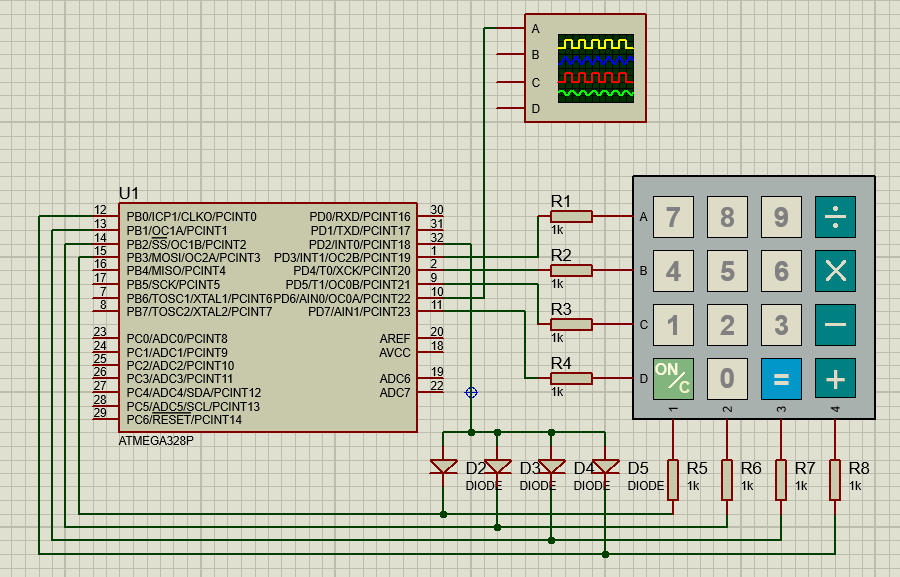
# ASSIGNMENT 2: วงจร Music Keypad

## อุปกรณ์

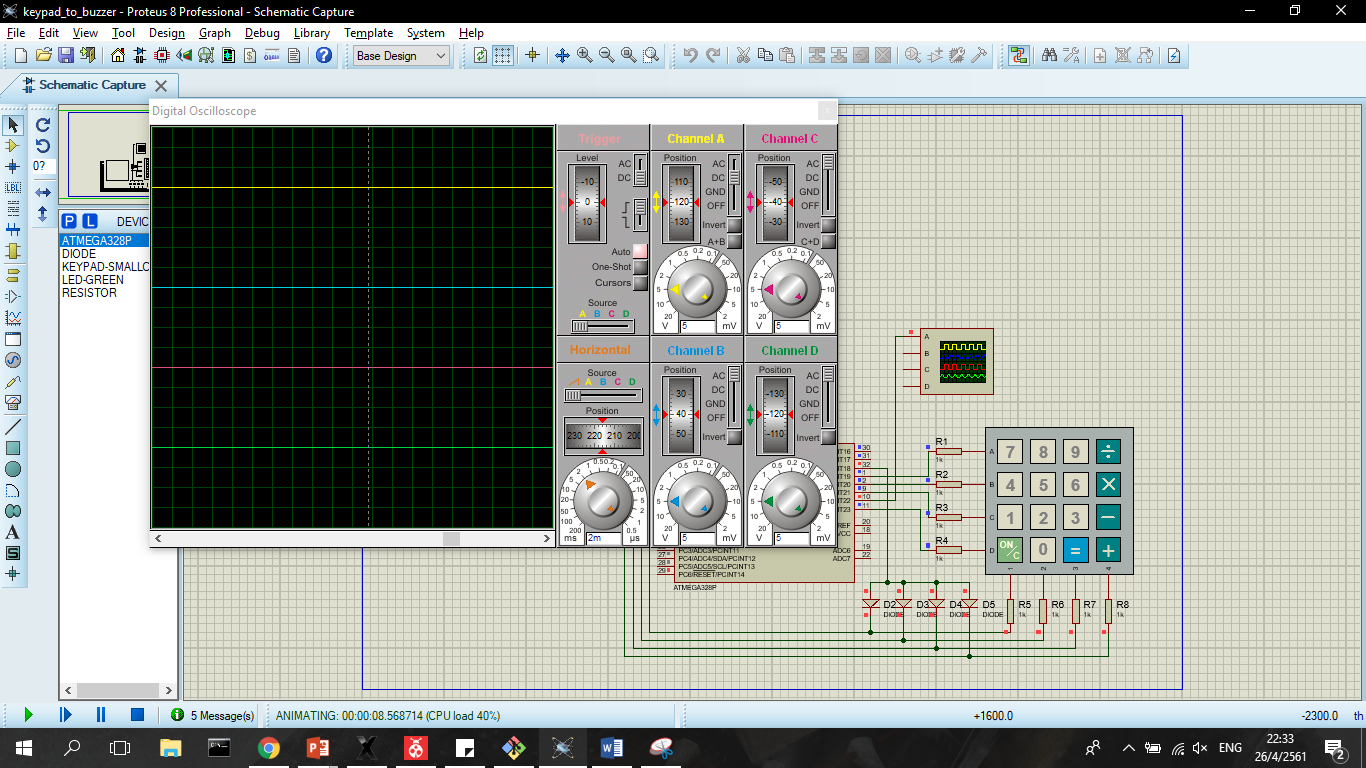
1. บอร์ด Arduino UNO R3 X 1
2. Breadboard X 1
3. Buzzer X 1
4. 4 X 4 Keypad X 1
5. Diode X 4
6. สายไฟ X 19

## การจำลองด้วย Proteus

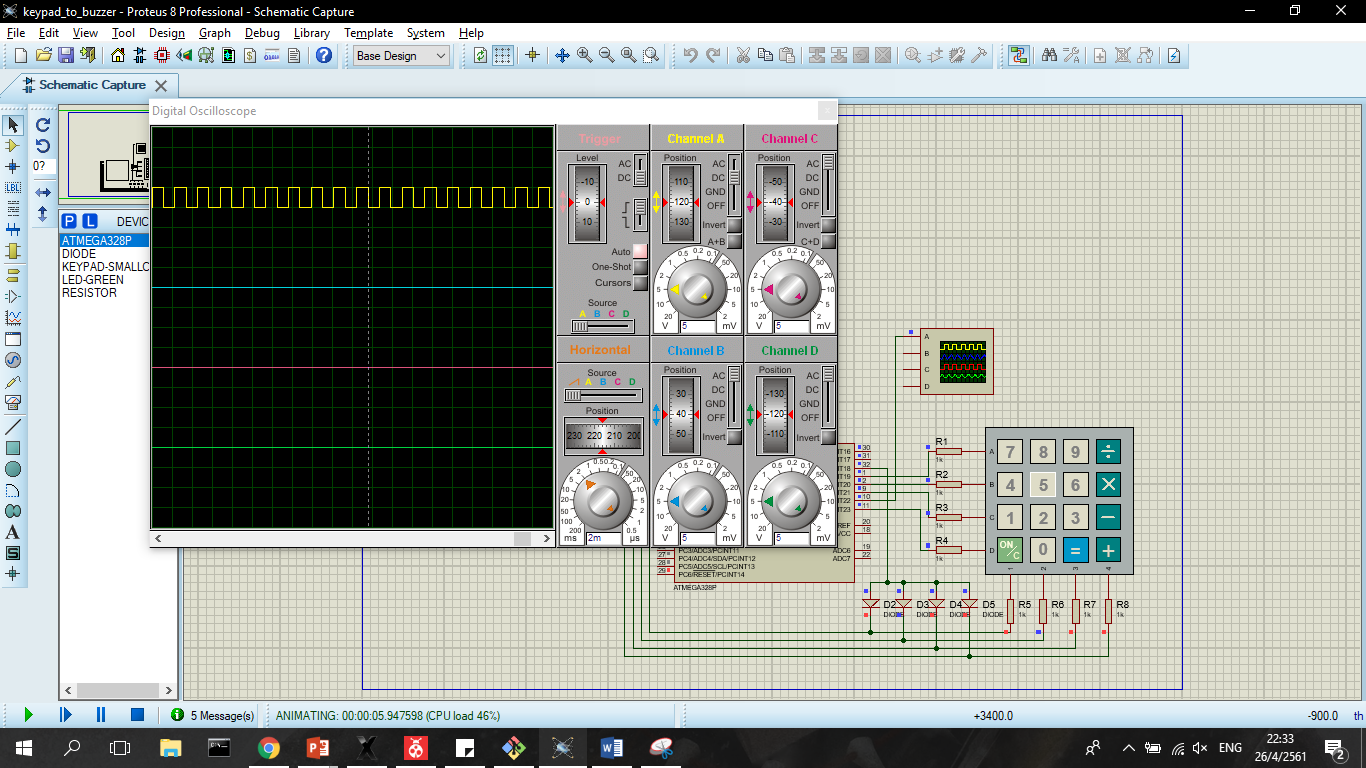


## ทดสอบการทำงานของวงจรใน Proteus

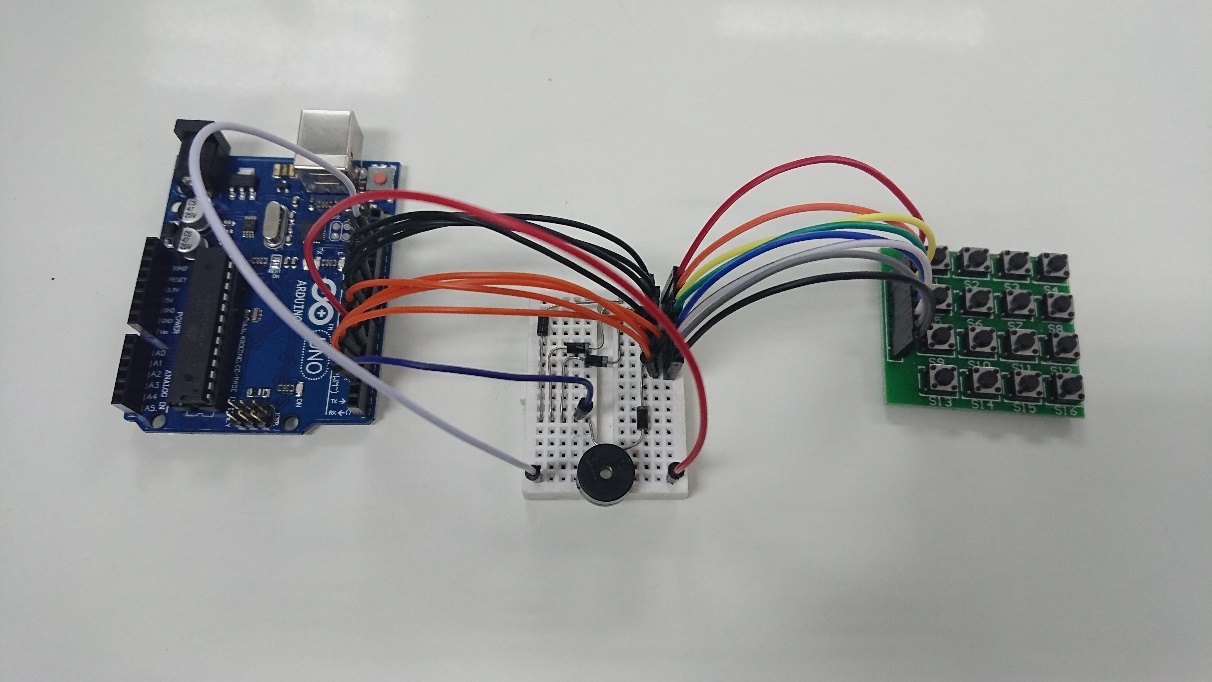
* เมื่อไม่ได้กด หรือ ปล่อยปุ่ม



* เมื่อกดปุ่ม



## ลักษณะวงจรจริง



## การทำงานของวงจร

1. การกดปุ่มหรือปล่อยปุ่มบน keypad จะเกิดอินเตอร์รัพท์ไปยังขา PD2 หรือ INT0
2. การทำงานเมื่อเกิดการอินเตอร์รัพท์คือการอ่านค่าจาก keypad โดยจะส่งค่าลอจิกต่ำ (logic 0) ไปยังแถวแต่ละแถวของ keypad และอ่านค่าคอลัมน์เข้ามา
3. ค่าที่อ่านจากคอลัมน์ของ keypad จะนำไปเปรียบเทียบหาค่าในตาราง LOOKUPTB ที่สร้างขึ้นเพื่อนำไปใช้ในการคำนวณต่อไป
4. ค่าที่ได้จาก LOOKUPTB จะถูกนำไปพิจารณาว่าเป็นค่า 0xFF หรือไม่ เนื่องจาก 0xFF เป็นค่าที่หมายถึงไม่มีการกดเกิดขึ้น การทำงานในขั้นตอนนี้สามารถแบ่งเป็นกรณีการทำงานได้ 2 กรณีดังนี้
   1. กรณีเท่ากับ 0xFF ก็จะทำการอ่านค่าจากแถวต่อไป
   2. กรณีไม่เท่ากับ 0xFF ก็จะทำการตั้งค่าให้ Timer0 ทำงานในโหมด CTC เปิดการอินเตอร์รัพท์จาก Timer0 และตั้งค่า OCR0A ตามค่าที่อ่านได้จาก LOOKUPTB เพื่อเป็นค่าที่ Timer0 นับจากศูนย์และเมื่อถึงค่า OCR0A ก็จะเกิดอินเตอร์รัพท์ Compare Match A การอินเตอร์รัพท์ก็จะทำการ toggle ขา OC0A หรือขา PD6 และเกิดเป็นพัลส์ที่มีความถี่แตกต่างกันในการกดแต่ละปุ่มเนื่องจากค่าที่ได้จาก LOOKUPTB ที่แตกต่างกัน
5. ในกรณีที่อ่านค่าจาก keypad ทั้งหมดแล้วไม่พบการกดหรือเป็นกรณีที่เป็นการปล่อยปุ่ม จะมีการตั้งค่าให้ Timer0 หยุดการทำงานและปิดการอินเตอร์รัพท์จาก Timer0 เพื่อไม่ให้เกิด pulse ออกทางขา OC0A หรือ PD6
6. พัลส์ที่ส่งออกทาง OC0A หรือ PD6 จะถูกนำไปขับ buzzer เพื่อให้เกิดเสียงต่าง ๆ ตามค่าที่ได้ตั้งไว้

## การออกแบบการทำงาน

การคำนวณเพื่อให้ Timer0 สามารถสร้าง pulse ตามความถี่ที่ต้องการในโหมด CTC คำนวณดังนี้

สมการที่ 1

ค่า OCR0A คือค่าที่ต้องเซ็ตให้เพื่อให้ Timer0 นับจนถึงค่านี้แล้วจะเกิด Compare Match A Interrupt

F\_CPU คือความถี่ของหน่วยประมวลผลหรือในทีนี้คือความถี่ของ AVR ATmega328p

f คือความถี่ที่ต้องการ

N คือค่า Prescaler ที่จะนำมาใช้เพื่อหารลงเพื่อให้ค่า OCR0A มีค่าไม่เกิน 255

จากอุปกรณ์ต่าง ๆ งานชิ้นนี้ได้สมการคำนวณค่า OCR0A ดังนี้

สมการที่ 2

เนื่องจากการคำนวณหาค่า OCR0A จะนำไปสร้างเป็นตาราง LOOKUPTB ต่อไปเพื่อนำไปใช้สร้างความถี่ที่ขา OC0A ค่าความถี่ที่เลือกใช้จะเป็นความถี่ของเสียงโน้ตเพลง โด เร มี ฟา ซอล ลา ทีและอื่น ๆ ดังนั้นจึงเลือกใช้ค่าในการคำนวณ OCR0A ดังสมาการที่ 2 โดยค่าความถี่และค่า OCR0A ที่คำนวณได้ดังตารางต่อไปนี้

**ตารางที่ 1** ตารางค่าความถี่ของโน้ตเพลงและค่า OCR0A ที่คำนวณได้

|  |  |  |  |
| --- | --- | --- | --- |
| **เสียง** | **ความถี่ Hz** | **คำนวณได้** | **OCR0A** |
| โด | 262 | 118.27 | 118 |
| เร | 294 | 105.29 | 105 |
| มี | 330 | 93.69 | 94 |
| ฟา | 349 | 88.54 | 89 |
| ซอล | 392 | 78.71 | 79 |
| ลา | 440 | 70.02 | 70 |
| ที | 494 | 62.26 | 62 |
| โด ^ | 523 | 58.75 | 59 |
| **เสียง** | **ความถี่ Hz** | **คำนวณได้** | **OCR0A** |
| เร ^ | 587 | 52.24 | 52 |
| มี ^ | 659 | 46.42 | 46 |
| ฟา ^ | 698 | 43.77 | 44 |
| ซอล ^ | 748 | 40.77 | 39\* |
| ลา ^ | 880 | 34.51 | 35 |
| ที ^ | 988 | 30.63 | 31 |
| โด ^^ | 1047 | 28.85 | 29 |
| เร ^^ | 1175 | 25.60 | 26 |

\* โน้ตเสียงซอลจำเป็นต้องลดลงเหลือ 39 เนื่องจากเสียงที่ได้เมื่อใช้เป็น 41 หรือ 40 ทำให้ได้เสียงที่ไม่ตรง

## โค้ดภาษาแอสแซมบลี

.INCLUDE "m328pdef.inc"

.DEF tmp = r16

.DEF distance= r17

.DEF row = r18

.DEF col = r19

.DEF value = r20

.CSEG

.ORG 0x00

jmp start

jmp INT0\_handler

start:

; SETUP STACK POINTER

ldi tmp, low(RAMEND)

out SPL, tmp

ldi tmp, high(RAMEND)

out SPH, tmp

; PORTD

; 7, 5, 4, 3 OUTPUT row scan

; 2 INPUT INT0 interrupt

; 6 OUTPUT OC0A

ldi tmp, 0xFB

out DDRD, tmp

ldi tmp, 0x04

out PORTD, tmp

; PORTB 3, 2, 1, 0 INPUT column read

ldi tmp, 0xF0

out DDRB, tmp

ldi tmp, 0x0F

out PORTB, tmp

; SETUP INTERRUPT

ldi tmp, 0x01

sts EICRA, tmp

out EIMSK, tmp

; SETUP TIMER0

ldi tmp, 0

out TCNT0, tmp

out OCR0A, tmp

ldi tmp, 0b01000010;

out TCCR0A, tmp ; CTC mode

ldi tmp, 0b00000000;

out TCCR0B, tmp ; First no source clk

ldi tmp, 0x00

sts TIMSK0, tmp ; Disable Timer0 ComMatchA Interrupt

sei

main:

rjmp main

.MACRO KEYPRESSED

out OCR0A, value

ldi tmp, 0b00000100; ; Set CLK/256

out TCCR0B, tmp

ldi tmp, 0x02

sts TIMSK0, tmp ; Enable Timer0 ComMatchA Interrupt

.ENDMACRO

.MACRO SCAN

ldi ZL, low(LOOKUPTB \* 2)

ldi ZH, high(LOOKUPTB \* 2)

out PORTD, row

nop

nop

in col, PINB

andi col, 0x0F

add col, distance

ldi tmp, 0x00

subi col, 0x07

add ZL, col

adc ZH, tmp

lpm value, Z

.ENDMACRO

.MACRO READ\_KEY\_PAD

ROW1:

ldi row, 0b11110111

ldi distance, 0

SCAN

cpi value, 0xFF

breq ROW2

KEYPRESSED

rjmp Finish

ROW2:

ldi row, 0b11101111

ldi distance, 9

SCAN

cpi value, 0xFF

breq ROW3

KEYPRESSED

rjmp Finish

ROW3:

ldi row, 0b11011111

ldi distance, 18

SCAN

cpi value, 0xFF

breq ROW4

KEYPRESSED

rjmp Finish

ROW4:

ldi row, 0b01111111

ldi distance, 27

SCAN

cpi value, 0xFF

breq OFF

KEYPRESSED

rjmp Finish

OFF:

ldi tmp, 0b00000000; ; Set CLK OFF

out TCCR0B, tmp

ldi tmp, 0x00

sts TIMSK0, tmp ; Disable Timer0 ComMatchA Interrupt

Finish:

.ENDMACRO

INT0\_handler:

push tmp

in tmp, SREG

push tmp

READ\_KEY\_PAD

ldi tmp, 0x04

out PORTD, tmp

pop tmp

out SREG, tmp

pop tmp

reti

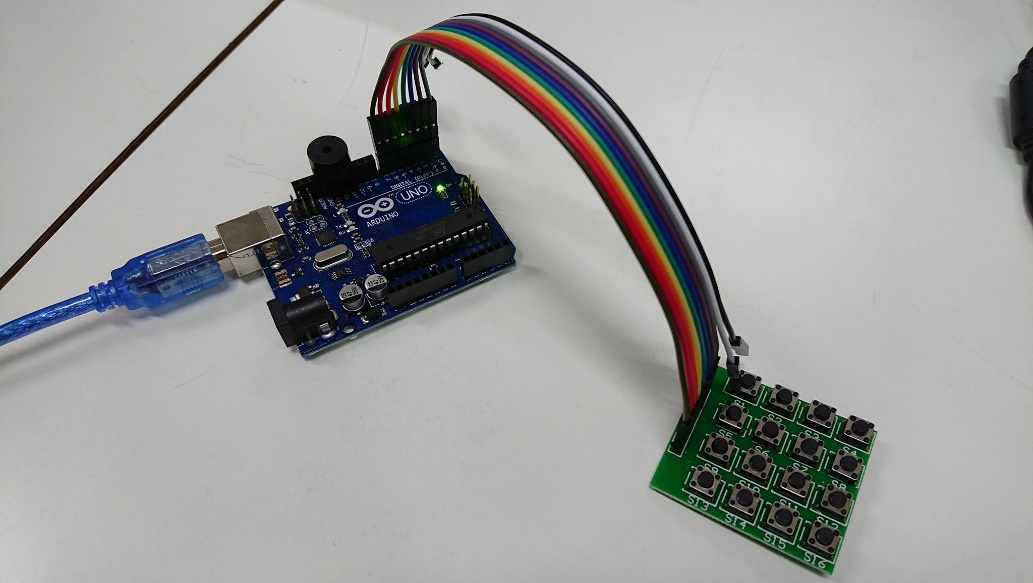
LOOKUPTB:

.DB 118, 0xFF, 0xFF, 0xFF, 105, 0xFF, 94, 89, 0xFF, 79, 0xFF, 0xFF, 0xFF, 70, 0xFF, 62, 59, 0xFF

.DB 52, 0xFF, 0xFF, 0xFF, 46, 0xFF, 44, 39, 0xFF, 35, 0xFF, 0xFF, 0xFF, 31, 0xFF, 29, 26, 0xFF

# เปรียบเทียบความแตกต่างจาก Assignment I

## วงจรเดิม



## โค้ดภาษาแอสแซมบลีเดิม

.INCLUDE "m328Pdef.inc"

.DEF TMP = r16

.DEF READ\_V = r17

.DEF VAR\_A = r18

.DEF DISTANCE = r19

.DEF TL = r20

.DEF TA = r21

.DEF TB = r22

.MACRO COMPARE\_AND\_BRANCH\_IF\_KEYPRESSED

cpi VAR\_A, 0xff

brne KEYPRESSED

.ENDMACRO

.CSEG

.ORG 0x00

rjmp RESET

RESET:

ldi TMP, 0x28

out DDRB, TMP

ldi TMP, 0x00

out PORTB, TMP

ldi TMP, 0xf0

out DDRD, TMP

READ\_KEY\_PAD:

;----- row A

ldi TMP, 0b11101111

ldi DISTANCE, 0

call SCAN\_KEYPAD

COMPARE\_AND\_BRANCH\_IF\_KEYPRESSED

;----- row B

ldi TMP, 0b11011111

ldi DISTANCE, 9

call SCAN\_KEYPAD

COMPARE\_AND\_BRANCH\_IF\_KEYPRESSED

;----- row C

ldi TMP, 0b10111111

ldi DISTANCE, 18

call SCAN\_KEYPAD

COMPARE\_AND\_BRANCH\_IF\_KEYPRESSED

;----- row D

ldi TMP, 0b01111111

ldi DISTANCE, 27

call SCAN\_KEYPAD

COMPARE\_AND\_BRANCH\_IF\_KEYPRESSED

ldi TMP, 0x00

out PORTB, TMP

rjmp READ\_KEY\_PAD

KEYPRESSED:

call VOL\_UP

rjmp READ\_KEY\_PAD

VOL\_UP:

ldi TL, 0x28

out PORTB, TL

rcall \_settime

rcall \_delay

ldi TL, 0x00

out PORTB, TL

rcall \_settime

rcall \_delay

ret

\_settime:

ldi TA, 0xff

mov TB, VAR\_A

ret

\_delay:

dec TA

brne \_delay

dec TB

brne \_delay

ret

SCAN\_KEYPAD:

ldi ZL, low(KEYPAD\_TABLE\*2)

ldi ZH, high(KEYPAD\_TABLE\*2)

out PORTD, TMP

nop

in READ\_V, PIND

ldi TMP, 0x0f

and READ\_V, TMP

add READ\_V, DISTANCE

subi READ\_V, 7

ldi TMP, 0x00

add ZL, READ\_V

adc ZH, TMP

lpm VAR\_A, Z

ret

KEYPAD\_TABLE:

.DB 0x0f, 0xff, 0xff, 0xff, 0x10, 0xff, 0x11, 0x12, 0xff, 0x13, 0xff, 0xff, 0xff, 0x14, 0xff, 0x15, 0x16, 0xff

.DB 0x17, 0xff, 0xff, 0xff, 0x18, 0xff, 0x19, 0x1a, 0xff, 0x1b, 0xff, 0xff, 0xff, 0x1c, 0xff, 0x1d, 0x1e, 0xff

## ความแตกต่าง

1. อุปกรณ์ที่ใช้เพิ่มขึ้นเพื่อการทำงานแบบอินเตอร์รัพท์นั่นคือ Diode ที่ต่อไปยังขา INT0 เพื่อตรวจสอบอินเตอร์รัพท์จากการกดหรือปล่อยปุ่มจาก keypad
2. การต่ออุปกรณ์ที่เปลี่ยนไปเนื่องจากการทำงานแบบอินเตอร์รัพท์ของงานชิ้นนี้ต้องใช้ทั้งขา INT0 และ OC0A ดังนั้นการบริหารจัดการขาของ AVR ATmega328p จึงต้องมีการเปลี่ยนไปตาม
3. การทำงานของโปรแกรม
   1. เนื่องจากงานเดิมไม่ได้ใช้งานอินเตอร์รัพท์ลักษณะการทำงานจะทำงานแบบ Polling หรือวนทำงานในฟังก์ชันเมนเพื่ออ่านค่าจาก keypad ตลอดเวลาและนำไปขับ buzzer ตามค่าที่อ่านได้ต่อไป การทำงานในลักษณะนี้ทำให้ AVR ต้องทำงานอ่านค่าจาก keypad และเซ็ตค่า output ตลอดเวลา
   2. ส่วนของโปรแกรมที่ทำงานแบบอินเตอร์รัพท์จะมีการทำงานก็ต่อเมื่อเกิดการอินเตอร์รัพท์หรือในงานนี้คือการกดปุ่มและปล่อยปุ่มบน keypad เท่านั้น ซึ่งเป็นการลดการทำงานของหน่วยประมวลผลหรือซีพียูลง รวมทั้งการใช้งาน Timer0 Interrupt ช่วยในการนับและจับเวลาเพื่อนำไปใช้ toggle เอาต์พุตเพื่อสร้างพัลส์ที่ต้องการ สามารถทำได้ง่ายกว่าจากเดิมที่ต้องเขียนฟังก์ชันดีเลย์ขึ้นเองซึ่งคำนวณได้ยากกว่าการคำนวณค่า OCR0A ซึ่งเห็นได้จากค่าในตาราง LOOKUPTB ที่เมื่อใช้ Timer0 Interrupt สามารถคำนวณออกมาเป็นค่าที่ชัดเจนง่ายกว่าและได้ค่าที่เที่ยงตรงมากกว่า