and repetitions in the algorithms.

How have you used these steps in your drones?

[Catherine] When we're flying a drone, say in a field and we're mapping say, a farmer's field and we're looking for weeds versus crops.

So what we want to do is detect individual plants or patches of plants, which we want to then go and spray with herbicide, for example.

Now when writing that code, it's really great to get an accurate representation of - you want the drone to fly that way.

Great.

Then you want it to turn around and come back.

But effectively it's flying the same distance again.

So you just reposition the drone and then repeat what it is that you told it, reposition the drone, repeat what it is that you told it, reposition the drone, repeat what it is that you told it and that's what you call the repetition.

Coding is effectively a language.

So once you've got your sentence grammatically correct, you just ask it to repeat it to repeat it and to repeat it.

[Stacey] Catherine Ball began her career in drones through a passion for environmental science, but for James Miller, it was a different story.

Let's meet a software engineer from Insitu Pacific, an aviation company that operates drones to collect all sorts of information.

[Stacey] So you're using algorithms to code drones and we've been exploring branching and repetition.

How do you use those steps in the drones that you use?

[James] Well, branching and repetition are really sort of key components of any drone's control software.

So, an example that comes to mind is we have a drone similar to this one in the background that searches for things buried underground.

So, the control software would program the drone to actually fly back and forth in a pattern to cover the entire area.

So we use repetition in the code to do that same pattern over and over again.

Another example, say for branching, would be those drones [have] also got sensors on them that detect obstacles and stops the drone from flying into them.

So in programming languages, we call that like an IF statement or a branch.

So IF an obstacle is detected, then the drone will stop.

[Stacey] What's a good example of a feature that you've thought about using in the planning stage of developing a coded solution when working with drones?

[James] Well, depending on, really, how complicated the problem is you're trying to solve will depend on how much time you need planning.

So, an example of something like, an automated landing system, that would make the drone land by itself.

That could take weeks of research and planning, even before we write any code.

[Stacey] What other considerations should we be making when in this planning stage?

[James] I think having a good understanding of the situation before you do something is good for all things in life.

So the same with coding, normally when we start to plan code, we start just with a pen and paper or whiteboard.

So we start writing diagrams or flowcharts to sort of try to help us understand how we think the code might work.

[Stacey] Do the hidden coded features of drones, as opposed to when to when someone is controlling them, require careful planning?

[James] Yes, Stacey they do.

We call these hidden features an autonomous system, so that's when the software in the drone makes the drone control itself.

As soon as you take the human out of the loop, things get a lot more complex and you need to write

a lot more code to let the drone know about its surroundings.

This is from sensors, like cameras, or special sensors that tell you how fast the drone's moving, or its position, or how far away it is from things.

So systems like these need a lot of planning and testing to make sure that the software can actually handle all the situations that the drone can be in.

[Stacey] Private Mackenzie Togo and Gunner Lewis Day are drone pilots in the Army Drone Racing Team.

Some of these drones race at up to speeds of 260 kilometres per hour.

And if you think that's impressive, they build and code these drones too.

[Mackenzie] the best part about being on this team is building your own drone.

So we start from all the basic components, solder them together and eventually code it to be able to fly the way that we want it to.

So races are super fun.

There's four people in a race and basically, it's to get across the finish line first.

So there's a lot of drones in the air.

And for that reason, we need to think about how we code our drones because we want to make sure that the rates are right for us.

So the first step, when planning to build a drone is determining what the drone is going to be used for.

In this case.

It's racing.

So they need to be very, very fast and agile.

If we were to build a freestyle drone, which we'd use to to capture footage, it needs to be nice and slow and deliberate.

It's very important to be able to change the code between the freestyle drones and the racing drones.

But we do this using some coding software on a computer and fine-tuning those algorithms to improve the flight characteristics of our drones.

[Stacey] They're not just fun to fly, but drones are also helping the Army.

[Lewis] The Army actually possesses a number of drones.

We have about three main systems.

We have the bigger drones which are used to give a bigger team a larger reach for the intelligence reconnaissance and surveillance kind of aspects.

We also have these smaller drones like the Black Hornet, which is used at a small team level for more in an urban environment.

So the Black Hornet is one of the most recent drones the Australian Army has acquired.

It has an approximate flight time of 25 minutes and about a range of up to two k's weather-dependent.

Its noise signature is extremely low. It can sit about 10 metres above your head and you would not know it was there. Upon launching, the Black Hornet

UAV it notates its location using GPS.

From there, the user can either fly the Black Hornet manually or it'll fly on a predetermined flight path.

If in the event, the Black Hornet loses GPS, using its GPS algorithm, it will then gain altitude to regain GPS signal.

Also, if it loses connection with the user, it will then fly back to the user using its Return To Home algorithm.

[Isla] We've done our research and now it's time to more closely build, develop, and evaluate the algorithm for our specific audience.

[Somayeh] Hi everyone. I'm Somayeh Hussaini a QUT STEM Ambassador for the QUT STEM for Schools Program.

I'm in my final year of Mechatronics Engineering, my areas of interest are unmanned aircrafts and machine learning.

So what are drones?

Well in industry, we refer to them as remotely piloted aircraft systems or unmanned aircrafts.

Essentially, they are aircrafts operating remotely without a human pilot onboard.

They rely on communication links, sensors, hardware, and algorithms to fly autonomously.

Many of you would have seen drones already, typically small with 4 to 8 spinning rotors that fly a bit like a helicopter.

Well, drones range in size and configuration and perform a wide range of beneficial applications.

Right now in Australia, we have drones as big as commercial passenger jets flying at extremely high altitudes capturing data, like a satellite.

At the other end of the scale, we use small multi-rotor drones for firefighting, police work, farming, agriculture, and inspecting infrastructure, like dams and roads.

Key to it all, is automation. Making drones smart.

Drones are packed with sensors that enable them to perceive the world around them, and then intelligently respond to that information.

To do this we need algorithms.

To give you an example, we're going to explore how to develop an algorithm for a delivery drone.

Did you know, Australia was one of the first countries in the world to have commercial drone delivery service?

You can order fast food via an app on your phone and have it delivered to your front, or back yard.

So, how do you write algorithms for something like this?

Well, we start with three building blocks for any algorithms.

Branches. This allows the drone to make decisions based on sensory inputs.

Repetition. Also referred to as Loops that allows the drone to continue an action if required.

Error Checking. Making sure the code is robust to errors.

You can't have your drone falling out of the sky because of an error in the algorithm.

So we're going to build an algorithm for our drone.

The result is a blueprint which we can then use to write our software code.

We can describe our algorithm in plain English, which we call pseudocode, or graphically as a flow chart.

Let's start with the take off. Before we take off we need to make sure that the remote pilot has authorised the drone to take off and that it has the right order loaded.

If not, we don't want to take off.

This situation we can describe as a branch - a decision or condition to take off or not.

We could describe that as,

IF the pilot has given us approval to fly,

AND we have our delivery order loaded on board, THEN take off.

OTHERWISE do not take off.

The next thing the drone does will be to climb up until it reaches its minimum cruising altitude.

A drone uses a pressure altimeter or GPS to measure its height above the ground.

We would expect a drone to climb up until it reaches a predefined cruising altitude.

Well what algorithm building block could be used here?

Well, the trick is in the description.

I said, "Continue to climb UNTIL it reaches its minimum cruising altitude."

UNTIL implies that we need to do something again,

and again or repetitively UNTIL a condition is met.

In this case, we want to keep climbing UNTIL the drone reaches the required cruising altitude.

This is a repetition.

Do something over and over UNTIL.

So, let's write that in our pseudocode, continue to climb UNTIL

current height is equal [to] or greater than cruising altitude.

We could also write this in terms of checking the outputs of the drone's height sensors.

Climb UNTIL current sensor value is equal to or greater than cruising altitude.

Sensors aren't perfect.

And neither is the human remote pilot.

What happens if the remote pilot enters a negative value for cruising altitude?

We would need to make sure that only positive values could be entered.

Otherwise the drone would never take off.

We could do this error prevention and checking a number of ways.

Perhaps initially, when the drone is powered on, we would want to check the value of the cruising altitude.

If it was negative, we could send the pilot an error message and not allow the drone to take off.

So what sort of algorithm building block could be use for this?

It's a check something to do something, or something else, style of question.

So this is a branch.

For something different, let's show this branch as a flowchart.

Check the value of cruising altitude loaded in flight plan,

IF less than 0 or negative, THEN Send the error message and abort take-off.

We can use the same simple elements of branches and repetition to implement all sorts of behaviours and intelligence in our drones.

You could try doing some algorithm planning at home with the remainder of [the] delivery drone's tasks.

WHILE the Drone is flying, it continually uses its onboard sensors to check for obstacles that it could collide with.

IF it detects something it will stop or try to fly around it.

When the delivery drone gets to its destination,

It has to check no-one is below it before it can deliver the package.

It uses little cameras to scan the area.

It will hold, continually scanning, until the area is clear, before delivering the package.

The drone lowers the package down on a cable.

It uses a tiny onboard winch to lower it.

It continues to lower it until the package touches the ground.

It knows when the package is on the ground, when the load on the winch is zero.

The person who ordered the package uses the app to tell the drone they have taken the package.

The drone hovers and won't start retracting the cable UNTIL it receives the notification from the user.

[Isla] Maybe you can write your own algorithm for something you do every day.

Maybe it's for brushing your teeth or getting dressed for the day.

What might the flow diagram for that look like?

Where would the branches be?

And what steps would need to be repeated in order to avoid any problems?

Stay cool! Look out for each other and we'll catch you next week.

Authorised by the Queensland Government, Brisbane.