KÄLLKOD

```
#include <avr/interrupt.h>
#include <util/delay.h>
#include <util/delay_basic.h>
#include <avr/io.h>
#define forward 0b01111001 //knapp 2 på fjärrkontrollen
#define backward 0b01111011 //knapp 0 på fjärrkontrollen
#define left 0b00111010 //knapp 5 på fjärrkontrollen
#define right 0b00111000 //knapp 7 på fjärrkontrollen
#define stop 0b00111001 //knapp 6 på fjärrkontrollen
#define fire 0b01111010 //knapp 1 på fjärrkontrollen
#define fireOCR 0b00001100 // värde på OCR2
#define postFireOCR 0b00101000 // värde på OCR2

volatile int constant;

void command()
{
           switch(PINB & 0b01111011)
           {
                      case stop:
                                 //!Knapp!(6)
                                 PORTA = 0x00;
                                 break;
                      case backward:
                                 //!Knapp!(0)
                                 PORTA = 0x36;
                                 break;
                      case forward:
                                 //!Knapp!(2)
                                 PORTA = 0x1B;
                                 break;
                      case left:
                                 //!Knapp!(5)
                                 PORTA = 0x33;
                                 break;
                      case right:
                                 //!Knapp!(7)
                                 PORTA = 0x1E;
                                 break;
                      case fire:
                                 //!Knapp!(1)
                                 OCR2 = fireOCR;
                                 _delay_ms(80000);
                                 OCR2 = postFireOCR;
                                 break;
                      }
}

void initPorts(){ //för att sätta portarna till rätt startvärden
```

```
            DDRA = 0b00111111;
            PORTA = 0b00000000;
            DDRB = 0b00000000;
            PORTB = 0b01111111;
            DDRC = 0b00000001;
            PORTC = 0b00000001;
            DDRD = 0b10011000;
            PORTD = 0b00000000;
            return;
}

void determineEdge(){ //gör att det genereras interrupt om värdet på T0
ändras
            GICR = (0<<INT2); // Stänger av INT2 för att kunna ändra mcucsr
utan att ett interuppt skapas
            if ((PINB & 0b00000100) == 0b00000100){ // T0 är 1
                        MCUCSR = (0<<ISC2); // Interrupt on falling edge
            } else {
            MCUCSR = (1<<ISC2); // Interrupt on rising edge
    }
            GICR = (1<<INT2); // Aktivera interupts från INT2;
            GIFR = (1<<INTF2); // Clearar intf i gifr för interupts
    return;
}

void initInt(){ //möjliggör interupts
             GICR = (0<<INT2); // Stänger av INT2 för att kunna ändra mcucsr
utan att ett interuppt skapas
     determineEdge(); // Lyssna på B
             GICR = (1<<INT2); // Aktivera interupts från INT2;
     GIFR = (1<<INTF2); // Clearar intf i gifr för interupts
     sei(); // Bra för interrupts
     return;
}

void initPwm(){ //gör en PWM-puls
     TCNT2 = 0;
     TCCR2 |= (1<<COM21); // phase correct mode
     TCCR2 |= (1 << CS22)|(1 << CS21) ; // set prescale to 256
     TCCR2 |= (1<<WGM20); // phase correct pwm
     TIMSK |= (1<<OCIE2);
     OCR2 = 0; // initialize counter
     TIMSK |= (1 << TOIE2); // enable overflow interrupt
     return;
}

int main(void)        { //main-metod
            initPorts();
            initInt();
            initPwm();
            while(1){
               constant = OCR2; //loop där programmet hamnar i väntan på
interrupts.
```

```
            }
            return 1;
}



ISR(BADISR_vect) {
  constant  = OCR2;
}



ISR(INT2_vect){  //Avbrott på INT2
            determineEdge();
            command();
}
```