

วิชา Data Communication Laboratory
ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

การทดลองที่ 6 การสร้างกราฟสัญญาณรูปแบบต่างๆ

วัตถุประสงค์

1. ศึกษาการใช้งานโปรแกรม MATLAB เป็นต้น
2. เข้าใจการทำงานของคำสั่ง MATLAB เป็นต้น
3. สามารถสร้างสัญญาณด้วยคำสั่ง MATLAB
4. ศึกษาการเขียนโปรแกรมบนบอร์ด Arduino
5. ทดลองการสร้างสัญญาณด้วยการเขียนโปรแกรมสั่งงานจากบอร์ด Arduino

MATLAB เป็นต้น

MATLAB ย่อมาจาก MATrix LABoratory โดยพัฒนาโดย The Mathworks, Inc เป็นโปรแกรมที่ใช้ในการทำงานทางวิศวกรรม และวิทยาศาสตร์ เนื่องจากมีฟังก์ชันทางคณิตศาสตร์ให้เลือกใช้หลากหลาย และมีการพัฒนาอยู่เสมอ สามารถใช้ทดสอบ และนำมาพัฒนาอัลกอริทึมต่อได้สะดวกรวดเร็ว ทั้งยังใช้ประมวลผล ร่วมกับโปรแกรมอื่นได้ เช่น Fortran, Borland C/C++, Microsoft Visual C++ เป็นต้น ทั้งบังสามารถนำไปใช้งาน ด้านกราฟิก แสดงผลได้ทั้งภาพสองมิติ ภาพสามมิติ นำภาพมาต่อ กันเพื่อแสดงภาพเคลื่อนไหว จุดเด่นที่สำคัญคือ MATLAB เป็นระบบ Interactive คือ ข้อมูลพื้นฐานส่วนใหญ่เป็น Matrix หรือ Array ที่ไม่ต้องกำหนดขนาด หรือ มิติเหมือนโปรแกรมภาษาอื่น และมีส่วนที่เป็นโครงสร้างแบบจำลอง (Simulink) ให้ใช้งานสำหรับจำลองระบบ ต่างๆ

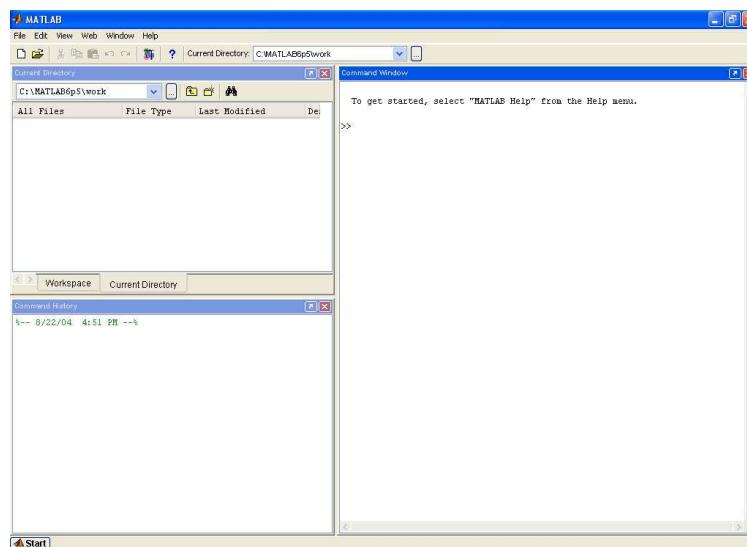
การใช้งานโปรแกรม MATLAB เป็นต้น

หลังจากเปิดโปรแกรม MATLAB จะมีหน้าต่างซึ่งประกอบด้วยส่วนต่างๆ ได้แก่ Current Directory, Command History, Command Windows และอาจมีหน้าต่างอื่น ดังรูปที่ 6.1 ในส่วนของการทดลองนี้จะเริ่มกล่าวถึง การใช้งานส่วนที่เป็น Command Windows ซึ่งเป็นส่วนที่สั่งงานหลัก เพื่อการทำการคำนวณหรือเรียกใช้ฟังก์ชัน จะมีลักษณะเป็นส่วนต่อประสานแบบชุดคำสั่ง (Command Line Interface) โดยโปรแกรม MATLAB จะพร้อมใช้งาน เมื่อมีเครื่องหมาย prompt ">>" ขึ้นมาสำหรับทำการคำนวณ หรือเรียกใช้คำสั่ง หรือใช้งานฟังก์ชัน

การคำนวณ

สามารถทำการคำนวณโดยใส่ค่าต่างๆ และเครื่องหมาย สัญลักษณ์สำหรับการกระทำทางคณิตศาสตร์ เป็นต้น ได้แก่ + บวก, - ลบ, * คูณ, / หาร, ^ ยกกำลัง ฯลฯ ได้ทันที

```
>> 19+1*29+1
ans =
    49
```



รูปที่ 6.1 แสดงโปรแกรม MATLAB

การใช้งานตัวแปร

การกำหนดตัวแปร ไม่จำเป็นต้องมีการประกาศตัวแปร และกำหนดชนิดของตัวแปร สามารถทำการกำหนดค่า หรือตัวแปรต่างๆ ได้โดยกำหนดให้ตัวแปรมีค่า (และขนาดต่างๆ) ด้วยเครื่องหมาย “=” โดยการกำหนดชื่อตัวแปรจะต้องขึ้นต้นด้วยตัวอักษร และไม่ซ้ำกับคำเฉพาะ (ชื่อฟังก์ชันอื่น) โดยโปรแกรม MATLAB จะตีความหมายของตัวแปรที่เป็นตัวอักษรตัวใหญ่และตัวเล็กจะไม่เหมือนกัน (Case Sensitive) และสามารถเปลี่ยนชนิดตัวแปรโดยกำหนดค่าให้ใหม่ได้ทันที

```
>> ong = 19+1*29+1
ong =
    49
>> ong = 'Jirasak'
ong =
Jirasak
```

คำสั่ง who

who เป็นคำสั่งเพื่อใช้ตรวจสอบว่ามีการใช้งานตัวแปรใดบ้าง (สามารถดูรายละเอียดของตัวแปรได้จากหน้าต่าง Workspace)

```
>> A = 2^2
A =
    4
>> a = 3^2
a =
    9
>> b = 4^2
b =
    16
>> B = 5^2
B =
    25
>> who
Your variables are:
A B a b
```

Name	Value	Min	Max
A	4	4	4
B	25	25	25
a	9	9	9
b	16	16	16

คำสั่ง clear

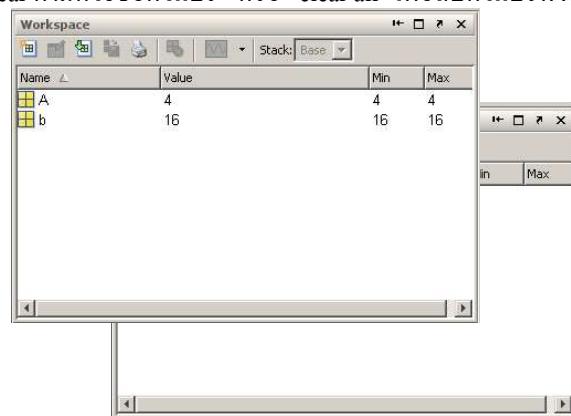
เป็นคำสั่งเพื่อใช้ลบตัวแปรออก โดยสั่ง “clear ตามด้วยชื่อตัวแปร” หรือ “clear all” เพื่อลบตัวแปรทั้งหมดที่ใช้อยู่

```
>> who
Your variables are:
A B a b
>> clear a B
>> who
Your variables are:
A b
>> clear all
>> who
>>
```

การใช้เครื่องหมายเชมิโคลอน “;”

การใช้เครื่องหมาย “;” ตามหลังคำสั่งต่างๆ เพื่อกำหนดให้โปรแกรม MATLAB ไม่แสดงผลการทำงาน

```
>> a = 10;
>> A = 19
A =
    19
>> who
Your variables are:
A b
```



ตัวแปร Vectors และ Matrices

การกำหนดตัวแปร Vectors และ Matrices เหมือนการกำหนดตัวแปรชาร์มดา แต่จะเพิ่มการกำหนดในลำดับ และถ้า โดยใช้เครื่องหมาย คอมมา “,” หรือเว้นวรรค (Space Bar) เพื่อกำหนดลำดับในแล้ว และใช้เครื่องหมายเชมิโคลอน “;” ในการแบ่งระหว่างแล้ว

การกำหนดแถว

```
>> row1 = [1 2 3 4]
row1 =
    1     2     3     4
>> row2 = [1, 2, 3, 4]
row2 =
    1     2     3     4
```

การกำหนดหลัก

```
>> column = [1; 2; 3]
column =
    1
    2
    3
```

การกำหนดทั้งค่าตัวและถ้า Vs การสั่งสับเปลี่ยนระหว่าง ถ้า และ หลัก

```
>> x=[11 12 13 14; 21 22 23 24; 31 32 33 34]
x =
    11     12     13     14
    21     22     23     24
    31     32     33     34
```

```
>> y = x'  
y =  
11    21    31  
12    22    32  
13    23    33  
14    24    34
```

การกำหนดตัวแปร Vectors และ Matrices ที่เป็นเชิงเส้น และการอ้างอิงค่าที่ตำแหน่งต่างๆ

การกำหนดตัวแปร Vectors และ Matrices ที่เป็นเชิงเส้น โดยการใช้เครื่องหมาย โคลอน ":"

การกำหนดตัวแปรที่เพิ่มค่าทีละ 1

โดยสามารถกำหนดโดยใช้เครื่องหมาย ":" ระหว่างค่าเริ่มต้นและค่าสุดท้ายในเครื่องหมาย []

```
>> a = [1:9]  
a =  
1    2    3    4    5    6    7    8    9  
>> b = [-9:-1]  
b =  
-9   -8   -7   -6   -5   -4   -3   -2   -1
```

การกำหนดตัวแปรที่เพิ่มเป็นค่าคงที่

โดยสามารถกำหนดได้ดังนี้ [ค่าเริ่มต้น : ค่าที่เปลี่ยนแปลงไป : ค่าสุดท้าย]

```
>> test1 = [1:2:9]  
test1 =  
1    3    5    7    9  
>> test2 = [9:-2:-5]  
test2 =  
9    7    5    3    1    -1   -3   -5
```

การอ้างอิงค่าที่ตำแหน่งต่างๆ

```
>> ooo = [0:5:37]  
ooo =  
0    5    10   15   20   25   30   35  
>> ooo(4)  
ans =  
15  
>> test = [0:11:100;1:11:100]  
test =  
0    11   22   33   44   55   66   77   88   99  
1    12   23   34   45   56   67   78   89   100  
>> test(2,4)  
ans =  
34  
>> test(1,[2:5])  
ans =  
11    22    33    44
```

การใช้เครื่องหมายสำคัญ Vector และ Matrix

เครื่องหมาย + , -

การบวก หรือ ลบ เมตริกซ์จะใช้เครื่องหมาย “+” หรือ “-” ซึ่งจะกระทำการบวกกัน (หรือลบกัน) เมตริกซ์ธรรมดาก็จะนำค่าแต่ละตัวมาทำการบวกกัน (หรือลบกัน)

```
>> a = [5:2:15]
a =
      5     7     9    11    13    15
>> b = [5:-1:0]
b =
      5     4     3     2     1     0
>> a+b
ans =
    10    11    12    13    14    15
>> b-a
ans =
      0    -3    -6    -9   -12   -15
```

เครื่องหมาย *, /

การคูณ หรือ หาร เมตริกซ์จะใช้เครื่องหมาย “*” หรือ “/” ซึ่งจะกระทำการคูณ (หรือหาร) เมตริกซ์ ซึ่งจากตัวอย่างที่ผ่านมาไม่สามารถใช้คำสั่ง $a*b$ ได้เนื่องจากคูณเมตริก จำนวนหลักของตัวตั้งต้องเท่ากับจำนวนแถวของตัวคูณ จึงต้องใช้คำสั่ง $a * b'$ (เข่นเดียวกันไม่สามารถใช้คำสั่ง a/b' ได้เช่นกัน)

```
>> a = [5:2:15]; b = [5:-1:0];
>> a * b'
ans =
    115
>> a / b
ans =
    2.0909
```

เครื่องหมาย .* , ./

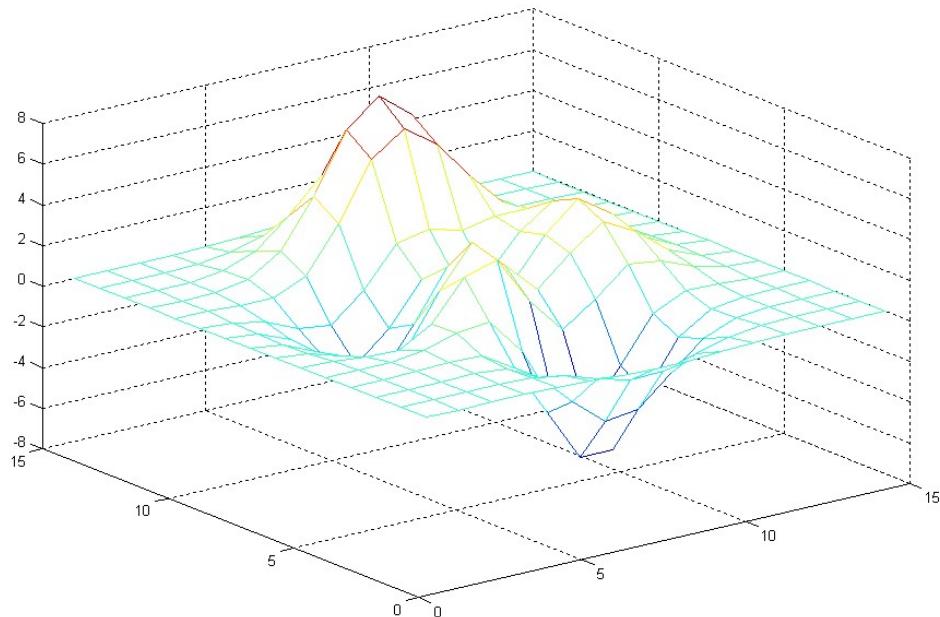
เป็นการสั่งให้เมตริก 2 เมตริกทำการคูณ หรือหารที่ตำแหน่งเดียวกัน (แล้วหลักเดียวกัน คูณหรือการหาร)

```
>> a = [5:2:15]; b = [5:-1:0];
>> a .* b
ans =
    25     28     27     22     13     0
>> b ./ a
ans =
    1.0000    0.5714    0.3333    0.1818    0.0769    0
```

การวาดกราฟ

กราฟกล่าวได้ว่าเป็นสิ่งที่สำคัญในการวิเคราะห์ และแสดงผลของโปรแกรม MATLAB โดยสามารถวาดกราฟได้หลายชนิดทั้งสองมิติ และสามมิติ

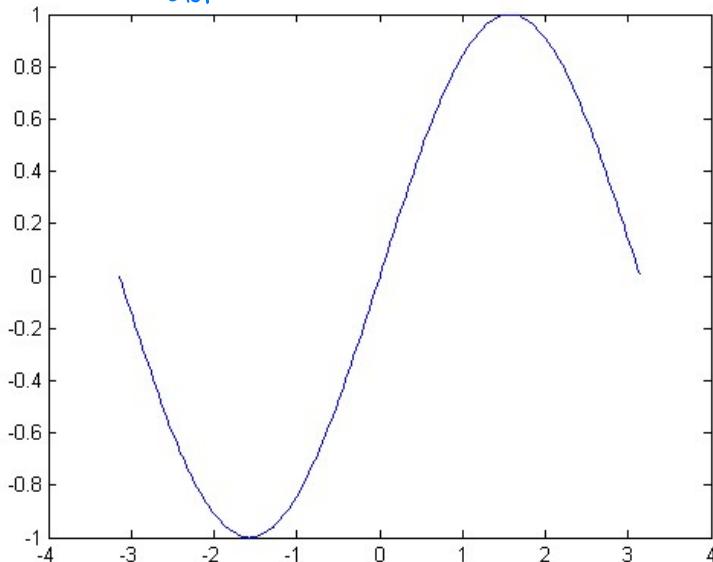
```
>> mesh(peaks(15))
```



รูปที่ 6.2 แสดงการวาดกราฟจากคำสั่ง `mesh(peaks(15))`

```
>> x = [-pi : pi/100 : pi];
>> y = sin(x);
>> plot(x,y);
```

start step stop



รูปที่ 6.3 แสดงการวาดกราฟจากคำสั่ง `plot(x,y)`;

ตารางที่ 6.1 รูปแบบเส้น สัญลักษณ์ และสี สำหรับการวาดกราฟ

สีที่แสดง		สัญลักษณ์ที่ใช้		รูปแบบของเส้น	
สัญลักษณ์ที่ใช้	สี	สัญลักษณ์ที่ใช้	เครื่องหมาย	สัญลักษณ์ที่ใช้	รูปแบบของเส้น
R	red	.	:	.	เส้นจุด
G	green	○	วงกลม	-	เส้นทึบ
B	blue	+	บวก	-.	เส้นประและจุด
C	cyan	×	kakunath	--	เส้นประ
m	magenta	*	ดอกจัน		
y	yellow	s	สีเหลี่ยม		
k	black	d	ข้าวหลามตัด		
w	write	v	สามเหลี่ยมล่าง		
		^	สามเหลี่ยมนบน		
		>	สามเหลี่ยมขวา		
		<	สามเหลี่ยมซ้าย		

ตารางที่ 6.2 คำสั่งในรูปแบบการวาดกราฟสองมิติ

คำสั่ง	การทำงาน
plot (x, y)	วาดกราฟของ x, y
plot (x, y, x, z)	วาดกราฟของ x, y และ วาดกราฟของ x, z ในแกน x เดียวกัน
subplot (m, n, p)	แบ่งหน้าต่างรูปภาพเป็นหน้าต่างย่อย m และ n หลัก และใช้ p เป็นลำดับการวาดกราฟ
figure (n)	สร้างหน้าต่างรูปภาพหมายเลข n
clf	ลบรูปภาพบนหน้าต่าง
close (n)	ลบหน้าต่างรูปภาพหมายเลข n
close all	ลบหน้าต่างรูปภาพทั้งหมด

ตารางที่ 6.3 คำสั่งการแบ่งเขต

คำสั่ง	การทำงาน
axis([xmain xmax ymain ymax])	กำหนดค่าสูงสุด และค่าต่ำสุดในแกน x และแกน y
axis([xmain xmax ymain ymax zmin zmax])	กำหนดค่าสูงสุด และค่าต่ำสุดในแกน x แกน y และแกน z
X = axis	การหาค่าสูงสุด และค่าต่ำสุด ในแกนต่างๆ
grid on	การเพิ่มเส้น grid ให้แกนปั๊กจุบัน
grid off	ยกเลิกเส้น grid ให้แกนปั๊กจุบัน

ตารางที่ 6.4 คำสั่งการแบ่งเสกต

คำสั่ง	การทำงาน
title ('????')	ตั้งชื่อหัวเรื่องกราฟ
xlabel ('????')	เขียนข้อความที่แกน x
ylabel ('????')	เขียนข้อความที่แกน y
zlabel ('????')	เขียนข้อความที่แกน z
gtext ('????')	เขียนข้อความที่ตำแหน่งมาส์ชี
text ('????')	เขียนข้อความที่ตำแหน่ง x y

ตารางที่ 6.5 คำสั่งการวาดกราฟเส้น 3 มิติ

คำสั่ง	การทำงาน
plot3 (x, y, z)	การเขียนกราฟ 3 มิติ
plot3 (x1, y1, z1, s1, x2, y2, z2, s2, x3, y3, z3, s3)	

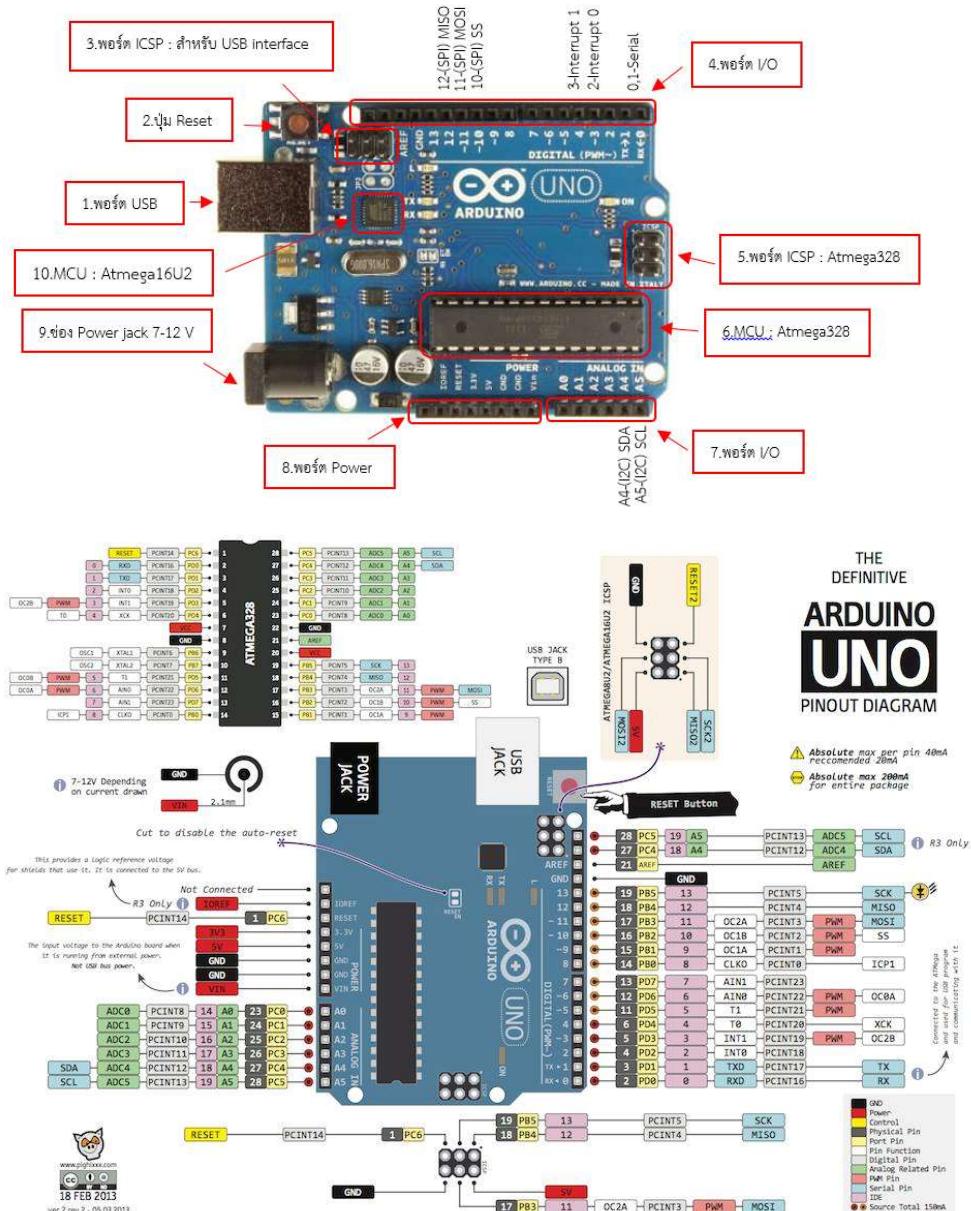
สามารถศึกษาการวาดกราฟอื่นได้ โดยการใช้ Help คุณคำสั่งต่างๆ เช่น contour, contour3, pcolor (การวาดกราฟเส้น โครงร่าง), mesh, meshc, meshz (การวาดกราฟพื้นผิวร่างแท้), surf, surfc (การวาดกราฟพื้นผิว), ฯลฯ

การเขียนโปรแกรมลงบน Arduino

Arduino อ่านว่า (อา-ดู-อี-โน่ หรือ อะดิยโน่) เป็นบอร์ดไมโครคอนโทรเลอร์ระดับ AVR ที่มีการพัฒนาแบบ Open Source คือ มีการเปิดเผยข้อมูลทั้งด้าน Hardware และ Software โดยสามารถใช้งานได้ง่ายและสามารถใช้ในติดต่อสื่อสารกับอุปกรณ์ภายนอกผ่านทางขา I/O ของบอร์ดได้ง่าย โดยขา I/O ของ บอร์ด ประกอบด้วย (โครงสร้างบอร์ด Arduino UNO R3 แสดงในรูปที่ 6.4)

1. ขาอิงรับสัญญาณ Analog (A0-A5) จำนวน 6 ขา
2. ขาอิงรับสัญญาณ Digital (0-13) จำนวน 14 ขา ซึ่งบางขาสามารถโปรแกรมให้สร้างสัญญาณ Pulse Width Modulation (PWM) (3, 5-6, 9-11) จำนวน 6 ขา

การเขียนโปรแกรมควบคุมการทำงานของบอร์ด Arduino สามารถเขียนผ่านทาง Arduino IDE ภาษาที่ใช้จะเป็นภาษา C ที่มีโครงสร้างการเขียนพื้นฐานที่แตกต่างจากภาษา C ทั่วไป คือ จะมีการทำงานแบบวนทำซ้ำ (loop) ตัวแต่โปรแกรมบรรทัดแรก จนถึงบรรทัดสุดท้าย ใน 100% หลังจากนั้นจะวนกลับมาเริ่มทำงานบรรทัดแรกอีกครั้ง



รุ่นที่ 6.4 โครงสร้างบอร์ด Arduino UNO R3 (<http://www.mvarduino.net>)

ໂຄຮງສ້າງໂໄຣແກຣມຈະແປ່ງເປົ້ານ 2 ສ່ວນ ໄກສອນເຊົ້າວາ

1. ส่วนหัวโปรแกรม ซึ่งคล้ายกับภาษา C ทั่วไป ประกอบด้วย พรีโปรดักชันเรติฟ (Preprocessing Directives) ตัวแปรชนิดโภบล (Global Variable) เป็นต้น ซึ่งในส่วนนี้Arduino พื้นฐาน หากไม่ใช้ก็ไม่จำเป็นต้องมี
 2. ส่วนตัวโปรแกรมต้องประกอบด้วยฟังก์ชันหลัก 2 ฟังก์ชัน ได้แก่
 - 1) ส่วน `setup()` เป็นส่วนตั้งค่าการทำงาน เช่น ตั้งค่าการสื่อสารระหว่างบอร์ด Arduino กับคอมพิวเตอร์ ผ่านทาง Serial communication หรือ ตั้งค่ารูปแบบของขา I/O เป็นต้น
 - 2) ส่วน `loop()` เป็นส่วนการทำงานของโปรแกรมจริงที่วนทำซ้ำตลอด

คำสั่งเบื้องต้นในการเขียนโปรแกรมเพื่อติดต่อกับ I/O

1. **pinMode(pin, mode)** ใช้กำหนด mode การทำงานให้กับขา I/O ที่ต้องการใช้
 - เมื่อ pin : หมายเลข I/O ของบอร์ดที่ต้องการกำหนด
 - mode : เลือกเป็น INPUT, OUTPUT หรือ INPUT_PULLUP
2. **digitalWrite(pin, logic)** ใช้ส่งข้อมูล Digital ไปยังขา I/O ที่ต้องการ
 - เมื่อ pin : หมายเลข I/O ของบอร์ดที่ต้องการใช้ส่งข้อมูล
 - logic : ข้อมูลที่ต้องการส่ง HIGH หรือ LOW
3. **digitalRead(pin)** ใช้อ่านค่าข้อมูล Digital จากขา I/O ที่ต้องการ
 - เมื่อ pin : หมายเลข I/O ของบอร์ดที่ต้องการใช้รับข้อมูลหลังจาก Return ค่า โดยมีสถานะข้อมูล เป็น HIGH หรือ LOW
4. **analogWrite(pin, duty)** ใช้ส่งข้อมูล Analog PWM ไปยังขา I/O ที่ต้องการ
 - เมื่อ pin : หมายเลข I/O ของบอร์ดที่ต้องการใช้ส่งข้อมูล
 - duty : ความกว้างของ Active pulse (PWM duty cycle) ตั้งแต่ 0 – 255 (8 bits)
 - หมายเหตุ ความถี่ของสัญญาณ PWM ของขา 3, 9-11 ประมาณ 490 Hz
 - ความถี่ของสัญญาณ PWM ของขา 5-6 ประมาณ 980 Hz
5. **analogRead(pin)** ใช้อ่านค่าข้อมูล Analog จากขา I/O ที่ต้องการ
 - เมื่อ pin : หมายเลข I/O ของบอร์ดที่ต้องการใช้รับข้อมูลหลังจาก Return ค่า
 - หมายเหตุ เนื่องจากส่วนแปลงสัญญาณจาก Analog-to-Digital (ADC) ของบอร์ด Arduino UNO R3 มีความเร็วที่ไม่สูงมาก อยู่ที่ประมาณ 80 KHz (เมื่อไม่มีคำสั่งอื่นๆ) และ มีความละเอียดอยู่ที่ 0-1024 (10 bits) รับสัญญาณ Analog อยู่ในช่วง 0-5 Volt (ไม่สามารถให้สัญญาณติดลบได้) ทำให้ความละเอียดของสัญญาณที่อ่านได้ อยู่ที่ 5V / 1024 หน่วย หรือ 0.0049V / หน่วย โดยความละเอียดนี้สามารถปรับเปลี่ยนได้โดยใช้คำสั่ง analogReference()
 - บอร์ด Arduino จะใช้วิวัฒนาการอ่านข้อมูลสูงสุดได้เพียง 10,000 ครั้ง / วินาที เท่านั้น
 - ถ้าขา Analog ไม่มีการเชื่อมต่อกับอุปกรณ์ใด จะทำให้ค่าที่อ่านได้จากคำสั่ง analogRead() เป็นค่าไม่แน่นอน
6. **analogReference(Type)** ใช้ในการตั้งค่า Voltage อ้างอิงสูงสุดของ Analog input เมื่อกำหนดค่า Type เป็น
 - DEFAULT : จะใช้ค่า 5V สำหรับบอร์ดที่จ่ายไฟ 5V
 - และ 3.3V สำหรับบอร์ดที่จ่ายไฟ 3.3V
 - INTERNAL : จะใช้ค่าอ้างอิงภายในบอร์ด 1.1V สำหรับ ATmega168 / ATmega328
 - และ 2.56V volts สำหรับ ATmega8 (ไม่ใช้กับบอร์ด Arduino Mega)
 - INTERNAL1V1 : จะใช้ค่าอ้างอิงภายในบอร์ด 1.1V สำหรับบอร์ด Arduino Mega เท่านั้น
 - INTERNAL2V56 : จะใช้ค่าอ้างอิงภายในบอร์ด 2.56V สำหรับบอร์ด Arduino Mega เท่านั้น
 - EXTERNAL : จะใช้ค่าอ้างอิงภายนอกบอร์ดที่รับมาจาก AREF (0 - 5V)

หมายเหตุ ถ้าเลือกใช้ Type = EXTERNAL จากขา AREF ต้องตั้งค่า ก่อนเรียกใช้ analogRead() มิฉะนั้นจะทำให้เกิดการเชื่อม Active Reference Voltage (Internal Reference) กับ External Reference (AREF) ทำให้

Microcontroller ของบอร์ดพังได้ เพื่อป้องการการตั้งค่าผิดพลาด สามารถต่อเข้ามอเตอร์ด้วยหัวต้านทาน 5K ที่ขา AREF เพื่อลดผลกระทบได้

7. **delay(time)** ใช้ในการหยุดโปรแกรมตามเวลาที่กำหนด โดยวัดเป็น Milliseconds (unsigned long)
8. **millis()** จะให้ค่าเวลาหน่วย Miliseconds (unsigned long) ตั้งแต่บอร์ดเริ่มรัน โปรแกรมปัจจุบัน Description ชี้ว่าจะวนกับไปที่ 0 หลังจากเวลาผ่านไปประมาณ 50 วัน
9. **sin(rad)** ให้ค่า sine ของมุมที่กำหนด (rad) หน่วยเป็น Radian ซึ่งมีค่าอยู่ระหว่าง -1 ถึง 1

การทดลองที่ 6.1 การใช้คำสั่ง MATLAB บน Command Windows และ บน Editor (M-File)

1. ให้นักศึกษาพิมพ์คำสั่งต่อไปนี้บน Command Windows แล้วอธิบายผลที่ได้

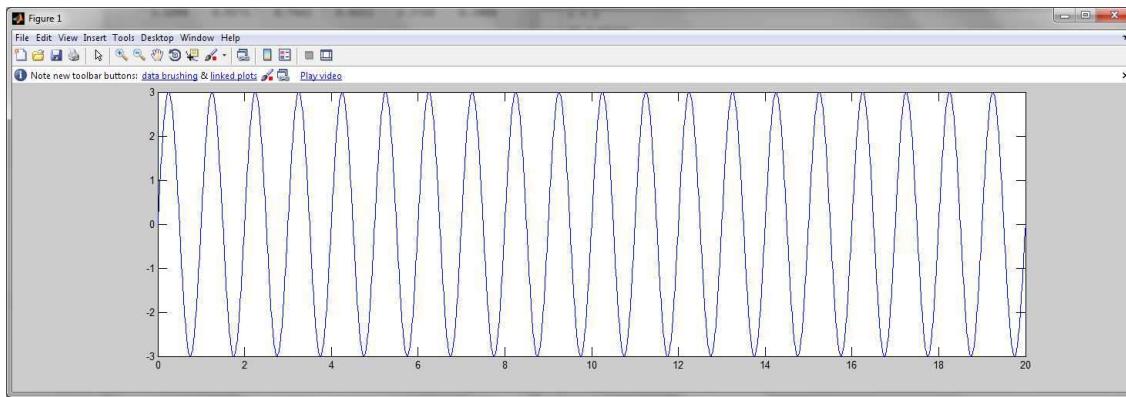
```
>> clear all; % ลบสัมภาระ
>> close all; % ปิดหน้าจอ
>> A = [1:10] % สร้าง vector [1,...,10]
>> B = [1:100]/10
>> C = [1:0.1:10]
>> D = [0.1:0.1:10]
```

2. เลือก File => New => Blank M-File หลักจากนั้นให้นักศึกษาพิมพ์คำสั่งต่อไปนี้บน Editor (M-File) แล้วเลือก Debug => Run ทดสอบการทำงาน พร้อมอธิบายผลที่ได้

```
clear all;
close all;
x = [0:10]
m = -2
c = 3
y1 = m*x+c
y2 = c.^x
figure (1)
plot (x,y1)
figure (2)
plot (x,y1)
figure (3)
subplot(2,1,1), plot (x,y1) % แก้
subplot(2,1,2), plot (x,y2) % แก้
figure (4)
subplot(1,2,1), plot (x,y1)
subplot(1,2,2), plot (x,y2)
```

3. ให้นักศึกษาทดลองสร้างสัญญาณ Sine Wave ที่ความถี่ 1 Hz ขนาด 3 Volt ตามค่าเวลา t = [0:20] แล้วแสดงผลกราฟสัญญาณด้วยคำสั่ง plot บน Editor (M-File)

-
-
-
-
-
-
-
4. ให้แนกศึกษาพิมพ์คำสั่งบน Command Windows (!!เก็บคำสั่งในข้อ 3.) เพื่อให้ได้ผลดังรูปที่ 6.5



รูปที่ 6.5 สัญญาณ $3 \sin(2\pi t)$

-
-
-
-
-
-
-
5. เขียนอาจารย์ตรวจสอบการทดลอง

.....
ลายเซ็นอาจารย์ผู้ตรวจสอบการทดลอง

การทดลองที่ 6.2 การสร้าง Pulse Wave บน Arduino

1. ให้นักศึกษาเขียนโปรแกรมบนบอร์ด Arduino UNO R3 ดังนี้
 - 1.1. ส่วนหัวโปรแกรมให้กำหนดเปรี้ยงนี้
 - 1) zeta เป็น Array ชนิด float เก็บค่ามุม 4 มุม ได้แก่ ($0^\circ, 90^\circ, 180^\circ, 270^\circ$)
 - 2) S เป็น Array ชนิด float สำหรับเก็บค่าสัญญาณ Sine ของมุมต่างๆ
 - 3) pwmDuty เป็น Array ชนิด uint16_t สำหรับเก็บค่า Duty Cycle ของสัญญาณ PWM
 - 1.2. ส่วนฟังก์ชัน setup()
 - 1) ให้ตั้งค่าขา Digital ที่ต้องการสร้างสัญญาณ Pulse Width Modulation (PWM) โดยสามารถเลือก pin_number (3, 5-6, 9-11) ด้วยคำสั่งรูปดังนี้

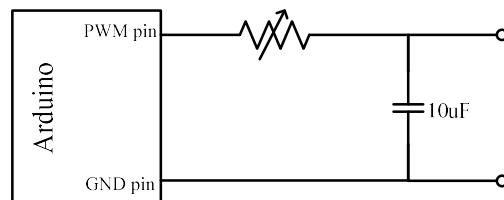

```
Serial.begin(115200);
pinMode(pin number, OUTPUT);
```
 - 2) ทำการคำนวณค่า S[i] และ pwmDuty[i] เพื่อนำไปใช้ต่อในฟังก์ชัน loop() โดยที่
 - o ให้ S[i] เก็บค่าจากการคำนวณค่า Sine ของมุมต่างๆ ได้แก่ $0^\circ, 90^\circ, 180^\circ$ และ 270°
 - o ให้ pwmDuty[i] เก็บค่า Duty Cycle ของ สัญญาณ PWM Pulse ($0 \leq \text{Duty Cycle} \leq 255$) ตามค่า Sine ที่ได้จาก S[i]
 - o Debug ค่า S[i] และ pwmDuty[i] แต่ละลำดับด้วยคำสั่ง Serial.println(); บันทึกค่าที่สัมพันธ์กับ zeta ต่างๆ

-
-
-
-
-

- 1.3. ส่วนฟังก์ชัน loop() ให้เขียนโปรแกรมเพื่อวนแสดงค่า PWM Pulse จากข้อ 1.2 ทีละชุดด้วยคำสั่ง

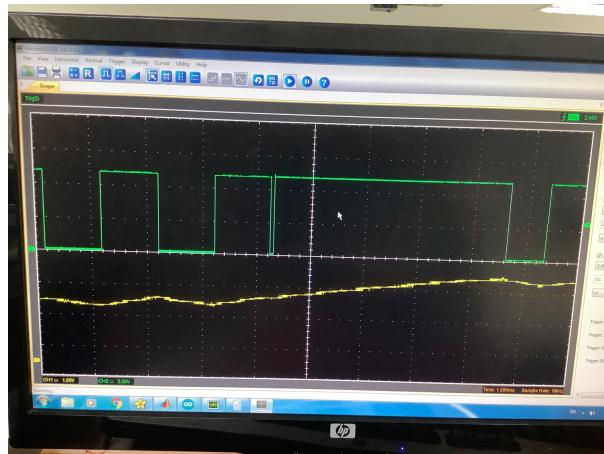
```
for(int i=0; i<4; i++){
    analogWrite(pin number, pwmDuty[i]);
    delayMicroseconds(4000);
}
```

2. ต่อวงจร Low Pass Filter เพื่อกรองสัญญาณความถี่ต่ำ โดยด้วยตัวต้านทานปรับค่าได้ $10\text{K}\Omega$ (ต้องบัดกรีสายไฟที่ขาตัวต้านทานปรับค่า) และ ตัวเก็บประจุขนาด 10\mu F ดังรูปที่ 6.6 แล้วใช้ Oscilloscope Ch1 วัดสัญญาณตอกคร่อม C กับ Ground และ Ch2 วัดสัญญาณตอกคร่อม PWM กับ Ground ปรับค่าตัวต้านทานปรับค่า เพื่อให้สัญญาณที่ Ch1 ใกล้เคียง Sine Wave มากที่สุด

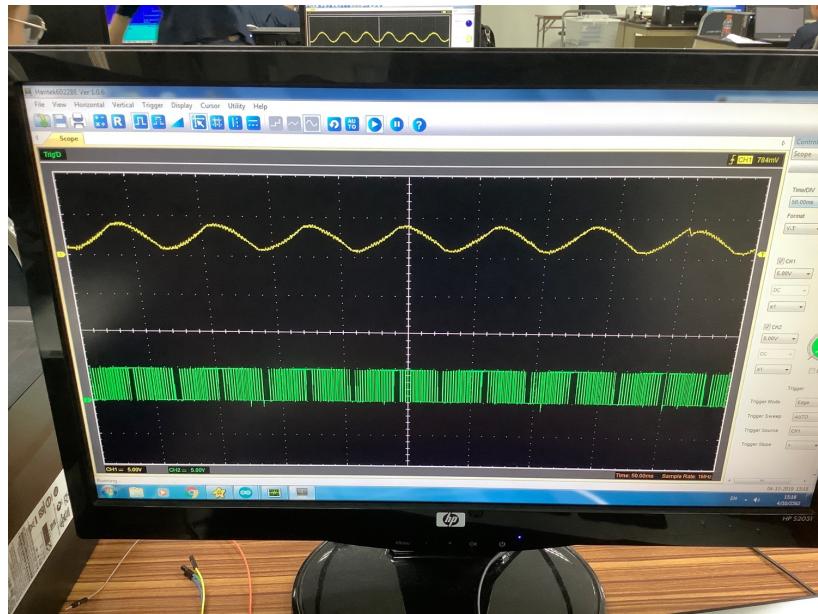


รูปที่ 6.6 วงจร Low Pass Filter

3. ให้นักศึกษาครุ่น功ร่างของสัญญาณและเปรียบเทียบความแตกต่างของสัญญาณที่ได้จากช่อง 1 (Ch2) และ ช่อง 2 (Ch1)



4. ทำการทดลองเพิ่มเติม โดยปรับเปลี่ยนโปรแกรมให้เพิ่มจำนวนมุ่งที่คำนวณค่าองค์ Sine เป็นจำนวน 8 และ 16 มุ่ง และคำนวณค่าของ PWM ที่สัมพันธ์กัน
 5. ให้นักศึกษาครุ่น功ร่างของสัญญาณ และเปรียบเทียบความแตกต่างของสัญญาณที่ตอกคร่อม C เมื่อจำนวนมุ่งเป็น 4, 8 และ 16 มุ่ง



6. เซี่ยงอาจารย์ตรวจสอบการทดลอง

ลายเซ็นอาจารย์ผู้ตรวจการทดลอง

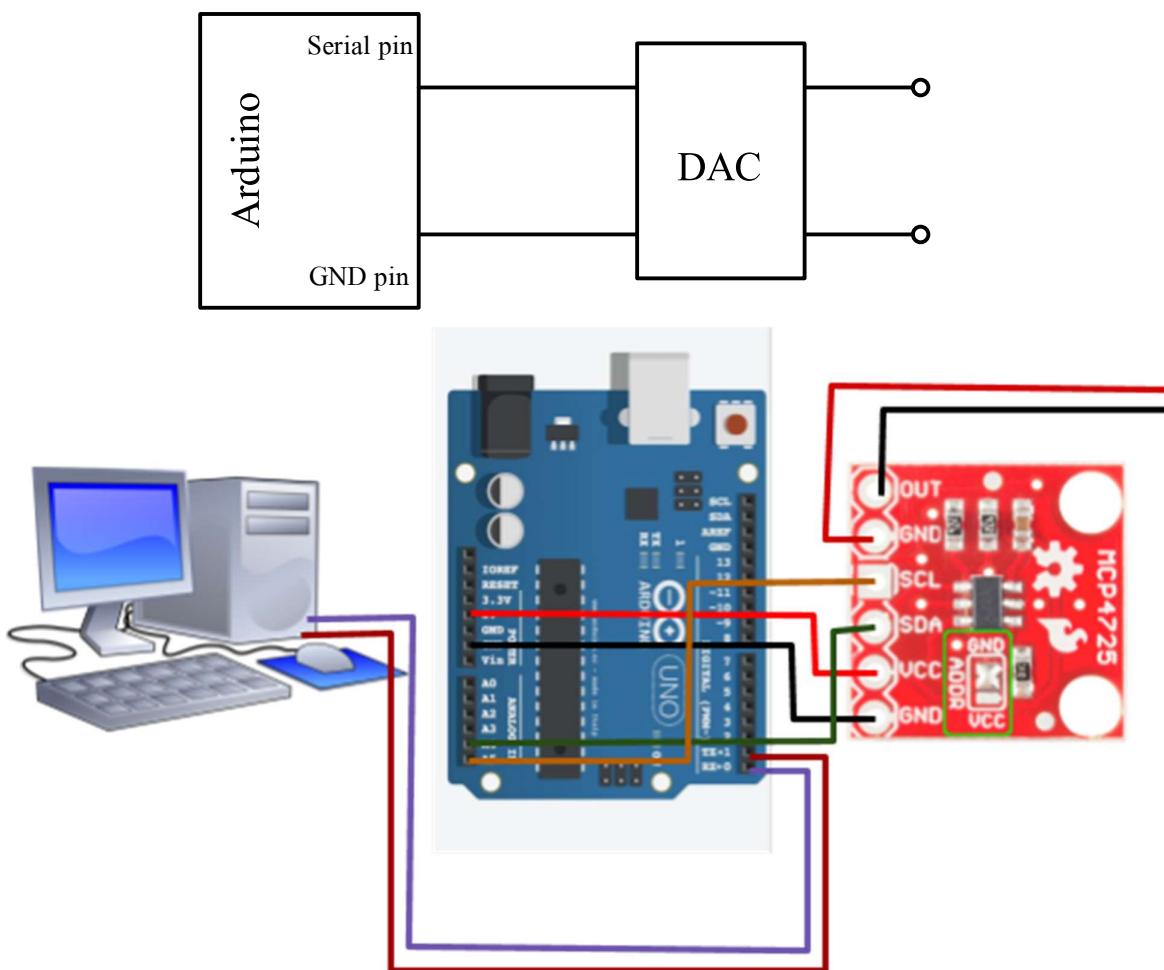
การทดลองที่ 6.3 การสร้าง Sine Wave บน Arduino

- Download Library Adafruit เพื่อใช้ส่งสัญญาณติดต่อกัน วงจร DAC (MCP4725) จาก

<https://learn.adafruit.com/mcp4725-12-bit-dac-tutorial/using-with-arduino>

https://github.com/adafruit/Adafruit_MCP4725

<https://cdn-learn.adafruit.com/downloads/pdf/mcp4725-12-bit-dac-tutorial.pdf>



รูปที่ 6.7 การต่อ Arduino กับ วงจร DAC (MCP4725)

- ให้นักศึกษาเขียนโปรแกรมบนบอร์ด Arduino UNO R3 ดังนี้

2.1. ส่วนหัวโปรแกรม

- ให้เขียนโปรแกรมดังนี้

```
#include <Wire.h>
#include <Adafruit_MCP4725.h>
Adafruit_MCP4725 dac;
int delay0;

#define defaultFreq 1700 // dac speed max
#define freq0 500 // sine wave frequency
```

- พร้อมกำหนดค่าfre0 อีกที่จำเป็น

2.2. ส่วนฟังก์ชัน setup()

- 1) ให้ตั้งค่า Serial Communication และการเชื่อมต่อวงจรแปลงสัญญาณ Digital-to-Analog (DAC:

MCP4725 12-Bit Digital to Analog (DAC) Converter I2C Interface Module) ดังนี้

```
void setup(void)
{
    Serial.begin(115200); // set baudrate serial
    dac.begin(0x62); // set to default For MCP4725A1
    //dac.begin(0x64); // set For MCP4725A2
    //dac.begin(0x60); // set For MCP4725A0
    delay0 = (1000000/freq0 - 1000000/defaultFreq)/4;
    // delay for period of sine
    Serial.print("delay0 is ");
    Serial.println(delay0);
}
```

- 2) ทำการคำนวณค่า $S[i]$ และ $S_{DAC}[i]$ เพื่อนำไปใช้ต่อในฟังก์ชัน loop() โดยที่

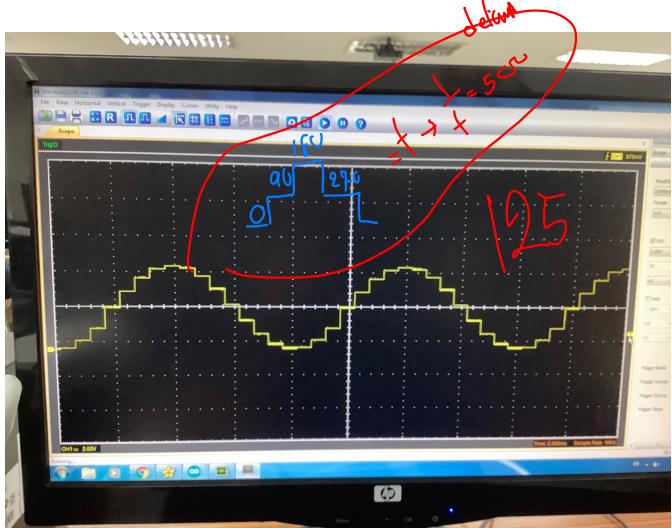
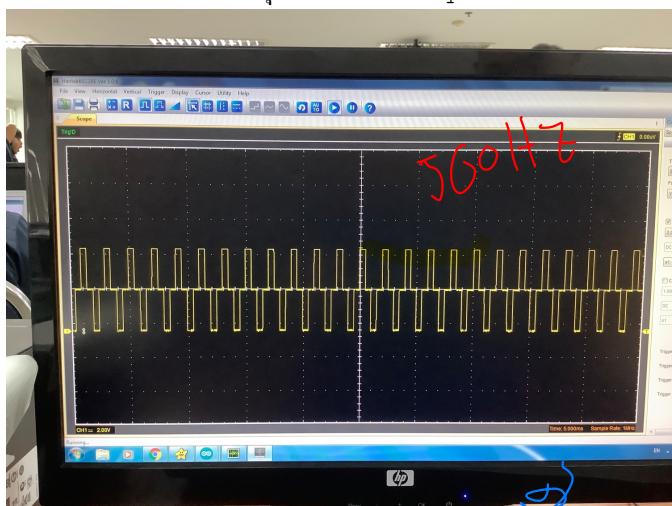
- o ให้ $S[i]$ เก็บค่าจากการคำนวณค่า Sine ของมุมต่างๆ ได้แก่ $0^\circ, 90^\circ, 180^\circ$ และ 270°
- o ให้ $S_{DAC}[i]$ เก็บค่า Sine ที่ได้ ในรูปแบบของ Digital ขนาด 12 บิต เพื่อส่งให้วงจร DAC ($0 \leq DAC \leq 4095$)
- o Debug ค่า $S[i]$ และ $S_{DAC}[i]$ แต่ละลำดับด้วยคำสั่ง Serial.println();

2.3. ส่วนฟังก์ชัน loop() ให้เขียนโปรแกรมเพื่อล�สัญญาณ Sine จากมุมต่างๆ ไปยังบั้งวงจร DAC ด้วยคำสั่ง

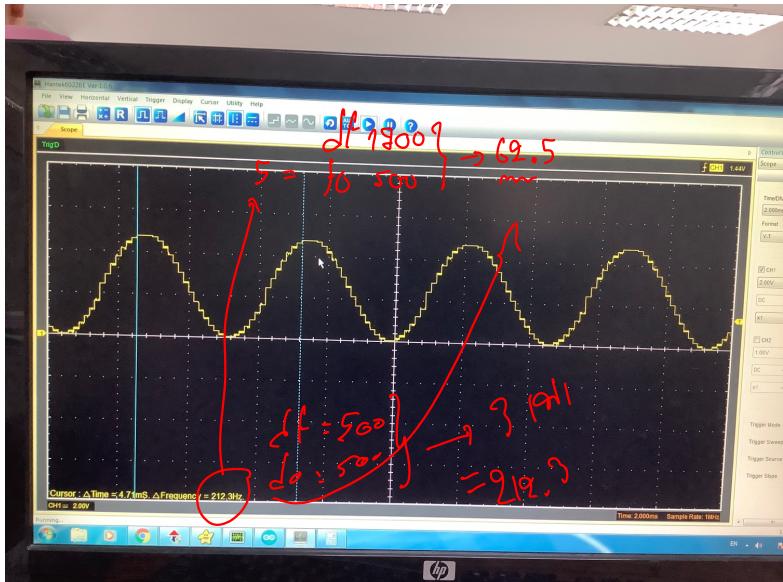
```
for(int i=0 ; i<4 ; i++){
    dac.setVoltage(S_DAC[i], false);
    delayMicroseconds(delay0); // สำหรับ sine freq0
}
```

3. ทำการทดลองเพิ่มเติม โดยปรับโปรแกรมในข้อ 2 ให้เพิ่มจำนวนมุมที่คำนวณค่าองค์ Sine เป็นจำนวน 8, 16 และ 32 มุม ที่ความถี่ $freq0 = 500$ Hz เท่าเดิม จำนวน 2 ต่อ 1 ต่อ 1 กะ

4. ให้นักศึกษาตรวจสอบกราฟของสัญญาณ และอธิบายความแตกต่างของสัญญาณที่ได้ เมื่อจำนวนมุมเป็น 4 และ 16 มุม ที่ความถี่ $freq0 = 500$ Hz



- 125
5. ทำการทดลองเพิ่มเติม โดยเขียนโปรแกรมสร้างสัญญาณ Sine จำนวน 32 บุฟ ที่ความถี่ f_{req0} เป็น 500, 1,000 และ 5,000 Hz ตามลำดับ
 6. ให้นักศึกษาวิเคราะห์รูปกราฟของสัญญาณ และอธิบายความแตกต่างของสัญญาณที่ได้จากข้อ 5 เมื่อส่งค่า Sine ที่ความถี่ 500 และ 5,000 Hz

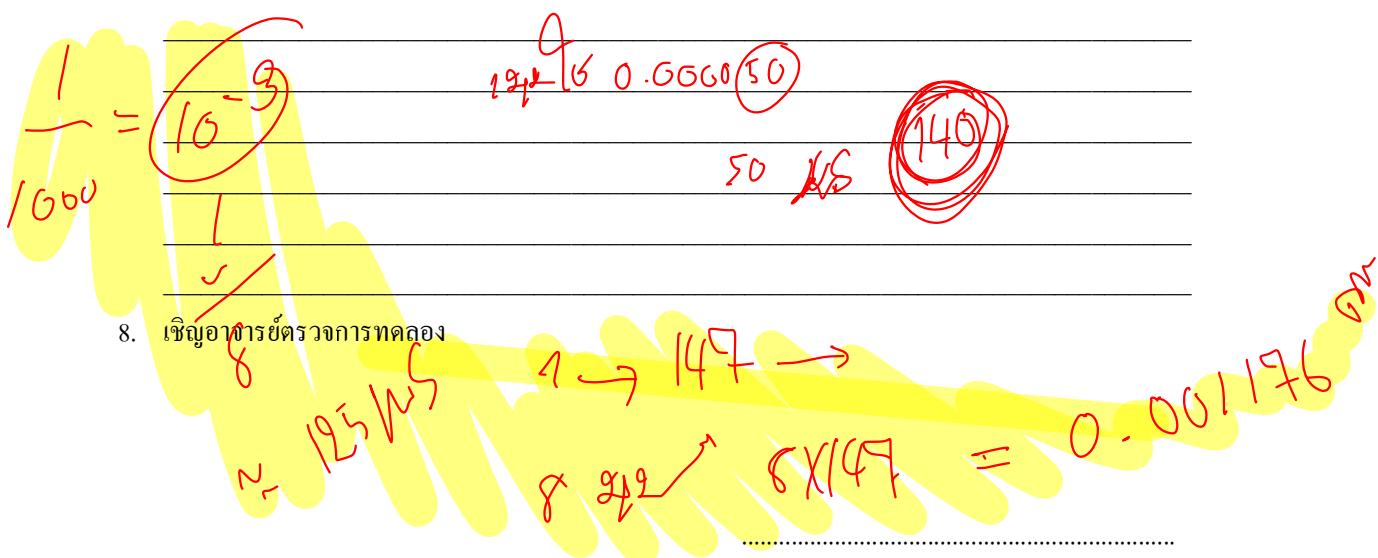


$$\left. \begin{array}{l} f = 1500 \\ f_0 = 500 \end{array} \right\} \rightarrow 2 \text{ Hz}$$

$$\left. \begin{array}{l} f = 5000 \\ f_0 = 500 \end{array} \right\} \rightarrow 212 \text{ Hz}$$

delay $\left(\frac{+}{\downarrow} - \frac{x}{f_0} \right)$

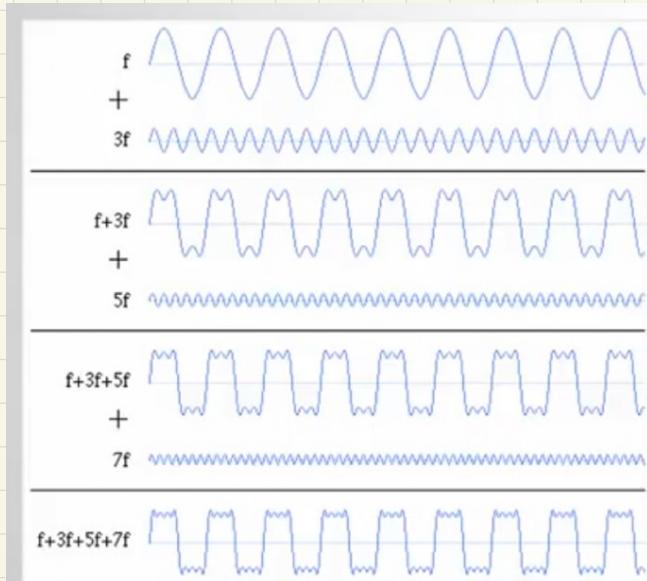
7. สรุป และวิเคราะห์ผลการทดลอง



ลายเซ็นอาจารย์ผู้ตรวจการทดลอง

Converting an Arduino PWM Output to DAC output

↳ How to turn PWM signal to analog signal → every waveform can be represent either as DC or summation of sine wave.



↳ a pulse width modulated (PWM) signal is made up adding sine waves. (square wave)
→ adding the frequency odd harmonic like
 $DC + \sin(f) + \frac{1}{3}\sin(3f) + \frac{1}{5}\sin(5f) + \dots$
 pwm ↳ if default frequency is 500 Hz mean:
 our PWM would be DC + 500 Hz + 1500 Hz...
 ↳ if n harmonic → infinite
 ↳ it would be perfect square wave.

Important Notes

- Choosing the cutoff frequency of the filter:
 - The lower the cutoff frequency, the cleaner the DC output
 - The higher the cutoff frequency, the faster you can change the analog output
- Setting the PWM fundamental frequency as high as possible provides flexibility when choosing the filter cutoff frequency
- Arduino Uno default PWM frequency is only ~500Hz, but it can be turned up

→ also PWM max (higher)
cutoff frequency