Arduino Programming

Arduino UNO & Innoesys Educational Shield

www.devobox.com
Electronic Components &
Prototyping Tools
79 Leandrou, 10443, Athens
+30 210 51 55 513, info@devobox.com

Arduino UNO

Arduino is a small device (microprocessor) that connects with a PC by USB cable. With current device professors, hobbyists and students learn programming and can implement automation systems & applications (ex. Robotics).

The Arduino PCB has an open source microcontroller on which someone could program, even if he is novice, in Windows, Linux and MAC OS X operation systems. The Arduino has I/O's (inputs, outputs) on which several components and peripherals could connect (Led, potentiometers, relay, sensors and many more) to provide us data (like temperature sensor) or for us to set data (e.g. LED On).

The connection of the peripherals is done with a certain way for every electronic component, so the user should have knowledge of the theory in electronics. In current guide thou we will not refer to these kind of electronic connections and circuits. That means that even the novice user could easily take his first steps in the Arduino programming. To be able to do this we will use the Educational Shield of Innoesys.

Innoesys Educational Shield

The Educational Shield of Innoesys combines many peripheral components on its PCB. There are solder and ready to use by the novice user, who is not necessary to have knowledge in electronics theory. So no connections with cables, resistors and breadboards are needed.

Educational Shield Features:

- LED (R+G+B): Three LEDs in different colors connected at the digital outputs D6 (Red), D9 (Green) and D10 (Blue).
- Buttons: Five buttons connected at the digital inputs D7 (S1), D8 (S2), D11 (S3), D12 (S4) και D13 (S5)
- On-Off Switch: A two position switch (On-Off) connected at the digital input D4
- Relay: A relay connected at the digital output D2
- Potentiometer: A potentiometer connected at the analog input A1
- Buzzer: A buzzer for sound notification connected at the digital output D5
- Photoresistor: A photoresistor in analog input A2
- Temperature Sensor: A temperature sensor connected at the analog input A0
- Servo Motor Connector: A three pin connector for connecting servo motor. The pin is connected at the digital output D3

Details and further information about the educational shield can be found in the manufacturer datasheet (www.innoesys.com)

Before we start

As we have seen previously the Arduino UNO has inputs and outputs. The status – input or output – should be configured depending on the peripheral. A temperature sensor for example provide us with data so it has to be configured as input. On the other hand a LED has to be configured as an output for us to be able to set its status (on-off). We will see how this configuration is set in the sections below.

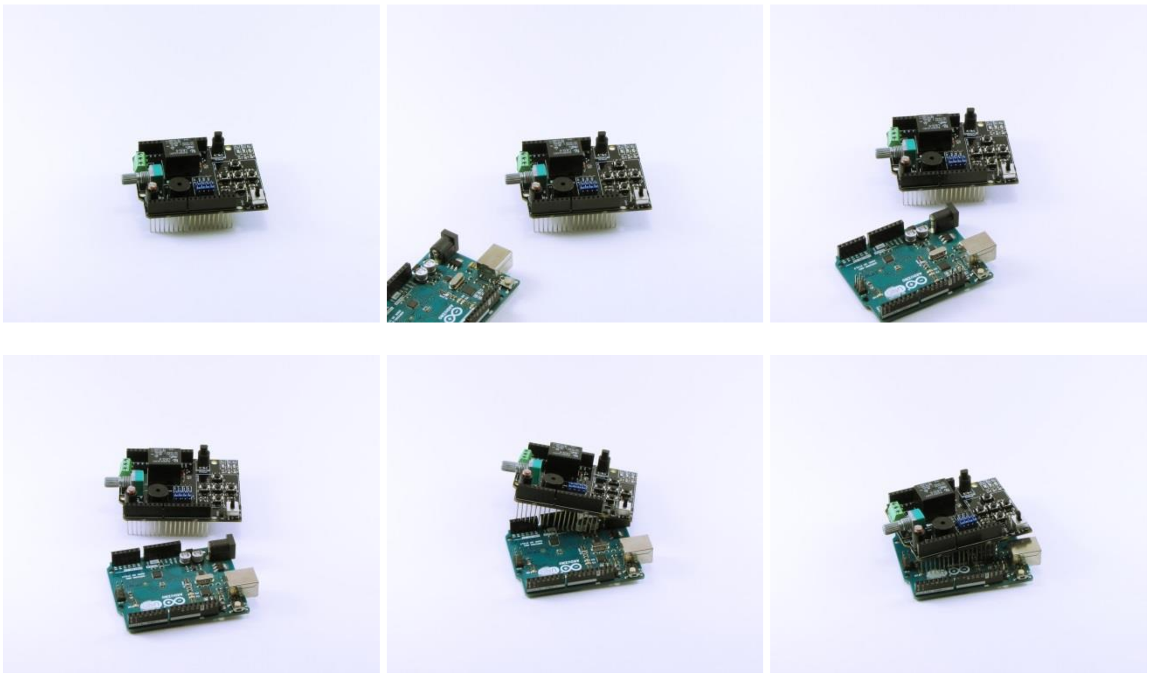
The Arduino is manufactured this way for the user to be able to add shield on top of it. This way there is no need of cabling or other connections. So in order to complete the following examples we only need the following:

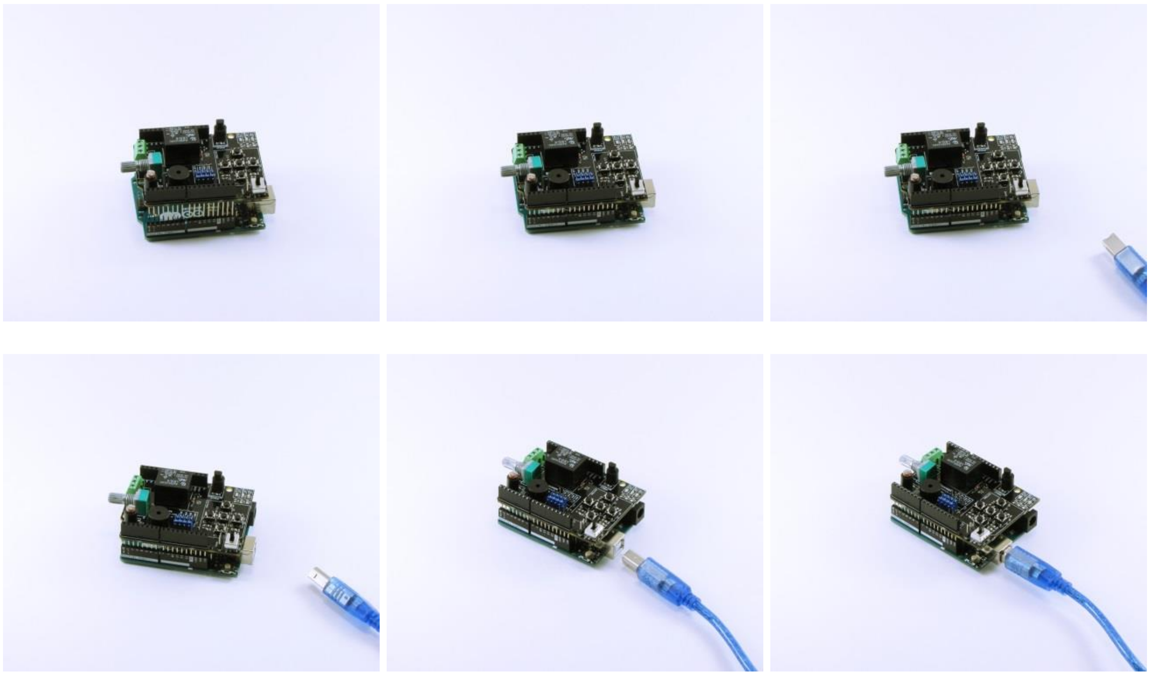
- 1 x Arduino UNO
- 1 x USB cable
- 1 x Innoesys Educational Shield

For programming the Arduino UNO board we only need the Arduino IDE interface. This IDE software should be downloaded from the official manufacturer website (https://www.arduino.cc/en/Main/Software) and must be installed on the PC.

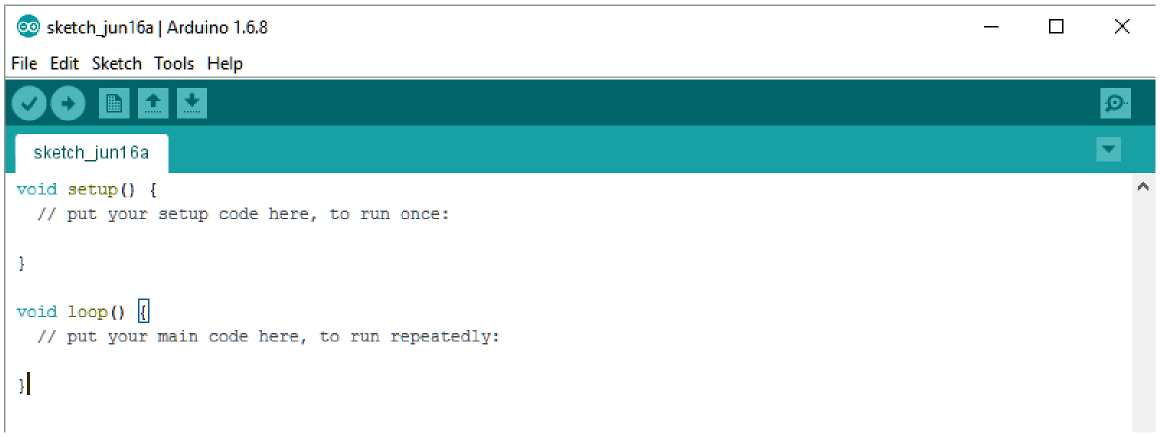
Connection

The connection of the Arduino UNO with the educational shield and the PC is very simple. We place the Educational Shield at the top of the Arduino board and we press it for the pins to contact well with headers. Then we connect the Arduino board to the PC by the USB cable. See the picture below for details.



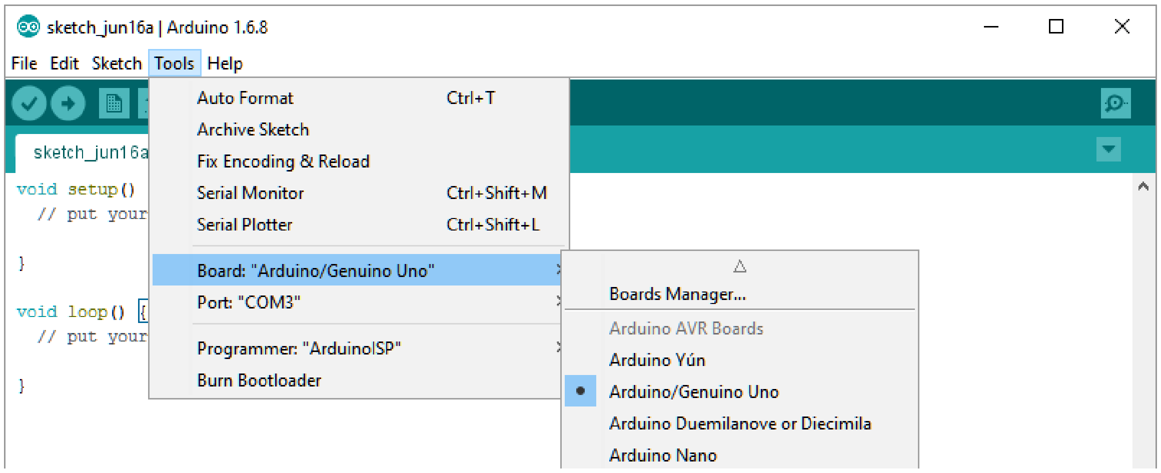


Finally we start the Arduino IDE environment, which we have previously installed in our PC. In the window that opens we see to sections. In the first section *void setup()* we place the configuration. This section of code will "run" only once when the Arduino board powers on, In the second section *void loop()* we place the code lines that will be run repeatedly.



```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

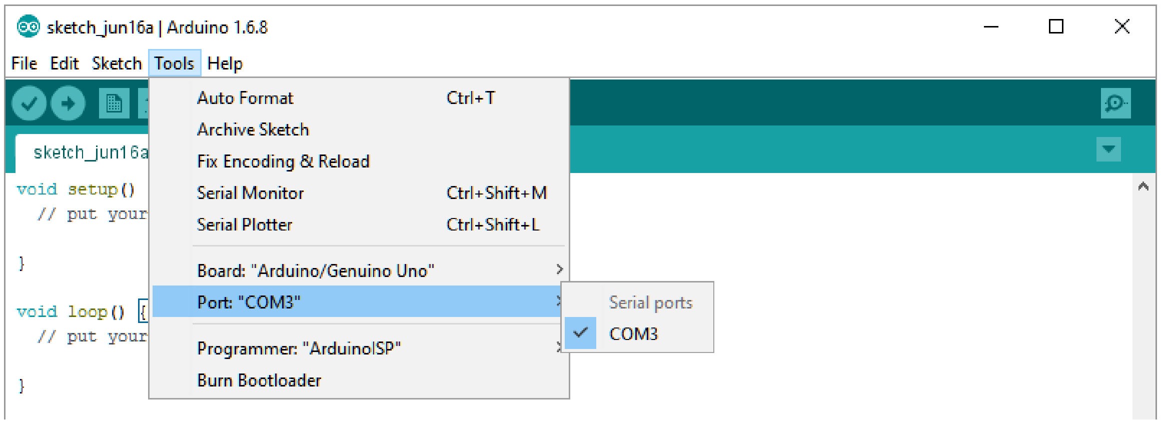
In order to confirm the communication with the Arduino board, from the menu → tools → Board we select the Arduino UNO board.


sketch_jun16a | Arduino 1.6.8

File Edit Sketch Tools Help

```
void setup() {
  // put your

}

void loop() {
  // put your

}
```

Finally in the menu → tools → Port we choose the communication port where the Arduino board is connected.


sketch_jun16a | Arduino 1.6.8

File Edit Sketch Tools Help

```
void setup() {
  // put your

}

void loop() {
  // put your

}
```

We are now ready to proceed to the examples.

1. Power on the LEDs

The Educational Shield, which is connected to the Arduino, has the LED connected at the digital outputs D6 (Red), D9 (Green) and D10 (Blue).

Firstly we set the digital outputs. This is done with the command *pinMode()* that we set it in the setup section *void setup()*. So we have to write:

```
pinMode(6, OUTPUT);
pinMode(9, OUTPUT);
pinMode(10, OUTPUT);
```

In the second step we have to power on-off the LED. This is possible with the command *digitalWrite()* and we place it in the loop section *void loop()*, in order to run repeatableı. So in order for the led to power on-off we write:

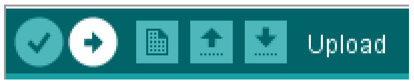
```
digitalWrite(6, HIGH);
digitalWrite(6, LOW);
```

We should add a delay between the on and off in order to be visible. For this we use the command *delay()*

Here is the full code:

```
void setup() {
  // put your setup code here, to run once:

  // initialize digital pins
  pinMode(6, OUTPUT); //Red
  pinMode(9, OUTPUT); //Green
  pinMode(10, OUTPUT); //Blue
}

void loop() {
  // put your main code here, to run repeatedly:

  digitalWrite(6, HIGH);
  delay(1000); //Delay for 1 sec
  digitalWrite(6, LOW);
  delay(1000); //Delay for 1 sec
}
```

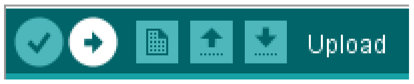
We choose Upload(→) for the Arduino to be updated with the changes.



We copy the code of one LED and paste it with the appropriate changes. So the code changes to:

```
void setup() {
  // put your setup code here, to run once:

  // initialize led pins
  pinMode(6, OUTPUT); //Red
  pinMode(9, OUTPUT); //Green
  pinMode(10, OUTPUT); //Blue
}

void loop() {
  // put your main code here, to run repeatedly:

  digitalWrite(6, HIGH);
  delay(1000); //Delay for 1 sec
  digitalWrite(6, LOW);
  delay(1000); //Delay for 1 sec

  digitalWrite(9, HIGH);
  delay(1000); //Delay for 1 sec
  digitalWrite(9, LOW);
  delay(1000); //Delay for 1 sec

  digitalWrite(10, HIGH);
  delay(1000); //Delay for 1 sec
  digitalWrite(10, LOW);
  delay(1000); //Delay for 1 sec
}
```

We choose Upload(→) for the Arduino to be updated with the changes.



2. Adjustable/Automatic Brightness

With the help of the potentiometer, connected to the analog input A1 we can control the brightness of a LED.

First we set the LED as output and the potentiometer as input

```
pinMode(6, OUTPUT); //Red
pinMode(A1, INPUT);
```

and then we write, *analogWrite()* command, to the LED the value, that we read with the *analogRead()* command of the porentiometer.

```
analogWrite(6, (analogRead(A1) / 4));
```

**** Notice: The values that we read from the analog input of the potentiometer are between 0 and 1023, but the values that we write to the output for the LED are between 0 and 255. That is why we divide by 4.*

So the source code is:

```
void setup() {
  // put your setup code here, to run once:

  // initialize led pins
  pinMode(6, OUTPUT); //Red
  // initialize potentiometer
  pinMode(A1, INPUT);
}

void loop() {
  // put your main code here, to run repeatedly:

  analogWrite(6, (analogRead(A1) / 4));
}
```

The same we can do with the photoresistor. We can add the blue LED connected to the output 10 and the photoresistor connected to the input A2.

```
pinMode(10, OUTPUT); //Blue
pinMode(A2, INPUT);
```

So by using the same commands as before we have

```
analogWrite(10, (analogRead(A2) / 4));
```

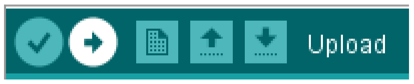
**** Notice: The values that we read from the analog input of the photoresistor are between 0 and 1023, but the values that we write to the output for the LED are between 0 and 255. That is why we divide by 4.*

So the source code would be

```
void setup() {
  // put your setup code here, to run once:

  // initialize led pins
  pinMode(6, OUTPUT); //Red
  pinMode(10, OUTPUT); //Blue

  // initialize potentiometer
  pinMode(A1, INPUT);

  // initialize photoresistor
  pinMode(A2, INPUT);
}
```

```
void loop() {
  // put your main code here, to run repeatedly:

  analogWrite(6, (analogRead(A1) / 4));
  analogWrite(10, (analogRead(A2) / 4));
}
```

We choose Upload(→) for the Arduino to be updated with the changes.



3. Alarm

In this example we will use the sound alarm of the buzzer which is connected to the output D5.

First we set the D5 as output

```
pinMode(5, OUTPUT);
```

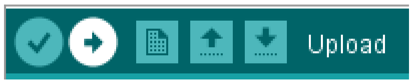
And with the command *tone()* we can set the tone that the buzzer will sound.

```
tone(5, 150, 500);        *Output, Tone(0-255), Tone time
```

With this command the tone will sound for half a second. In order to create an alarm sound we have to add a delay between every execution of the *tone()* command:

```
void setup() {
  // put your setup code here, to run once:

  // initialize buzzer pin
  pinMode(5, OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:

  tone(5, 150, 500);
  delay(1000); //delay for 1 sec
}
```

We choose Upload(→) for the Arduino to be updated with the changes.



4. Start the Fan

At the digital output D2 of the Educational Shield is connected a relay. We can change the status of the relay simply by writing the appropriate values. To do this we must first set the D2 digital as output

```
pinMode(2, OUTPUT);
```

And with the help of the *digitalWrite()* command we change its status

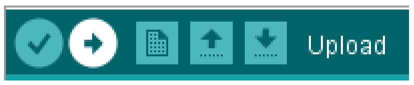
```
digitalWrite(6, HIGH);    ή    digitalWrite(6, LOW);
```

Let's add a delay to the change of every status of the relay and see what happens:

```
void setup() {
  // put your setup code here, to run once:

  // initialize relay pin
  pinMode(2, OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:

  digitalWrite(2, HIGH);
  delay(5000);
  digitalWrite(2, LOW);
  delay(5000);
}
```

**** If we connect a fan in the terminal blocks on the Educational Shield we will be able to see the fan turning ON and Off. Refer to the datasheet of Innoesys Educational Shield for more information about the connection.*

We choose Upload(→) for the Arduino to be updated with the changes.



5. Controlling the Fan with a Switch

A two position switch (On-Off) is connected at digital D4. To use this switch we set it as input.

```
pinMode(4, INPUT);
```

We can read the status of the switch with the *digitalRead()* command and control the fan. We also set the digital D2 as before, for the fan to work.

```
void setup() {
  // put your setup code here, to run once:

  // initialize relay pin
  pinMode(2, OUTPUT);

  // initialize on-off switch
  pinMode(4, INPUT);
}
```

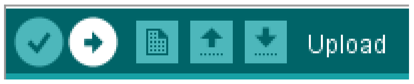
In the previous example we changed the relay status every 5 seconds. Instead of that we will read the ON-OFF Switch status and write it directly to the relay.

```
digitalWrite(2, digitalRead(4));
```

So the source code will be:

```
void setup() {
  // put your setup code here, to run once:

  // initialize relay pin
  pinMode(2, OUTPUT);

  // initialize on-off switch
  pinMode(4, INPUT);
}

void loop() {
  // put your main code here, to run repeatedly:

  digitalWrite(2, digitalRead(4));
}
```

We choose Upload(→) for the Arduino to be updated with the changes.



6. Press and See!

On the Educational Shield there are five pushbuttons connected to the digitals D7 (S1), D8 (S2), D11 (S3), D12 (S4) και D13 (S5). We could set what will these buttons do. So let's set them to light the LEDs, open the Relay and sound the buzzer.
Firstly we set the buttons as digital inputs:

```
pinMode(7, INPUT); //S1
pinMode(8, INPUT); //S2
pinMode(11, INPUT); //S3
pinMode(12, INPUT); //S4
pinMode(13, INPUT); //S5
```

Then we set the LEDs as outputs

```
pinMode(6, OUTPUT); //Red
pinMode(9, OUTPUT); //Green
pinMode(10, OUTPUT); //Blue
```

The relay also as output

```
pinMode(2, OUTPUT);
```

And lastly the buzzer as output

```
pinMode(5, OUTPUT);
```

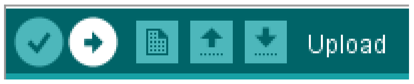
Now that we have configured the I/O's we read the status of the buttons and write value to the outputs

```
digitalWrite(6, digitalRead(7)); //Button S1 -> LED Red
digitalWrite(9, digitalRead(8)); //Button S2 -> LED Green
digitalWrite(10, digitalRead(11)); //Button S3 -> LED Blue
digitalWrite(2, digitalRead(12)); //Button S4 -> Relay
digitalWrite(5, digitalRead(13)); //Button S5 -> Buzzer
```

The source code will be:

```
void setup() {
  // put your setup code here, to run once:

  // initialize the buttons
  pinMode(7, INPUT); //S1
  pinMode(8, INPUT); //S2
  pinMode(11, INPUT); //S3
  pinMode(12, INPUT); //S4
  pinMode(13, INPUT); //S5

  // initialize the LED pins
  pinMode(6, OUTPUT); //Red
  pinMode(9, OUTPUT); //Green
  pinMode(10, OUTPUT); //Blue

  // initialize relay pin
  pinMode(2, OUTPUT);

  // initialize the Buzzer
  pinMode(5, OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:

  digitalWrite(6, digitalRead(7)); //Button S1 -> LED Red
  digitalWrite(9, digitalRead(8)); //Button S2 -> LED Green
  digitalWrite(10, digitalRead(11)); //Button S3 -> LED Blue
  digitalWrite(2, digitalRead(12)); //Button S4 -> Relay
  digitalWrite(5, digitalRead(13)); //Button S5 -> Buzzer
}
```

We choose Upload(→) for the Arduino to be updated with the changes.



7. Is it Hot Today?

In the analog input A0 there is a temperature sensor. With this sensor we are able to read the temperature of the environment.

In order to view the values of the sensor we will use the serial port of the Arduino. So we have to set the serial port and the analog input of the temperature sensor:

```
Serial.begin(9600);
pinMode(A0, INPUT);
```

Now the only thing to be done is to read the values from the sensor with the `analogRead()` command and print them in the serial port with the `Serial.println()` command.

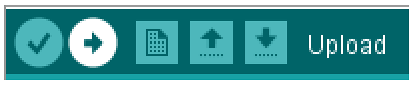
```
Serial.println((analogRead(0) * 0.004882812 * 100) - 273.15);
```

**** The calculations above are necessary for the data to be converted to Celsius.*

So the full source code:

```
void setup() {
  // put your setup code here, to run once:

  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);

  // initialize the temperature Sensor
  pinMode(A0, INPUT);

}

void loop() {
  // put your main code here, to run repeatedly:

  // show the values of the temperature screen in the serial monitor
  Serial.println((analogRead(0) * 0.004882812 * 100) - 273.15);
  delay(1000); //delay for 1 sec
}
```

We choose Upload(→) for the Arduino to be updated with the changes.



To see what we print to the serial port we can use the Serial Monitor from the menu → Tools. In the window that opens we can see the values.