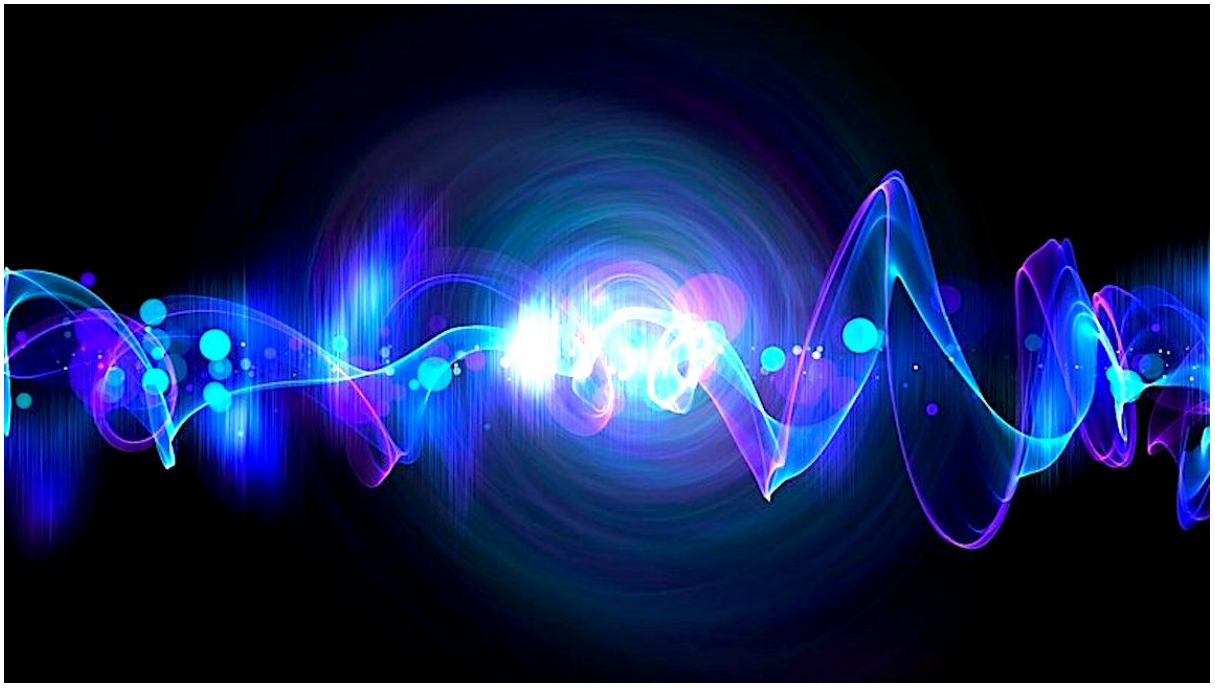
SIGNALS AND SYSTEMS

CONVOLUTION AND CORRELATION



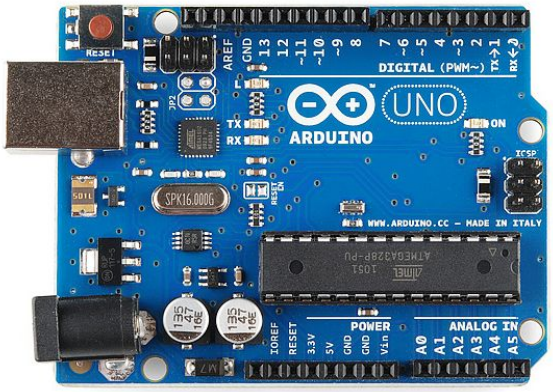
DATE - 13TH September 2019

NAME - Shivam Malviya

ROLL NO. - 18EC01044

INTRODUCTION

Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing.



Arduino Uno is a microcontroller board based on the ATmega328P (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

THEORY

• CONVOLUTION

Convolution is a mathematical way of combining two signals to form a third signal. It is the single most important technique in Digital Signal Processing. Using the strategy of impulse decomposition, systems are described by a signal called the impulse response. Convolution is important because it relates the three signals of interest: the input signal, the output signal, and the impulse response.

Convolution is used to express the relation between input and output of an LTI system. It relates input, output and impulse response of an LTI system as

$$y(t)=x(t)*h(t)$$

Where y (t) = output of LTI

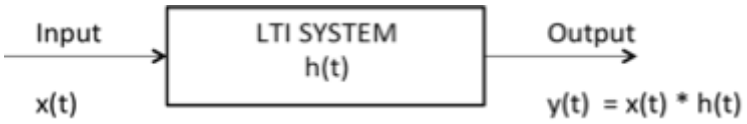
x (t) = input of LTI

h (t) = impulse response of LTI

There are two types of convolutions:

- *Continuous convolution*
- *Discrete convolution*

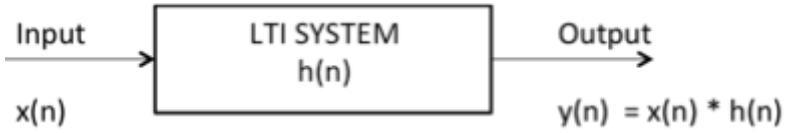
CONTINUOUS CONVOLUTION



$$y(t)=x(t)*h(t)$$

$$y(t) = \int_{-\infty}^{\infty} x(\tau)h(t-\tau)d\tau$$

DISCRETE CONVOLUTION



$$y(n)=x(n)*h(n)$$

$$y(n) = \Sigma^{\infty}_{k=-\infty} x(k)h(n-k) = \Sigma_{k=-\infty} x(n-k)h(k)$$

● CROSS CORRELATION

In general, correlation describes the mutual relationship which exists between two or more things. The same definition holds good even in the case of signals. That is, correlation between signals indicates the measure up to which the given signal resembles another signal.

In other words, if we want to know how much similarity exists between the signals 1 and 2, then we need to find out the correlation of Signal 1 with respect to Signal 2 or vice versa.

Depending on whether the signals considered for correlation are the same or different, we have two kinds of correlation:

- *Autocorrelation .*
- *Cross-correlation.*

Autocorrelation

This is a type of correlation in which the given signal is correlated with itself, usually the time-shifted version of itself. Mathematical expression for the autocorrelation of continuous time signal x (t) is given by

$$Rxx(\tau)=\int^{\infty}_{-\infty} x(t) * x^{\star}(t-\tau)dt$$

where ⋆ denotes the complex conjugate.

Similarly the autocorrelation of the discrete time signal x[n] is expressed as

$$Rxx[m]=\sum^{\infty}_{n=-\infty} x[n] * x^{\star}[n-m]$$

Next, the autocorrelation of any given signal can also be computed by resorting to graphical technique. The procedure involves sliding the time-shifted version of the given signal upon itself while computing the samples at every interval. That is, if the given signal is digital, then we shift the given signal by one sample every time and overlap it with the original signal. While doing so, for every shift and overlap, we perform multiply and add.

Cross-Correlation

This is a kind of correlation, in which the signal in-hand is correlated with another signal so as to know how much resemblance exists between them. Mathematical expression for the cross-correlation of continuous time signals x (t) and y (t) is given by

$$R_{xy}(\tau) = \int_{-\infty}^{\infty} x(t)\, y^{*}(t-\tau)dt$$

Similarly, the cross-correlation of the discrete time signals x [n] and y [n] is expressed as

$$R_{xy}[m] = \sum_{n=-\infty}^{\infty} x[n]\, y^{*}[n-m]$$

Next, just as is the case with autocorrelation, cross-correlation of any two given signals can be found via graphical techniques. Here, one signal is slid upon the other while computing the samples at every interval. That is, in the case of digital signals, one signal is shifted by one sample to the right each time, at which point the sum of the product of the overlapping samples is computed.

DISCUSSION

ADVANTAGES OF ARDUINO

***1 Ready to Use**: The biggest advantage of Arduino is its ready to use structure. As Arduino comes in a complete package form which includes the 5V regulator, a burner, an oscillator, a microcontroller, serial communication interface, LED and headers for the connections. You don't have to think about programmer connections for programming or any other interface. Just plug it into the USB port of your computer and that's it. Your revolutionary idea is going to change the world after just a few words of coding.*

***2 Examples of codes:** Another big advantage of Arduino is its library of examples present inside the software of Arduino. I'll explain this advantage using an example of voltage measurement. For example if you want to measure voltage using ATmega8 microcontroller and want to display the output on the computer screen then you have to go through the whole process. The process will start from learning the ABC's of micro-controller for measurement, went through the learning of serial communication for display and will end at USB - Serial converters.*

3 Effortless functions: *During coding of Arduino, you will notice some functions which make life so easy. Another advantage of Arduino is its automatic unit conversion capability. You can say that during debugging you don't have to worry about the units conversions. Just use your all force on the main parts of your projects. You don't have to worry about side problems.*

4 Large community: *There are many forums present on the internet in which people are talking about the Arduino. Engineers, hobbyists and professionals are making their projects through Arduino. You can easily find help about everything. Moreover the Arduino website itself explains each and every functions of Arduino.*

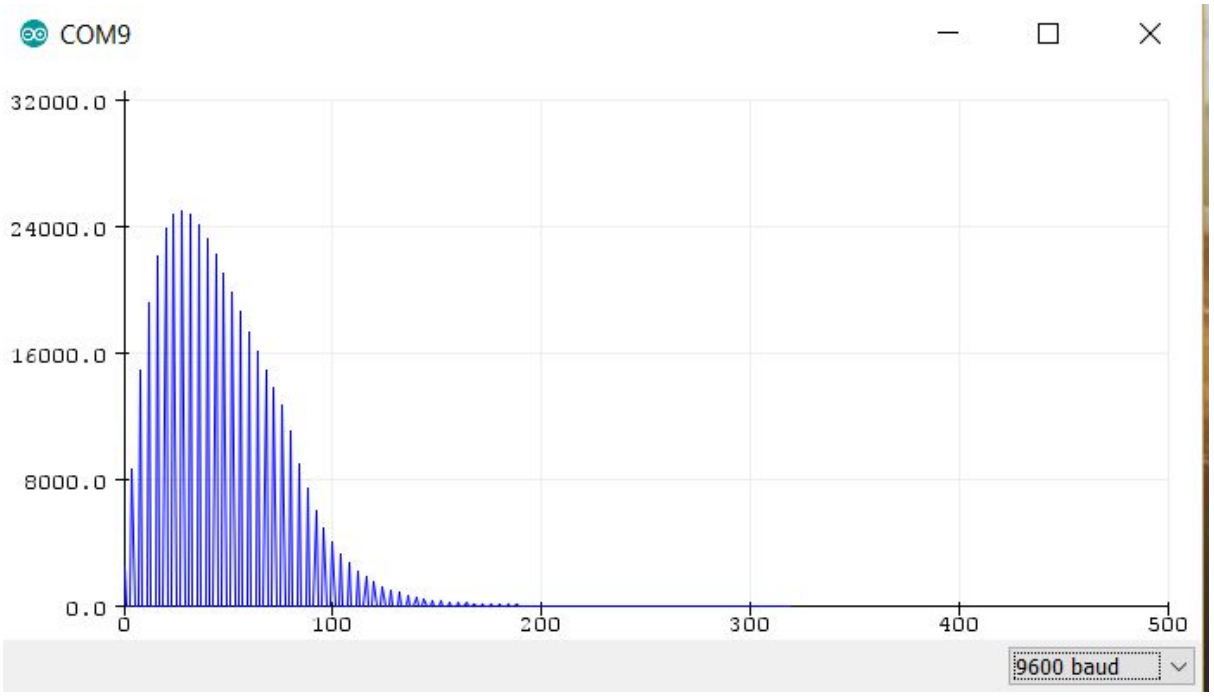
CONCLUSION

We should conclude the advantage of Arduino by saying that while working on different projects you just have to worry about your innovative idea. The remaining will handle by Arduino itself.

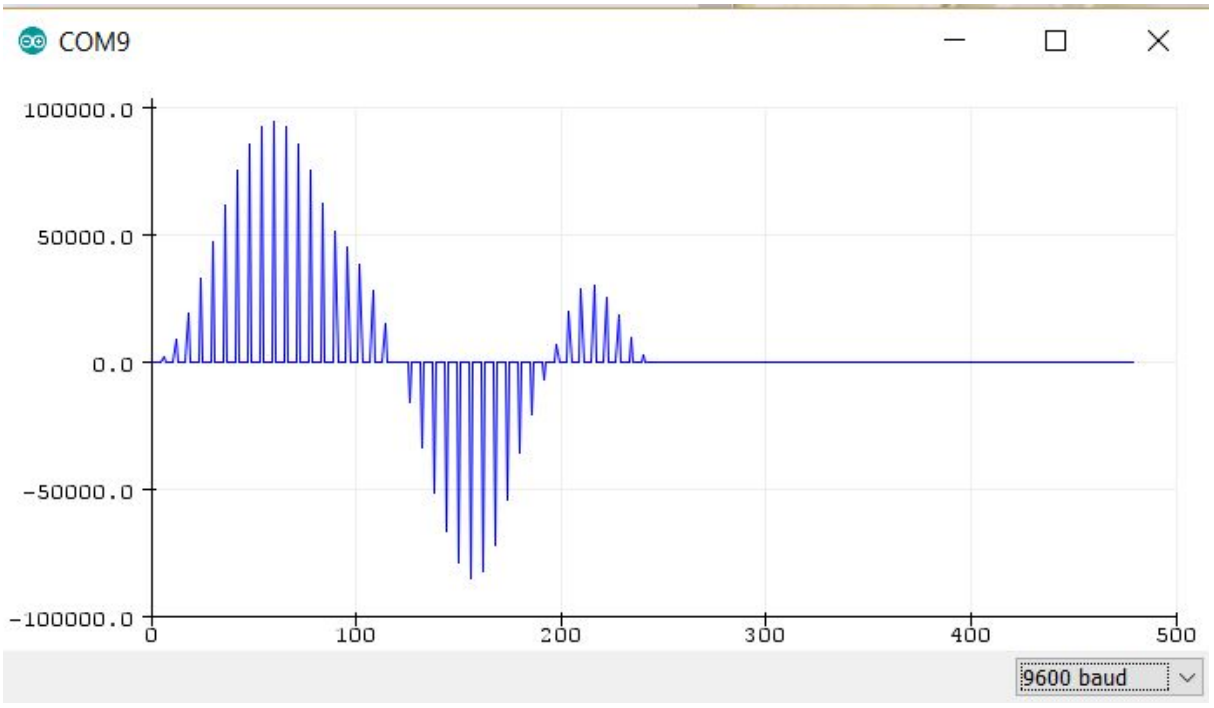
RESULTS

1. PART A

1.

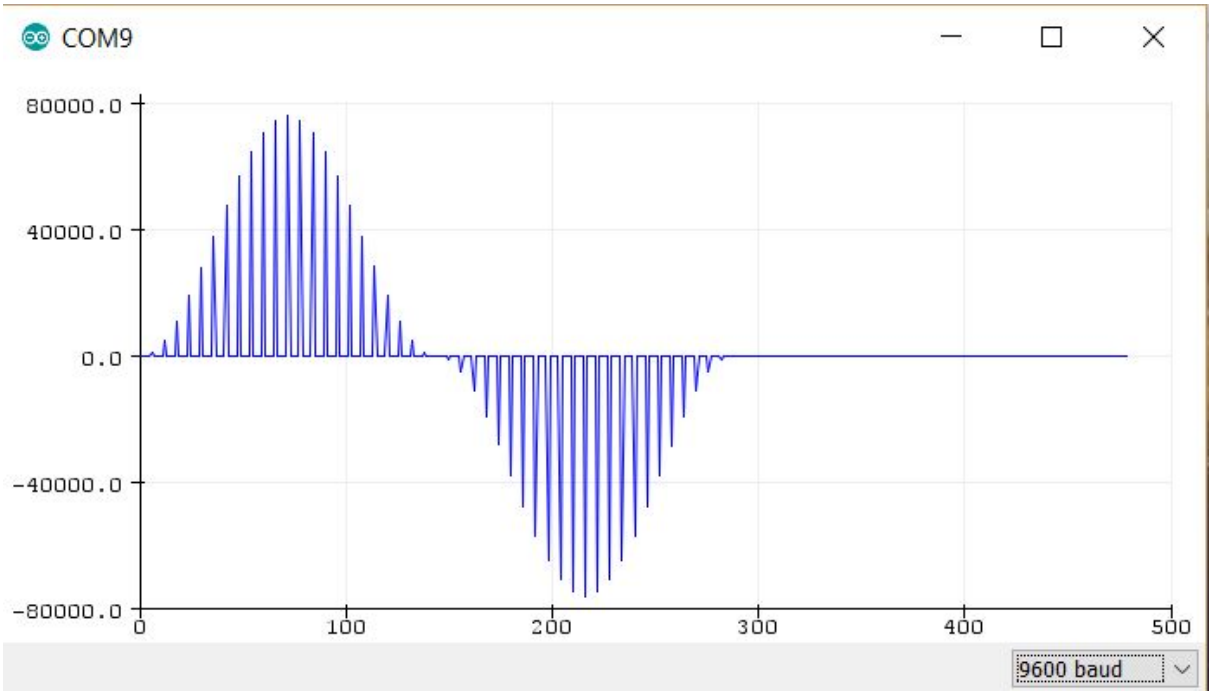


2.

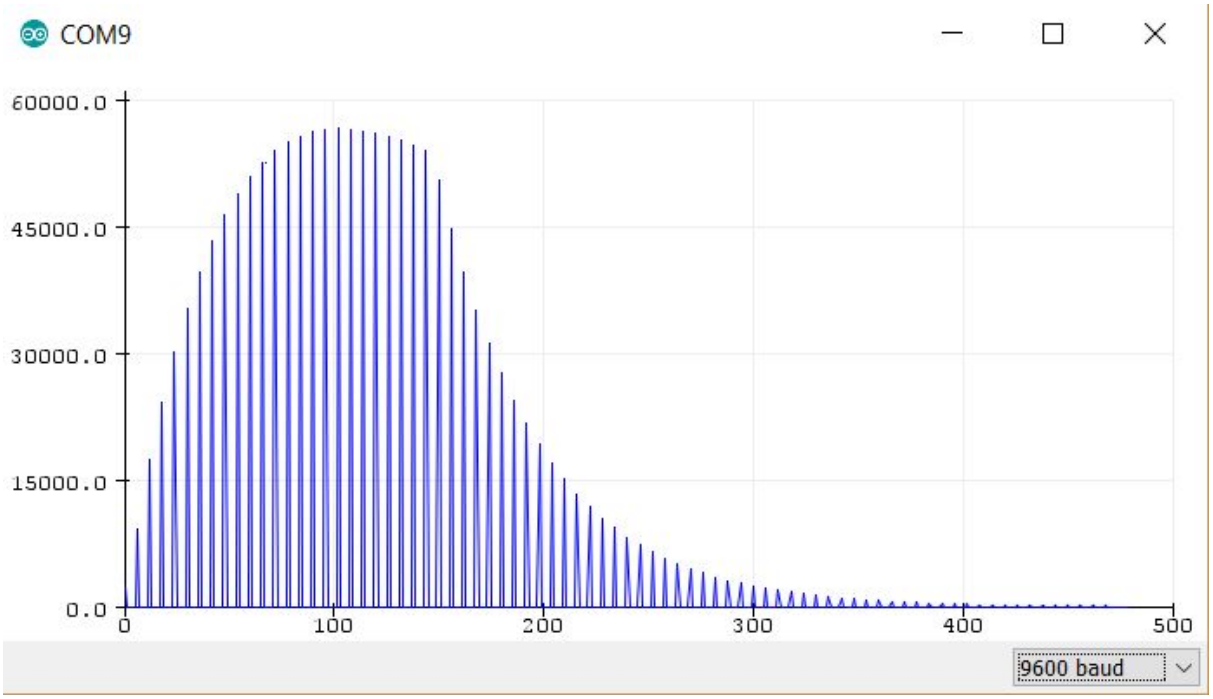


2. PART B

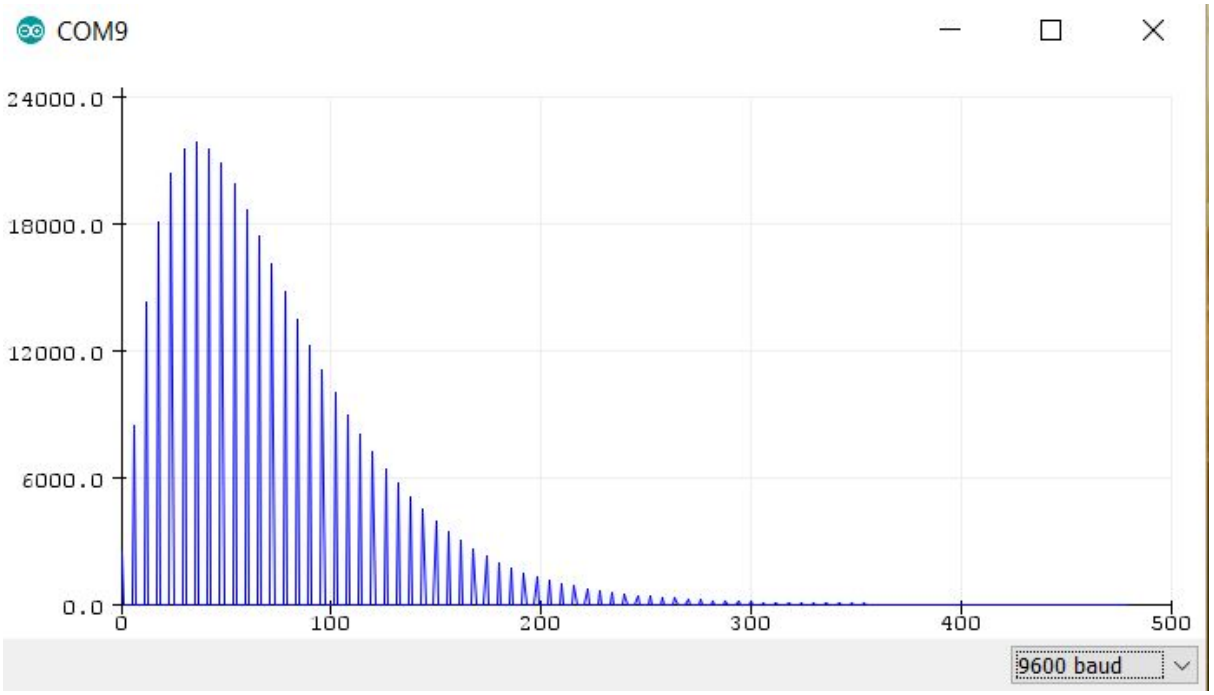
1.


COM9
80000.0
40000.0
0.0
-40000.0
-80000.0
0
100
200
300
400
500
9600 baud

2.

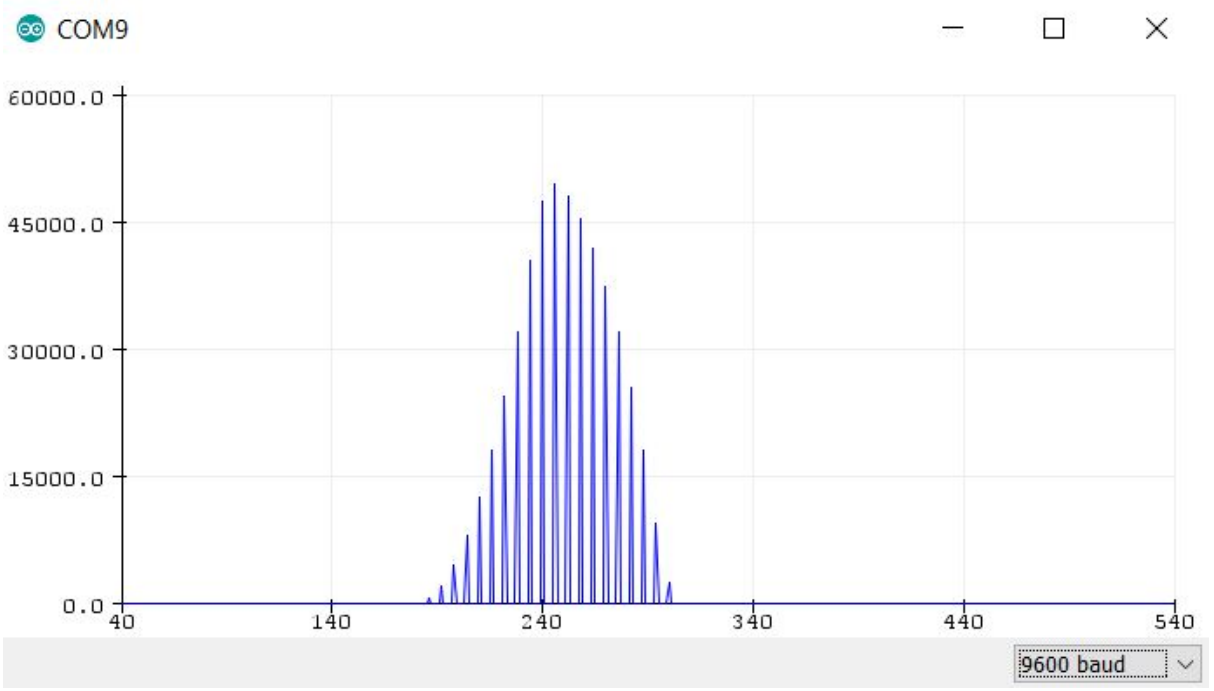

COM9
60000.0
45000.0
30000.0
15000.0
0.0
0
100
200
300
400
500
9600 baud

3.

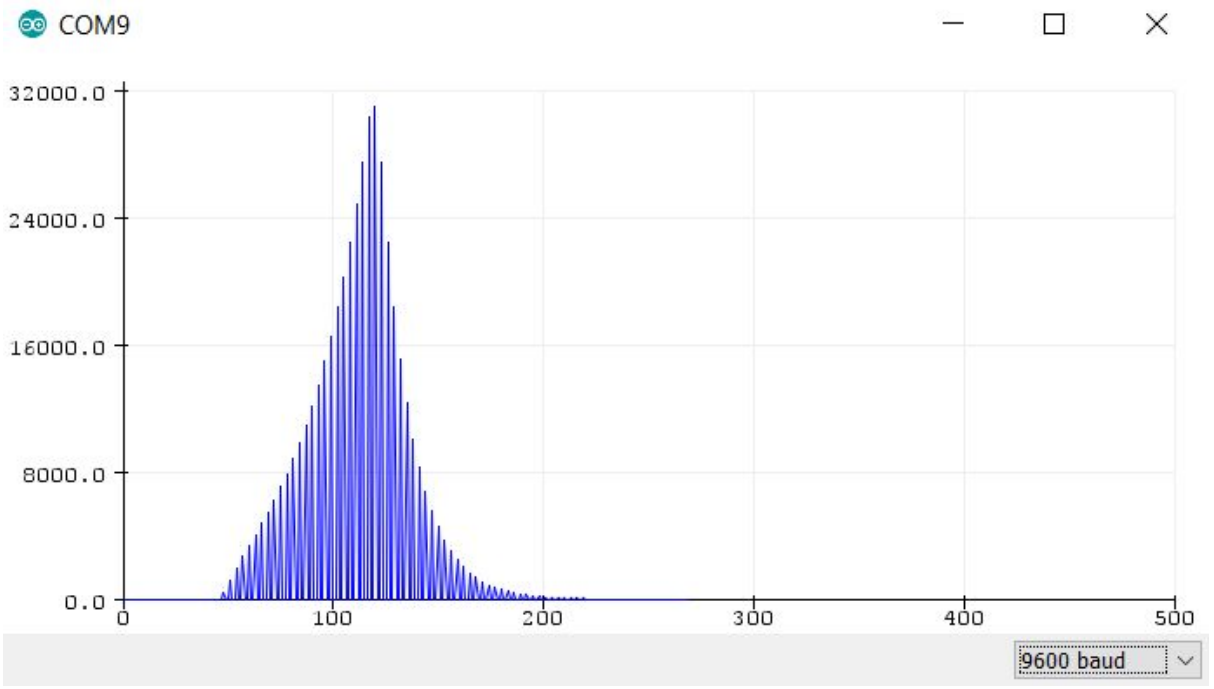

COM9
24000.0
18000.0
12000.0
6000.0
0.0
0
100
200
300
400
500
9600 baud

3.PART C

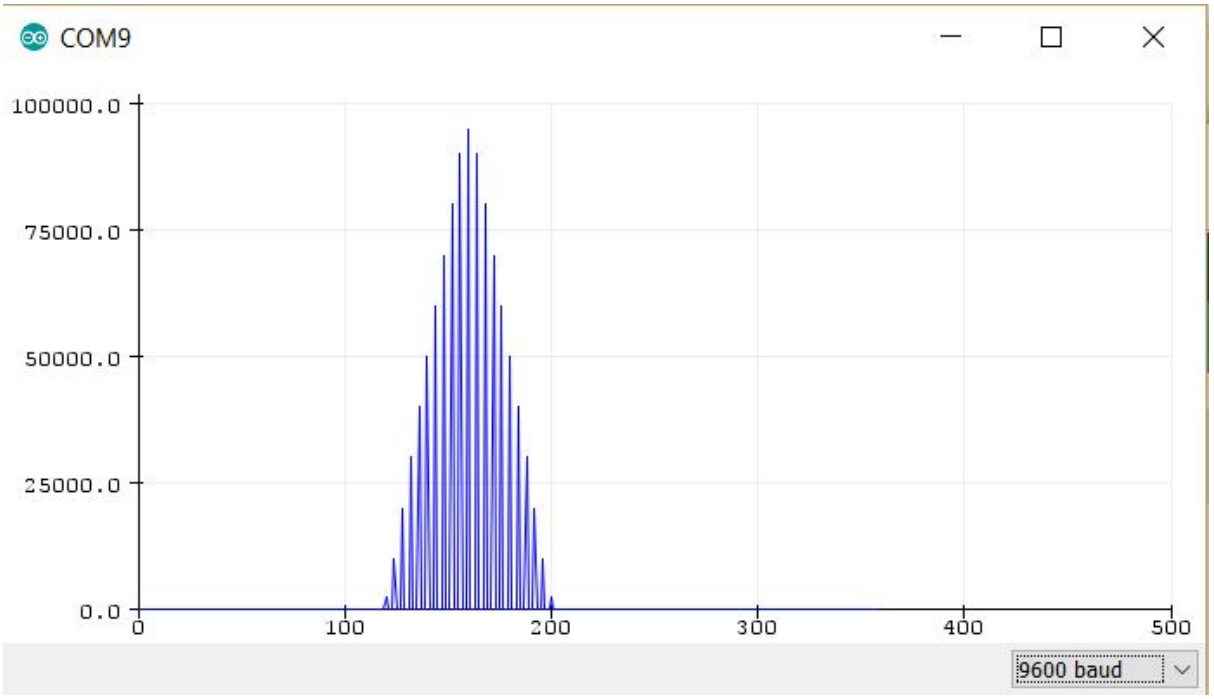
1.


COM9
60000.0
45000.0
30000.0
15000.0
0.0
40
140
240
340
440
540
9600 baud

2.

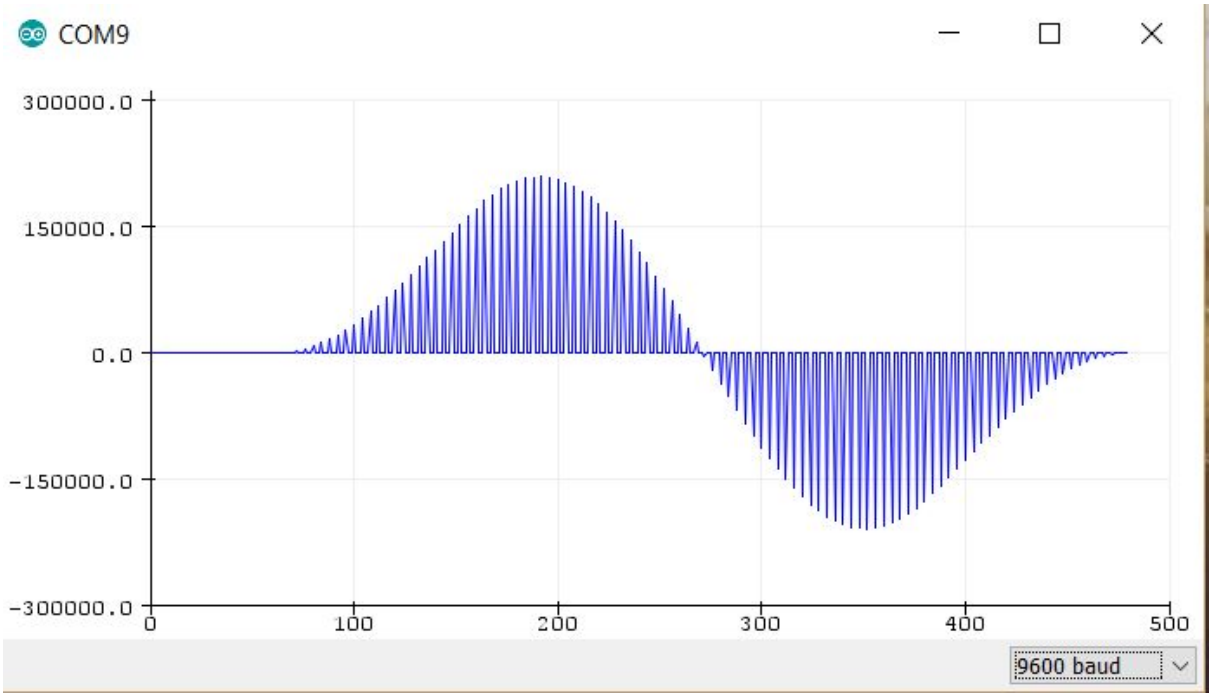

COM9
32000.0
24000.0
16000.0
8000.0
0.0
0
100
200
300
400
500
9600 baud

3.

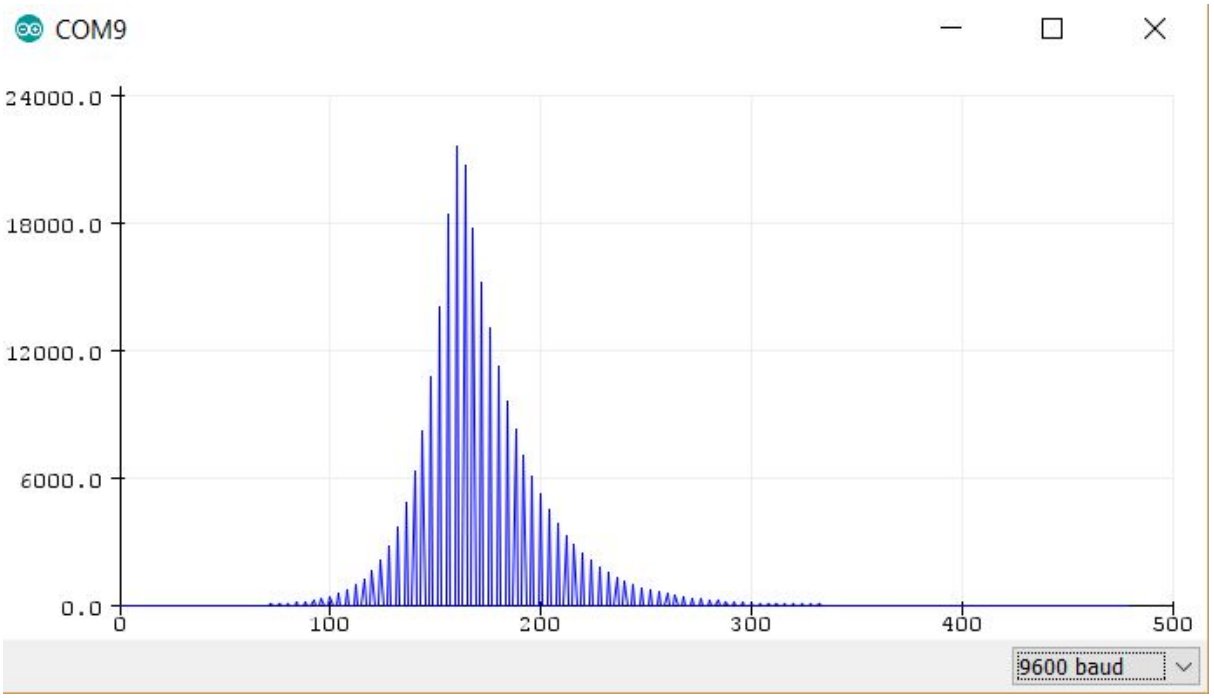

COM9
100000.0
75000.0
50000.0
25000.0
0.0
0
100
200
300
400
500
9600 baud

4. PART D

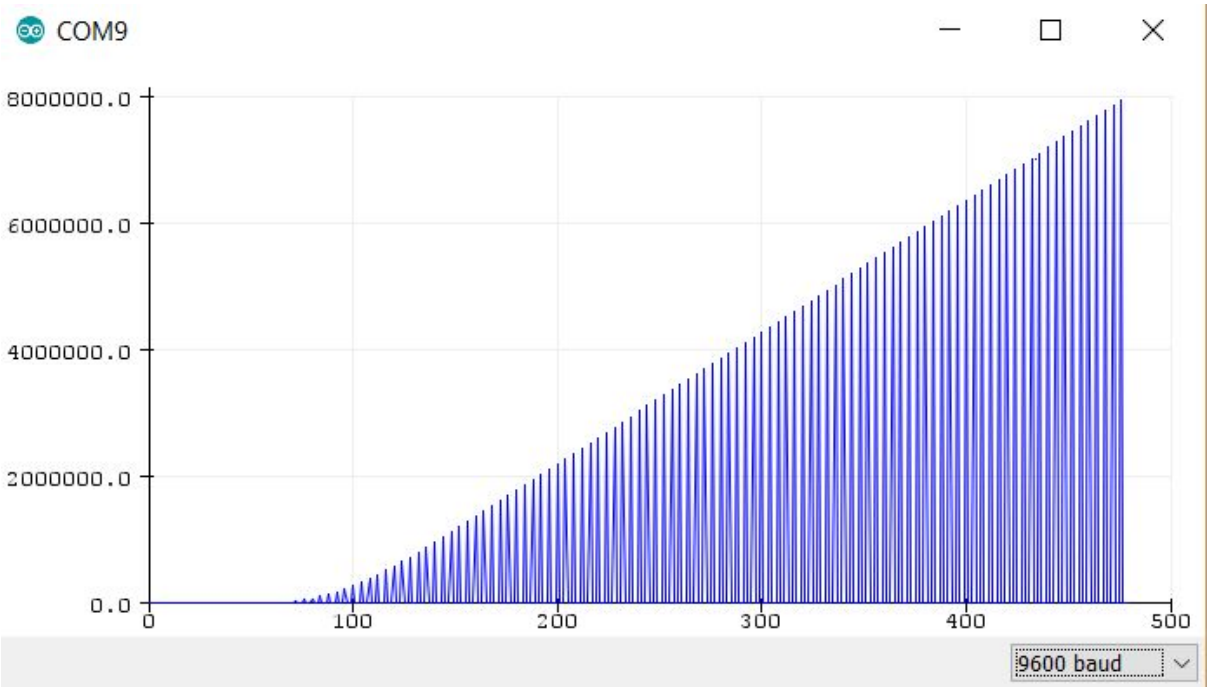
1.


COM9
300000.0
150000.0
0.0
-150000.0
-300000.0
0
100
200
300
400
500
9600 baud

2.


COM9
24000.0
18000.0
12000.0
6000.0
0.0
0
100
200
300
400
500
9600 baud

3.


COM9
8000000.0
6000000.0
4000000.0
2000000.0
0.0
0
100
200
300
400
500
9600 baud

APPENDIX

1. PART A1

```
#include <math.h>

double y_scale = 10;
double x_scale = 10;

void setup() {

  Serial.begin(9600);
}

void loop() {

    static int i = 0;
    int j;
    double y = 0;

    for(j = 0; j < 20; j += 1)
    {
        y += x(j)*h(i-j);
    }

    if (i < 80)
    {
        Serial.println(y);
```

```
      for (j = 0; j < 3; j += 1)
      {
         Serial.println(0);
      }
   }
   i += 1;
}


double heaviside (float t) {

   if (t < 0)
      return 0;

   else if (t == 0)
      return 0.5 * y_scale;

   else
      return 1 * y_scale;
}

double x (float t) {

   t = t / x_scale;
   return y_scale * exp(-t) * heaviside(t);

}
```

```
double h (float t) {

    t = t / x_scale;
    return y_scale * exp(-2*t) * heaviside(t);

}
```

2. PART A2

```
#include <math.h>


double y_scale = 10;
double x_scale = 10;
double pi = 3.14;

void setup() {

  Serial.begin(9600);
}

void loop() {

   static int i = 0;
   int j;
   double y = 0;

   for(j = 0; j < 20; j += 1)
   {
      y += x(j)*h(i-j);
   }

   if (i < 80)
   {
      Serial.println(y);
```

```
      for (j = 0; j < 5; j += 1)
      {
         Serial.println(0);
      }
   }
   i += 1;
}

double heaviside (float t) {

   if (t < 0)
      return 0;

   else if (t == 0)
      return 0.5 * y_scale;

   else
      return 1 * y_scale;
}

double x (float t) {

   t = t / x_scale;
   return y_scale * sin(pi * t) * (heaviside(t) -
heaviside(t-1.5));

}
```

```
double h (float t) {

    t = t / x_scale;
    return y_scale * (1.5 * (heaviside(t) -
heaviside(t-1.5)) - heaviside(t - 2) +
heaviside(t-2.5)) ;

}
```

3. PART B1

```
#include <math.h>


double y_scale = 10;
double x_scale = 10;
double pi = 3.14;

void setup() {

  Serial.begin(9600);
}

void loop() {

    static int i = 0;
```

```
    int j;
    double y = 0;

    for(j = 0; j < 20; j += 1)
    {
       y += x(j)*h(i-j);
    }

    if (i < 80)
    {
       Serial.println(y);

       for (j = 0; j < 5; j += 1)
       {
          Serial.println(0);
       }
    }
    i += 1;
}

double heaviside (float t) {

    if (t < 0)
       return 0;

    else if (t == 0)
       return 0.5 * y_scale;
```

```
    else
        return 1 * y_scale;
}

double x (float t) {

    t = t / x_scale;
    return y_scale * sin(pi * t) * (heaviside(t) -
heaviside(t-1.5));

}

double h (float t) {

    t = t / x_scale;
    return y_scale * (1.5 * (heaviside(t) -
heaviside(t-1.5)) - heaviside(t - 2) +
heaviside(t-2.5)) ;

}
```

4. PART B2

```
#include <math.h>


double y_scale = 10;
double x_scale = 10;
double pi = 3.14;

void setup() {

  Serial.begin(9600);
}

void loop() {

    static int i = 0;
    int j;
    double y = 0;

    for(j = 0; j < 25; j += 1)
    {
       y += x(j)*h(i-j);
    }

    if (i < 80)
    {
       Serial.println(y);
```

```
      for (j = 0; j < 5; j += 1)
      {
         Serial.println(0);
      }
   }
   i += 1;
}

double heaviside (float t) {

   if (t < 0)
      return 0;

   else if (t == 0)
      return 0.5 * y_scale;

   else
      return 1 * y_scale;
}

double x (float t) {

   t = t / x_scale;
   return y_scale * pow(0.8, t) * heaviside(t);

}
```

```
double h (float t) {

    t = t / x_scale;
    return y_scale * pow(0.3, t) * heaviside(t);

}
```

5. PART B3

```
#include <math.h>


double y_scale = 10;
double x_scale = 5;
double pi = 3.14;
double e = 2.71828;

void setup() {

  Serial.begin(9600);
}

void loop() {

    static int i = 0;
    int j;
    double y = 0;

    for(j = 0; j < 25; j += 1)
    {
        y += x(j)*h(i-j);
    }

    if (i < 80)
```

```
    {
      Serial.println(y);

      for (j = 0; j < 5; j += 1)
      {
        Serial.println(0);
      }
    }
    i += 1;
}

double heaviside (float t) {

    if (t < 0)
      return 0;

    else if (t == 0)
      return 0.5 * y_scale;

    else
      return 1 * y_scale;
}

double x (float t) {

    t = t / x_scale;
    return y_scale * pow(e, -t) * heaviside(t);
```

```
}

double h (float t) {

    t = t / x_scale;
    return y_scale * pow(2, -t) * heaviside(t);

}
```

6. PART C1

```
#include <math.h>


double y_scale = 10;
double x_scale = 10;


void setup() {

  Serial.begin(9600);
}

void loop() {

    static int i = -40;
    int j;
    double y = 0;

    for(j = 0; j < 25; j += 1)
    {
        y += x(j)*h(i+j);
    }

    if (i < 50)
    {
```

```
      Serial.println(y);

      for (j = 0; j < 5; j += 1)
      {
         Serial.println(0);
      }
   }
   i += 1;
}

double heaviside (float t) {

   if (t < 0)
      return 0;

   else if (t == 0)
      return 0.5 * y_scale;

   else
      return 1 * y_scale;
}

double x (float t) {

   t = t / x_scale;
   return y_scale * (heaviside(t)-heaviside(t-1));

}
```

```
double h (float t) {

    t = t / x_scale;
    return y_scale * t * (heaviside(t)-heaviside(t-1));

}
```

7. PART C2

```
#include <math.h>


double y_scale = 10;
double x_scale = 10;


void setup() {

  Serial.begin(9600);
}

void loop() {

    static int i = -40;
    int j;
    double y = 0;

    for(j = 0; j < 25; j += 1)
    {
       y += x(j)*h(i+j);
    }

    if (i < 50)
    {
       Serial.println(y);
```

```
      for (j = 0; j < 2; j += 1)
      {
         Serial.println(0);
      }
   }
   i += 1;
}

double heaviside (float t) {

   if (t < 0)
      return 0;

   else if (t == 0)
      return 0.5 * y_scale;

   else
      return 1 * y_scale;
}

double x (float t) {

   t = t / x_scale;
   return y_scale * exp(-t) * heaviside(t);

}
```

```
double h (float t) {

    t = t / x_scale;
    return y_scale * exp(-2*t) * heaviside(t);

}
```

8. PART C3

```
#include <math.h>


double y_scale = 10;
double x_scale = 10;


void setup() {

  Serial.begin(9600);
}

void loop() {

    static int i = -40;
    int j;
    double y = 0;

    for(j = 0; j < 25; j += 1)
    {
       y += x(j)*x(i+j);
    }

    if (i < 50)
    {
       Serial.println(y);
```

```
      for (j = 0; j < 3; j += 1)
      {
         Serial.println(0);
      }
   }
   i += 1;
}

double heaviside (float t) {

   if (t < 0)
      return 0;

   else if (t == 0)
      return 0.5 * y_scale;

   else
      return 1 * y_scale;
}

double x (float t) {

   t = t / x_scale;
   return y_scale * (heaviside(t)-heaviside(t-1));

}
```

9.PART D1

```
#include <math.h>


double y_scale = 10;
double x_scale = 10;
double pi = 3.14;

void setup() {

  Serial.begin(9600);
}

void loop() {

    static int i = -40;
    int j;
    double y = 0;

    for(j = 0; j < 25; j += 1)
    {
        y += x(j)*h(i+j);
    }

    if (i < 80)
    {
        Serial.println(y);
```

```
      for (j = 0; j < 3; j += 1)
      {
         Serial.println(0);
      }
   }
   i += 1;
}

double heaviside (float t) {

   if (t < 0)
      return 0;

   else if (t == 0)
      return 0.5 * y_scale;

   else
      return 1 * y_scale;
}

double x (float t) {

   t = t / x_scale;
   return y_scale * (heaviside(t)-heaviside(t-8));

}
```

```
double h (float t) {

   t = t / x_scale;
   return y_scale * sin(2 * pi * t / 8) *
(heaviside(t)-heaviside(t-8));

}
```

10. PART D2

```
#include <math.h>


double y_scale = 10;
double x_scale = 6;
double pi = 3.14;

void setup() {

  Serial.begin(9600);
}

void loop() {

   static int i = -40;
   int j;
   double y = 0;
```

```
    for(j = 0; j < 25; j += 1)
    {
       y += x(j)*h(i+j);
    }

    if (i < 80)
    {
       Serial.println(y);

       for (j = 0; j < 3; j += 1)
       {
          Serial.println(0);
       }
    }
    i += 1;
}

double heaviside (float t) {

    if (t < 0)
       return 0;

    else if (t == 0)
       return 0.5 * y_scale;

    else
       return 1 * y_scale;
}
```

```
double x (float t) {

   t = t / x_scale;
   return y_scale * pow(0.2, t) * heaviside(t);

}

double h (float t) {

   t = t / x_scale;
   return y_scale * pow(0.4, t) * heaviside(t);

}
```

11. PART D3

```
#include <math.h>


double y_scale = 10;
double x_scale = 6;
double pi = 3.14;

void setup() {

  Serial.begin(9600);
}
```

```
void loop() {

   static int i = -40;
   int j;
   double y = 0;

   for(j = 0; j < 25; j += 1)
   {
      y += x(j) * x(i+j);
   }

   if (i < 80)
   {
      Serial.println(y);

      for (j = 0; j < 3; j += 1)
      {
         Serial.println(0);
      }
   }
   i += 1;
}

double heaviside (float t) {

   if (t < 0)
      return 0;
```

```
    else if (t == 0)
        return 0.5 * y_scale;

    else
        return 1 * y_scale;
}

double x (float t) {

    t = t / x_scale;
    return y_scale * t * heaviside(t);

}
```