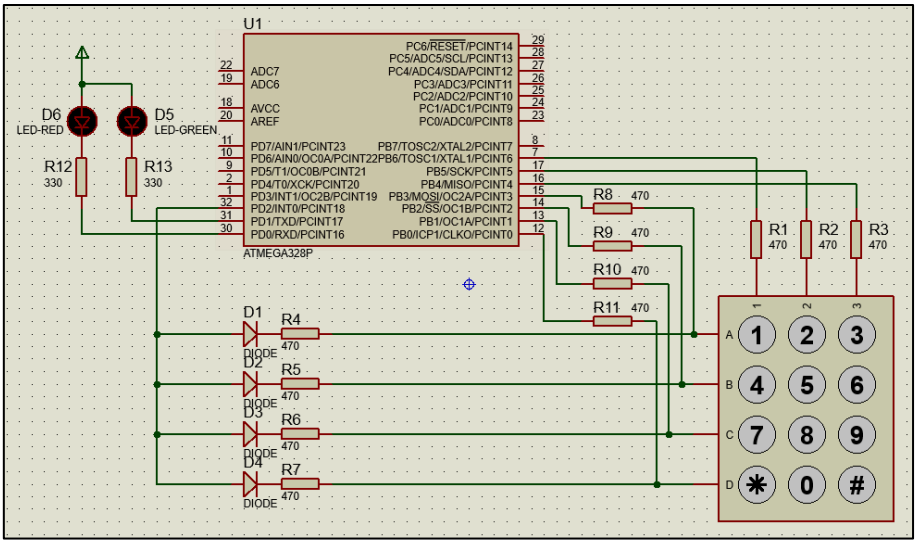
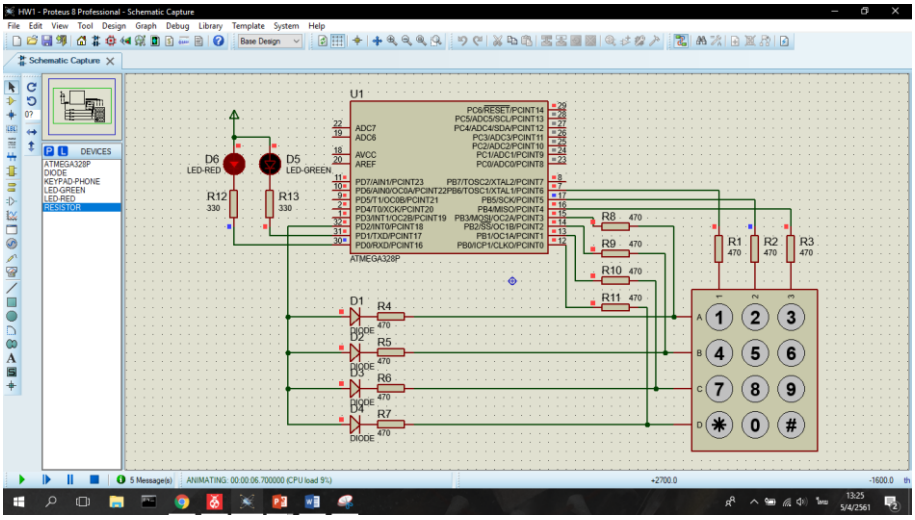
โค้ดภาษาซี

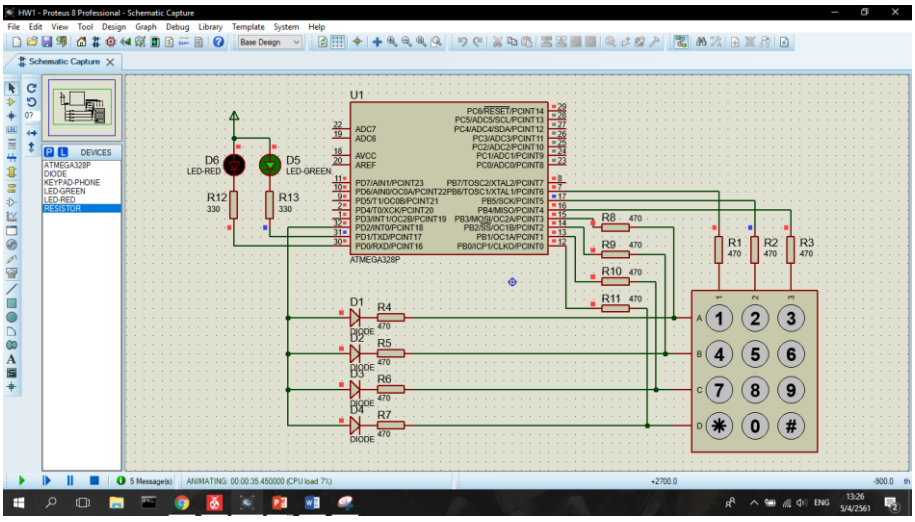
```
#include <avr/io.h>
#include <avr/interrupt.h>
#define F_CPU 1000000ul
#include <util/delay.h>

unsigned char ROW, SWITCH_V;
void blink(unsigned char num, unsigned char p );

unsigned char LOOKUPTB[] = {
                0x01, 0xFF, 0xFF, 0xFF, 0x04, 0xFF, 0x07, 0x0A, 0xFF,
                0x02, 0xFF, 0xFF, 0xFF, 0x05, 0xFF, 0x08, 0x00, 0xFF,
                0x03, 0xFF, 0xFF, 0xFF, 0x06, 0xFF, 0x09, 0x0B, 0xFF
};

int main(void)
{
        DDRB = 0xF0;
        PORTB = 0x0F;
        DDRD = 0xFB;
        PORTD = 0xFF;
        EICRA = (1 << ISC01);     // Falling Edge Interrupt
        EIMSK = 1 << INT0;        // Enable INT0
        sei();
 while(1)
 {
                PORTB = 0x0F;
                sei();
 }
}

ISR(INT0_vect)
{
        cli();
        for(unsigned char i=0; i<3; i++)
        {
                PORTB = ~(1 << (6 - i));
                ROW = PINB & 0x0F;
                ROW = ROW - 7 + (i * 9);
                SWITCH_V = LOOKUPTB[ROW];
                if(SWITCH_V == 0x00)            blink(5, 0);
                else if(SWITCH_V != 0xFF)       blink(SWITCH_V, 1);

        }
}

void blink(unsigned char num, unsigned char p )
{
        for(unsigned char i=0; i<num; i++)
        {
                PORTD ^= (0x01 << p);
                _delay_ms(300);
                PORTD ^= (0x01 << p);
                _delay_ms(300);
        }
}
```

ภาพวงจรบน Proteus



ภาพการทำงานของวงจรบน Proteus





การทำงานของโปรแกรม

1. กำหนด PORT B บิตที่ 0-3 เป็น Input และบิตที่ 4-7 เป็น output
   ผ่านคำสั่ง DDRB = 0xF0; และ PORTB = 0x0F; เพื่อใช้ Internal pull up ให้บิตที่ 0-3
2. กำหนด PORTD บิตที่ 2 เป็น Input และบิตอื่นๆ เป็น Output
   ผ่านคำสั่ง DDRD = 0xFB และ PORTD = 0xFF เพื่อกำหนดค่าเริ่มต้นให้ output ⇒ LED ดับ
3. เซ็ตโหมดของการอินเตอร์รัพท์เป็นแบบ Falling Edge. ด้วยคำสั่ง EICRA = (1<<ISC01)|(1<<ISC00);
4. เซ็ตให้ อินเตอร์รัพท์จากขา INT0 ทำงาน ด้วยคำสั่ง EIMSK = 1<<INT0;
5. เซ็ตให้ มีการอินเตอร์รัพท์ ด้วยคำสั่ง sei();
6. มี ลูปอันนต์ ที่ กำหนดให้ PORTB บิตที่ 0-3 มีการใช้ Internal pull-up.
7. การใน Interrupt Service Routine เมื่อเกิดอินเตอร์รัพท์ จากขา INT0 มีการทำงานดังนี้.
   7.1 ใช้คำสั่ง cli(); เพื่อไม่ให้เกิดอินเตอร์รัพท์ซ้อน.
   7.2 มี For Loop วนอ่านค่าจาก keypad ในแนว dolumn.
   7.3 จากการอ่านจะนำมา เข้าเคส เพื่อเลือกการทำงานการกระพริบของไฟ.
8. ฟังก์ชัน blink เป็นฟังก์ชันกำหนดการทำงานของ LED.
   8.1 จะกระพริบตามจำนวนค่า num ที่รับมา.
   8.2 LED ดวงที่จะกระพริบจะถูกกำหนดด้วยค่า p โดยสมการ 0x01<<p

สรุป

โปรแกรมจะมีการเซ็ตค่าให้ แต่ละ PORT เป็น Output และ Input ส่วนของอินเตอร์รัพท์ INT0 ต้องเซ็ตค่าให้เป็น INPUT ด้วย จากนั้นจะเป็นการเซ็ตโหมดของการอินเตอร์รัพท์เป็นแบบ Falling Edge. และกำหนดให้มีการอินเตอร์รัพท์ ในส่วนของ ISR กำหนดให้ทำงานจาก INT0_vect ทำงานตรวจสอบการกด keypad ว่าเป็นปุ่มใดและเลือกการทำงานไปยังฟังก์ชัน blink โดยส่งพารามิเตอร์ที่แตกต่างกันจากแต่ละปุ่ม.

อธิบาย

การกำหนดการทำงานให้ตอบรับอินเตอร์รัพจาก INT0 จำเป็นต้องเซ็ตตัวแปร EICRA เพื่อกำหนดโหมดของการอินเตอร์รัพท์ และตัวแปร EMISK เพื่อกำหนดให้ INT0 ตอบรับการอินเตอร์รัพท์. รวมทั้งคำสั่ง sei(); ที่ทำให้ซีพียูตอบสนองต่ออินเตอร์รัพท์ สำหรับ keypad จะส่งค่าอินเตอร์รัพท์ไปยัง INT0 เมื่อเกิดการกดปุ่ม (การกดปุ่มทำให้เกิด Logic 0 ส่งไป INT0 จึงตั้งโหมดเป็น Falling Edge ใน EICRA).