1. **Preparation tasks** (done before the lab at home). Submit:
   * Table with voltage divider, calculated, and measured ADC values for all buttons.

|  |
| --- |
|  |
| **Push button** | **PC0[A0] voltage** | **ADC value (calculated)** | **ADC value (measured in simulation)** |
| Right | 0 V | 0 | 0 |
| Up | 0.495 V | 101 | 101 |
| Down | 1.203 V | 246 | 245 |
| Left | 1.970 V | 403 | 402 |
| Select | 3.182 V | 651 | 650 |
| none | 5 V | 1023 | 1022 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Operation** | **Register(s)** | **Bit(s)** | **Description** |
| Voltage reference | ADMUX | REFS1:0 | 01: AVcc voltage reference, 5V |
| Input channel | ADMUX | MUX3:0 | 0000: ADC0, 0001: ADC1, ... |
| ADC enable | ADCSRA | ADEN | 1: enables the ADC |
| Start conversion | ADCSRA | ADSC | 1: start conversion in single mode and start first conversion in free mode. |
| ADC interrupt enable | ADCSRA | ADIE | 1: interrupt is enabled … When the I-bit in SREG is set, „ADC completed“ interrupt is activated |
| ADC clock prescaler | ADCSRA | ADPS2:0 | 000: Division factor 2, 001: 2, 010: 4, ... (for us: 111) |
| ADC result | ADCH  ADCL | ADCH7:0  ADCL7:0 | When an ADC conversion is complete, the result is found in these two registers. |

|  |  |
| --- | --- |
|  | |
| **Function name** | **Function parameters** | **Description** | **Example** |
| uart\_init | UART\_BAUD\_SELECT  (9600, F\_CPU) | Initialize UART to 8N1 and set baudrate to 9600 Bd | uart\_init(UART\_BAUD\_SELECT(9600, F\_CPU)); |
| uart\_getc | - | get data from UART | uart\_getc( |
| uart\_putc | c | Send one symbol to UART | uart\_putc(a); |
| uart\_puts | “string“ | Send given string to UART | uart\_putc(“DE\_2“); |

1. ADC. Submit:
   * Listing of ADC\_vect interrupt routine with complete code for sending data to the LCD/UART and identification of the pressed button.

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

/\*\*

\* ISR starts when ADC completes the conversion. Display value on LCD

\* and send it to UART.

\*/

ISR(ADC\_vect)

{

*uint16\_t* value = 0;

char lcd\_string[4] = " ";

// Copy ADC result to 16-bit variable

value = ADC;

// Print to LD in decimal

*itoa*(value, lcd\_string, 10); // Convert to string in decimal

lcd\_gotoxy(8, 0);

lcd\_puts(" ");

lcd\_gotoxy(8, 0);

lcd\_puts(lcd\_string);

// Send to uart in decimal

if (value < 700)

{

uart\_puts("ADC value in decimal:");

uart\_puts(lcd\_string);

uart\_puts("\n"); // \n... next line , \r line before

}

// Print on lcd in hex

*itoa*(value, lcd\_string, 16);

lcd\_gotoxy(13, 0);

lcd\_puts(" ");

lcd\_gotoxy(13, 0);

lcd\_puts(lcd\_string);

// Odd Parity bit counter

if(value %2 == 0)

{

lcd\_gotoxy(15,1);

lcd\_puts("1");

}

else

{

lcd\_gotoxy(15,1);

lcd\_puts("0");

}

// print pressed key

lcd\_gotoxy(8, 1);

lcd\_puts(" ");

if (value >= 1016)

{

lcd\_gotoxy(8, 1);

lcd\_puts("None"); // None

}

else if (value == 0)

{

lcd\_gotoxy(8, 1);

lcd\_puts("Right"); // Right

}

else if (value == 101)

{

lcd\_gotoxy(8, 1);

lcd\_puts("Up"); // Up

}

else if (value == 245)

{

lcd\_gotoxy(8, 1);

lcd\_puts("Down"); // Down

}

else if (value == 402)

{

lcd\_gotoxy(8, 1);

lcd\_puts("Left"); // Left

}

else if (value == 650)

{

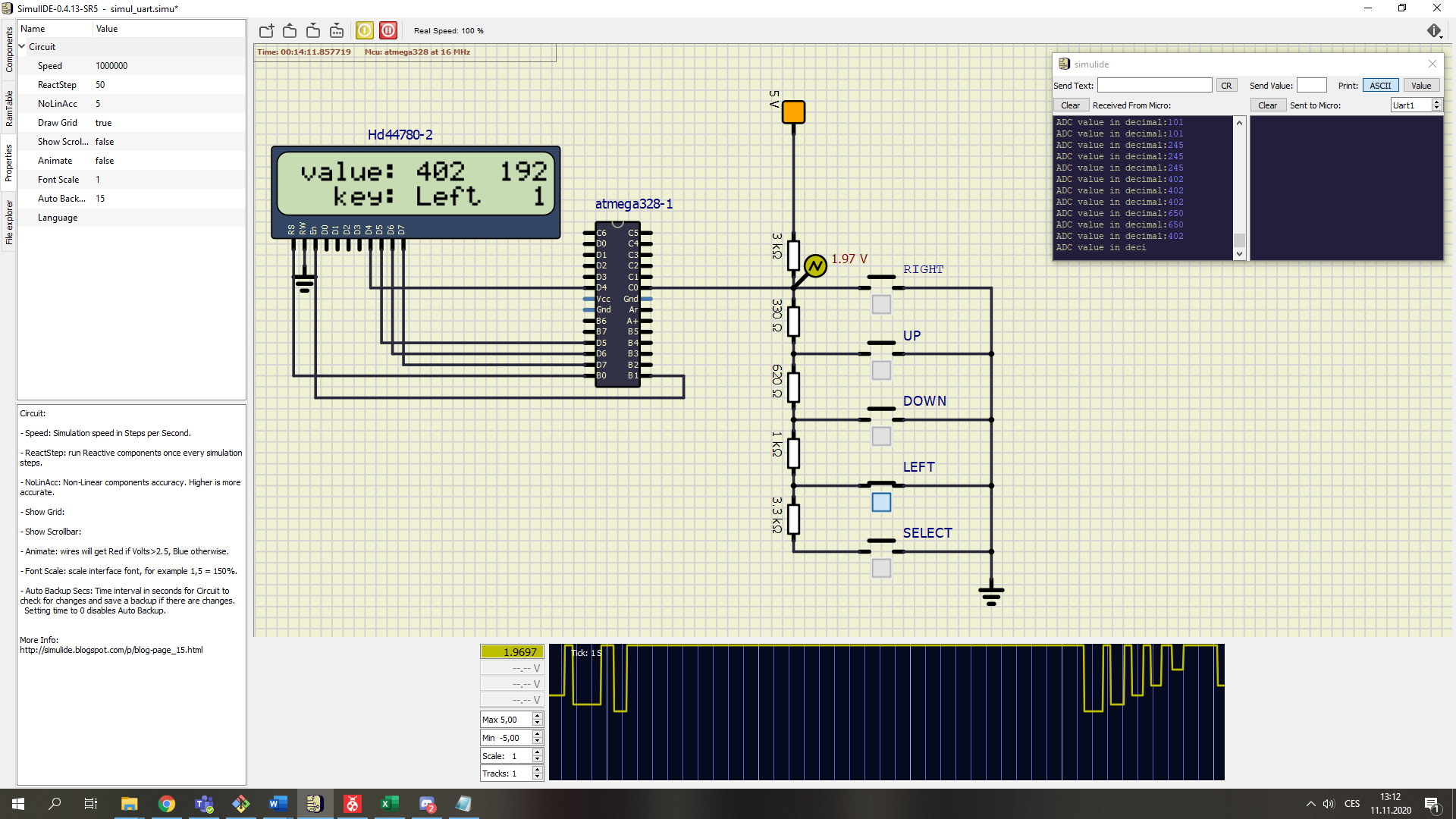
lcd\_gotoxy(8, 1);

lcd\_puts("Select"); // Select

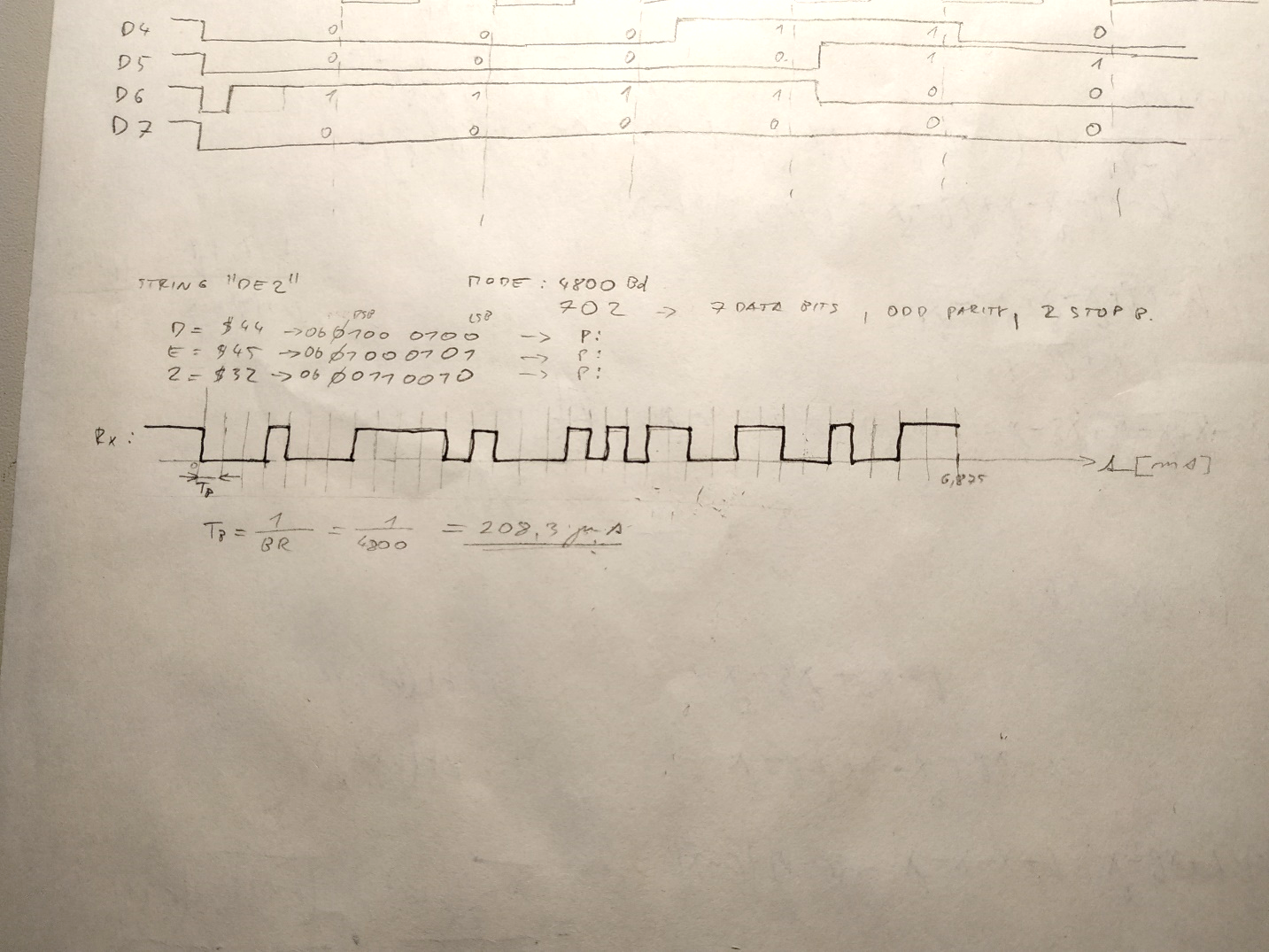
}

}

* + Screenshot of SimulIDE circuit when "Power Circuit" is applied.



1. UART. Submit:
   * (Hand-drawn) picture of UART signal when transmitting data DE2 in 4800 7O2 mode (7 data bits, odd parity, 2 stop bits, 4800 Bd),



* + Listing of code for calculating/displaying parity bit.

// Odd Parity bit counter

if(value %2 == 0)

{

lcd\_gotoxy(15,1);

lcd\_puts("1");

}

else

{

lcd\_gotoxy(15,1);

lcd\_puts("0");

}