บทที่ 3
การใช้งานพอร์ตอินพุตและเอาต์พุตดิจิตอล

บอร์ดไมโครคอนโทรเลอร์ Arduino UNO R3 มีจำนวนขาใช้งานทั้งหมด 20 ขา แบ่งเป็นขาสัญญาณดิจิทัลจำนวน 12 ขา ได้แก่ ขา 2 - 13 ขาสัญญาณแอนะล็อก จำนวน 6 ขา ได้แก่ ขา A0 – A5 และขาสัญญาณพอร์ตอนุกรมจำนวน 2 ขา ได้แก่ ขา 0 และ ขา 1 ซึ่งสามารถเลือกใช้ขาสัญญาณแต่ละขาให้เหมาะสมกับอุปกรณ์ที่มาเชื่อมต่อ

3.1 การกำหนดโหมดทำงานของขาไมโครคอนโทรลเลอร์

ขาของไมโครคอนโทรลเลอร์แต่ละขามีคุณสมบัติที่เป็นได้ทั้งอินพุตและเอาต์พุต การใช้งานในโหมดอินพุตเป็นการรับข้อมูลจากอุปกรณ์ภาคอินพุต เช่น สวิตช์ เซนเซอร์ โหมดเอาต์พุตเป็นการส่งข้อมูลหรือสถานะลอจิกไปยังอุปกรณ์ภาคเอาต์พุต เช่น การขับหลอดไฟ หลอดไฟ LED มอเตอร์

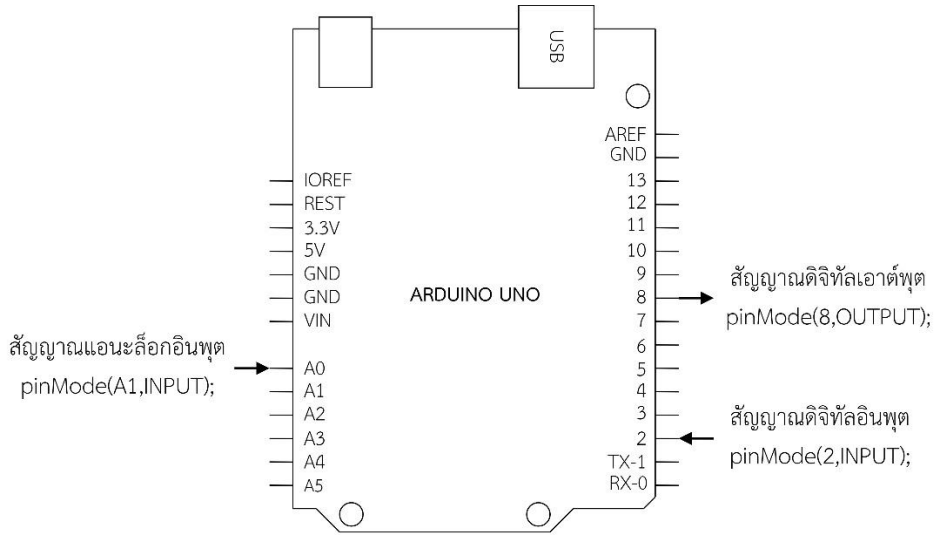
การเขียนโปรแกรมด้วยซอฟแวร์ ARDUINO IDE สามารถกำหนดโหมดของขาด้วยฟังก์ชัน pinMode โดยสามารถเลือกเป็นโหมดอินพุต (input) และโหมดเอาต์พุต (output)

ตัวอย่างการกำหนดโหมดของขาไมโครคอนโทรลเลอร์

pinMode(2, INPUT); หมายถึง กำหนดให้ขา 2 เป็นอินพุตแบบดิจิทัล

pinMode(8, OUTPUT); หมายถึง กำหนดให้ขา 8 เป็นเอาต์พุตแบบดิจิทัล

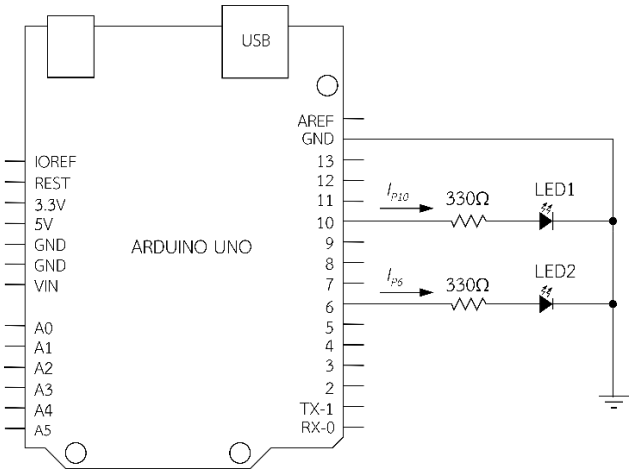
pinMode(A0, INPUT); หมายถึง กำหนดให้ขา A0 เป็นอินพุตแบบแอนะล็อก



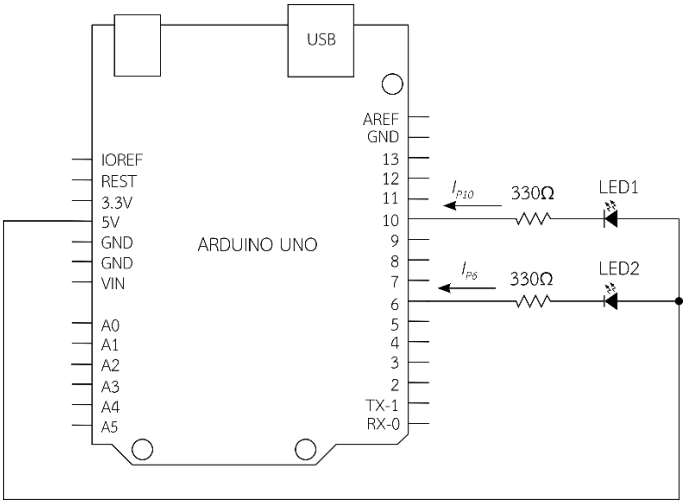
รูปที่ 3.1

3.2 การใช้งานพอร์ตเอาต์พุตดิจิทัล

บอร์ดไมโครคอนโทรเลอร์ Arduino UNO R3 มีขาที่สามารถใช้เป็นขาเอต์พุตได้ทั้งหมด 12 ขา ได้แก่ขา 2 ถึง ขา 13 แต่ละขาสามารถจ่ายกระแสไฟฟ้าได้ไม่เกิน 25 mA ดังนั้นการต่อใช้งานร่วมกับอุปกรณ์เอาต์พุตจะต้องออกแบบวงจรให้มีความเหมาะสมเพื่อไม่ให้กระแสไฟฟ้าเกินค่าพิกัด การต่อใช้งานขาเอาต์พุตของไมโครคอนโทรลเลอร์สามารถต่อได้ทั้งแบบจ่ายกระแสไฟฟ้า (sourcing current) และแบบรับกระแสไฟฟ้า (sinking current) และต้องจำกัดกระแสไฟฟ้าของพอร์ตเอาต์พุตไม่ให้เกินค่ากระแสพิกัด โดยการต่อตัวต้านทานอนุกรมกับอุปกรณ์เอาต์พุต ดังรูปที่ 3.2 และ 3.3



รูปที่ 3.2 การต่อขาเอาต์พุตกับหลอดไฟ LED แบบจ่ายกระแสไฟฟ้า



รูปที่ 3.3 การต่อขาเอาต์พุตกับหลอดไฟ LED แบบรับกระแสไฟฟ้า

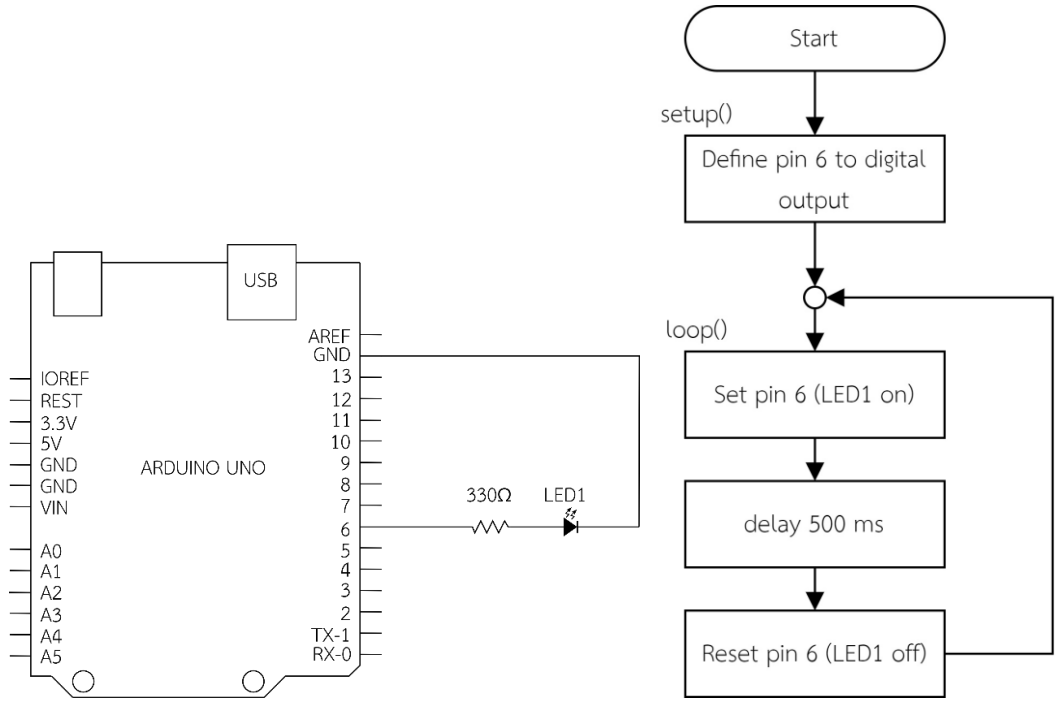
ตัวอย่างการกำหนดค่าลอจิกเอาต์พุต

digitalWrite(6, 0); หมายถึง กำหนดให้ขา 6 มีค่าลอจิกเอาต์พุตเป็น 0

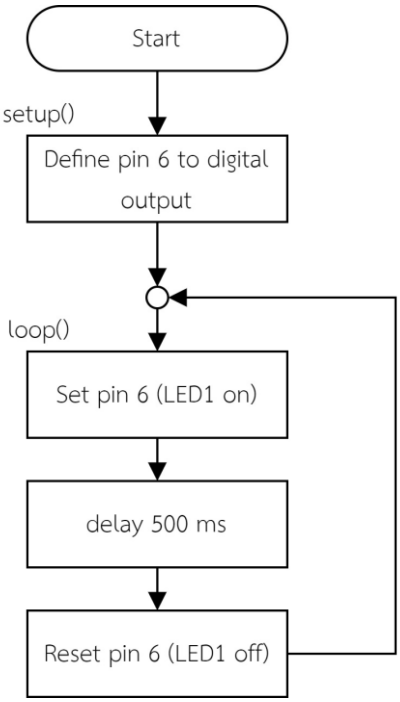
digitalWrite(10, 1); หมายถึง กำหนดให้ขา 10 มีค่าลอจิกเอาต์พุตเป็น 1

ตัวอย่างที่ 3.1 การควบคุมหลอดไฟ LED ให้ติดและดับสลับกัน (ไฟกะพริบ)

การต่อวงจร



รูปที่ 3.4 วงจรทดลองการควบคุมหลอดไฟ LED ให้ติดและดับ



รูปที่ 3.5

การเขียนโปรแกรม

```
1   void setup()
2   {
3         pinMode(6, OUTPUT)      // กำหนดให้ขา 6 เป็นขาเอาต์พุต
4   }
5   void loop()
6   {
7         digitalWrite(6, 1);           // หลอด LED1 ติด
8         delay(500);                   // หน่วงเวลา 500 ms
9         digitalWrite(6, 0);           // หลอด LED1 ดับ
10        delay(500);                   // หน่วงเวลา 500 ms
11  }
```

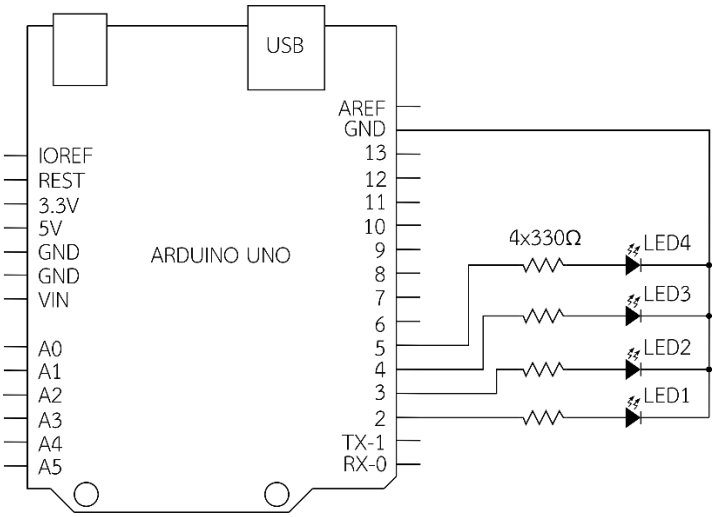
จากโปรแกรมสามารถอธิบายรายละเอียดได้ดังนี้

บรรทัดที่ 1-4 ส่วนของฟังก์ชัน setup() ในส่วนนี้จะใช้ฟังก์ชัน pinMode() กำหนดโหมดของขา 6 เป็นดิจิทัลเอาต์พุตเพื่อใช้ในการขับหลอดไฟ LED1

บรรทัดที่ 5-11 ส่วนของฟังก์ชัน loop() เป็นส่วนของการควบคุมการติดและดับของหลอดไฟ LED1 โดยใช้ฟังก์ชัน digitalWrite() และใช้ฟังก์ชัน delay() ในการหน่วงเวลาของการติดและดับของหลอดไฟ

ตัวอย่างที่ 3.2 การควบคุมหลอดไฟ LED 4 หลอด ให้ติดที่ละหลอด (ไฟวิ่ง) โดยเรียงลำดับจากหลอดไฟ LED1 LED2 LED3 และ LED4 ตามลำดับ

การต่อวงจร



รูปที่ 3.5 วงจรทดลองการควบคุมหลอดไฟ LED 4 หลอด ให้ติดที่ละหลอด

การเขียนโปรแกรม

```
1   void setup()
2   {
3         pinMode(LED1, OUTPUT)          // กำหนดให้ขา 2 - 5 เป็นเอาต์พุต
4         pinMode(LED2, OUTPUT)
5         pinMode(LED3, OUTPUT)
6         pinMode(LED4, OUTPUT)
7   }
8   void loop()
9   {
10        digitalWrite(LED4, 0);
11        digitalWrite(LED1, 1);
```

```
12          delay(500);
13          digitalWrite(LED1, 0);
14          digitalWrite(LED2, 1);
15          delay(500);
16          digitalWrite(LED2, 0);
17          digitalWrite(LED3, 1);
18          delay(500);
19          digitalWrite(LED3, 0);
20          digitalWrite(LED4, 1);
21          delay(500);
22  }
23
24
25
26
```

จากโปรแกรมสามารถอธิบายรายละเอียดได้ดังนี้

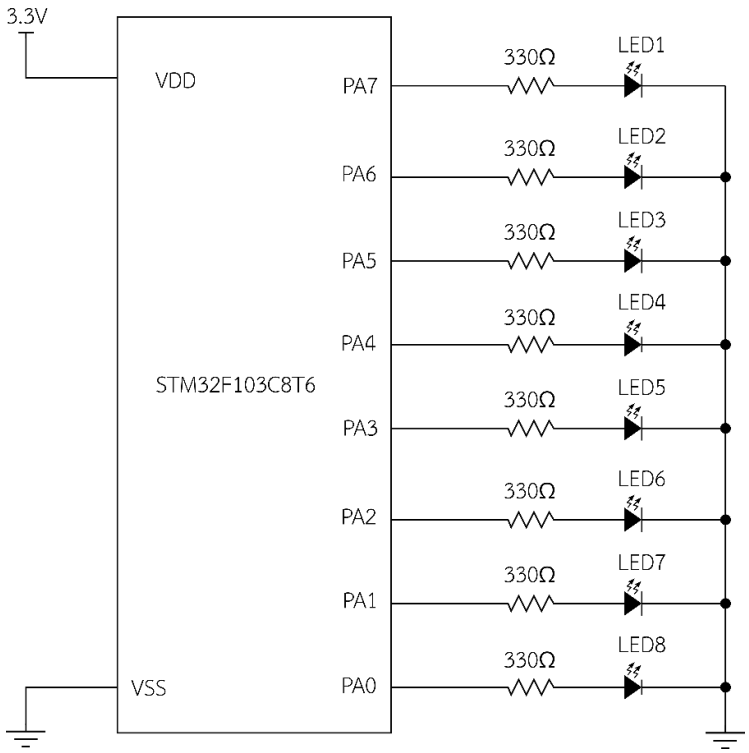
บรรทัดที่ 1 การใช้ไดเร็กทีฟ #define กำหนดชื่อขาต่าง ๆ ให้ตรงกับชื่อของอุปกรณ์ที่เชื่อมต่อ โดยกำหนดชื่อ LED1-4 แทนการเรียกใช้ขา 2-5

บรรทัดที่ 2-9 ส่วนของฟังก์ชัน setup() กำหนดโหมดของขา 2 – PA3 เป็นดิจิทัลเอาต์พุตเพื่อใช้ในการขับหลอดไฟ LED

บรรทัดที่ 10-17 ส่วนของฟังก์ชัน loop() เป็นส่วนของการควบคุมการติดและดับของหลอดไฟ LED ใช้คำสั่ง for ในการวนรอบเพื่อสั่งให้หลอดไฟ LED ติดและดับทีละ 1 หลอด โดยเรียงตามลำดับของขาจาก 2 ถึง 5

ตัวอย่างที่ 3.3 การส่งข้อมูลขนาด 1 ไบต์ โดยวนรอบส่งข้อมูลจาก 0 - 255 ออกพอร์ตเอาต์พุตและแสดงค่าลอจิกของข้อมูลแต่ละบิตด้วยหลอดไฟ LED

การต่อวงจร



รูปที่ 3.5 วงจรทดลองการส่งข้อมูลขนาด 1 ไบต์ แสดงผลด้วยหลอดไฟ LED

การเขียนโปรแกรม

```
1    unsigned char n;
2    void setup()
3    {
4          pinMode(PA0, OUTPUT)          // กำหนดให้ขา PA0-PA7 เป็นเอาต์พุต
5          pinMode(PA1, OUTPUT)
6          pinMode(PA2, OUTPUT)
7          pinMode(PA3, OUTPUT)
8          pinMode(PA4, OUTPUT)
9          pinMode(PA5, OUTPUT)
10         pinMode(PA6, OUTPUT)
11         pinMode(PA7, OUTPUT)
```

```
12  }
13  void loop()
14  {
15          for(n = 0; n <= 255 ; n++)
16          {
17                  digitalWrite(PA0, (0b00000001 & n) != 0);    // บิตที่ 0
18                  digitalWrite(PA1, (0b00000010 & n) != 0);
19                  digitalWrite(PA2, (0b00000100 & n) != 0);
20                  digitalWrite(PA3, (0b00001000 & n) != 0);
21                  digitalWrite(PA4, (0b00010000 & n) != 0);
22                  digitalWrite(PA5, (0b00100000 & n) != 0);
23                  digitalWrite(PA6, (0b01000000 & n) != 0);
24                  digitalWrite(PA7, (0b10000000 & n) != 0);    // บิตที่ 7
25                  delay(1000);
26          }
27  }
```

จากโปรแกรมสามารถอธิบายรายละเอียดได้ดังนี้

บรรทัดที่ 1 การประกาศตัวแปรชื่อ n ชนิด integer ใช้เก็บข้อมูลที่ต้องการส่งของพอร์ตเอาต์พุต

บรรทัดที่ 4 - 11 กำหนดโหมดของขา PA0 - PA7 เป็นขาเอาต์พุต

บรรทัดที่ 15 การวนซ้ำด้วยคำสั่ง for เพื่อส่งค่า 0 – 255 ออกขา PA0 – PA7

บรรทัดที่ 17 - 24 การตรวจสอบค่าของข้อมูลแต่ละบิตและส่งค่าออกขาเอาต์พุต แสดงตัวอย่างการทำงานของโปแกรมได้ดังนี้

เมื่อ n = 100 หรือในรูปของเลขฐานสอง คือ 01100100

บิต 0 (00000001 & 01100100) = 00000000 ค่าเท่ากับ 0 ดังนั้น PA0 = 0

บิต 1 (00000010 & 01100100) = 00000000 ค่าเท่ากับ 0 ดังนั้น PA1 = 0

บิต 2 (00000100 & 01100100) = 00000100 ค่าไม่เท่ากับ 0 ดังนั้น PA2 = 1

บิต 3 (00001000 & 01100100) = 00000000 ค่าเท่ากับ 0 ดังนั้น PA3 = 0

บิต 4 (00010000 & 01100100) = 00000000 ค่าเท่ากับ 0 ดังนั้น PA4 = 0

บิต 5 (00100000 & 01100100) = 00100000 ค่าไม่เท่ากับ 0 ดังนั้น PA5 = 1

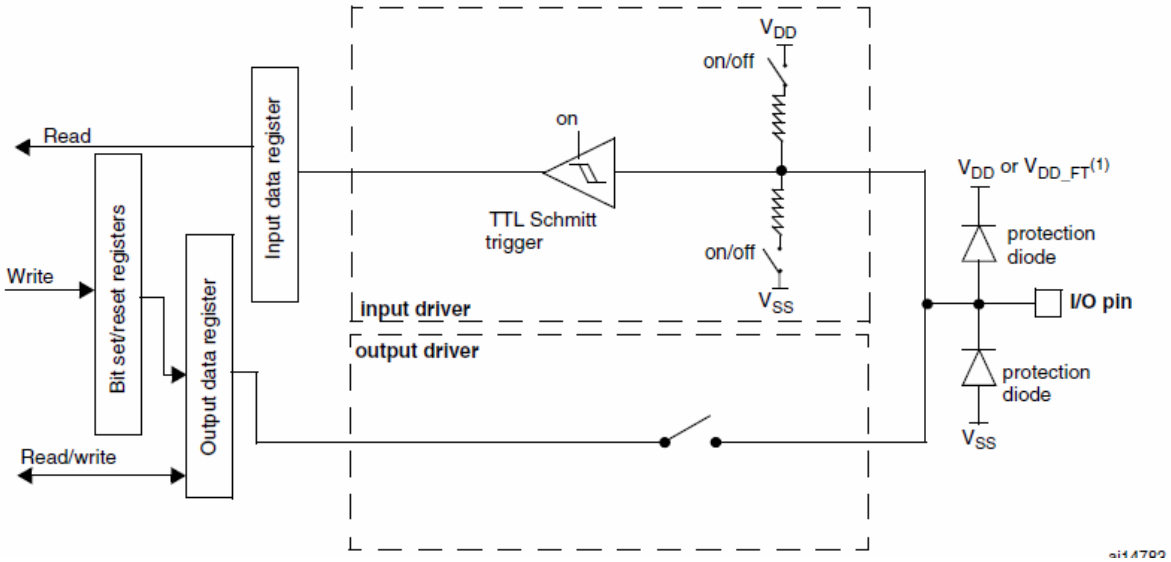
บิต 6 (01000000 & 01100100) = 01000000 ค่าไม่เท่ากับ 0 ดังนั้น PA6 = 1

บิต 7 (10000000 & 01100100) = 00000000 ค่าเท่ากับ 0 ดังนั้น PA7 = 0

บรรทัดที่ 25 หน่วงเวลาการวนซ้ำของโปรแกรม เพื่อค้างสภาวะของขาเอาต์พุตเป็นเวลา 1 วินาที

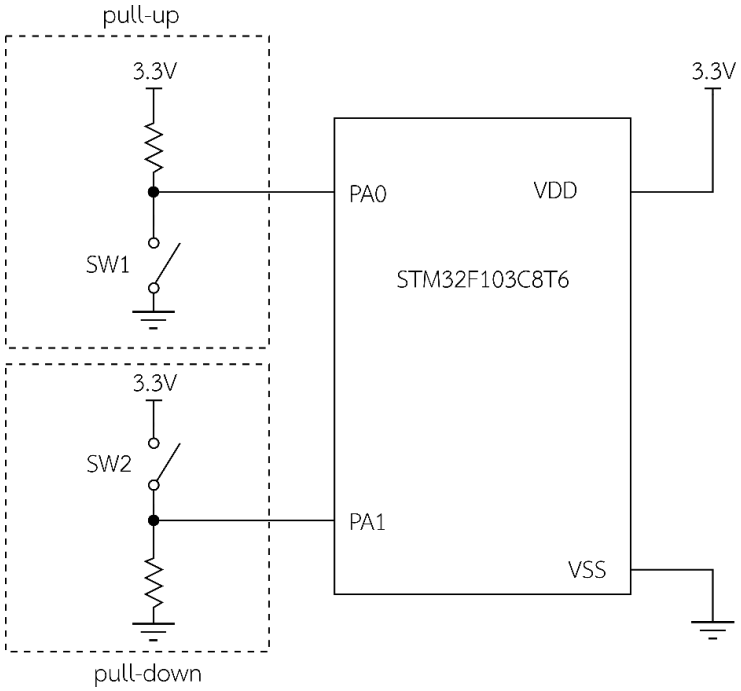
3.3 การใช้งานพอร์ตอินพุตดิจิตอล

ขา GPIO ของไมโครคอนโทรลเลอร์ STM32F103C8T6 สามารถกำหนดโหมดของพอร์ตได้ 3 แบบ ได้แก่ อินพุตพูลอัป (input pull-up) อินพุตพูลดาวน์ (input pull-down) และอินพุตลอย (input floating) แสดงโครงสร้างภาคอินพุตได้ดังรูปที่ 3.6



รูปที่ 3.6 โครงสร้างขาอินพุตของไมโครคอนโทรลเลอร์ STM32F103C8T6

การต่อใช้งานขาไมโครคอนโทรเลอร์ร่วมกับอุปกรณ์อินพุตส่วนมากจะกำหนดโหมดของขาอินพุตให้เป็นอินพุตลอยและใช้การต่อตัวต้านทานพูลอัปหรือพูลดาวน์จากอุปกรณ์อินพุต ดังแสดงในรูปที่ 3.6



รูปที่ 3.7 การต่อสวิตช์แบบตัวต้านทานพูลอัปและตัวต้านทานพูลดาวน์

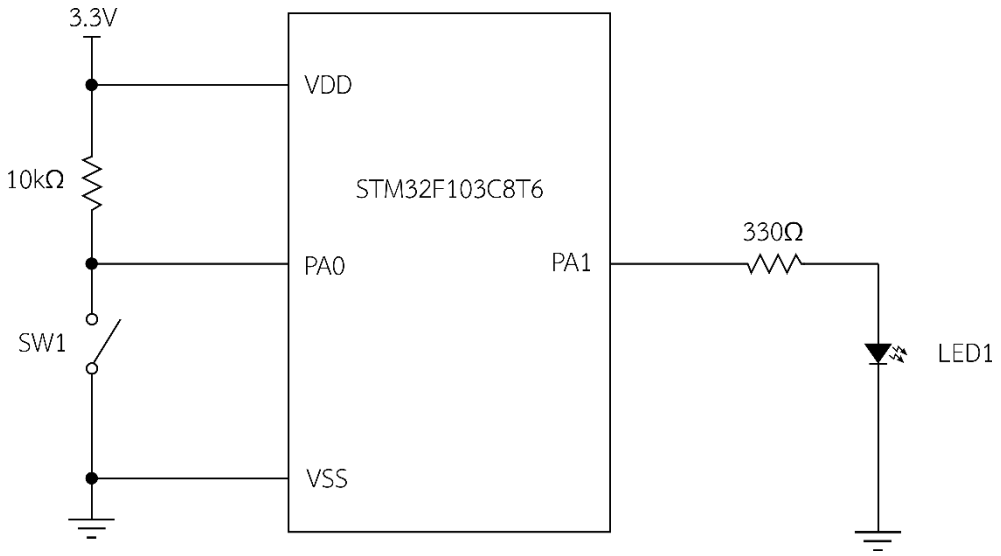
จากรูปที่ 3.7 แสดงการต่อสวิตช์ SW1 กับขา PA0 แบบตัวต้านทานพูลอัปกับแหล่งจ่ายไฟฟ้า ดังนั้น ลอจิกอินพุตของขา PA0 จะมีค่าเป็น 0 เมื่อกดสวิตช์ SW1 และมีค่าเป็น 1 เมื่อปล่อยสวิตช์ SW1 สวิตช์ SW2 ต่อกับขา PA1 แบบตัวต้านทานพูลดาวน์กับไฟ 0 V ดังนั้น ลอจิกอินพุตของขา PA1 จะมีค่าเป็น 1 เมื่อกดสวิตช์ SW2 และมีค่าเป็น 0 เมื่อปล่อยสวิตช์ SW2

การเขียนโปรแกรมด้วยซอฟแวร์ ARDUINO IDE สามารถอ่านค่าลอจิกของขาอินพุตด้วยฟังก์ชัน
digitalRead() แสดงตัวอย่างการใช้งานดังนี้

SW1 = digitalRead(PA0); หมายถึง อ่านค่าสถานะจากขา PA0 และเก็บค่าไว้ที่ตัวแปร SW1

ตัวอย่างที่ 3.4 การเขียนโปรแกรมรับค่าลอจิกจากสวิตช์ กำหนดให้หลอดไฟ LED1 ติดเมื่อกดสวิตช์ SW1 และให้หลอดไฟ LED1 ดับเมื่อปล่อยสวิตช์ SW1

การต่อวงจร



รูปที่ 3.8 วงจรทดลองการรับค่าลอจิกจากสวิตช์

จากวงจรแสดงการต่อสวิตช์ SW1 แบบตัวต้านทานพูลอัปกับขา PA0 เมื่อกดสวิตช์ SW1 สถานะอินพุตของขา PA0 จะมีค่าลอจิกเป็น 1 และเมื่อปล่อยสวิตช์ SW1 สถานะอินพุตของขา PA0 จะมีค่าลอจิกเป็น 0 และต่อหลอดไฟ LED1 ผ่านตัวต้านทานเข้ากับพอร์ต PA1 ดังนั้น การเขียนโปรแกรมจะต้องกำหนดโหมดของขา PA0 เป็นพอร์ตอินพุตแบบดิจิทัล และขา PA1 เป็นพอร์ตเอาต์พุตแบบดิจิทัล

การเขียนโปรแกรม

```
1   #define SW1   PA0                  // การกำหนดชื่อ SW1 แทนการเรียกใช้ PA0
2   #define LED1  PA3                  // การกำหนดชื่อ LED1 แทนการเรียกใช้ PA1
3   bool  SW1_status;                  // ประกาศตัวแปรชนิด boolean ชื่อ SW1_status
4   void setup()
5   {
6       pinMode(SW1, INPUT)            // กำหนดให้ขา PA0 เป็นพอร์ตอินพุต
7       pinMode(LED1, OUTPUT)          // กำหนดให้ขา PA1 เป็นพอร์ตเอาต์พุต
```

```
8   }
9   void loop()
10  {
11          SW1_status = digitalRead(SW1);
12          if(SW1_status == 0)
13                  digitalWrite(LED1, 1);          // หลอด LED1 ติดเมื่อกดสวิตช์ SW1
14          else
15                  digitalWrite(LED1, 0);          // หลอด LED1 ดับเมื่อปล่อยสวิตช์ SW1
16  }
```

จากโปรแกรมสามารถอธิบายรายละเอียดได้ดังนี้

บรรทัดที่ 1 - 2 การใช้ไดเร็กทีฟ #define กำหนดชื่อ SW1 แทนการใช้ PA0 และ LED1 แทนการใช้ PA1

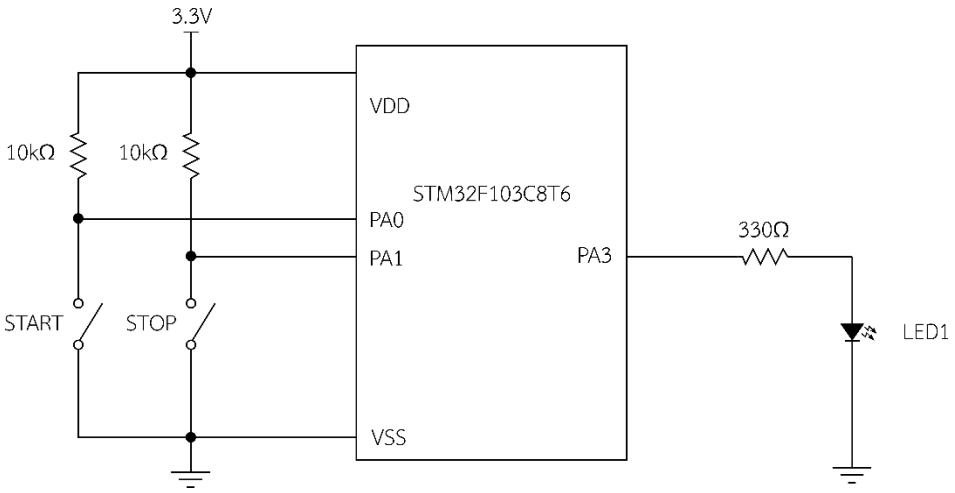
บรรทัดที่ 3 การประกาศตัวแปรชื่อ SW1_status ชนิดบูลีน เพื่อใช้สำหรับเก็บค่าสถานะลอจิกของขา PA0 (SW1)

บรรทัดที่ 4 - 8 ส่วนของฟังก์ชัน setup() การกำหนดโหมดของขา PA0 ให้เป็นดิจิทัลอินพุตเพื่อใช้รับค่าสถานะลอจิกจากสวิตช์ SW1 และกำหนดโหมดของขา PA1 ให้เป็นดิจิทัลเอาต์พุตเพื่อใช้ในการขับหลอดไฟ LED1

บรรทัดที่ 9 - 16 ส่วนของฟังก์ชัน loop() ทำการอ่านค่าสถานะลอจิกจากขา PA0 และเก็บค่าไว้ที่ตัวแปร status และตรวจสอบสถานะของสวิตช์ด้วยคำสั่ง if else เมื่อตัวแปร SW1_status มีค่าเป็น 1 (ปล่อยสวิตช์) จะสั่งให้ขา PA1 มีค่าลอจิกเป็น 0 ด้วยฟังก์ชัน digitalWrite() เพื่อให้หลอดไฟ LED1 ดับ และเมื่อตัวแปร SW1_status มีค่าเป็น 0 (กดสวิตช์) จะสั่งให้ขา PA1 มีค่าลอจิกเป็น 1 เพื่อให้หลอดไฟ LED1 ติด

ตัวอย่างที่ 3.4 การควบคุมคุมหลอดไฟแอลอีดีด้วยสวิตช์ Start - Stop

การต่อวงจร



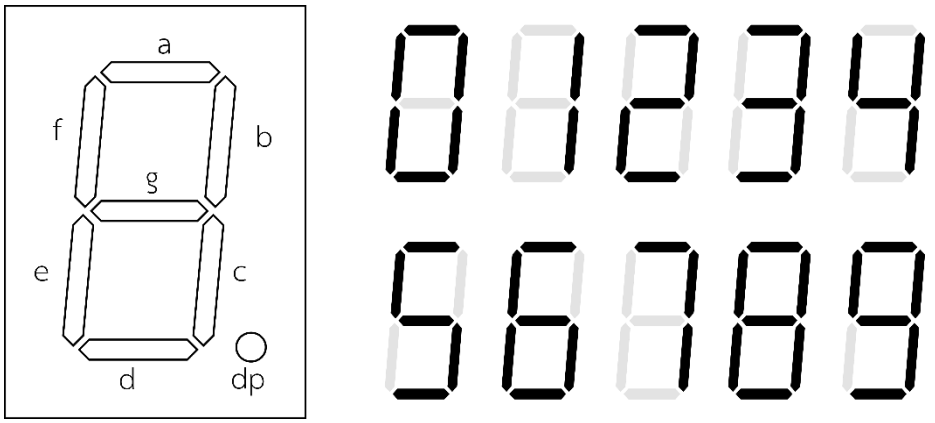
การเขียนโปรแกรม

```
1   #define START  PA0                  // การกำหนดชื่อ START แทนการเรียกใช้ PA0
2   #define STOP    PA1
3   #define LED1    PA3                 // การกำหนดชื่อ LED1 แทนการเรียกใช้ PA3
4   bool  SW1_status;                   // ประกาศตัวแปรชนิด boolean ชื่อ SW1_status
5   void setup()
6   {
7         pinMode(START, INPUT)
8         pinMode(STOP, INPUT)
9         pinMode(LED1, OUTPUT)
10  }
11  void loop()
12  {
13        if(digitalRead(START) == 0)
14              digitalWrite(LED1, 1);      // หลอด LED1 ติดเมื่อกดสวิตช์ START
15        else if(digitalRead(STOP) == 0)
16              digitalWrite(LED1, 0);      // หลอด LED1 ดับเมื่อปล่อยสวิตช์ STOP
```

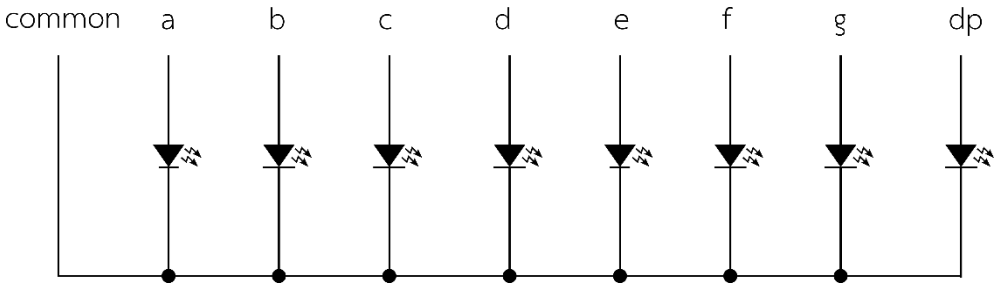
```
17  }
```

3.2 การต่อไมโครคอนโทรลเลอร์ร่วมกับหลอดไฟ LED 7-segment

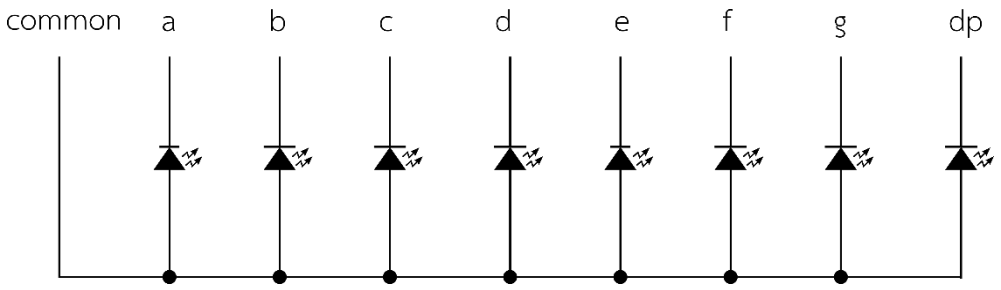
หลอดไฟ LED 7-Segment คือ หลอดไฟ LED จำนวน 7 ลอด ที่เรียงต่อกันในลักษณะเลข 8 ใช้สำหรับแสดงผลข้อมูลแบบตัวเลข และมีหลอดไฟ LED อีก 1 หลอด สำหรับแสดงจุดทศนิยม หลอด LED แต่ละหลอดจะมีอักษรประจำตำแหน่ง ได้แก่ a b c d e f g และ dp ดังรูปที่ 3.9 ลักษณะของหลอดไฟ LED 7-segment สามารถแบ่งออกเป็น 2 ชนิด ตามโครงสร้างของวงจรภายใน คือ ชนิดแอโนดร่วม (common anode) และชนิแคโทดร่วม (common cathode) ดังรูปที่ 3.10



รูปที่ 3.9 ลักษณะของหลอดไฟ LED 7-segment และการแสดงผลตัวเลข
(กิตติศักดิ์ แสนประสิทธิ์, 2557)



ก) หลอดไฟ LED 7-segment ชนิดแคโทดร่วม



ข) หลอดไฟ LED 7-segment ชนิดแอโนดร่วม

รูปที่ 3.10 หลอดไฟ LED 7-segment ชนิดแคโทดร่วมและแอโนดร่วม
(เดชฤทธิ์ มณีธรรม, 2560)

ตารางที่ 3.1 ค่าลอจิกสำหลับควบคุมหลอดไฟ LED 7-segment ชนิดแคโทดร่วม
(เดชฤทธิ์ มณีธรรม, 2560)

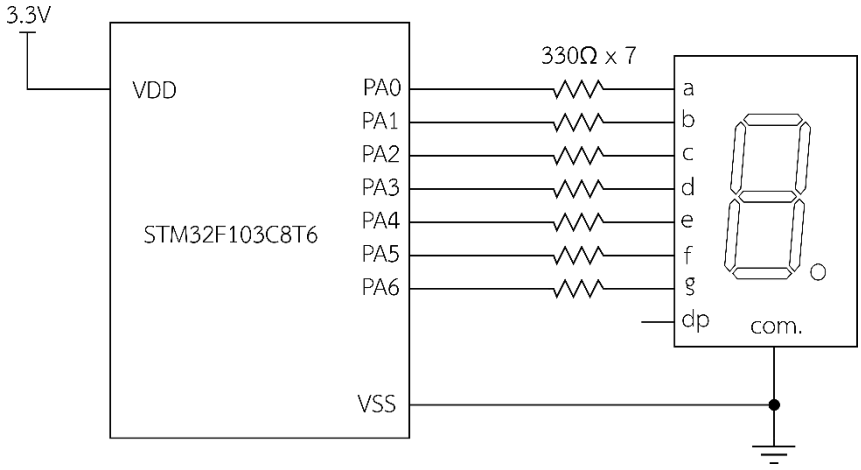
ตัวเลข	ค่าลอจิกของหลอดไฟ LED 7-segment							ข้อมูลเลขฐาน 16
	g	f	e	d	c	b	a	
0	0	1	1	1	1	1	1	0x3F
1	0	0	0	0	1	1	0	0x06
2	1	0	1	1	0	1	1	0x5B
3	1	0	0	1	1	1	1	0x4F
4	1	1	0	0	1	1	0	0x66
5	1	1	0	1	1	0	1	0x6D
6	1	1	1	1	1	0	1	0x7D
7	0	0	0	0	1	1	1	0x07
8	1	1	1	1	1	1	1	0x7F
9	1	1	0	1	1	1	1	0x6F

ตารางที่ 3.2 ค่าลอจิกสำหลับควบคุมหลอดไฟ LED 7-segment ชนิดแอโนดร่วม
(เดชฤทธิ์ มณีธรรม, 2560)

ตัวเลข	ค่าลอจิกของหลอดไฟ LED 7-segment							ข้อมูลเลขฐาน 16
	g	f	e	d	c	b	a	
0	1	0	0	0	0	0	0	0x40
1	1	1	1	1	0	0	1	0x79
2	0	1	0	0	1	0	0	0x24
3	0	1	1	0	0	0	0	0x30
4	0	0	1	1	0	0	1	0x19
5	0	0	1	0	0	1	0	0x12
6	0	0	0	0	0	1	0	0x02
7	1	1	1	1	0	0	0	0x78
8	0	0	0	0	0	0	0	0x00
9	0	0	1	0	0	0	0	0x10

ตัวอย่างที่ 3.5 การแสดงผลแบบตัวเลขด้วยหลอดไฟ LED 7-segment ชนิดแคโทดร่วม โดยให้แสดงผลแบบนับขึ้น 0 - 9

การต่อวงจร



รูปที่ 3.11 วงจรทดลองการแสดงผลแบบตัวเลขด้วยหลอดไฟ LED 7-segment ชนิดแคโทดร่วม

การเขียนโปรแกรม

```
1   #define a   PA0                  // การกำหนดชื่อ a - g แทนการเรียกใช้ PA0 - PA6
2   #define b   PA1
3   #define c   PA2
4   #define d   PA3
5   #define e   PA4
6   #define f   PA5
7   #define g   PA6
8   unsigned char seg[10] = {0x3F, 0x06, 0x5B, 0x4F, 0x66,     // เลข 0 - 4
9                            0x6D, 0x7D, 0x07, 0x7F, 0x6F} ;   // เลข 5 - 9
10  unsigned char count;
11  void seg_put(unsigned char s);
12  void setup()
13  {
14      pinMode(a, OUTPUT)              // กำหนดให้ขา PA0 - PA6 เป็นขาเอาต์พุต
15      pinMode(b, OUTPUT)
16      pinMode(c, OUTPUT)
17      pinMode(d, OUTPUT)
18      pinMode(e, OUTPUT)
19      pinMode(f, OUTPUT)
20      pinMode(g, OUTPUT)
21  }
22  void loop()
23  {
24      for(count = 0; count <= 9; count++)
25      {
26          seg_put(seg[count]);
27          delay(1000);
28      }
29  }
30  void seg_put(unsigned char s)
31  {
```

```
32          digitalWrite(a, (0b00000001 & s) != 0);    // บิตที่ 0
33          digitalWrite(b, (0b00000010 & s) != 0);
34          digitalWrite(c, (0b00000100 & s) != 0);
35          digitalWrite(d, (0b00001000 & s) != 0);
36          digitalWrite(e, (0b00010000 & s) != 0);
37          digitalWrite(f, (0b00100000 & s) != 0);
38          digitalWrite(g, (0b01000000 & s) != 0);    // บิตที่ 6
39  }
```

จากโปรแกรมสามารถอธิบายรายละเอียดได้ดังนี้

บรรทัดที่ 1 - 7 การใช้ไดเร็กทีฟ #define กำหนดชื่อของขาเอาต์พุตให้ตรงกับตำแหน่งขาของหลอดไฟ LED 7-segment

บรรทัดที่ 8-9 การประกาศตัวแปรอาร์เรย์ชนิด integer ชื่อ data จำนวน 10 ตัว เก็บข้อมูลที่ใช้ในการแสดงตัวเลข 0 - 9

บรรทัดที่ 10 การประกาศตัวแปรชนิด integer ชื่อ count;

บรรทัดที่ 11 การประกาศฟังก์ชัน seg_put() ซึ่งเป็นฟังก์ชันส่งข้อมูลออกขาเอาต์พุตเพื่อแสดงผลที่หลอดไฟ LED 7-segment

บรรทัดที่ 14 - 20 กำหนดโหมดของขา PA0 - PA7 เป็นขาเอาต์พุต

บรรทัดที่ 24 การใช้คำสั่ง for วนซ้ำโปรแกรมบรรทัดที่ 25 - 28 จำนวน 10 รอบ โดยดำเนินการนับด้วยตัวแปร count จาก 0 - 9

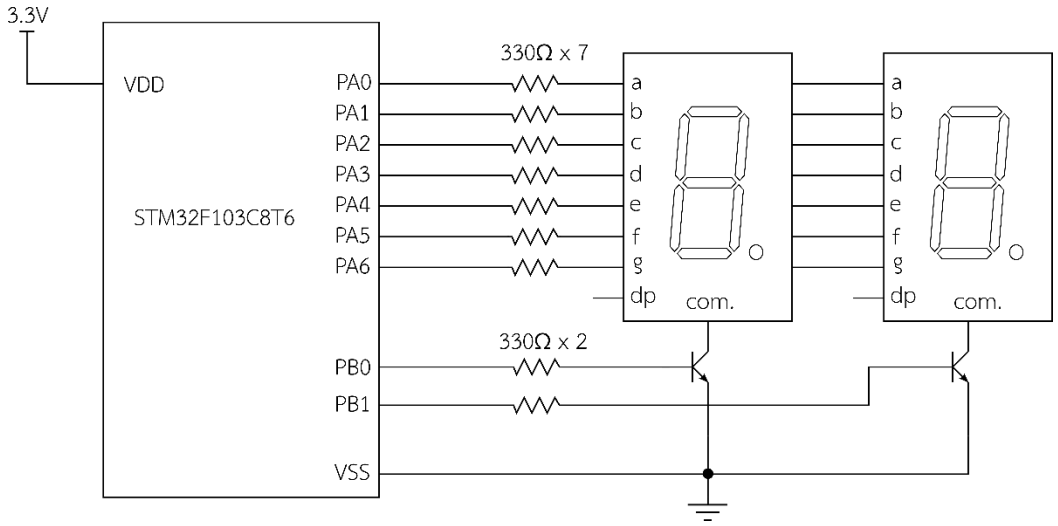
บรรทัดที่ 26 การเรียกใช้ฟังก์ชัน seg_put() เพื่อส่งข้อมูลตัวเลขจากตัวแปรอาร์เรย์ data ไปยังหลอดไฟ LED 7-segment

บรรทัดที่ 27 ฟังก์ชันหน่วงเวลา 1 วินาที

บรรทัดที่ 30 -39 ส่วนของฟังก์ชัน seg_put() เป็นฟังก์ชันส่งข้อมูลขนาด 7 บิต ออกขาเอาพุต

ตัวอย่างที่ 3.6 การแสดงผลแบบตัวเลข 2 หลัก ด้วยหลอดไฟ LED 7-segment ชนิดแคโทดร่วม โดยให้แสดงผลแบบนับขึ้น 0 - 99

การต่อวงจร



รูปที่ 3.12 วงจรการทดลองแสดงผลแบบตัวเลข 2 หลักด้วยหลอดไฟ LED 7-segment

การเขียนโปรแกรม

```
1   #define a   PA0          // กำหนดให้ขา PA0 - PA6 เป็นขาข้อมูล a - g ของ 7-segment
2   #define b   PA1
3   #define c   PA2
4   #define d   PA3
5   #define e   PA4
6   #define f   PA5
7   #define g   PA6
8   #define com1   PB0    // common 1 pin
9   #define com2   PB1    // common 2 pin
10  unsigned char seg[10] = {0x3F, 0x06, 0x5B, 0x4F, 0x66,     // เลข 0 - 4
11                           0x6D, 0x7D, 0x07, 0x7F, 0x6F} ;   // เลข 5 - 9
12  unsigned char count, digit1, digit2, i;
13  void seg_put(unsigned char s);
14  void setup()
15  {
16        pinMode(a, OUTPUT)              // กำหนดให้ขา PA0 - PA6 เป็นขาเอาต์พุต
17        pinMode(b, OUTPUT)
18        pinMode(c, OUTPUT)
```

```
19        pinMode(d, OUTPUT)
20        pinMode(e, OUTPUT)
21        pinMode(f, OUTPUT)
22        pinMode(g, OUTPUT)
23        pinMode(com1, OUTPUT)          // กำหนดให้ขา PB0 - PB1 เป็นขาอินพุต
24        pinMode(com2, OUTPUT)
25  }
26  void loop()
27  {
28        for(count = 0; count <= 99; count++)
29        {
30              digit1 = count / 10;
31              digit2 = count % 10;
32              for(i = 0; i <= 50; i++)
33              {
34                    digitalWrite(com1, 1);            // แสดงผลหลักที่ 1
35                    digitalWrite(com2, 0);
36                    seg_put(seg[digit1]);
37                    delay(10);
38                    digitalWrite(com1, 0);
39                    digitalWrite(com2, 1);            // แสดงผลหลักที่ 2
40                    seg_put(seg[digit2]);
41                    delay(10);
42              }
43        }
44  }
45  void seg_put(unsigned char s)
46  {
47        digitalWrite(a, (0b00000001 & s) != 0);    // บิตที่ 0
48        digitalWrite(b, (0b00000010 & s) != 0);
49        digitalWrite(c, (0b00000100 & s) != 0);
```

```
50          digitalWrite(d, (0b00001000 & s) != 0);
51          digitalWrite(e, (0b00010000 & s) != 0);
52          digitalWrite(f, (0b00100000 & s) != 0);
53          digitalWrite(g, (0b01000000 & s) != 0);     // บิตที่ 6
54  }
```

จากโปรแกรมสามารถอธิบายรายละเอียดได้ดังนี้

บรรทัดที่ 1 - 9 การใช้ไดเร็กทีฟ #define กำหนดชื่อของขาเอาต์พุตให้ตรงกับตำแหน่งขาของหลอดไฟ LED 7-segment

บรรทัดที่ 10 การประกาศตัวแปรอาร์เรย์ชนิด integer ชื่อ data จำนวน 10 ตัว เก็บข้อมูลที่ใช้ในการแสดงตัวเลข 0 - 9

บรรทัดที่ 12 การประกาศตัวแปรชนิด integer ชื่อ count digit1 digit2 และ i

บรรทัดที่ 16 - 24 กำหนดโหมดของขา PA0 - PA7 และขา PB0 - PB1 เป็นขาเอาต์พุต

บรรทัดที่ 28 การใช้คำสั่ง for วนซ้ำโปรแกรมบรรทัดที่ 30 - 41 จำนวน 100 รอบ โดยดำเนินการนับด้วยตัวแปร count จาก 0 - 99

บรรทัดที่ 30 การเก็บค่าหลักสิบของจำนวนในตัวแปร count โดยการหารเด้วย 10 และเก็บค่าไว้ที่ตัวแปร digit1

บรรทัดที่ 31 การเก็บค่าหลักหน่วยของจำนวนในตัวแปร count โดยการหารแบบเอาเศษ (เครื่องหมาย %) ด้วย 10 และเก็บค่าไว้ที่ตัวแปร digit2

บรรทัดที่ 32 การใช้คำสั่ง for วนซ้ำโปรแกรมบรรทัดที่ 34 - 41 จำนวน 50 รอบ เพื่อแสดงผลหลักที่ 1 และหลักที่ 2 สลับกัน จำนวน 50 รอบ โดยแต่ละหลักจะหน่วงเวลาการคงสถานะ 10 มิลิวินาที รวมเวลาการแสดงผลแต่ละค่าเป็นเวลา 1 วินาที

บรรทัดที่ 34 - 36 การส่งข้อมูลหลักสิบออกขาเอาต์พุตและให้แสดงผลที่หลอดไฟ LED 7-segment หลักที่ 1

บรรทัดที่ 38 - 41 การส่งข้อมูลหลักหน่วยออกขาเอาต์พุตและให้แสดงผลที่หลอดไฟ LED 7-segment หลักที่ 2

บรรทัดที่ 44 - 54 ส่วนของฟังก์ชัน seg_put() เป็นฟังก์ชันส่งข้อมูลขนาด 7 บิต ออกขาเอาพุต

3.3 การใช้งานไมโครคอนโทรลเลอร์ร่วมกับจอแสดงผลแอลซีดี

จอแสดงผลแอลซีดี (Liquid Crystal Display, LCD) เป็นจอแสดงผลรูปแบบหนึ่งที่นิยมนำมาใช้งานกันกับระบบสมองกลฝังตัวอย่างแพร่หลาย จอแสดงผลแอลซีดีแยกออกเป็น 2 ชนิด ได้แก่ จอแสดงผลแอลซีดีชนิดตัวอักษร (character LCD) และจอแสดงผลแอลซีดีชนิดกราฟิก (graphic LCD) นอกจากนี้บางชนิดเป็นจอที่มีการผลิตขึ้นมาใช้เฉพาะงาน ทำให้มีรูปแบบและรูปร่างเฉพาะเจาะจงในการแสดงผล เช่น นาฬิกาดิจิตอล เครื่องคิดเลข หรือ หน้าปัดวิทยุ เป็นต้น เนื้อหาในบทนี้จะกล่าวถึงเฉพาะจอแสดงผลชนิดตัวอักษร เนื่องจากมีรูปแบบการใช้งานที่ไม่ซับซ้อนและสามารถประยุกต์ใช้งานได้หลากหลายในงานทางไฟฟ้าและอิเล็กทรอนิกส์

จอแสดงผลแอลซีดีชนิดตัวอักษรเป็นจอแสดงผลที่สามารถแสดงตัวอักษร อักขระและสัญลักษณ์พิเศษ ซึ่งจอแสดงผลที่ใช้งานทั่วไปจะมีรูปแบบการแสดงผลพื้นฐานที่เหมือนกัน ได้แก่ การแสดงตัวอักษรในภาษาอังกฤษ ตัวเลข สัญลักษ์ทางคณิตศาสตร์ และอาจจะมีตัวอักษรหรือสัญลักษณ์พิเศษอื่น ๆ ตามที่ผู้ผลิตกำหนด การเลือกใช้งานจะมีจำนวนบรรทัดของการแสดงผลตั้งแต่ 1 2 และ 4 บรรทัด และมีจำนวนตัวอักษรต่อบรรทัด ตั้งแต่ 16 20 และ 40 เช่น จอแสดงผลแอลซีดีขนาด 16X2 หมายถึงจอแสดงผลแอลซีดีที่สามารถแสดงผลได้ 2 บรรทัด และแต่ละบรรทัดสามารถแสดงผลได้ 16 ตัวอักษร ดังรูปที่ 3.13

รูปที่ 3.13 ลักษณะของจอแสดงผลแอลซีดีขนาด 16x2

จอแอลซีดีชนิดตัวอักษรมีขาต่อใช้งาน 16 ขา ประกอบด้วยขาแหล่งจ่ายไฟฟ้า ขาสัญญาณควบคุมและขาสัญญาณข้อมูล ดังแสดงในตารางที่ 3.3

ตารางที่ 3.3 รายละเอียดขาของจอแสดงผลแอลซีดีชนิดตัวอักษร
(กิตติศักดิ์ แสนประสิทธิ์, 2557)

ตำแหน่งขา	สัญลักษณ์	รายละเอียด
1	VSS	แหล่งจ่ายแรงดันไฟฟ้าด้านลบ (GND)
2	VDD	แหล่งจ่ายแรงดันไฟฟ้าด้านบวก (+5 V)
3	VO	ขาอินพุตเพื่อปรับความเข้มของตัวอักษร
4	RS	ขาอินพุต แยกชนิดระหว่างคำสั่งหรือข้อมูล “0” หมายถึง ข้อมูลที่ส่งมาเป็นคำสั่ง “1” หมายถึง ข้อมูลที่ส่งมาเป็นข้อมูล
5	R/W	เป็นขาเลือกอ่านหรือเขียนข้อมูลให้กับจอแสดงผลอแลซีดี “0” หมายถึง การเขียนข้อมูล “1” หมายถึง การอ่านข้อมูล
6	E	ขาสำหรับรับสัญญาณพัลส์อีนาเบิลจอแสดงผลแอลซีดีให้ทำงาน
7 - 14	D0 - D7	ขาสัญญาณข้อมูลที่ 1 - 8
15	A	ไฟบวก back light (LED+)
16	K	ไฟลบ back light (LED-)

การใช้งานจอแสดงผลแอลซีดีแบบตัวอักษรปกติจะใช้งานในโหมดเขียนข้อมูล ดังนั้นสามารถต่อขา R/W กับแหล่งจ่าย 0 โวลต์ เพื่อลดจำนวนสายสัญญาณ การส่งข้อมูลไปยังจอแสดงผลแอลซีดีสามารถทำได้ 2 แบบ ได้แก่ แบบ 4 บิต และแบบ 8 บิต การส่งข้อมูลแบบ 4 บิต ใช้ขาสัญญาณข้อมูลจำนวน 4 ขา ได้แก่ ขา D4 - D7 ซึ่งใช้ขาสัญญาณน้อยกว่าการส่งข้อมูลแบบ 8 บิต แต่ใช้เวลาในการส่งข้อมูลมากกว่าการส่งข้อมูลแบบ 8 บิต เนื่องจากการส่งข้อมูล 1 ตัวอักษร หรือขนาด 1 ไบต์ (8 บิต) ไปยังจอแสดงผลแอลซีดีต้องแบ่งส่งข้อมูลครั้งละ 4 บิต จำนวน 2 ครั้ง ดังนั้น การต่อไมโครคอนโทรลเลอร์ร่วมกับจอแสดงผลแอลซีดีจะต้องเลือกรูปแบบการต่อวงจรให้เหมาะสมกับการนำไปใช้งาน

การใช้งานไมโครคอนโทรลเลอร์ STM32F103C8T6 ด้วยซอฟแวร์ ARDUINO IDE สามารถเรียกใช้เฮดเดอร์ไฟล์ชื่อ LiquidCrystal.h เข้ามาร่วมกับโปรแกรมหลักเพื่อเรียกใช้คลาสฟังก์ชันชนิด LiquidCrystal ซึ่งเป็นคลาสที่รวบรวมคำสั่งและฟังก์ชันเกี่ยวกับการใช้งานจอแสดงผลแอลซีดีชนิดตัวอักษร ซึ่งมีรูปแบบและการเรียกใช้ฟังก์ชันดังนี้

1) การเรียกใช้คลาสฟังก์ชันชนิด LiquidCrystal และการกำหนดขาสัญญาณ เช่น

LiquidCrystal lcd(PA0, PA1, PA2, PA3, PA4, PA5); หมายถึง การเรียกใช้คลาสฟังก์ชัน LiquidCrystal และตั้งชื่อคลาสเป็น lcd ใช้การส่งข้อมูลแบบ 4 บิต ประกอบด้วยขาสัญญาณจำนวน 6 ขา ได้แก่ ขา PA0 เป็นขาสัญญาณ RS ขา PA1 เป็นขาสัญญาณ E และขา PA2 - PA5 เป็นขาสัญญาณข้อมูล D4 - D7

LiquidCrystal lcd(PB0,PB1,PA0, PA1, PA2, PA3, PA4, PA5,PA6,PA7);

หมายถึง การเรียกใช้คลาสฟังก์ชัน LiquidCrystal และตั้งชื่อคลาสเป็น lcd ใช้การส่งข้อมูลแบบ 8 บิต ประกอบด้วยขาสัญญาณจำนวน 10 ขา ได้แก่ ขา PB0 เป็นขาสัญญาณ RS ขา PB1 เป็นขาสัญญาณ E และขา PA0 - PA7 เป็นขาสัญญาณข้อมูล D1 - D7

2) การเรียกใช้ฟังก์ชันย่อยพื้นฐานในคลาสฟังก์ชันชนิด LiquidCrystal ได้แก่

ฟังก์ชัน begin() เป็นฟังก์ชันย่อยในคลาสฟังก์ชันชนิด LiquidCrystal ใช้กำหนดขนาดและค่าเริ่มต้นของจอแสดงผลแอลซีดี เช่น

lcd.begin(16, 2); หมายถึง การใช้งานไมโครคอนโทรเลอร์กับจอแสดงผลแอลซีดีขนาด 16x2

ฟังก์ชัน clear() เป็นฟังก์ลบข้อความที่แสดงบนจอแสดงผลแอลซีดี

ฟังก์ชัน setCursor() เป็นฟังก์ชันกำหนดตำแหน่งเคอร์เซอร์ของจอแสดงผลแอลซีดี การนับตำแหน่งคอลัมน์และแถวจะเริ่มนับจากค่า 0 เช่น จอแสดงผลแอลซีดีขนาด 16x2 มีค่าตำแหน่งแถว 0 - 15 และค่าตำแหน่งแถว 0 - 1

คอลัมน์

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
แถว	0	A	B	C	D	f	F	G	H	I	J	K	L	M	N	O	P
	1	q	r	s	t	u	w	x	y	z	1	2	3	4	5	6	7

รูปที่ 3.14 ตำแหน่งเคอร์เซอร์ของจอแสดงผลแอลซีดีขนาด 16x2

lcd.setCursor(4, 0); หมายถึง การกำหนดเคอร์เซอร์ของจอแสดงผลแอลซีดีไปที่ตำแหน่งคอลัมน์ที่ 4 แถวที่ 0

ฟังก์ชัน write() เป็นฟังก์ชันส่งข้อมูลขนาด 1 ไบต์ หรือค่ารหัสแอสกีของตัวอักษรที่ต้องการแสดงที่จอแสดงผลแอลซีดี เช่น

lcd.write(0x31); หมายถึง การส่งค่า 0x31 (ค่ารหัสแอสกีของตัวอักษร ‘1’) ไปจอแสดงผลแอลซีดี

lcd.write(0x47); หมายถึง การส่งค่า 0x47 (ค่ารหัสแอสกีของตัวอักษร ‘G’) ไปจอแสดงผลแอลซีดี

ฟังก์ชัน print() เป็นฟังก์ชันส่งข้อมูลตัวอักษรเพื่อแสดงผลที่จอแสดงผลแอลซีดี เช่น

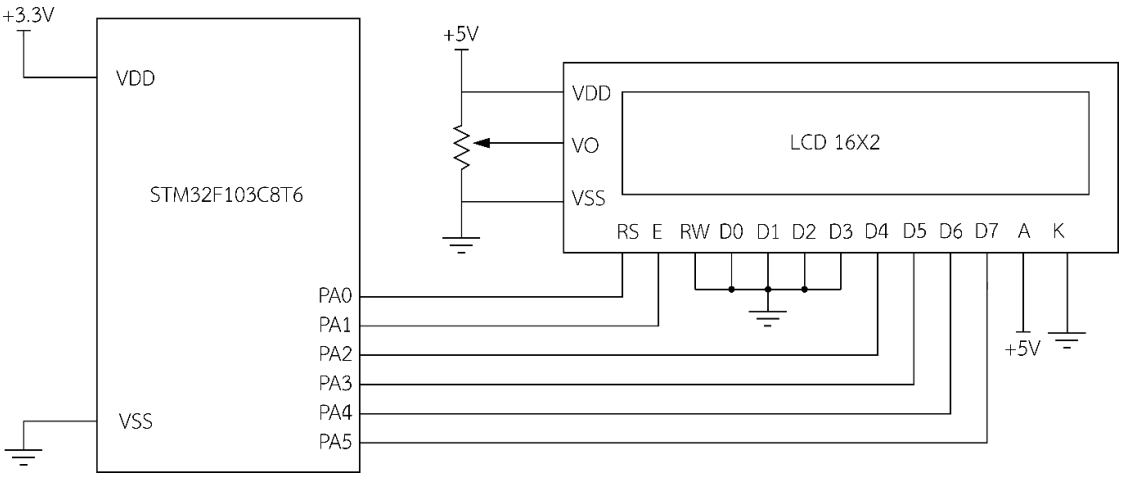
lcd.print(“Hellow”); หมายถึง การแสดงข้อความ “Hellow” ที่จอแสดงผลแอลซีดี

lcd.print(123); หมายถึง การค่าตัวเลข 123 ที่จอแสดงผลแอลซีดี

lcd.print(123, HEX); หมายถึง การค่าตัวเลข 123 ในรูปของเลขฐานสิบหก (7B) ที่จอแสดงผลแอลซีดี

ตัวอย่างที่ 3.7 การส่งข้อมูลไปจอแสดงผลแอลซีดีขนาด 16x2 แบบ 4 บิต เพื่อแสดงข้อความและการนับตัวเลขจากค่า 0 - 255

การต่อวงจร



รูปที่ 3.15 วงจรการทดลองแสดงข้อความและตัวเลขที่จอแสดงผลแอลซีดี

การเขียนโปรแกรม

```
1  #include <LiquidCrystal.h>
2  LiquidCrystal lcd(PA0, PA1, PA2, PA3, PA4, PA5);
3  byte n;
```

```
4   void setup()
5   {
6           lcd.begin(16,2);
7           lcd.setCursor(4,0);
8           lcd.print("COUNTER");
9           n = 0;
10  }
11  void loop()
12  {
13          lcd.setCursor(3,1);
14          lcd.print(n);
15          lcd.setCursor(7,1);
16          lcd.print(n, HEX);
17          n++;   delay(500);
18  }
```

จากโปรแกรมสามารถอธิบายรายละเอียดได้ดังนี้

บรรทัดที่ 1 เรียกใช้ไฟล์ LiquidCrystal.h เข้ามาร่วมกับโปรแกรมหลักเพื่อใช้คำสั่งและฟังก์ชันเกี่ยวกับการใช้งานจอแสดงผลแอลซีดี

บรรทัดที่ 2 - 3 เรียกใช้คลาสฟังก์ชัน LiquidCrystal และตั้งชื่อคลาสเป็น lcd ส่งข้อมูลแบบ 4 บิต ใช้ขาสัญญาณจำนวน 6 ขา ได้แก่ ขา PA0 เป็นขาสัญญาณ RS ขา PA1 เป็นขาสัญญาณ E และขา PA2 - PA5 เป็นขาสัญญาณข้อมูล D4 - D7

บรรทัดที่ 6 การใช้งานไมโครคอนโทรเลอร์ร่วมกับจอแสดงผลแอลซีดีขนาด 16x2

บรรทัดที่ 7 การกำหนดเคอร์เซอร์ของจอแสดงผลแอลซีดีไปที่ตำแหน่งคอลัมน์ที่ 4 แถวที่ 0

บรรทัดที่ 8 การแสดงข้อความ “COUNTER” ที่จอแสดงผลแอลซีดี

บรรทัดที่ 13 การกำหนดเคอร์เซอร์ของจอแสดงผลแอลซีดีไปที่ตำแหน่งคอลัมน์ที่ 3 แถวที่ 1

บรรทัดที่ 14 การค่าตัวเลขของตัวแปร n ที่จอแสดงผลแอลซีดี

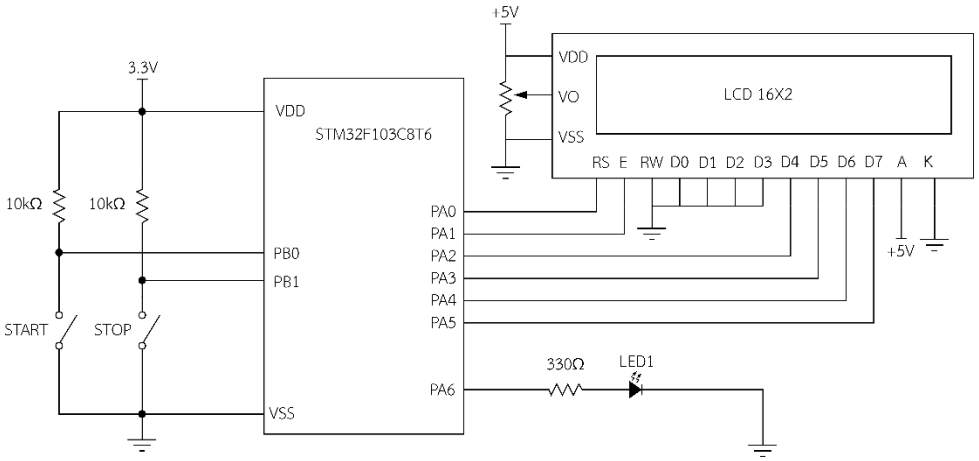
บรรทัดที่ 15 การกำหนดเคอร์เซอร์ของจอแสดงผลแอลซีดีไปที่ตำแหน่งคอลัมน์ที่ 7 แถวที่ 1

บรรทัดที่ 14 การค่าตัวเลขฐานสิบหกของตัวแปร n ที่จอแสดงผลแอลซีดี

บรรทัดที่ 15 เพิ่มค่าของตัวแปร n

ตัวอย่างที่ 3.8 การแสดงสถานะของหลอดไฟแอลอีดีบนจอแสดงผลแอลซีดี

การต่อวงจร



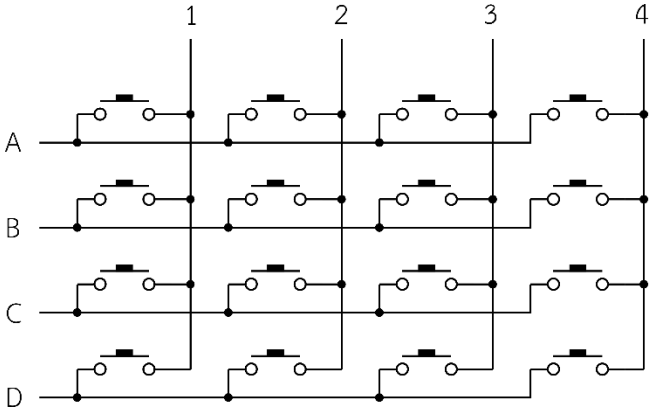
การเขียนโปรแกรม

```
1   #include <LiquidCrystal.h>
2   #define START   PB0
3   #define STOP    PB1
4   #define LED1    PA6
5   LiquidCrystal lcd(PA0, PA1, PA2, PA3, PA4, PA5);
6   void setup()
7   {
8         lcd.begin(16,2);
9         pinMode(START, OUTPUT);
10        pinMode(STOP, OUTPUT);
11        pinMode(LED1, INPUT);
12  }
13  void loop()
14  {
15        lcd.setCursor(3,1);
16        if(digitalRead(START)==0)
17        {
18              digitalWrite(LED1, 1);
```

```
            lcd.print(“LED ON ”);
        }
        else if(digitalRead(STOP)==0)
        {
            digitalWrite(LED1, 0);
            lcd.print(“LED OFF”);
        }
    }
```

3.3 การใช้งานไมโครคอนโทรเลอร์กับสวิตช์เมทริกซ์

สวิตช์เมทริกซ์ คือสวิตช์ที่มีการต่อขาร่วมในแนวตั้งและแนวนอน เหมาะกับการใช้งานที่ต้องการรับค่าจากสวิตช์หลายตัวเนื่องจากช่วยลดจำนวนขาสัญญาณที่ใช้ในการเชื่อมต่อระหว่างสวิตช์กับไมโครคอนโทรลเลอร์ เช่น คีย์สวิตช์แบบเมทริกซ์ขนาด 4x4 มีปุ่มสวิตช์ทั้งหมด 16 ปุ่ม แต่ใช้ขาสัญญาณเพียง 8 ขา แบ่งเป็นขาแนวแถว (row) 4 ขา และขาแนวคอลัมน์ (column) 4 ขา ดังรูปที่ 3.13



ก) วงจรสวิตช์เมทริกซ์

1	2	3	A
4	5	6	B
7	8	9	C
*	0	#	D

ข) ตำแหน่งอักษรของสวิตช์เมทริกซ์

รูปที่ 3.13 สวิตช์เมทริกซ์ขนาด 4x4

(เดชฤทธิ์ มณีธรรม, 2560)

การต่อสวิตช์เมทริกซ์กับไมโครคอนโทรเลอร์จะต่อขาแนวแถวกับขาเอาพุต์และต่อขาแนวคอลัมน์กับอินพุตของไมโครคอนโทรเลอร์ โดยกำหนดโหมดขอเอาต์พุตเป็นแบบ open drain และขาอินพุตเป็นแบบอินพุตพูลอัปหรือการต่อตัวท้านตานภายนอก การอ่านค่าของสวิตช์ใช้วิธีการสแกนค่าที

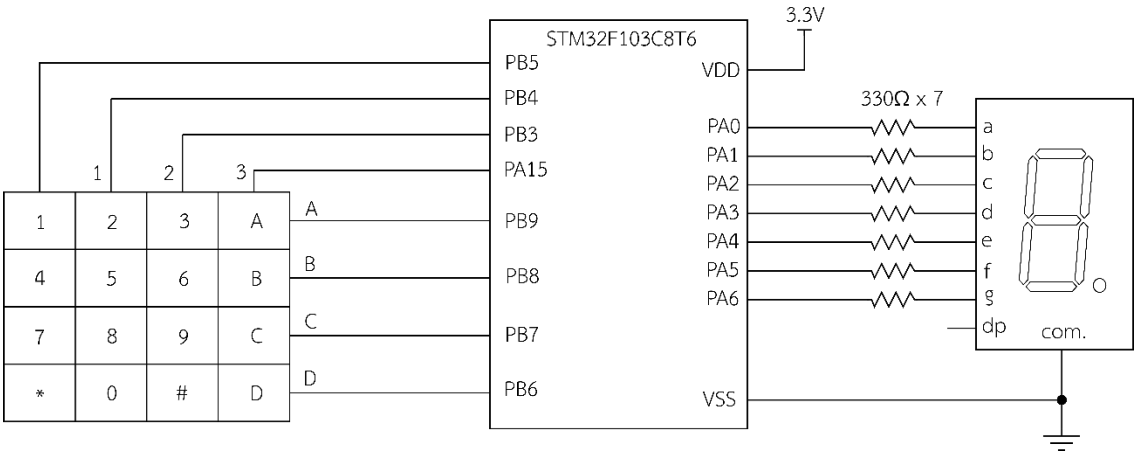
ละแถวโดยการค่าลอจิก 0 ออกขาแนวแถวทีละขา (ขา A - D) และอ่านค่าจากขาสวิตช์แนวคอลัมน์ เมื่อไม่มีการกดสวิตช์ลอจิกของขาอินพุตจะมีเป็น 1 แต่เมื่อมีการกดสวิตช์ลอจิกของขาอินพุตจะมีเป็น 0 การเก็บข้อมูลค่าคีย์สวิตช์สามารถทำได้หลายแบบ เช่น การเก็บข้อมูลแบบบิต แบบตัวอักษร และแบบลำดับ ซึ่งการเก็บข้อมูลแบบตัวอักษรและแบบลำดับจะสามารถอ่านค่าจากสวิตช์ได้ครั้งละ 1 ตำแหน่ง แต่การเก็บข้อมูลแบบบิตจะสามารถอ่านค่าจากสวิตช์พร้อมกันได้หลายตำแหน่งโดยการแทนจำนวนบิตของข้อมูลให้เท่ากับจำนวนสวิตช์ ดังแสดงในตารางที่ 3.3

ตารางที่ 3.3 การทำงานของสวิตช์เมทริกซ์

ตัวอักษร	**แนวแถว (เอาต์พุต)**				**แนวคอลัมน์ (อินพุต)**				**ข้อมูลแบบตัวอักษร**	**ข้อมูลแบบลำดับเลข**	**ข้อมูลแบบบิต (16 บิต)**	
	A	**B**	**C**	**D**	**1**	**2**	**3**	**4**			**เลขฐานสอง**	**เลขฐานสิบหก**
1	0	1	1	1	0	1	1	1	1	1	0000000000000001	0x0001
2	0	1	1	1	1	0	1	1	2	2	0000000000000010	0x0002
3	0	1	1	1	1	1	0	1	3	3	0000000000000100	0x0004
A	0	1	1	1	1	1	1	0	A	4	0000000000001000	0x0008
4	1	0	1	1	0	1	1	1	4	5	0000000000010000	0x0010
5	1	0	1	1	1	0	1	1	5	6	0000000000100000	0x0020
6	1	0	1	1	1	1	0	0	6	7	0000000001000000	0x0040
B									B	8	0000000010000000	0x0080
7	1	1	0	1	0	1	1	1	7	9	0000000100000000	0x0100
8	1	1	0	1	1	0	1	1	8	10	0000001000000000	0x0200
9	1	1	0	1	1	1	0	0	9	11	0000010000000000	0x0400
C									C	12	0000100000000000	0x0800
*	1	1	1	0	0	1	1	1	*	13	0001000000000000	0x1000
0	1	1	1	0	1	0	1	1	0	14	0010000000000000	0x2000
#	1	1	1	0	1	1	0	0	#	15	0100000000000000	0x4000
D									D	16	1000000000000000	0x8000

การเก็บข้อมูลแบบบิตจะต้องทำการสแกนค่าลอจิกของสวิตช์ทุกตัว จากนั้นนำค่าของสวิตช์ที่ถูกกดมารวมกันเพื่อหาค่าของข้อมูลที่ได้จากสวิตช์เมทริกซ์ ตัวอย่างเช่น เมื่อกดสวิตช์ตัวอักษร 3 5 และ 7 พร้อมกัน ข้อมูลที่ได้คือ 0x0004 + 0x0010 + 0x0040 เท่ากับ 0x0054 (0000000001010100)

ตัวอย่างที่ 3.7 การรับค่าจากสวิตช์เมทริกซ์ขนาด 4x3 ชนิดตัวเลข ใช้การเก็บข้อมูลแบบลำดับ และแสดงผลตัวเลข 0 - 9 ด้วยหลอดไฟ LED 7-segment



รูปที่ 3.14 วงจรการทดลองรับค่าจากสวิตช์เมทริกซ์ขนาด 4x4

การเขียนโปรแกรม

```
#include <Keypad.h>
const byte ROWS = 4;
const byte COLS = 4;
char keys[ROWS][COLS] = {
  {1,2,3,'A'},
  {4,5,6,'B'},
  {7,8,9,'C'},
  {'*','0','#','D'}};
byte rowPins[ROWS] = {PB9, PB8, PB7, PB6};
byte colPins[COLS] = {PB5, PB4, PB3, PA15};
Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );
unsigned char seg[10] = {0x3F, 0x06, 0x5B, 0x4F, 0x66,      // เลข 0 - 4
```

```
13                                  0x6D, 0x7D, 0x07, 0x7F, 0x6F} ;   // เลข 5 - 9
14  byte segPins[7] = { PA0, PA1, PA2, PA3, PA4, PA5, PA6};
15  void seg_put(unsigned char s);
16  byte KeyData;
17  void setup()
18  {
19        for(i=0;i<7;i++)
20              pinMode(segPins[i], OUTPUT);
21  }
22  void loop()
23  {
24        KeyData = GetKey();
25        if(KeyData == 0)                          // ไม่มีการกดสวิตช์
26              seg_put(0);                         // LED off
27        if((KeyData>=1) && (KeyData<=9))
28              seg_put(seg[KeyData]);              // แสดงผลเลข 1 - 9
29        if(KeyData == “0”)
30              seg_put(seg[0]);                    // แสดงผลเลข 0
31       delay(100);
32  }
33  void seg_put(unsigned char s)
34  {
35        digitalWrite(segPins[0], (0b00000001 & s) != 0);    // บิตที่ 0
36        digitalWrite(segPins[1], (0b00000010 & s) != 0);
37        digitalWrite(segPins[2], (0b00000100 & s) != 0);
38        digitalWrite(segPins[3], (0b00001000 & s) != 0);
          digitalWrite(segPins[4], (0b00010000 & s) != 0);
          digitalWrite(segPins[5], (0b00100000 & s) != 0);
          digitalWrite(segPins[6], (0b01000000 & s) != 0);   // บิตที่ 6
    }
```

จากโปรแกรมสามารถอธิบายรายละเอียดได้ดังนี้

บรรทัดที่ 1 การเรียกใช้ไฟล์ Keypad.h เข้ามาร่วมกับโปรแกรมหลักเพื่อใช้คำสั่งและฟังก์ชันเกี่ยวกับการใช้งานสวิตช์เมทริกซ์

บรรทัดที่ 2 -3 การประกาศตัวแปรชื่อ ROWS และ COLS สำหรับกำหนดขนาดของสวิตช์เมทริกซ์เป็น 4x4

บรรทัดที่ 4-8 การประกาศตัวแปรอาร์เรย์ ชื่อ keys จำนวน 16 ตัว (ขนาด 4x4) เพื่อเก็บค่าคงที่ประจำตำแหน่งของสวิตช์

บรรทัดที่ 9 การประกาศตัวแปรอาร์เรย์ชนิด byte ชื่อ rowPins จำนวน 4 ตัว เก็บข้อมูลขาที่ใช้ต่อกับขาของสวิตช์เมทริกซ์แนวแถว โดยต่อขา PB9 - PB6 เข้ากับขาแนวแถว A - D ของสวิตช์เมทริกซ์

บรรทัดที่ 10 การประกาศตัวแปรอาร์เรย์ชนิด byte ชื่อ colPins จำนวน 3 ตัว เก็บข้อมูลขาที่ใช้ต่อกับขาของสวิตช์เมทริกซ์แนวคอลัมน์ โดยต่อขา PB5 - PB3 และ PA15 เข้ากับขาแนวคอลัมน์ 1 - 4 ของสวิตช์เมทริกซ์

บรรทัดที่ 12 - 13 การประกาศตัวแปรอาร์เรย์ชนิด integer ชื่อ data จำนวน 10 ตัว เก็บข้อมูลที่ใช้ในการแสดงตัวเลข 0 - 9

บรรทัดที่ 14 การประกาศตัวแปรอาร์เรย์ชนิด byte ชื่อ segPins จำนวน 7 ตัว เก็บข้อมูลขาที่ใช้ต่อกับหลอดไฟ LED 7-segment

บรรทัดที่ 19 - 20 การวนรอบเพื่อกำหนดโหมดของขาที่ใช้ต่อกับหลอดไฟ LED 7-segment (ขา PA0 - PA6) ให้เป็นขาเอาต์พุต

บรรทัดที่ 15 - 16 การวนรอบเพื่อกำหนดโหมดของขาที่ใช้ต่อกับขาแนวแถว A - D ของสวิตช์เมทริกซ์ (ขา PB3 - PB6) ให้เป็นขาเอาต์พุตแบบ open drain

บรรทัดที่ 17 - 18 การวนรอบเพื่อกำหนดโหมดของขาที่ใช้ต่อกับขาแนวแถว 1 - 3 ของสวิตช์เมทริกซ์ (ขา PB7 - PB9) ให้เป็นขาอินพุตพูลอัป

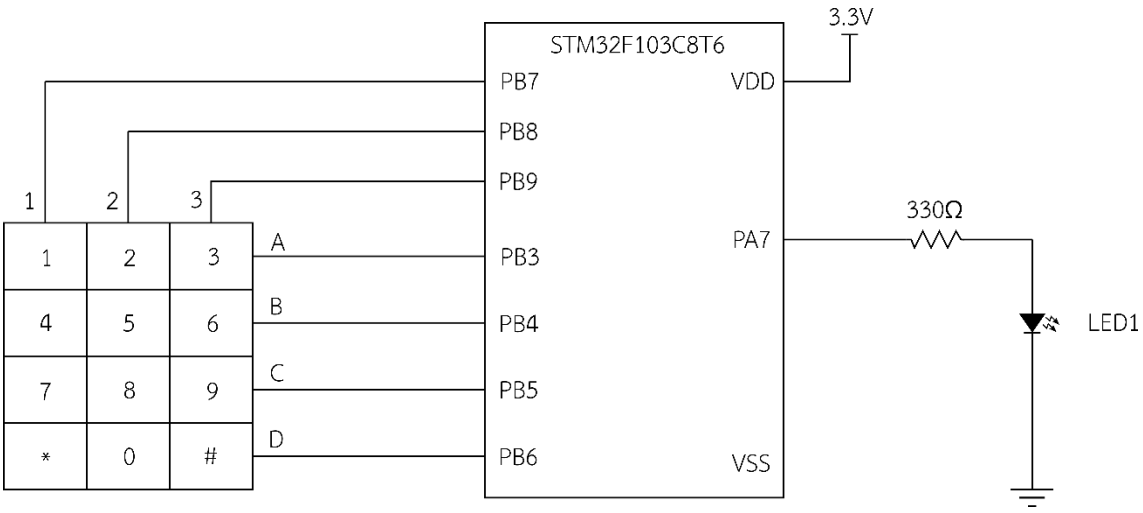
บรรทัดที่ 22 การเรียกใช้ฟังก์ชัน GetKey() เพื่ออ่านข้อมูลจากสวิตช์เมทริกซ์และเก็บค่าไว้ที่ตัวแปรชื่อ KeyData

บรรทัดที่ 23 - 28 การตรวจสอบข้อมูลที่ได้จากการกดสวิตช์เมทริกซ์และแสดงข้อมูลตัวเลข 0 - 9 ออกหลอดไฟ LED 7-segment

บรรทัดที่ 31 - 50 ฟังก์ชัน GetKey() เป็นฟังก์ชันอ่านข้อมูลจากสวิตช์เมทริกซ์ด้วยวิธีการสแกน โดยฟังก์ชัน GetKey() จะตอบกลับค่าลำดับตัวเลขของสวิตช์ที่กด ซึ่งมีค่าในช่วงระหว่าง 1 - 12 และตอบกลับค่า 0 เมื่อไม่มีการกดสวิตช์

บรรทัดที่ 51 - 50 ฟังก์ชัน seg_put() เป็นฟังก์ชันแสดงข้อมูลแบบตัวเลขออกหลอดไฟ LED 7-segment

ตัวอย่างที่ 3.8 การป้อนรหัสผ่าน 4 หลัก ด้วยสวิตช์เมทริกซ์ขนาด 4x3 ชนิดตัวเลขเพื่อเปิดหลอดไฟ LED .ให้ติดแบบค้างสภาวะเป็นเวลา 5 วินาที



รูปที่ 3.15 วงจรการทดลองป้อนรหัสผ่าน 4 หลัก เพื่อเปิดหลอดไฟ LED

การเขียนโปรแกรม

```
1   #define  LED1   PA7
2   byte password[4] = {1, 2, 3, 4};
3   byte count;
4   byte keys[4][3] = {{1,2,3}, {4,5,6}, {7,8,9}, {10,11,12}};
5   byte rowPins[4] = {PB3, PB4, PB5, PB6};
6   byte colPins[3] = {PB7, PB8, PB9};
7   byte KeyData;
8   byte GetKey(void);
9   void setup()
10  {
11          unsigned char i;
12          for(i=0;i<4;i++)
```

```
13              pinMode(rowPins[i], OUTPUT_OPEN_DRAIN);
14      for(i=0;i<3;i++)
15              pinMode(colPins[i], INPUT_PULLUP);
16      pinMode(LED1, OUTPUT);
17      count = 0;
18  }
19  void loop()
20  {
21      KeyData = GetKey();
22      if(KeyData != 0)
23      {
24              if(KeyData == password[count])
25              {
26                      count++;
27                      if(count == 4)
28                      {
29                              digitalWrite(LED1,1);
30                              delay(5000);
31                              digitalWrite(LED1,0);
32                              count = 0;
33                      }
34              }
35              else    count = 0;
36              while(KeyData != 0)
37                      KeyData = GetKey();
38      }
39      delay(100);
40  }
41  byte GetKey(void)
42  {
43      unsigned char r, c;
```

```
44        bool sw_status;
45        for(r=0; r<=3; r++)
46        {
47              digitalWrite(rowPins[r], 0);
48              for(c=0; c<=2; c++)
49              {
50                    sw_status = digitalRead(colPins[c]);
51                    if(sw_status == 0)
52                    {
53                          return  keys[r][c];
54                    }
55              }
56              digitalWrite(rowPins[r], 1);
57              delay(1);
58        }
59        return 0;
60  }
```

รายละเอียดของโปรแกรมบางส่วนที่เกี่ยวกับการอ่านข้อมูลจากสวิตช์เมทริกซ์ได้อธิบายไว้ในตัวอย่างที่ 3.8 จึงอธิบายเฉพาะส่วนของการป้อนรหัสและการตรวจสอบรหัส ดังนี้

บรรทัดที่ 2 การประกาศตัวแปรอาร์เรย์ ชื่อ password จำนวน 4 ตัว เก็บค่ารหัสผ่าน 4 หลัก

บรรทัดที่ 3 การประกาศตัวแปรชนิด byte ชื่อ count เพื่อใช้กำหนดลำดับของรหัสผ่าน 4 หลัก

บรรทัดที่ 21 การเรียกใช้ฟังก์ชัน GetKey() เพื่ออ่านข้อมูลจากสวิตช์เมทริกซ์และเก็บค่าไว้ที่ตัวแปรชื่อ KeyData

บรรทัดที่ 22 ตรวจสอบการข้อมูลที่ได้จากสวิตช์เมทริกซ์ว่ามีการกดสวิตช์หรือไม่

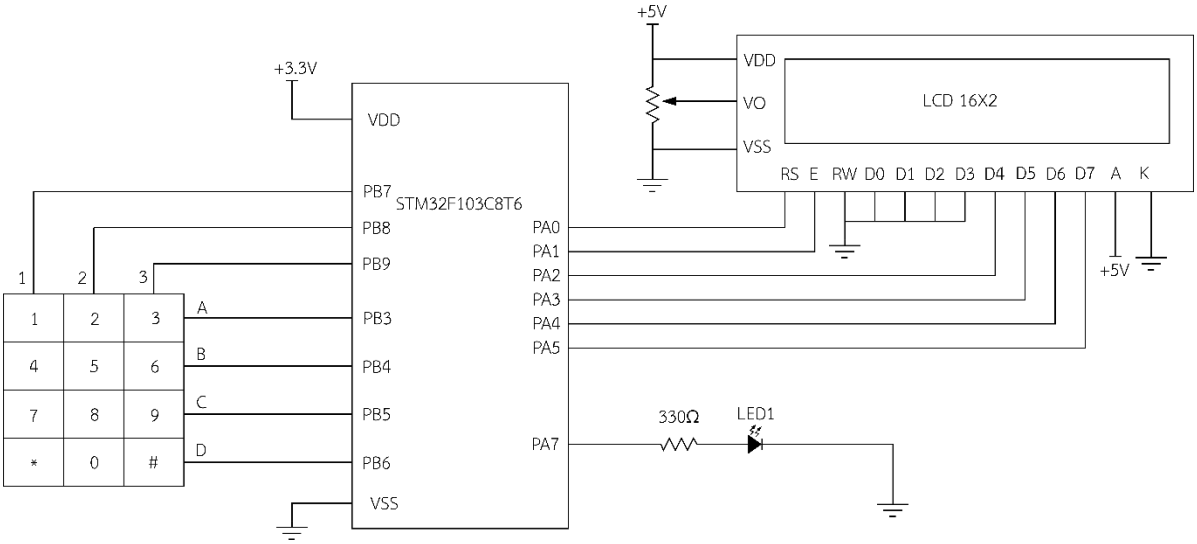
บรรทัดที่ 24 - 26 เปรียบเทียบข้อมูลที่ได้จากการกดสวิตช์เมทริกซ์กับรหัสที่กำหนดไว้ในตัวแปรอาร์เรย์ password ที่ละลำดับ ถ้าข้อมูลตรงกับรหัสที่กำหนดให้เพิ่มค่าของตัวแปร count เพื่อเลื่อนลำดับหลักของรหัสที่ใช้ในการเปรียบเทียบเมื่อมีการกดสวิตช์ในครั้งถัดไป

บรรทัดที่ 27 - 33 ตรวจสอบการกดรหัสตรงกับรหัสที่กำหนดครบ 4 หลัก หรือไม่ ถ้ากดรหัสตรงกับรหัสที่กำหนดครบ 4 หลัก สั่งให้หลอดไฟ LED1 ติดแบบค้างสภาวะเป็นเวลา 5 วินาที

บรรทัดที่ 35 เมื่อกดรหัสไม่ตรงกับรหัสที่กำหนดในแต่ละหลักให้รีเซ็ตค่าของตัวแปร count เป็น 0 เพื่อเริ่มนับลำดับของรหัสใหม่

บรรทัดที่ 36 - 37 การวนรอบซ้ำด้วยคำสั่ง while เพื่อตรวจสอบการปล่อยสวิตช์ก่อนกดรหัสในหลักถัดไป

ตัวอย่างที่ 3.10 การป้อนรหัสผ่าน 4 หลัก ด้วยสวิตช์เมทริกซ์ขนาด 4x3 ชนิดตัวเลขเพื่อเปิดหลอดไฟ LED ให้ติดแบบค้างสภาวะเป็นเวลา 5 วินาที และแสดงการกดรหัสผ่านที่จอแสดงผลแอลซีดี



รูปที่ 3.19 วงจรการทดลองการป้อนรหัสผ่าน 4 หลัก และแสดงการกดรหัสผ่านที่จอแสดงผลแอลซีดี

การเขียนโปรแกรม

```
1   #include <LiquidCrystal.h>
2   #define  LED1   PA7
3   LiquidCrystal lcd(PA0, PA1, PA2, PA3, PA4, PA5);
4   byte password[4] = {‘1’,  ‘2’, ‘3’, ‘4’};
5   byte keys[4][3] = {{‘1’, ‘2’, ‘3’}, {‘4’, ‘5’, ‘6’}, {‘7’, ‘8’, ‘9’}, {‘*’, ‘0’, ‘#’}};
6   byte rowPins[4] = {PB3, PB4, PB5, PB6};
7   byte colPins[3] = {PB7, PB8, PB9};
8   byte KeyData;
9   unsigned byte GetKey(void);
```

```
10  void setup()
11  {
12          lcd.begin(16,2);
13          lcd.clear();
14          lcd.setCursor(0,0);
15          byte i;
16          for(i=0;i<4;i++)
17                  pinMode(rowPins[i], OUTPUT_OPEN_DRAIN);
18          for(i=0;i<3;i++)
19                  pinMode(colPins[i], INPUT_PULLUP);
20          pinMode(LED1, OUTPUT);
21          count = 0;
22  }
23  void loop()
24  {
25          KeyData = GetKey();
26          if(KeyData != 0)
27          {
28                  lcd.write(KeyData);
29                  if(KeyData == password[count])
30                  {
31                          count++;
32                          if(count == 4)
33                          {
34                                  digitalWrite(LED1,1);
35                                  delay(5000);
36                                  digitalWrite(LED1,0);
37                                  count = 0;
38                          }
39                  }
40                  else
```

```
41              {
42                      lcd.clear();
43                      lcd.setCursor(0,0);
44                      count = 0;
45              }
46              while(KeyData != 0)
47                      KeyData = GetKey();
48      }
49      delay(100);
50  }
51  unsigned byte GetKey(void)
52  {
53      unsigned char r, c;
54      bool sw_status;
55      for(r=0; r<=3; r++)
56      {
57              digitalWrite(rowPins[r], 0);
58              for(c=0; c<=2; c++)
59              {
60                      sw_status = digitalRead(colPins[c]);
61                      if(sw_status == 0)
62                      {
63                              return  keys[r][c];
64                      }
65              }
66              digitalWrite(rowPins[r], 1);
67              delay(1);
68      }
69      return 0;
70  }
```

จากโปรแกรมสามารถอธิบายรายละเอียดได้ดังนี้

บรรทัดที่ 1 เรียกใช้ไฟล์ LiquidCrystal.h เข้ามาร่วมกับโปรแกรมหลักเพื่อใช้คำสั่งและฟังก์ชันเกี่ยวกับการใช้งานจอแสดงผลแอลซีดี

บรรทัดที่ 3 เรียกใช้คลาสฟังก์ชัน LiquidCrystal และตั้งชื่อคลาสเป็น lcd ส่งข้อมูลแบบ 4 บิต ใช้ขาสัญญาณจำนวน 6 ขา ได้แก่ ขา PA0 เป็นขาสัญญาณ RS ขา PA1 เป็นขาสัญญาณ E และขา PA2 - PA5 เป็นขาสัญญาณข้อมูล D4 - D7

บรรทัดที่ 4 การประกาศตัวแปรอาร์เรย์ ชื่อ password จำนวน 4 ตัว เก็บค่ารหัสผ่าน 4 หลัก

บรรทัดที่ 6 การใช้งานไมโครคอนโทรเลอร์ร่วมกับจอแสดงผลแอลซีดีขนาด 16x2

บรรทัดที่ 21 การเรียกใช้ฟังก์ชัน GetKey() เพื่ออ่านข้อมูลจากสวิตช์เมทริกซ์และเก็บค่าไว้ที่ตัวแปรชื่อ KeyData

บรรทัดที่ 22 ตรวจสอบการข้อมูลที่ได้จากสวิตช์เมทริกซ์ว่ามีการกดสวิตช์หรือไม่

บรรทัดที่ 24 - 26 เปรียบเทียบข้อมูลที่ได้จากการกดสวิตช์เมทริกซ์กับรหัสที่กำหนดไว้ในตัวแปรอาร์เรย์ password ที่ละลำดับ ถ้าข้อมูลตรงกับรหัสที่กำหนดให้เพิ่มค่าของตัวแปร count เพื่อเลื่อนลำดับหลักของรหัสที่ใช้ในการเปรียบเทียบเมื่อมีการกดสวิตช์ในครั้งถัดไป

บรรทัดที่ 27 - 33 ตรวจสอบการกดรหัสตรงกับรหัสที่กำหนดครบ 4 หลัก หรือไม่ ถ้ากดรหัสตรงกับรหัสที่กำหนดครบ 4 หลัก สั่งให้หลอดไฟ LED1 ติดแบบค้างสภาวะเป็นเวลา 5 วินาที

บรรทัดที่ 35 เมื่อกดรหัสไม่ตรงกับรหัสที่กำหนดในแต่ละหลักให้รีเซ็ตค่าของตัวแปร count เป็น 0 เพื่อเริ่มนับลำดับของรหัสใหม่

บรรทัดที่ 36 - 37 การวนรอบซ้ำด้วยคำสั่ง while เพื่อตรวจสอบการปล่อยสวิตช์ก่อนกดรหัสในหลักถัดไป