

Processing



สารบัญ

| | | |
|-----|---|---|
| 1 | คำนำ..... | 2 |
| 2 | การติดตั้งโปรแกรม..... | 2 |
| 3 | รูปร่างหน้าต่าง Windows ของ Processing..... | 2 |
| 4 | การทดลอง | 3 |
| 4.1 | โปรแกรมที่ 1..... | 4 |
| 4.2 | โปรแกรมที่ 2..... | 5 |
| 4.3 | โปรแกรมที่ 3..... | 6 |
| 4.4 | โปรแกรมที่ 4..... | 7 |
| 4.5 | โปรแกรมที่ 5 | 7 |

1 คำนำ

Processing เป็นซอฟต์แวร์ระบบเปิด เหมาะสำหรับผู้ต้องการพัฒนาโปรแกรมเกี่ยวกับการสร้างภาพเคลื่อนไหวและการมีปฏิสัมพันธ์ ที่เรียกว่า GUI Graphics User Interface สำหรับผู้ที่เคยใช้ชุดพัฒนาโปรแกรม Arduino เมื่อเห็นรูปร่างหน้าต่างการอินเตอร์เฟซของ Processing แล้วจะรู้สึกคุ้นเคยมาก ด้วยเพราะเหมือนกันนั่นเอง อีกทั้ง Processing และ Arduino ใช้หลักการในการเขียนโปรแกรมเหมือนกัน โดยมีพื้นฐานมาจากภาษา C/C++ รวมถึงการติดตั้งชุดพัฒนาก็เหมือนกันด้วย

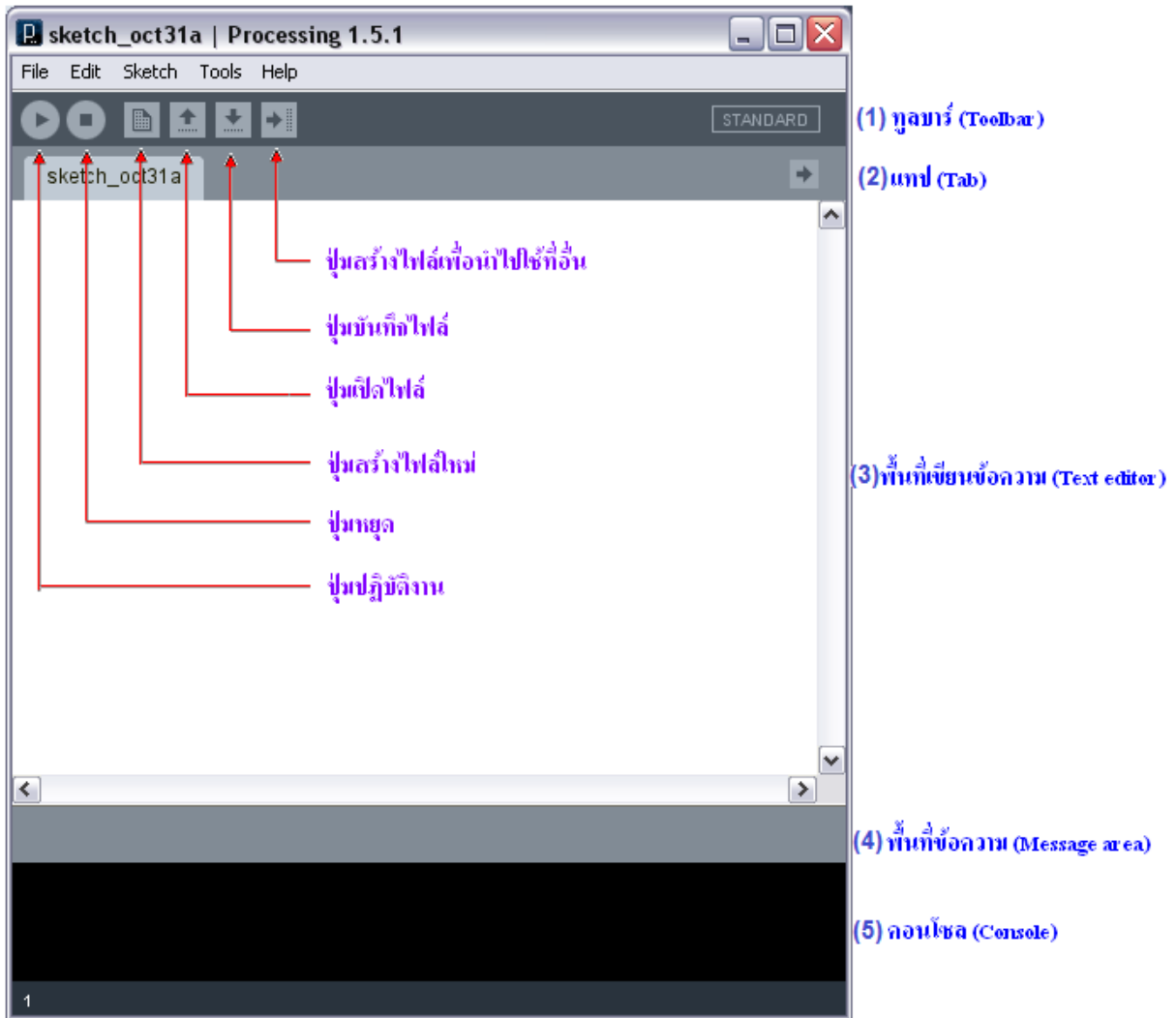
2 การติดตั้งโปรแกรม

- 1) ไปที่ <http://processing.org/> เพื่อดาวน์โหลดโปรแกรม processing ซึ่งมีให้เลือกทั้งระบบปฏิบัติการ Linux Mac OSX และ Windows สำหรับ Windows มีให้เลือกทั้งแบบ มี Java และ ไม่มี Java (สำหรับเบราว์เซอร์ที่มี Java อยู่แล้ว) ในที่นี้ได้ดาวน์โหลด processing-1.5.1-windows-expert.zip เป็นเวอร์ชัน 1.5.1 แบบไม่มี Java ถ้าต้องการความสะดวกสบายควรใช้รุ่นที่มี Java แต่ก็อาจไม่ได้ Java รุ่นล่าสุด
- 2) ให้แตกไฟล์ จะได้โฟลเดอร์ processing-1.5.1 จะย้ายไปไว้ที่ไหนก็ได้ เช่นย้ายไปเก็บที่ C:/Program File
- 3) เวลาจะเปิดโปรแกรม ให้เข้าไปดับเบิลคลิกไฟล์ processing.exe ในโฟลเดอร์ processing-1.5.1 หรือจะสร้างช็อตคัต (Shortcut) แล้วดับเบิลคลิกที่ ช็อตคัต ก็ได้

3 รูปร่างหน้าต่าง Windows ของ Processing

หน้าต่าง windows ของ Processing เป็นตามรูปที่ 1 โดยมีรายละเอียดต่างๆดังนี้

- (1) ทูลบาร์ เป็นแถบคำสั่งและปุ่ม Shortcut ของคำสั่งที่ใช้บ่อยๆ
- (2) แท็บ แสดงพื้นที่เขียนคำสั่งของแต่ละไฟล์ ใช้ในกรณีที่จะมีหลายๆไฟล์อยู่ในโปรแกรมเดียวกัน
- (3) เป็น Text editor เป็นพื้นที่เขียนคำสั่ง
- (4) พื้นที่ข้อความ (Message Area) ใช้แสดงข้อความของคำสั่งที่เกิด Error
- (5) คอนโซล (Console) ใช้แสดงข้อความการโต้ตอบของโปรแกรมในขณะที่ทำงาน
- (6) นอกจากนี้ถ้าคอมพิวเตอร์ต่ออินเทอร์เน็ตอยู่ สามารถดูการเริ่มต้นใช้งานได้จากเมนู Help ของโปรแกรม



รูปที่ 1

4 การทดลอง

ฟังก์ชันหลักๆของโปรแกรมจะมี 2 ฟังก์ชันคือ

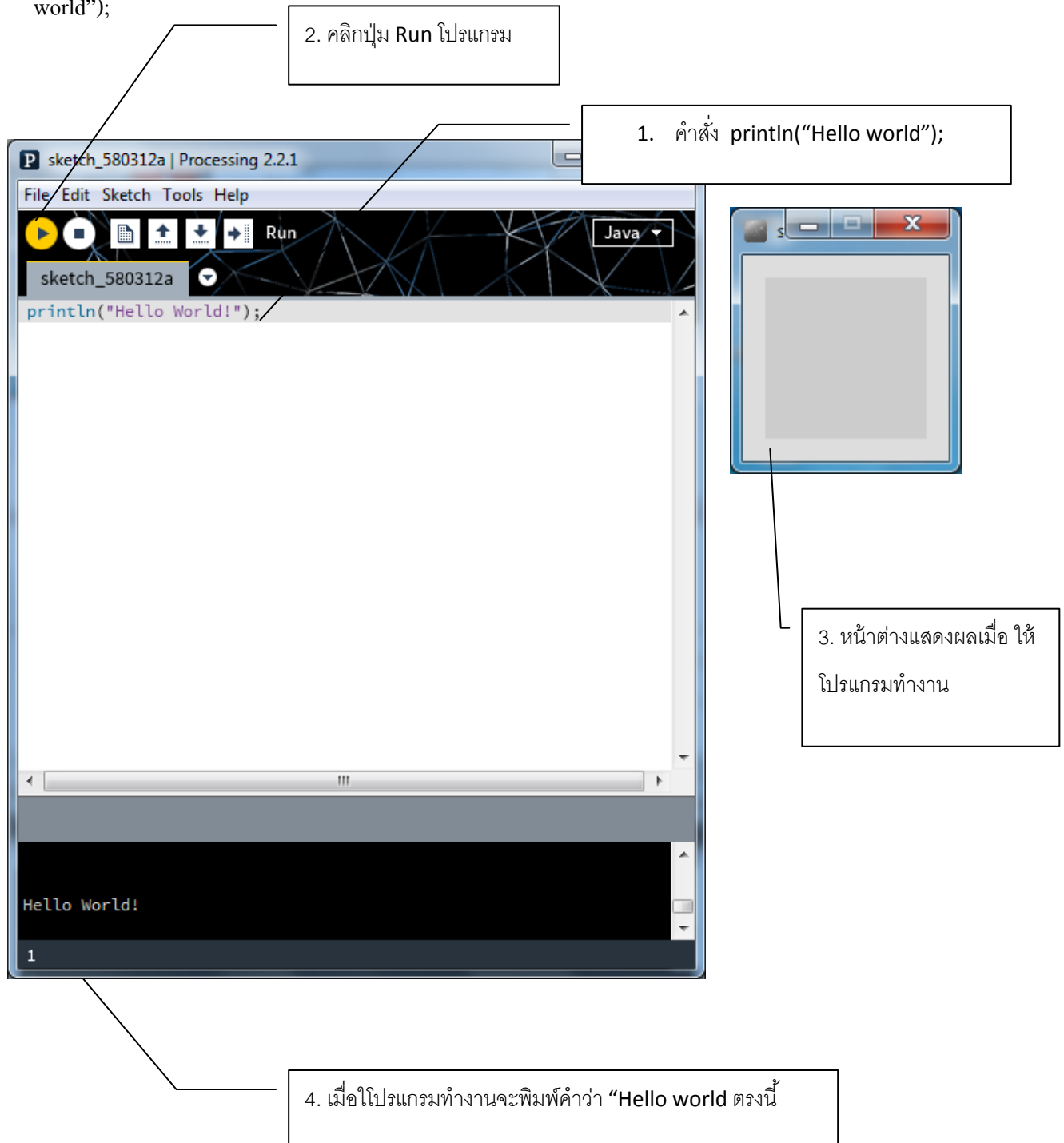
```
void setup() {
}
void draw() {
}
```

ซึ่งจะเป็นแบบเดียวกับโปรแกรม Arduino คือฟังก์ชัน Setup() ใช้กำหนดค่าเริ่มต้นต่างๆ เช่นขนาดพื้นที่ทำงาน คำสั่งในฟังก์ชันนี้จะถูกทำงานก่อน และทำเพียงครั้งเดียว ส่วนฟังก์ชัน draw() จะทำหลังจากฟังก์ชัน Setup() และการทำงานจะวนอยู่ในฟังก์ชันนี้ตลอดไป เหมือนกับฟังก์ชัน loop() ใน Arduino การเขียนโปรแกรมหนึ่งๆ อาจมีครบทั้ง 2 ฟังก์ชัน หรือมีมากกว่า หรือมีไม่ครบก็ได้ ดังตัวอย่างการทดลอง

ต่างๆ ที่จะเริ่มจากง่ายๆ ไม่มีฟังก์ชันเลย ไปจนมีหลายฟังก์ชัน ขอให้นักศึกษาทดลองให้ละเอียด เพราะจะนำไปใช้ได้ดียิ่งขึ้น

4.1 โปรแกรมที่ 1

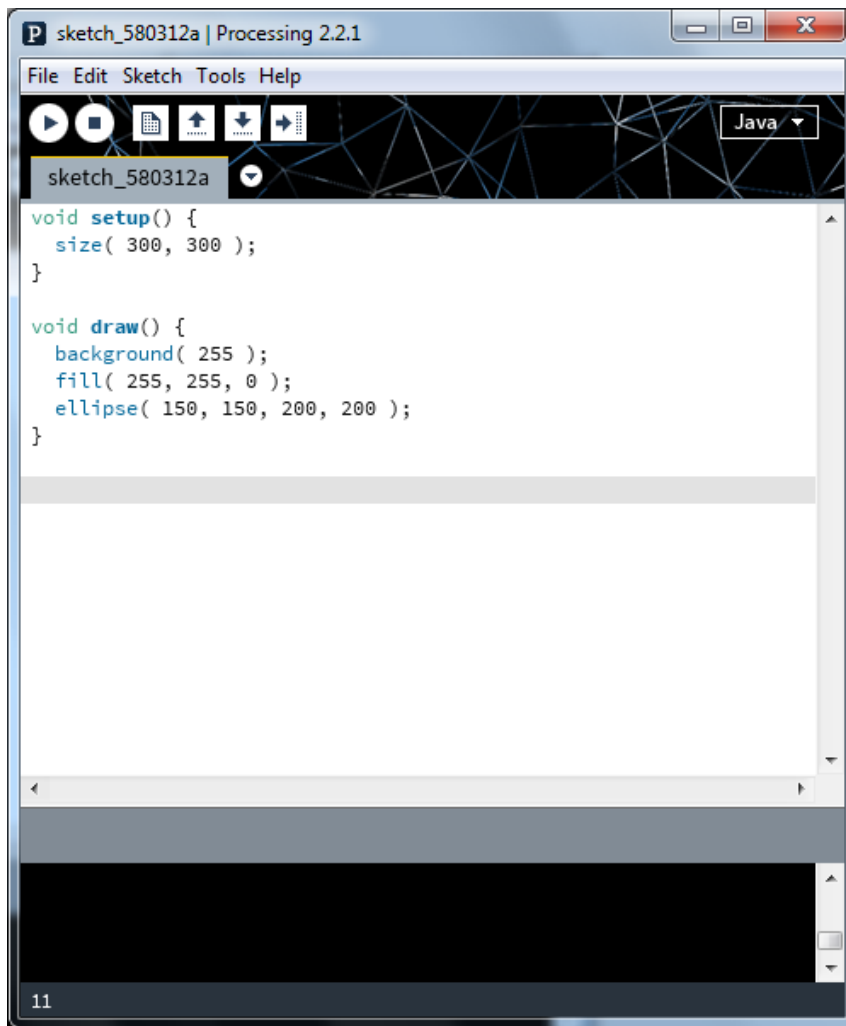
พิมพ์คำว่า “Hello world” ที่พื้นที่คอนโซล โปรแกรมนี้มีคำสั่ง เพียงบรรทัดเดียวคือ `println(“Hello world”);`



4.2 โปรแกรมที่ 2

วาดรูปภาพทางหน้าต่างแสดงผล ด้วยคำสั่ง

```
void setup() {  
  size( 300, 300 );    // ขนาดหน้าต่าง กว้าง 300 จุด ยาว 300 จุด  
}  
void draw() {  
  background( 255 );  // ให้สีพื้นหน้าต่างเป็นสีขาว  
  fill( 255, 255, 0 ); // เติมสีแดงกับเขียว (สีเหลืองในคำสั่งถัดไป  
  ellipse( 150, 150, 200, 200 ); // เขียนวงกลมที่ตำแหน่งจุดศูนย์กลาง 150 x150 ด้วยขนาด กว้าง  
                                // ยาว = 200x200 (ถ้าขนาด กว้างยาวไม่เท่ากันจะได้วงรี)  
}
```



เมื่อคลิกปุ่ม Run โปรแกรมให้ดูผลการทำงานที่หน้าต่างแสดงผลการทำงาน
แก้ไขโปรแกรมใหม่ แล้วลองปุ่ม Run โปรแกรมให้ดูผลการทำงาน

```

void setup() {
  size( 300, 300 );
}
void draw() {
  background( 255 );
  fill( 255, 255, 0 );
  ellipse( 150, 150, 200, 200 );
  fill(0);          //เติมสีดำในคำสั่งถัดไป
  ellipse( 120, 120, 20, 20 );
  ellipse( 180, 120, 20, 20 );
  noFill();          //ไม่เติมสี
  arc( 150, 150, 100, 100, 0, PI ); //เขียนเส้นโค้ง
}

```

ถ้าสงสัยการทำงานของคำสั่งไหนให้ดูคำอธิบายรายละเอียด ที่ <https://processing.org/reference/> หรือถ้าต่ออินเทอร์เน็ตอยู่ใช้วิธีคลิกที่เมนูคำสั่ง Help -> Reference ตอนเปิดโปรแกรม Processing อยู่ก็ได้

4.3 โปรแกรมที่ 3

เขียนวงกลม โดยจุดศูนย์กลางวงกลมเลื่อนไปตามเมาส์

```

int x,y;
void setup() {
  size(500,500);          // กำหนดขนาดหน้าต่าง กว้าง 300 จุด ยาว 300 จุด
  x = width/2;           // ให้ ค่า x เท่ากับ 1/2 ของความกว้างหน้าต่าง
  y = height/2;          // ให้ ค่า y เท่ากับ 1/2 ของความสูงหน้าต่าง
}

void draw() {
  background(#00FF00);    //ให้สีพื้นหน้าต่างเป็นสีเขียว (R = 0, G =255, B=0)
  ellipse(x, y, 55, 55);  //เขียนวงกลมรัศมี 55 จุด ที่มีจุดศูนย์กลางอยู่ที่ x และ y
  if (mousePressed == true) //ตรวจสอบเมาส์ว่ามีการกดปุ่มขวาหรือไม่ ถ้ากด
  {
    x = mouseX;           //ให้ค่า x เท่ากับ ตำแหน่งตัวชี้ (Pointer) x ของเมาส์
    y = mouseY;           //ให้ค่า y เท่ากับ ตำแหน่งตัวชี้ (Pointer) y ของเมาส์
  }
}
//วนกลับไปทำคำสั่งแรกในฟังก์ชัน Draw

```

เมื่อ Run โปรแกรม ให้เลื่อนเมาส์พร้อมกับกดปุ่มขวาเมาส์ไปด้วย ดูผลการทำงาน

คำถาม

ถ้าตัดคำสั่ง background(#00FF00); ออกผลการทำงานระหว่างที่มีคำสั่ง background กับไม่มีต่างกันอย่างไร อธิบายความแตกต่าง ประกอบเหตุผลว่าทำไมถึงเป็นแบบนี้

4.4 โปรแกรมที่ 4

การเขียนตัวหนังสือแบบกำหนดฟอนต์ และขนาดบนหน้าต่าง

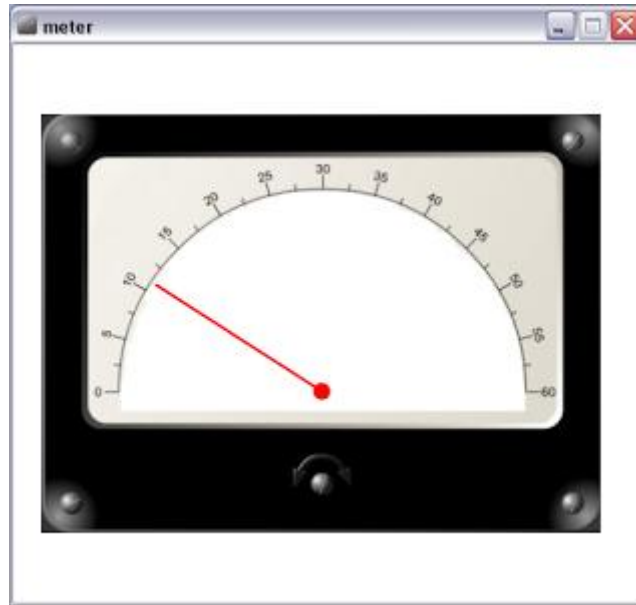
```
PFont myFont;          //กำหนด myFont เป็นFont class
void setup() {
  size(800,500);        //กำหนดขนาดหน้าต่าง
  myFont = createFont("Courier New", 32);    //กำหนดรูปแบบ font ให้กับ Myfont
  textFont(myFont);    // กำหนดให้ใช้ font เป็น myFont
}

void draw() {
  background(255);      //ให้สีพื้นของหน้าต่างเป็นสีขาว
  fill(0);              //ใสสีดालงในคำสั่งถัดไป (ให้ตัวหนังสือเป็นสีดำ)
  textSize(150);       //ขนาดตัวหนังสือ
  int s = second();    // s มีค่าเป็นวินาที จาก 0 - 59
  int m = minute();    // m มีค่าเป็นนาที จาก 0 - 59
  int h = hour();      // h มีค่าเป็นชั่วโมง จาก 0 - 23
  //กำหนดตัวแปร string clock มีค่าเท่ากับ string ของ h+m+s
  String clock = nf(h,2)+':' +nf(m,2)+':' +nf(s,2);
  text(clock, 40, 200); //พิมพ์ตัวแปร string clock ที่ตำแหน่ง x= 40 y=200
}
```

ฟังก์ชัน nf เป็นฟังก์ชันเปลี่ยนตัวเลขให้เป็นตัวหนังสือ เช่น nf(h,2) เปลี่ยนตัวเลขชั่วโมงเป็นตัวหนังสือ 2 ตัว

4.5 โปรแกรมที่ 5

วาดรูปที่มีการเคลื่อนไหว เป็นโปรแกรมที่แสดงการเอาภาพมิเตอร์แบบเข็มผสมกับการเขียนเส้นตรงที่ใช้แทนเข็มมิเตอร์ที่สามารถเคลื่อนไหวของเข็มได้ตามค่าที่กำหนด ในที่นี้ค่าที่ใช้กำหนดตำแหน่งของเข็มได้จากการสุ่มค่าขึ้นมา



รูปที่ 2 ภาพหน้าปัทของมิเตอร์แบบเข็ม

ทั้งนี้รูปภาพมิเตอร์ต้องเก็บอยู่ในโฟลเดอร์ชื่อ data ซึ่งอยู่ในโฟลเดอร์โปรเจก หรือ sketch ของโปรแกรมนี้ รูปภาพมิเตอร์ชื่อ voltmeter_3.jpg สามารถ Download ได้จาก <http://narong.ece.engr.tu.ac.th/microlab/document/index.php> รายการที่ 14

```

PImage img;          //กำหนด img เป็น class รูปภาพ
void setup() {
  size(450,400);      //กำหนดขนาดหน้าต่าง
  img = loadImage("voltmeter_3.jpg");    //โหลดไฟล์ภาพมาเก็บที่ img
}

void draw() {
  background(255);    //ให้สีพื้นของหน้าต่างเป็นสีขาว
  float v = random(0, 60); //กำหนดค่าตัวแปร v เป็นแบบ floating และได้ค่ามาจากการ Random
                          //ตัวเลขระหว่าง 0 ถึง 60
  draw_meter(20,50,v); //เรียกใช้ฟังก์ชันเขียนภาพมิเตอร์ โดย 20,50 เป็นตำแหน่งมุมบนซ้ายของ
                       //ภาพมิเตอร์ที่ต้องการวางไว้ในหน้าต่าง และ v คือค่าที่ใช้กำหนดตำแหน่ง
                       //ของเข็ม
}

// x,y = Upper left corner.
// v = 0 to 60

```



```

void draw_meter(int x,int y,float v)    //ฟังก์ชันเขียนภาพมิเตอร์
{
  image(img, x, y);                    //วางรูปมิเตอร์ที่ตำแหน่ง x, y
  float angle = PI*(60-v)/60;          //จากค่า v เปลี่ยนเป็นมุมเพื่อวาดเข็มมิเตอร์
  float length = 140;                  //ความยาวเข็มเท่ากับ 140 จุด
  float px = width/2 + cos(angle)*length; //คำนวณหาค่า x ของปลายเข็ม
  float py = height/2 + sin(angle)*length; //คำนวณหาค่า y ของปลายเข็ม
  strokeWeight(2);                      //กำหนดขนาดความกว้างของเส้นตรงที่ใช้เป็นเข็มมิเตอร์
  stroke(255, 0, 0);                    //กำหนดสีเส้นตรงให้เป็นสีแดง (R G B)
  line(200+x, 198+y, 200+x+cos(angle)*length, 198+y-sin(angle)*length); //เขียนเส้น
  fill(255,0,0);                         //กำหนดให้สีสีแดง (R=255 G=0 B=0)
  ellipse(200+x, 198+y, 10, 10);        //เขียนวงกลมสีแดงที่เป็นจุดหมุนของเข็ม
}

```