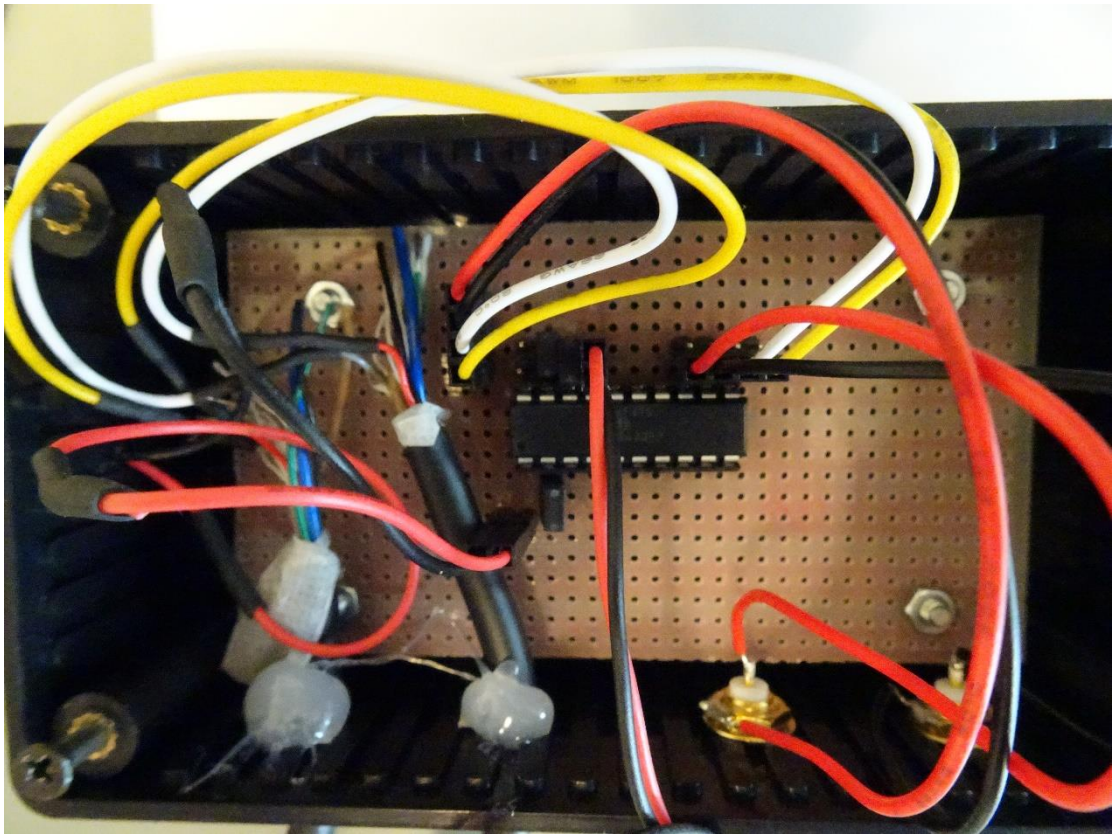
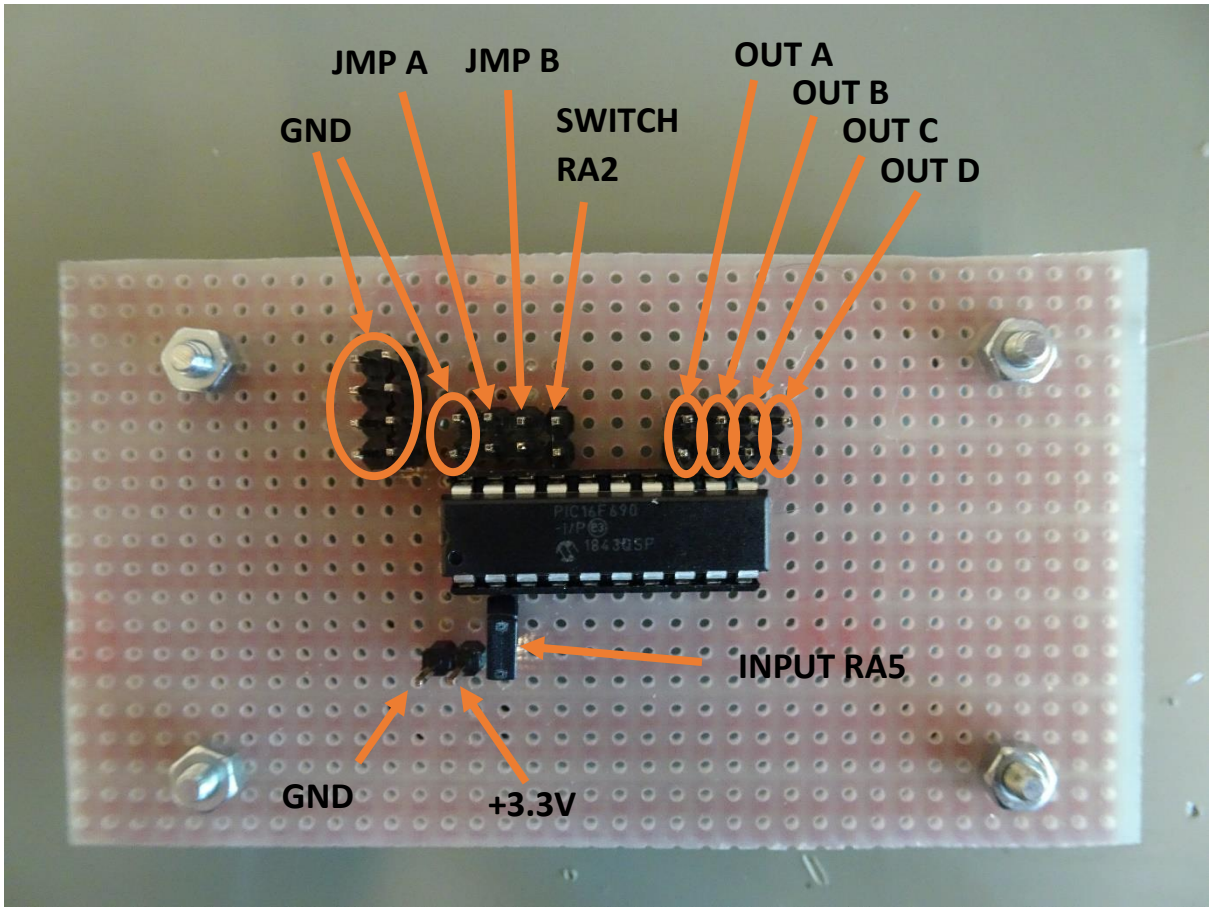
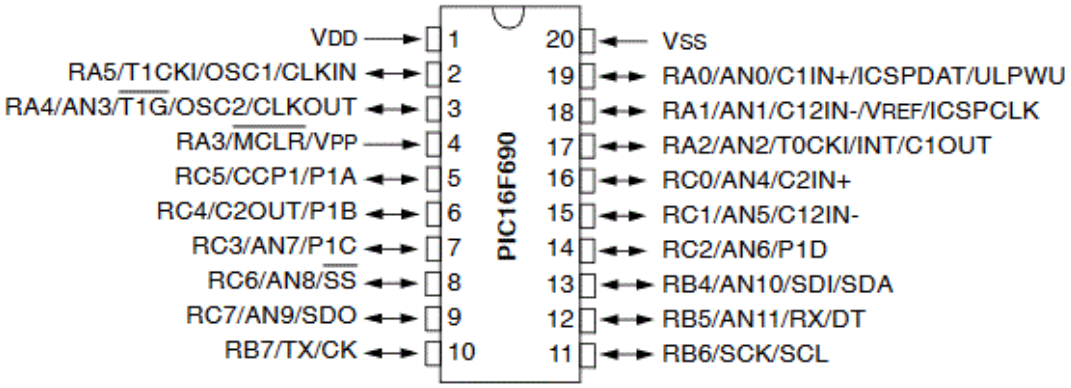
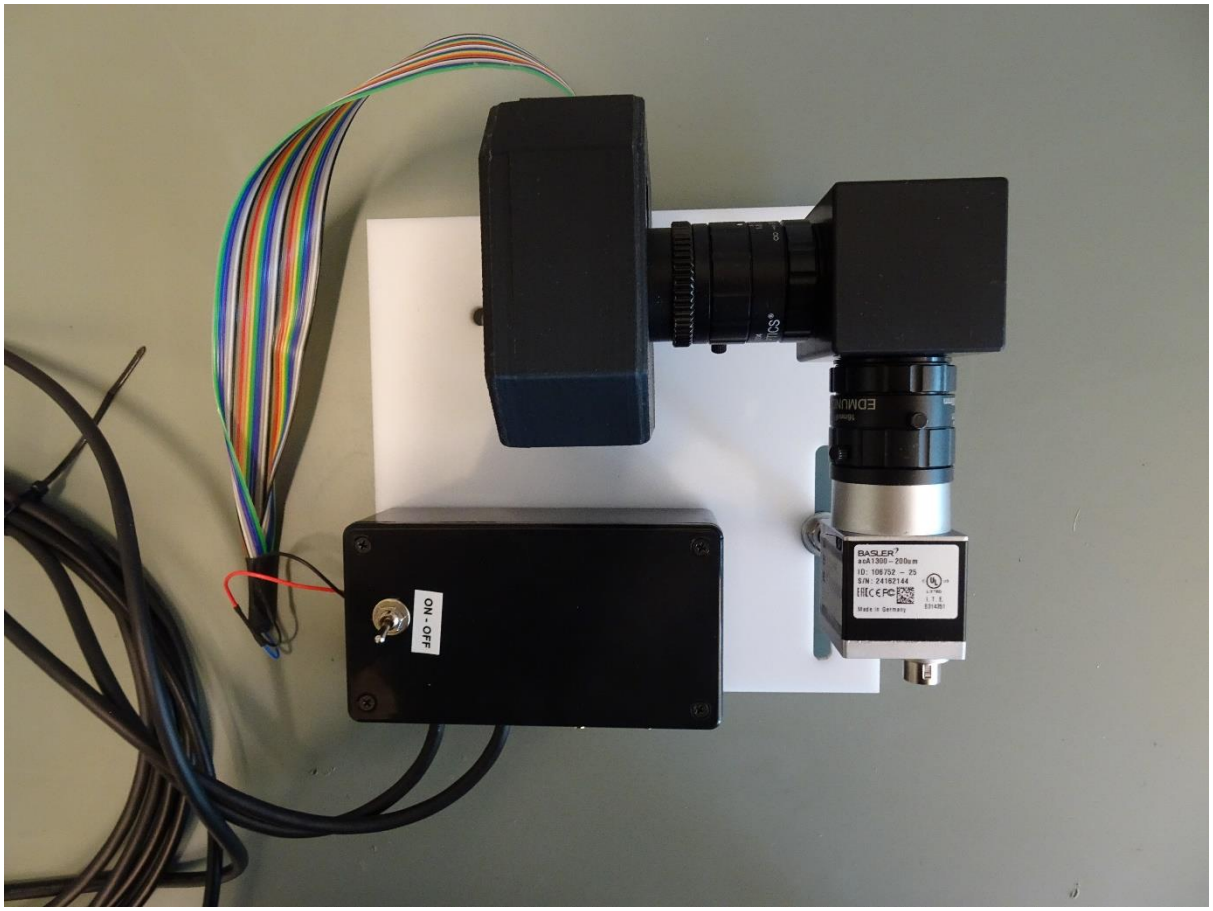
Triggerbox




JMP A
JMP B
GND
SWITCH
RA2
OUT A
OUT B
OUT C
OUT D
INPUT RA5
GND
+3.3V


VDD
RA5/T1CKI/OSC1/CLKIN
RA4/AN3/T1G/OSC2/CLKOUT
RA3/MCLR/VPP
RC5/CCP1/P1A
RC4/C2OUT/P1B
RC3/AN7/P1C
RC6/AN8/SS
RC7/AN9/SDO
RB7/TX/CK
1
2
3
4
5
6
7
8
9
10
PIC16F690
20
19
18
17
16
15
14
13
12
11
VSS
RA0/AN0/C1IN+/ICSPDAT/ULPWU
RA1/AN1/C12IN-/VREF/ICSPCLK
RA2/AN2/T0CKI/INT/C1OUT
RC0/AN4/C2IN+
RC1/AN5/C12IN-
RC2/AN6/P1D
RB4/AN10/SDI/SDA
RB5/AN11/RX/DT
RB6/SCK/SCL

JMP A	JMP B	Frequency [Hz]
shorted	shorted	30
open	shorted	60
shorted	open	90
open	open	120



```
/*
 * File:   TriggerBox.c
 * Author: stru_ro
 *
 * Date: 2022-03-11 13:48
 */

#pragma config FOSC = INTRCIO   // Oscillator Selection bits (INTOSCIO oscillator: I/O function on RA4/OSC2/CLKOUT pin, I/O function on RA5/OSC1/CLKIN)
#pragma config WDTE = OFF       // Watchdog Timer Enable bit (WDT disabled)
#pragma config PWRTE = OFF      // Power-up Timer Enable bit (PWRT disabled)
#pragma config MCLRE = ON       // MCLR Pin Function Select bit (MCLR pin function is MCLR)
#pragma config CP = OFF         // Code Protection bit (Program memory code protection is disabled)
#pragma config CPD = OFF        // Data Code Protection bit (Data memory code protection is disabled)
#pragma config BOREN = ON       // Brown-out Reset Selection bits (BOR enabled)
#pragma config IESO = OFF       // Internal External Switchover bit (Internal External Switchover mode is enabled)
#pragma config FCMEN = OFF      // Fail-Safe Clock Monitor Enabled bit (Fail-Safe Clock Monitor is disabled)

#include <xc.h>
#include <stdbool.h>

#define _XTAL_FREQ 4000000

const unsigned short FREQ_LUT[4] = {0xBEE5, 0xDF72, 0xEA4C, 0xEFB9};  //timer1 values for 30Hz, 60Hz, 90Hz, 120Hz
unsigned short timer_load_value;
unsigned char freq, pulse, dummy;
```

```
void main(void)
{

   // Set interrupt enable bits
   GIE = 1;   // Global Interrupt Enable Bit
   RABIE = 1; // PORTA/PORTB Change Interrupt Enable bit
   PEIE = 1;  // Peripherals Interrupt Enable Bit
   TMR1IE = 1;   //enable timer1
   IOCA2 = 1;  //enable interrupt on change of RA2 pin
   IOCA5 = 1;  //enable interrupt on change of RA5 pin

   T1CON = 0x00; //turn off Timer1

   ANSEL = 0;  //configure I/O pins as 'digital function', not 'analog input'
   ANSELH = 0; //configure I/O pins as 'digital function', not 'analog input'

   TRISA0 = 1; //configure RA0 as input
   TRISA1 = 1; //configure RA1 as input
   TRISA2 = 1; //configure RA2 as input
   TRISA5 = 1; //configure RA5 as input
   TRISB4 = 0; //configure RB4 as output
   TRISB5 = 0; //configure RB5 as output
   TRISB6 = 0; //configure RB6 as output
   TRISB7 = 0; //configure RB7 as output

   TRISC7 = 0; //for testing purposes

   nRABPU = 0;  //pull-ups are enabled by individual port latches
   WPUA0 = 1;  //enable RA0 weak pull-up
   WPUA1 = 1;  //enable RA1 weak pull-up
   WPUA2 = 1;  //enable RA2 weak pull-up
```

```
    //pull-up of RA5 is not enabled. RA5 is configured as an input for a signal from the event-camera

    timer_load_value = FREQ_LUT[0];
    TMR1 = timer_load_value; //set initial timer1 load value
    pulse = 0;

    if(RA2 == 0) T1CON = 0x01;   //if RA2==0 initially (switch is already closed), then turn on timer1
    if(RA5 == 1) T1CON = 0x01;   //if RA5==1 initially (input is already high), then turn on timer1

    while(1)
    {
      freq = RA1 << 1;
      freq |= RA0;
      timer_load_value = FREQ_LUT[freq];
    }
}
```

```
void interrupt ISR(void)
{
  if(TMR1IF)
  {
    pulse ^= 0x1;   //toggle pulse
    if (pulse==0x0) PORTB = 0x00;
    if (pulse==0x1) PORTB = 0xFF;
    TMR1 = timer_load_value; //set timer1 load value
    TMR1IF = 0; //clear timer1 interrupt
  }
  if(RABIF)
  {
    //read PORTA to end the mismatch condition
    dummy = PORTA;
    __delay_ms(10);   //wait 10ms until switch-bouncing is finished
    //read RA2 and RA5 again after delay
    if((RA2 == 0) || (RA5 == 1)){
      T1CON = 0x01;   //turn on timer1
      TMR1 = timer_load_value; //set timer1 load value
    }else{
      T1CON = 0x00;  //turn off timer1
      TMR1IF = 0; //clear timer1 interrupt
      PORTB = 0x00;
    }
    RABIF = 0;  //clear IOC interrupt flag
  }
}
```