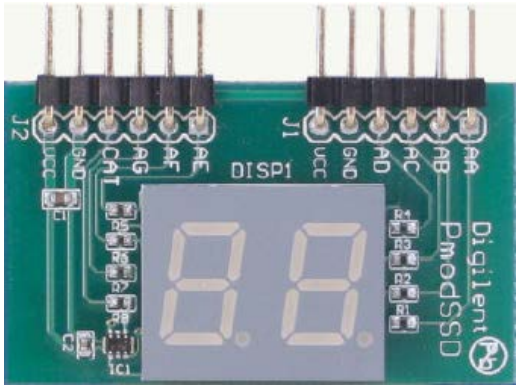
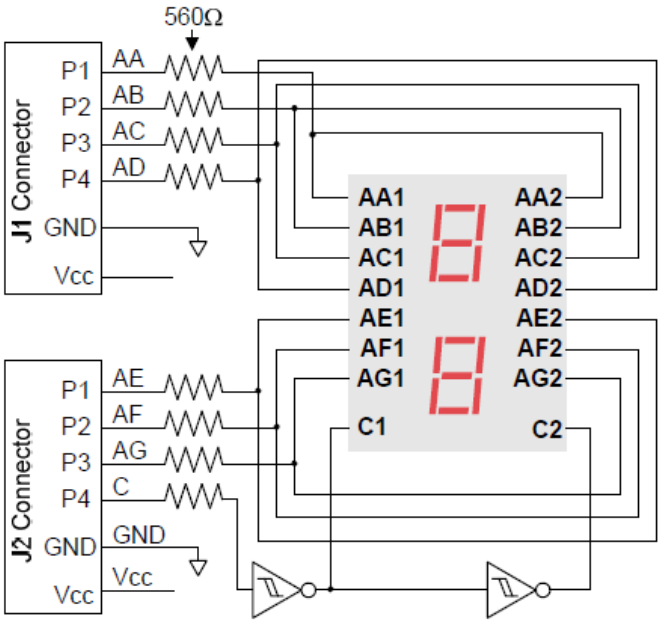
Laboratory 2 – applications with simple I/O modules.

1. The 2x7 segment display

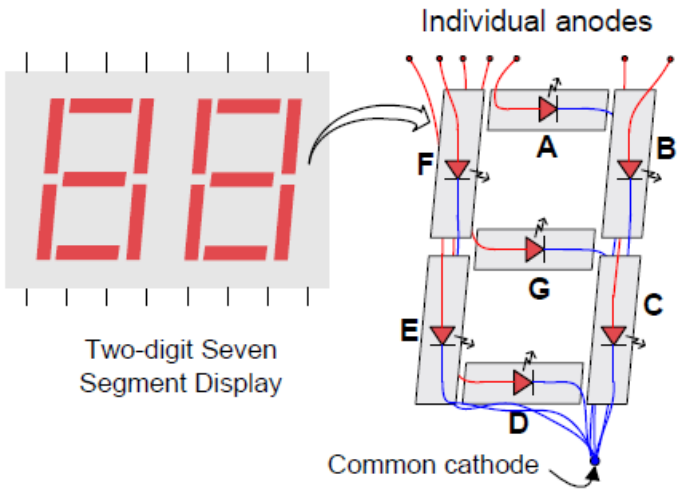
The PmodSSD (Seven Segment Display) from the figure bellow provides the possibility of displaying two characters. Each segment is a LED, which gives light if there is the right combination of voltages on the segments anode and cathode (anode High, cathode LOW).



In order to reduce the number of pins necessary, the two digits cannot be lit simultaneously. The selection between the two digits is done by the CAT signal (the C on the schema below). If this signal is 0 the segments corresponding to the unities are giving light and if the CAT signal is 1 the segments corresponding to the digit on the left are lit.



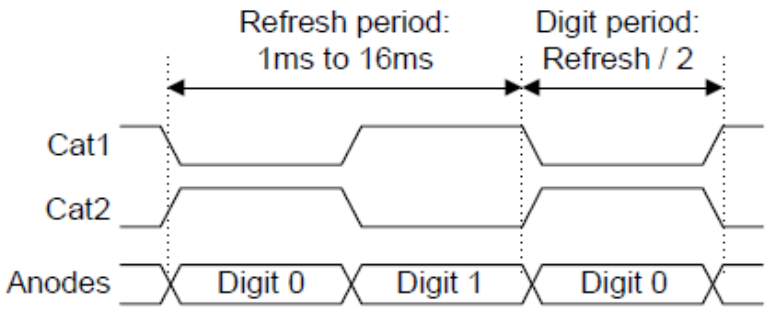
The signs denoted by the letters AA…AC correspond to the anodes of the segment LEDs. The correspondence of these signals is indicated by the figure bellow:



As an example, for displaying the digit 3 we will need the segments to receive the following logical levels:

G	F	E	D	C	B	A
1	0	0	1	1	1	1

In order to show a number of two digits we need to switch(multiplex) very fast between the two blocks using the CAT signal. The following time diagram illustrates the process.



To realize the functionality exemplified in the above diagram we use the following pseudo code.

Loop:
    AA…AG = cod_cifra_unitati
    CAT=0
    Delay()
    AA…AG=cod_cifra_zeci
    CAT=1
    Delay()
Goto loop

2. The usage the microcontroller port for Input / Output operations

Arduino through its digitalRead/ digitalWrite functions hides the mechanism through which the microcontroller does these operations. Furthermore these functions induce delays which may be significant when we need to transfer data on several bits.
Any microcontroller is connected to the outside through ports of intput / output. The AVR Atmega 2560 is an 8-bit microcontroller on so it has its 8-bit ports. Each port is associated with three registers (x will be replaced with A, B, C, D, ..., depending on the port used):
– DDRx – direction reister

– PORTx – output register
– PINx – register for input data

The direction register DDRx
DDRx (Data Direction Register) configures the data direction of the port pins (if a bit of a port will be used for input or output). A written on a bit of 0 makes the pin DDRx port corresponding to the input pin, and a bit set to 1 makes the corresponding pin to be output pin.
Examples:

- In order to configure all the bits port A as input pins
  DDRA = 0b00000000;
- To configure all the bits of port A as output
  DDRA = 0b11111111;
- To configure the lower half of port B as output and the uppe half as input
  DDRB = 0b00001111;

The input register PINx
PINx (Port IN) is used to read data from the set the set of pins configured as input. To read data, these pins must be set as input pins, setting all of DDRx bits to zero. Example: reading data from port A:
DDRA = 0;
char a = PINA;

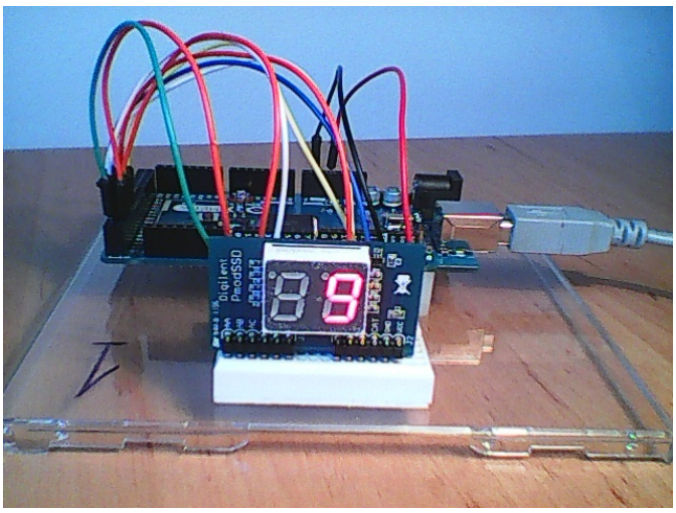
The output register PORTx
Register PORTx is used to transmit data to the microcontroller peripherals x connected to the port pins. For data output to be visible, corresponding bits of DDRx register must be set the direction of value 1. Example: switching of 2 in 2 of 8 LEDs connected to Port A
DDRA = 0xFF
PORTA = 0b10101010

3. Connecting the Seven Segments display to the Arduino board

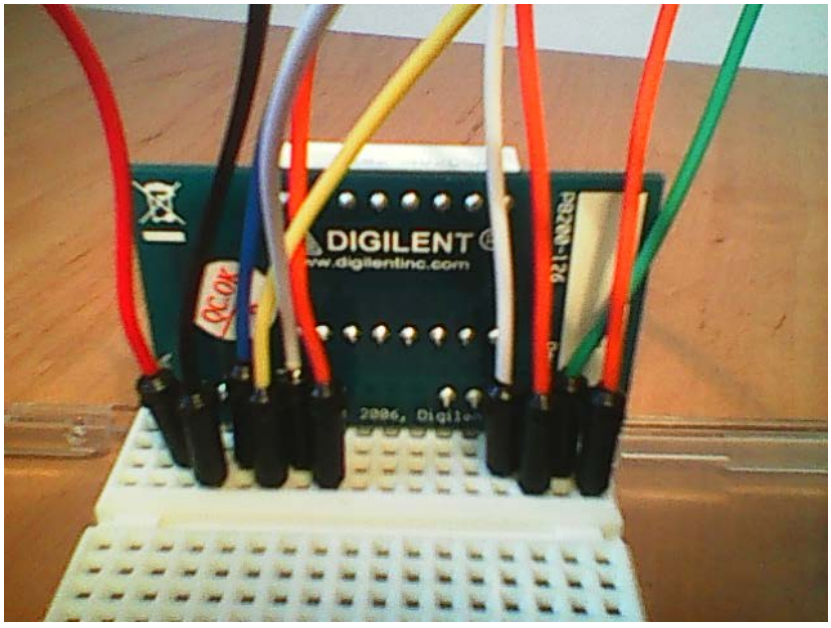
The assembly is done with the boars unplugged from the PC. The first step is the introduction of the display into the breadboard, gently without hitting or damaging the system. The we will connect the power wires in the semi-columns corresponding to GND and VCC for the second connector (J2) of the display. The other end of the powering wires will be connected to the 5V and GND of the Arduino board. We are not required to apply extra voltage to the J1 connector.

For the wires corresponding to the anodes of the display AA…Ag and for the cathode CAT, we will connect, successively wires that we will introduce into Arduinos digital pins starting from position 22, 23,…29 corresponding to bits 0…7 of port A.

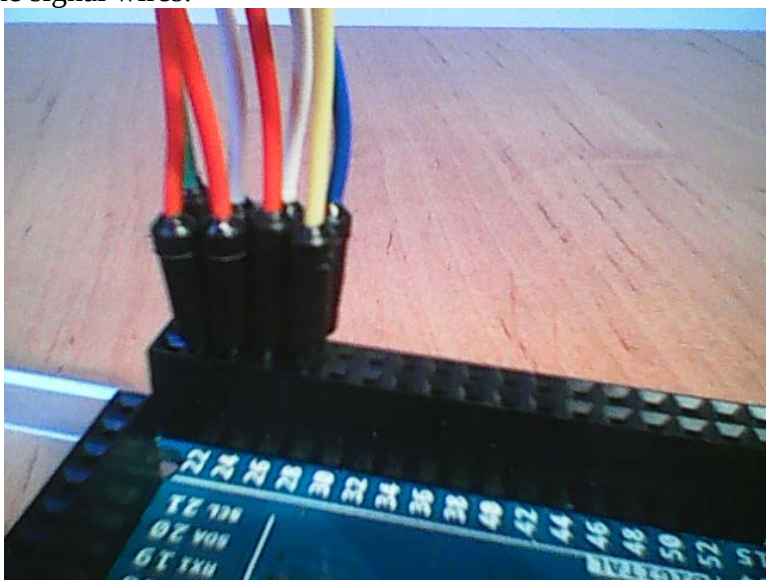
The integral assembly:



Details of how the wires are connected:



The connection of the signal wires:



Please Note:
The first two pins of the connector from the Arduino board are power pins. The digital pins start from the second pair. Pay much attention of their position.

Create a new project (sketch) in Arduino, and introduce the following code:

```
// Display on SSD
// connected at PORTA

// Table of values, or look up table (LUT) with the BCD codes for every digit from 0 to 9. Every bit corresponds to a LED, 1 means the LED is lit and 0 means it is not giving light.

const unsigned char ssdlut[] = {0b00111111, 0b00000110, 0b01011011, 0b01001111, 0b01100110, 0b01101101, 0b01111101, 0b00000111, 0b01111111, 0b01101111};
// the size of the lut
const int lutsize = 10;

int cpos = 0; // current position
int cdigit = 0; // first digit from the two
unsigned char outvalue = 0;

void setup() {
  // setting port A as output
  DDRA = 0b11111111;
}

void loop() {

  outvalue = cdigit>0 ? 0x80 : 0;
// which cathode are we choosing ? (00000000 sau 10000000)
// the cathode is wired to bit7 from port A, through this operation we are setting bit 7 on logical 1 or
//0, alternatively, the following bits will be attached through a logical OR operation in the following
//line of code
  PORTA = (ssdlut[cpos] | outvalue); // we make an OR between the value from the LUT and the selected cathode

  cpos++; // we increment the current position

  if (cpos>=lutsize) { // if we reached the final position
    cpos = 0; // we come back at 0
    cdigit^=1; // if the previus digit was 0 we make it a 1 and vice versa
  }

 // wait 0.5 sec
 delay(500);
}
```

This small program will display the digits from 0 to 9 on the first element and then do the same on the second.

Individual Work

1) Realize and run the presented assembly from the document

2) Fill in the LUT with the corresponding values for hexadecimal numbers(A,B,C,D,E). Test the program with these kind of numbers.
3) Change the program so that you can show any number from 0 to 99 on the display, in a decimal format. Take into account that it is not the same as its representation as digits on half of byte(this thing is true in the case of hexadecimal numbers). For example, the number 16 is represented binary as 00010000, which will lead to the display of number 10. For correctly displaying the decimal numbers the following steps will need to be realized:
   -find the tens figure = the quotient from the division with 10
   -find the number of unities = the remainder from the division with 10

   CZ = nr div 10
   CU = nr mod 10 = nr – (CZ*10)

4) Connect the button block PmodBTN, that you have used in the first lab, to the Arduino board, using the desired digital pins. Use a button to increment the displayed value and a button to decrement it. Do not forget to supply the buttons with voltage.
5) Use the display mechanism implemented at point 3 and the buttons to realize a stopwatch that cound second. Use a button for start, one for stop and one for reset. Note! The counting used to wait for a second does not need to produce delays in the display process. Hint: use the millis function.