**GROUP NUMBER: 11**

ROLL NUMBER 1: 200260008 NAME 1: Anurag Abhijit Pendse  
ROLL NUMBER 2: 20B030011 NAME 2: Aswin Suresh

**TITLE:**

Modelling a 3-body active system using Arduino

**ABSTRACT:**In this project, we model a 3-body actively driven system using Arduino. The Arduino produces the noise inputs which we need to supply the system as well as carry out the required computations as needed to solve the coupled system of differential equations. The output is seen on a DSO as well as in a real time plot of all 3 particles on Microsoft Excel.

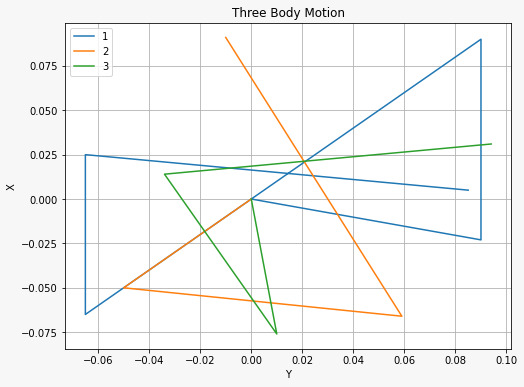
**PROJECT DETAILS:**   
The project deals with solving the system of coupled differential equations which model a 3- body system of actively driven interacting particles with Brownian Noise ()

The values used for the parameters are as follows:

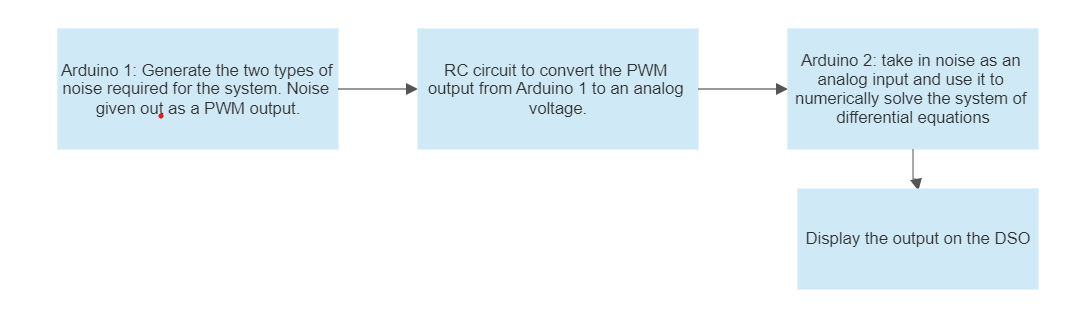
α is a combination of a simple harmonic attraction as well as a random self-drive to counter it.

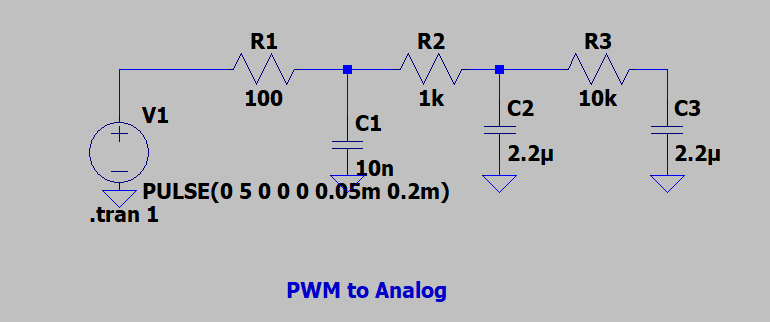
We set

While a common has been written in all 6 equations, the values of aren’t necessarily same for all 6 at a given time. This has been taken care of in the code where we are sampling the noise at different times for different coordinates.



To view all the particles simultaneously, we are using the PLX-DAQ library to write the coordinates to excel in real time to plot them.

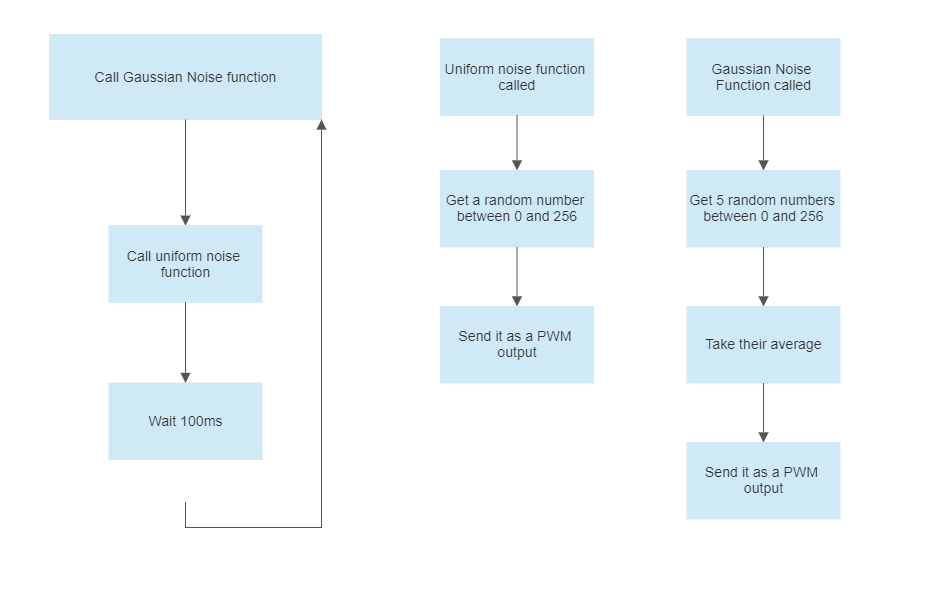
  
The RC circuit used is as follows:

  
The outputs for various duty cycles are as follows:

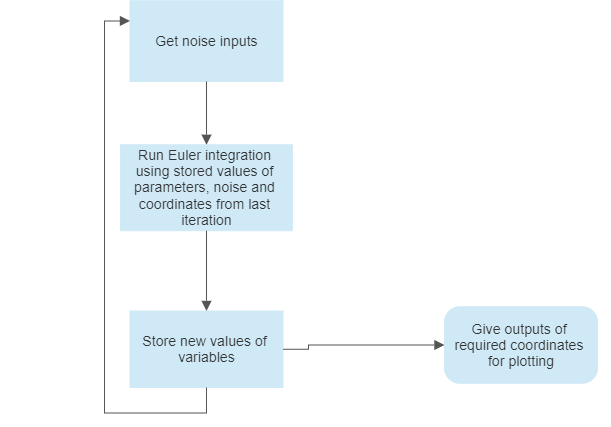




Arduino 1: (Done by Anurag)



Arduino 2: (Done by Aswin)



Other than this, Anurag built the required circuits for converting the PWM output to an analog voltage. Aswin wrote the code to print values to Excel in real time. Both of us worked on obtaining suitable values for the parameters which gave a stable system.

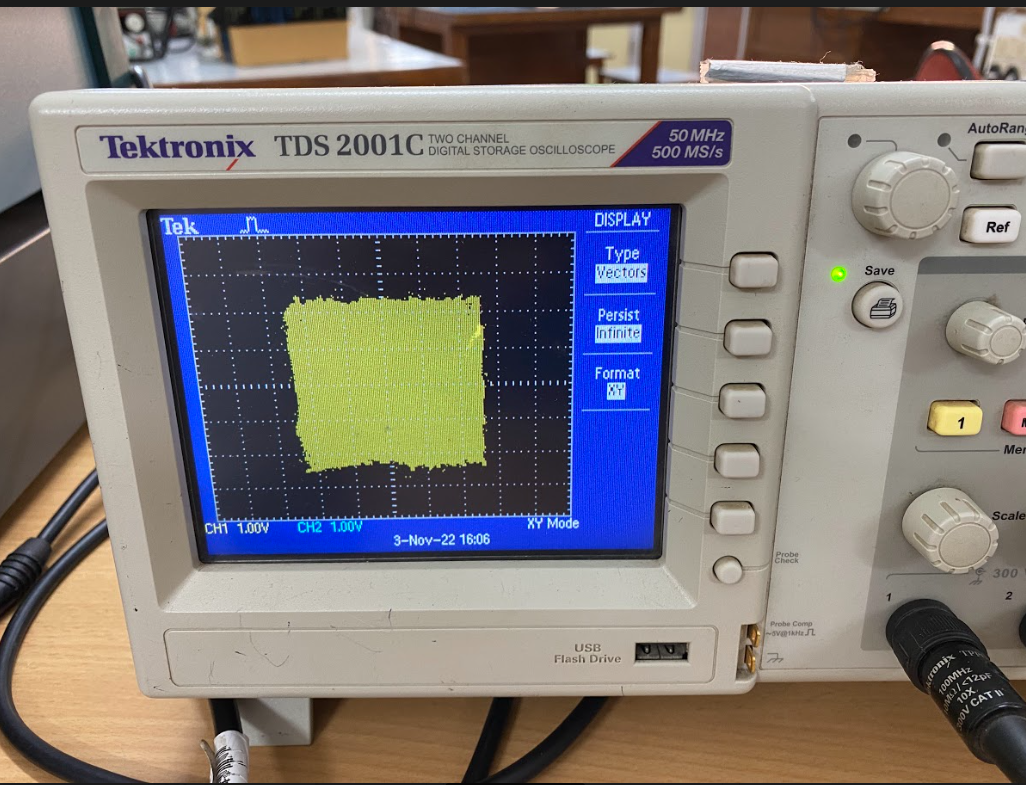
**MAIN COMPONENTS NEEDED TO BUILD THE PROJECT:**

2 Arduinos, a breadboard, 12 resistors, 12 capacitors and a DSO are required to build the project.

**RESULTS:**

Using the design as specified above, we can observe the path the particle follows. The path appears to tally with what a random walk should look like. Furthermore, we can see that the path followed by a particular particle when looked at as confined within the box bounded [-2.5,2.5]X[-2.5,2.5] with periodic boundary conditions appears to be space filling. This tallies with expectations of the nature of the path (DOI: 10.1103/PhysRevE.87.042136).

<https://drive.google.com/file/d/1VXDTxzpSW4ZKI-8aVeH_sBpfOTNgYA0h/view?usp=share_link>



Output on Excel: <https://drive.google.com/file/d/1xLORWKbZz_7Js30jniKJcTbHGEJfyRLa/view?usp=sharing>

Output on DSO:

<https://drive.google.com/file/d/1st0ec8QbnQ_1sgwmdLaT2NbmYkuRxsXa/view?usp=sharing>

<https://drive.google.com/file/d/1g8CH7KGgDnRfq8yMoJBHCS_-kOtnIkng/view?usp=sharing>

**APPENDIX:**

Arduino 1:

int x1;

int x2;

int x3;

int x4;

int x5;

int x;

int y=0;

int alph1;

int alph;

int eta;

void setup() {

Serial.begin(9600);

pinMode(5,OUTPUT);

pinMode(6,OUTPUT);

}

void loop() {

alph1=unigen();

alph=int(alph1\*y/1023);

eta=gaussiangen();

analogWrite(5,eta);

analogWrite(6,alph1);

Serial.println(eta);

delay(100);

}

int gaussiangen(){ [\\Gaussian](file:///\\Gaussian) Noise(eta)

x1=random(0,256);

x2=random(0,256);

x3=random(0,256);

x4=random(0,256);

x5=random(0,256);

x=int((x1+x2+x3+x4+x5)/5);

return x ;

}

int unigen(){ [\\Noise](file:///\\Noise) to be used added to alpha

int x1=random(0,256)

return x1;

}

Arduino 2:

#include<stdio.h>

#include<math.h>

int InputPin\_1 = A0;

int InputPin\_2 = A1;

const double gamma = 0.3;

const double step\_time = 1;

double x1 = 3, x2 = 2, x3 = 3, y1 = 3, y2 = 2.5, y3 = 2;

void setup() {

Serial.begin(9600);

pinMode(InputPin\_1, INPUT);

pinMode(InputPin\_2, INPUT);

pinMode(5,OUTPUT);

pinMode(6,OUTPUT);

Serial.println("CLEARDATA");

Serial.println("LABEL,x1,y1,x2,y2,x3,y3");

Serial.println("RESSETTIMER");

}

double f(double p1, double p2, double p3, double N, double u){

return N + u\*p1 + gamma\*(p2 + p3);

}

// double RK1(double f(double x, double y), double x0, double y0, double x, double h){ [\\Runge-Kutta](file:///\\Runge-Kutta) Integration

// double y,k1,k2;

// while(fabs(x-x0)>0.0000000001){

// k1=h\*f(x0,y0);

// k2=h\*f(x0+h/2.0,y0+k1/2.0);

// y=y0+k2;

// y0=y;

// x0=x0+h;

// }

// return y;

// }

//

// double RK2(double f(double x, double y), double x0, double y0, double x, double h){

// double y,k1,k2,k3,k4;

// while(fabs(x-x0)>0.0000000001){

// k1=h\*f(x0,y0);

// k2=h\*f(x0+h/2.0,y0+k1/2.0);

// k3=h\*f(x0+h/2.0,y0+k2/2.0);

// k4=h\*f(x0+h,y0+k3);

// y=y0+1/6.0\*(k1+2\*k2+2\*k3+k4);

// y0=y;

// x0=x0+h;

// }

// return y;

double euler(double p1, double p2, double p3, double N, double u){ [\\Euler](file:///\\Euler) integration

if (p1 > 4.99)

{p1=-4.5 + step\_time\*f(p1, p2, p3, N, u);}

else if (p1 < -4.99)

{p1=4.5 +step\_time\*f(p1, p2, p3, N, u);}

else

{p1=p1+step\_time\*f(p1, p2, p3, N, u);}

return p1;

}

double fullupdate(double p1, double p2, double p3){

int N1=analogRead(InputPin\_1);

int u1=analogRead(InputPin\_2);

double N = -1+(N1\*2/1024);

double u =-0.1+(u1/1024);

p1=euler(p1,p2,p3,N,u);

return p1;

}

void loop() {

Serial.print("DATA,");

\\Serial.println("x1");

\\Serial.println(255\*(x1+5)/10);

\\Serial.println("y1");

\\Serial.println(255\*(y1+5)/10);

x1=fullupdate(x1,x2,x3);

delay(20);

x2=fullupdate(x2,x3,x1);

delay(20);

x3=fullupdate(x3,x1,x2);

delay(20);

y1=fullupdate(y1,y2,y3);

delay(20);

y2=fullupdate(y2,y3,y1);

delay(20);

y3=fullupdate(y3,y1,y2);

analogWrite(5,255\*(x1+5)/10); [\\ give](file:///\\give) PWM output

analogWrite(6,255\*(y1+5)/10);

Serial.print(x1); [\\Write](file:///\\Write) to excel

Serial.print(",");

Serial.print(y1);

Serial.print(",");

Serial.print(x2);

Serial.print(",");

Serial.print(y2);

Serial.print(",");

Serial.print(x3);

Serial.print(",");

Serial.print(y3);

Serial.println("");

}