

รายงาน Assignment

วิชา 240-228 Digital Microcontrollers

ชื่อ นกฤษ สกุล กริไชยชนะ

รหัสนักศึกษา 6510110212

Section 02

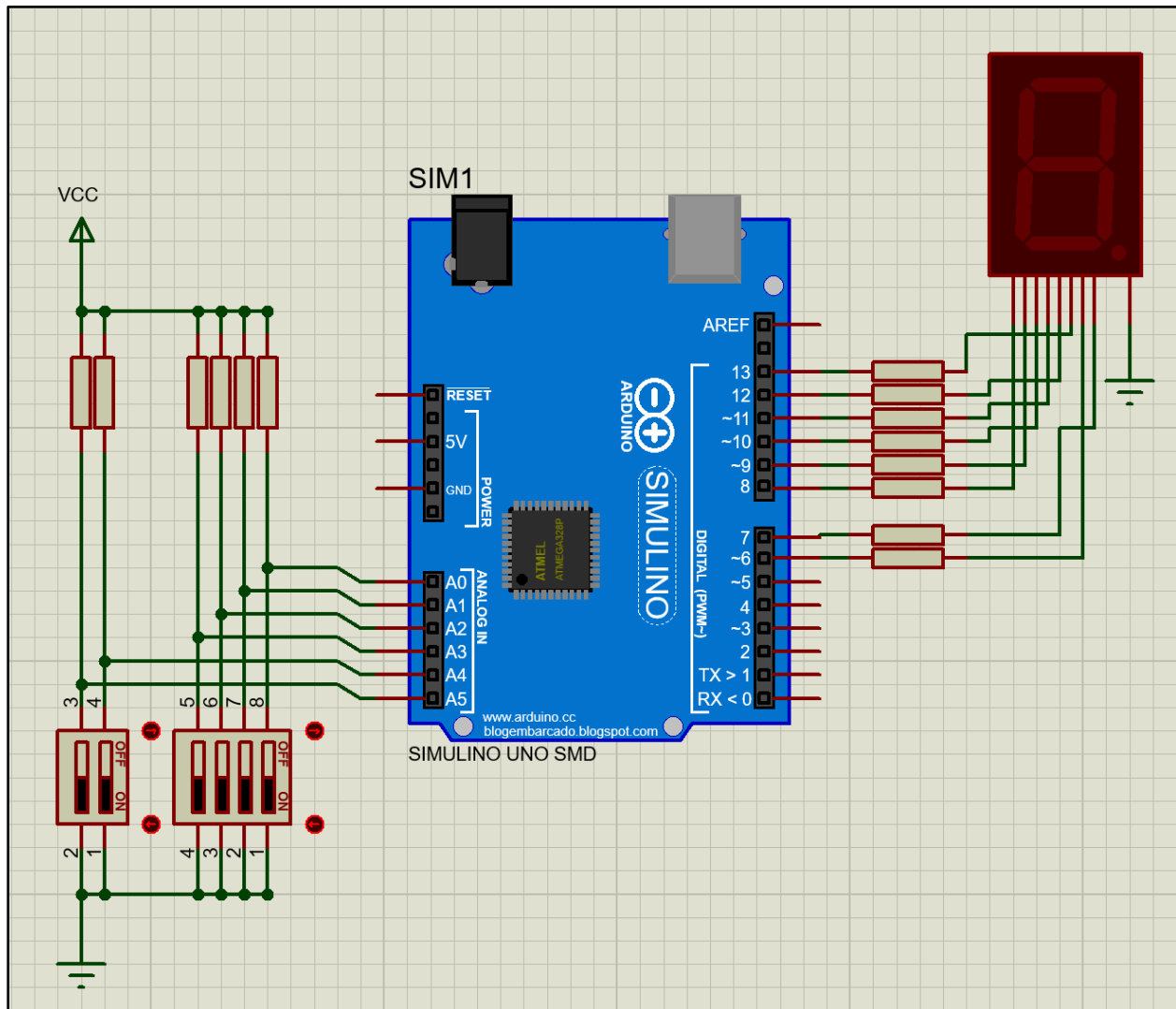
หัวข้อ Assignment

ออกแบบวงจรเพื่อประยุกต์ใช้งาน AVR

สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยสงขลานครินทร์

## Schematic Diagram



ออกแบบวงจรเพื่อประยุกต์ใช้งาน AVR กำหนดให้วงจรจะต้องมีคุณสมบัติดังนี้

- ✓ ต้องมีการใช้งานพอร์ตของเอวีอาร์อย่างน้อย 2 พอร์ต (14-Ports)
- ✓ มีการรับอินพุตอย่างน้อย 5 บิต (6-bit)
- ✓ มีการส่งเอาต์พุตอย่างน้อย 5 บิต (8-bit)
- ✓ มีการรับอินพุตด้วยวิธีการอินเตอร์รัพอย่างน้อย 1 บิต (2-bit-polling)

วิดีโอการทำงานจริง (INC, DEC) => <https://youtu.be/40ZCoGelZ7Y?feature=shared>

Assembly Code

```

;=====
.def MODE = R18                ; สำหรับเก็บสถานะจาก PINC
.def TMP1 = R20                ; สำหรับกำหนด I/O port
.def TMP2 = R21                ; สำหรับอ่านค่า BCH จาก Switch
.def INCDEC = R22              ; สำหรับบันทึกค่าเลขปัจจุบัน
.def INPUT_BCH = R25           ; ใช้เป็นพารามิเตอร์นำเข้า
.def OUTPUT7SEG = R25         ; ใช้เป็นพารามิเตอร์ส่งออก
;=====

.cseg                          ; เริ่มการทำงานแบบ CodeSegment
.org 0x0000                    ; ให้ Pointer ชี้ไปที่ตำแหน่ง 0x0000 ใน memory
;=====

start:

    ldi    TMP1, 0b00111111    ; ตั้งจำนวน port-b ที่จะใช้ (a, b, c, d, e, f)
    out    DDRB, TMP1
    ldi    TMP1, 0b11000000    ; ตั้งจำนวน port-d ที่จะใช้ (g, dp)
    out    DDRD, TMP1
    ldi    TMP1, 0b11000000    ; ตั้งจำนวน port-c ที่จะใช้ (PC0-PC3 as Input-BCH),
    out    DDRC, TMP1          ; (PC4-PC5 as Input-Mode)
    eor    R25, R25            ; กำหนดค่าเริ่มต้นของ INPUT, OUTPUT เป็น 0
    eor    R22, R22            ; กำหนดค่าเริ่มต้นของ INC, DEC เป็น 0
;=====

loop:

    rcall  MCHECK              ; ตรวจสอบสถานะ Switch-2
    rcall  SELECT              ; เลือก mode การทำงาน
    rjmp   loop
;=====

MCHECK:

    in     MODE, PINC          ; บันทึกค่า SW2 จาก PD4
    lsr    MODE                ; เลื่อนบิตไปขวา
    lsr    MODE                ; เลื่อนบิตไปขวา
    lsr    MODE                ; เลื่อนบิตไปขวา
    lsr    MODE                ; เลื่อนบิตไปขวา
    andi   MODE, 0x03          ; กรองให้เหลือแค่ INPUT ของ Mode (PC4-PC5)
    ret
;=====

```

```

SELECT:
    cpi    MODE,          0x00    ; Switch-2 ปิดหมด
    breq   READ_SW        ; อ่านค่า BCH จาก Switch-1
    rcall  INC_MODE        ; เช็คว่าสถานะ Switch-2 เป็นอะไร
    rjmp   loop

INC_MODE:
    cpi    MODE,          0x01    ; Switch-2 เปิดอันแรก
    breq   INCREASE        ; แสดงการเพิ่มขึ้นจาก 0-F ทีละ 1
    rcall  DEC_MODE        ; เช็คว่าสถานะ Switch-2 เป็นอะไร
    rjmp   loop

DEC_MODE:
    cpi    MODE,          0x02    ; Switch-2 เปิดอันที่สองเปิด
    breq   DECREASE        ; แสดงการเพิ่มลดจาก F-0 ทีละ 1
    rcall  READ_SW        ; Switch-2 เปิดทุกอันให้อ่านค่า BCH จาก Switch-1
    rjmp   loop

READ_SW:
    rcall  INPUT_FILTER    ; กรองเอาแค่ PB0-PB3
    rcall  Display          ; แสดงผลไปยัง 7-SEG
    rjmp   READ_SW

;=====

INPUT_FILTER:
    in     TMP2,           PINC    ; นำค่าที่อ่านจาก PC0-PC3 มากรองแค่ 4 บิตแรก
    ldi    INPUT_BCH,      0x0F    ; เติม 0b00001111
    and    INPUT_BCH,      TMP2    ; ลบค่าบิตสูงออก
    ret

;=====

INCREASE:
    cpi    INCDEC,         0x0F    ; ตรวจสอบว่าปัจจุบันเกิน F ไหม
    breq   INC_REST        ; มากกว่า F ทำการรีเซ็ตค่า
    inc    INCDEC           ; น้อยกว่า F ให้เพิ่ม +1
    mov    INPUT_BCH,      INCDEC  ; อัปเดตเลขปัจจุบัน
    rjmp   Display          ; นำผลลัพธ์ไปแสดงบน 7-Seg

INC_REST:
    ldi    INCDEC,         0x00    ; เปลี่ยนให้เป็น 0
    mov    INPUT_BCH,      INCDEC  ; อัปเดตเลขปัจจุบัน
    rjmp   Display          ; นำผลลัพธ์ไปแสดงบน 7-Seg

;=====

```

```

DECREASE:
    cpi    INCDEC,      0x00    ; ตรวจสอบว่าปัจจุบันเป็น 0 ไหม
    breq   DEC_REST      ; มากกว่า 0 ทำการรีเซ็ตค่า
    dec    INCDEC          ; น้อยกว่า 0 ให้ลด -1
    mov    INPUT_BCH,     INCDEC ; อัปเดตเลขปัจจุบัน
    rjmp   Display        ; นำผลลัพธ์ไปแสดงบน 7-Seg

DEC_REST:
    ldi    INCDEC, 0x0F      ; เปลี่ยนให้เป็น F
    mov    INPUT_BCH,     INCDEC ; อัปเดตเลขปัจจุบัน
    rjmp   Display        ; นำผลลัพธ์ไปแสดงบน 7-Seg

;=====

Display:
    rcall  DISP_7SEG        ; แสดงผลลัพธ์บน 7-Seg
    rcall  DELAY_500MS      ; หน่วงเวลา 0.5 วินาที
    rjmp   loop

;=====

DISP_7SEG:
    call   BIN_TO_7SEG      ; แปลง bits เป็นสัญญาณขาของ 7-Seg
    out    PORTD, OUTPUT7SEG ; ส่งสัญญาณขา PB (a - f)
    out    PORTB, OUTPUT7SEG ; ส่งสัญญาณขา PD (g, dp)
    ret

;=====

BIN_TO_7SEG:
    push   ZL               ; เก็บค่าของเรจิสเตอร์ ZL ไว้ในสแต็ก
    push   ZH               ; เก็บค่าของเรจิสเตอร์ ZL ไว้ในสแต็ก
    push   R0               ; เก็บค่าของเรจิสเตอร์ R0 ไว้ในสแต็ก
    sub    R0,              R0 ; ตั้งให้เรจิสเตอร์ R0 มีค่าเท่ากับศูนย์
    rjmp   LOOK_TABLE       ; กระโดดข้ามตารางค้นหาไปยังป้าย LOOK_TABLE

;=====

;---ส่วนนี้ของโปรแกรมเป็นการเก็บตารางค้นหาของค่ารหัสการติดดับของแอลอีดีชนิด 7-Segment---;
TB_7SEG: .DB 0b00111111, 0b00000110    ; 0 และ 1      ----a----
        .DB 0b01011011, 0b01001111    ; 2 และ 3      f      b
        .DB 0b01100110, 0b01101101    ; 4 และ 5      ----g----
        .DB 0b01111101, 0b00000111    ; 6 และ 7      e      c
        .DB 0b01111111, 0b01101111    ; 8 และ 9      ----d----
        .DB 0b01110111, 0b01111100    ; A และ B
        .DB 0b00111001, 0b01011110    ; C และ D
        .DB 0b01111001, 0b01110001    ; E และ F
        .DB 0b01001001, 0b00110110    ; special value

;=====

```

LOOK\_TABLE:

```
ldi    ZL,    low(TB_7SEG*2) ; บรรจุค่าตำแหน่งไบต์ต่ำของ TB_7SEG ใส่ ZL
ldi    ZH,    high(TB_7SEG*2) ; บรรจุค่าตำแหน่งไบต์สูงของ TB_7SEG ใส่ ZH
add    ZL,    INPUT_BCH      ; บวกค่า ZL ด้วยค่ารหัสบีซีดีอินพุต
adc    ZH,    R0              ; บวกค่าที่อาจมีการทดใน Carry ใส่ใน ZH
lpm                                ; อ่านหน่วยความจำโปรแกรมที่ Z ซ้ำอยู่ใน R0
mov    OUTPUT7SEG, R0        ; ส่งค่าใน R0 ไปยังเรจิสเตอร์สำหรับคีนค่า
pop    R0                    ; อ่านค่าเก่าของ R0 คีนมาจากสแต็ก
pop    ZH                    ; อ่านค่าเก่าของ ZH คีนมาจากสแต็ก
pop    ZL                    ; อ่านค่าเก่าของ ZL คีนมาจากสแต็ก
ret
```

=====

-----จับรูทีนสำหรับหน่วยเวลา 10 มิลลิวินาที (CPU 16 MHz)-----;

DELAY10MS:

```
push    R16
push    R17
ldi     R16,    0x00
LOOP2:  inc     R16
ldi     R17,    0x00
LOOP1:  inc     R17
cpi     R17,    249
brlo    LOOP1
nop
cpi     R16,    160
brlo    LOOP2
pop     R17
pop     R16
ret
```

=====

-----จับรูทีนสำหรับหน่วยเวลา 500 มิลลิวินาที (CPU 16 MHz)-----;

DELAY\_500MS:

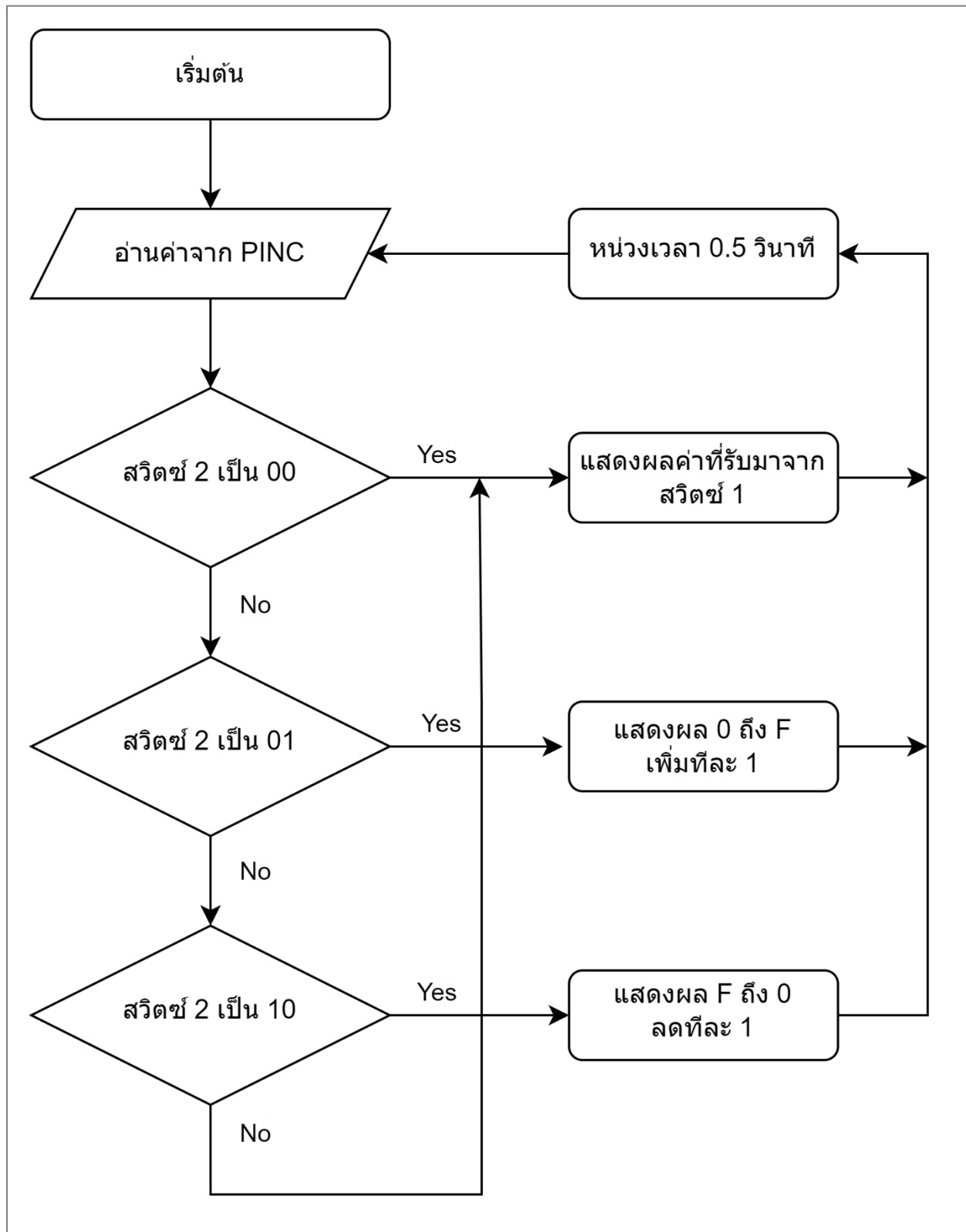
```
ldi     R16, 50 ; 100 x 10 milliseconds = 500 milliseconds
```

delay\_loop:

```
rcall   DELAY10MS ; เรียกใช้จับรูทีน DELAY10MS
dec     R16        ; ลดค่า R16 ลงทีละ 1
brne    delay_loop ; เมื่อครบรอบตามที่กำหนดเป็นการจบ Delay
ret
```

=====

## Flowcharts



### การทำงานของโปรแกรมควบคุมอย่างละเอียด

เมื่อเริ่มต้นจะกำหนด Register ที่ใช้ 4 ตัวคือ

- R18 = MODE = สำหรับเก็บสถานะจาก PINC
- R20 = TMP1 = สำหรับกำหนด I/O port
- R21 = TMP2 = สำหรับอ่านค่า BCH จาก Switch
- R22 = INCDEC = สำหรับบันทึกค่าเลขใน Subroutine INC, DEC
- R25 = INPUT\_BCH = OUTPUT7SEG = ใช้เป็นพารามิเตอร์นำเข้าและส่งออก 7-Segment

จากนั้นจะโหลดค่าเข้าไปใน R20 เพื่อกำหนดการใช้งาน PORT-C ,PORT-B, PORT-D และทำการเคลียร์ค่าของ R25, R22 ก่อนจะเริ่มเข้าสู่การทำงาน เมื่อเข้ามาถึง Loop จะเริ่มจากการตรวจสอบค่าจาก Switch-2 ที่เชื่อมต่อกับ PC4, PC5 โดยที่สองพอร์ตนี้นี้จะทำหน้าที่เป็นอินเทอร์รัพต์แบบ Polling ขนาด 2 บิตที่จะเปลี่ยนโหมดการทำงานได้ทั้งหมด 3 แบบ ในการเลือกโหมดการทำงาน MCHECK จะรับค่าจาก PINC มาทั้งหมดก่อนจะทำการเลื่อน BIT ทั้งหมดไปทางขวาสี่ครั้ง เนื่องจาก Switch-2 นั้นทำงานอยู่ในบิตสูง แล้วบันทึกค่า Switch-2 เก็บไว้ใน R18 แล้วกลับไปยัง Loop จากนั้นเรียกใช้ SELECT เพื่อนำค่าจาก R18 ไปเทียบว่าจะได้ไปทำงานต่อที่ Subroutine ไหน

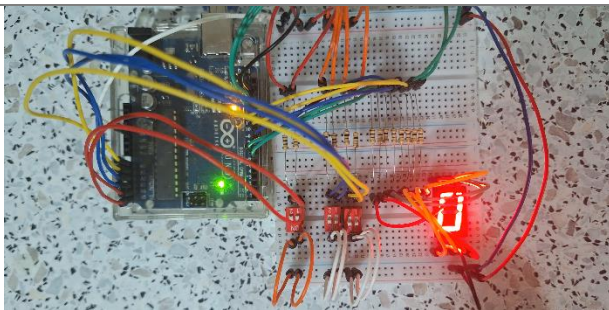
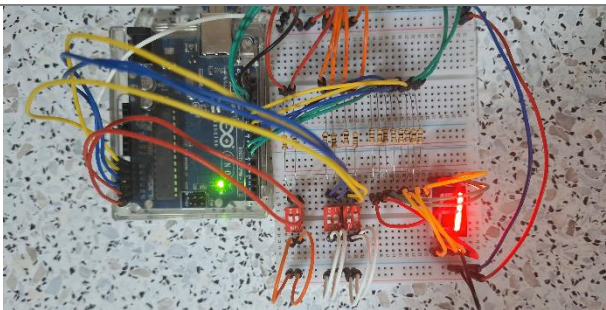
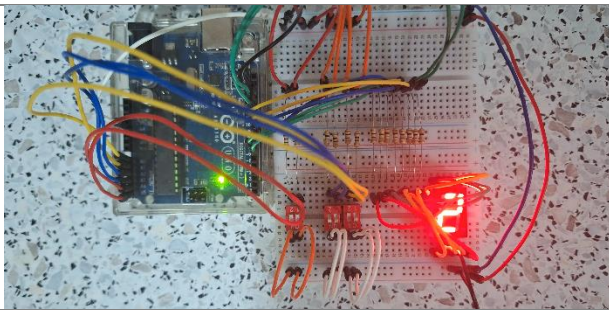
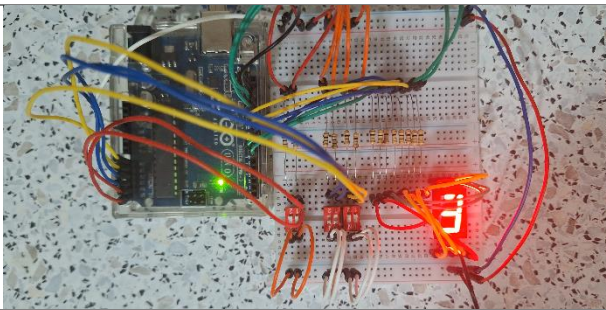
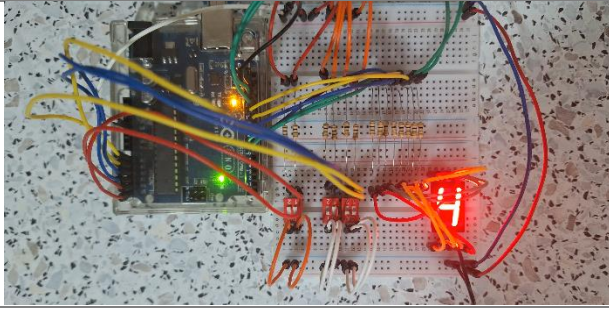
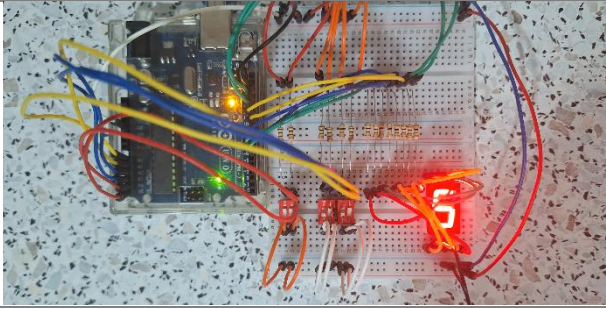
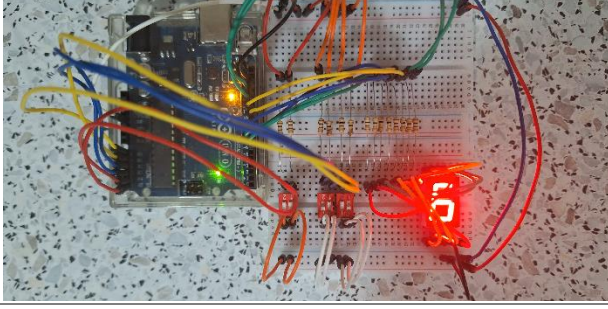
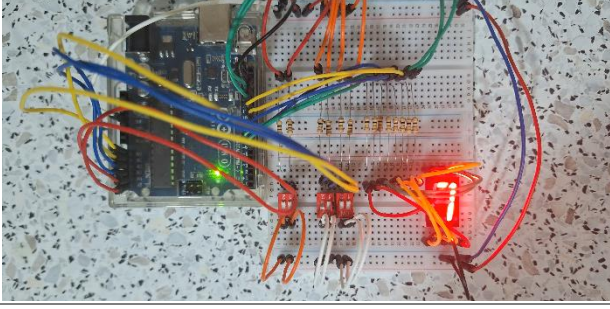
- ถ้า R18 (MODE) = 0x00 จะถูก branch ไปที่ READ\_SW ถ้าไม่จะถูก Branch ไปยัง INC\_MODE
- ถ้า R18 (MODE) = 0x01 จะถูก branch ไปที่ INCREASE ถ้าไม่จะถูก Branch ไปยัง DEC\_MODE
- ถ้า R18 (MODE) = 0x02 จะถูก branch ไปที่ INCREASE ถ้าไม่จะถูก Branch ไปยัง READ\_SW

ในกรณีที่ถูก Branch ไปที่ READ\_SW ค่าที่อ่านได้จาก PINC จะถูกกรองด้วย INPUT\_FILTER ที่ค่าของ PINC จะถูก Copy ไปยัง R21 จากนั้นจะถูก AND ด้วย 0x0F ใน R25 เพื่อกรองเอาแค่บิตต่ำส่งไปยัง Display เพื่อแสดงผลบน 7-Segments แล้วกลับไป Loop อีกครั้ง

ในกรณีที่ถูก Branch ไปที่ INCREASE หรือ DECREASE จะไม่มีการอ่าน PINC แต่ค่าที่นับจะถูกเก็บไว้ใน R22 ในตอนเริ่มต้น Subroutine จะตรวจสอบก่อนว่า R22 นั้นเป็นค่าที่ตรงเงื่อนไขการ RESET ใหม่ถ้าใช่ R22 ก็จะถูก RESET เป็น 0 หรือ F แล้วส่งไปยัง R25 เพื่อแสดงผลบน 7-Segments แล้วกลับไป Loop อีกครั้ง แต่ถ้าไม่เข้าเงื่อนไข R22 ก็จะถูกเพิ่มค่าหรือลดค่าทีละ 1 แล้วส่งไปยัง R25 เพื่อแสดงผลบน 7-Segments จนกระทั่ง R22 ตรงกลับเงื่อนไขการ RESET หรือไม่ก็ MODE นั้นถูกเปลี่ยนไป โดยที่ค่าใน R22 จะยังคงเดิม ทำให้สามารถเห็นค่าเดิมเพิ่มหรือลดได้อย่างต่อเนื่อง

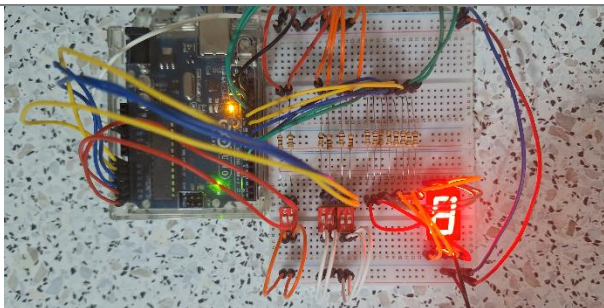
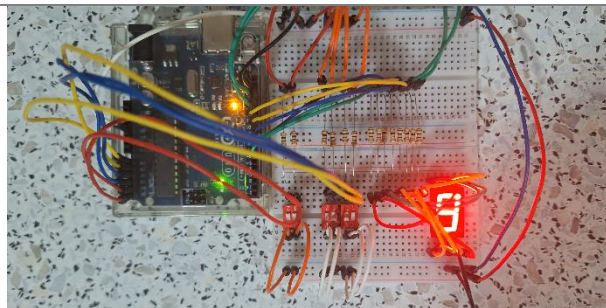
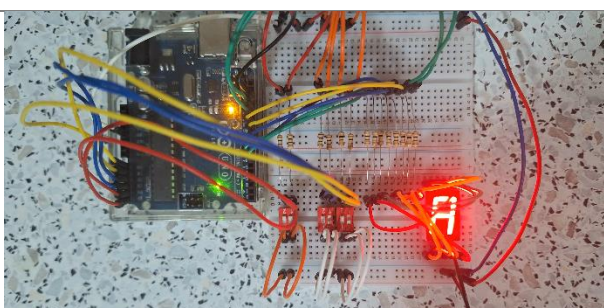
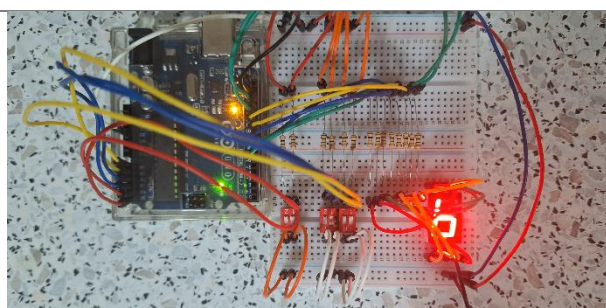
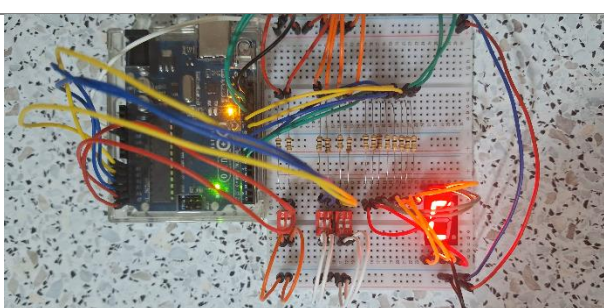
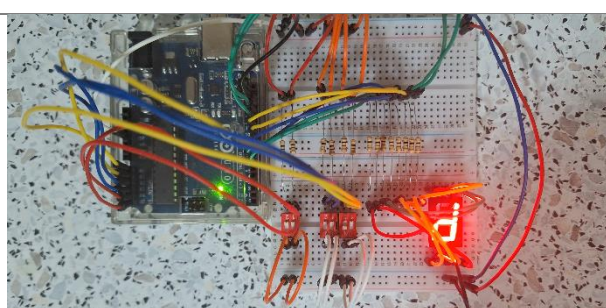


ผลการทำงานของวงจรจริง

Input = 0b00000000	Input = 0b00000001
	
Input = 0b00000010	Input = 0b00000011
	
Input = 0b00000100	Input = 0b00000101
	
Input = 0b00000110	Input = 0b00000111
	



ผลการทำงานของวงจรจริง

Input = 0b00001000	Input = 0b00001001
	
Input = 0b00001010	Input = 0b00001011
	
Input = 0b00001100	Input = 0b00001101
	
Input = 0b00001110	Input = 0b00001111
