# Microbit projects: caterpillar

## The aim

We’re going to build a crawling robot that’s inspired by some types of caterpillar. See:

https://www.youtube.com/watch?v=fRVGWCSij\_M

or

https://www.youtube.com/watch?v=a9Km0edRFG4

In our case, we’ll only have 3 body segments but, as you can see from the video, a wave passes along the caterpillar so we’ll try to mimic that. For this project, we’re going to need some power, provided by a particular type of motor called a servo motor, and we’re going to need to control that using a program written on a MicroBit.

## A sketch

# Skills needed

* Servo motor (PWM)
* Programming

## Let’s begin: Servo Motors

First we’re going to take a look at the servo motor. There are two types of servo – the first is one where you can choose an angle to which the arms should rotate; the second goes round and round at a speed you set. We’re going to use the first type here and the second type in the next project.

Servos are used all over the place in both daily life and in robotics. A servo might retract the tray of a DVD player or might be used in radio controlled vehicles and aeroplanes to control the rudder of a boat, or the ailerons on a plane. They are a motor we can set to a certain angle – in our case ranging from 0° to 180°.

### PWM

The angle of the motors is set using what is called a pulse width modulation signal (PWM).

**0.5ms**

**0**

**1.5ms**

**90**

**2.5ms**

**180**

**20ms**

**0ms**

**Time /ms**

### Connecting a motor to the microbit

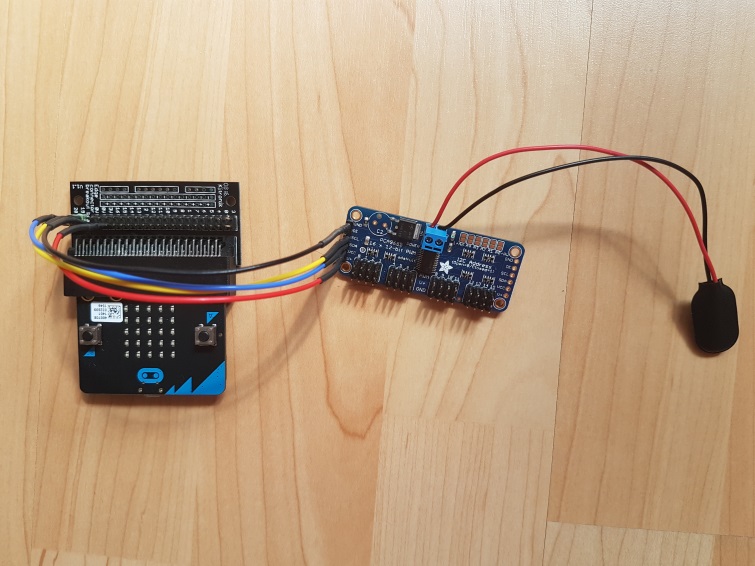
Although the MicroBit can drive upto 3 servo motors, we have decided to use a second board to connect the MicroBit to the servos, for two reasons:

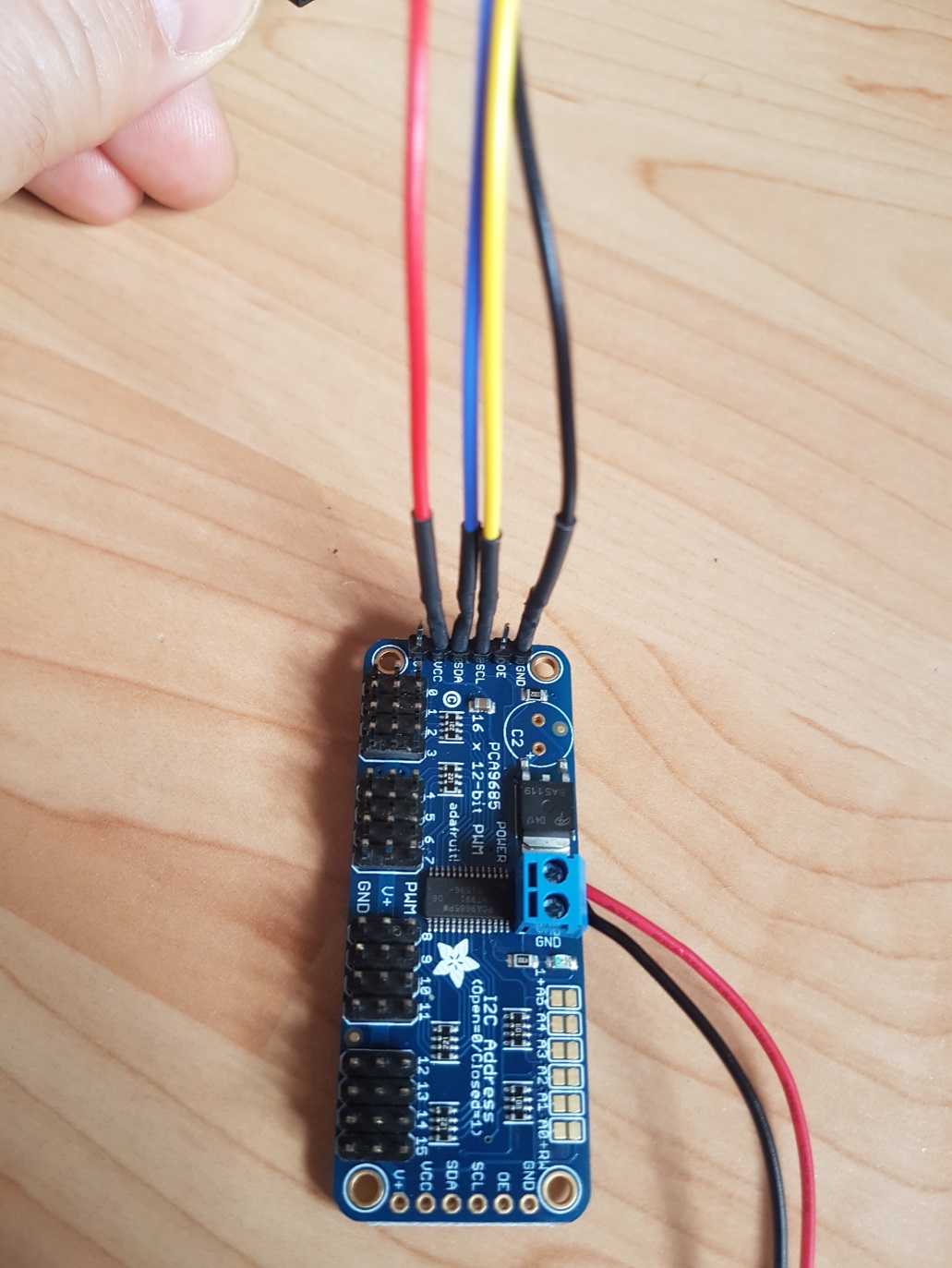
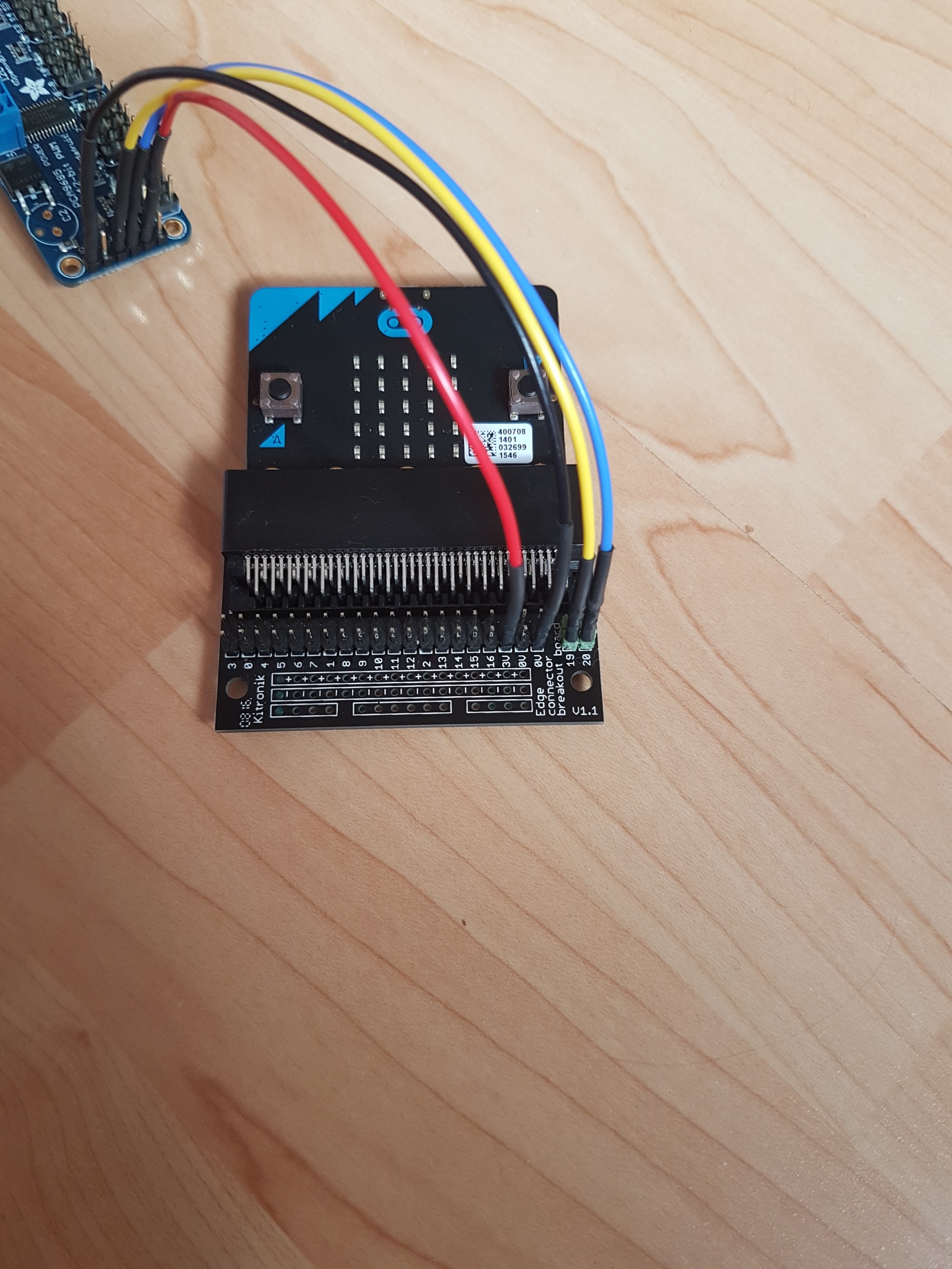
* The MicroBit works on a 3.3V circuit, and the servos work on ~5V. Consequently, we can’t just connect the two together anyway.
* We can connect up to 16 servos to the MicroBit using the interface board – we *could* make a very long snake ☺

The PWM board communicates with the MicroBit using a special digital communications channel – I2C. We have written some code to hide these details from you.

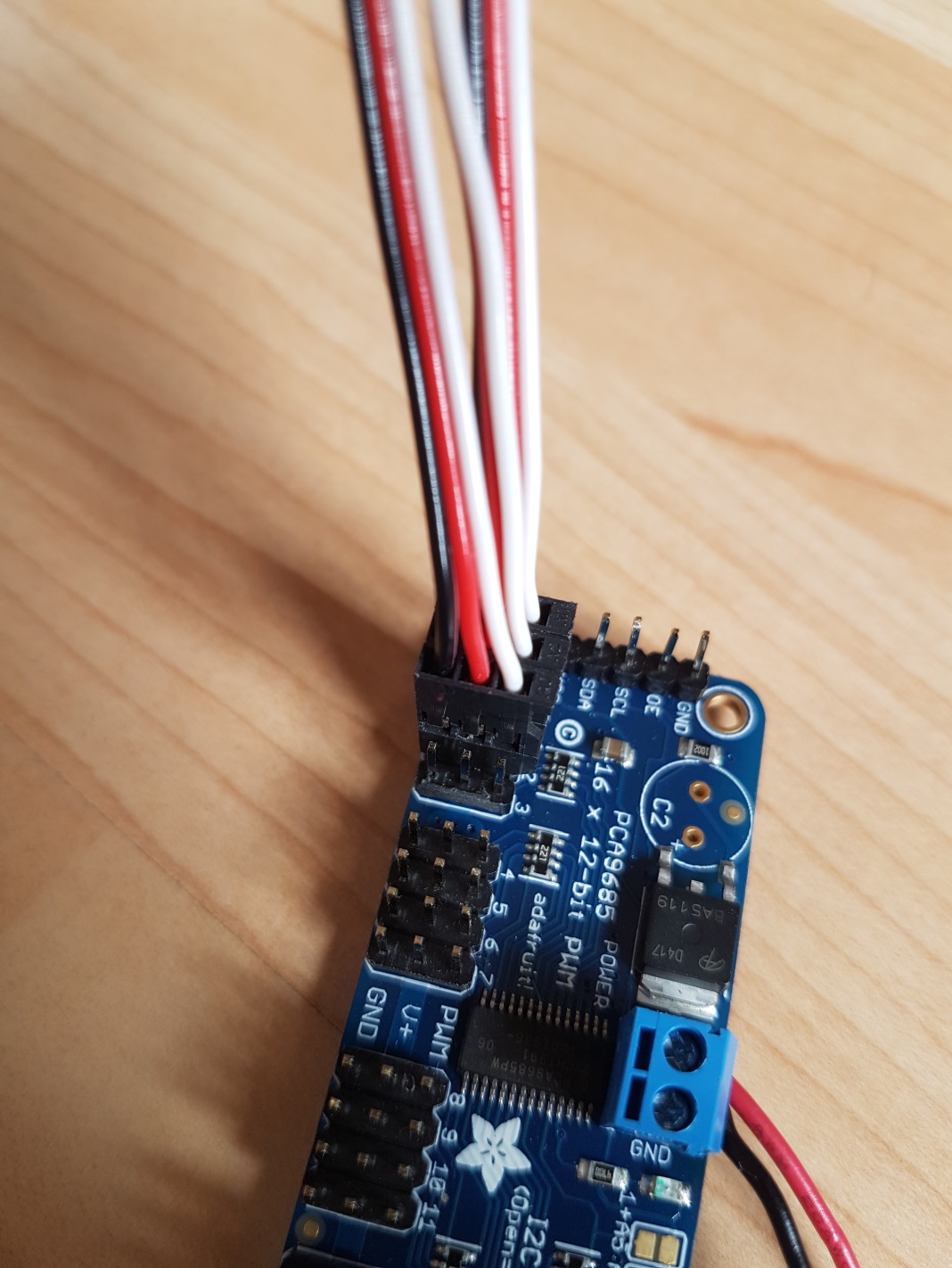
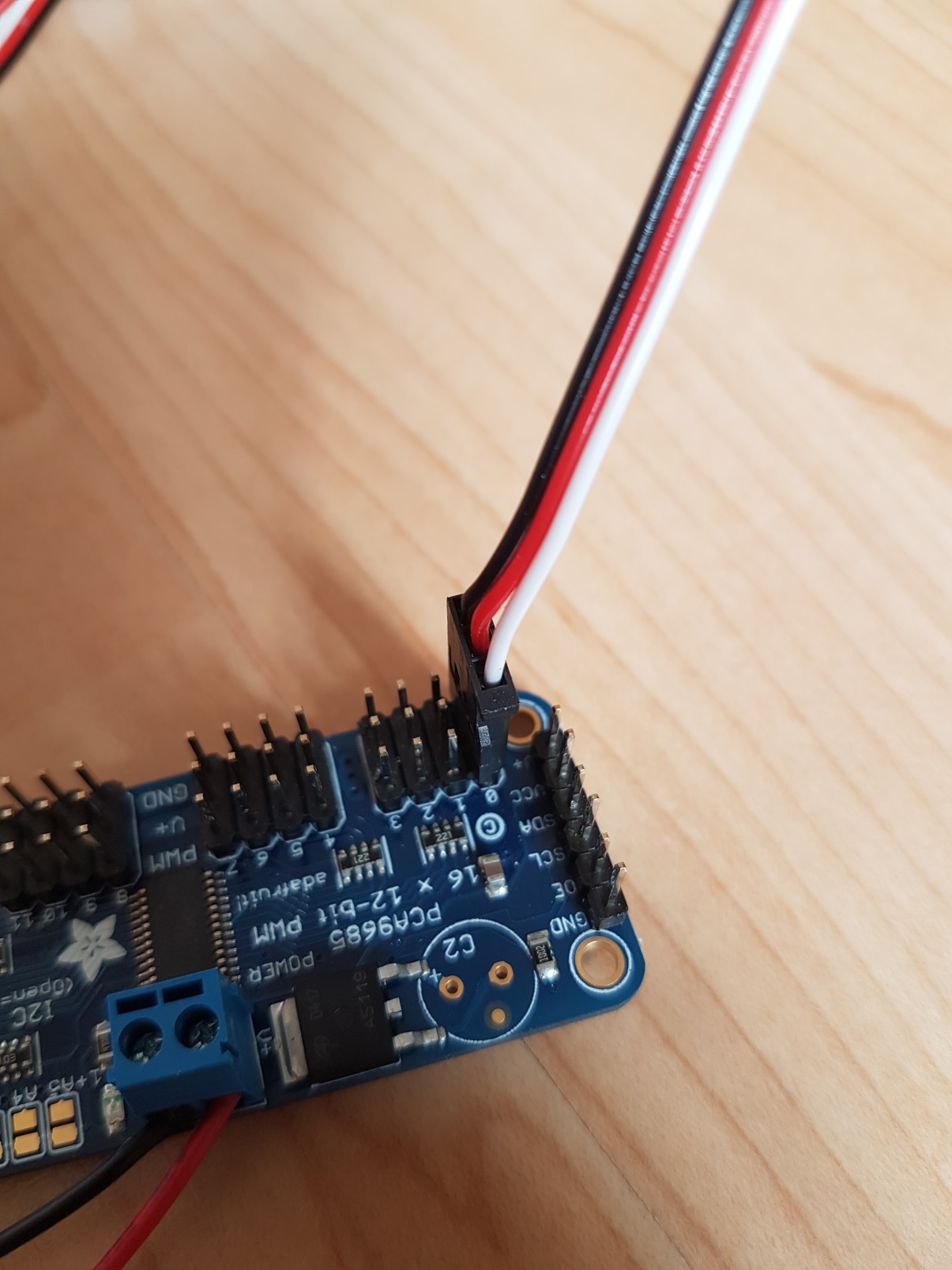
**DO NOT CONNECT THE BATTERY UNTIL YOUR CIRCUIT HAS BEEN CHECKED**

The PWM board should be connected to the MicroBit like this:





And the servo(s) should be connected to the pwm board this way round – on the left is a servo connected in position 0. (It’s easier if you start at 0 and work up if you have more servos connected).



Writing code to drive the motors

As I said above, we have written some code to make life a bit easier for you. You can set an angle on a servo like this:

Note the & here

Just idle waiting for an event

When the button is pressed move the servo by 10 degrees

Listen for any button event – call onButton if there is one

1. **Create a new project for the MicroBit, called ‘test-servo' and copy the files PCA9685.h PCA9685.cpp into that directory.**
2. **Type in the following code and save it as test-servo.cpp in the source directory**

#include "MicroBit.h"

#include "PCA9685.h"

MicroBit uBit;

PCA9685 pwm(&uBit, angleServo);

int angle = 0;

void onButton(MicroBitEvent)

{

// Set the angle using setAngle

//

angle += 10;

if (angle > 180) angle = 0;

pwm.setAngle(0, angle);

uBit.display.scroll("Angle");

uBit.display.scroll(angle);

}

int main()

{

uBit.init();

uBit.messageBus.listen(MICROBIT\_ID\_BUTTON\_A, MICROBIT\_BUTTON\_EVT\_CLICK, onButton);

// Initialise the angle to zero and turn the pwm on

pwm.setAngle(0, angle);

pwm.start();

while (1)

uBit.sleep(1000);

}

1. **Now compile the code and flash it to a MicroBit**
2. **Connect a single servo to position 0 on the PWM board, and the PWM board to the MicroBit – *ask someone to check it for you before connecting the battery*.**

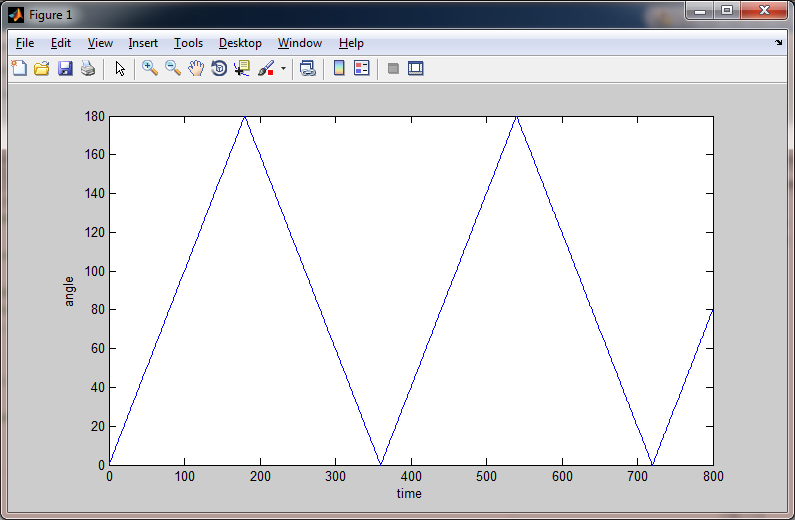
extending this

We need to be able to make segments of the snake move forward by lifting and stretching. Conveniently, this turns out to be done just by changing the angles between segments in the caterpillar – but the servo moves way too fast on its own and we need to be able to control it better.

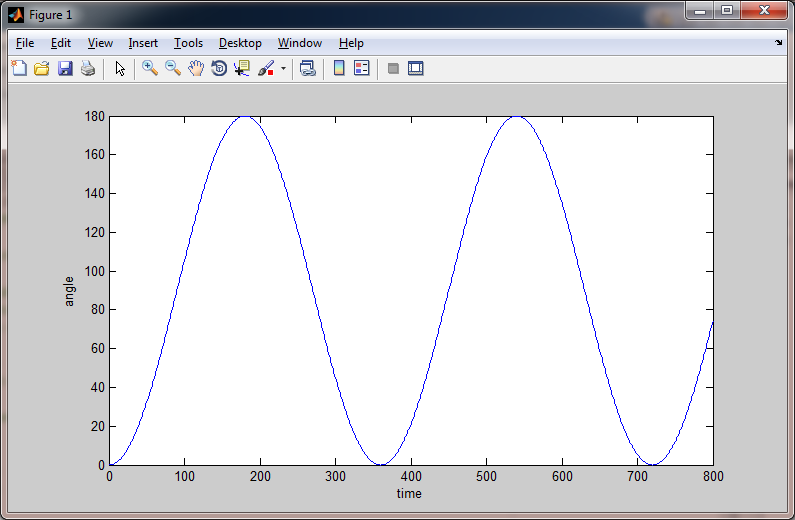
1. **Create a new project for the MicroBit, called ‘test-servo2' and copy the same files over.**
2. **You should write a program to make the servo move *slowly and smoothly* from one extreme to the other – i.e. from 0° to 180° and back**
3. **To do this, you could:**
   1. **break the change angle down into small increments – try a few degrees at a time**
   2. **make a change and insert a delay before making the next**
4. **Now compile the code and flash it to the MicroBit**

What do you see?

If we look at this on a graph, we see that the changes happen linearly



As it happens, many things in biology happen like sine waves. That’s true for this kind of movement – what we’d like to see is the following



What this means is that the change in angle is slower the closer you get to 180 and to 0. So, it starts slowly, then gets faster, then slows down, changes direction, and so on.[[1]](#endnote-1)

1. **If we want make a change in angle at time t we need to know what the time is:**

long t = uBit.systemTime(); // Gives the time since startup in milliseconds

1. **See if you can figure out the angle you should set the servo to – this will be a function of t (remember this is in ms) and will involve a sin function. Useful code:**

y = sin(x); // calculates the sine of x, where x is in radians

1. **To make life easier, you might like to use the following function:**

double degrees2radians(double angle) {

return (angle\*PI/180);

}

...

y = sin(degrees2radians(d)); // Now d is in degrees

1. **Answer…. see endnote i**

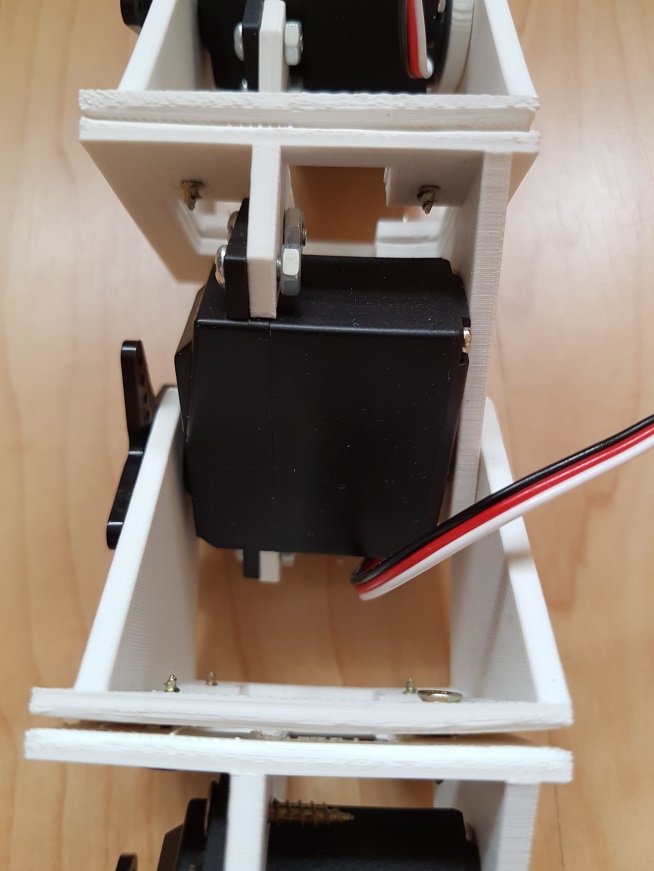
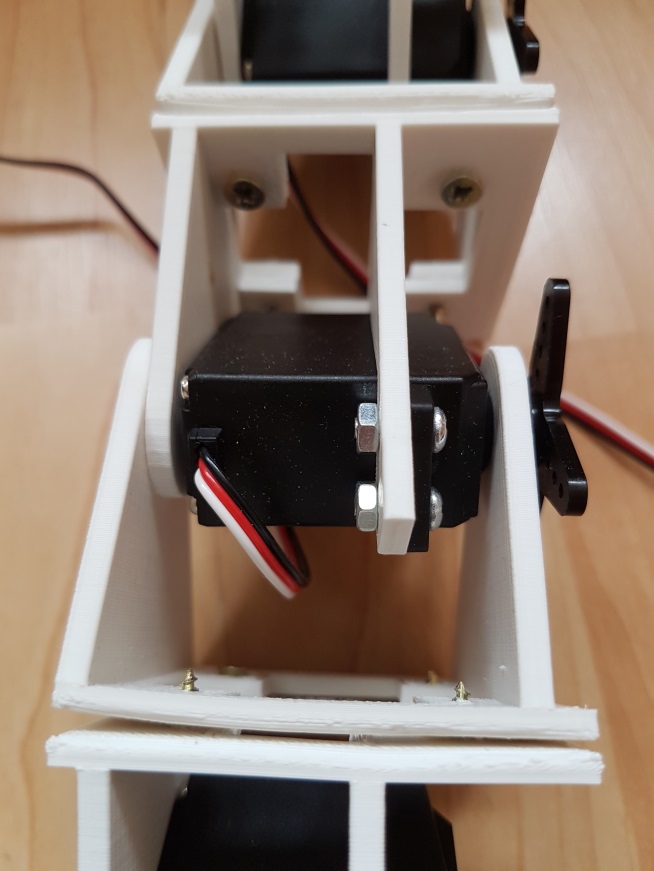
## Oscillator

Congratulations – you made an oscillator.

## Building the snake

Make a snake – it should look like this. There’s one at the front of the room if you need to look





## Making the caterpillar move

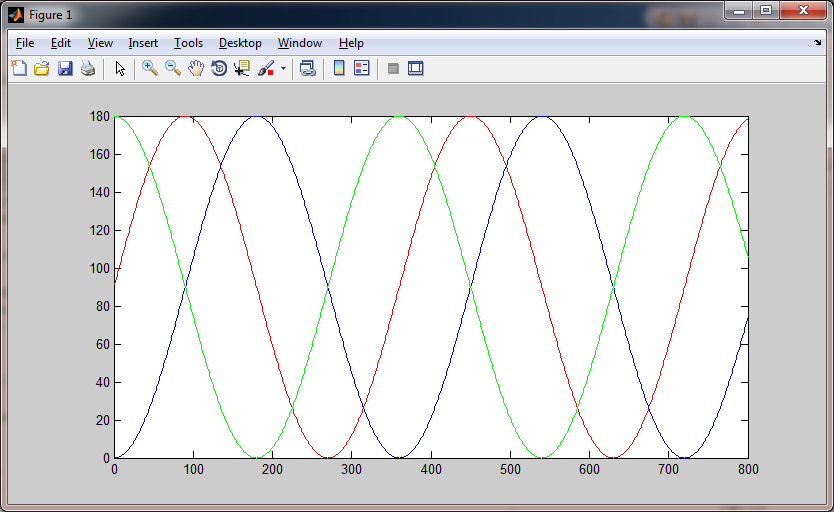
1. **Create a new project for the MicroBit, called ‘caterpillar'**
2. **Make all the servos in the caterpillar move in sync.**
3. **Does the caterpillar crawl?**
4. **Take another look at the videos. See if you can figure out what might make the caterpillar work.**

## central pattern generators

*Central pattern generators (CPGs) are biological neural networks that produce rhythmic patterned outputs without sensory feedback. CPGs have been shown to produce rhythmic outputs resembling normal "rhythmic motor pattern production" even in isolation from motor and sensory feedback from limbs and other muscle targets. To be classified as a rhythmic generator, a CPG requires: 1. "two or more processes that interact such that each process sequentially increases and decreases, and 2. that, as a result of this interaction, the system repeatedly returns to its starting condition. CPGs have been found in practically all vertebrate species investigated, including human. Wikipedia*

We’re going to mimic a CPG – the control of movement in the caterpillar will be done with very limited sensory input. The electronics requires us to know the angle of each servo but the same approach would work if all we had was an oscillator (or an oscillator per segment).

## Phase



Here we have three sine waves out of phase with one another. The blue and the green lines are 180° out of phase – completely out of phase: when one is at 180°, the other is at 0° and vice versa. The red line is somewhere in between.

1. **If we have one angle calculated from time by:**

angle\_0 = C + A\*sin(degrees2radians(360\*t/T));

**We can have a second calculated by:**

angle\_1 = C + A\*sin(degrees2radians(360\*t/T + P));

**where P is an angle between 0° and 360°, depending on what phase difference we want.**

1. **Can you use this to make the caterpillar move more effectively?**

## Competition

1. **How fast can you go? Things to think about:**
   1. **Should each segment move through the same amplitude?**
   2. **Should we use the same centre position for each segment?**
   3. **Should the movement of each segment be symmetric?**
   4. **How fast should the oscillation be?**
   5. **Can you customise the caterpillar to change the friction between it and the ground?**
2. **Can you make the robot move on its side?**

# Endnotes

1. angle = 90 + 90\*sin(degrees2radians(360\*t/4000));

   t is the system time.

   The 4000ms is the amount of time is takes to go from angle zero to zero. Make it bigger to go slower.

   sin varies between 0 and 1 so the angle varies between 0° and 180° as things stand. The 90\*… is the amplitude – here we want it to go through all 180° of motion. Make this smaller if you want less – if the value was, say 40, then the movement would vary between 60° and 130°.

   The 90+… is the centre position of the movement. The servo moves equally either side of that centre. You could choose to centre the movement in a different place. If this was, say, 100 and the amplitude was 40 then the servo would move between 60° and 140°.

   You should perhaps create variables for amplitude and time period, say A and T, with another for the centre, C. This makes the formula: angle = C + A\*sin(degrees2radians(360\*t/T)); [↑](#endnote-ref-1)