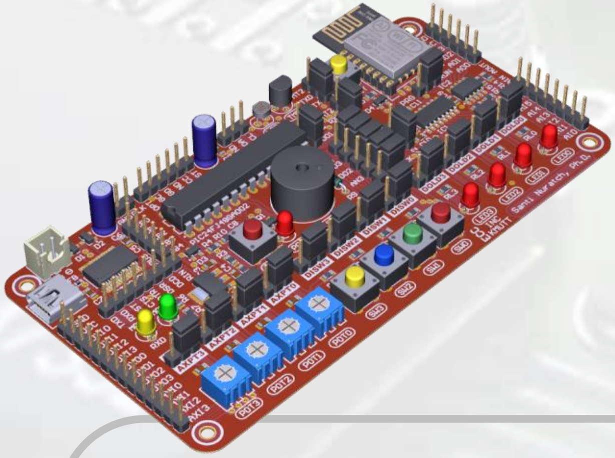
Week #03
Data Analysis and Machine Learning

Embedded Systems
&
Cloud Computing

we make computers do more.



Santi Nuratch., Ph.D.

Embedded Computing and Control Lab. @ INC-KMUTT

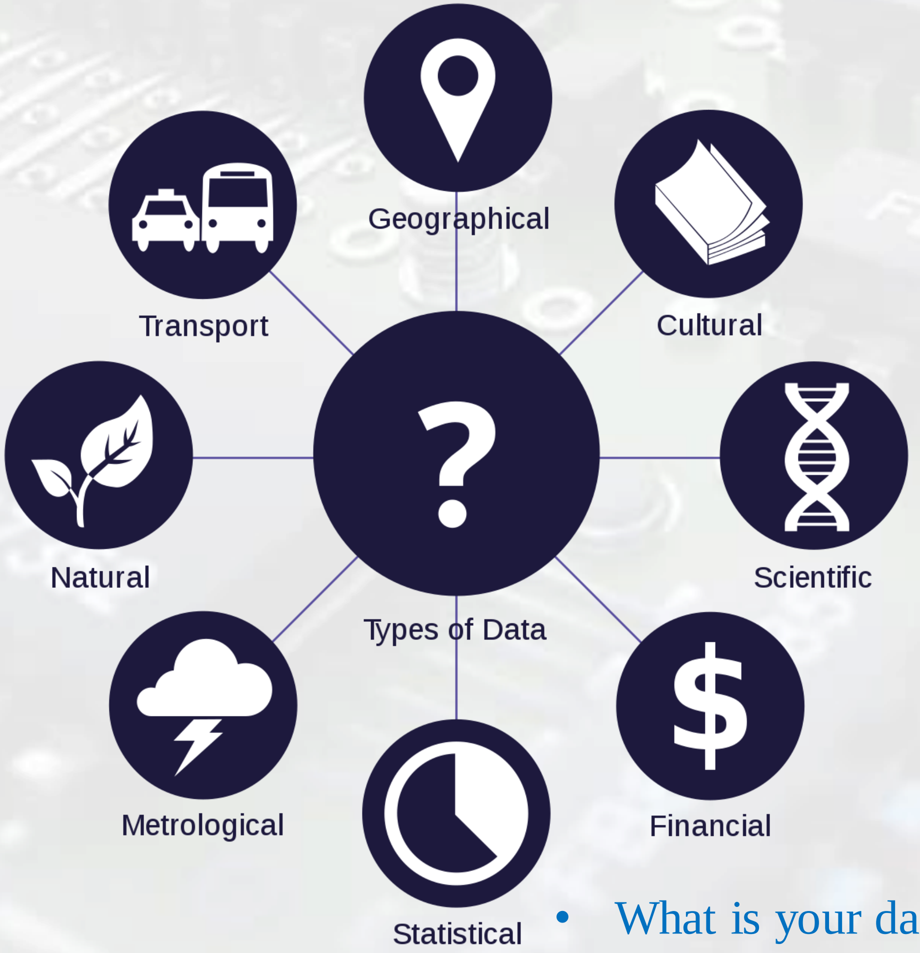
santi.inc.kmutt@gmail.com, santi.nur@kmutt.ac.th

Department of Control System and Instrumentation Engineering,
King Mongkut's University of Technology Thonburi, **KMUTT**

Data

http://www.mathsisfun.com/data/data.html

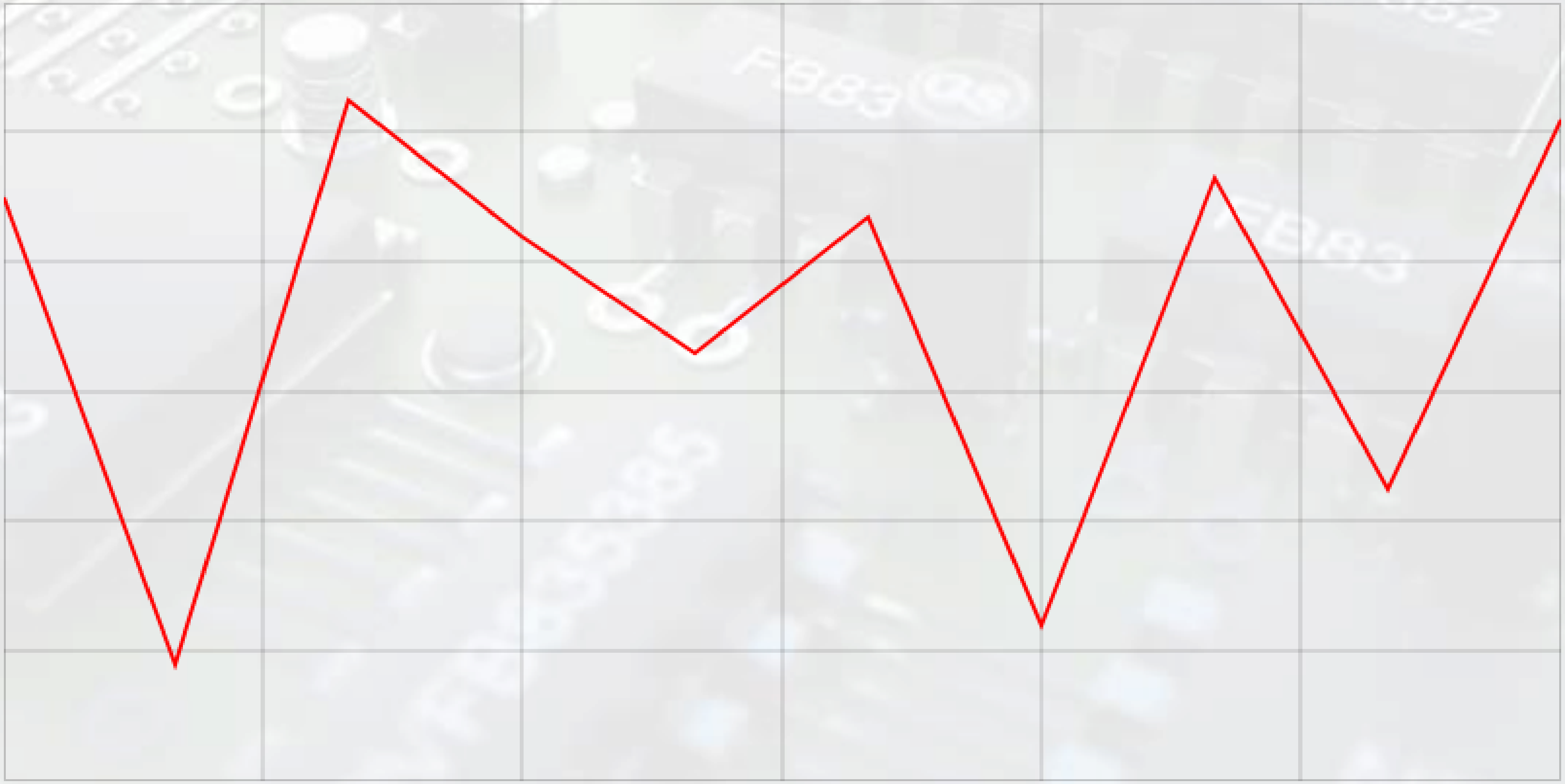
What is the Data? **Where** does it come from? **Why** we need it? **When** we use it?



- What is your data?
- How to extract the useful information from the data?
- What the appropriate algorithm used to perform the data?

Visualization (see before think)

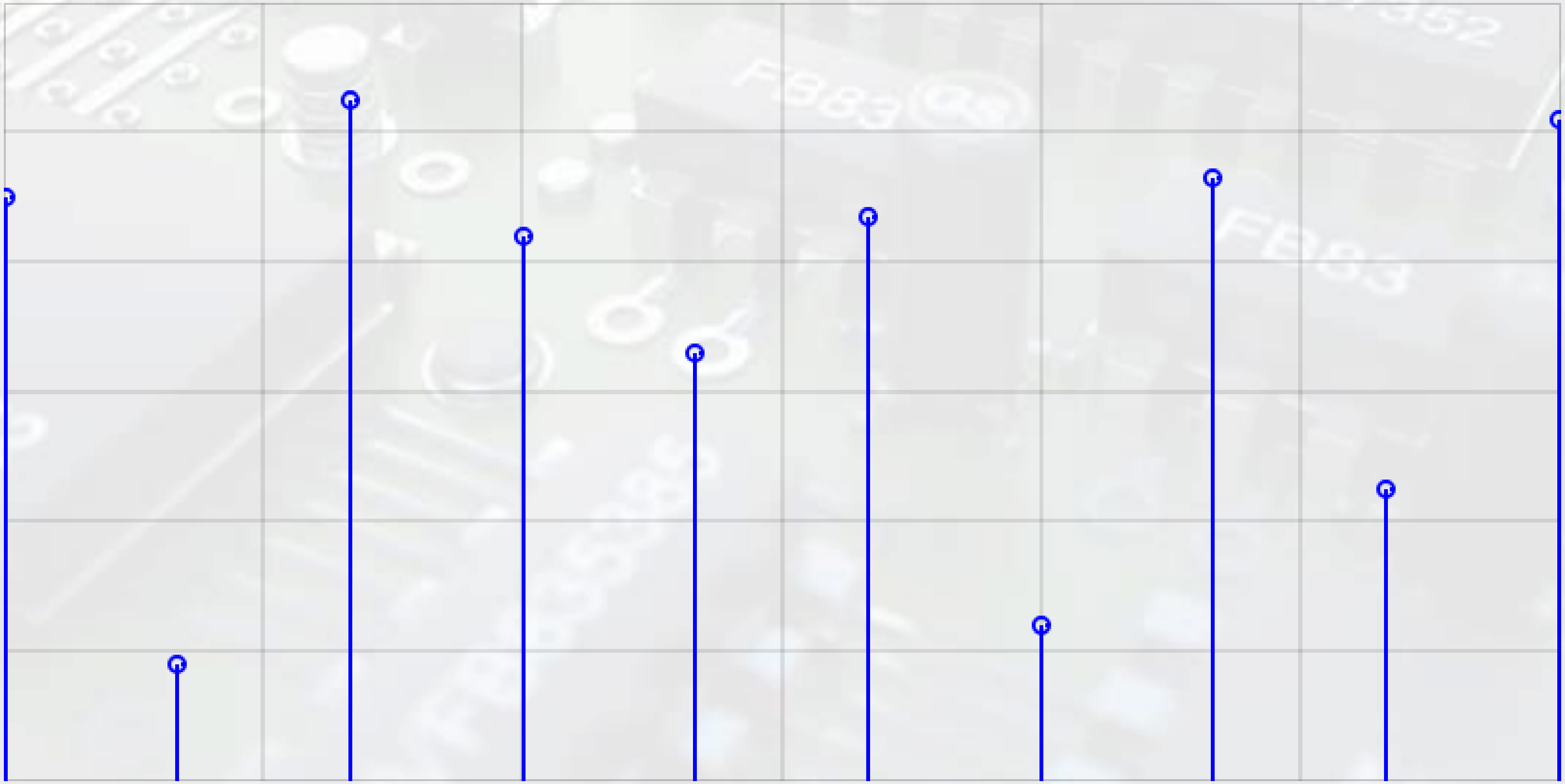
```
var data = [10, 14, 20, 8, 2, 9, 17];
const viewer = new DataViewer('canvas', 800, 400);
viewer.line(data, 'red', 2, true);
```



```
DataViewer.prototype.line = function(data, lineColor, lineWidth, redraw)
```

Visualization (see before think)

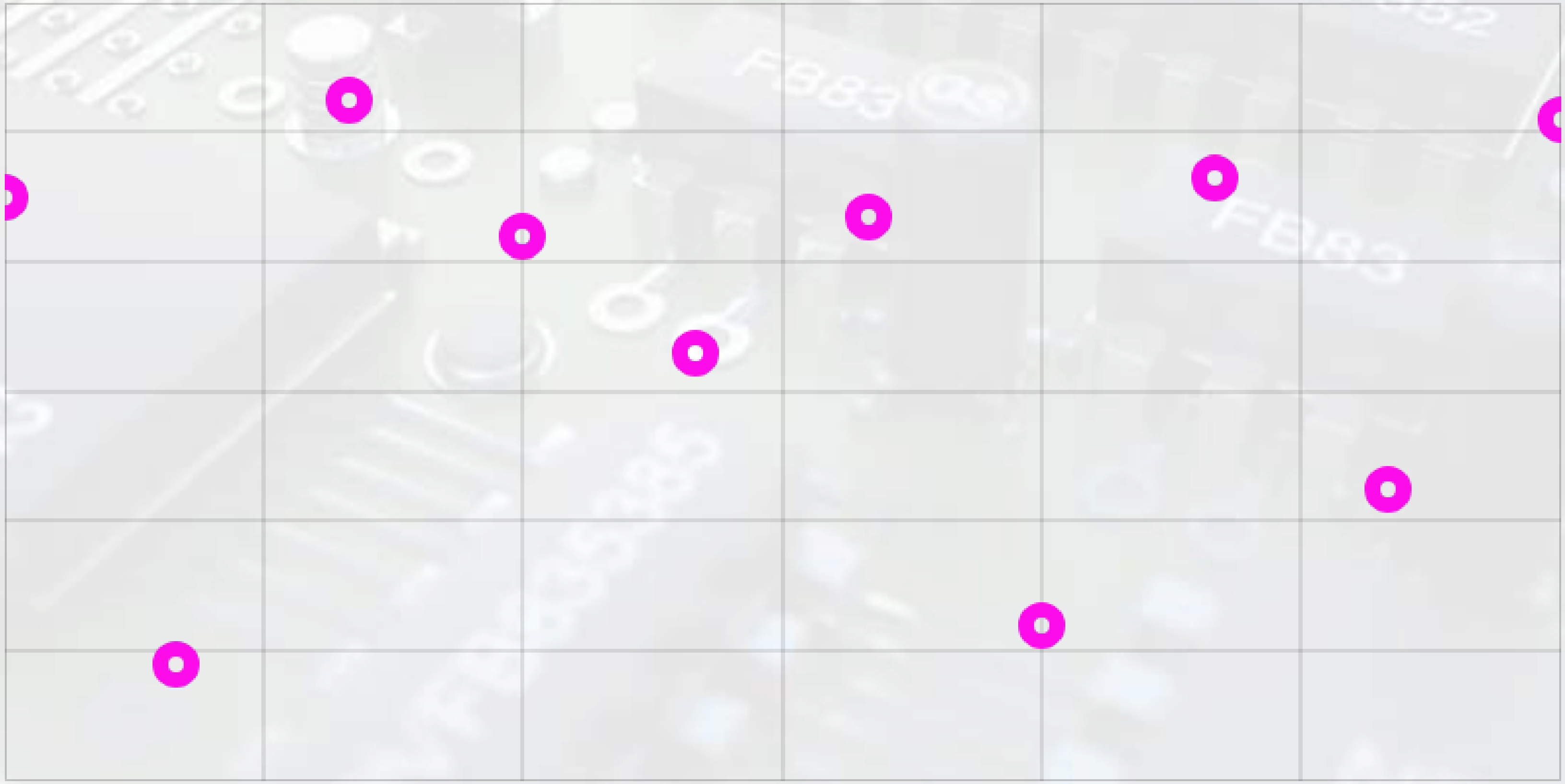
```
var data = [10, 14, 20, 8, 2, 9, 17];
const viewer = new DataViewer('canvas', 800, 400);
viewer.stem(data, ‘blue', 2, true);
```



```
DataViewer.prototype.stem = function(data, lineColor, lineWidth, redraw)
```

Visualization (see before think)

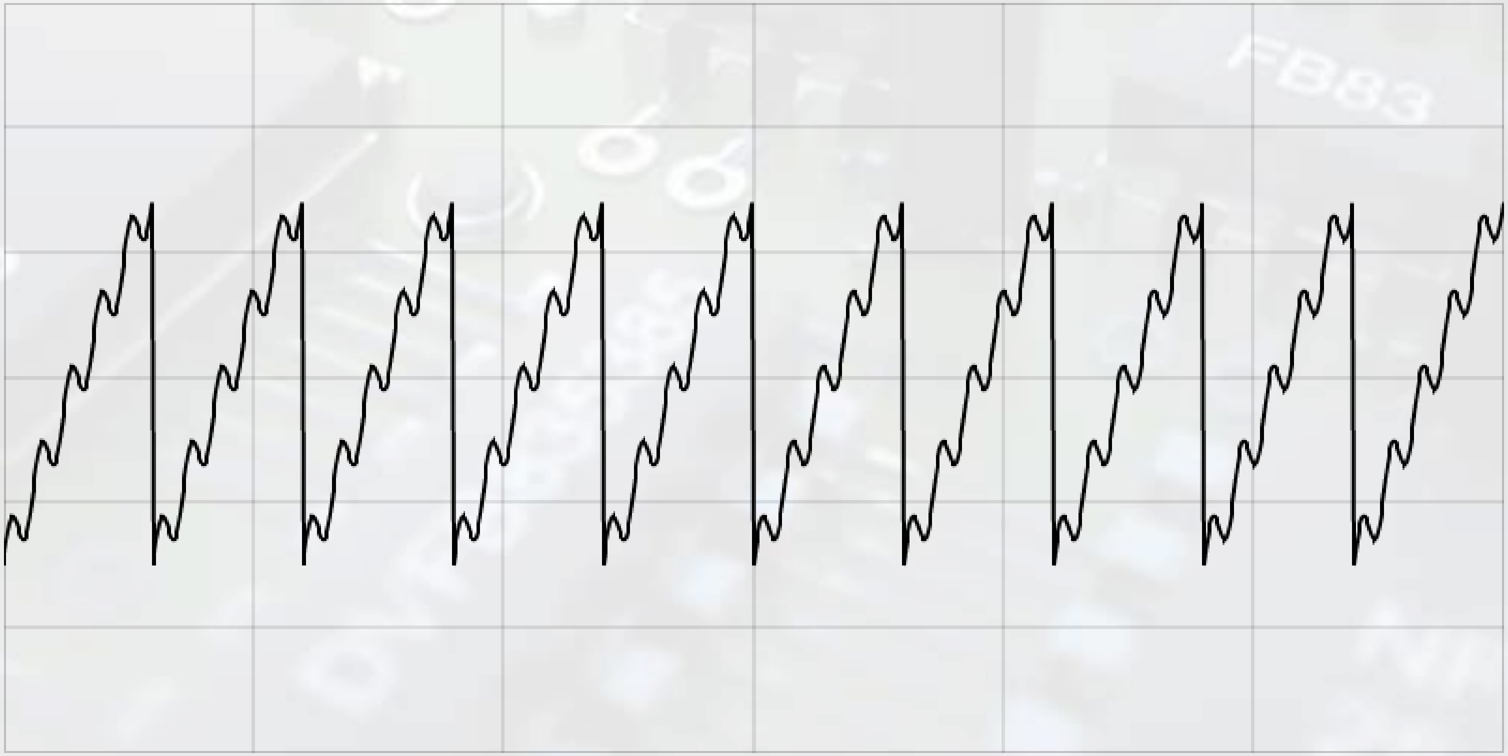
```
var data = [10, 14, 20, 8, 2, 9, 17];
const viewer = new DataViewer('canvas', 800, 400);
viewer.mark(data, '#fc0ce8', 8, true);
```



```
DataViewer.prototype.mark = function(data, lineColor, lineWidth, redraw)
```

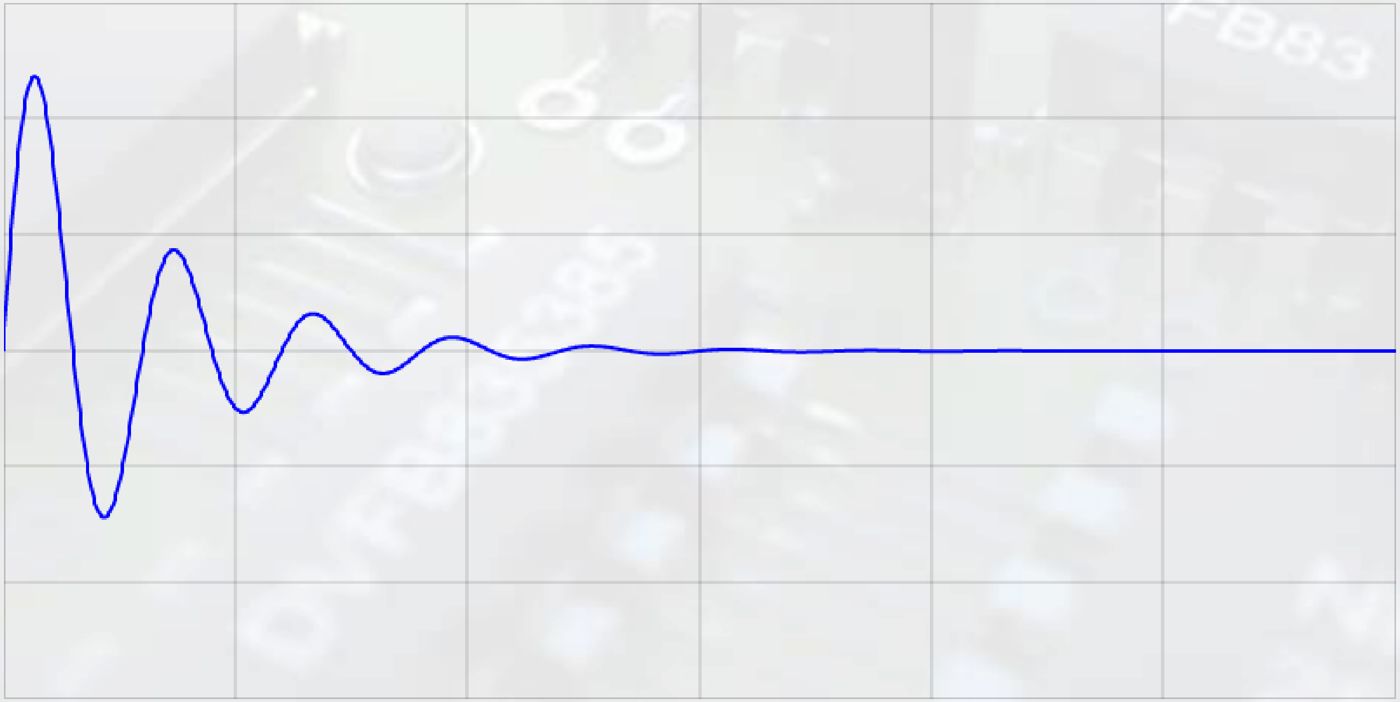
Engineering Data

```
let data = [];
for(let i=0; i<1000; i++) {
    data[i] = 15 * Math.sin(Math.PI*2*50*i/1000);
    data[i] += 2 * (i%100) - 100;
}
const viewer = new DataViewer('canvas', 800, 400);
viewer.line(data, 'black', 2, true);
```



Engineering Data

```
let data = [];
for(let i=0; i<1000; i++) {
    data[i] = 200 * Math.sin(Math.PI*2*10*i/1000);
    data[i] *= Math.exp(-i/100);
}
const viewer = new DataViewer('canvas', 800, 400);
viewer.line(data, 'blue', 2, true);
```



Data Generation (Normal Distribution)

1) Generate $\mu_1 = \mu(-1,+1)$ and $\mu_2 = \mu(-1,+1)$

2) Calculate $w = \ (\mu_1)^2 + (\mu_2)^2$

3) Repeat step 2) untill $0 < w < 1$

4) Calculate $x_1 = \mu_1 \sqrt{\dfrac{-2\ln(w)}{w}}, \quad x_2 = \mu_2 \sqrt{\dfrac{-2\ln(w)}{w}}$

The $\mu(-1,+1)$ is the Uniform Distribution with range $[-1,+1]$

The x_1 and x_2 are normal random variables with mean 0 and standard deviation 1

To generate normal random variable from mean μ and standard deviation σ

we need to do the following transformation

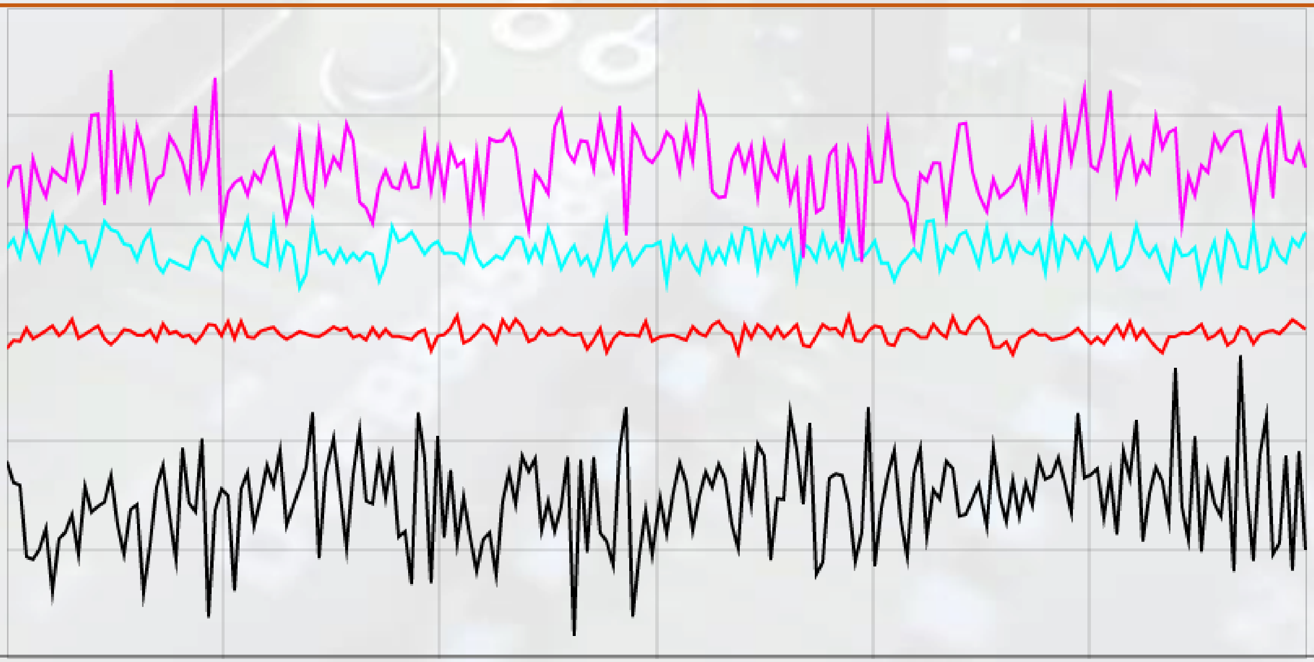
$$X_{\mu,\sigma} = \mu + \sigma X_{0,1}$$

$X_{0,1}$ □ □ $(0,1)$ and $X_{\mu,\sigma}$ □ □ (μ,σ)

```
Data.prototype.randomMultipleNumbers = function (mu, sigma, items)
```

Data Generation (Normal Distribution)

```
const generator   = new Data();
const viewer = new DataViewer('canvas', 800, 400);
const data1 = generator.randomMultipleNumbers(   0,  5,  200);
const data2 = generator.randomMultipleNumbers(  50, 10,  200);
const data3 = generator.randomMultipleNumbers( 100, 20,  200);
const data4 = generator.randomMultipleNumbers(-100, 30,  200);
viewer.line(data1, '#f00', 2, true);
viewer.line(data2, '#0ff', 2, false);
viewer.line(data3, '#f0f', 2, false);
viewer.line(data4, '#000', 2, false);
```



Data Analysis

Mean

$$\mu = \frac{1}{N}\sum_{i=1}^{n} X_i = \frac{1}{N}\sum X_i$$

Variance

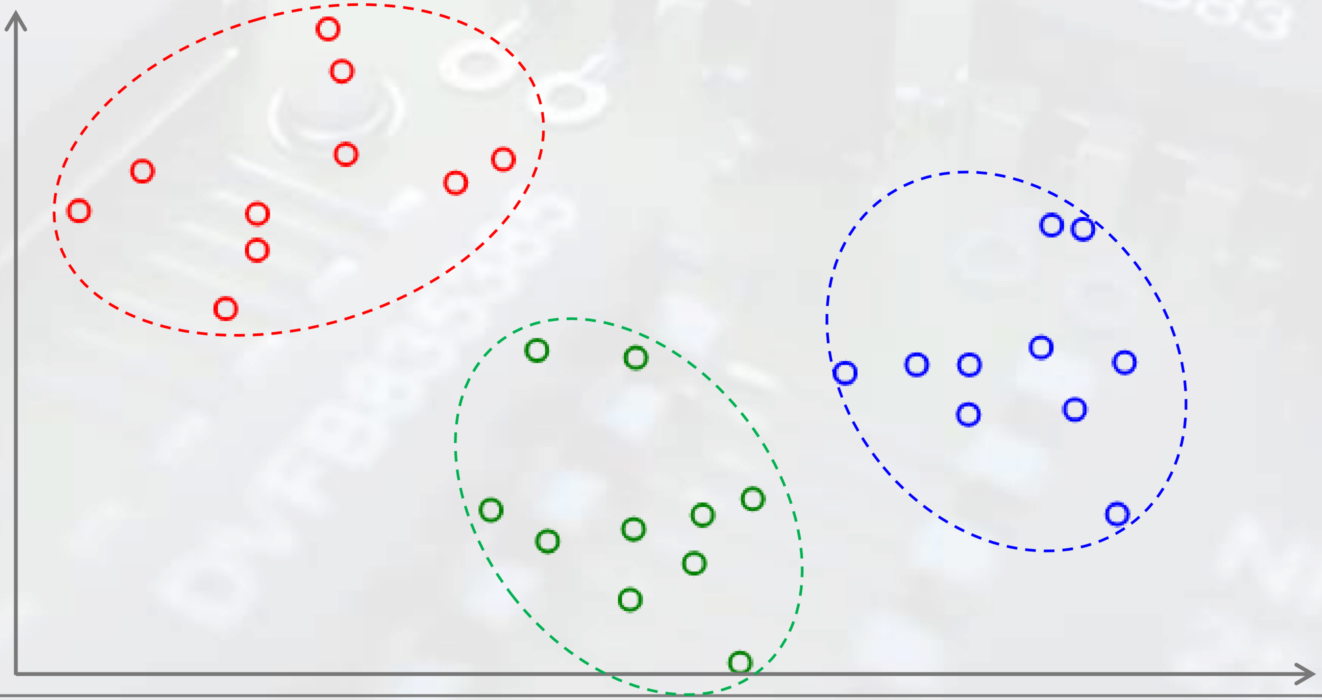
$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{n}\left(X_i - \mu\right)^2 = \frac{1}{N}\sum\left(X - \mu\right)^2$$

Standard Deviation

$$\sigma\sqrt{\frac{1}{N}\sum_{i=1}^{n}\left(X_i - \mu\right)^2} = \sqrt{\frac{1}{N}\sum\left(X - \mu\right)^2} = \sqrt{\sigma^2}$$

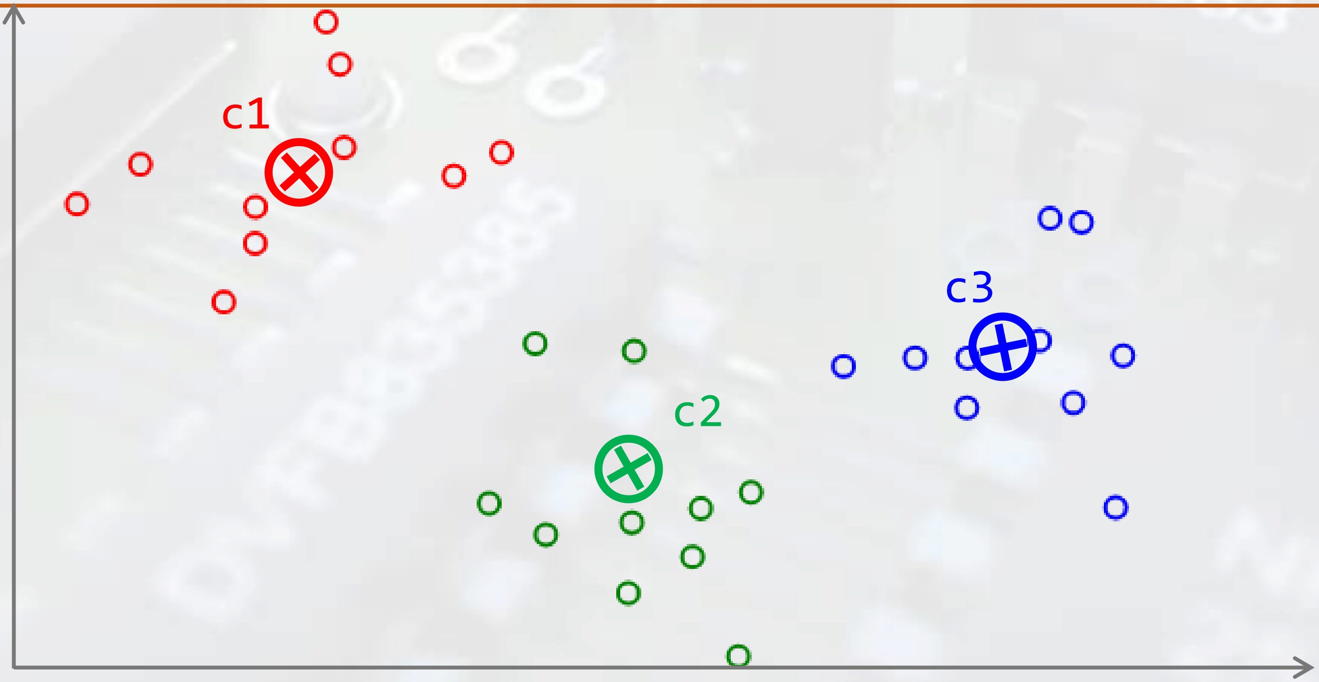
Two Dimensional Data

```
let viewer = new PointViewer('canvas', 600, 600);
let point = new PointData();
let numPoints = 10;
let d1 = point.randomMultiplePoints(150, 50,  200, 50, numPoints);
let d2 = point.randomMultiplePoints(300, 50,  350, 50, numPoints);
let d3 = point.randomMultiplePoints(500, 50,  300, 50, numPoints);
d1.map(d=>d.color='red');   viewer.draw(d1);
d2.map(d=>d.color='green'); viewer.draw(d2);
d3.map(d=>d.color='blue');  viewer.draw(d3);
```



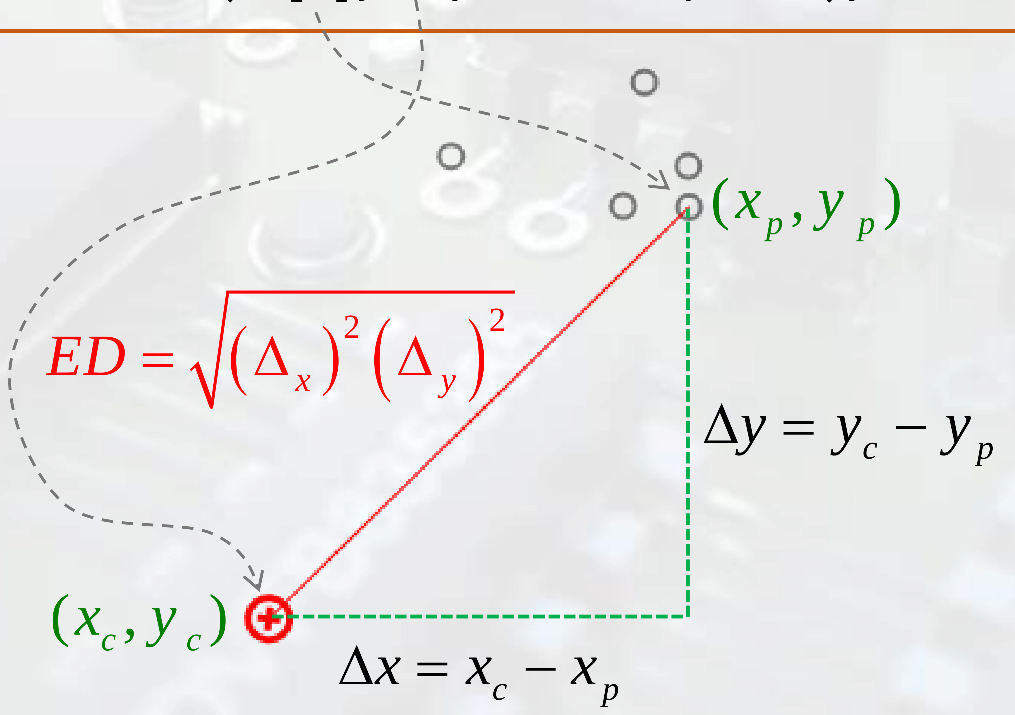
Centroids (centroid is a moveable point)

```
let viewer = new PointViewer('canvas', 600, 600);
let point = new PointData();
let d1 = point.randomMultiplePoints(150, 50,  100, 20, 30);
let d2 = point.randomMultiplePoints(300, 40,  350, 30, 30);
let d3 = point.randomMultiplePoints(500, 20,  300, 50, 30);
viewer.draw(d1); viewer.draw(d2); viewer.draw(d3);
let c1 = point.randomSinglePoint(150, 20,  100, 20).toCentroid('red');
let c2 = point.randomSinglePoint(300, 20,  350, 20).toCentroid('green');
let c3 = point.randomSinglePoint(500, 20,  300, 20).toCentroid('blue');
viewer.draw(c1); viewer.draw(c2); viewer.draw(c3);
```



Euclidean Distance Between Points

```
let viewer = new PointViewer('canvas', 600, 600);
let point = new PointData();
let d1 = point.randomMultiplePoints(300, 50,  100, 20, 5);
viewer.draw(d1);
let c1 = point.randomSinglePoint(100, 100,  300, 20).toCentroid('red');
viewer.draw(c1);
viewer.drawEucredient(d1[2], c1, c1.color, true);
```



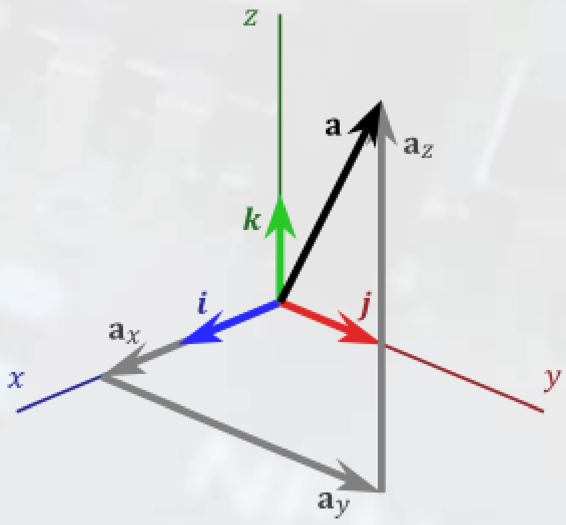
$$(x_p, y_p)$$

$$ED = \sqrt{(\Delta_x)^2 (\Delta_y)^2}$$

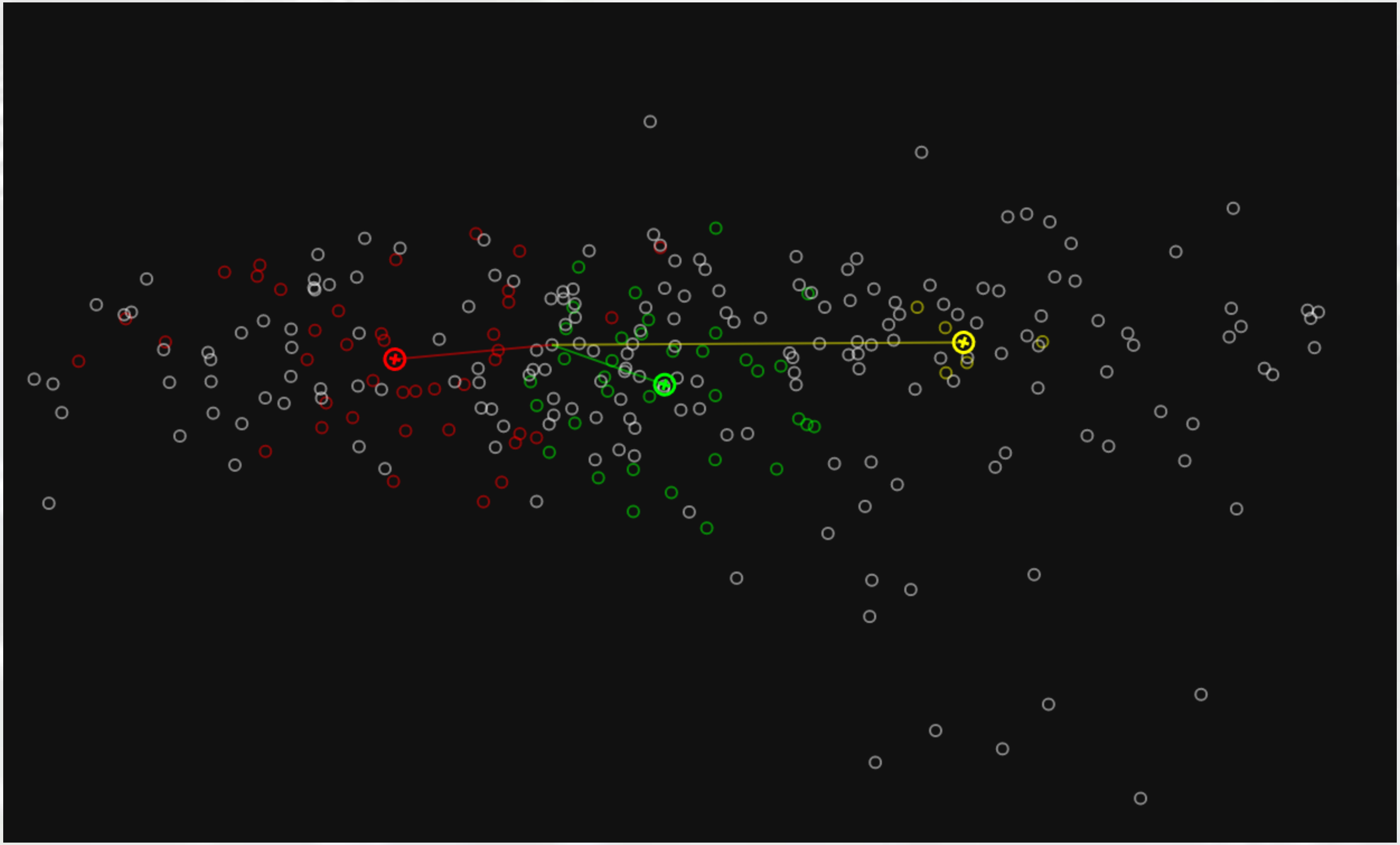
$$\Delta y = y_c - y_p$$

$$(x_c, y_c)$$

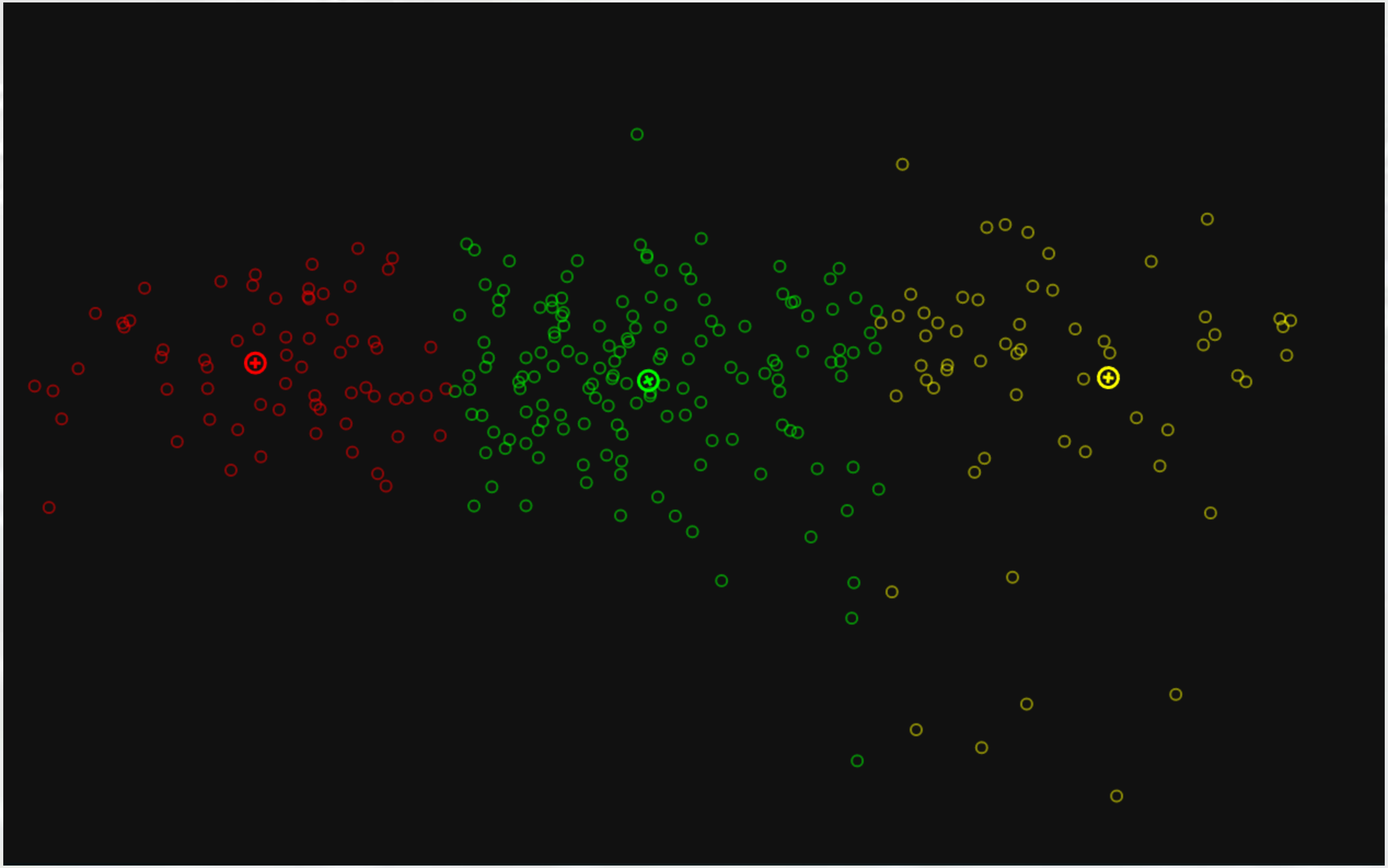
$$\Delta x = x_c - x_p$$



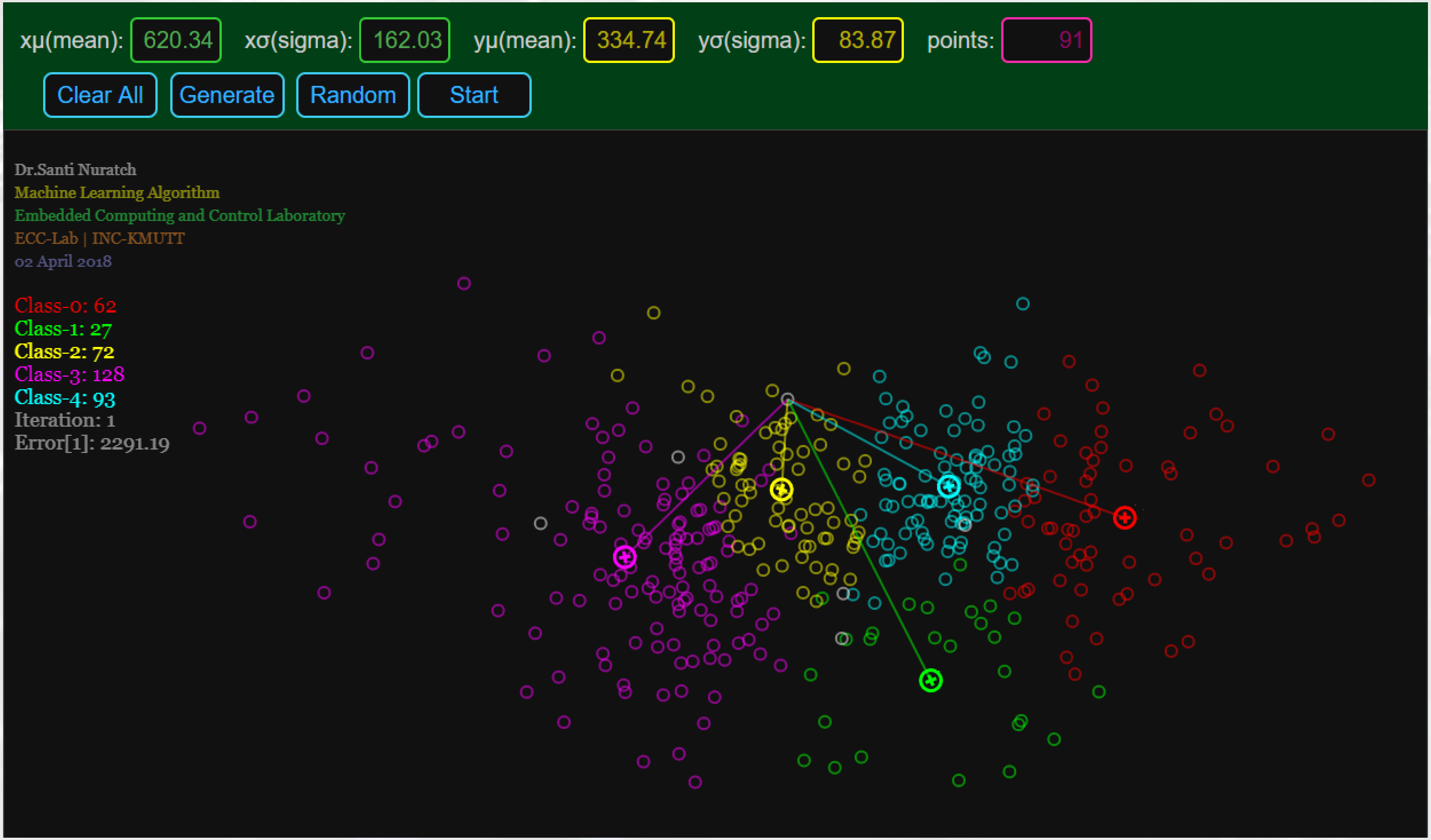
Class/Group Assignment



Minimize Distance



Machine Learning Algorithm (KMeans)



xμ(mean): 620.34 xσ(sigma): 162.03 yμ(mean): 334.74 yσ(sigma): 83.87 points: 91

Dr.Santi Nuratch
Machine Learning Algorithm
Embedded Computing and Control Laboratory
ECC-Lab | INC-KMUTT
02 April 2018

Class-0: 62
Class-1: 27
Class-2: 72
Class-3: 128
Class-4: 93
Iteration: 1
Error[1]: 2291.19

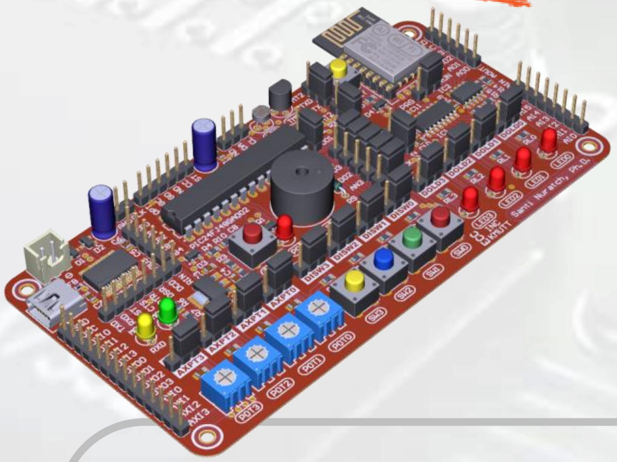
THANK YOU!

ECC

KMUTT

KMUTT INC

we make computers do more.



MPLAB XC COMPILER

Santi Nuratch., Ph.D.

Embedded Computing and Control Lab. @ INC-KMUTT

santi.inc.kmutt@gmail.com, santi.nur@kmutt.ac.th

Department of Control System and Instrumentation Engineering,
King Mongkut's University of Technology Thonburi, **KMUTT**