Co-funded by the
Erasmus+ Programme
of the European Union

WirelessUP!

UPraising VET skills for innovation in European electrotechnical sector

Project number: 2017-1-HR01-KA202-035434

WirelessUP! Toolkit

Module 3.1:
Implementing Wireless Technologies in Automation Systems

Arduino MKR 1000 project

Intellectual Output 3

February 2019

Content

1. Introduction to Arduino MKR 1000

MKR 1000 is one of the many development tools of the Arduino product family. The board includes a microcontroller, a WiFi connectivity module, LiPo battery power supply, USB charging via USB cable, as well as other hardware required for microcontroller operation. The MKR1000 board is shown in Figure

Figure 1. MKR 100 board

We can actually consider a microcontroller as a little computer. However, it has a different purpose than a typical desktop or laptop. We can read the status of different sensors and manage various devices via programs written in the Arduino IDE development interface, which then runs on the microcontroller itself.

You can imagine a microcontroller as a black box with a dozen extensions. The microcontroller functions are used to connect some electronic elements to it and control it with the Arduino IDE program.

In order for the microcontroller to know what to do and how to manage what we have connected to it, we first have to write the program and then we have to load it onto the microcontroller.

To attach additional elements to Arduino tiles, we run microcontrollers that can be accessed via edge connectors (black connectors at the edge of the Arduino plate). The overview of all microcontroller outputs is available in Figure 2.

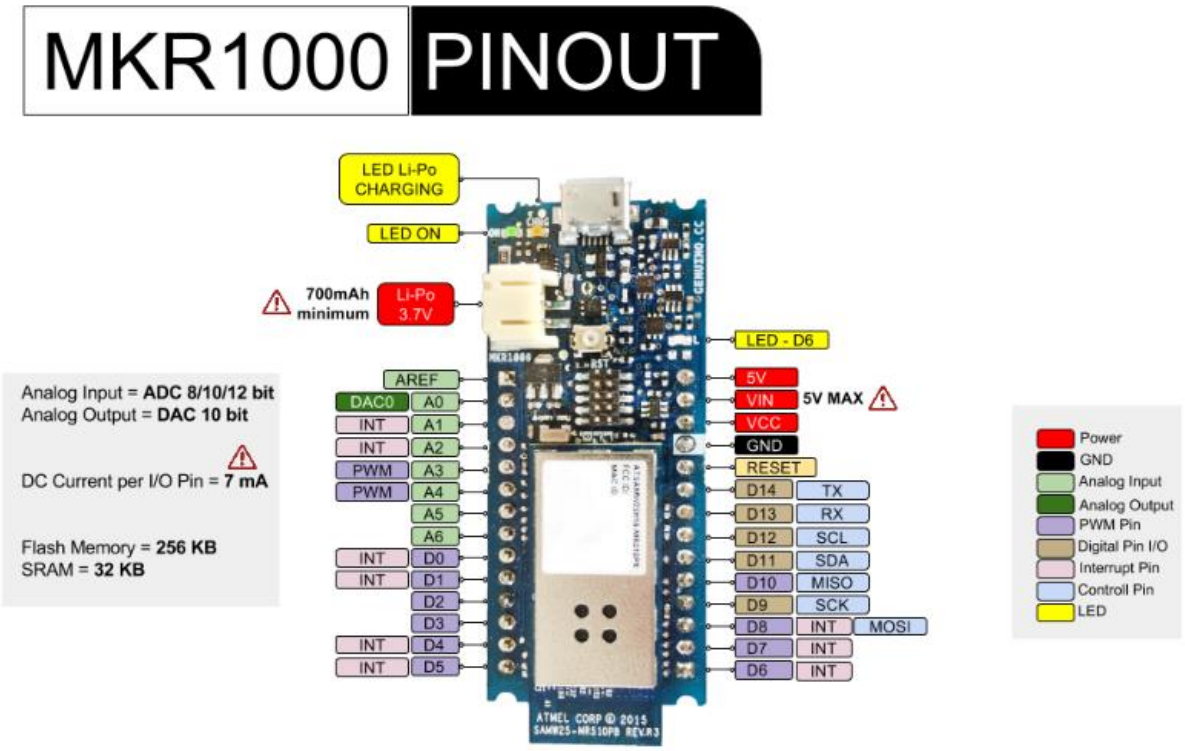


Figure 2. MKR 1000 pinout

As you can see, each pin has its own unique tag. These tags will also be used in the programs we write in order to control the Arduino pins. The Arduino plate is marked with letters and numbers. There are two types of pins, digital outputs (D0) and analog outputs (A0).

Analog outputs are marked with the letter A and numbers - A0, A1, A2 up to A6. These outputs can measure an analog voltage of 0-3.3V.

Digital outputs are marked only by the numbers - 0, 1, 2 up to 14, or the letter D and a number. They can be in only two states - 0 or 1 (LOW or HIGH), or switched ON or OFF.

Analog outputs can also be used as digital ones but digital outputs can not be analog. All pins on the board can work in two modes - in the input and output mode.

The output mode of operation is used when using a microcontroller that we want to manage some of the components that are connected to this output (LEDs, motors, relays, etc.). The input mode is used when we want to read the state of a single output or when connecting simple sensors such as button switches, thermostats and so on.

Let's start with the work and practical examples, so we can best understand the Arduino way of doing things.

2. IOT weather station – part one

This project consists of sensors for the level of brightness, temperature, humidity, pressure and microparticles in the air (air quality measurement) on the MKR1000 platform. Moreover, it contains making a mobile phone application for the visualization of the sensor values. There are few steps for making the weather station unit.

1. Assembling a device for measuring the level of brightness,temperature, pressure and humidity.
2. Assembling a device for measuring microparticles in the air (air quality).
3. Getting tokens to connect to the Blynk server and create mobile applications.
4. Uploading the Arduino program in measuring devices and testing connected devices.
5. Feedback about tokens.

2.1. Assembling a weather station – temperature, humidity, pressure

Part one of the IOT weather station consist of three sensors for measuring lights (APDS sensor), humidity (DHT 11 sensor) and air pressure (BMP 180) – Figure 3.

The schematics in Figure 4 shows the mentioned sensors mounted on the experimental board. Yellow-colored contacts indicate the contacts of the individual sensors that we will connect to the Arduino MKR1000 platform.

Figure 3. Sensors for weather ststion

2.1.1. Connection step 1 (Figure 4):

1. Insert the Arduino MKR1000 platform into the experimental board.
2. Insert the BMP180 sensor into the experimental board.
3. Insert the APDS sensor into the experimental board.
4. Insert the DHT11 sensor into the experimental board.

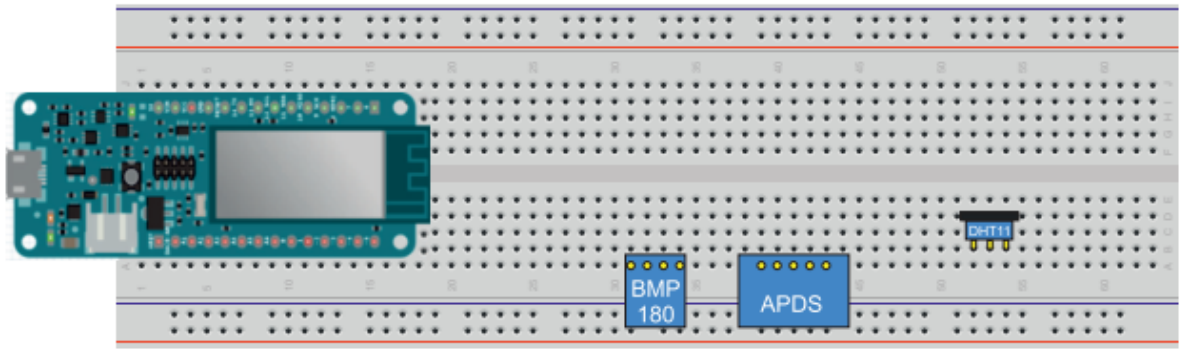


Figure 4. Connection step 1

2.1.2. Connection step 2 (Figure 5):

1. Connect the middle contact of the DHT 11 sensor with the number 2 Arduino board (blue wire in the image).
2. Connect the middle APDS sensor contact to one column of the experimental board (marked with the orange rectangle).
3. Connect the first (left) contact of the BMP180 sensor to the same experiment board column (marked with an orange rectangle).
4. Attach this column (marked with an orange rectangle) with an additional wire to the Arduino digital input pin number 11.

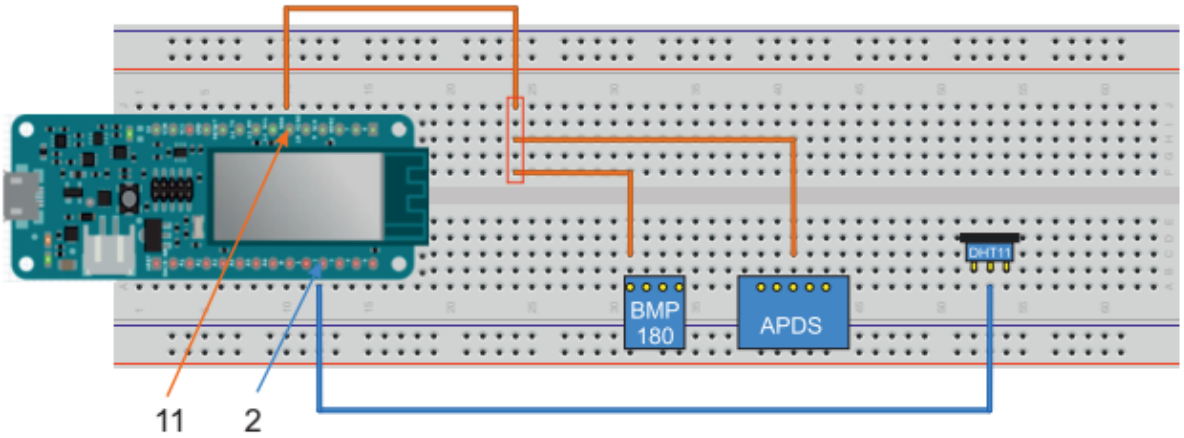


Figure 5. Connection step 2 - connecting sensors to Arduino MKR bord

2.1.3. Connection step 3 (Figure 6):

1. Connect the second APDS sensor contact (yellow wire) to the experimental board column (marked with a blue rectangle).
2. Connect the second BMP180 sensor contact (yellow wire) to the same experimental board column (marked with a blue rectangle).
3. Attach the column (marked with a blue rectangle) to an additional wire to the Arduino digitalinput pin number 12.

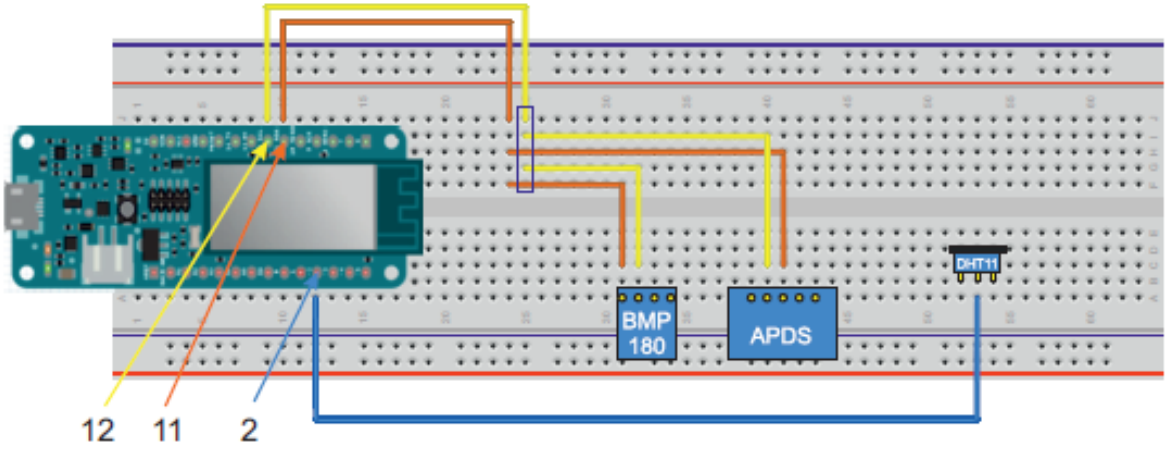


Figure 6. Connection step 3

2.1.4. Connection step 4 (Figure 7):

1. Connect the Arduino „VCC“ (5V) pin to the common (+) row on the experimental board (red wire on Figure 8).
2. Connect the Arduino „GND“ pin to the common (-) line on the experimental board (black wire on Figure 8).
3. Connect the third pin on the BMP180 sensor to the common (-) line on the experimental board (black wire on Figure 8).
4. Connect the fourth pin on the BMP180 sensor to the common (+) line on the experimental board (red wire on Figure 8).
5. Connect the fourth pin on the APDS sensor to the common (+) line on the experimental plate (red wire on Figure 8).
6. Connect the fifth pin on the APDS sensor to a common (-) on the experimental board (black wire on Figure 8).

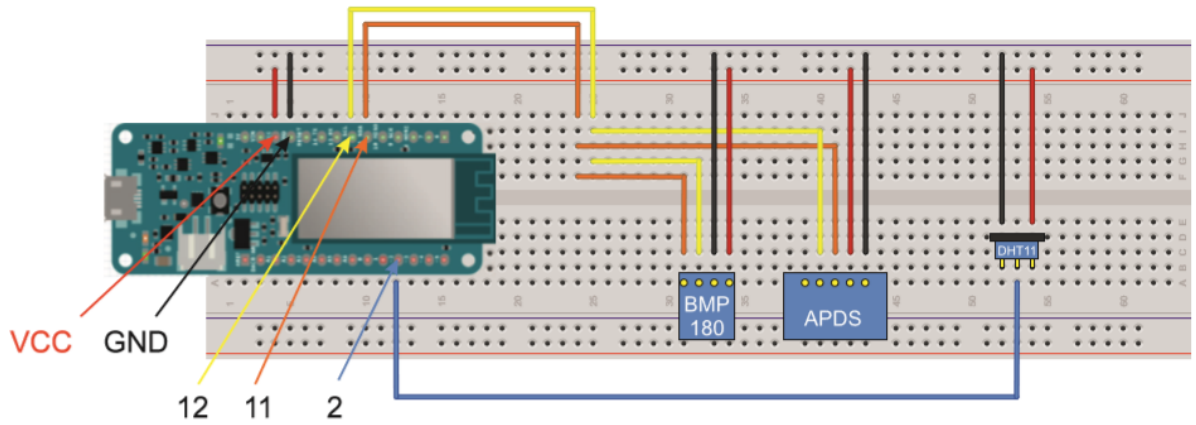


Figure 7. Connection step 4

3. IOT weather station – part two

This project consists of a plantower sensor for measuring microparticles in the air (air quality) on the MKR1000 platform and creating a mobile phone application for the visualization of the sensor values. There are a few steps for making the weather station unit.

3.1. Assembling the weather station – microparticles in the air

Part two of the IOT weather station consists of one sensor for measuring microparticles in the air lights (air quality) - Figure 8.

The schematics in Figure 10 show the mentioned sensor connected to the experimental board.

Figure 8. Plantower sensor for measuring microparticles in the air

3.1.1. Connection step 1 (Figure 9):

Plug in the white connector to the printed circuit bord and mount it on the plantower sensor for measuring microparticles in the air. Make sure to position the printed circuit board to fit in the Plantower sensor.

Figure 9. Connecting printed circuit board to Plantower sensor

3.1.2. Connection step 2 (Figure 10):

1. Connect the red wire of the Plantower sensor to the „VCC“ (5V) pin on the Arduino MKR 1000 board.
1. Connect the black wire of the Plantower sensor to the „GND“ pin on the Arduino MKR 1000 board.
2. Connect the blue wire of the Plantower sensor to the digital pin 14 on the Arduino MKR 1000 board.
4. Connect the yellow wire of the Plantower sensor to the digital pin 13 on the Arduino MKR 1000 board.

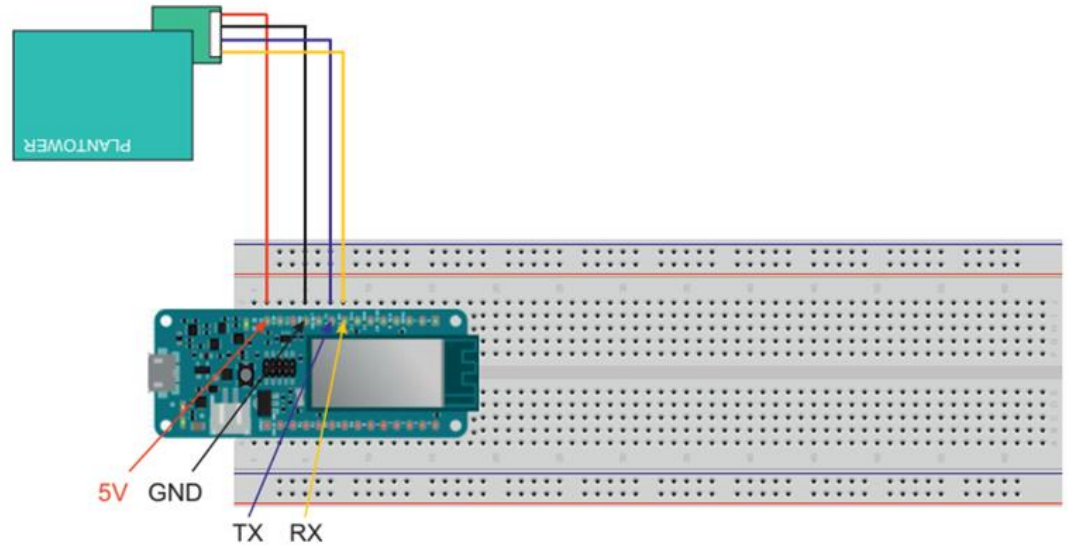


Figure 10. Connection of Plantower sensor to Arduino MKR 1000 board

Remark:

The Plantower sensor wires have to be extended with male-male wires before the connection to the Arduino MKR 1000 board.

4. Blynk platform

Blynk is an Internet of Things open-source platform aimed to simplify building mobile and web applications for the Internet of Things. It can control hardware remotely, display sensor data, store data, vizualize it and do many other things. The Blink platform can easily connect up to 400+ hardware models like Arduino, ESP8266, ESP32, Raspberry Pi and similar MCUs and drag-n-drop IOT mobile apps for iOS and Android.

There are three major components in the platform:

1. **Blynk App** - allows creating the amazing interfaces for the projects using various widgets.
2. **Blynk Server** - responsible for all the communications between the smartphone and hardware. It also allows running private Blynk server locally for the projects. The Blynk server can easily handle thousands of devices and it can also be launched on a Raspberry Pi.
3. **Blynk Libraries** – enables communication with the server and process all the incoming and outcoming commands.

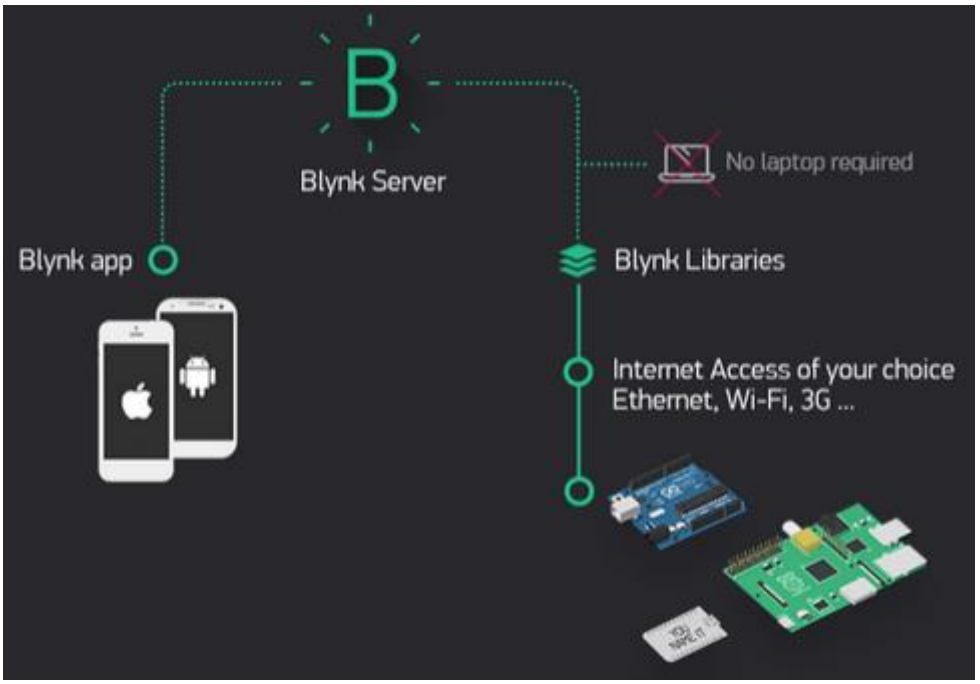


Figure 11. Blynk platform

4.1. Downloading the application

Before starting the first project, it is necessary to download and install the Blynk application on a smartphone or tablet. The application can be downloaded from Google Store (Figure 12.) if working on the Android operating system, or from App Store (Figure 12) if working on an iOS operating system. It can be found by searching the „Blynk“ keyword.

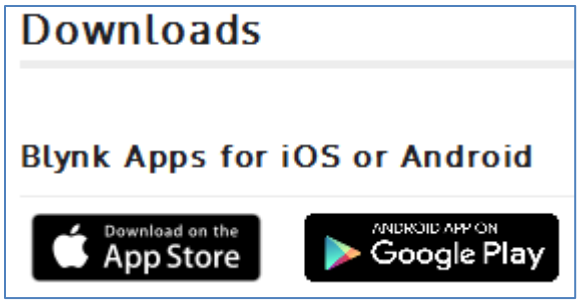


Figure 12. Blynk operating system for smartphones

Direct links for download are:

Android: https://play.google.com/store/apps/details?id=cc.blynk

IOS:https://itunes.apple.com/us/app/blynk-control-arduino- raspberry/id808760481?ls=1&mt=8

4.2. Creating the project

After finishing the download, open the application and create a registration by clicking „Create New Account" (Figure 13.). A registration is required to prepare the projects directly on the Blynk server.

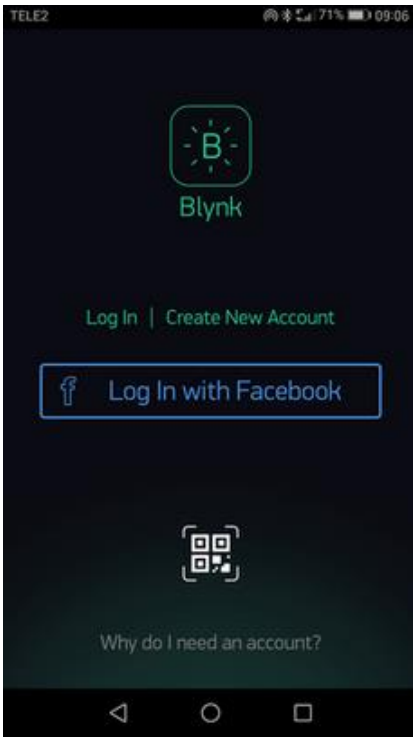


Figure 13. Blink application - creating the account

Enter the e-mail and password for the registration.

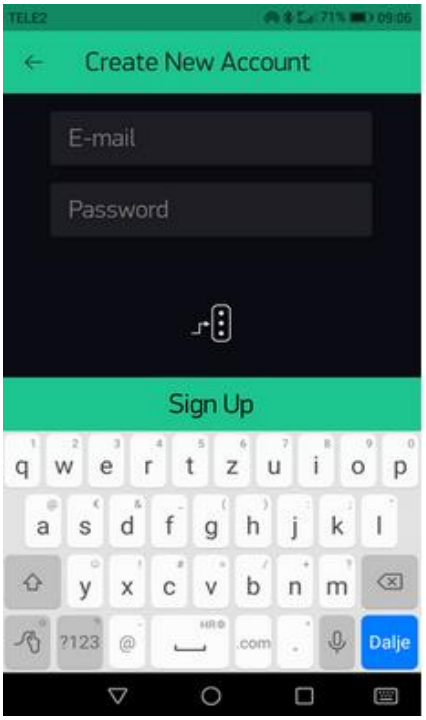


Figure 14. Blink application - email and password

After the registration, we will create a new project by selecting the „New Project“ option (Figure 15.).

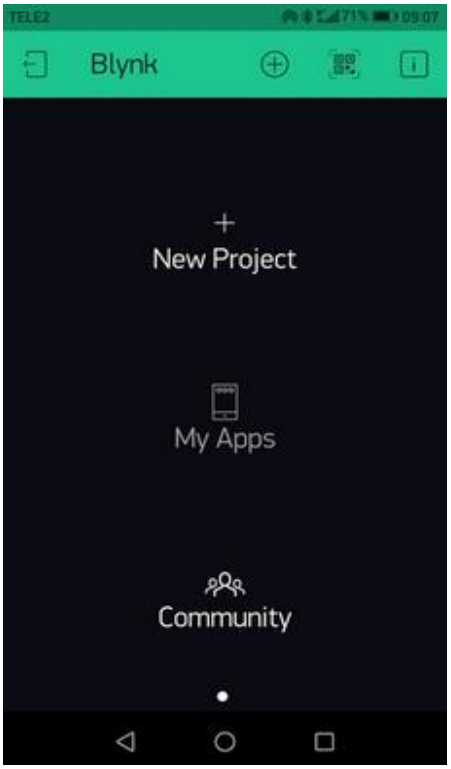


Figure 15. Blink application – creting the new project

Enter the basic project information - select the project name, the hardware you are using (Arduino MKR1000), the connection type (WiFi) and the color of the interface – Light (Figure 16.).

Figure 16. Blynk application - project options

After the project creation, a notification is displayed that tells you that an authentication token was created and that it was sent to the e-mail address. This token is unique to each project and it is used to connect the hardware with the Blynk application. It is needed later in the Arduino program code.

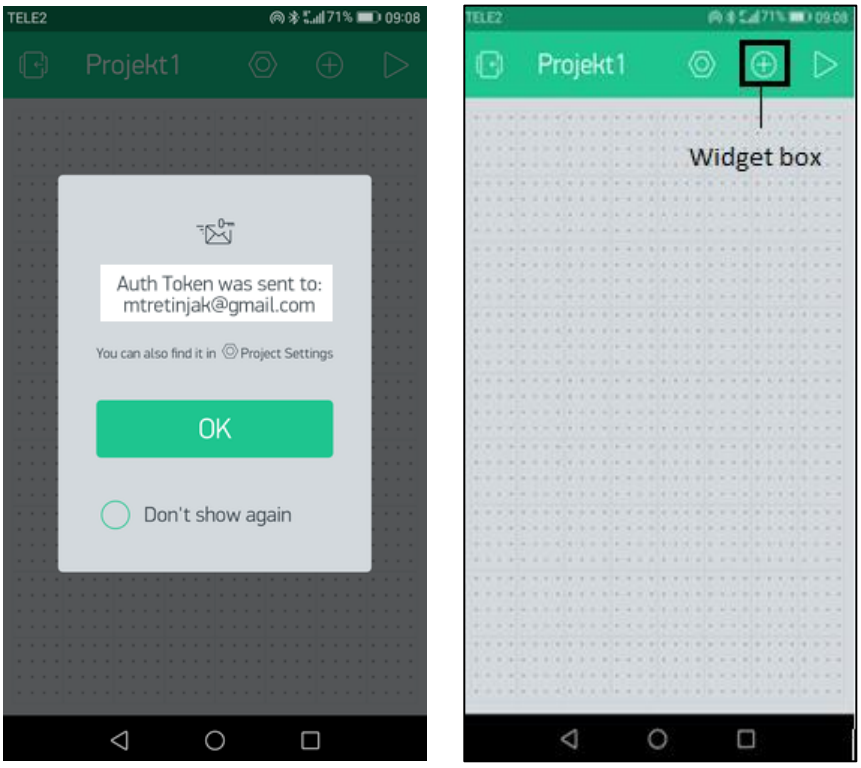


Figure 17. Blynk application - authentification token and widget box

Write down your autentification token from the registered e-mail for project 1 and later for project 2.

4.3. Blink widget box

Add widgets needed for project 1 and project 2 in the Blynk application window as follows.

4.3.1. Project 1 – IOT weather station (first part)

- Add the Value display widget module and the Button widget module to project 1. In the Value name, type the measuring value text and set the pin to show the value as follows in the next step.
- Repeat step 4 several times to get a total of 5 value displays that will display the measured values according to the list:

➔ Value display - Temp - PIN: Virtual V1
➔ Value display - Pressure - PIN: Virtual V2
➔ Value display - UVI - PIN: Virtual V3
➔ Value display - Light - PIN: Virtual V4
➔ Value display - Humidity - PIN: Virtual V5

Add one Button module to the project which will switch the LED on/off on the Arduino digital pin - D6. Set it as a SWITCH.

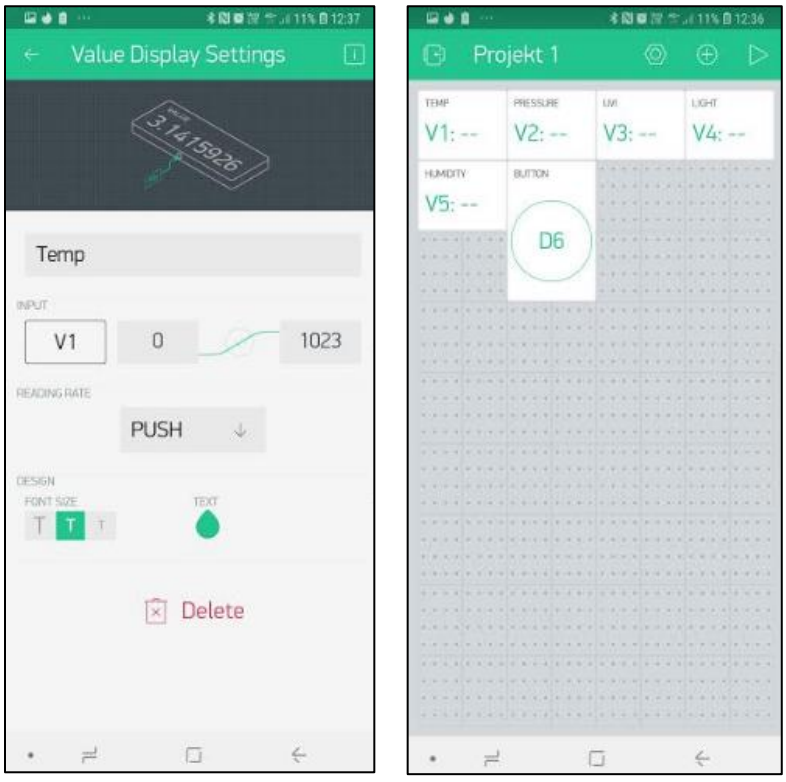


Figure 18. Blynk application - adding widgets and settings

4.3.2. Project 2 – IOT weather station (second part)

Add another project (project 2) by clicking the „+“ icon in the top toolbar, and don't forget to write down your authentification token for the second project. Under the project name, enter „project2“, all without spacing. The other data are the same as in Project 1.

• Add Value display widget modules to project 2, equally as in project 1.

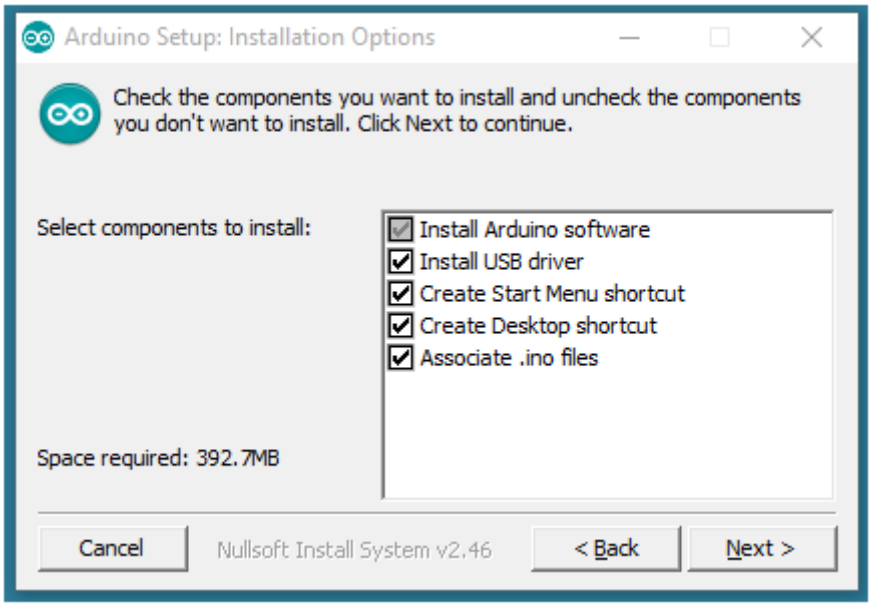
- ➔ Value display - PM 1 - PIN: Virtual V10
- ➔ Value display - PM 2.5 - PIN: Virtual V11
- ➔ Value display - PM 10 - PIN: Virtual V12

5. Arduino IDE program

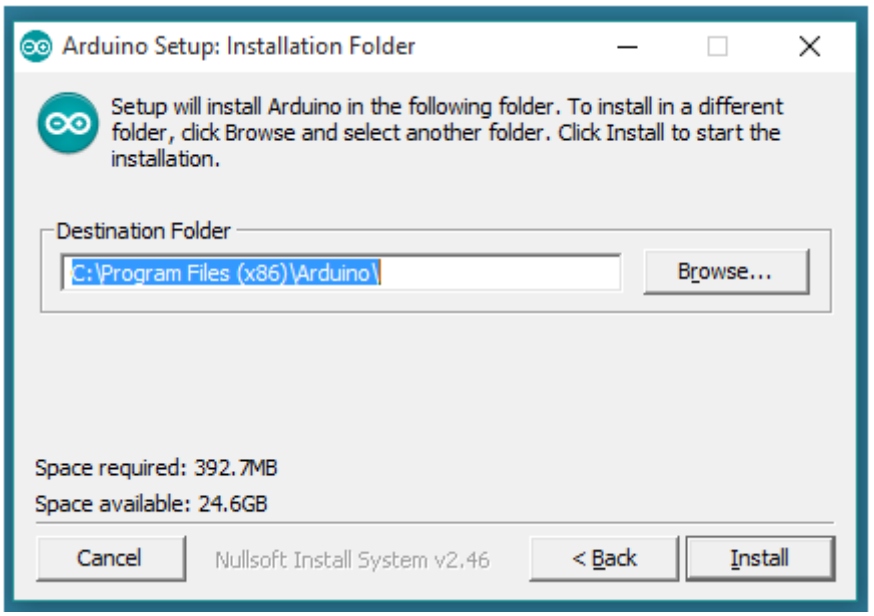
The open-source Arduino Software (IDE) makes it easy to write a code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on processing and other open-source software. This software can be used with any Arduino board.

5.1. Download the Arduino IDE software

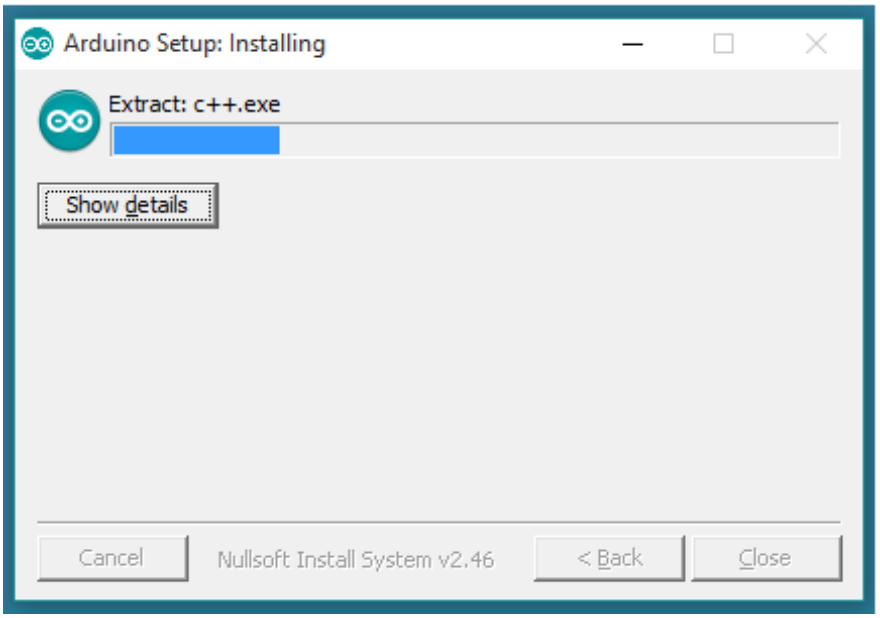
Get the latest version of the Arduino IDE software from the download page. You can choose between the Installer (.exe) and the Zip packages. It is better to use the first one because installs directly everything needed to use the Arduino Software (IDE), including the drivers. With the Zip package you need to install the drivers manually. The Zip file is also usefull if you want to create a portable installation. When the download finishes, proceed with the installation and please allow the driver installation process when you get a warning from the operating system.



Choose the components to install.



Choose the installation directory (keep the default one).



The process will extract and install all the required files to execute properly the Arduino Software (IDE). When the Arduino Software (IDE) is properly installed you can go back to the Getting Started Home and choose your board from the list on the right of the page.

5.2. Arduino IDE software explanation

After installing Arduino, please open it up and refer to the photos above where I showed you several key parts of the IDE (Integrated Development Environment).

1. **The circle** labeled with a 1 is circling the upload button. When your Arduino is plugged into the computer, click this button to upload your code to the Arduino.

2**. Void setup** is the section of your code that will run once and only once. When your Arduino is powered on or reboots, it will run all the codes you have placed in your void setup section

3. **Void loop** is the section of your code that will run over and over again. After the void setup is run, all your codes in void loop will run. The difference is that after all your codes in the void loop are run, the Arduino will go back to the start of your void loop section and run it all again. This repeats over and over again and thus is called a loop.

4. **The serial Monitor** button. This button opens up the serial monitor. The serial monitor allows you to send commands to the Arduino and receive messages back from the Arduino. More on how to do this later.

5. This is the **Arduino Console**. Arduino will print out error messages here if it has issues uploading to the board or compiling the code.

6. In the **Tools menu** at the top there is a Board option that you can select. Use this to select the correct type of board. Usually the type of Arduino board is written on the device somewhere.

7. This is the option for selecting which **com port** connects your Arduino to the computer. If you have troubles figuring it out, unplug your Arduino, look at the list and plug your Arduino back in. The new com port is the one for your Arduino.

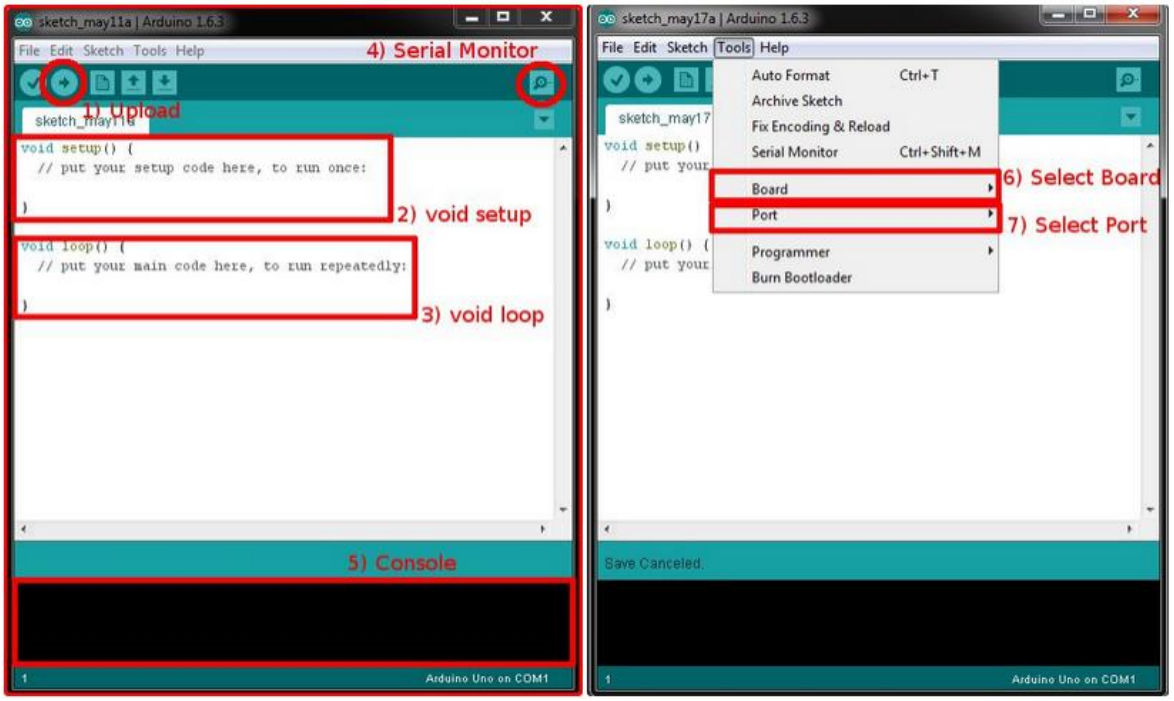


Figure 19. Arduino IDE environment

5.3. Uploading Arduino's program into measuring devices and testing

We have two programs for our projects, TTVSUmeasuring for project 1 and PMS measuring for project 2. The programs will be stored as a zip file and shared through Google Drive.

TTVSUmeasuring - program for measuring light, temperature, pressure and humidity.

PMSmeasuring - program for measuring microparticles in the air.

5.3.1. Inserting programs to our projects Project 1 – TTVSU measurement

- ➔ Store the program on your computer.
- ➔ Open Arduino IDE on your computer.
- ➔ Open the „TTVSUmeasuring“ program.

Step 1. - sketch

After opening the sketch, we need to install several libraries. Click Sketch, then Include Library, then Manage Libraries. Make sure you have all these libraries. If you do not have them, install them:

- ➔ Blynk,
- ➔ WiFi101,
- ➔ Adafruit BMP085,
- ➔ SimpleDHT,
- ➔ PMS.

In Google documents you will also find email, another library - APDS9200. Save it (.zip file) somewhere on your computer.

Click on Sketch, then Include Library, then Add zip library, and open the APDS9200.zip file.

```
#include <Wire.h>                                          //using library wire.h
#include <Adafruit_BMP085.h>                               //using library for reading BMP180 sensor
#include "APDS9200.h"                                      //using library for reading light level sensor
#include <SimpleDHT.h>                                     //using library for DHT11 sensor

#include <SPI.h>                                           //using library for SPI comununication
#include <WiFi101.h>                                       //using WiFi 101 library for internet connection
#include <BlynkSimpleMKR1000.h>                            //using library for Blynk application

int pinDHT11 = 2;                                          //DHT11 sensor pin connected to digital pin 2

Adafruit_BMP085 bmp;                                       //define object bmp
APDS9200 light;                                            //define object light
SimpleDHT11 dht11;                                         //define object dht11

BlynkTimer timer;                                          //using Blynk timer

double temp180[6];                                         //variable for measuring temperature
double temp180sr = 0;                                      //variable for measuring temperature
double tlak180[6];                                         //variable for measuring pressure
double tlak180sr = 0;                                      //variable for measuring pressure

double uv[6];                                              //variable for measuring uv
double uvindexsr = 0;                                      //variable for measuring uv
double svjetlo[6];                                         //variable for measuring level of light
double svjetlosr = 0;                                      //varijable za mjerenje level of light

double vlaga11[6];                                         //variable for measuring humidity
double vlaga11sr = 0;                                      //variable for measuring humidity

int brojac = 0;                                            //varijable counter

char auth[] = "  ";                                        //enter the autentification token for project 1

// Your WiFi credentials.
// Set password to "" for open networks.
char ssid[] = "  ";                                        //enter the network name
char pass[] = "  ";                                        //enter the network password

void setup() {
  Serial.begin(9600);                                      //serial connection initialization
  if (!bmp.begin()) {
    Serial.println("BMP senzor nije pronaden");            //checking of MBP sensor initialization
    while (1) {}                                           //stop the program
  }
  Wire.begin();                                            //starting the wire comunication

  Blynk.begin(auth, ssid, pass);                           //starting the Blynk servis
  timer.setInterval(10000L, myTimerEvent);                 //setting the values interval on 10 sec.
  brojac = 0;                                              //setting the starting value of the counter
}

void myTimerEvent()                                        //function active every 10 sec. using timer
{
  if (brojac < 6) {

    //----TEMPERATURE MEASUREMENT AND PRINTING THE VALUE--------TEMPERATURE MEASUREMENT AND PRINTING THE VALUE--------
    temp180[brojac] = bmp.readTemperature();
    tlak180[brojac] = bmp.readPressure();
    temp180sr = temp180sr + temp180[brojac];
    tlak180sr = tlak180sr + tlak180[brojac];

    Serial.println("-----------BMP180");
    Serial.print("Temperature = ");
    Serial.print(temp180[brojac]);
    Serial.println(" *C");

    Serial.print("Pressure = ");
    Serial.print(tlak180[brojac]);
    Serial.println(" Pa");
    //---------------------------------------------------------------------------------------------------------------
```

```
//----LIGHT LEVEL MEASUREMENT AND PRINTING THE VALUE--------LIGHT LEVEL MEASUREMENT AND PRINTING THE VALUE--------
light.enableUV();
delay(500);
uv[brojac] = light.getUV();
light.enableLight();
delay(500);
svjetlo[brojac] = light.getLight();
uvindexsr = uvindexsr + uv[brojac];
svjetlosr = svjetlosr + svjetlo[brojac];
Serial.println("-----------UV&LIGHT");
Serial.print("UV:");
Serial.println(uv[brojac]);
Serial.print("Svjetlo:");
Serial.println(svjetlo[brojac]);
//-------------------------------------------------------------------------------------------------------------

//----HUMIDITY MEASUREMENT AND PRINTING THE VALUE--------HUMIDITY MEASUREMENT AND PRINTING THE VALUE--------
byte temperature = 0;
byte humidity = 0;
int err = SimpleDHTErrSuccess;
if ((err = dht11.read(pinDHT11, &temperature, &humidity, NULL)) != SimpleDHTErrSuccess) {
  //Serial.print("Read DHT11 failed, err="); Serial.println(err); delay(1000);
  //return;
}

//temp11 = (double)temperature;
vlagall[brojac] = (double)humidity;
vlagallsr = vlagallsr + vlagall[brojac];
Serial.println("-----------DHT11");
Serial.print(vlagall[brojac]); Serial.println(" %");
//-------------------------------------------------------------------------------------------------------------

  brojac++;

} else {
  //----TEMPERATURE AND PRESSURE MEASUREMENT AND PRINTING THE VALUE-----------------------------------------
  temp180sr = temp180sr / 6;
  tlak180sr = tlak180sr / 600;
  Blynk.virtualWrite(V1, temp180sr);
  Blynk.virtualWrite(V2, tlak180sr);

  Serial.print("Srednja temp = ");
  Serial.print(temp180sr);
  Serial.println(" *C");
  Serial.print("Srednji tlak = ");
  Serial.print(tlak180sr);
  Serial.println(" hPa");

  temp180sr = 0;
  tlak180sr = 0;
  //-------------------------------------------------------------------------------------------------------------
```

```
//----UV AND LIGHT LEVEL--------UV AND LIGHT LEVEL--------UV AND LIGHT LEVEL--------UV AND LIGHT LEVEL------
uvindexsr = uvindexsr / 6;
svjetlosr = svjetlosr / 6;
if ((uvindexsr >= 0) && (uvindexsr < 275)) {
  uvindexsr = 1;
} else if ((uvindexsr >= 275) && (uvindexsr < 525)) {
  uvindexsr = 2;
} else if ((uvindexsr >= 525) && (uvindexsr < 775)) {
  uvindexsr = 3;
} else if ((uvindexsr >= 775) && (uvindexsr < 1050)) {
  uvindexsr = 4;
} else if ((uvindexsr >= 1050) && (uvindexsr < 1325)) {
  uvindexsr = 5;
} else if ((uvindexsr >= 1325) && (uvindexsr < 1675)) {
  uvindexsr = 6;
} else if ((uvindexsr >= 1675) && (uvindexsr < 1960)) {
  uvindexsr = 7;
} else if ((uvindexsr >= 1960) && (uvindexsr < 2240)) {
  uvindexsr = 8;
} else if ((uvindexsr >= 2240) && (uvindexsr < 2520)) {
  uvindexsr = 9;
} else if ((uvindexsr >= 2520) && (uvindexsr < 2800)) {
  uvindexsr = 10;
} else if ((uvindexsr >= 2800) && (uvindexsr < 3080)) {
  uvindexsr = 11;
} else {
  uvindexsr = 12;
}

Blynk.virtualWrite(V3, uvindexsr);
Blynk.virtualWrite(V4, svjetlosr);

Serial.print("UV index = ");
Serial.println(uvindexsr);
Serial.print("Svjetlost = ");
Serial.println(svjetlosr);

uvindexsr = 0;
svjetlosr = 0;

//-------------------------------------------------------------------------------------------------------------------

//----HUMIDITY MEASUREMENT--------HUMIDITY MEASUREMENT--------HUMIDITY MEASUREMENT------------------------------
vlagallsr = vlagallsr / 6;

Blynk.virtualWrite(V5, vlagallsr);

Serial.print("Srednja vlaga: ");
Serial.print(vlagallsr); Serial.println(" %");

    vlagallsr = 0;
    //-----------------------------------------------------------------------------------------------------------

    brojac = 0;                                                    //setting counter to 0
  }

}

void loop() {
  Blynk.run();                                                     //function for server connection and sending data
  timer.run();                                                     //timer function
}
```

Figure 20. Program code for the project 1

Step 2 – connecting Arduino (project 1)

Connect the Arduino meter to measure light, temperature, pressure and humidity. In Arduino's programming environment, make sure Arduino MKR1000 is selected under Tools-> Bord.

At the registered e-mail, you will have received the authorization token for the **first project** and you have to write it in the sketch.

```
char auth[] = "";       // insert the blink token into the quotation marks
char ssid[] = "";       // insert the wifi network name into the quotation marks
char pass[] = "";       // insert the password for connection to the wifi newtwork into
                           the quotation marks
```

Upload the program on the Arduino board. After completing the upload, open the Serial Monitor by clicking the magnifying glass icon at the top right corner of the Arduino program environment.

In a new window, you should get the measured values from all the connected sensors. Values will be printed every 10 seconds. In the first draft made within the Blynk application, make sure that the values are updated (every 60 seconds).

5.3.2. Uploading a program to project 2 – *PMSmeasurement*

Step 1 – connecting Arduino (project 2)

Connect the Arduino MKR 1000 to measure microparticles in the air (air quality). In Arduino's programming environment, make sure Arduino MKR1000 is selected under Tools-> Bord.

At the registered e-mail, you will have received the authorization token for the second project and you have to write it in the sketch.

```
char auth[] = "";       // insert the blink token into the quotation marks
char ssid[] = "";       // insert the wifi network name into the quotation marks
char pass[] = "";       // insert the password for connection to the wifi newtwork into
                           the quotation marks
```

Upload the program on the Arduino board. After completing the upload, open the Serial Monitor by clicking the magnifying glass icon at the top right corner of the Arduino program environment.

```
#include "PMS.h"

#include <SPI.h>
#include <WiFi101.h>
#include <BlynkSimpleMKR1000.h>

PMS pms(Serial1);
PMS::DATA data;

double pm1sr=0;
double pm25sr=0;
double pm10sr=0;

char auth[] = "";
char ssid[] = "";
char pass[] = "";

void setup()
{
  Serial.begin(9600);
  Serial1.begin(9600);
  Blynk.begin(auth, ssid, pass);
}
void loop()
{

  if (pms.read(data))
  {

    pm1sr = data.PM_AE_UG_1_0;
    pm25sr = data.PM_AE_UG_2_5;
    pm10sr = data.PM_AE_UG_10_0;

    Serial.println("Data:");

    Serial.print("PM 1.0 (ug/m3): ");
    Serial.println(pm1sr);

    Serial.print("PM 2.5 (ug/m3): ");
    Serial.println(pm25sr);

    Serial.print("PM 10.0 (ug/m3): ");
    Serial.println(pm10sr);

    Serial.println();

    Blynk.virtualWrite(V10, pm1sr);
    Blynk.virtualWrite(V11, pm25sr);
    Blynk.virtualWrite(V12, pm10sr);
  }
}
```

Figure 21. Program code for the project 2

In a new window, you should get the measured values from all the connected sensors. The values will be printed every 10 seconds. In the first draft made within the Blynk application, make sure that the values are updated (every 60 seconds).

5.3.3. Merging program codes

Additionally, program codes from both projects can be merged into one project. Merging programs save one MKR1000 platform for other projects. After merging all the measured values from both projects, they will be displayed in your Blynk application and serial monitor in the Arduino IDE program (Figure 22).

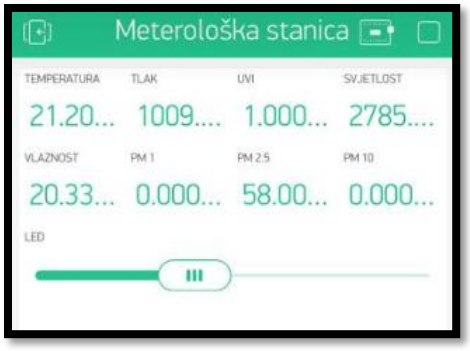


Figure 22. Measured values through Blyink application