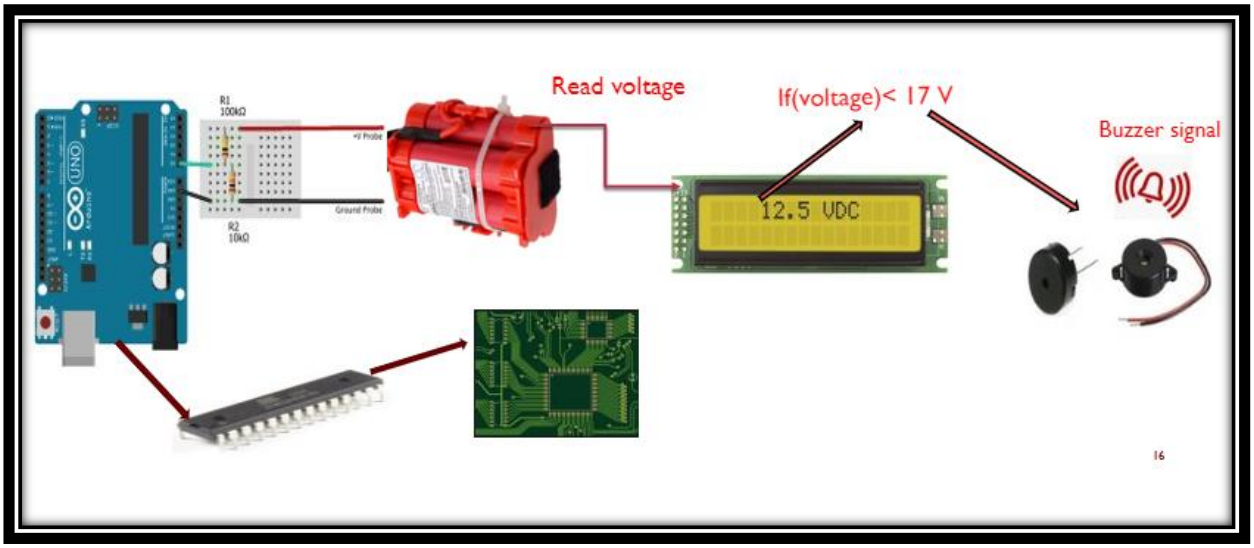
Voltage measurement battery

Introduction:

The utility of this work is that we need to know when the battery is charged to less than 17 V , when it is less than 17 V a buzzer will work as an alarm with blinking of one led he display of value of voltage on a LCD.



STEPS:

Building an Arduino DC Voltmeter

In this tutorial you will see how

to read voltages from the Arduino's

analog inputs and build a voltmeter

that measures direct current (DC) voltages.

The circuit and information presented below assume a basic **knowledge of digital electronics and circuits**, and how to use an Arduino. **Learning the Arduino is a step-by-step** process, covering both programming in C and working with electronics circuits.

To complete the tutorial, you will need:

- An Arduino or Arduino-compatible board with analog inputs.
- The Arduino IDE (integrated development environment).
- One 100Ko resistor.
- One 10Ko resistor.
- Four wires, in at least two different colors (red and black are recommended).
- A breadboard (or suitable stripboard and soldering equipment).

In preparation, you should solder crocodile clips to two differently-colored wires, as this will make it easier to attach them to components when measuring voltages.

Working with electricity, even at low voltages, can be dangerous. There a risk of damage to equipment and yourself – follow the connection diagrams and instructions carefully, and always seek advice from a qualified and experienced adult if you are unsure.

The Arduino Sketch

To eliminate the possibility that the Arduino will run a previous sketch and operate in an unknown way, you can program the sketch first.

To create the voltmeter sketch:

1. Open the Arduino IDE.
2. Paste in the following code:

```
/*
 * Building an Arduino DC Voltmeter
*/

float vPow = 4.7;
float r1 = 100000;
float r2 = 10000;

void setup() {
  Serial.begin(9600);

  // Send ANSI terminal codes
  Serial.print("\x1B");
  Serial.print("[2J");
  Serial.print("\x1B");
  Serial.println("[H");
  // End ANSI terminal codes

  Serial.println("--------------------");
  Serial.println("DC VOLTMETER");
  Serial.print("Maximum Voltage: ");
  Serial.print((int)(vPow / (r2 / (r1 + r2))));
  Serial.println("V");
  Serial.println("--------------------");
  Serial.println("");

  delay(2000);
}
```

```
29
30  void loop() {
31    float v = (analogRead(0) * vPow) / 1024.0;
32    float v2 = v / (r2 / (r1 + r2));
33
34    // Send ANSI terminal codes
35    Serial.print("\x1B");
36    Serial.print("[1A");
37    // End ANSI terminal codes
38
39    Serial.println(v2);
40  }
41
```

And then save the sketch:

3. On the **File** menu, click **Save As...**

This sketch begins by initializing the serial port and declaring a few variables:

vPow – When powered over a USB cable, it is common for the Arduino's 5V power supply to be a little less than that ideal.

r1 – the value (in ohms) of the first resistor in the circuit.

r2 – the value (in ohms) of the second resistor in the circuit.

The sketch then sends some basic information to the terminal, and it displays the maximum voltage than can safely be measured by the current circuit.

The Serial Port Monitor in the IDE can be used to view the messages sent by this sketch. However, this is not a particularly advanced monitor, and cannot display the ANSI terminal sequences that are used to provide a friendlier display. Better results can be had by using a terminal package such as Hyperterminal, RealTerm or Putty.

The serial port you need to connect to can be found from the Arduino IDE:

- On the **Tools** menu, click **Serial Port** and look for the item that is ticked.

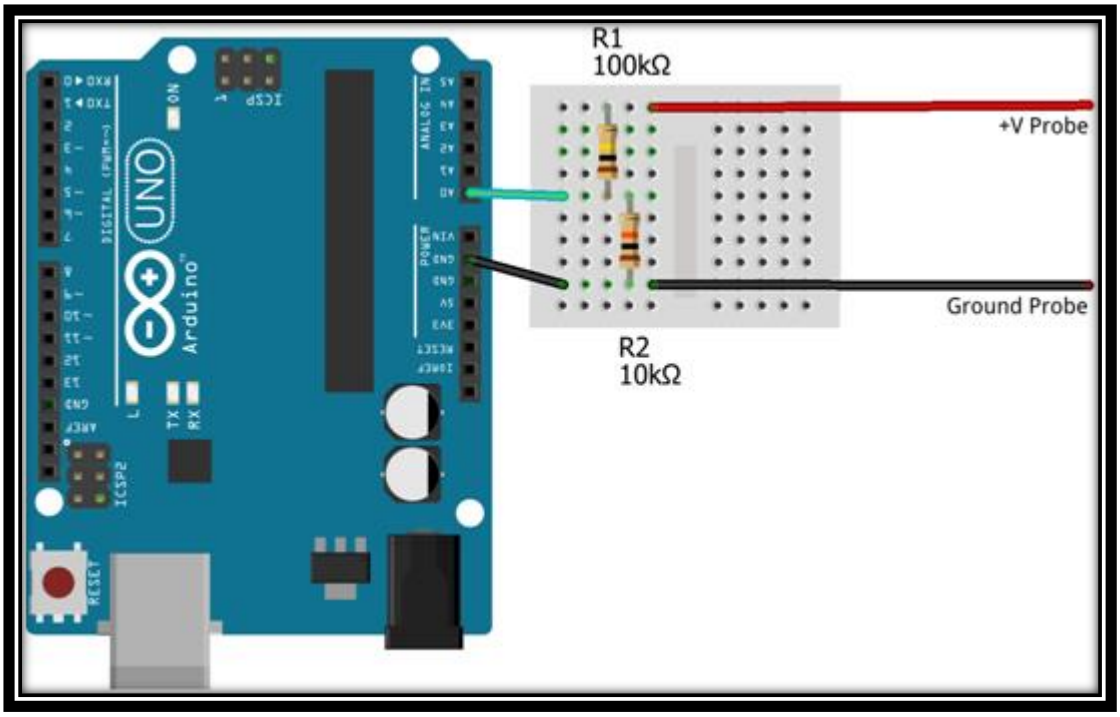
The other setting you should use are:

Display: ANSI
Speed: 9600
Parity: None
Data Bits: 8
Stop Bits: 1
Hardware Flow Control: None
Software Flow Control: None

Building the Circuit

Disconnect the Arduino from your computer before building this circuit!

The circuit can be constructed on a breadboard:



The analog inputs of an Arduino can measure up to 5V (when using the built-in analog reference voltage). Even when only connecting to a 5V circuit, you should use the resistors to help protect the Arduino from short-circuits or unexpected voltage surges.

Those two resistors form a potential divider that is used to lower the voltage being measured to a level that the Arduino can read. This actually extends the range that can be used. For example, if resistors are used to halve the input voltage then the Arduino can effectively read up to 10V (since 10V will be read as 5V, 5V will be read as 2.5V…). This comes at the expensive of accuracy – the ADCs in the Arduino can read up to 1024 different levels between 0V and 5V. By expanding the range to 10V, those 1024 levels are spread across a wider range and are therefore less able to detect small changes.

You can increase the resistance value of R2, then the maximum voltage that can be read will be decreased; giving a slightly more accurate reading. With R1 at 100Ko and R2 at 10Ko, the input voltage is reduced by a factor of around 11 – allowing the voltmeter to read from 0V–55V.

The formula for calculating values in a potential divider is:

```
Vout = (R2 / (R1 + R2)) * Vin
```

If the divider for the Arduino voltmeter is functioning correctly then Vout will be a maximum of 5V, and so you can calculate the maximum input voltage to the circuit:

```
Vmax = 5.0 / (R2 / (R1 + R2))
```

You can see a variation of this expression used in the *setup()* routine of the sketch.

Note: If you use different resistors from the ones suggested here, you must be remember to adjust the values of *r1* and *r2* in the sketch.

When measuring the voltage in the *loop()* routine, *analogRead(0)* is used to read the level from analog input 0. The returned value is an integer in the range 0 through 1023, so it must first be adjusted to a range 0 through 5. This is done by multiplying it by the power supply level, and then dividing by 1024.

To transform the 0V–5V value into a reading that reflects the range of values that can be measured by the circuit, the resistors must be taken into account in the same way as was done to calculate the maximum voltage the circuit could measure:

```
v2 = v / (r2 / (r1 + r2))
```

With all the calculations completed, the value now represents the actual voltage measured by the circuit, and is sent to the display.

Enhancing the Voltmeter

The voltmeter presented here is extremely basic and we will see now considerable room for enhancements such as adding an LCD display, two leds and a buzzer that plays the role of an alarm.

Adding an LCD display

To make my voltmeter with my Arduino Uno, I used the LCD display.

To complete the tutorial, you will need:

- 1x LCD (Liquid Crystal Display)
- 1x 10 k ohm potentiometer

Connection of the following pins:

- LCD RS pin to digital pin 12

-LCD Enable pin to digital pin 11

-LCD D4 pin to digital pin 5

-LCD D5 pin to digital pin 4

-LCD D6 pin to digital pin 3

-LCD D7 pin to digital pin 2

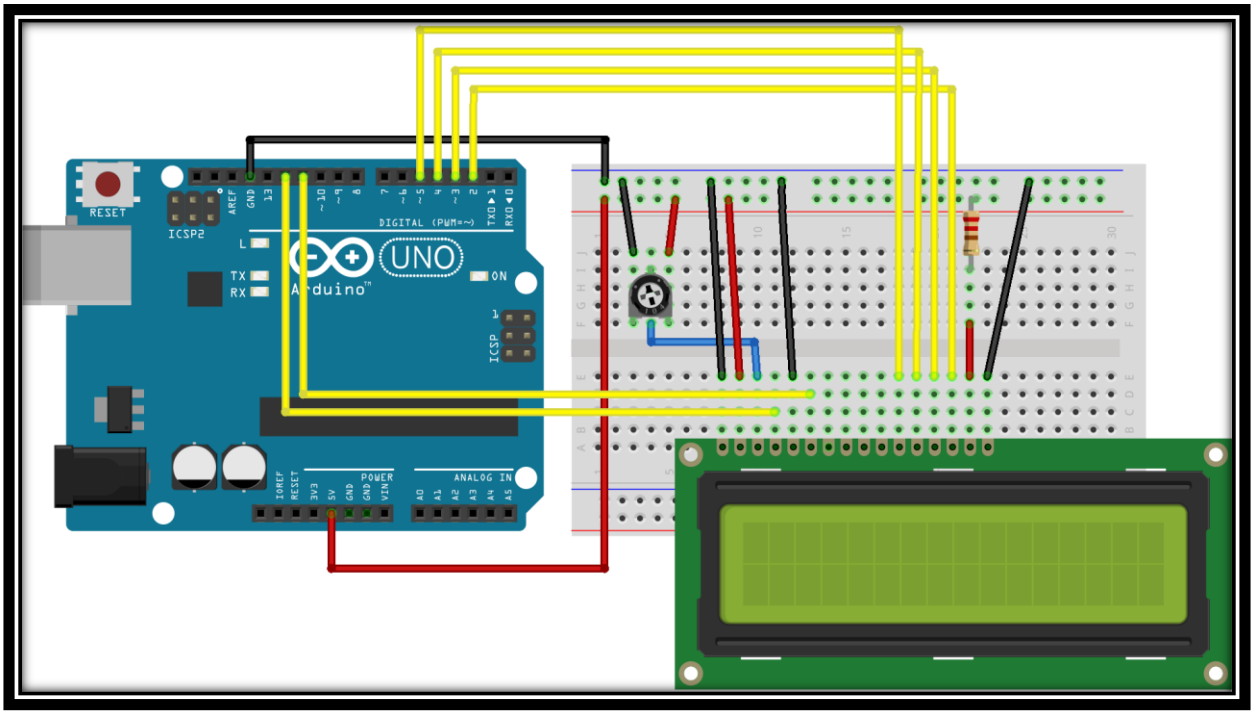
-LCD R/W pin to ground

- LCD VSS pin to ground

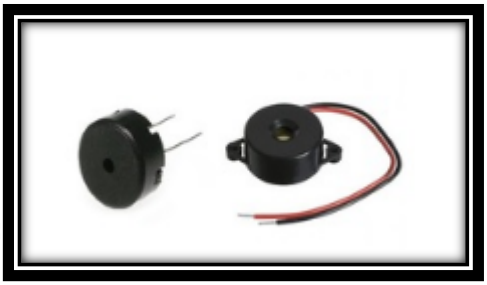
-LCDD VCDD pin to 5V

- Additionally, wire a 10k pot to +5V and GND, with it's wiper (output) to LCD screens VO pin (pin3). A 220 ohm resistor is used to power the backlight of the display, usually on pin 15 and 16 of the LCD connector.

  The circuit can be constructed on a breadboard:



Adding a buzzer



I added a buzzer which work as alarm

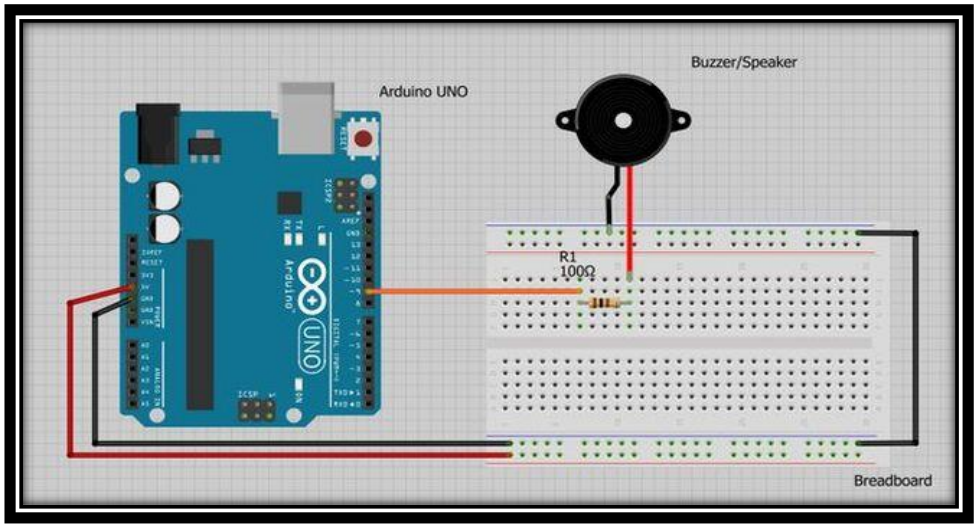
When the battery is charged to less than 17 V

What you will need To complete the tutorial **– Hardware :**

For this part you will need to add:

- Buzzer/piezo speaker
- 100 Ohm resistor

The circuit :



The connections are pretty easy, see the image above with breadboard circuit schematic.

The code

How it works? It's simple, tone(buzzer, 1000) sends a 1KHz sound signal to pin 9, delay(100) pause the program for one second and noTone(buzzer) stops the signal sound. The loop() routine will make this run again and again making a short beeping sound.

The final Arduino Sketch

project

```
float vPow = 4.7;
float r1 = 100000;
float r2 = 10000;
int ledOutput = 7;
const int buzzer = 9; //buzzer to arduino pin 9

#include <LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup() {
   pinMode(buzzer, OUTPUT);
 pinMode(ledOutput, OUTPUT);
   lcd.begin(16, 2);

   Serial.begin(9600);

   Serial.println("--------------------");
   Serial.println("DC VOLTMETER");
   Serial.print("Maximum Voltage: ");
   Serial.print((int)(vPow / (r2 / (r1 + r2))));
   Serial.println("V");
   Serial.println("--------------------");
   Serial.println("");

   delay(2000); }

void loop() {
    float v = (analogRead(A0) * vPow) / 1024.0;
    float v2 = v / (r2 / (r1 + r2));
      digitalWrite(ledOutput, LOW);


       if (v2 <= 17){
                lcd.clear();
          Serial.println(" Voltage=");
      Serial.println(v2);
  Serial.println("V");
    lcd.print("Voltage= ");
      lcd.print(v2);
        lcd.print(" V");
         tone(buzzer, 1000);
         delay(1000);
         noTone(buzzer);
         delay(100);
    digitalWrite(ledOutput, HIGH);
     delay(100);

        digitalWrite(ledOutput, LOW);
     delay(100);

  }
```

```
  else {
    lcd.clear();
     Serial.println("Voltage=");
      Serial.println(v2);
  Serial.println("V");
    lcd.print("Voltage= ");
      lcd.print(v2);
      lcd.print("V");
  delay(500);

    noTone(buzzer);     // Stop sound...
    digitalWrite(ledOutput, LOW);
       delay(100);
     }
}
```

Link to the code on Ecam Eurobot Github :

PCB

Building an Arduino on a Breadboard

This part explains how to migrate from an Arduino board to a standalone microcontroller on a breadboard. It's similar to this tutorial, but uses an Arduino board to program the ATmega on the breadboard.

Unless you choose to use the minimal configuration described at the end of this tutorial, you'll need four components (besides the Arduino, ATmega328PU, and breadboard):

- a 16 MHz crystal,

-a 10k resistor, and

-two 18 to 22 picofarad (ceramic) capacitors.

Burning the Bootloader

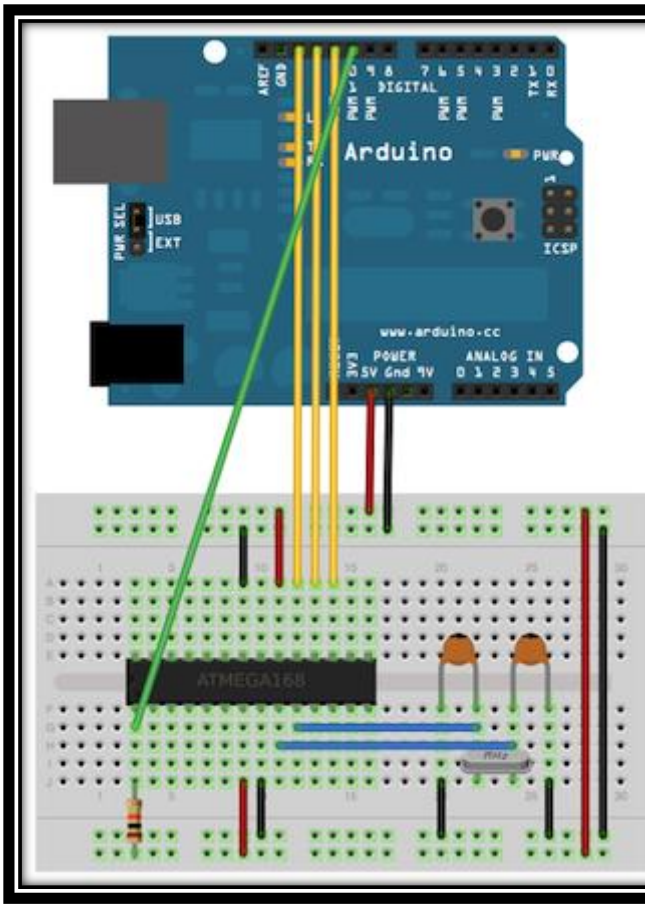
If you have a new ATmega328 (or ATmega168), you'll need to burn the bootloader onto it. You can do this using an Arduino board as an in-system program (ISP). If the microcontroller already has the bootloader on it (e.g. because you took it out of an Arduino board or ordered an already-bootloaded ATmega), you can skip this section.

To burn the bootloader, follow these steps:

Upload the ArduinoISP sketch onto your Arduino board. (You'll need to select the board and serial port from the Tools menu that correspond to your board.)
Wire up the Arduino board and microcontroller as shown in the diagram to the right.
Select "Arduino Duemilanove or Nano w/ ATmega328" from the Tools > Board menu. (Or "ATmega328 on a breadboard (8 MHz internal clock)" if using the minimal configuration described below.)
Select "Arduino as ISP" from Tools > Programmer
Run Tools > Burn Bootloader

You should only need to burn the bootloader once. After you've done so, you can remove the jumper wires connected to pins 10, 11, 12, and 13 of the Arduino board.

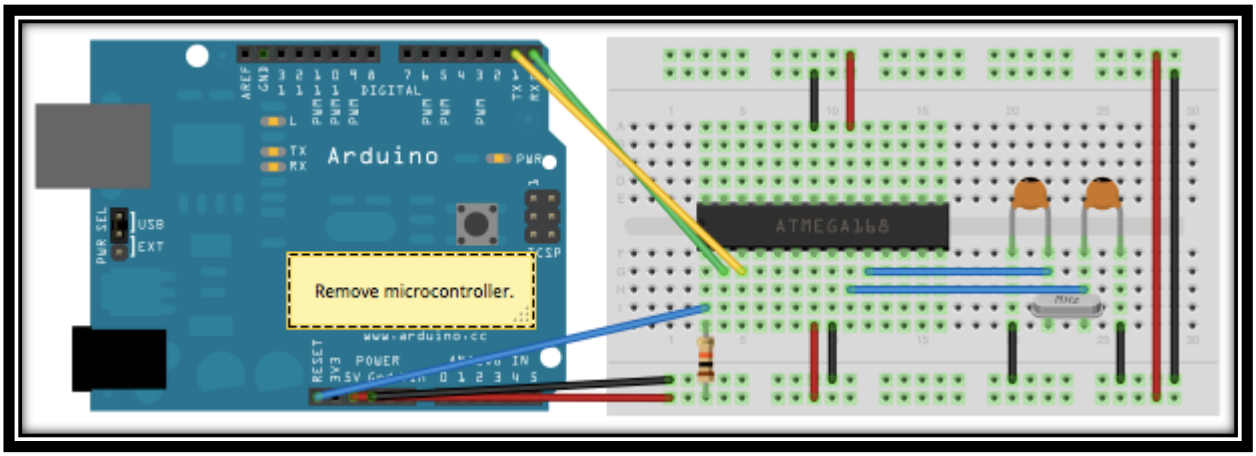
Be careful when you wire the cables between arduino and l'Atmega.



Using an Arduino board to burn the bootloader onto an ATmega on a breadboard.

Uploading Using an Arduino Board

Once your ATmega328p has the Arduino bootloader on it, you can upload programs to it using the USB-to-serial convertor (FTDI chip) on an Arduino board. To do, you remove the microcontroller from the Arduino board so the FTDI chip can talk to the microcontroller on the breadboard instead. The diagram at right shows how to connect the RX and TX lines from the Arduino board to the ATmega on the breadboard. To program the microcontroller, select "Arduino Duemilanove or Nano w/ ATmega328" from the the Tools > Board menu (or "ATmega328 on a breadboard (8 MHzinternal clock)" if you're using the minimal configuration described below). Then upload as usual.

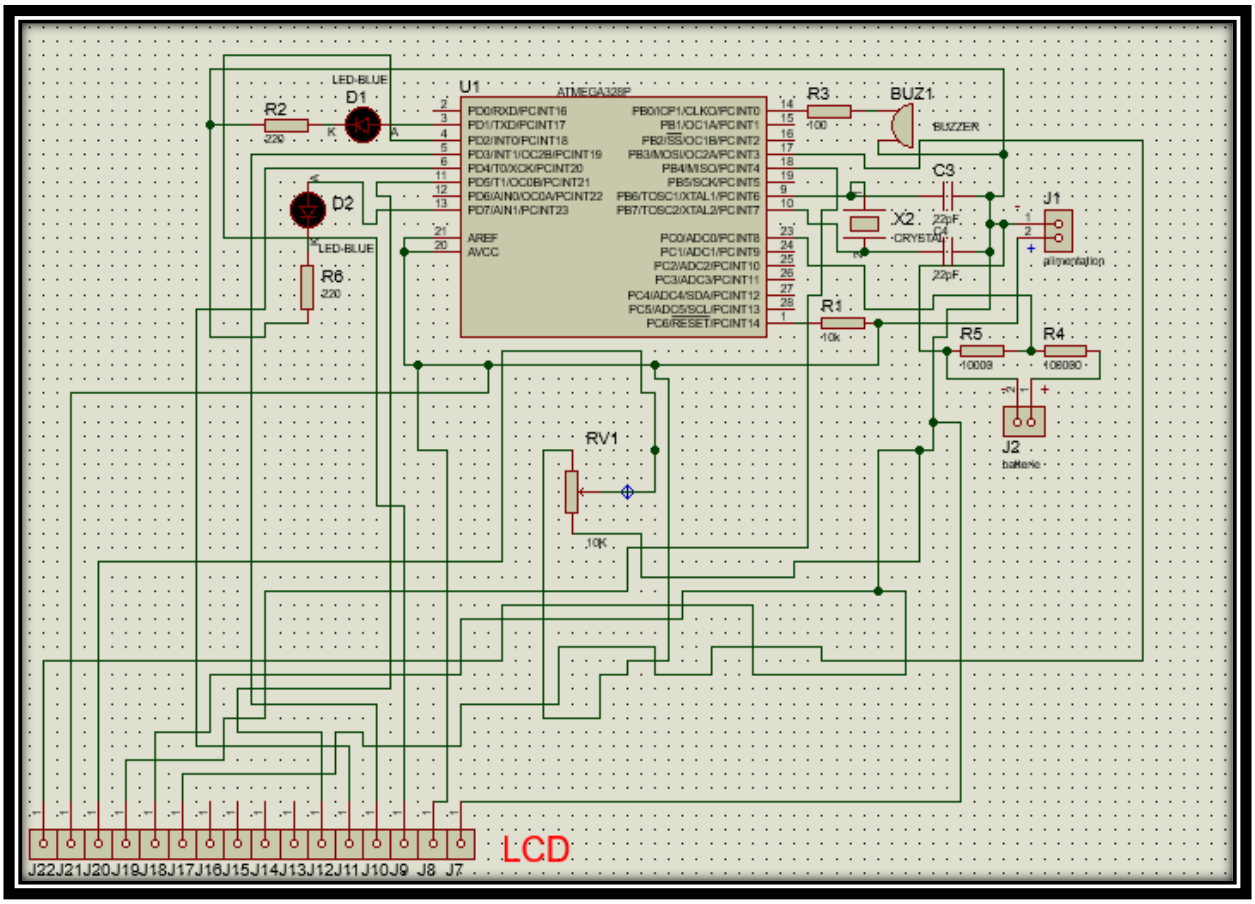


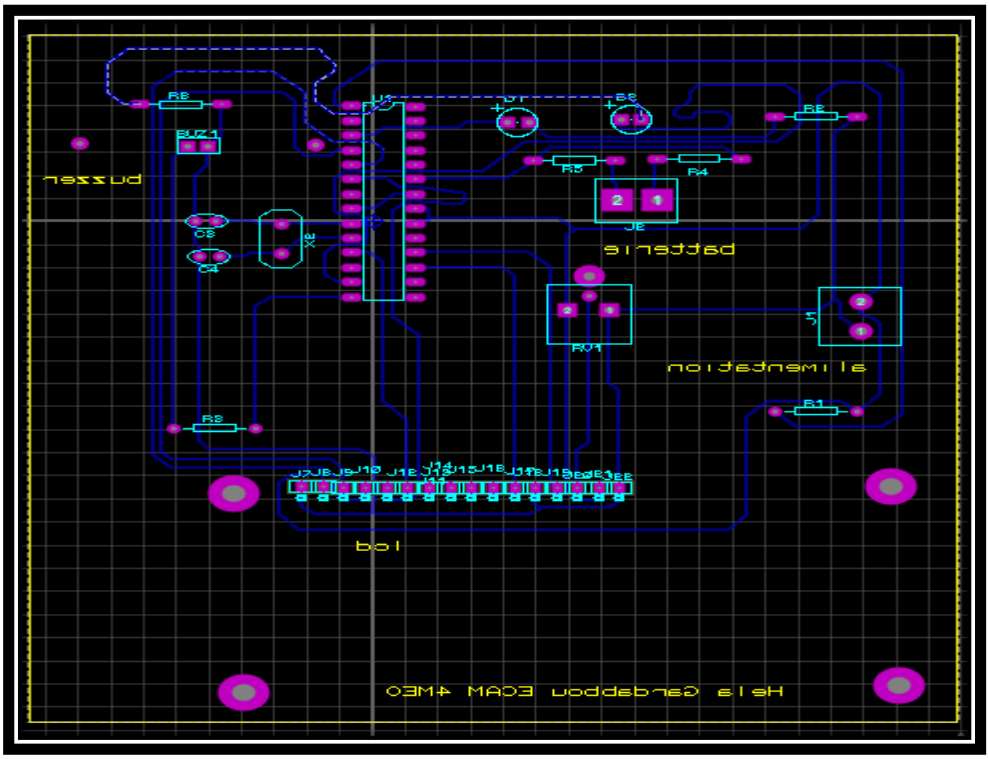
Uploading sketches to an ATmega on a breadboard. Remember to remove the microcontroller from the Arduino board!

Getting Rid of the Arduino Board

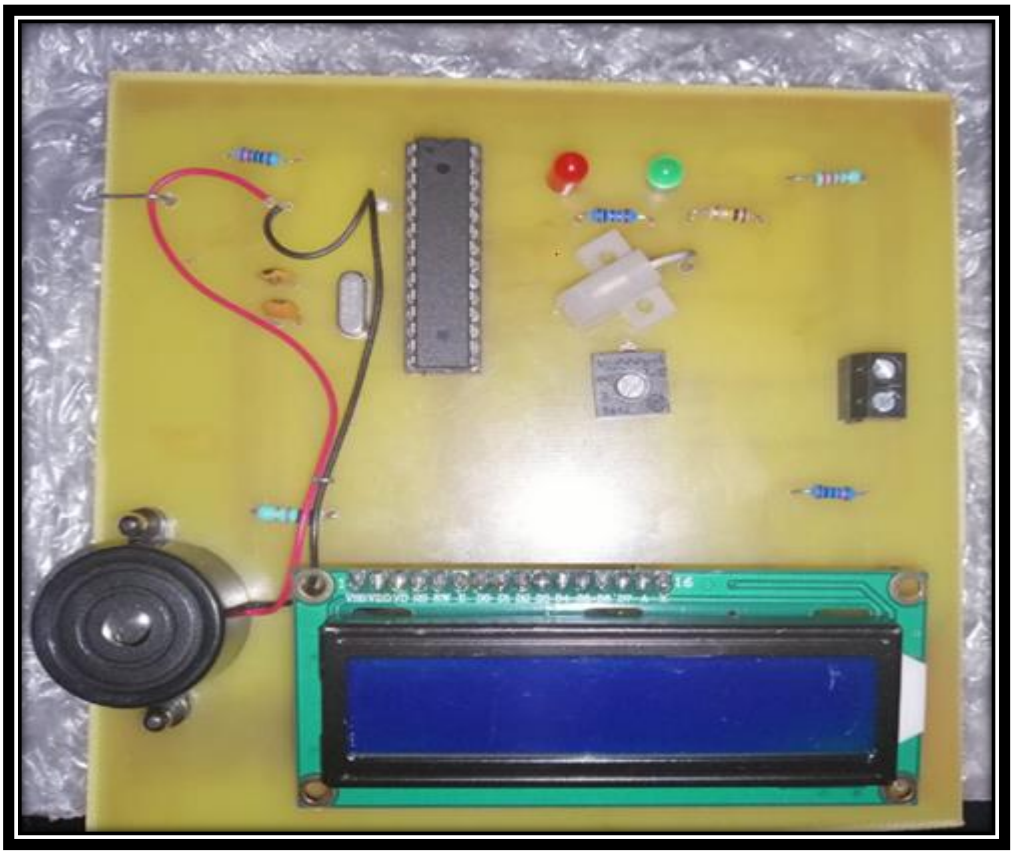
Once you've programmed the ATmega on the breadboard, you can eliminate the Arduino. To do so, you'll need to provide an alternative power supply for the microcontroller.

Schematic with proteus 8 professional





The final printed circuit:



Issue :

when the card was printed there are missed tracks that's why I did the circuit on the breadboard.

References

https://www.jameco.com/jameco/workshop/jamecobuilds/arduinocircuit.html

https://create.arduino.cc/projecthub/next-tech-lab/voltmeter-using-arduino-00e7d1