src\main.c

```c
/*
 * HelloWorld.c
 *
 * Created: 3/27/2025 20:13:27 AM
 * Author : Magnus Trip
 */

 #include <stdio.h>
 #include <avr/io.h>
 #include <util/delay.h>

 #include <avr/eeprom.h>

 #include "usart.h"
 #include "i2cmaster.h"
 #include "lm75.h"
 #include "lcd.h"

void delay_ms(uint16_t milliseconds);
void delay_hs(uint16_t hundred_milliseconds);

 int main(void) {

 // i2c_init();


  uart_init(); // open the communication to the microcontroller
  io_redirect(); // redirect input and output to the communication



  // Setting Data direction register for port D as well as enabling pullups
  DDRD = 0xff;
  DDRD &= ~(1<<DDD4);
  PORTD |= (1<<PORTD4);

  //Turn on the counter0 to count on external rise on PortD4
  TCCR0B |= (1<<CS00)|(1<<CS01)|(1<<CS02);

   //Infinite While loop
   while(1) {
    //Delay function 1s + a counter counting external rising edges on PD4
    delay_hs(10);
    printf("%d\n",TCNT0);

 /*
    //500ms Delay + PB5 LED Flashing
    PORTB=0x20;
    delay_ms(500);
    PORTB=0x00;
    delay_ms(500); */
```

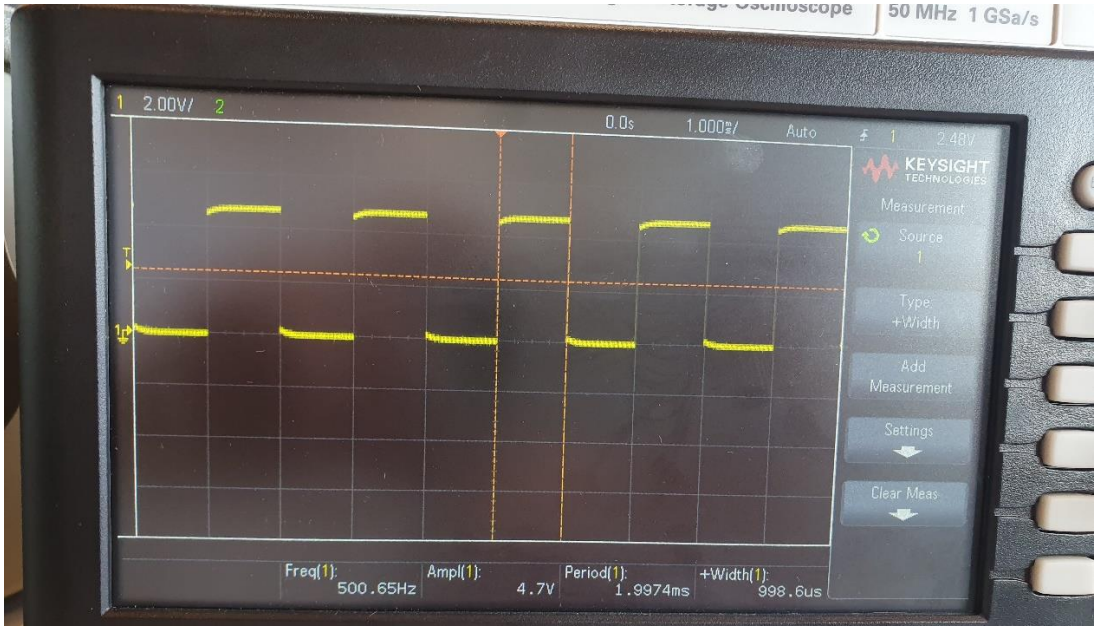
```
52
53  /*
54      //1s Delay + PB5 LED Flashing
55      PORTB=0x20;
56      delay_hs(10);
57      PORTB=0x00;
58      delay_hs(10); */
59
60  /*
61      //2s Delay + PB5 LED Flashing
62      PORTB=0x20;
63      delay_hs(20);
64      PORTB=0x00;
65      delay_hs(20); */
66
67     }
68     return 0;
69   }
70
71  //8-bit Clock for millisecond delay using timer register 0
72  void delay_ms(uint16_t milliseconds){
73      //Set Mode to CTC
74      TCCR0A |=(1<<WGM01);
75
76      //Top - set to 1 ms with this: Since 16MHz / 64 =  250 KHz : 1/250KHz = 4µs : 4µs *
    250 ticks = 1ms : max size of 8bit is 255
77      OCR0A = 249;
78
79      // Setting Prescaler to 64
80      TCCR0B |=(1<<CS01)|(1<<CS00);
81
82    for (uint16_t i = 0; i < milliseconds; i++)
83    {
84    //Delay Start + waiting for Overflow flag
85    while ((TIFR0 & (1<<OCF0A)) == 0)
86    {
87    }
88    //Flag Reset at Overflow
89    TIFR0 = (1<<OCF0A);
90    }
91  }
92
93
94  //16 bit Counter for Hundred millisecond delay using timer register
95  void delay_hs(uint16_t hundred_milliseconds){
96    //Set Mode to CTC
97    TCCR1B |=(1<<WGM12);
98
99    //Top - Determines the delay time to 100 ms - it should be 100 since the timer resoltion
    is 16MHz/64 = 250KHz  1/250KHz=4µs and 4µs * 25000 ticks = 100 ms : Max size of 16bit is
    65535
100   OCR1A = 24999;
101
102   // Setting Prescaler to 64
```

```
  TCCR1B |=(1<<CS11)|(1<<CS10);
  for (uint16_t i = 0; i < hundred_milliseconds; i++)
  {
    //Delay Start waiting for Overflow flag
    while ((TIFR1 & (1<<OCF1A)) == 0)
    {
    }
    //Flag Reset at Overflow
    TIFR1 = (1<<OCF1A);
  }
}
```

Assignment 5 - Timers and counters

Oscilloscope pictures from electrical lab: Using Keysight EDUX1002A

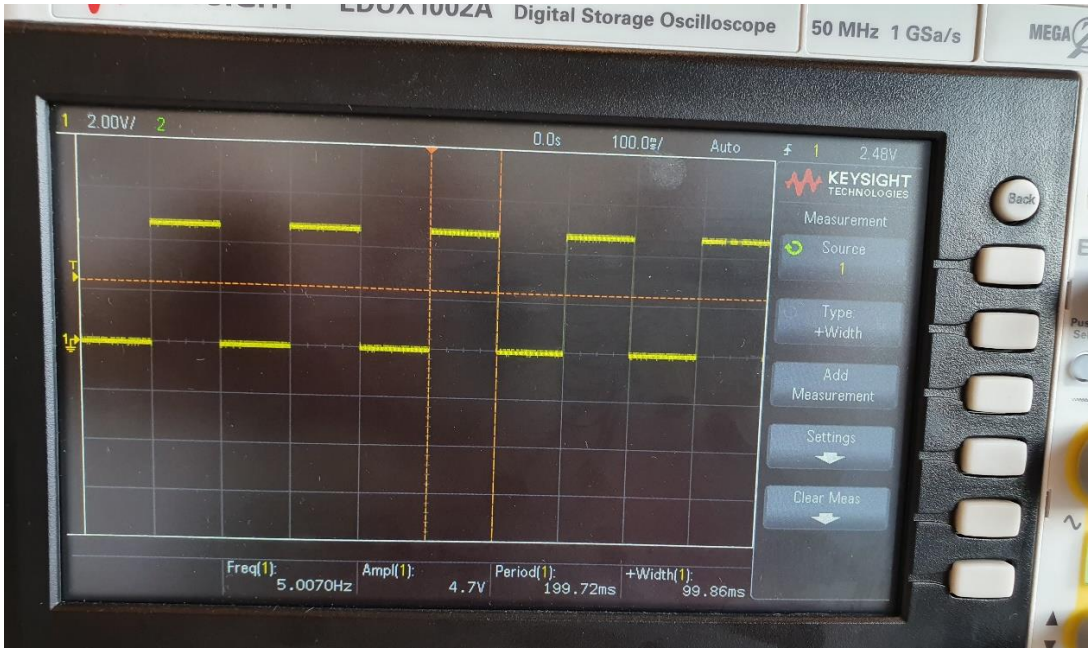
Function: delay_ms : 1ms



The delay for is 998,6µs which means that the delay is 99,86% accurate.

$$\frac{998{,}6\ ms}{1000\ ms} \cdot 100 = 99{,}86\%$$

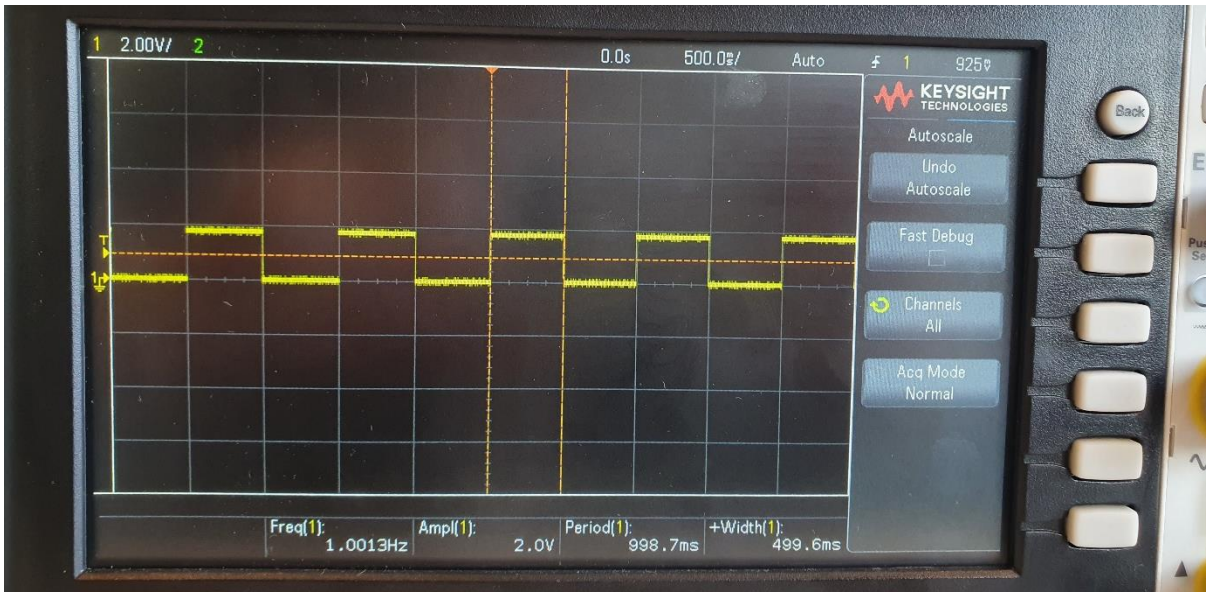
Function: delay_hs: 100ms



The delay is 99,86ms which means that the delay is 99,86% accurate.

$$\frac{99{,}86\ ms}{100\ ms} \cdot 100 = 99{,}86\%$$

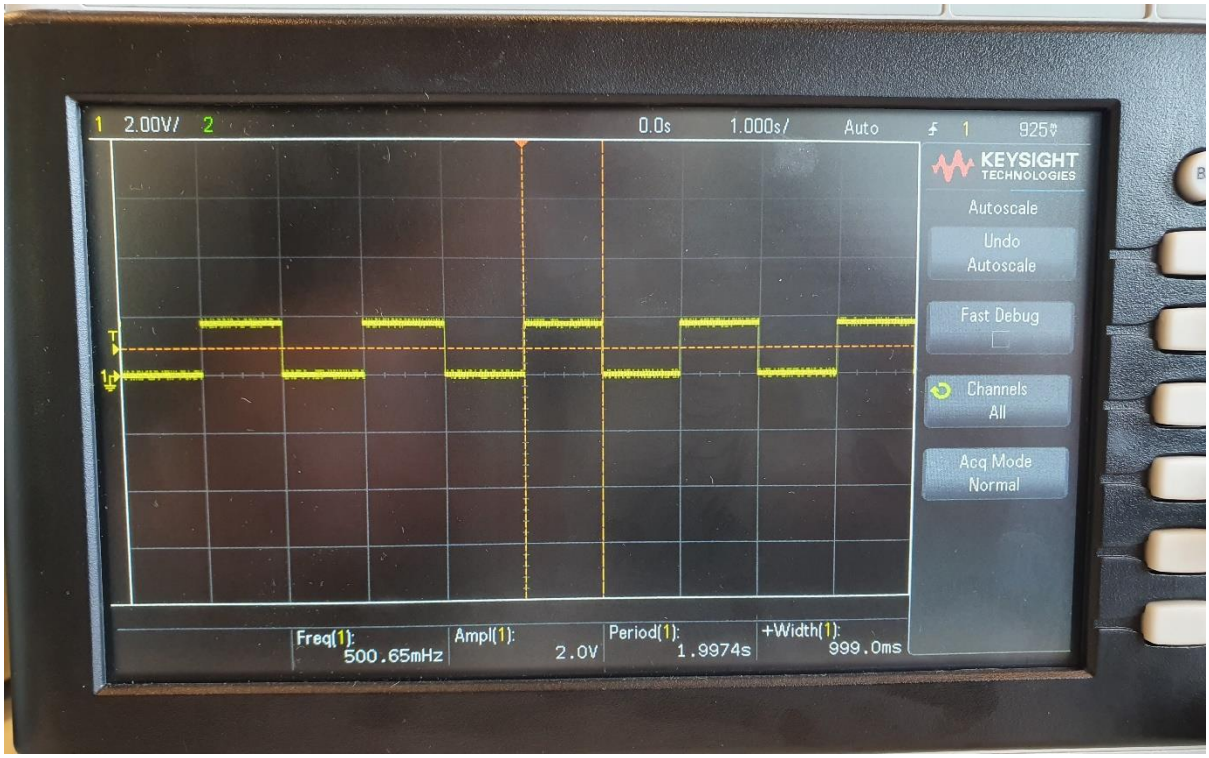
500ms delay using function: delay_ms(500);



The delay is 499,6ms which means that the delay is 99,9% accurate.

$$\frac{499{,}6}{500} \cdot 100 \approx 99{,}92\ \%$$

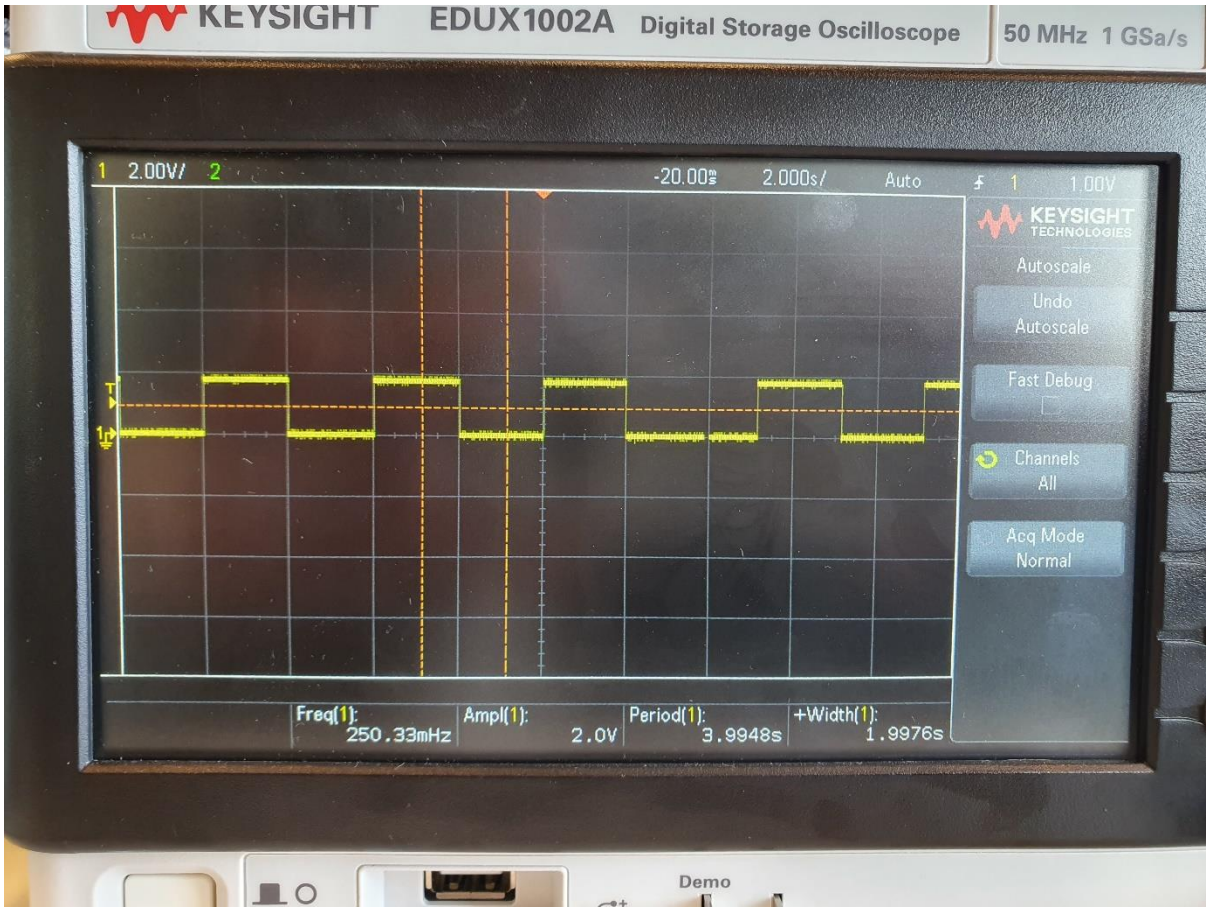
1s delay using function: delay_hs(10);



The delay is 999ms which is 99,9% accurate.

$$\frac{999}{1000} \cdot 100 = 99{,}9\%$$

2s delay using the function: delay_hs(20);



The delay is 1997,6ms which mean that the delay is roughly 99,9% accurate.

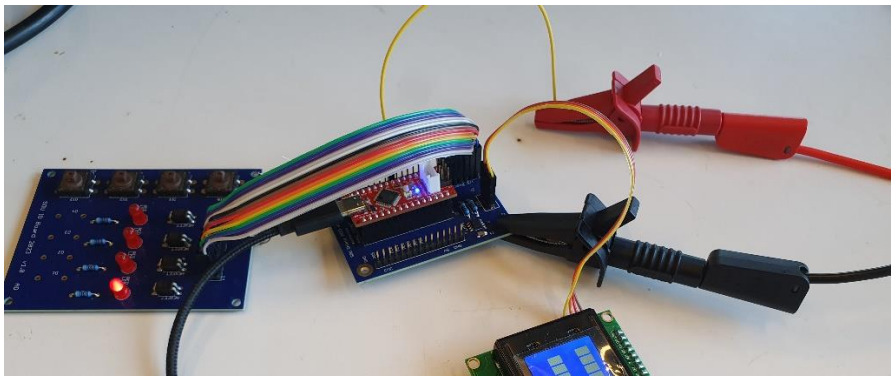
$$\frac{1997,6}{2000} \cdot 100 = 99,88\%$$

Testing the external edge detection of PD4

Arbitrary Function Generator: GWINSTEK AFG-2125

Results from rising edge detection on PD4:

Setup:



The red clip is connected to PD4 meanwhile the black clip is connected to ground.