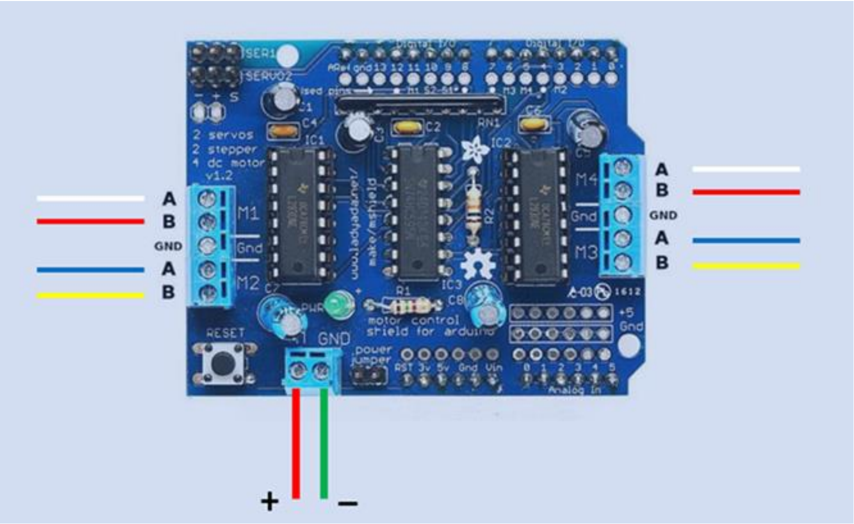
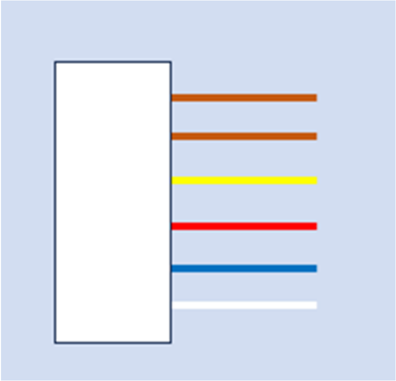
Albus Severus (son of Harry)

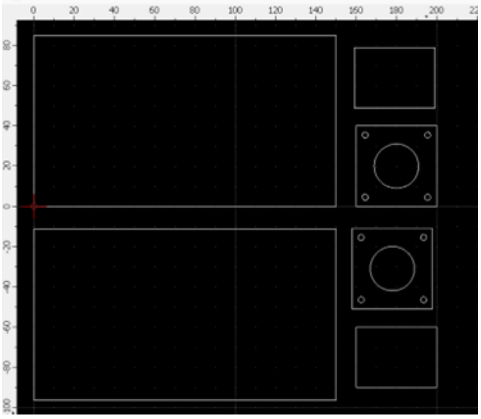
Wiring motors to motor shield





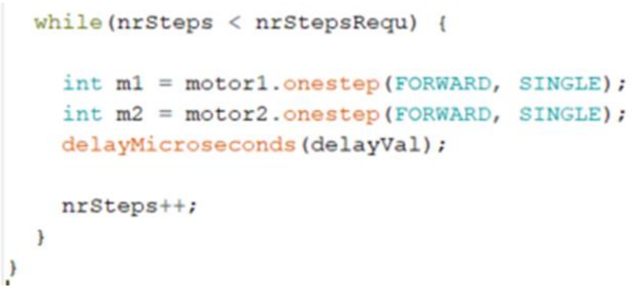
- Top left shows how to connect steppers to the motor shield.
- To right shows the stepper plug. Brown wires connected to red wires on harness supplied.

Design and Build the Chassis



- Template QCad file provided **AlbusSeverus.dxf**
- Has useful plates which fit the stepper motor
- Design rule to obtain exact 90° pivot: **50 * axleLength / wheelRadius** is a whole number.
- Need to have space for
    - Arduino Uno with motor shield on top
    - 6V ‘green’ battery
- Need to have front plate to allow addition of pixyCam, HuskyLens etc. Consult Harry.

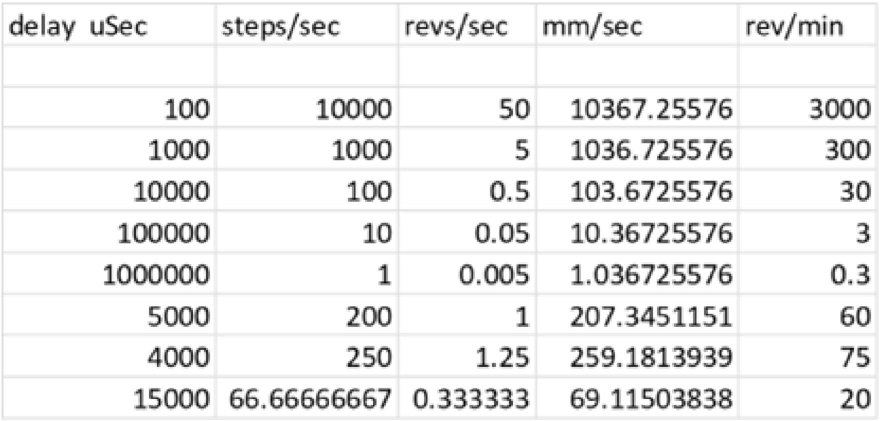
Driving the Motors



```
  while(nrSteps < nrStepsRequ) {

    int m1 = motor1.onestep(FORWARD, SINGLE);
    int m2 = motor2.onestep(FORWARD, SINGLE);
    delayMicroseconds(delayVal);

    nrSteps++;
  }
}
```

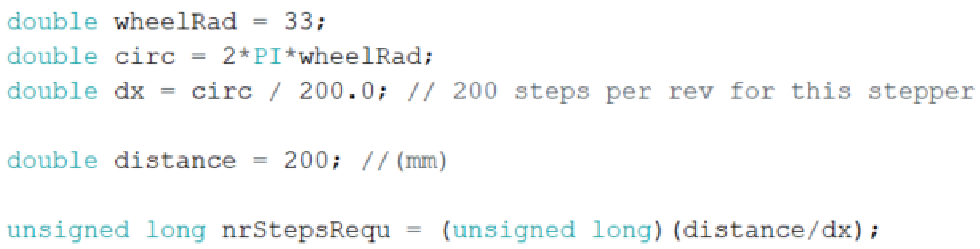
- While loop increments **nrSteps** which is the nr steps actually taken.
- Need to pre-compute **nrStepsRequ** (explained below) and initialize **nrSteps**.
- **delayMicroseconds(…)** determines rotational speed of motors.

Some Motor Speeds



delay uSec	steps/sec	revs/sec	mm/sec	rev/min
100	10000	50	10367.25576	3000
1000	1000	5	1036.725576	300
10000	100	0.5	103.6725576	30
100000	10	0.05	10.36725576	3
1000000	1	0.005	1.036725576	0.3
5000	200	1	207.3451151	60
4000	250	1.25	259.1813939	75
15000	66.66666667	0.333333	69.11503838	20

Movement in a straight line



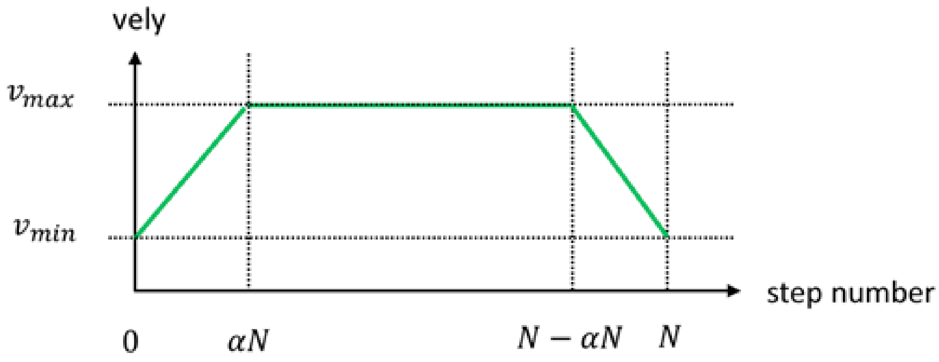
```
double wheelRad = 33;
double circ = 2*PI*wheelRad;
double dx = circ / 200.0; // 200 steps per rev for this stepper

double distance = 200; //(mm)

unsigned long nrStepsRequ = (unsigned long)(distance/dx);
```

- First need to find **dx** the distance moved for each step (ask for the theory if you like)
- Then decide on your desired **distance.**
- Calculate steps required **nrStepsRequ** and use unsigned long integers.

DO IT : Get Albus Severus to move in a straight line.

- Use the sketch CBP_FBO_Albus_Template.ino
- Before the while loop you must:
  - Find the number of steps required
  - Set the **delayVal** for a sensible speed.
- Investigate the largest speed the motors can handle. Ask for the theory how to convert speeds in rmp to delayVal.

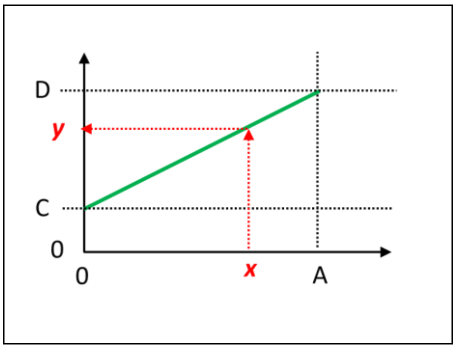
DO IT: Write a function to ramp the speed.



- Velocity changes with step number over the total steps required, N.
- Rises for fraction alpha of N and falls over same number of steps.
- Code will look like this. Need to finish the ifs and code vely =
- Function outputs **usDelay** (microseconds delay) which we need to set the speed.

```
unsigned long ramp(int vMin, int vMax, unsigned long N, float alpha,
               unsigned long n, bool upRamp, bool downRamp) {

  float vely;
  unsigned long uSDelay;

  if((n < alpha*N) && (upRamp == true)) {
    vely = …
  } else if ((n < alpha*N) && (upRamp == false)) {
    vely = …
  }

  if((n >= alpha*N) && (n < (N - alpha*N))) {

  }

  .. and more ifs

  uSDelay = (unsigned long)(60*5000.0/vely);

  return uSDelay;

}
```

- Diagram below will help with coding the vely = expressions.
- Red arrows are what you put in and get out
- By simple trig we have **y = C + x * (D – C) / A**



Code for Pivoting

```
void pivot(float degs, int vMin, int vMax, float dx,
        float axleLength, bool upRamp, bool downRamp) {

  float theta = (PI * degs)/180.0;

  int dirL;
  int dirR;

  unsigned long nL,nR;
  unsigned long vL,vR;
  float sC;

  unsigned long delayVal;

  if(theta <= 0) {
    dirL = FORWARD;
    dirR = BACKWARD;
  }
  else {
    dirL = BACKWARD;
    dirR = FORWARD;
  }

  sC = axleLength/2 * theta;
  nL = sC/dx;
  nR = nL;

  int i = 0;
  while(i <= nL) {
    delayVal = ramp(vMin, vMax, nL, 0.25, i, upRamp,
downRamp);

    int m1 = motor1.onestep(dirL, SINGLE);
    int m2 = motor2.onestep(dirR, SINGLE);
    delayMicroseconds(delayVal);

    i++;
  }
}
```

- Above code should work (has not been *extensively* tested)
- Based on the code in the library **CBPFBO_StepperA** in portable > sketchbook > libraries
- Perhaps see how it has been changed.

Code for Arcing

- Go to the library **CBPFBO_StepperA** in portable > sketchbook > libraries.
- Find the function for the Arc.
- Copy into a sketch and modify it based on your understanding of how the pivot(…) code was adapted.