saw™
science art writing

Build your own DNA Dave

Complete Handbook

CONTENTS

Front cover image designed by macrovector/Freepik

Introduction

Robots once seemed like the stuff of science fiction but in a little over 100 years we have seen robotics develop from simplistic conveyor belts in factories speeding up the assembly of products such as cars, to present day humanoid robots capable of machine learning. Robots are important tools for society, from performing basic tasks in areas too dangerous for humans to carrying out complex actions such as surgery, robots provide solutions and our increasing understanding of Artificial Intelligence (AI) is bringing us closer to developing things that have the potential to change the way society fundamentally works.

To build a robot, first you need to identify the purpose, what job or jobs will the robot do?

Then it takes a combination of skills including electronics, engineering, coding and design to bring it to life. In this project the function of the robot is as an educational tool to explain the processes that take place inside cells to "read" DNA code and make products. We made a robot to do this at science festivals to help explain key steps in the process to the public and we called him DNA Dave!

Inside this workbook you can find all the information on how DNA Dave was designed and built and the coding that was written to make him work.

This provides you with a blueprint that you can follow to learn how to build your own DNA robot but we would encourage the following;

1, Consider the environmental footprint of the materials you use – reuse & recycle!
2, Be creative when designing the appearance of your robot.
3, We included some steps of the biology we wanted to explain to the public but there are more parts to the story that you could add to your robot.....
4, Think about power usage, could renewable energy feature?
5, Storing a robot can be an issue, DNA Dave's head and arms are stored in his body, so consider how you can use clever design to make storage easier.

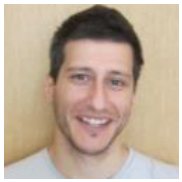
The Team

To build our robot a diverse team of professionals from a variety of backgrounds with different skills collaborated to create DNA Dave

Molly Barrett

Artist

Molly studied Theatre Design at Nottingham Trent University, and applies her skills and knowledge to a range of arts disciplines including set design, installation art, puppetry and arts workshops



Ioannis Tamvakis

Computational Scientist

Ioannis is currently a PhD student at the University of Cambridge studying plant architecture. He also uses coding, art and design in his work.

Dr Nadia Radzman

Plant Biologist

Nadia has a PhD in plant science and has previously been involved in interdisciplinary research involving computational and systems biology. She currently works as a post doctoral researcher at the Sainsbury Laboratory, Cambridge University.

Dr Jenni Rant

SAW Trust Programme Manager

Whilst studying for her PhD in plant pathology at the John Innes Centre, Jenni became interested in science communication and began volunteering for the Science, Art and Writing (SAW) Trust, which she now runs full-time as programme manager

Dr Colette Matthewman

OpenPlant Coordinator

Colette worked as a post doctoral researcher in Denmark before returning to the John Innes Centre in Norwich, to become the coordinator of OpenPlant, a synthetic biology research centre.

Marco Graziano

Illustrator

Whilst studying for his PhD in evolutionary ecology of fish and population dynamics at the University of East Anglia, Marco's love of nature has led him to become a scientific illustrator in his spare time, with an aim to represent the natural world for all to see.

So why did we do it?

Science Communication

Science communication is a broad field that aims to inform, educate and create discussions around science related topics.

Effective science communication can dispel myths, educate, inspire and refine future research questions. Science communication has grown hugely in the last few decades, fueled by interest in the increasing role science plays in our lives.

By building bridges between the scientific community and the general public, we can give those who have no scientific background a voice to take part in debates and topics that affect their everyday lives.

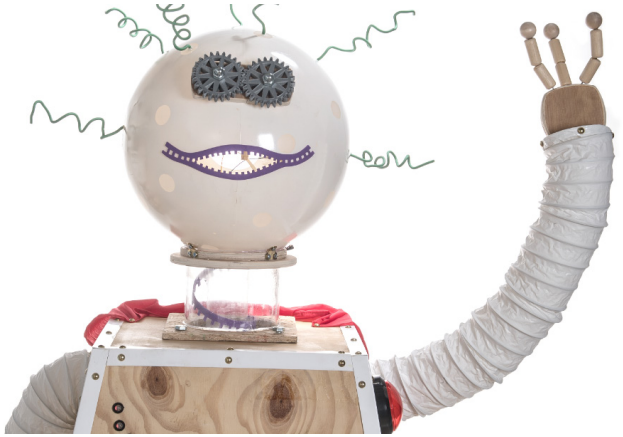
DNA Dave is a science communication tool. Initially designed to assist with explaining DNA transcription and translation, it is also used to successfully introduce synthetic biology as part of workshops, presentations and exhibitions.

For more examples of science communication see www.sawtrust.org

Introducing DNA Dave

DNA Dave was designed to be an approachable, engaging robot that helps explain the key principles of DNA transcription and translation. His build required the use and understanding of coding, electronics, DNA transcription and translation, and design skills.

Before you begin to build your own robot, lets have a look at how the original DNA Dave was built to provide a way for people to be able to interact with him.



Step 1.

DNA Transcription

When interacting with Dave the first step is to press and hold the button on Dave's side, this causes a motor (stored inside Dave's head) to spin Dave's eyes and neck around.
Dave's head represents the nucleus of the cell, where DNA transcription takes place. The movement of the eyes and neck symbolise the DNA code being read and the production of single-stranded RNA which leaves the nucleus via pores in the membrane, represented by Dave's neck.

Step 2.

DNA Translation part 1

Next, turn the small cog as indicated on Dave's body, this will make the bar underneath slide along and cause a line of LED lights to light up.

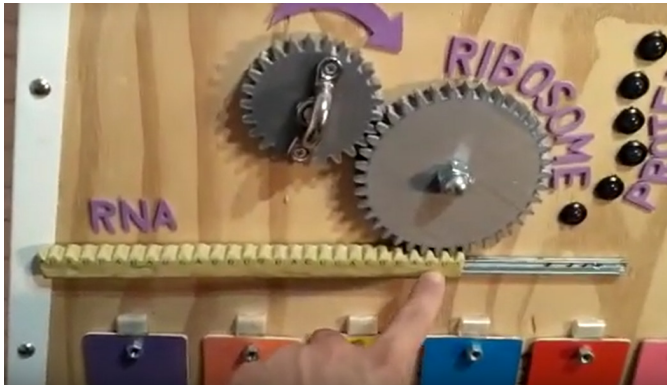
Dave's body represents the cytoplasm of the cell, where DNA translation takes place. The cogs represent the cell ribosome which move along the RNA and gathers the correct amino acids needed to build a protein.

Step 3.

DNA Translation part 2

If the cog fails to reach the end of the RNA bar, an alarm will sound and the LED lights will go out. To reset this simply turn the cog backwards to return the bar to it's starting position.

This signifies the presence of a mutation in the RNA sequence preventing the ribosome from producing the protein.

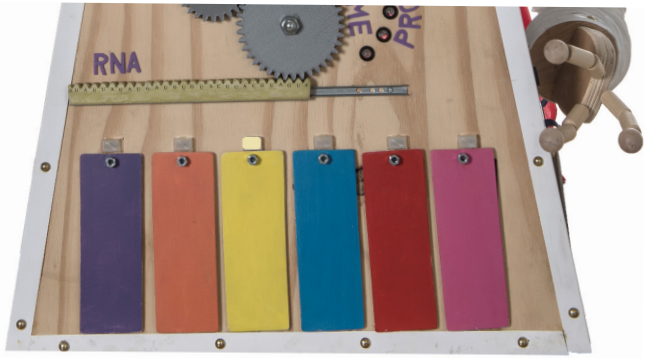


Step 4.

DNA Translation part 3

Once all the LED lights have lit up, the lights above the 6 brightly coloured doors will flash in a random order until just one remains on. You can now open the door to reveal the product.

Having acquired all the amino acids required to produce a protein. The amino acid chain then folds to produce a more complex structure - a protein!



What's Next?

Now that we have looked at how DNA Dave is used to explain the process of DNA Transcription and Translation, lets look at:

Part One

The Biology

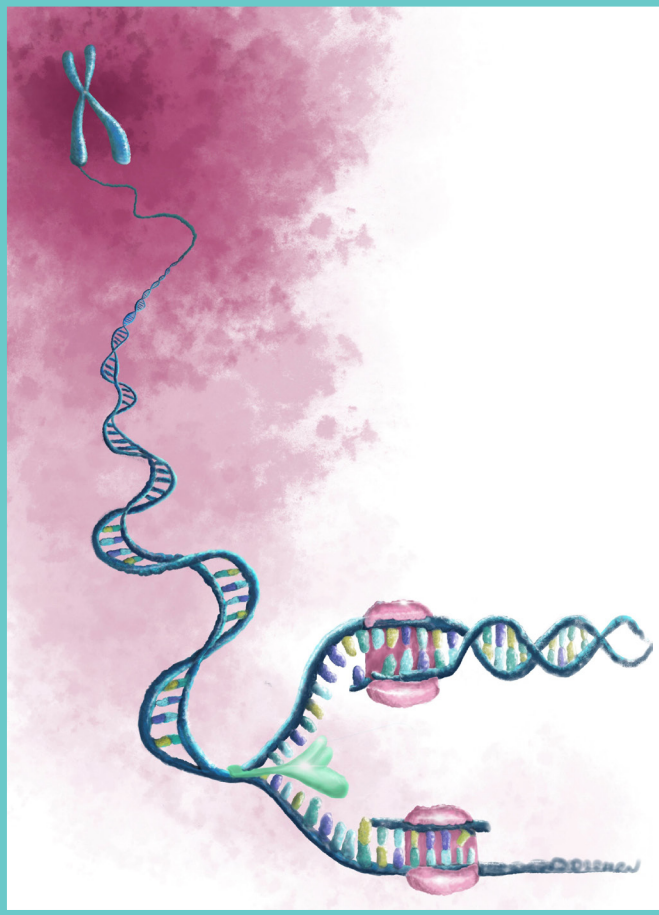
Understanding how proteins are made.

Part Two

Coding and Electronics

Have a go at coding using a BBC micro:bit and learn how DNA Dave works.

Notes



Part One
The Biology

Understanding how proteins are made.

Part One

Illustrations by Marco Graziano

The Biology

The scientists who worked together on the DNA Dave project were all part of OpenPlant, a large collaborative research programme between the John Innes Centre and the Earlham Institute in Norwich and the University of Cambridge. OpenPlant is one of six national centres researching synthetic biology and specialises in plants.

Synthetic Biology offers the potential to reprogramme biological systems to improve access to important natural products such as food, medicines and fuel using more sustainable, less polluting processes. Natural processes are tightly regulated by instructions found in an organism's genetic code. Synthetic biology picks apart the machinery required to read and follow the genetic code to make products, breaking down each step into a series of parts that enables exploration to reassemble parts in new ways to manipulate biological systems.

To introduce synthetic biology to a general audience it is important to first explain the role the genetic code plays in making products. DNA Dave was built to help the OpenPlant scientists explain this at science festivals and this chapter focuses on the biological steps that DNA Dave represents.

OpenPlant website - https://www.openplant.org/
Hackster project page - https://bit.ly/33fClf6

Cells

Living things are made up of cells. The word cell comes from the Latin word 'cella', meaning 'small room' and these small rooms are packed full of the coolest machinery! Plant and animal cells keep their genetic codes locked up in the cell nucleus while more simple, single-celled organisms like bacteria have no nucleus.

Multi-celled organisms like us have copies of our entire genetic code in almost every cell in our body, meaning each cell in our body has the ability to become anything i.e. skin, hair, gut etc.

Lets look at some of the key components in a plant cell and a description of the jobs they do:

Cell wall

This gives the cell its structure. Made of cellulose, this carbohydrate wall provides protection but also acts like a filter allowing nutrients, waste and ions to move between neighbouring cells through the permeable **cell membrane** *or through the* **plasmodesmata**.

Chloroplast

These are only found in plant cells where they capture energy from sunlight and convert it into sugars for plants to use in a process called photosynthesis.

Vacuole

The vacuole acts like a storage facility for cells but also helps a cell to maintain its shape.

Endoplasmic reticulum (ER)

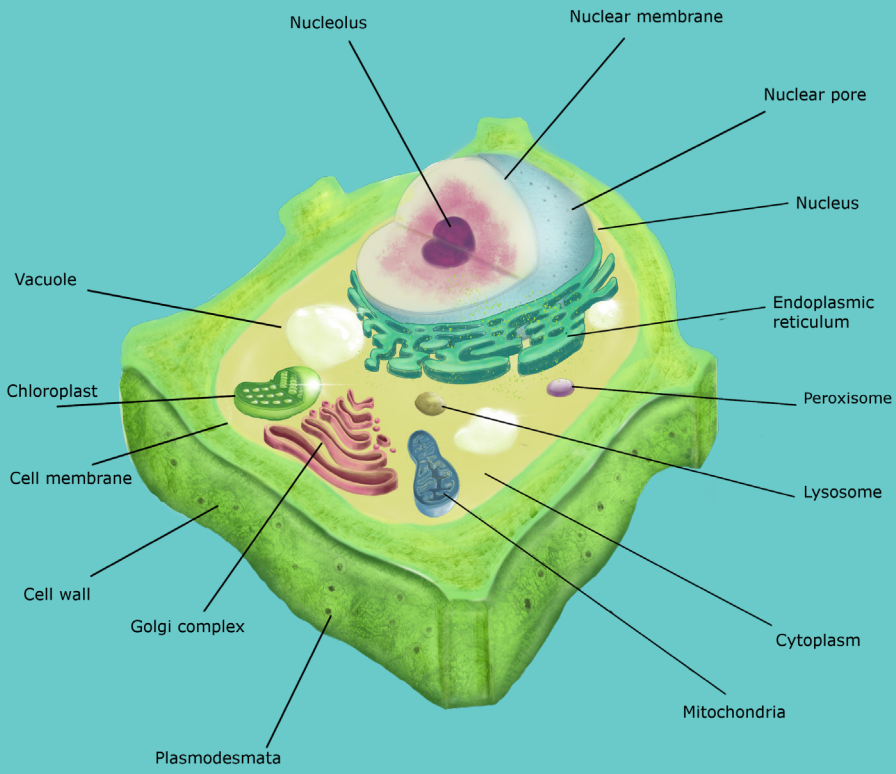
Attached to the cell nucleus, the ER is responsible for the transportation of proteins and other carbohydrates to other parts of the cell machinery for further processing.

Mitochondria

These are the power packs of the cell, converting food and oxygen to energy through a process called respiration.

Peroxisome

Free floating in the cell cytoplasm, peroxisomes produce hydrogen peroxide whilst breaking down organic molecules by oxidation to convert them to oxygen and water.



Nuclear membrane

This is a double membrane made of lipids that surrounds the nucleus. It contains **pores** *that allow chemicals to pass in and out of the nucleus.*

Nucleus

This is the control centre of the cell. It is a membrane bound organelle that contains all the cell's genetic information - its DNA that controls cell growth and reproduction.

Nucleolus

This membrane-less structure is found within the nucleus and consists of RNA and proteins. Its main function is to transcribe DNA to produce ribosomes.

Cytoplasm

The jelly-like substance that makes up the bulk of the cell and contains the cell organelles.

Golgi complex

This stack of flattened membranes is the manufacturing and sorting centre of the cell. Protein-containing vesicles fuse with the membrane allowing the protein to travel through the golgi complex where they are modified into various products such as glycoproteins. These products then leave the golgi in vesicles which travel through the cell..

Lysosome

These are small, membrane-bound organelles that contain enzymes for degrading worn out cell parts or other unwanted or invading items.

DNA Transcription and Translation

We have looked at the parts of a cell and found that DNA is stored in the nucleus. Deoxyribonucleic acid, or DNA for short, is the helix-shaped molecule which contains the code of life. It is composed of four nucleotides or bases: A, T, C and G which spell out this secret code but how do you unravel its hidden message? The answer lies in following its journey as it gets turned from code into the proteins on which life depends.

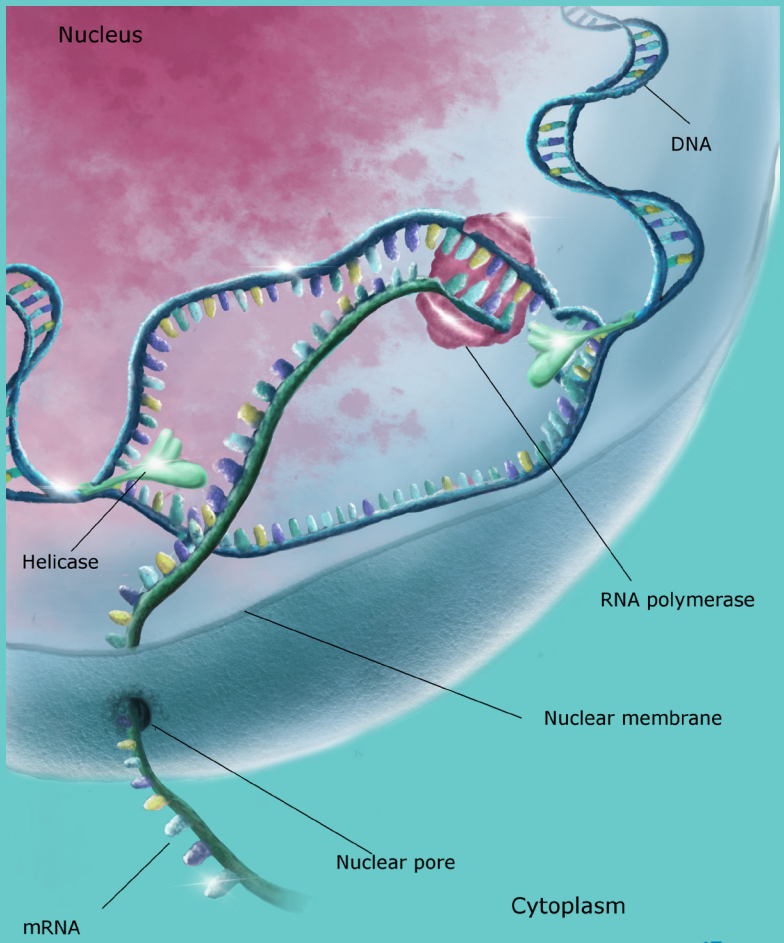
DNA Transcription

Where: Cell Nucleus

Inside the nucleus DNA is tightly wound into a double helix and forms structures called chromosomes. The DNA on chromosomes is divided into sections called genes, where each gene conveys a single instruction.

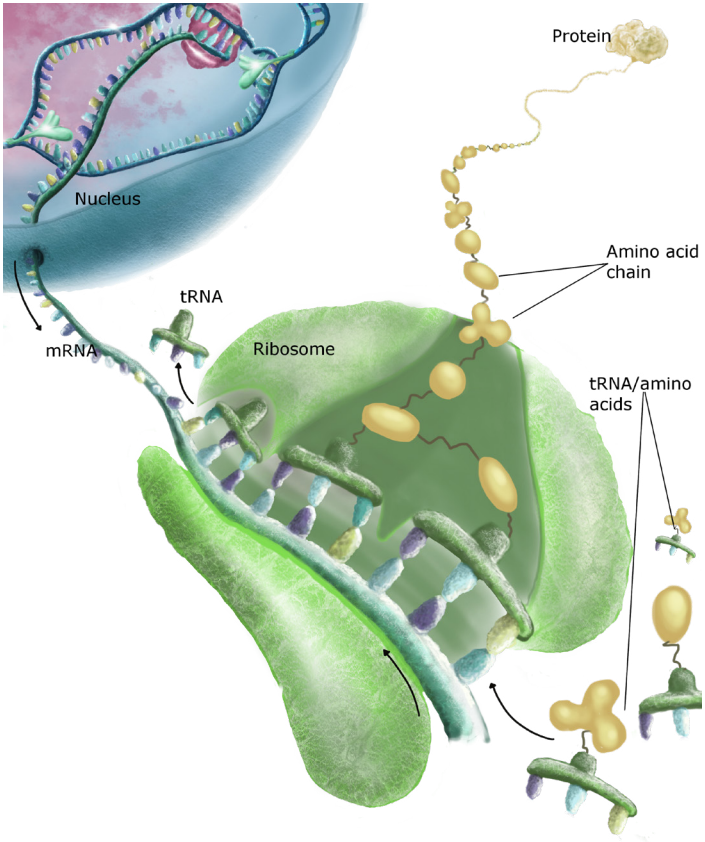
For the DNA code or sequence to be read, an enzyme called helicase relaxes the tight coils of the helix to allow the RNA polymerase enzyme to access the start site of the gene. The double-stranded DNA then "unzips" in the region of the appropriate gene.

The RNA polymerase molecule then moves along one of the unzipped strands and assembles a messenger RNA (mRNA) strand, using free nucleotides found within the nucleus. This single strand of mRNA is a complimentary copy of the gene which then leaves the nucleus through a pore in the nucleic membrane.



Nucleotides

*Building blocks of DNA, which consist of four different bases (**A**denine, **T**hymine, **G**uanine or **C**ytosine) plus a molecule of sugar and one phosphoric acid.*



DNA Translation

Where: Cell Cytoplasm

Once the mRNA enters the cell cytoplasm, a cell structure known as the ribosome bonds with it.

The ribosome begins reading the sequence of mRNA in blocks of three nucleotides known as the triplet code (codons), which translates into 20 amino acids.

The ribosome reads the codons and attaches the correct amino acids together to form a polypeptide chain.

Once the ribosome reaches the "stop" codon, translation ends and the resulting chain of amino acids is released.

The amino acid chain can then fold in a specific way to form the final protein.

The Genetic Code

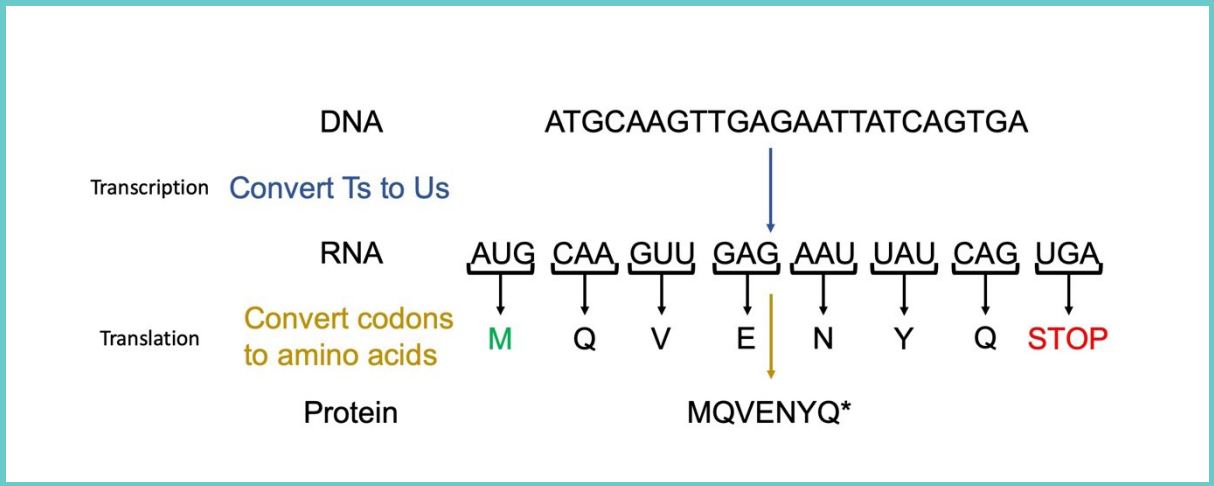
So as we have learnt DNA undergoes a process known as transcription where the code is copied into messenger ribonucleic acid (mRNA) by an enzyme called RNA polymerase in the nucleus of the cell. The mRNA is very similar to DNA using bases to spell the code however the Ts are turned into Us so a code written: ATCGCGTA now becomes: AUCGCGUA but the message remains the same.

This single-stranded mRNA can leave the nucleus through a pore and travel to ribosomes in the cytoplasm to undergo the second step. This is called translation where the mRNA is read by the ribosome and, following the code, a string of amino acids is constructed. This string can then fold into the final protein.

DNA has a few key features. Firstly, it is triplicate, that is it takes three bases (A/T/C/G) to code for one amino acid. Together these three bases are known as a codon. Secondly, it is degenerate, what this means is that more than one codon can code for the same amino acid. For example, both the codons CAA and CAG code for the amino acid glutamine.

Finally, just like a sentence the DNA has a start and an end point that define a 'gene', called Start and Stop codons. Almost always the Start codon is ATG (methionine) and the Stop codons can be either TAA, TAG or TGA. These don't code for an amino acid but cause the ribosome to stop building the chain and so finish off the protein.

How DNA is read in a triplet code



So how can we look at DNA and tell what amino acids the protein will contain?

Well you follow this process:

1. *Turn the DNA into RNA by changing the Ts to Us*
2. *Use the codon table (opposite) to unravel the triplicate code*
3. *Spell out the final protein*

Codon table

1st Position	2nd Position				3rd Position
	U	C	A	G	
U	Phenylalanine (F)	Serine (S)	Tyrosine (Y)	Cysteine (C)	U
	Phenylalanine (F)	Serine (S)	Tyrosine (Y)	Cysteine (C)	C
	Leucine (L)	Serine (S)	STOP	STOP	A
	Leucine (L)	Serine (S)	STOP	Tryptophan (W)	G
C	Leucine (L)	Proline (P)	Histidine (H)	Arginine (R)	U
	Leucine (L)	Proline (P)	Histidine (H)	Arginine (R)	C
	Leucine (L)	Proline (P)	Glutamine (Q)	Arginine (R)	A
	Leucine (L)	Proline (P)	Glutamine (Q)	Arginine (R)	G
A	Isoleucine (I)	Threonine (T)	Asparagine (N)	Serine (S)	U
	Isoleucine (I)	Threonine (T)	Asparagine (N)	Serine (S)	C
	Isoleucine (I)	Threonine (T)	Lysine (K)	Arginine (R)	A
	Methionine (START, M)	Threonine (T)	Lysine (K)	Arginine (R)	G
G	Valine (V)	Alanine (A)	Aspartic acid (D)	Glycine (G)	U
	Valine (V)	Alanine (A)	Aspartic acid (D)	Glycine (G)	C
	Valine (V)	Alanine (A)	Glutamic acid (E)	Glycine (G)	A
	Valine (V)	Alanine (A)	Glutamic acid (E)	Glycine (G)	G

Below is a test sequence of DNA:

ATG GAT AAT GCG GAC GCC GTA GAA TAG

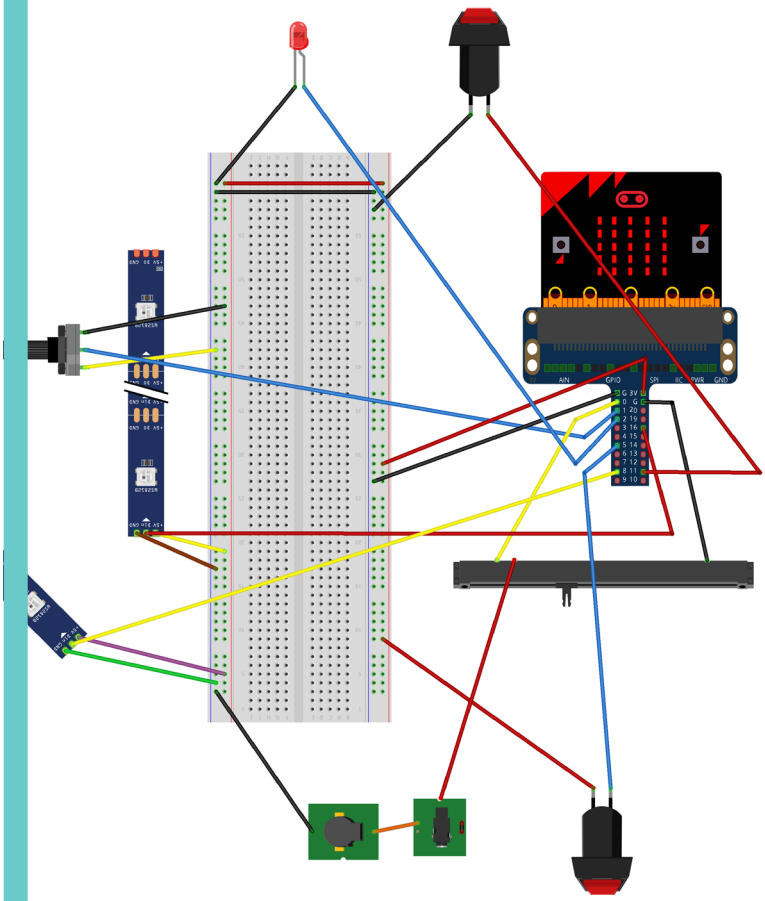
Using the codon table provided above, can you figure out the single letter amino acid code? - Dont forget that the T's have become U's!

The answer you should get is: MDNADAVE

*We have to have the methionine at the start and the STOP codon doesn't code for anything but the end. Often we use a * to indicate the STOP so it would really be;*

*MDNADAVE**

Notes



Part Two Coding and Electronics

Have a go at coding using a BBC micro:bit and learn how DNA Dave works.

Part Two

Coding and Electronics

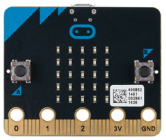
To make an engaging tool for people to interact with to explore the processes of transcription and translation we incorporated buttons, cogs and lights to tell the story and bring DNA Dave to life . These parts are central to the way DNA Dave was built and are operated by code on a BBC micro:bit board.
We divided the story into three parts;
1, Transcription of DNA into mRNA represented by Dave's spinning eyes and neck.
2, Translation of mRNA into protein represented by turning cogs on Dave's stomach.
3, Final protein production represented by flashing lights over compartments at the bottom of Dave's body that are opened to reveal the final protein made.

You can choose how you would like to tell the story and where you will place lights, buttons and cogs to start the different sequences on your robot. Therefore, this workbook will teach you how to code your BBC micro:bit to make representations of DNA and protein strings, to better understand the concepts involved at the heart of biology, whilst also learning some coding!

Hackster project page - https://bit.ly/33fCIf6

Micro:Dave

Coding practical

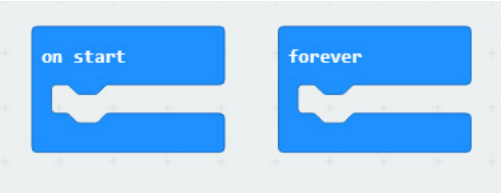


You will need:

a micro:bit
a computer with internet access, to get to the MakeCode website
a USB cable to connect the micro:bit
a LED string with individually addressable RGB LEDs
alligator clamps and cables to connect the micro:bit to the LED string
courage to learn to code!

Lets go first to the makecode website (makecode.micro:bit.org) and choose to code using the Makecode Editor. The editor has a lot of easy examples to learn how the micro:bit works and how we code for it, and we suggest you try them out. For now, let's start a new project.

You will notice the basic layout of the program at the right of the screen. There are two blocks of code, one that will execute "on start", only once, and one that will execute "forever" meaning that everything inside it will be executed again and again for as long as the micro:bit is powered:



Lets make a DNA string!

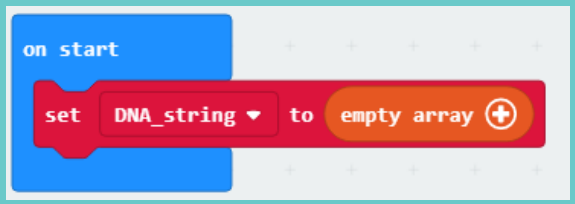
Part 1

This DNA string will be just letters (A, T, G, C) representing the nucleotides, one after the other in a text list. Lets go to the "Arrays" tab (if you can't see it, click "Advanced"), and click on the second choice, "set (text list) to (array of "a" "b" "c" - +)".

We want to put the text list we declare in the "on start" block, so that our micro:bit starts up and defines this list, and so it can remember and use it all the time.

We also change the name of the variable from "text list" to "DNA_string" by pressing on the "text list"button and renaming the variable.

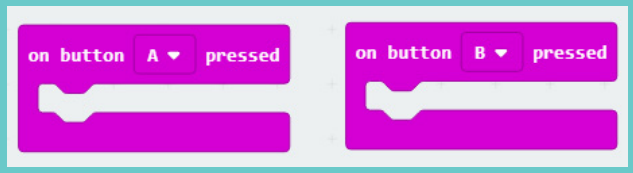
Also, press on the (-) minus sign at the end of the declaration to make the DNA_string be empty at the start. You should get something like this: (It might take 3 presses of -)



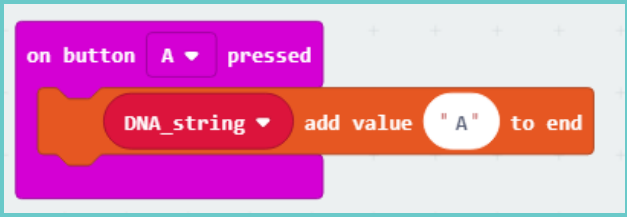
There are many ways to get to this result, so do not worry if you did it another way! We just need to define a variable called "DNA_string" that is an empty array (a list is a simple array).

You did it! Now the micro:bit will always think about a DNA_string. Now, let's make it so that when we press on the buttons of the micro:bit (called button A and B) we can add "nucleotides" in the DNA_string, which means we will just add a letter to the text list called DNA_string. For this we need to add the special blocks of makecode that define what happens when we press buttons.

We can find them in the "input" tab. Let's put in two of them, one for button A and one for button B:



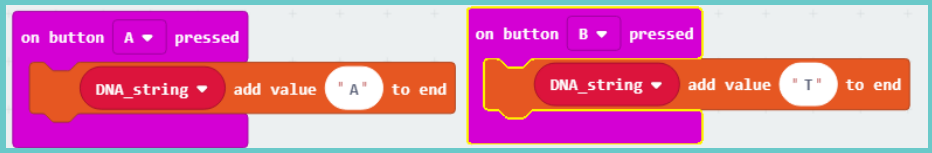
When we press a button, we want a letter to be added to DNA_string, so, we go to the "arrays" tab, and we add the "(list) add value () to end".



We add it inside these new blocks, and change it so that it adds a new letter to DNA_string. In order to get the letter there, we need to find under the "Text" tab the empty text block, and put an "A" in to it. You should get something like this:

You did it! Now when you press button A on the micro:bit, it will add "A" to the end of the DNA_string, so its like you have added a nucleotide in your DNA string!

Now do the same for button B, but add "T" instead. It should look like this:



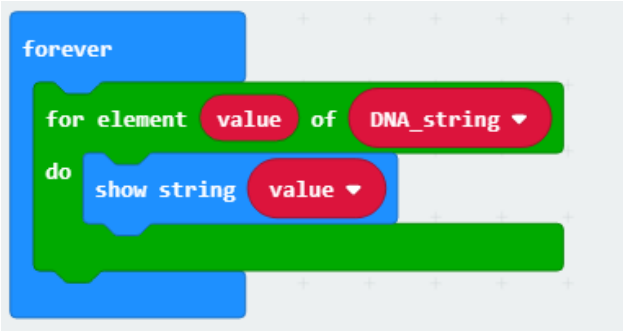
Ok, let's recap: you have made the micro:bit always remember a DNA string, and when you press button A you add an "A" to it (representing an Adenine) and when you press button B you add a "T" (Thymine). That's great!

All we need now is a way to see what the micro:bit remembers and we have completed a very simple program. Luckily for us, the micro:bit comes with a little screen at the front, consisting of a few red LED lights. We can display rolling messages there. We would like to see what the micro:bit remembers, so let's instruct the micro:bit to always display the DNA_string there, one letter at a time.

Under the "basic" tab, there is the "show string ()" block. Let's use it inside the "forever" block, so that it always displays the DNA_string without ever stopping. We need to display all the letters in DNA_string, so let's use a loop to do that! Loops are ways to do something multiple times, or for all the items in a list, etc. They are very helpful. Under the "Loops" tab, use the "for element (value) in (list)" block. Change the (list) to (DNA_string).

Now the loop will do something for every value of the list "DNA_string". Perfect. Inside this loop block, let's put the "show string ()" block, and instruct it to show the (value).

You can drag-and-drop the (value) block. You should get something like this:



That's it!

Now your micro:bit computes and remembers something that resembles a simple DNA string that is made up of only A and T, and displays that for you.

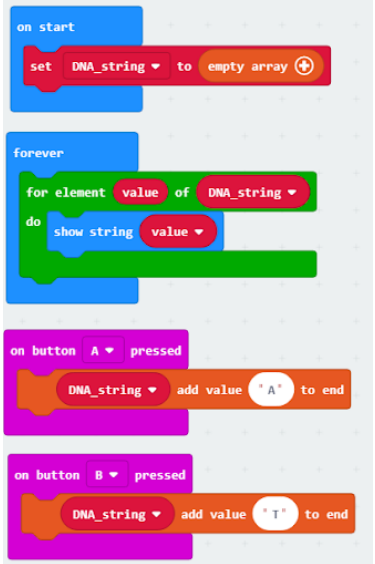
You can check if the behaviour of your program is correct by interacting with the micro:bit simulator at the left of the computer screen.

At this point you should upload this program to the actual micro:bit, and see if it works.

To upload the program, plug in the USB cable to the micro:bit, plug the other end into the computer, download the program using the "Download" button at the bottom left of the computer screen, and copy the file you get into the micro:bit USB drive. You can also go to the settings at the top right of the screen and synchronise your micro:bit with the browser to download effortlessly.

The micro:bit can run using the voltage in the USB cable, or it can run on batteries. You can also reset the micro:bit using the button on its back, for example if you want to start over and reset the DNA string.

The whole programme should look like this:



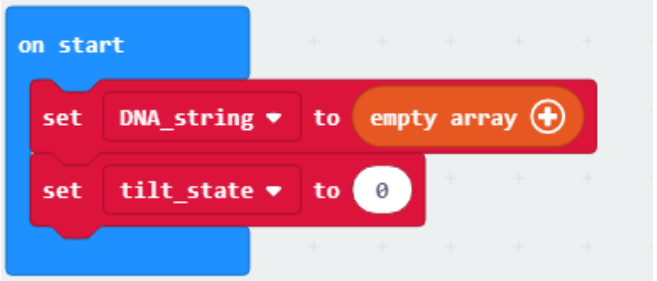
Lets make a DNA string!

Part 2

The first program we made has only A and T in the DNA alphabet. Can we also add G and C ?

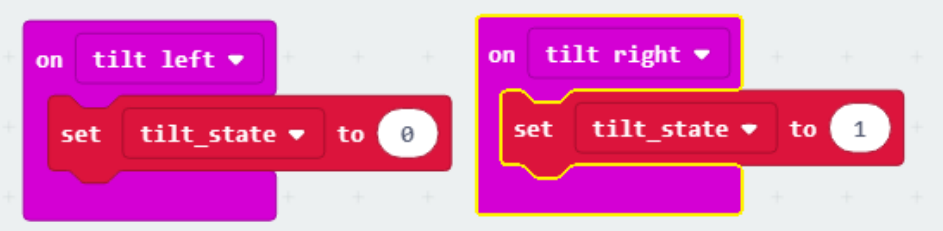
Well, the micro:bit has only two buttons, but it also has the ability to sense if we are tilting it left or right, and if we are shaking it, and many other movement interactions. You can find these blocks under the "Input" tab.

Let's change the code so that when we tilt the micro:bit left it will then write A and T and when we tilt it right it will then write C and G. You should note at this point that you can do this in other ways, and we welcome you to experiment with what method you like best.



With our method you will learn a bit about how to code so that your robot can be at different states and behave differently depending on which state it is in. Let's make a variable called "tilt_state", set it to 0, and put the block in the "on start" block:

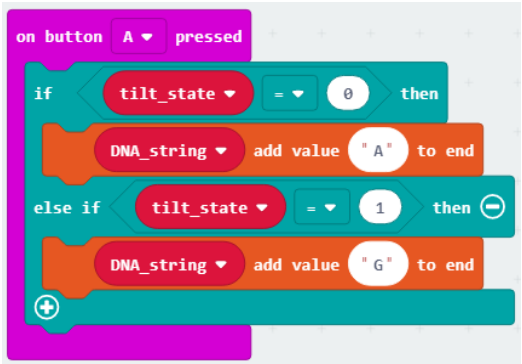
Now lets change what value this tilt_state is, depending on if we tilted the micro:bit left or right:



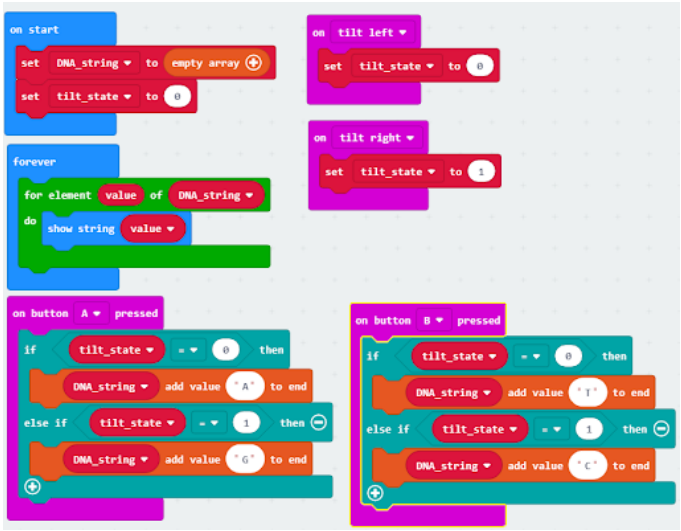
Good. Now lets change what the buttons A and B add to the DNA_string based on the tilt state.

If the micro:bit is tilted left and the tilt_state is 0, button A should write “A” to the DNA string, but if the micro:bit has been tilted to the right, and tilt_state is 1, it should write “G”. Likewise for button B. To do that, we will use one of the coolest things you can do with computers: the “if” statement.

Under the “logic” tab, you will find the “If (true) then .. else ..” block, along with comparison blocks. Let’s use them to make button A decide what to do depending on the tilt_state:

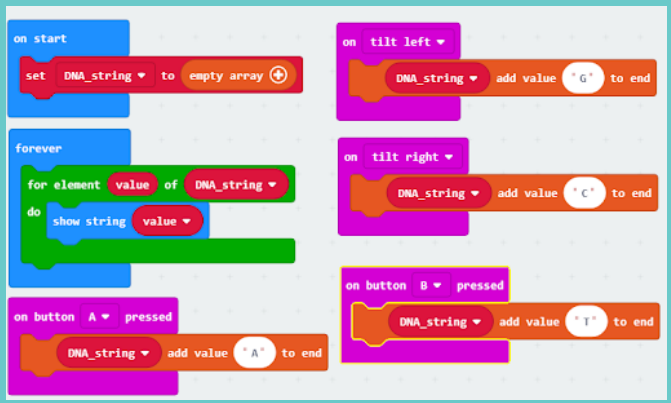


Do the same for button B and now you can write DNA using 4 letters!



Again, we need to say here that there are other ways of coding this. For example you could use 4 different input boxes such as A, B, tilt left and tilt right to input letters directly.

Can you spot how it is done in the code?

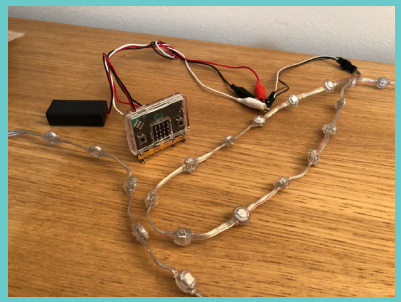


Lets make a DNA string!

Part 3

Now it's time to add some LEDs! Let's modify the simplest program so that it lights up a LED strip to represent our DNA!

First, connect the LED strip to the micro:bit. You need to see which way your LED strip works. As the micro:bit sends signals down the strip, each LED will take its command signal, and send the rest of the command signals down the line. So, the LED strip has an arrow to show which way this flow of information happens, and we need to connect the upstream part to the micro:bit. Also, each LED has an "IN" and "OUT" cable connection, so you can use that as well to figure out which side to connect to the micro:bit, which is the "IN" side.



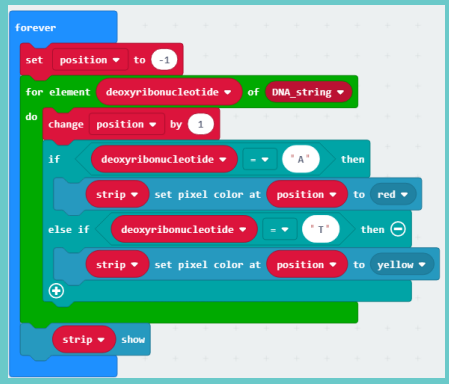
Connect the red cable, which gets the + voltage, to the +/3V/3 volts pin of the micro:bit. Connect the black cable, or the cable that is - voltage, to the GRD, -/ ground pin of the micro:bit. Connect the signal cable, that goes to the "IN" connection point to the LEDs, to the P0/pin 0 of the micro:bit.

Next, lets teach the micro:bit how to interact with these LEDs. To do so we have to add the "Neopixel" library, which you can find under the "Extensions" tab.

This will extend the functions (blocks) we can use. Normally when you add a peripheral to your robots, the fastest way to make them work is by using what code other people have made for them, which comes in libraries. This is a library that drives LEDs!

Under the new "Neopixel" tab, you can find the "set (strip) to ...." block. This sets up a strip of LEDs, and informs the micro:bit where it is connected, how many there are, and what type of colour values they recognise. Use the defaults, but change the number of LEDs to how many you have connected, and make sure the P0 pin is selected and connected to the LEDs.

Next, we need to instruct each LED to show our DNA_string. We will light up LEDs in a series with colours, one colour representing each type of nucleotide. So, instead of displaying the letters of the DNA_string in the "forever" block, now we will light up LEDs. We will use the neopixel block that sets a specific pixel to a specific colour. Our "forever" block should look like this:



At this point you should be good at making new variables and finding the blocks needed, so just copy the above after understanding what this code does. This code uses a "position" variable to count at which position on the LED string we are (0, 1, 2, 3 ...). Then, it has the green block that represents a loop. This loop, for each "deoxyribonucleotide" in our DNA_string, checks what it is, and sets the pixel colour of the LED at this position to the correct colour. It also keeps track of which position we are at, by incrementing the "position" variable.

We made it!

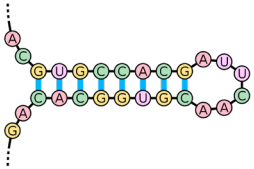
Now our micro:bit shows in the LEDs what sequence we have entered!

The code should look like this:

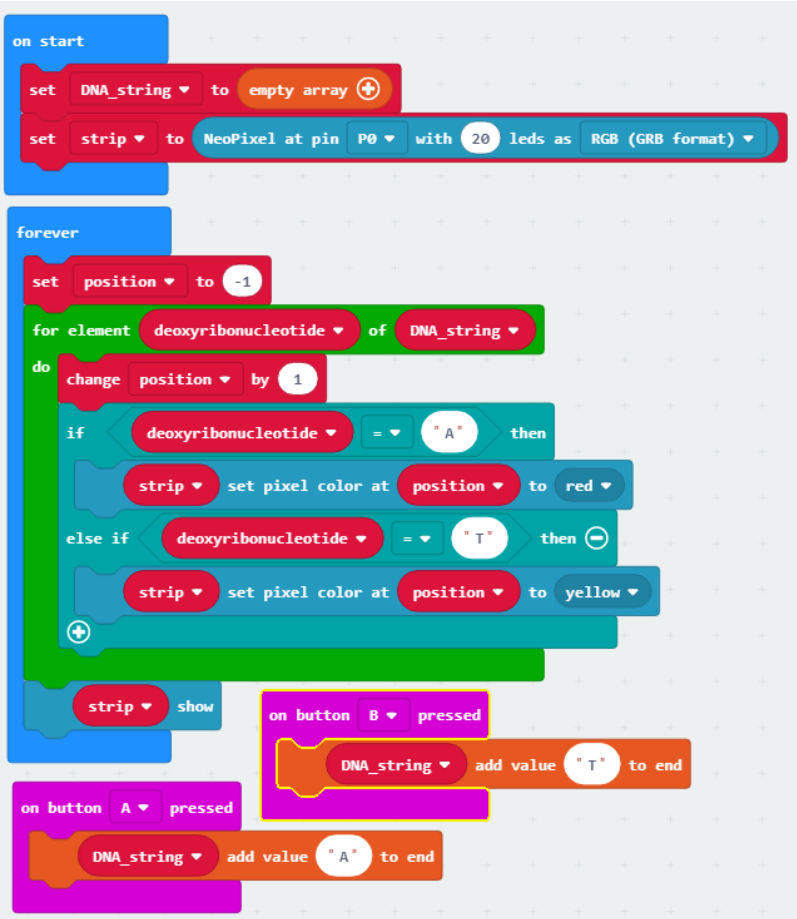
Now comes the fun part: In nature, nucleotides like to make pairs. A likes to go with T, and C with G.

Using the code, if you write a sequence of A's and T's, this sequence, in nature, is going to try to fold, so that each A is in contact with a T, while at the same time staying in a string.

Can you fold your sequence so that each yellow LED has a red next to it? Try not to squash it too much, the LED string, as well as the natural DNA strings, do not like to be bent beyond a point. Can you write a sequence that, if you fold it, will make a circle with some overlap? Can you make a hairpin loop?



Example of a hairpin loop



Extension Activities

Now you have learnt how to code the micro:bit to represent a DNA string. We now challenge you to get into groups and have a go at some of the following extension activities.

Universal code of life.
Consider which living organisms use this code and why?

4 nucleotides
Make a program that displays all 4 nucleotides on the LED string (combine programs 2 and 3).

BLAST
Investigate what a specific sequence of DNA or amino acids might be doing in nature using BLAST.

Electromagnetics
If you used a string of electromagnets instead of LEDs, how could you prgram the micro:bit to regulate the volatege in each magnet to cause some to attract to eachother like nucleotides.

Folding sequences.
Consider how and why, amino acid sequences fold.

START/STOP Codon.
Explore the effect of changing the position of these codons.

Check out BLAST: nlm.ncbi.blast.nih.gov

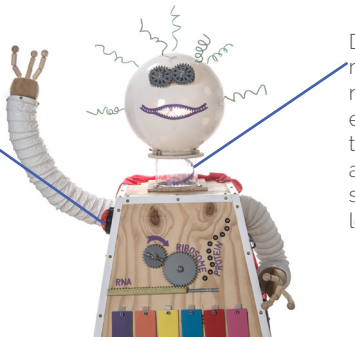
DNA Dave Electronics

Once you have worked out the messages you want your robot to show and how the LEDs are coded to reflect this, you can start to think about how the lights will be operated and where they will sit. You could have buttons to start sequences of lights or levers or cogs. You might want to try and add audio as well.

Lets recap on how the original DNA Dave was built to provide a way for people to interact with Dave and start the different processes and then lets take a look behind the scenes;

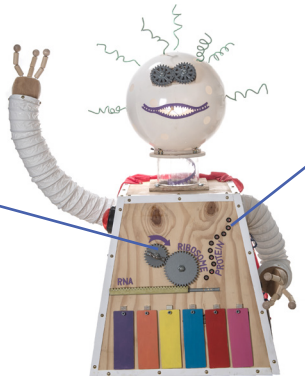
1.
The first step is to press and hold down the button on Dave's side.

This makes Dave's eyes spin around and symbolises the DNA code being "read" and transcribed into RNA



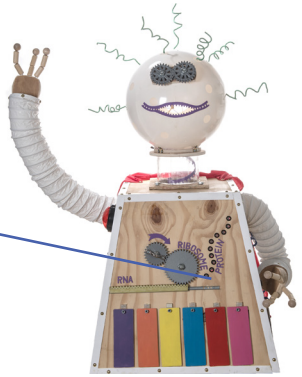
Dave's head represents the cell nucleus and when his eyes spin around, a tube inside his neck also spins, showing a single strand of mRNA leaving the nucleus.

2.
Next turn the small cog as indicated on Dave's body to drag the RNA strand through the ribosome.



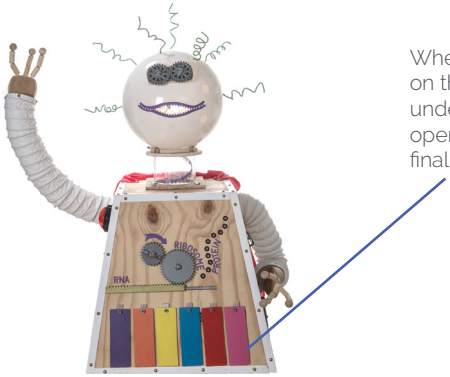
When the RNA strand reaches the end, the protein lights switch on to indicate translation of the RNA sequence and the gathering of amino acids needed to build a protein.

3.
If the RNA fails to reach the end, an alarm sounds to signify the presence of a mutation in the sequence



The protein lights go out and you must start again by turning the cog to feed the RNA strand all the way through the ribosome

4.
The lights above the 6 doors flash in a random order to signify the amino acids are being folded into a protein,

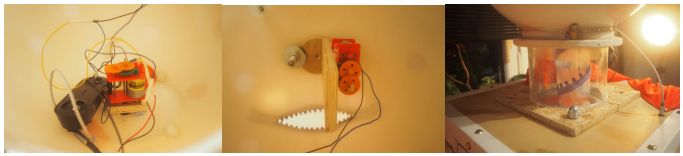


When one light stays on the door directly underneath should be opened to reveal the final protein.

DNA Dave's head - the nucleus

Inside Dave's head are two simple motors that control the spinning of his eye cogs and of his necks inner cylinder. The motors are operated by a button on the side of Dave's body.

This system doesn't require any coding as its just a simple on/off control for the motors to spin which is operated by holding the button in.



Daves eyes (cogs) and the inner tube of his neck are connected to motors stored within his head

DNA Dave's ribosome

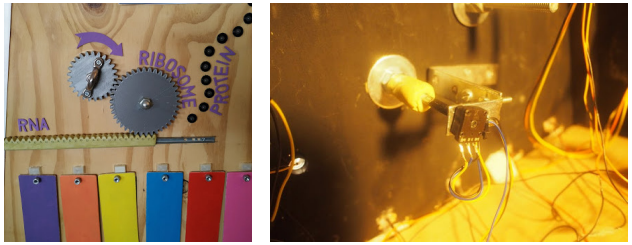
The next moving part is the 'ribosome' represented by cogs that move a mouldable glue bar with cog teeth.

The person is required to physically turn the cog until the wooden bar travels along to reach its farthest point. The position of the bar is read by a potentiometer that is connected to one of the cogs. When the bar is at the very left, the microbit starts to light up the LED sequence of 'amino acid' lights. As long as you turn the cogs of the ribosome so that the bar representing RNA goes to the right, a new "amino acid sequence" is made, and all the LEDs light up sequentially .

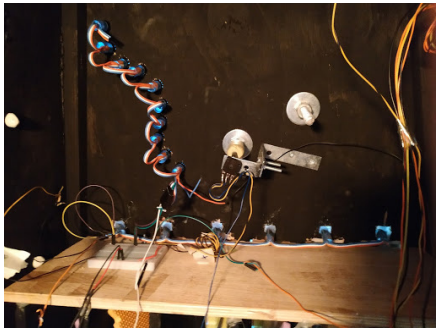
If the wooden bar doesn't reach the furthest point the code is started that tells the LED strip to only light up part of the way and then sound an alarm to indicate a failed translation of DNA to amino acid.

When all the amino acid lights are illuminated the code then instructs the LEDs above each of the protein flaps to light up in a random sequence before stopping with one LED illuminated.

This indicates the protein flap the person should open to reveal the protein they have created. In this case, the proteins produced are shown on large jigsaw pieces that the person interacting with DNA Dave can remove from inside the robot via the flap.



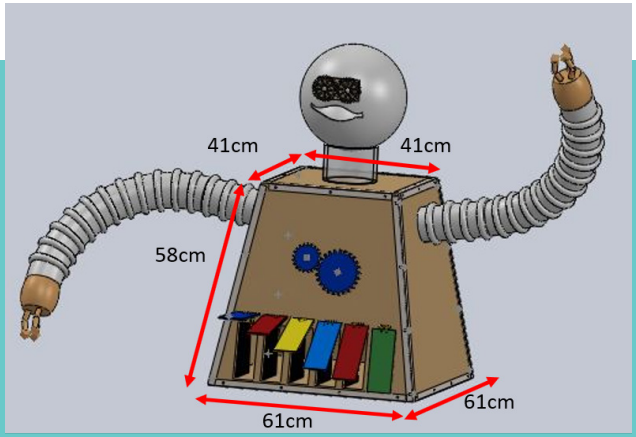
Front and back of the cog which feeds the RNA through the ribosome that has a potentiometer attached to it, to read the angle the cog is in.



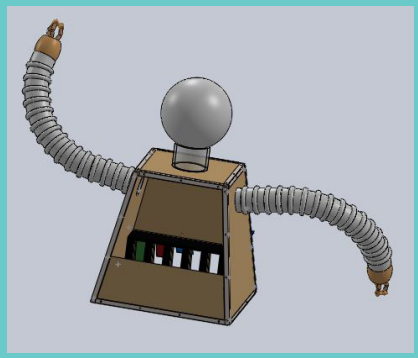
Inside Dave you can see the LEDs in a line just above the wooden platform that flash when the protein is being created.

Building DNA Dave

The large DNA Dave robot is constructed out of plywood with the following shapes and specifications; A square base, top and four trapezoid sides. See CAD drawings below.



His arms are made from PVC flexible ducting hose and his head out of a buoy. The cogs were 3D printed and the RNA strip turned by the cogs on his front is mounted on a runner from a set of drawers.
We encourage you to recycle/re-purpose materials that are available to you.

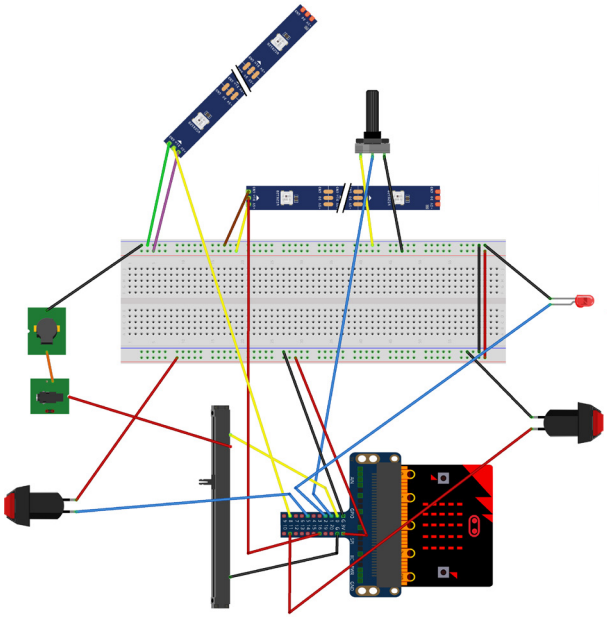


The back panel has a large section cut out to enable storage of his detachable head and arms inside.

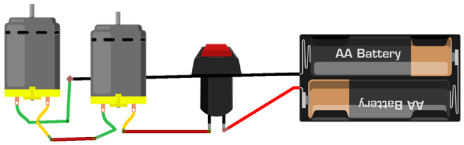
To download the full CAD drawings and more, go to: https://bit.ly/2RAd4yq

Planning and Schematics

DNA Dave micro:bit diagram



Simple motor set up for DNA Dave's eyes and neck.



Now that you understand why and how DNA Dave was built, its your turn to produce your own educational robot!

We suggest that you start by establishing a diverse project team and decide upon a project timeline.

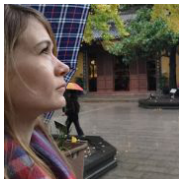
This booklet is available for reference but we encourage you to think big and create your own unique robot.

For more information and resources got to the DNA Dave hackster page: https://bit.ly/33fClf6

Notes

Acknowledgements

With thanks to additional team members who helped collate and assemble materials for the Build Your Own DNA Dave project.



Sami Stebbings

OpenPlant Administrator

Having obtained a BScHons in Ecology from the University of East Anglia, Sami now works for OpenPlant, a synthetic biology research centre and specialises in science communication.

Tom Mclean

SAW Trust intern

Whilst studying for his PhD in molecular microbiology at the University of East Anglia, Tom's interest in science communication led him to carry out a three-month internship with the SAW Trust.

Our Funders

A special thank you to the OpenPlant Biomakers Challenge and the Biochemical Society whose funding has enabled this project to go ahead.

OpenPlant
sharing tools for a sustainable future

BIOCHEMICAL SOCIETY

Reg. Charity no. 1113386
www.sawtrust.org