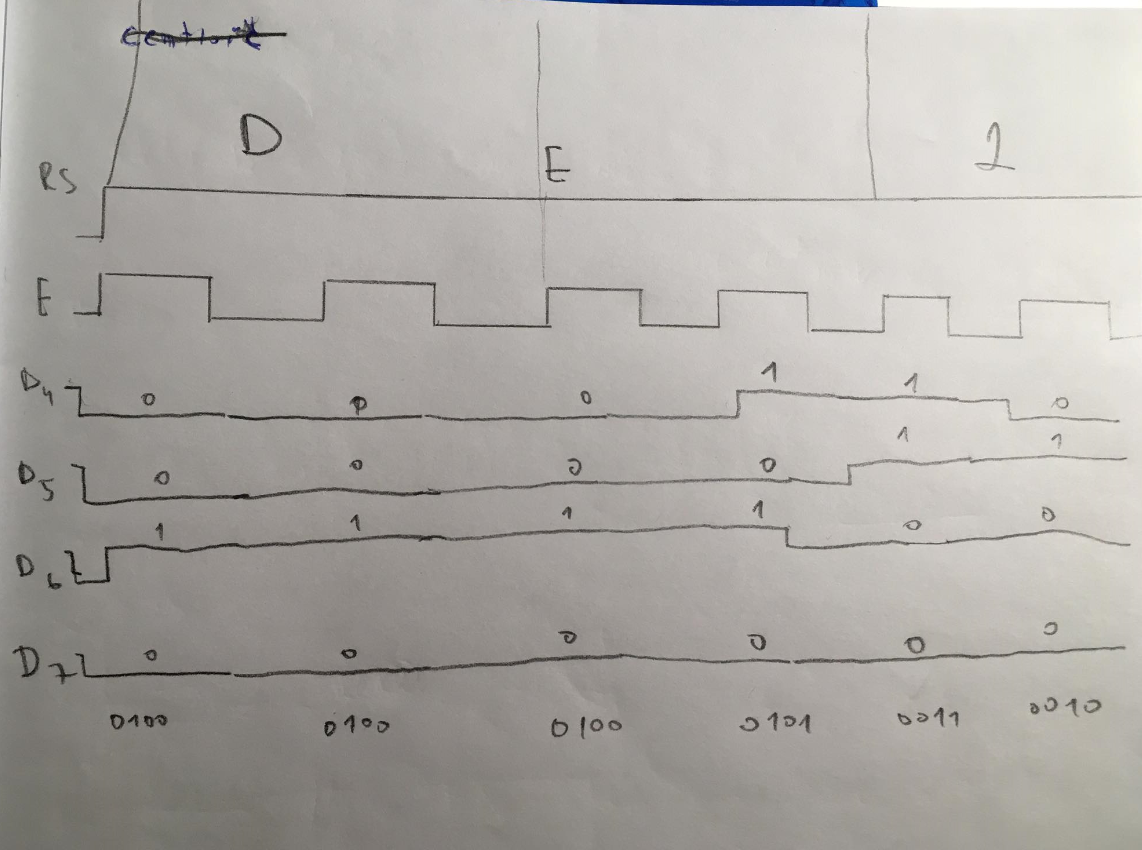
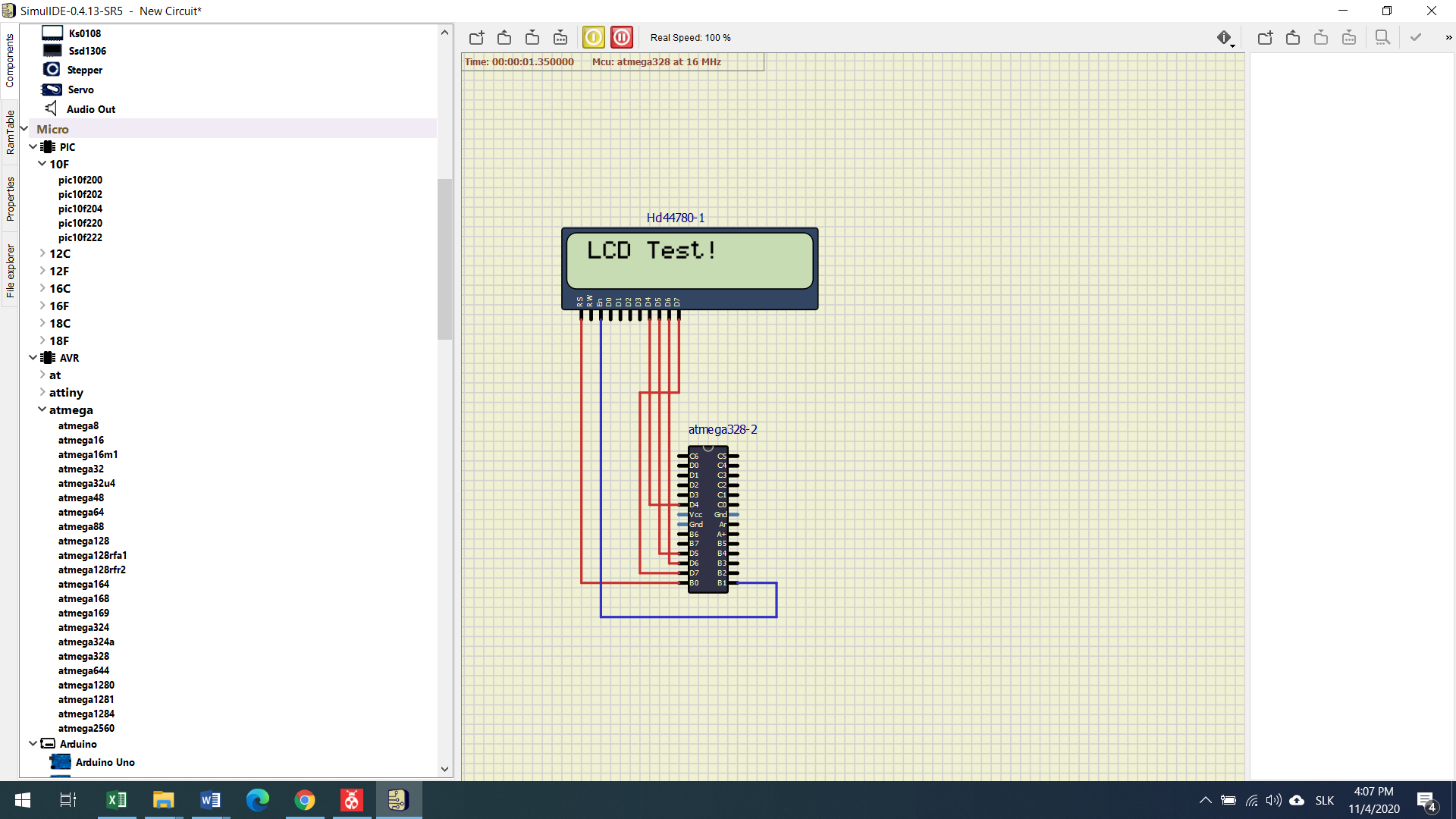
Kamil Káčer 211777

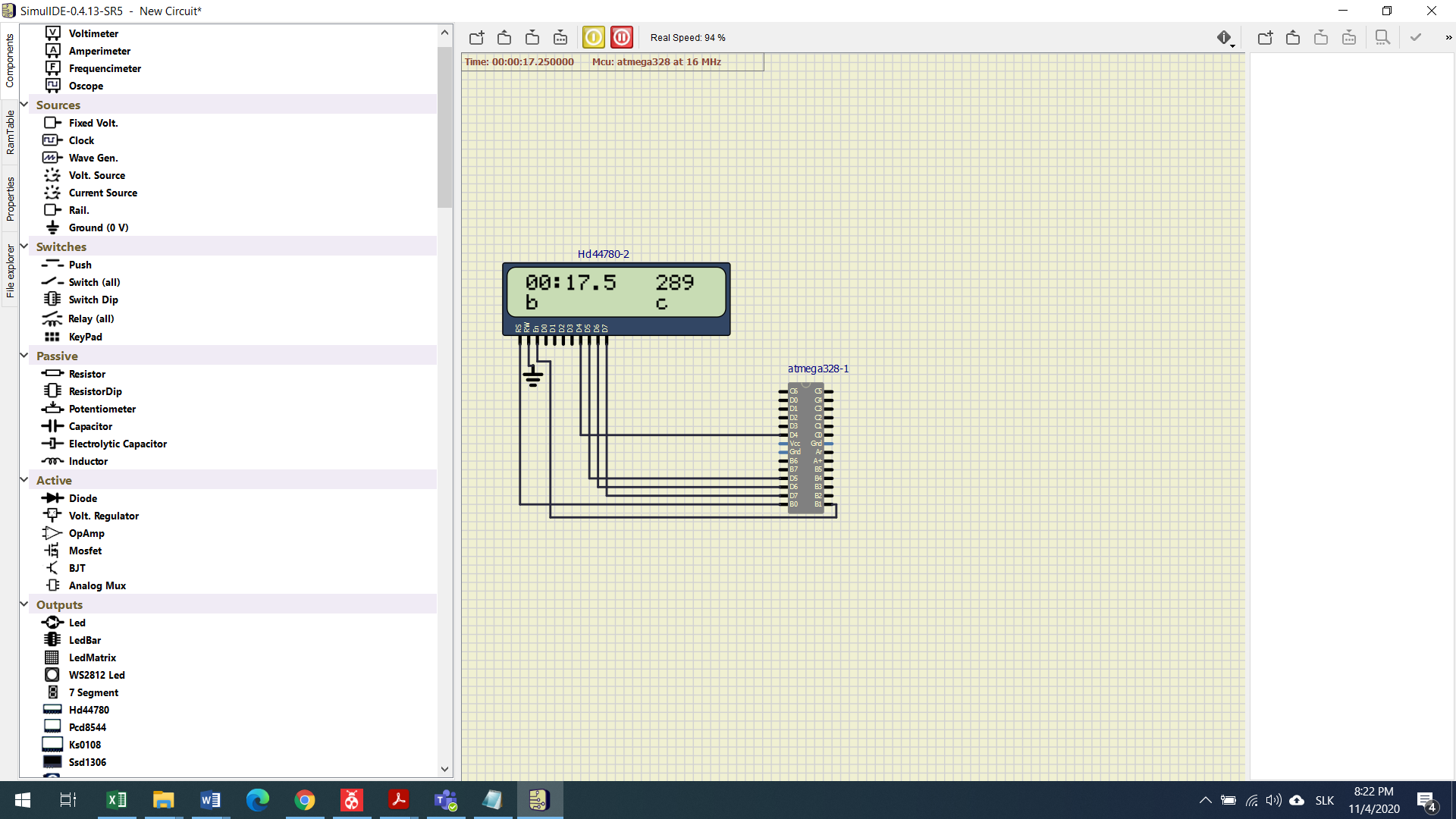
ASCII je kodovací systém znakov anglickej abecedy, číslic slúžiacich k riadeniu dátového prenosu

Obrázok časového signálu pri prenose dat ‘DE2‘



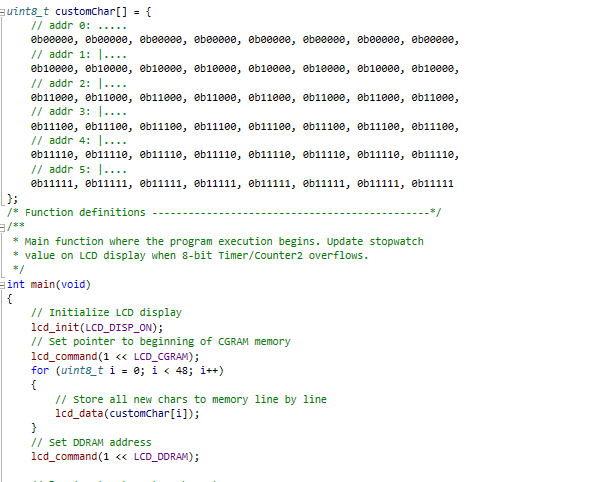


Lcd test



Timer with powered seconds.

Custom chararacter definition.



Stop watch code:

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* Stopwatch with LCD display output.

\* ATmega328P (Arduino Uno), 16 MHz, AVR 8-bit Toolchain 3.6.2

\*

\* Copyright (c) 2017-2020 Tomas Fryza

\* Dept. of Radio Electronics, Brno University of Technology, Czechia

\* This work is licensed under the terms of the MIT license.

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Includes ----------------------------------------------------------\*/

#include <avr/io.h> // AVR device-specific IO definitions

#include <avr/interrupt.h> // Interrupts standard C library for AVR-GCC

#include "timer.h" // Timer library for AVR-GCC

#include "lcd.h" // Peter Fleury's LCD library

#include <stdlib.h> // C library. Needed for conversion function

/\* Function definitions ----------------------------------------------\*/

/\*\*

\* Main function where the program execution begins. Update stopwatch

\* value on LCD display when 8-bit Timer/Counter2 overflows.

\*/

int main(void)

{

// Initialize LCD display

lcd\_init(LCD\_DISP\_ON);

// Put string(s) at LCD display

lcd\_gotoxy(1, 0);

lcd\_puts("00:00.0");

lcd\_gotoxy(11, 0);

lcd\_putc('a');

lcd\_gotoxy(1, 1);

lcd\_putc('b');

lcd\_gotoxy(11, 1);

lcd\_putc('c');

// Configure 8-bit Timer/Counter2 for Stopwatch

// Set prescaler and enable overflow interrupt every 16 ms

TIM2\_overflow\_16ms();

TIM2\_overflow\_interrupt\_enable()

// Enables interrupts by setting the global interrupt mask

sei();

// Infinite loop

while (1)

{

/\* Empty loop. All subsequent operations are performed exclusively

\* inside interrupt service routines ISRs \*/

}

// Will never reach this

return 0;

}

/\* Interrupt service routines ----------------------------------------\*/

/\*\*

\* ISR starts when Timer/Counter2 overflows. Update the stopwatch on

\* LCD display every sixth overflow, ie approximately every 100 ms

\* (6 x 16 ms = 100 ms).

\*/

ISR(TIMER2\_OVF\_vect)

{

static *uint8\_t* number\_of\_overflows = 0;

static *uint8\_t* tens = 0;

static *uint8\_t* secs = 0;

static *uint8\_t* min = 0;

*uint16\_t* squaresecs = secs \* secs;

//static uint8\_t secs\_1 = 0;

char lcd\_string[2] = " "; //string for converting numbers

number\_of\_overflows++;

if (number\_of\_overflows >= 6)

{

// Do this every 6 x 16 ms = 100 ms

number\_of\_overflows = 0;

tens ++;

if (tens > 9)

{

tens = 0;

secs ++;

}

*itoa*(tens, lcd\_string, 10);

lcd\_gotoxy(7, 0);

lcd\_puts(lcd\_string);

if (secs > 59)

{

secs = 0;

min ++;

lcd\_gotoxy(4, 0);

lcd\_putc('0');

}

*itoa*(secs, lcd\_string, 10);

if (secs > 9)

{

lcd\_gotoxy(4,0);

}

else

{

lcd\_gotoxy(5,0);

}

lcd\_puts(lcd\_string);

if (min > 60)

{

min = 0;

lcd\_putc('0');

lcd\_gotoxy(2, 0);

}

*itoa*(min, lcd\_string, 10);

if (min > 9)

{

lcd\_gotoxy(1,0);

}

else

{

lcd\_gotoxy(2,0);

}

lcd\_puts(lcd\_string);

*itoa*(squaresecs, lcd\_string, 10);

if (squaresecs = 0)

{

lcd\_gotoxy(11, 0);

lcd\_puts("0 ");

}

else

{

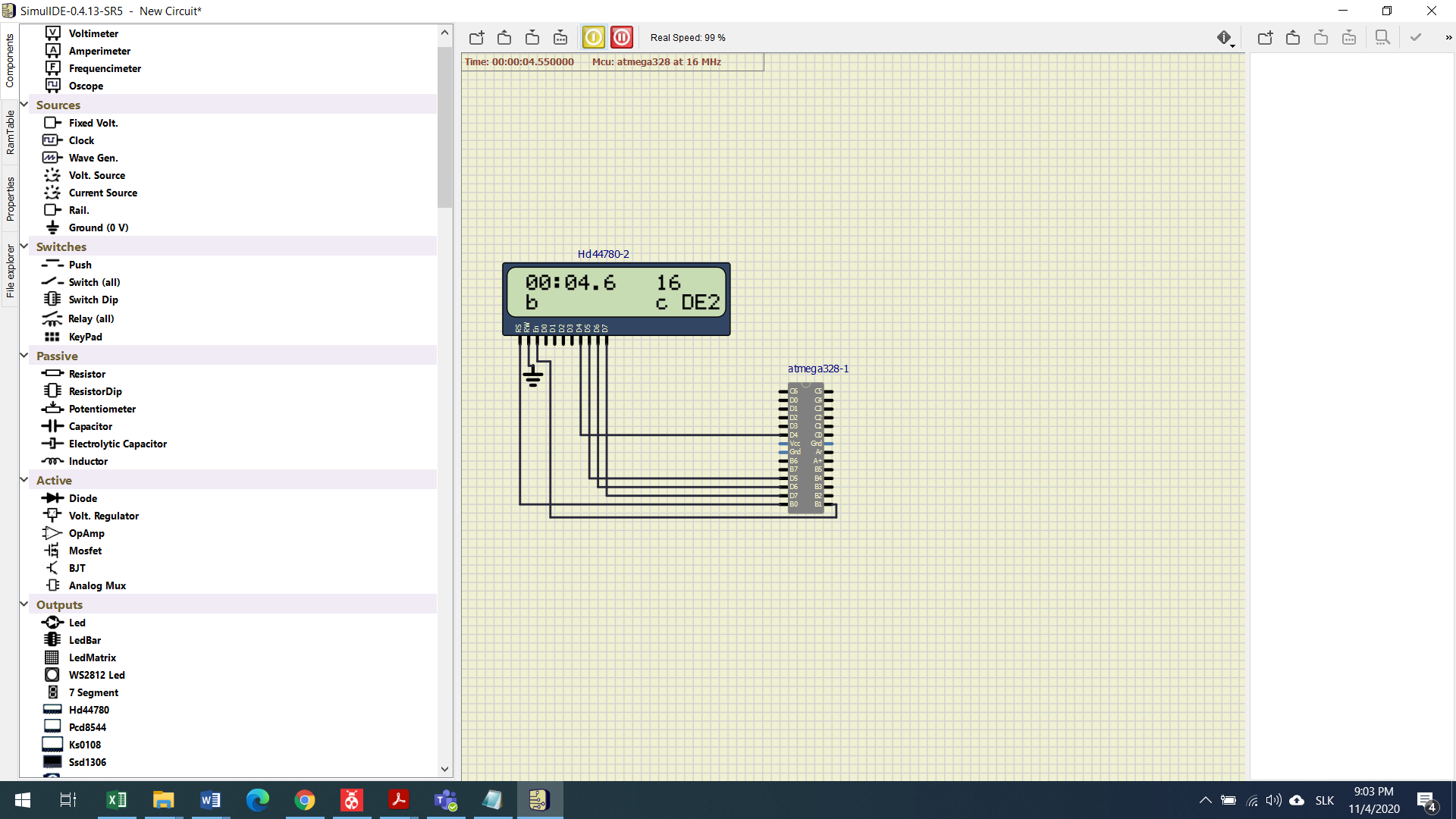
lcd\_gotoxy(11,0);

}

lcd\_puts(lcd\_string);

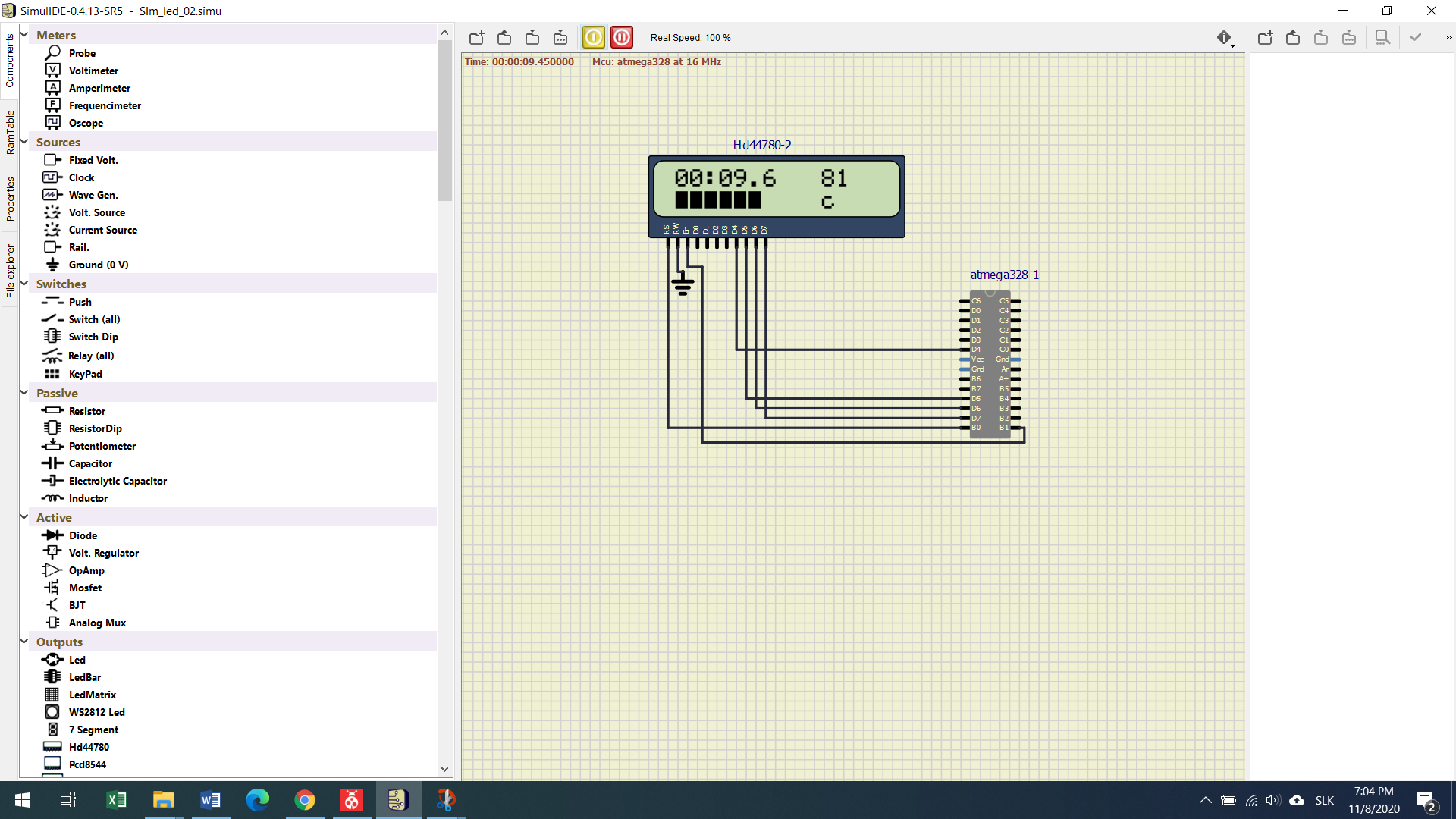
}

}









**Loading bar**

ISR(TIMER0\_OVF\_vect)

{

static *uint8\_t* symbol = 0;

static *uint8\_t* position = 0;

static *uint8\_t* ovrflov = 0;

ovrflov ++;

if(ovrflov >= 1)

ovrflov = 0;

lcd\_gotoxy(1 + position, 1);

lcd\_putc(symbol);

symbol ++;

if(symbol > 5)

{

position ++;

symbol = 0;

if(position == 9)

{

position = 0;

lcd\_gotoxy(9,1);

lcd\_putc(0);

lcd\_gotoxy(8,1);

lcd\_putc(0);

lcd\_gotoxy(7,1);

lcd\_putc(0);

lcd\_gotoxy(6,1);

lcd\_putc(0);

lcd\_gotoxy(5,1);

lcd\_putc(0);

lcd\_gotoxy(4,1);

lcd\_putc(0);

lcd\_gotoxy(3,1);

lcd\_putc(0);

lcd\_gotoxy(2,1);

lcd\_putc(0);

lcd\_gotoxy(1,1);

lcd\_putc(0);

}

}

}