รายงาน

วงจรคำนวณสี่เหลี่ยมคางหมู : พื้นที่/ปริมาตร

จัดทำโดย

นายวัชริศ ไทยาพงศ์สกุล รหัสนักศึกษา 5735512079

นายวีรศักดิ์ ราชวังเมือง รหัสนักศึกษา 5735512083

Section 02

เสนอ

อาจารย์ พัชรี เทพนิมิตร

รายงานนี้เป็นส่วนหนึ่งของวิชา 242-309 Microcontroller and Interfacing

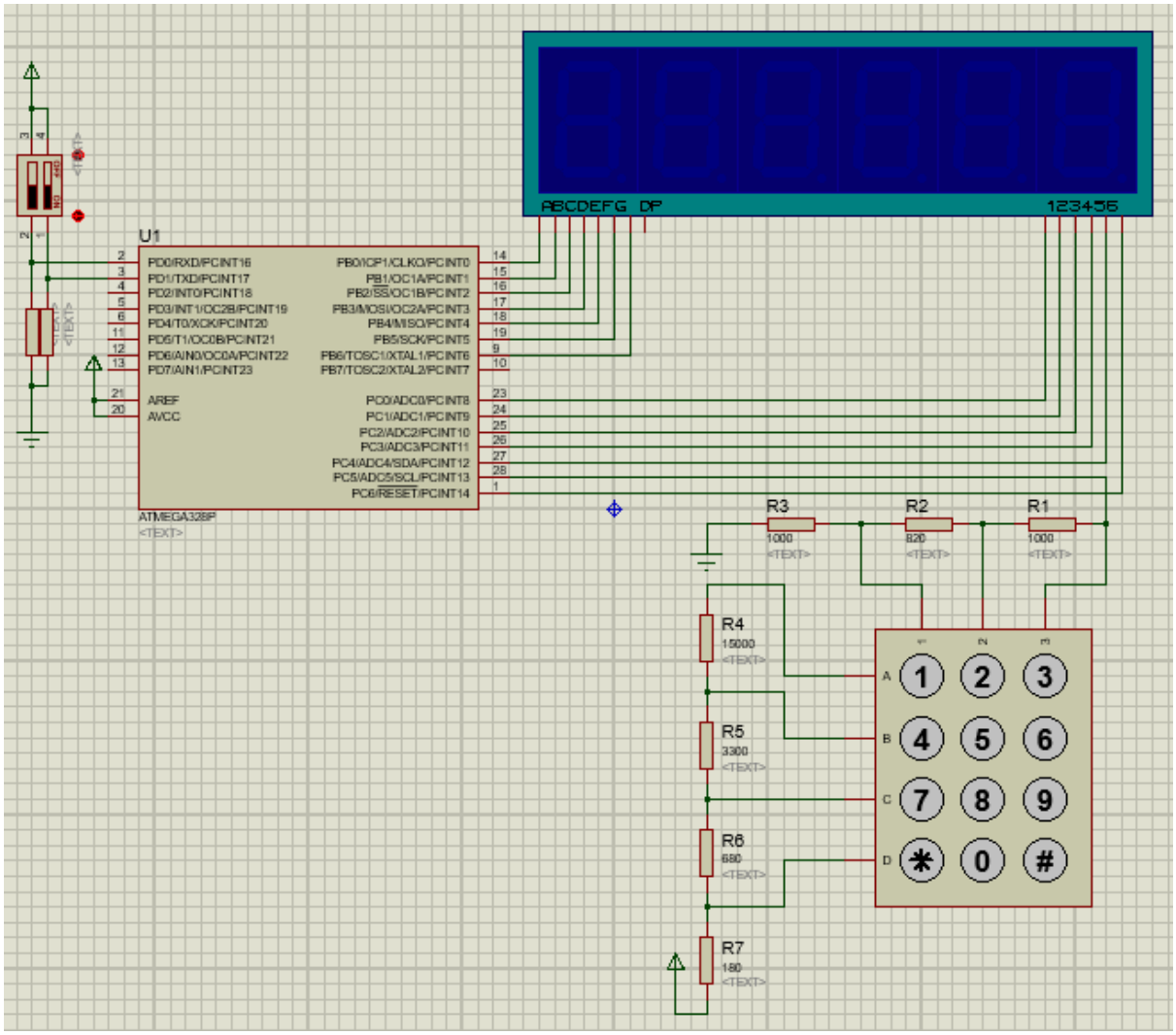
ภาคการศึกษาที่ 2 ปีการศึกษา 2560

ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

มหาวิทยาลัยสงขลานครินทร์

วงจรคำนวณสี่เหลี่ยมคางหมู : พื้นที่/ปริมาตร

โครงสร้างระบบ:



1. **อุปกรณ์ที่ใช้**

   Microcontroller ATmega328P

   Input

   - Dipswitch 2 pin
   - Keypad 3x4

   Output

   - 7-Segment 6 Digits

2. รายละเอียดอุปกรณ์

PORT B

ขา	ต่อเข้ากับขา
PB0	A ของ 7-Segment
PB1	B ของ 7-Segment
PB2	C ของ 7-Segment
PB3	D ของ 7-Segment
PB4	E ของ 7-Segment
PB5	F ของ 7-Segment
PB6	G ของ 7-Segment

PORT C

ขา	ต่อเข้ากับขา
PC0	1 ของ 7-Segment
PC1	2 ของ 7-Segment
PC2	3 ของ 7-Segment
PC3	4 ของ 7-Segment
PC4	5 ของ 7-Segment
PC5	Keypad
PC6	6 ของ 7-Segment

PORT D

ขา	ต่อเข้ากับขา
PD0	Dipswitch bit 1
PD1	Dipswitch bit 0

3. สูตรคำนวณ

สูตรคำนวณหาพื้นที่สี่เหลี่ยมคางหมู

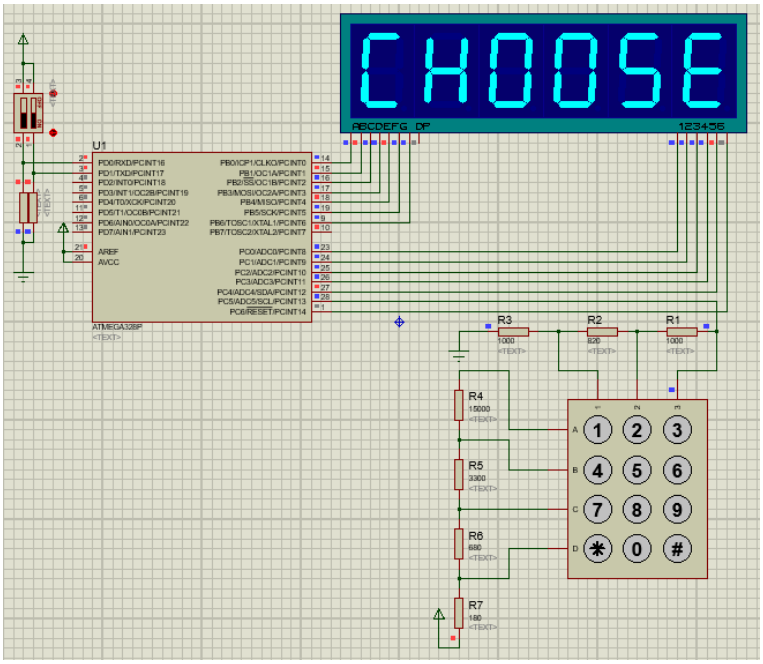
$$\text{Trapezoid area} = \frac{1}{2} \times (a + b) \times h$$

สูตรคำนวณหาปริมาตรสี่เหลี่ยมคางหมู

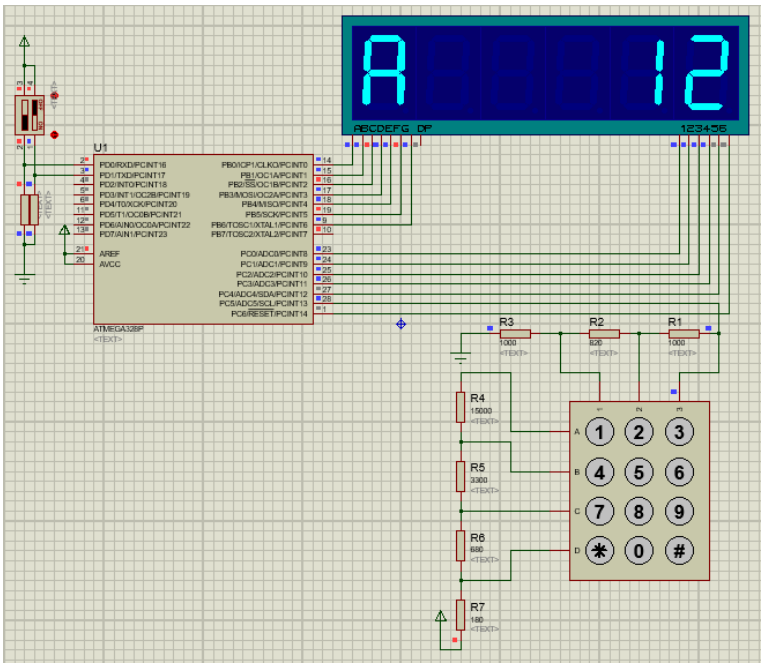
$$Trapezoidal\ volume = \left[\frac{1}{2} \times (a + b) \times h\right] \times l$$

<u>หลักการทำงาน:</u>

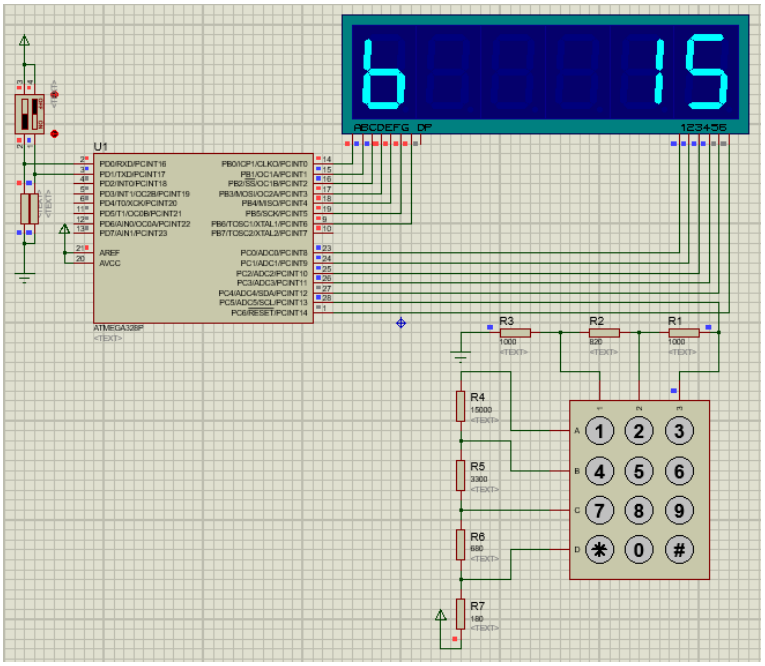
1. เมื่อรันโปรแกรม 7-Segment จะแสดงข้อความ “CHOOSE” เพื่อให้ผู้ใช้เลือกโหมดการทำงานของวงจร โดยทำการปรับ Dipswitch 2 Pin ถ้าปรับเป็น “01” วงจรจะเข้าสู่โหมดคำนวณหาพื้นที่รูปสี่เหลี่ยมคางหมู และถ้าปรับเป็น “10” วงจรจะเข้าสู่โหมดคำนวณหาปริมาตรรูปสี่เหลี่ยมคางหมู



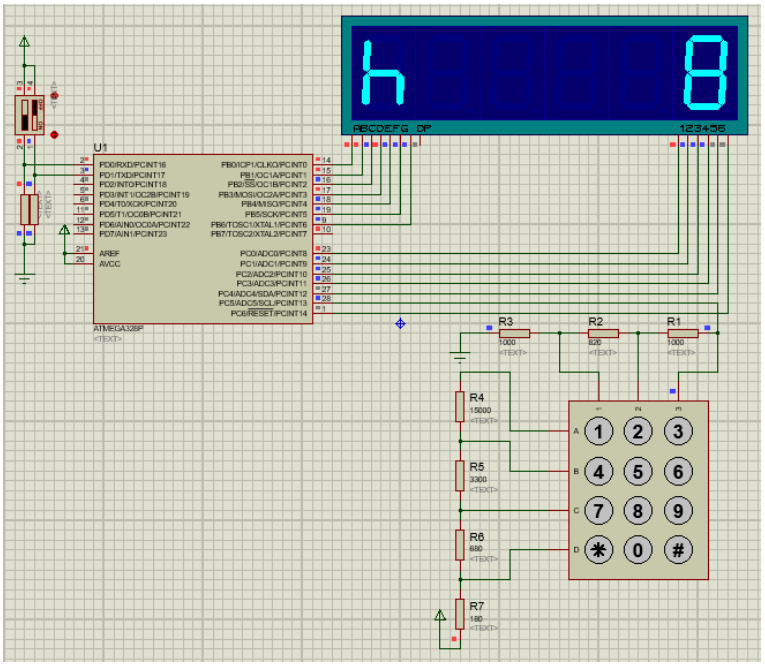
2. เมื่อผู้ใช้ปรับ Dipswitch เป็น “01” วงจรเข้าสู่โหมดคำนวณหาพื้นที่รูปสี่เหลี่ยมคางหมู 7-Segment จะแสดงข้อความ “A” เพื่อรอรับค่าผลบวกด้านขนาน จากผู้ใช้ และทำการเก็บค่าเมื่อผู้ใช้กด *



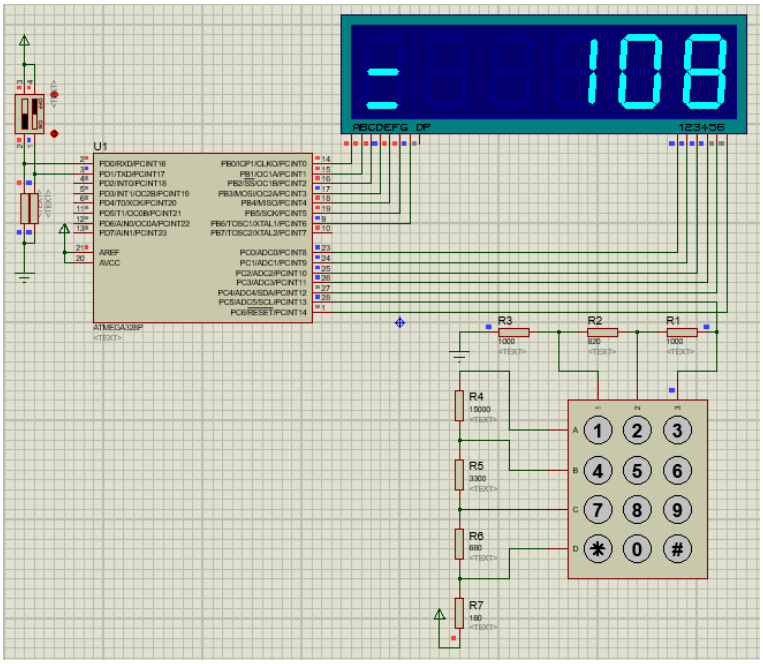
3. ถัดไป 7-Segment จะแสดงข้อความ “b” เพื่อรอรับค่าผลบวกด้านขนาน จากผู้ใช้ และทำการเก็บค่าเมื่อผู้ใช้กด *



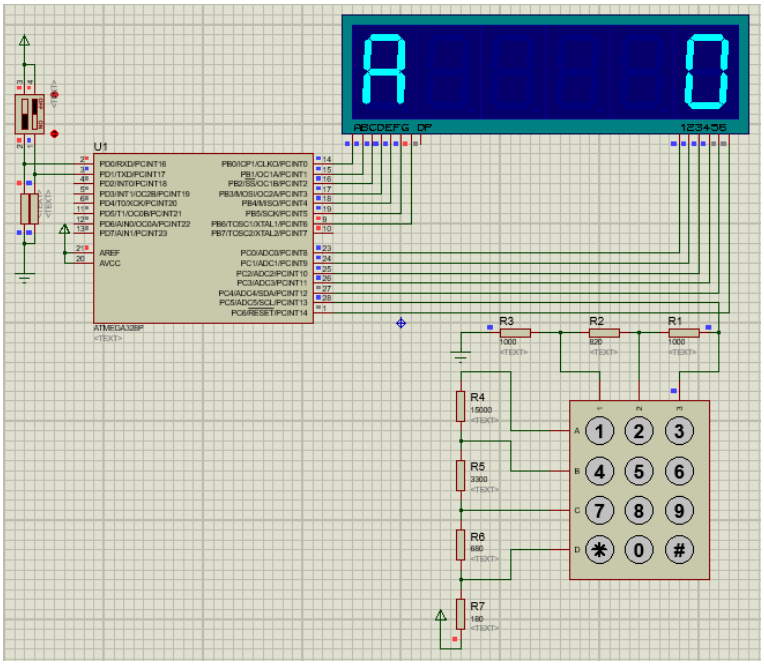
4. ถัดไป 7-Segment จะแสดงข้อความ “h” เพื่อรอรับค่าความสูง จากผู้ใช้ ทำการเก็บค่าเมื่อผู้ใช้กด * และส่งค่าไปทำการคำนวณ



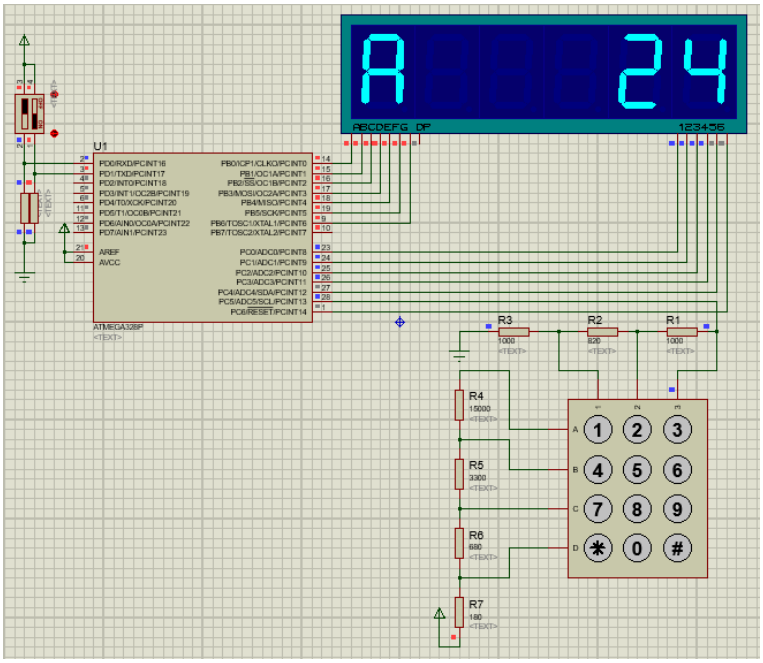
5. จากนั้นโปรแกรมจะนำค่าที่เก็บไว้มาทำการคำนวณหาพื้นที่รูปสี่เหลี่ยมคางหมู และแสดงผลผ่าน 7-Segment



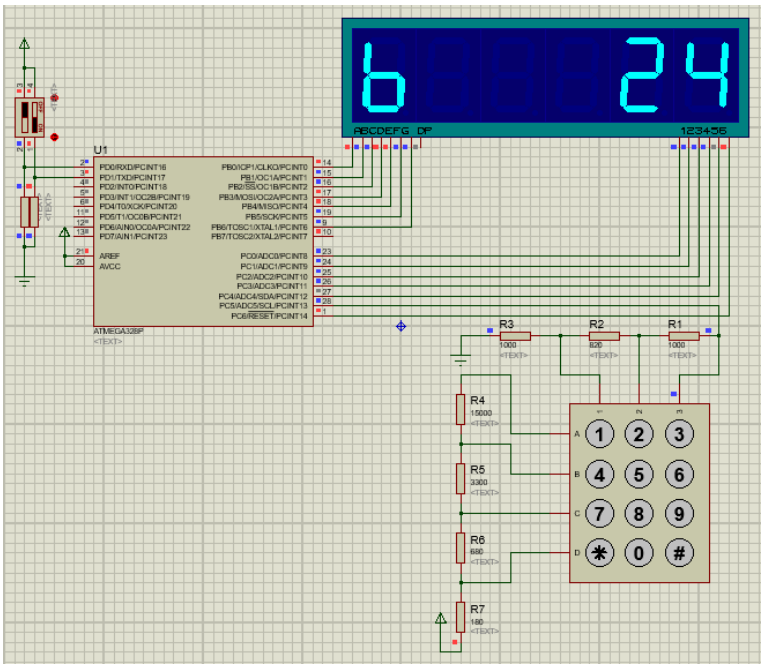
6. เมื่อผู้ใช้ต้องการเคลียร์ค่า เพื่อทำการคำนวณหาพื้นที่รูปสี่เหลี่ยมคางหมู รูปใหม่กด #



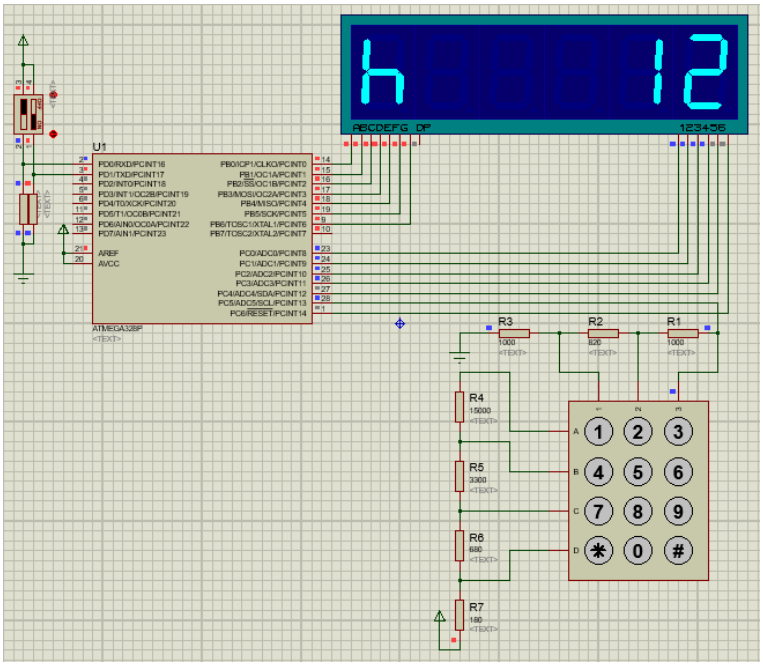
7. เมื่อผู้ใช้ปรับ Dipswitch เป็น “10” วงจรเข้าสู่โหมดคำนวณหาปริมาตรรูปสี่เหลี่ยมคางหมู 7-Segment จะแสดงข้อความ “A” เพื่อรอรับค่าผลบวกด้านขนาน จากผู้ใช้ และทำการเก็บค่าเมื่อผู้ใช้กด *



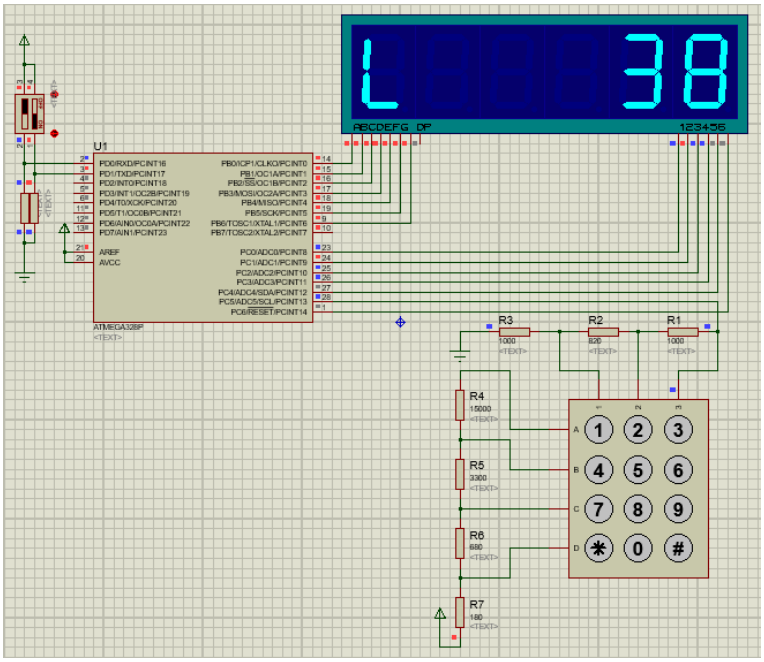
8. ถัดไป 7-Segment จะแสดงข้อความ “b” เพื่อรอรับค่าผลบวกด้านขนาน จากผู้ใช้ และทำการเก็บค่าเมื่อผู้ใช้กด *



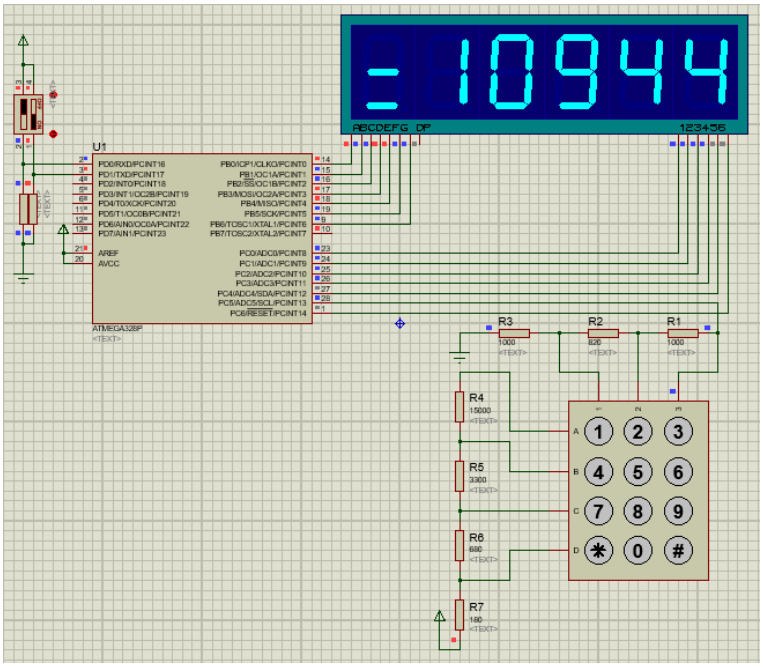
9. ถัดไป 7-Segment จะแสดงข้อความ “h” เพื่อรอรับค่าความสูง จากผู้ใช้ และทำการเก็บค่าเมื่อผู้ใช้กด *



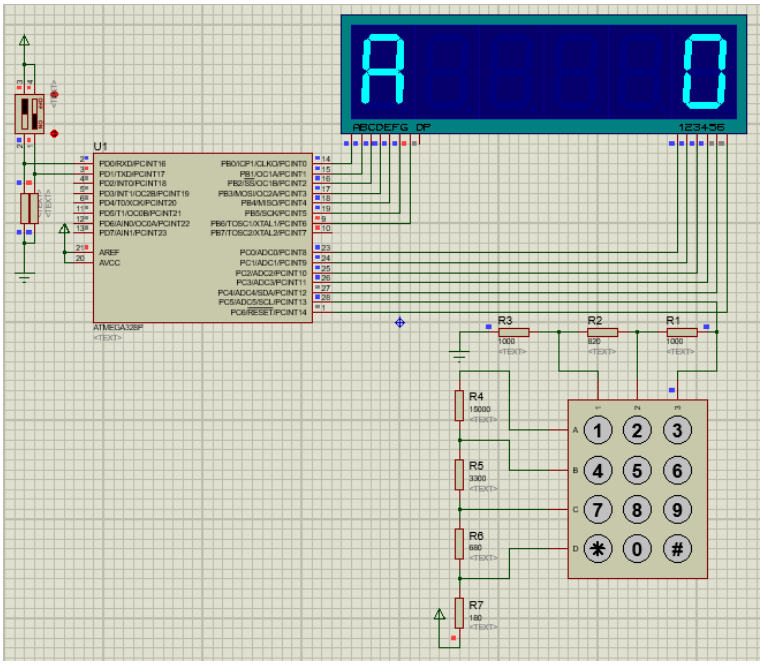
10. ถัดไป 7-Segment จะแสดงข้อความ “L” เพื่อรอรับค่าความยาว จากผู้ใช้ ทำการเก็บค่าเมื่อผู้ใช้กด * และส่งค่าไปทำการคำนวณ



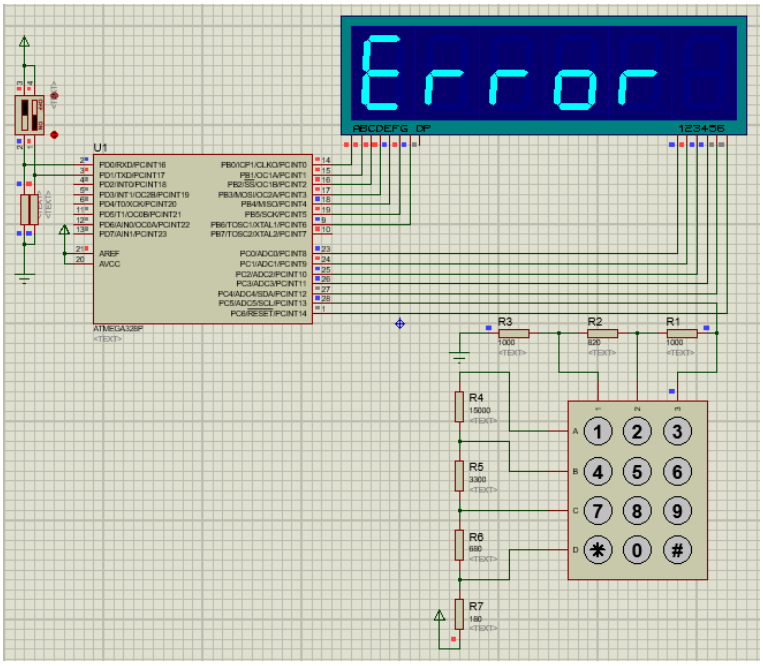
11. จากนั้นโปรแกรมจะนำค่าที่เก็บไว้มาทำการคำนวณหาปริมาตรรูปสี่เหลี่ยมคางหมู และแสดงผลผ่าน 7-Segment



12. เมื่อผู้ใช้ต้องการเคลียร์ค่า เพื่อทำการคำนวณหาปริมาตรรูปสี่เหลี่ยมคางหมู รูปใหม่กด #



13. หากผู้ใช้ทำการป้อนค่า เพื่อทำการคำนวณหาพื้นที่/ปริมาตรรูปสี่เหลี่ยมคางหมู แล้วได้ค่าผลลัพธ์เกิน 99,999 วงจรจะแสดงข้อความ “Error”



<u>สรุปเทคนิคที่ใช้ในวงจร:</u>

1. ใช้ความรู้เรื่อง PIN Change interrupt ในการเก็บสถานะการเปลี่ยนแปลงขาที่ต่อกับ Dipswitch เพื่อใช้ในการเลือกโหมดการคำนวณหาพื้นที่หรือปริมาตรรูปสี่เหลี่ยมคางหมู
2. ใช้ความรู้เรื่อง Timer/Counter1 (วงจรจับเวลา/นับ ขนาด 16 บิต) และโปรแกรมบริการ Interrupt ของ Timer/Counter1 ใช้ Timer1 ในการให้จังหวะสัญญาณนาฬิการ่วมกับการแปลง Analog to Digital ของ Keypad
3. ใช้ความรู้เรื่อง Delay ในการแสดงผลของ 7-Segment โดยจะใช้ Delay ในการหน่วงเวลาเพื่อแสดงผลบน 7-Segment

Source code:

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#define  TURN_ALL_LED_OFF 0xFF
#define  INTERVAL_time 65528
unsigned char TABLE7SEG[] = {
   0b00111111, 0b00000110, // 0 & 1
   0b01011011, 0b01001111, // 2 & 3
   0b01100110, 0b01101101, // 4 & 5
   0b01111101, 0b00000111, // 6 & 7
   0b01111111, 0b01101111, // 8 & 9
   0b01110111, 0b01111100, // A & b
   0b00111001, 0b01011110, // C & D
   0b01111001, 0b01110001, // E & F
   0b01110110, 0b01001000, // H & =
   0b00000000, 0b01010000, // off & r
   0b00111000, 0b01011100, // L & o
   0b01110100                    // h
};
unsigned char DIVISOR, QUOTIENT, DECODED, alphabet = 10, state = 0, next = 0, input;
unsigned char digit1, digit2, digit3, digit4, digit5; // Digits 7-segment
unsigned short ACCUMULATED;
unsigned long NUMBERS, TEMP_a, TEMP_b, TEMP_h, TEMP_l; // Input value
unsigned long area = 0, volume = 0; // Output value
unsigned short sw = 0, mode;
ISR(TIMER1_OVF_vect){
            TCNT1 = INTERVAL_time;
}
ISR(PCINT2_vect){
   sw = PIND;
   mode = sw & 0x03;
   if(mode == 0){ // Choose mode
      state = 0;
   }
   else if(mode == 1){ // Find trapezoid area mode
      state = 1;
      input = 0;
   }
   else if(mode == 2){ // Find trapezoidal volume mode
      state = 1;
      input = 1;
   }
```

```
}
ISR(ADC_vect){
   if(ADCH > 9){
      ACCUMULATED += ADCH;
      DIVISOR++;
   }
   else{
      ACCUMULATED = 0;
      DIVISOR = 0;
   }
   if(DIVISOR == 10){
      QUOTIENT = ACCUMULATED/DIVISOR;
      if((QUOTIENT>9)&&(QUOTIENT<16)) DECODED = 1;
      else if((QUOTIENT>18)&&(QUOTIENT<26)) DECODED = 2;
      else if((QUOTIENT>29)&&(QUOTIENT<37)) DECODED = 3;
      else if((QUOTIENT>46)&&(QUOTIENT<53)) DECODED = 4;
      else if((QUOTIENT>74)&&(QUOTIENT<81)) DECODED = 5;
      else if((QUOTIENT>99)&&(QUOTIENT<107)) DECODED = 6;
      else if((QUOTIENT>133)&&(QUOTIENT<141)) DECODED = 7;
      else if((QUOTIENT>169)&&(QUOTIENT<177)) DECODED = 8;
      else if((QUOTIENT>192)&&(QUOTIENT<199)) DECODED = 9;
      else if((QUOTIENT>212)&&(QUOTIENT<220)) DECODED = 10;
      else if((QUOTIENT>228)&&(QUOTIENT<236)) DECODED = 0;
      else if((QUOTIENT>236)&&(QUOTIENT<244)) DECODED = 11;
      else DECODED = 15;
      if(DECODED != 11 && DECODED != 10){ // Not * & #
            NUMBERS = (NUMBERS*10) + DECODED;
      }
      else if(DECODED == 10 && next==0 && input==0){
            alphabet=11;
            TEMP_a=NUMBERS;
            NUMBERS=0;
            next++;
      }
      else if(DECODED == 10 && next==1 && input==0){
            alphabet=22;
            TEMP_b=NUMBERS;
            NUMBERS=0;
            next++;
      }
      else if(DECODED == 10 && next==2 && input==0){
            alphabet=17;
            TEMP_h=NUMBERS;
```

```
            NUMBERS=0;
            next++;
            state++;
        }
        else if(DECODED == 10 && next==0 && input==1){
            alphabet=11;
            TEMP_a=NUMBERS;
            NUMBERS=0;
            next++;
        }
        else if(DECODED == 10 && next==1 && input==1){
            alphabet=22;
            TEMP_b=NUMBERS;
            NUMBERS=0;
            next++;
        }
        else if(DECODED == 10 && next==2 && input==1){
            alphabet=20;
            TEMP_h=NUMBERS;
            NUMBERS=0;
            next++;
        }
        else if(DECODED == 10 && next==3 && input==1){
            alphabet=17;
            TEMP_l=NUMBERS;
            NUMBERS=0;
            next++;
            state++;
        }
        else if(DECODED == 11){
            NUMBERS=0;
            next=0;
            alphabet=10;
            state=1;
        }
}
    // Choose mode
    if(state == 0){
        NUMBERS=0;
        next=0;
        state=0;
        PORTB = ~TABLE7SEG[12];
        PORTC = 0b00000001; // C
```

```
    _delay_ms(5);
    PORTC = 0b00000000;
    _delay_ms(5);
    PORTB = ~TABLE7SEG[16];
    PORTC = 0b00000010; // H
    _delay_ms(5);
    PORTC = 0b00000000;
    _delay_ms(5);
    PORTB = ~TABLE7SEG[0];
    PORTC = 0b00000100; // 0
    _delay_ms(5);
    PORTC = 0b00000000;
    _delay_ms(5);
    PORTB = ~TABLE7SEG[0];
    PORTC = 0b00001000; // 0
    _delay_ms(5);
    PORTC = 0b00000000;
    _delay_ms(5);
    PORTB = ~TABLE7SEG[5];
    PORTC = 0b00010000; // S
    _delay_ms(5);
    PORTC = 0b00000000;
    _delay_ms(5);
    PORTB = ~TABLE7SEG[14];
    PORTC = 0b01000000; // E
    _delay_ms(5);
    PORTC = 0b00000000;
    _delay_ms(5);
  }
  else if(state == 1){
    digit1 = (NUMBERS)%10;
    digit2 = (NUMBERS/10)%10;
    digit3 = ((NUMBERS)/100)%10;
    digit4 = ((NUMBERS)/1000)%10;
    digit5 = ((NUMBERS)/10000)%10;
    if(NUMBERS<10){
            digit5=digit4=digit3=digit2=18;
    }
    else if(NUMBERS<100){
            digit5=digit4=digit3=18;
    }
    else if(NUMBERS<1000){
            digit5=digit4=18;
```

```
    }
    else if(NUMBERS<10000){
            digit5=18;
    }
    PORTB = ~TABLE7SEG[alphabet];
    PORTC = 0b00000001;
    _delay_ms(5);
    PORTC = 0b00000000;
    _delay_ms(5);
    PORTB = ~TABLE7SEG[digit5];
    PORTC = 0b00000010;
    _delay_ms(5);
    PORTC = 0b00000000;
    _delay_ms(5);
    PORTB = ~TABLE7SEG[digit4];
    PORTC = 0b00000100;
    _delay_ms(5);
    PORTC = 0b00000000;
    _delay_ms(5);
    PORTB = ~TABLE7SEG[digit3];
    PORTC = 0b00001000;
    _delay_ms(5);
    PORTC = 0b00000000;
    _delay_ms(5);
    PORTB = ~TABLE7SEG[digit2];
    PORTC = 0b0010000;
    _delay_ms(5);
    PORTC = 0b00000000;
    _delay_ms(5);
    PORTB = ~TABLE7SEG[digit1];
    PORTC = 0b01000000;
    _delay_ms(5);
    PORTC = 0b00000000;
  }
  // Find trapezoid area mode
  else if(next == 3 && state == 2 && input == 0){
    area = (0.5*(TEMP_a + TEMP_b))*TEMP_h; // Formula to find trapezoid area
    digit1 = (area)%10;
    digit2 = ((area)/10)%10;
    digit3 = ((area)/100)%10;
    digit4 = ((area)/1000)%10;
    digit5 = ((area)/10000)%10;
    if(area > 0 && area < 10){
```

```
        digit5=digit4=digit3=digit2=18;
}
else if(area > 10 && area < 100){
        digit5=digit4=digit3=18;
}
else if(area > 100 && area < 1000){
        digit5=digit4=18;
}
else if(area > 1000 && area < 10000){
        digit5=18;
}
else if(area > 99999){
        alphabet=14; // E
        digit5=19;   // r
        digit4=19;   // r
        digit3=21;   // o
        digit2=19;   // r
        digit1=18;   // off
}
PORTB = ~TABLE7SEG[alphabet];
PORTC = 0b00000001;
_delay_ms(5);
PORTC = 0b00000000;
_delay_ms(5);
PORTB = ~TABLE7SEG[digit5];
PORTC = 0b00000010;
_delay_ms(5);
PORTC = 0b00000000;
_delay_ms(5);
PORTB = ~TABLE7SEG[digit4];
PORTC = 0b00000100;
_delay_ms(5);
PORTC = 0b00000000;
_delay_ms(5);
PORTB = ~TABLE7SEG[digit3];
PORTC = 0b00001000;
_delay_ms(5);
PORTC = 0b00000000;
_delay_ms(5);
PORTB = ~TABLE7SEG[digit2];
PORTC = 0b0010000;
_delay_ms(5);
PORTC = 0b00000000;
```

```
    _delay_ms(5);
    PORTB = ~TABLE7SEG[digit1];
    PORTC = 0b01000000;
    _delay_ms(5);
    PORTC = 0b00000000;
}
// Find trapezoidal volume mode
else if(next == 4 && state == 2 && input == 1){
    volume = (((0.5*(TEMP_a + TEMP_b))*TEMP_h)*TEMP_l; // Formula to find trapezoidal volume
    digit1 = (volume)%10;
    digit2 = ((volume)/10)%10;
    digit3 = ((volume)/100)%10;
    digit4 = ((volume)/1000)%10;
    digit5 = ((volume)/10000)%10;
    if(volume > 0 && volume < 10){
            digit5=digit4=digit3=digit2=18;
    }
    else if(volume > 10 && volume < 100){
            digit5=digit4=digit3=18;
    }
    else if(volume > 100 && volume < 1000){
            digit5=digit4=18;
    }
    else if(volume > 1000 && volume < 10000){
            digit5=18;
    }
    else if(volume > 99999){
            alphabet=14; // E
            digit5=19;   // r
            digit4=19;   // r
            digit3=21;   // o
            digit2=19;   // r
            digit1=18;   // off
    }
    PORTB = ~TABLE7SEG[alphabet];
    PORTC = 0b00000001;
    _delay_ms(5);
    PORTC = 0b00000000;
    _delay_ms(5);
    PORTB = ~TABLE7SEG[digit5];
    PORTC = 0b00000010;
    _delay_ms(5);
    PORTC = 0b00000000;
```

```
        _delay_ms(5);
        PORTB = ~TABLE7SEG[digit4];
        PORTC = 0b00000100;
        _delay_ms(5);
        PORTC = 0b00000000;
        _delay_ms(5);
        PORTB = ~TABLE7SEG[digit3];
        PORTC = 0b00001000;
        _delay_ms(5);
        PORTC = 0b00000000;
        _delay_ms(5);
        PORTB = ~TABLE7SEG[digit2];
        PORTC = 0b0010000;
        _delay_ms(5);
        PORTC = 0b00000000;
        _delay_ms(5);
        PORTB = ~TABLE7SEG[digit1];
        PORTC = 0b01000000;
        _delay_ms(5);
        PORTC = 0b00000000;
    }
}
int main(void){
    DDRB = 0xFF;
    DDRD = 0x00;
    PORTB = TURN_ALL_LED_OFF;
    DDRC = 0x0F;

    ADMUX = 0b00100101;
    ADCSRA = 0b10101101;
    ADCSRB = 0x06;

    TCNT1 = INTERVAL_time;
    TCCR1B = 0x05;
    TIMSK1 = 0x01;

    PCICR = 0x04;
    PCMSK2 = 0x03;
    sei();
    NUMBERS = 0;
    DIVISOR = 0;
    while(1){ ; }
}
```