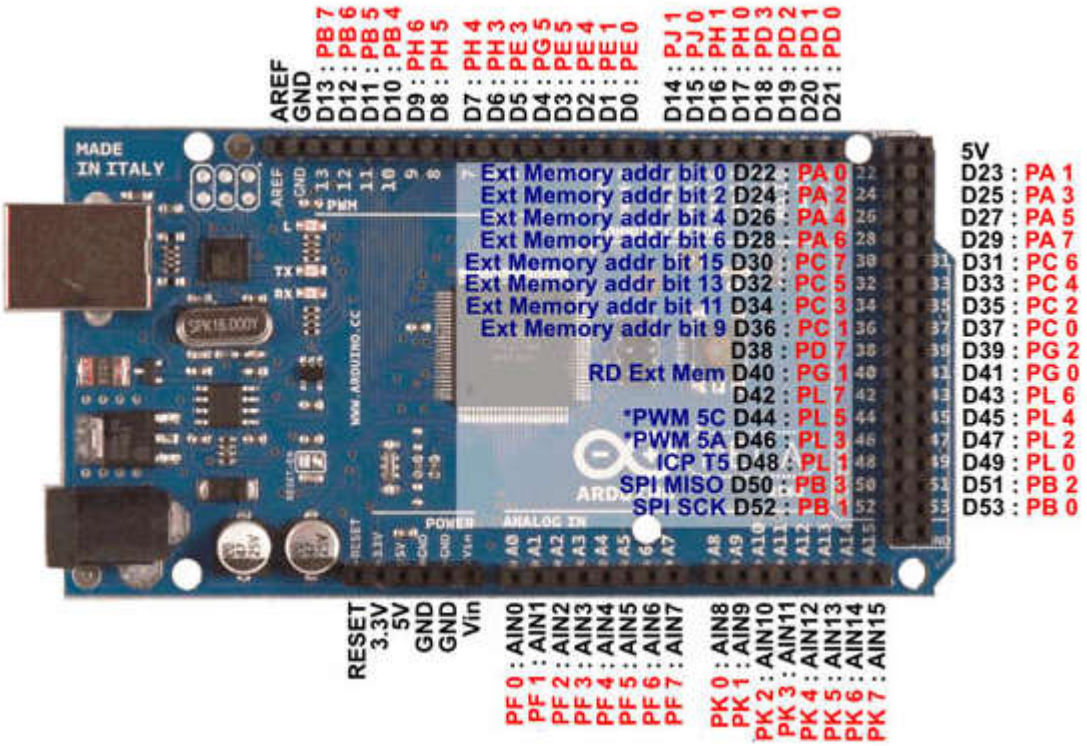
Laboratory 1 – Introduction to the Arduino boards

The set of Arduino development tools include µC (microcontroller) boards, accessories (peripheral modules, components etc.) and open source software tools which allows users to implement projects using a high-level unified approach, which is µC-model independent. The Arduino boards are mainly equipped with AVR (Atmel) MicroController Units (MCUs), but there are variants equipped with ARM or x86 MCUs. Besides the genuine versions of the Arduino boards there are a lot of third party ones (XDRuino, Freeduino, Funduino etc.) which are low cost and fully compatible (although their reliability could be questionable).

The used board at the lab is called Arduino Mega 2560 featuring the 8-bit Atmel ATMega 2560 MCU (8 bit due to the size of the internal registers). The board gives you access to 54 digital pins of the MCU for I/O operations and 16 pins for analog signals reading. Some pins can have multiple functions (i.e. providing additional communication facilities: UART, SPI, I2C etc.) – see the printed material on your desks. The MCU operates at a 16 MHz clock. The board can be power through the USB interface (common usage) or through an external power source at 7 … 12 DC V (minimum current of 0.25A). The second option is necessary to power current hungry peripherals (i.e. motors, GSM shields etc.)



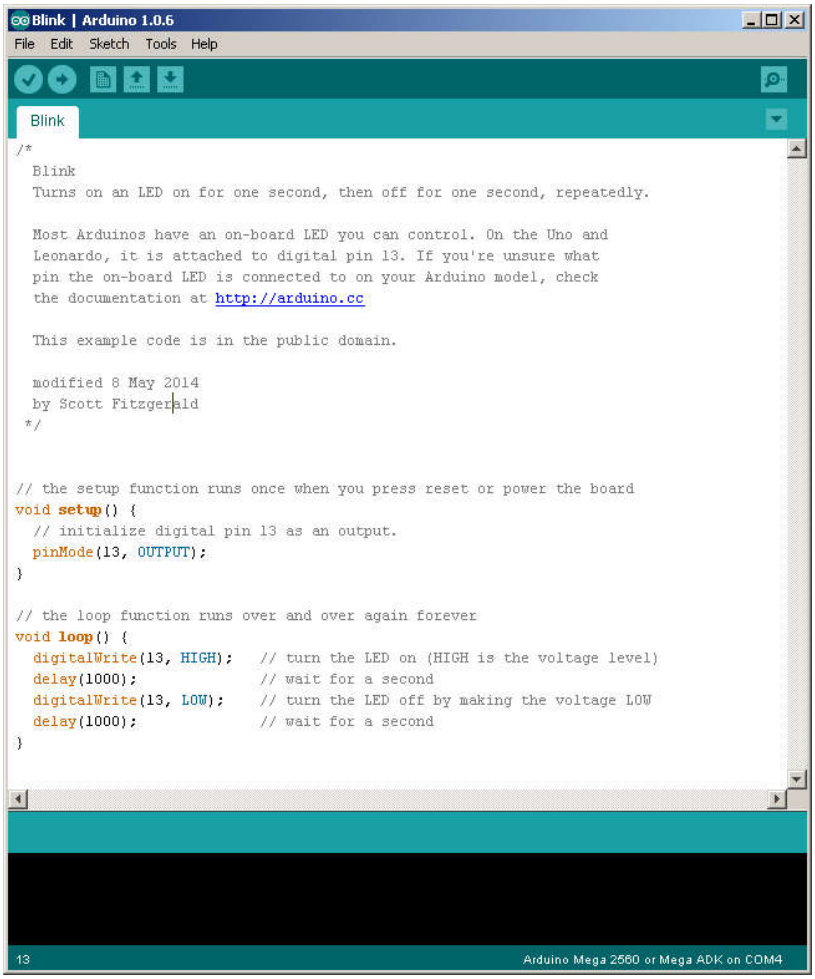
1. Using the Arduino development board – first example

For a safer usage the board is mounted on an acrylic base along with the breadboard.

For the beginning you should load the most basic example "Blink", available in the installation directory of the Arduino IDE (usually *C:\Program Files\Arduino\examples\01.Basics\Blink*). Copy the "*Blink*" folder in your working folder (compulsory **in D:\Studenti\Grupa30xxx**). Check that after the copy operation you have in for the "Blink" folder ant its contained "*blink.ino*" file write permissions!

Rule: every Arduino project (even if it has only one source file) should be placed in a folder with the same name as the source file (Ex: *D:\Studenti\Grupa30xxx\Nume_Prenume**Blink****Blink.ino***).

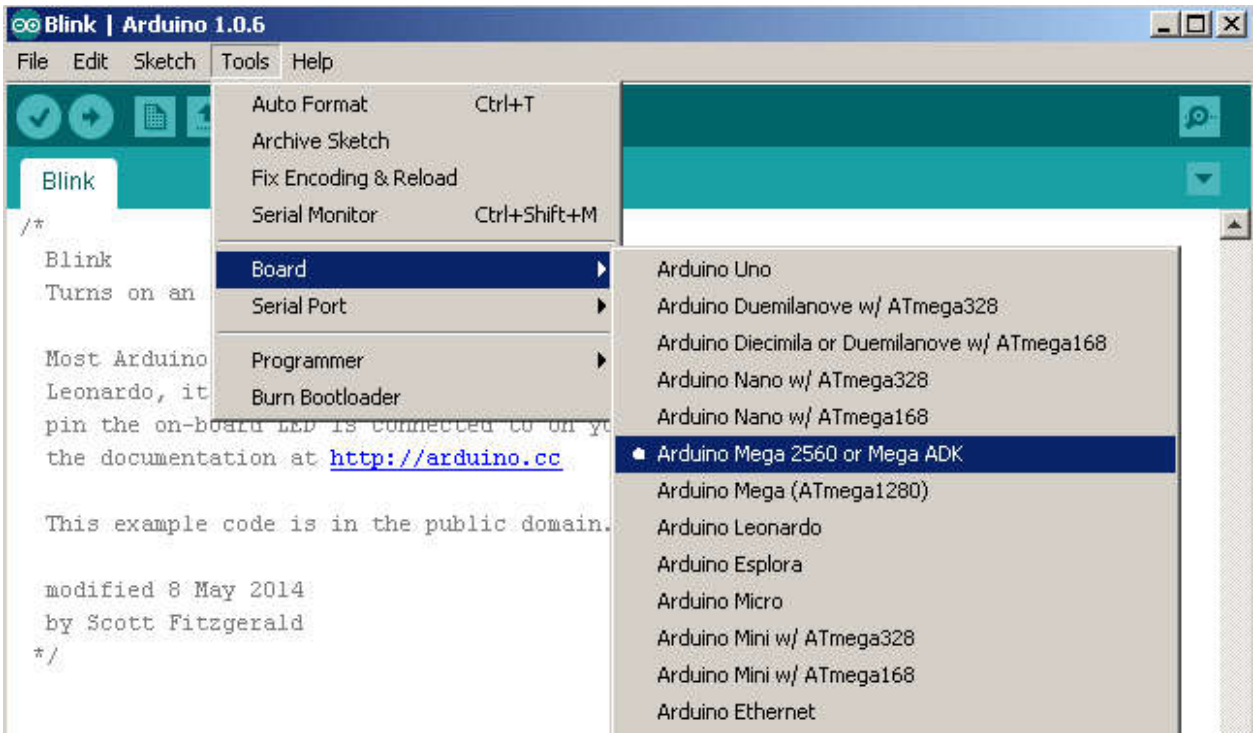
After these preparations are done, launch the Arduino IDE either by double-click on the *.ino file or by launching the IDE from the start menu/shortcut followed by an explicit *Open* operation on the *.ino file. The result should look like this one:



At this point you can connect the Arduino Mega 2560 board to the PC through the USB cable. It may be possible to be asked for the installation of a driver (in that case specify the following path: "*C:\Program Files\Arduino\drivers*"). If you encounter difficulties or not enough user rights ask the professor for help.

If the driver is correctly installed and the board is functioning (a green LED on the board is on) you can go ahead.

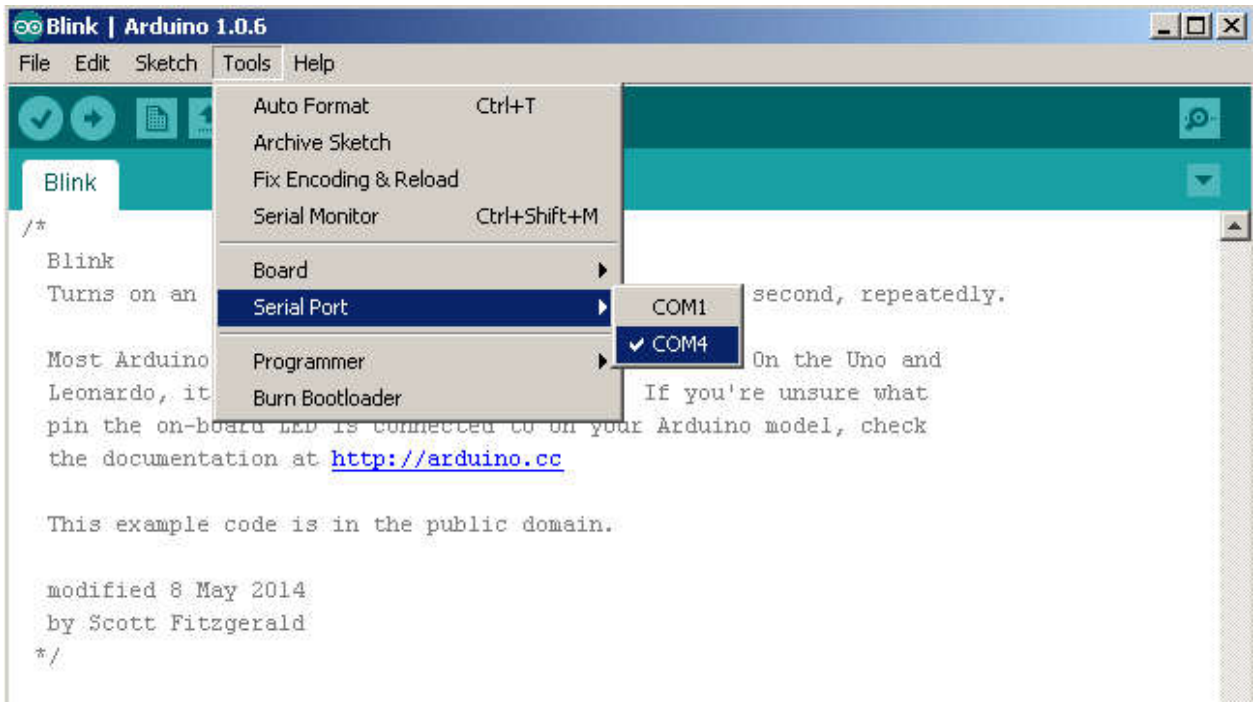
Before programming the board, you should check if the IDE is configured properly in the menu ***Tools->Board*:**



Alternatively, if you are using other development board (i.e. UNO you should make the settings accordingly).

Also the serial port use to communicate with the board should be properly configured. The USB driver/interface of the board will install a virtual serial port (COMx). COM 1 and 2 are usually the serial ports of the PC. The virtual serial ports are assigned to higher numbers (above 3).

Configure the serial port in the IDE ***Tools->Serial*** menu as in the figure bellow:



After the configuration is complete, you can compile & upload the first example using the button "Upload" (as shown in the figure below). If all the steps were completed successfully, the binary program will run on the board by blinking the on-board LED (which is connected to the digital pin 13.

Attention: there is an LCD shield added to the board. Therefore the on-board LED visibility is limited (you should look from one side to see the LED blinking).

It is completely forbidden to remove the LCD shield from the board (you may permanently damage the pins of the shield) !!!

2-nd example: digital input pins and the serial interface

As basic input device the PMod BTN peripheral module will be used. The PMod has 6 pins:

- Power (VCC) pin – requires a +5V.
- GND pin
- 4 data pins (BTN0… BTN3), indicating the button status (logic 1 = button pressed)

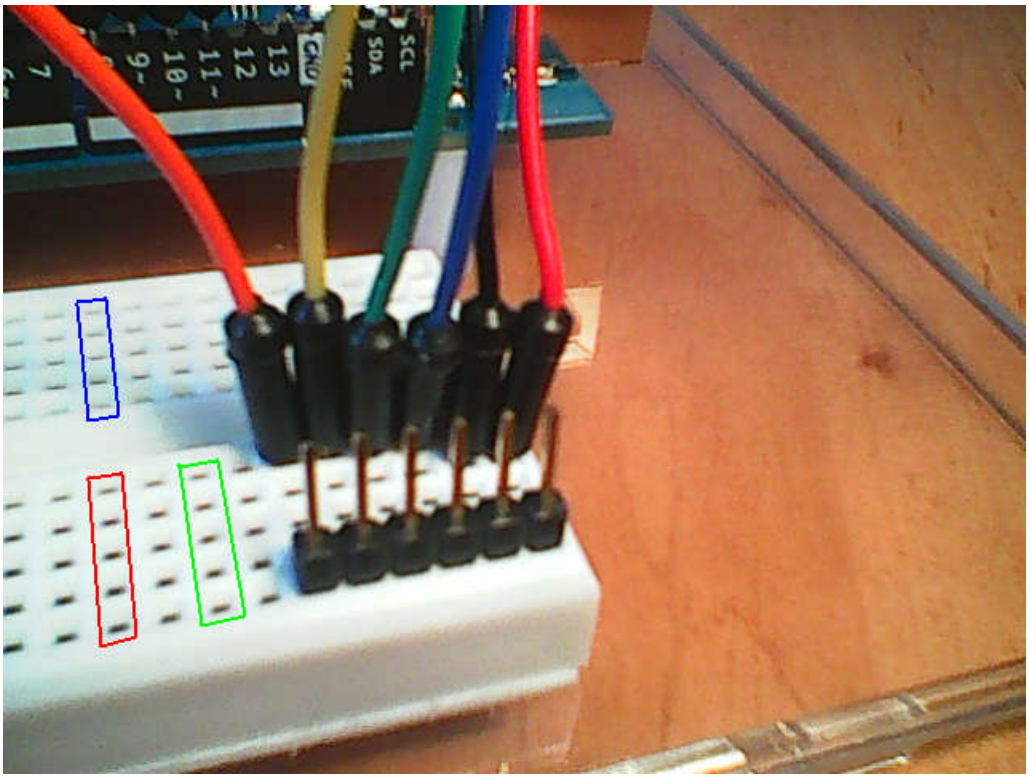
As visualization output, the serial interface will be used (allows to monitor the status of MCU pins on the PC). **All the connections between the peripheral modules, breadboard and MCu board will be done with the USB cable disconnected from thePC !!!**

To connect BTN PMod to the Arduino board the breadboard will be used:

- The 4 data pins will be connected to the digital pins 8, 9, 10, 11of the Arduino board
- GND pin of the PMod will be connected to the GND of the Arduino board
- VCC pin of the PMod will be connected to the +5V pin of the Arduino board

Because the pins of the Arduino board are not conveniently grouped in order to use a 6 pin cable, individual wires and the breadboard should be used

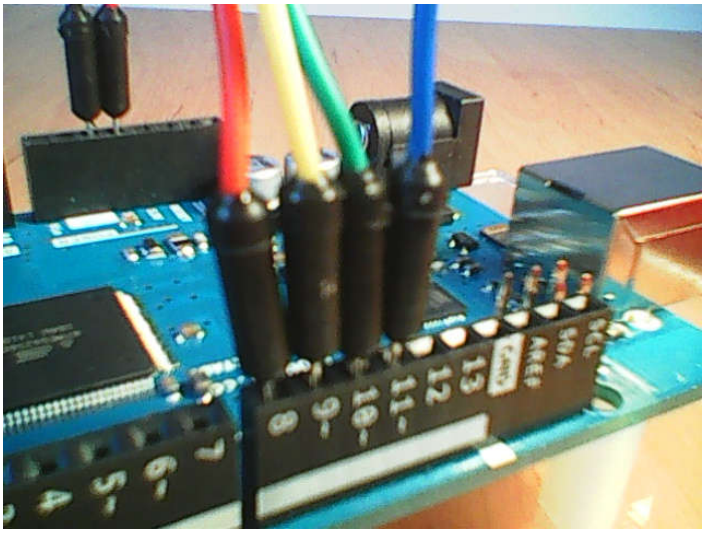
The breadboard has electrically connected pins in groups of 5 (a half column) as shown in the figure bellow:



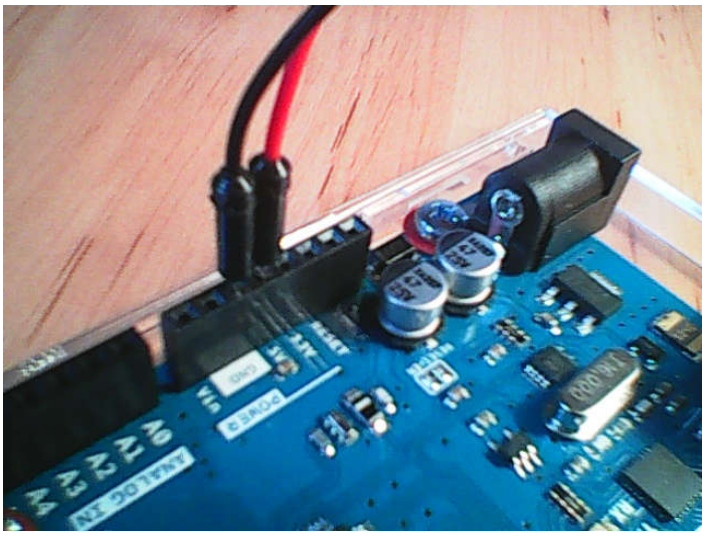
To connect the BTN PMod a 6 pin connector will be implemented using the breadboard. For that, insert a *6-pin header* in the breadboard as in the figure above. Insert a wire in each semi-column corresponding to ones connected to the 6 pin header (as in the figure above). If possible, use the following convention:

- Red wire for VCC (+5V)
- Black wire for GND
- Data pins can have other individual colors (different from red and black).

The free ends of the data wires will be connected to the Arduino board to the digital pins 8, 9, 10, 11:



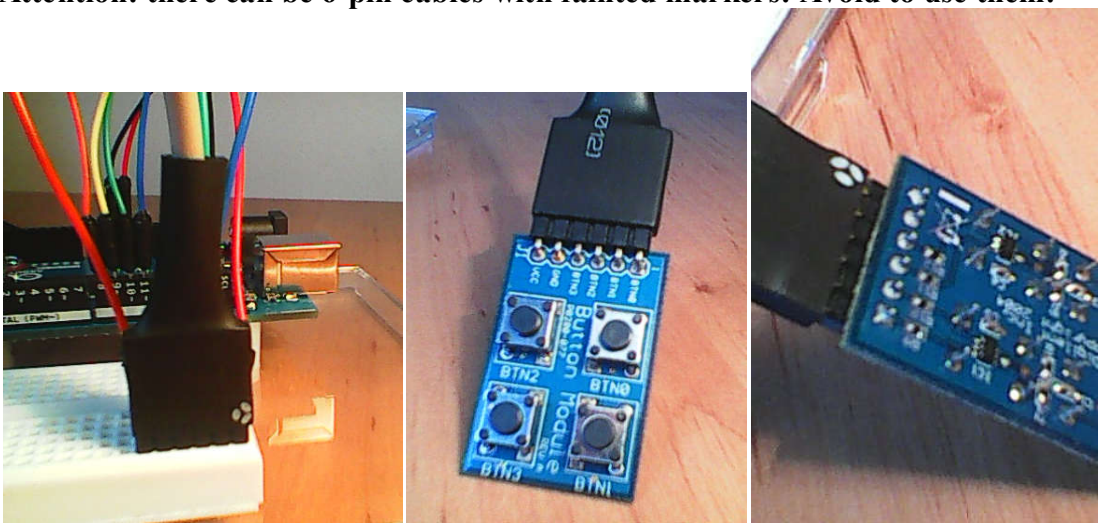
The powering wires will be connected to the pins marked with +5V (red wire) and GND (black wire) as in the figure bellow:



Attention: because of the LCD shield, the wires should be connected into the shield's pins, in the corresponding positions (the pins of the shield and of the Arduino boards are electrically connected). If you are note shore about the pin's positions, watch carefully the first figure or the printed schematic. **Double-check the correctness of the wires-setup (especially the powering wires: VCC and GND)!**

Now on you can connect the PMod BTN bloc using a 6-pin cable. To ensure the proper polarity of the powering wires (the +5V pin of the Arduino board should be electrically connected to the VCC pin of the Pmod and the GND should be connected together) **the (*) markers of the 6-pin cable will be used** and the convention that **(*) is used for VCC (+5V)**.

Attention: there can be 6-pin cables with fainted markers! Avoid to use them!

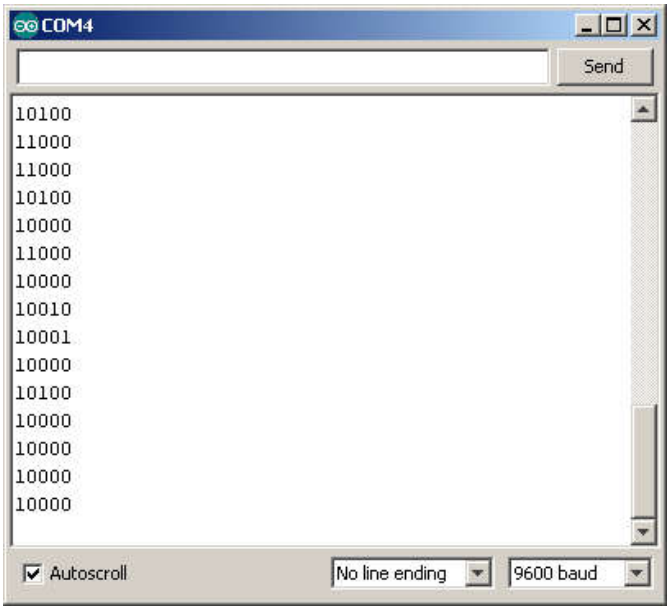


Now on the physical setup is ready. In the following, open the Arduino IDE and create a new program (*File->New*), which will contain the following code:

```
// Read status of buttons and send it to the PC over the serial connection

// Variables for reading the status of the buttons connected to the digital
pins 8, 9, 10, 11
int b1;
int b2;
int b3;
int b4;
```

```
// variable for the transmitted status (as a decimal number)
int stat = 0;

void setup() {
  // configure digital pins as inputs
  pinMode(8, INPUT);
  pinMode(9, INPUT);
  pinMode(10, INPUT);
  pinMode(11, INPUT);
  // activate serial communication for displaying the result on the PC
  Serial.begin(9600);
}

void loop() {
 // read BTNs status
 b1 = digitalRead(8);
 b2 = digitalRead(9);
 b3 = digitalRead(10);
 b4 = digitalRead(11);
 // combine the bits in a decimal number (stat)
 stat = 10000 + b4 * 1000 + b3 * 100 + b2 * 10 + b1;
 //transmit status
 Serial.println(stat);
 // delay 0.5 sec
 delay(500);
}
```

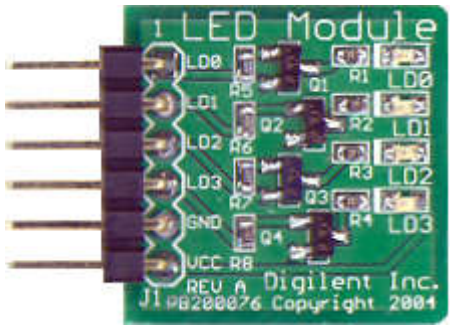
Connect the Arduino board to the PC and "Upload" the program. To visualize the result, open the "Serial Monitor" utility from the *Tools->Serial Monitor* menu. At every 0.5 sec the value of the stat variable will be displayed as a 5 digit number (1XXXX), the list significant 4 digits being the status of the buttons (actually the decimal number is displayed as a as human-readable ASCII text). For more options about the print formats see: https://www.arduino.cc/en/Serial/Print.



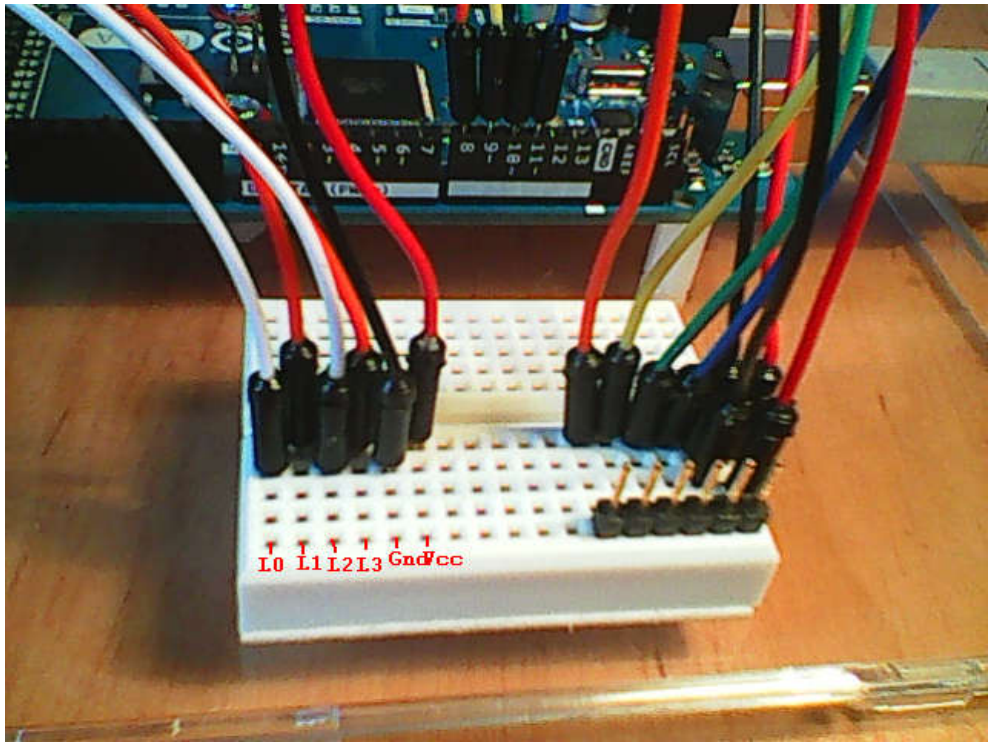
Attention: The Serial Monitor utility should be closed before disconnecting the Arduino board form the PC. Otherwise, it is possible the hang the virtual serial port in a blocked status and further communication with the board will be anymore possible only by restarting the PC.

3-rd example: the use of the LED Pmod as an output device.

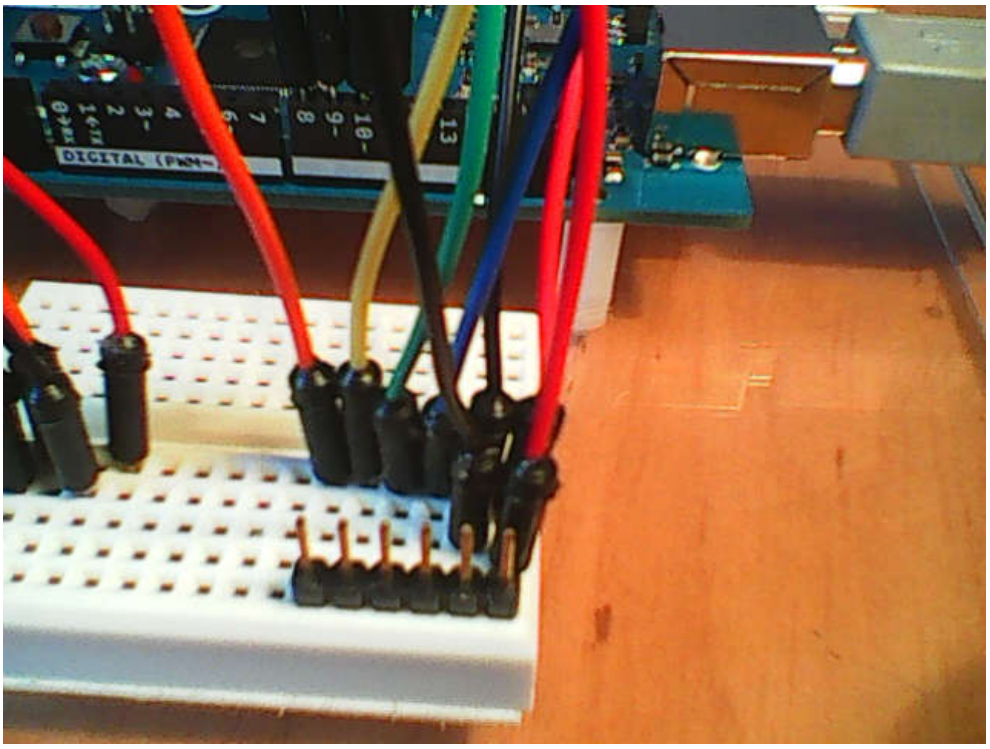
The LED PMod has 6 pins: 2 for powering (VCC and GND) and 4 data pins for the LEDs (logic value 1 = LED is lit)



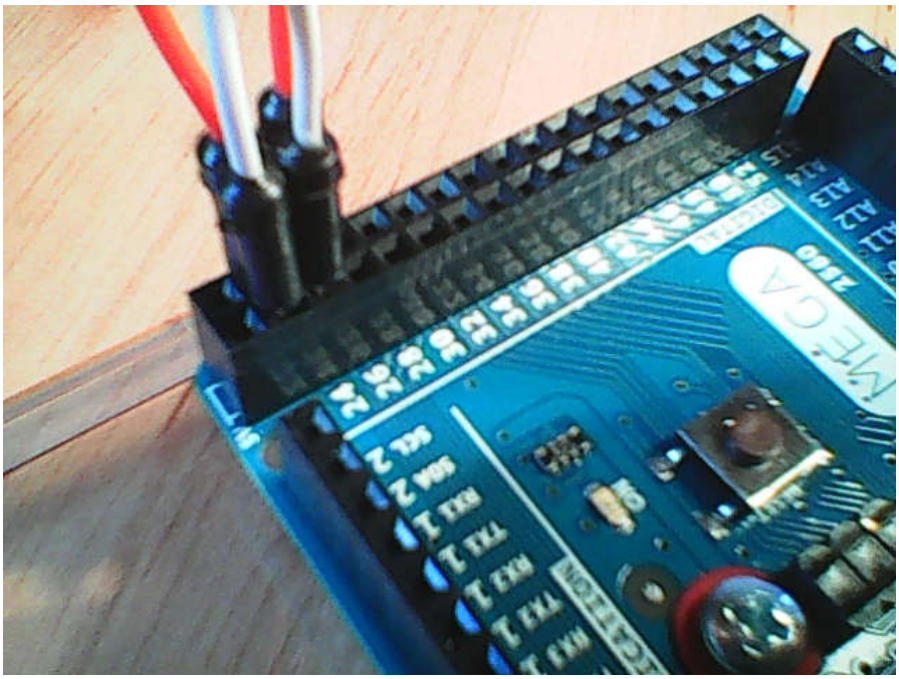
First step consist in the preparing 6 positions on the breadboard for the 6 pins of the LED PMod (L3…L0, GND, VCC). 6 wires will be used for the connection (as in the figure bellow). Zig-zag connection (if enough space is available) is preferred in order to avoid stress in the connectors/wires).



The powering wires (VCC-red and GND-black) will be connected to the corresponding +5V and GND semi-columns (which are already connected to the board powering pins – see the figure below).



Data wires will be connected to the digital pins 22, 23, 24, 25 of the Arduino board (which correspond to the bits of port A of the ATMega 2560 MCU: PA3, PA2, PA1, PA0) as in the figure bellow:



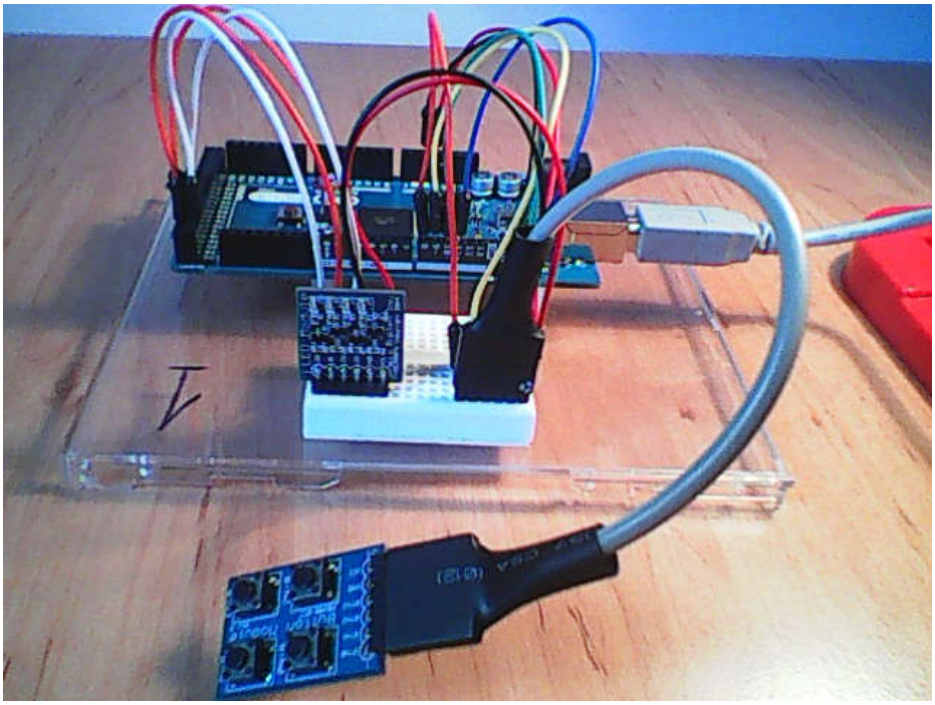
The correspondence between the Arduino board pins and the ATMega 2560 MCU pins (ports) is described in the "ATmega2560-Arduino Pin Mapping" document http://arduino.cc/en/Hacking/PinMapping2560 or in the first figure or in the printed schematic).

For port A, we have:

PA7 (AD7)	Digital pin 29
PA6 (AD6)	Digital pin 28

PA5 (AD5)	Digital pin 27
PA4 (AD4)	Digital pin 26
PA3 (AD3)	Digital pin 25
PA2 (AD2)	Digital pin 24
PA1 (AD1)	Digital pin 23
PA0 (AD0)	Digital pin 22

The LED PMod bloc will be inserted directly into the breadboard as in the figure bellow (**double-check the powering polarity**). After that, connect the PMod BTN module using the 6-pin cable (as explained in the 2-nd example). The whole setup will look like in the figure bellow:



Create a new Arduino project and add the following code:

```
// Read status of buttons and display it on LEDs connected to PORTA

// Variables for reading the status of the buttons connected to the digital
pins 8, 9, 10, 11
int b1;
int b2;
int b3;
int b4;

// variable for the LED status
unsigned char stat = 0;

void setup() {
  // configure digital pins as inputs
  pinMode(8, INPUT);
  pinMode(9, INPUT);
  pinMode(10, INPUT);
  pinMode(11, INPUT);
    // activate PORTA, bits 3..0, as output,
```

```
    DDRA = 0b00001111;
}

void loop() {
 // read BTNs status
 b1 = digitalRead(8);
 b2 = digitalRead(9);
 b3 = digitalRead(10);
 b4 = digitalRead(11);
// combine result: each button has its own bit
stat = (b4<<3) | (b3<<2) | (b2<<1) | b1;
 // Display status on the LEDs connected to port A
 PORTA = stat;
 // delay 50 ms
 delay(50);
}
```

Individual work:
1. Run examples 1 and 2.
2. Modify example 2, in order to transmit to the PC various information according to the pressed button. Ex: one button press will transmit the number of ms from the start of the program (call milis() function), another button press you can transmit the number of sec (millis()/1000), another button wil transmit a text etc.
3. Run example3.
4. Modify example 3 in order to display some animations on the LEDs (1 walking LED lit-on from right-left or left-right). The walking direction depends on the status of one button.

Additional references:
http://users.utcluj.ro/~tmarita/PMP/PMPcurs.htm
"ATmega2560-Arduino Pin Mapping" document http://arduino.cc/en/Hacking/PinMapping2560