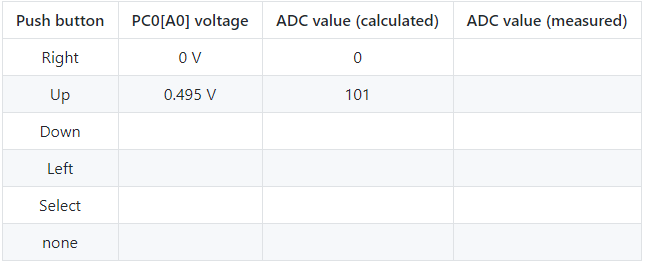
**DIGITAL ELECTRONICS 2 LAB ASSIGNMENT 7**

Name: Demirkan Korbey Baglamac Github Repository Link:

1. 

1022

650

402

0

245

101

1023

5 V

651

403

246

3.182 V

1.97 V

1.204 V



|  |  |  |  |
| --- | --- | --- | --- |
| Operation | Register(s) | Bit(s) | Description |
| Voltage reference | ADMUX | REFS1:0 | 00: AREF, Internal Vref turned off  01: Avcc voltage reference, 5V  10: Reserved  00: Internal 1.1V Vref with external capacitor at AREF pin |
| Input channel |  | MUX3:0 | 0000: ADC0, 0001: ADC1,  0010: ADC2, 0011: ADC3,  0100: ADC4, 0101: ADC5,  0110: ADC6, 0111: ADC7,  1000: ADC8  1001, 1010, 1011, 1100, 1101: reserved  1110: 1.1V (Vbg)  1111: 0V (GND) |
| ADC enable | ADCSRA | ADEN | 1: ADC is turned on  0: ADC is turned off |
| Start conversion |  | ADSC | 1: Starts conversion  0: Writing 0 has no effect, when conversion is complete, it returns to 0 automatically. |
| ADC interrupt enable |  | ADIE | 1: If I-bit in SREG is set, activates ADC Interrupt |
| ADC clock prescaler |  | ADPS2:0 | Division Factor  000: 2  001: 2  010: 4  011: 8  100: 16  101: 32  110: 64  111: 128 |
| ADC result | ADCL and ADCH | ADC9:0 | These bits represent the result from the ADC conversion. |

**ADC\_vect interrupt routine code:**

I include the <string.h> library, to use strcpy function. (to change the value of the char array)

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

/\*\*

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

\* and send it to UART.

\*/

ISR(ADC\_vect)

{

// WRITE YOUR CODE HERE

*uint16\_t* value = 0;

char lcd\_string[4] = "0000";

char pressed\_button[6] = " ";

// Copy ADC result to 16-bit variable

value = ADC;

// Displaying ADC result as decimal

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

lcd\_gotoxy(8,0);

lcd\_puts(" ");

lcd\_gotoxy(8,0);

lcd\_puts(lcd\_string);

// Determining the pressed key

// None of the buttons are pressed

if(value > 700)

*strcpy*(pressed\_button, "none");

// Select button is pressed

if(value < 700 && value > 452)

*strcpy*(pressed\_button, "select");

// Left button is pressed

if(value < 452 && value > 295)

*strcpy*(pressed\_button, "left");

// Down button is pressed

if(value < 295 && value > 151)

*strcpy*(pressed\_button, "down");

// Up button is pressed

if(value < 151 && value > 50)

*strcpy*(pressed\_button, "up");

// Right button is pressed

if(value < 50)

*strcpy*(pressed\_button, "right");

//Displaying the pressed button on LCD

lcd\_gotoxy(8,1);

lcd\_puts(" ");

lcd\_gotoxy(8,1);

lcd\_puts(pressed\_button);

// UART

if (value < 700)

{

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

uart\_puts(lcd\_string);

uart\_puts("\r\n");

uart\_puts("Pressed Button: ");

uart\_puts(pressed\_button);

uart\_puts("\r\n");

}

// Displaying ADC result as hexadecimal

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

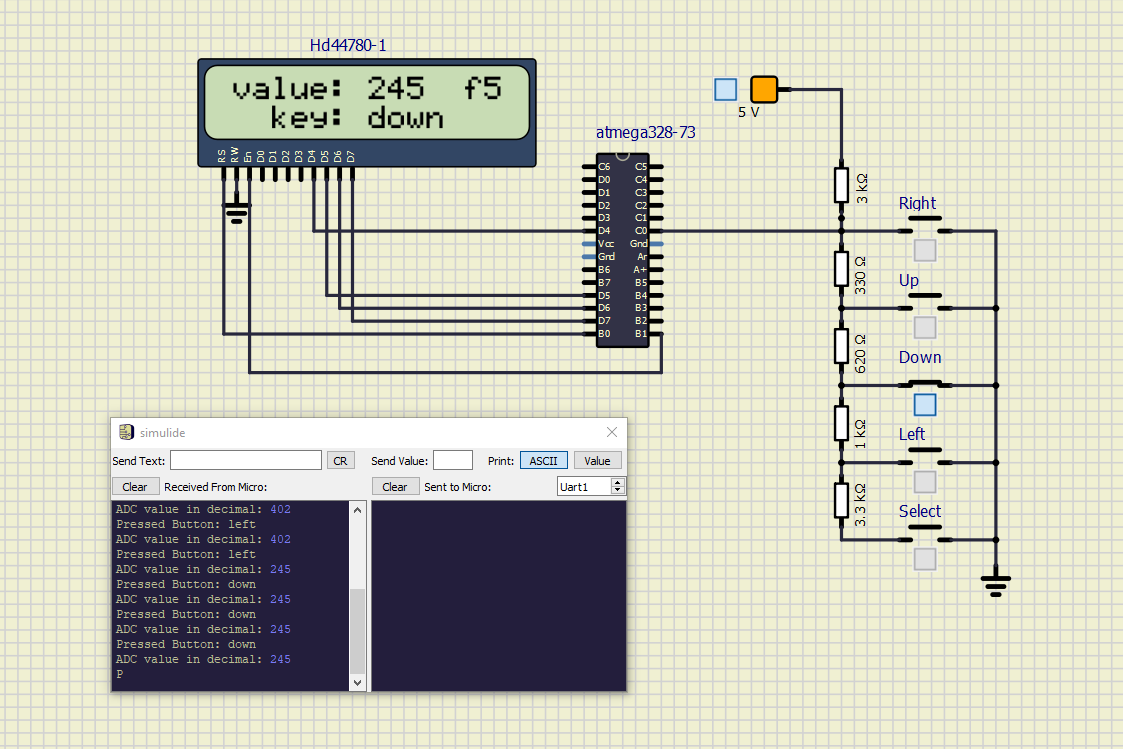
lcd\_gotoxy(13,0);

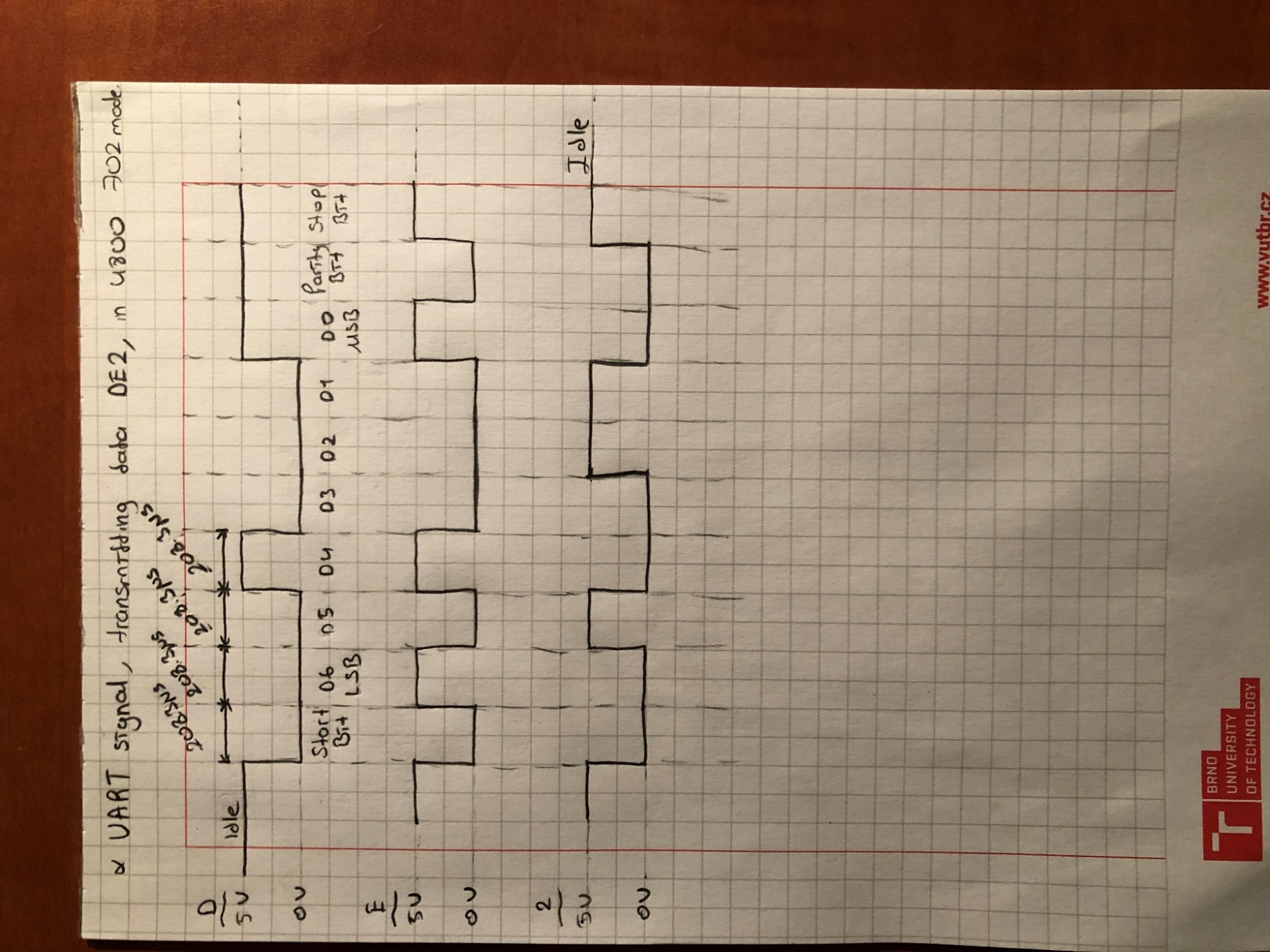
lcd\_puts(" ");

lcd\_gotoxy(13,0);

lcd\_puts(lcd\_string);

}

**Screenshot of the circuit:**

1. 

**Listing of code for calculating/displaying parity bit:** I assume we are using even parity.

ISR(ADC\_vect)

{

…

…

… Same code, written in the ADC\_vect interrupt routine code section.

…

…

// Calculating Parity Bit for ADC value

// Lets assume we use even parity bit

*uint8\_t* cnt = 0;

while(value > 0)

{

if(value & 1)

cnt++;

value = value >> 1;

}

// Displaying Parity Bit for ADC value

if((cnt % 2) == 0)

cnt = 0;

else

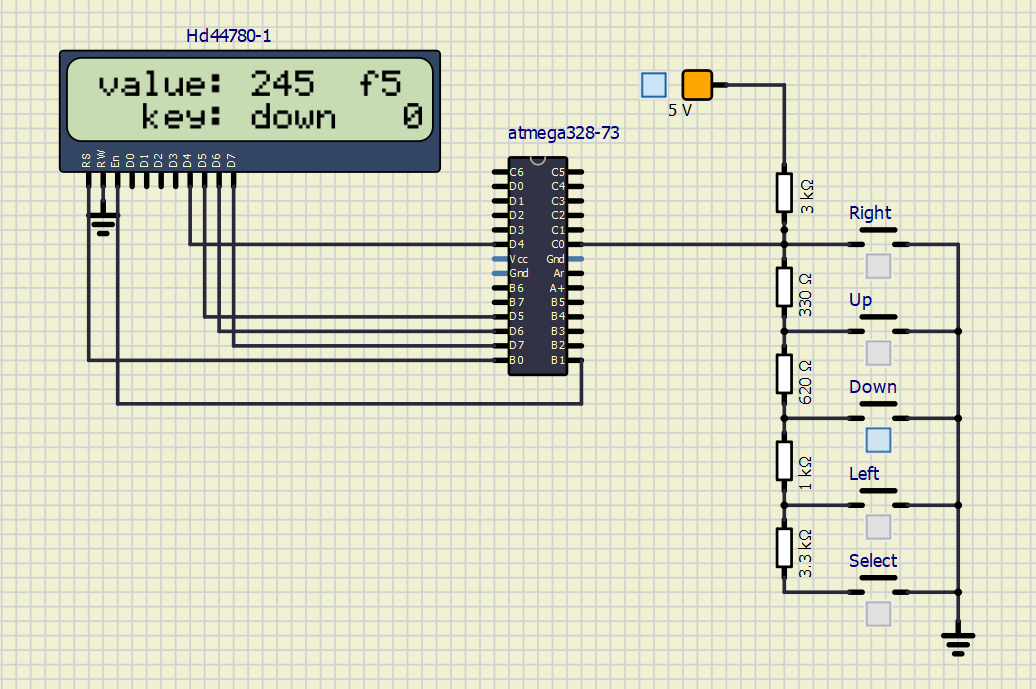
cnt = 1;

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

lcd\_gotoxy(15,1);

lcd\_puts(lcd\_string);

}

**Screenshot of the circuit:**