

# Load monitoring and control

## Data Visualization

**SESSION 1: Online data visualization using Arduino over Serial USB**

**Arduino and Blynk**

## Table of Contents

Introduction .....	3
Part 1: Install the libraries .....	4
Part 2: Download Blynk App.....	4
Part 3: Create a project using Blynk App.....	4
Part 3: Programming the Arduino IDE.....	9
Definition of variables .....	9
Setup Function .....	10
Main function .....	11
Blynk Function .....	11
Part 4. Connect the Arduino to the mobile App .....	12
Part 5. Exporting data to CSV .....	13

## Introduction

In this session, we will visualize data using two different approaches. The first one will consist in visualize the data acquired by our Arduino in real-time by means of a mobile App. The second approach will use python libraries to visualize and interact with data stored previously in a CSV file.

The main purpose of these two approaches are different, while the first one is designed to visualize data and control directly the appliance using the App, the second one is suitable to analyze the data recollected during all the project to do reports and extract analytical results.

These two tools are not mutually exclusive, so their use will depend on the needs of each students' project.

This part will consist in sending the data recollected by the Arduino to a mobile App to visualize it. Usually, this data would be sent by means of a Wi-Fi or Ethernet connection, however, Arduino UNO would need an Ethernet Shield for this purpose.

If you don't have any shield and your hardware doesn't have any connectivity, Blynk is a mobile App with high hardware compatibility that allows the Arduino to send the data directly over USB, using the internet connection of our Laptop for this purpose.

These guidelines will guide the student to install in our Laptop the Blynk libraries. These libraries will have two purposes:

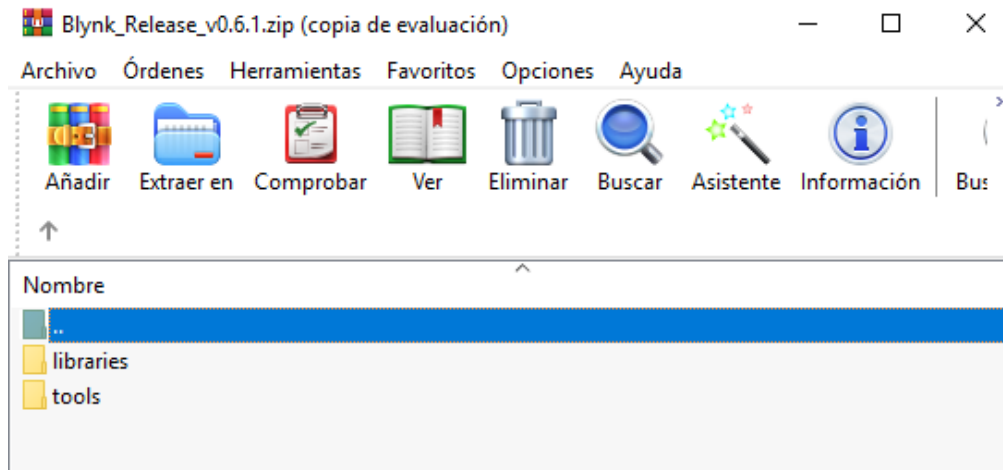
- Add extra functionalities to our Arduino IDE to program the board with the Blynk functionalities.
- Open a tunnel in our Laptop to recollect the data from the Arduino via USB and send it to the cloud.

## Part 1: Install the libraries

The first step will be to download the Blynk libraries and install them. First, go to the following link:

<http://help.blynk.cc/en/articles/512105-how-to-install-blynk-library-for-arduino>

Download the zip file with the libraries, open the file and it will contain the following folders:



Unzip the two folders to our local Arduino sketchbook folder, for example:

C:\Users\cristian.chillon\Documents\Arduino

To find the location of your sketchbook folder, go to the top menu in Arduino IDE:

Windows: File → Preferences

Mac OS: Arduino → Preferences

## Part 2: Download Blynk App

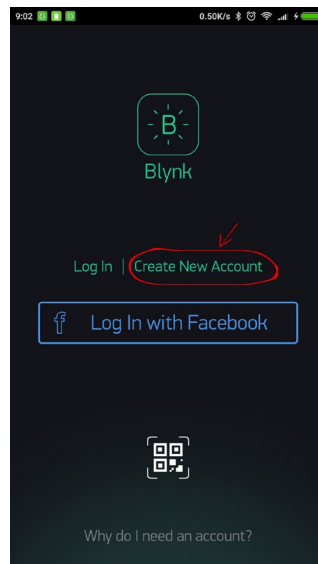
In the function of your mobile phone brand, download the corresponding version of the Blynk App using the following links:



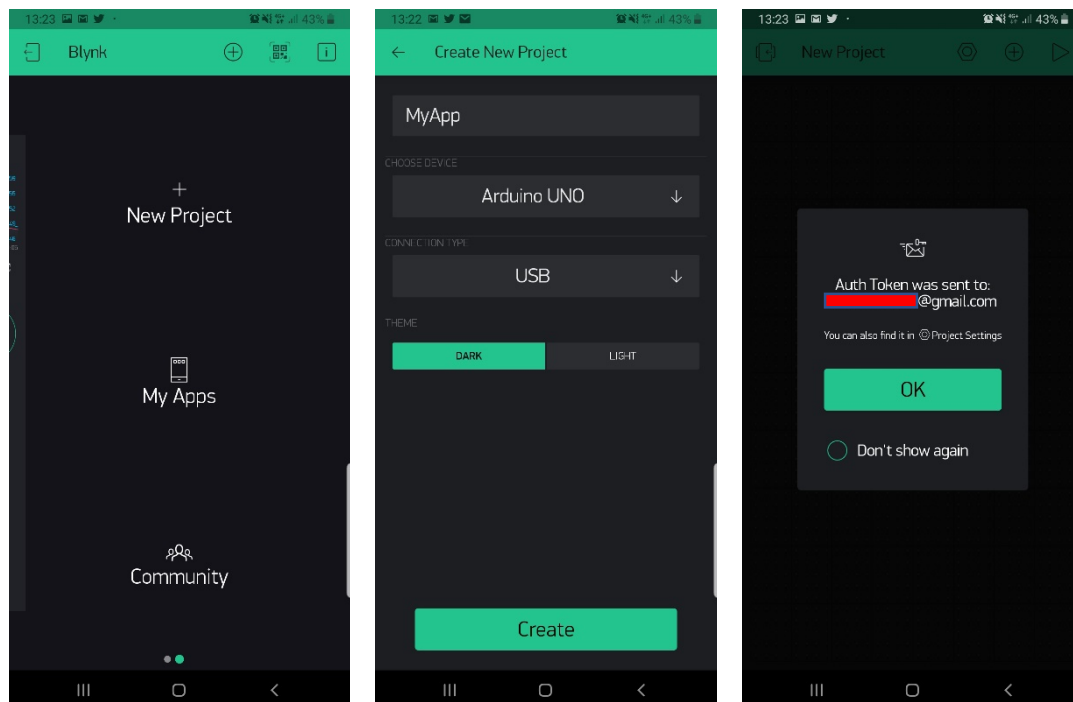
Once the app is installed, you can follow the instructions from <https://docs.blynk.cc/>

## Part 3: Create a project using Blynk App

The first step will be creating a new user account:

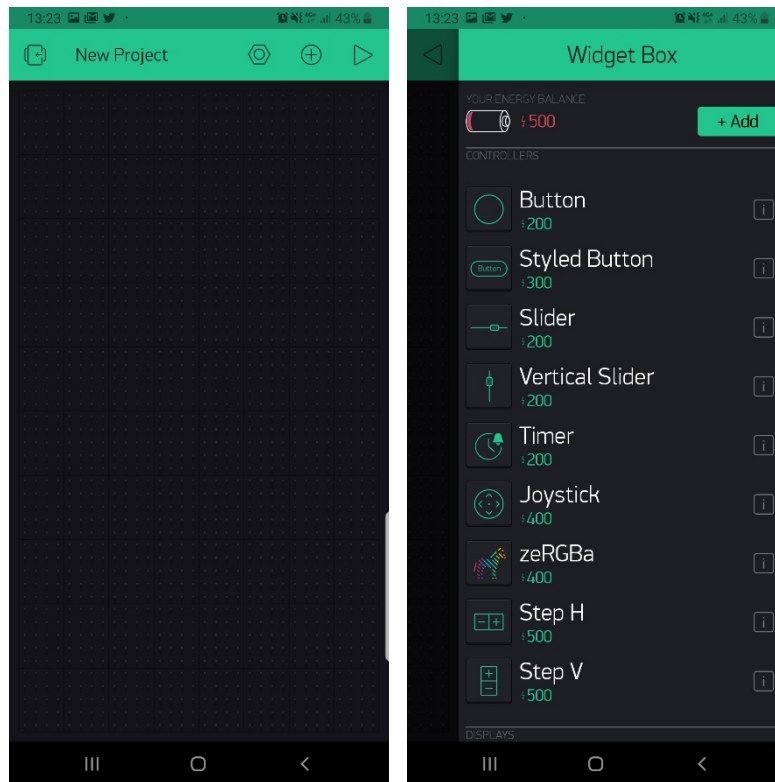


The second step, after logging with your new user, will be creating a new project and selecting our hardware, in this case, an Arduino UNO and the connection type, in this case, the USB.

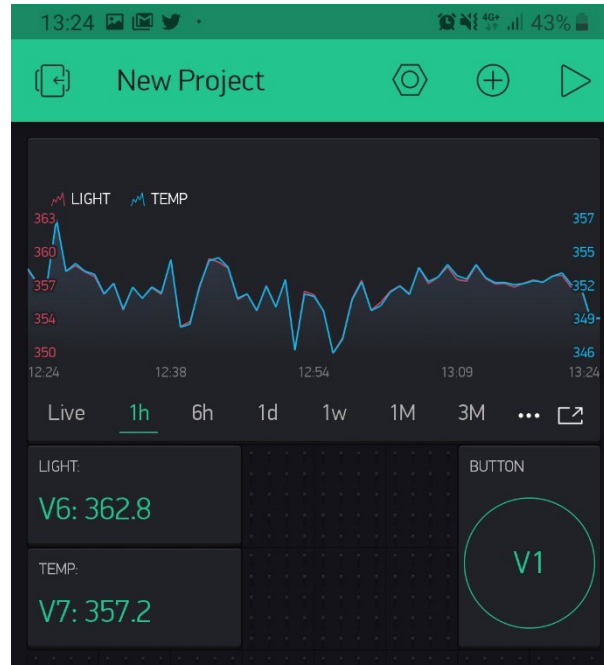


After creating the new project, Blynk will send us an email with a token. This token will be the authentication key to send data from our Arduino to the Blynk cloud service, to be later visualized in the mobile App.

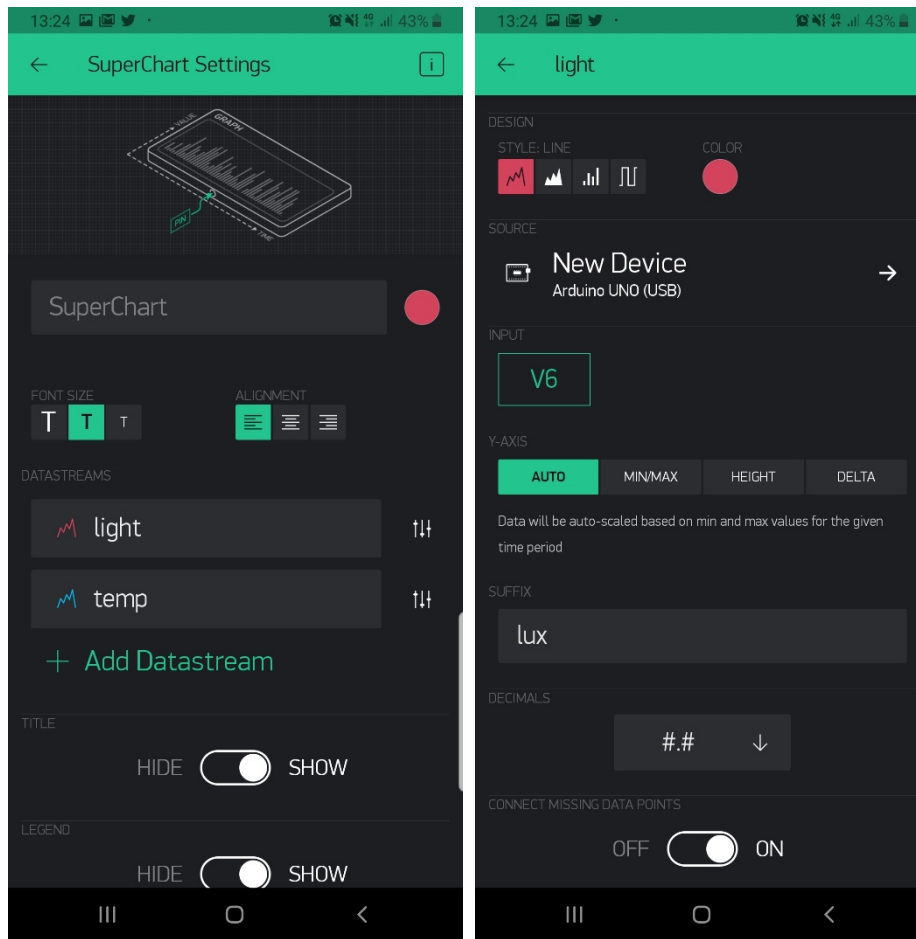
Your project is an empty canvas, tap the + button to open the Widget Box. All the available widgets are located here. Each widget has a cost and we have a total amount of 2000 points.



For today's session, we will implement a *Button* to control a LED, a *SuperChart* to visualize the historical data of two sets of values, and two *Value Display* to show the current value of these two parameters, resulting in a canvas like the following one:



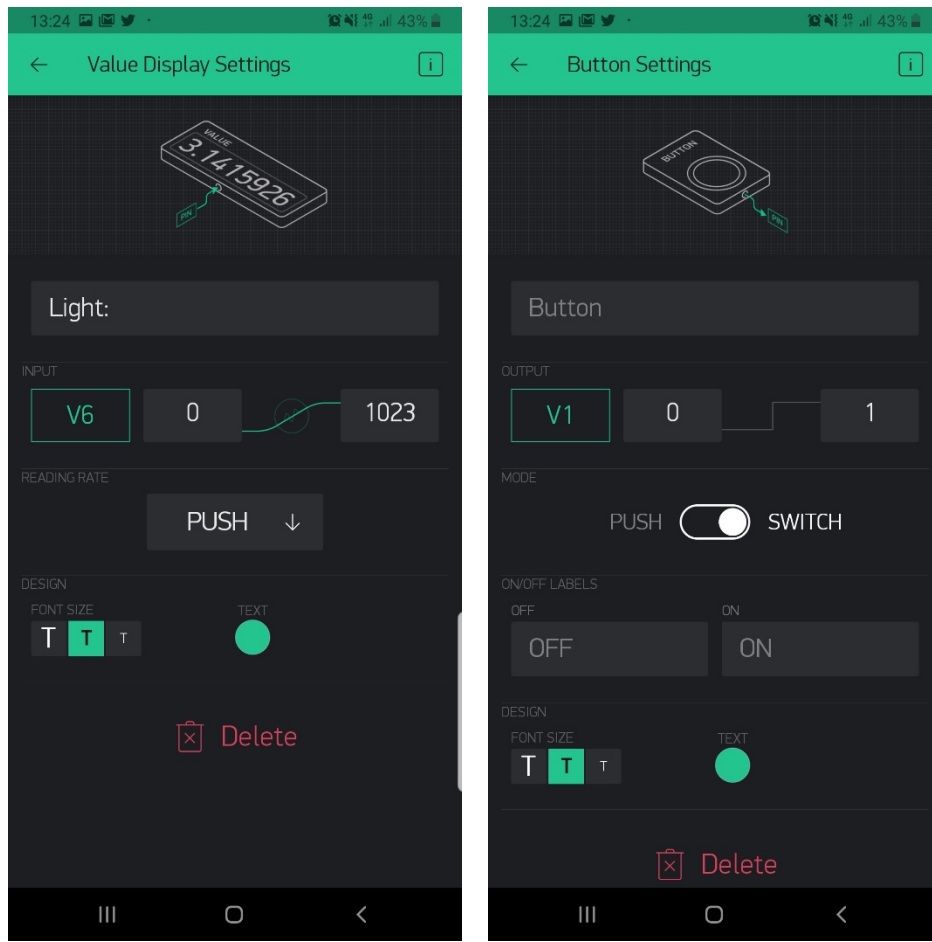
To configure the *SuperChart* widget, click on it and the next setting windows will appear:



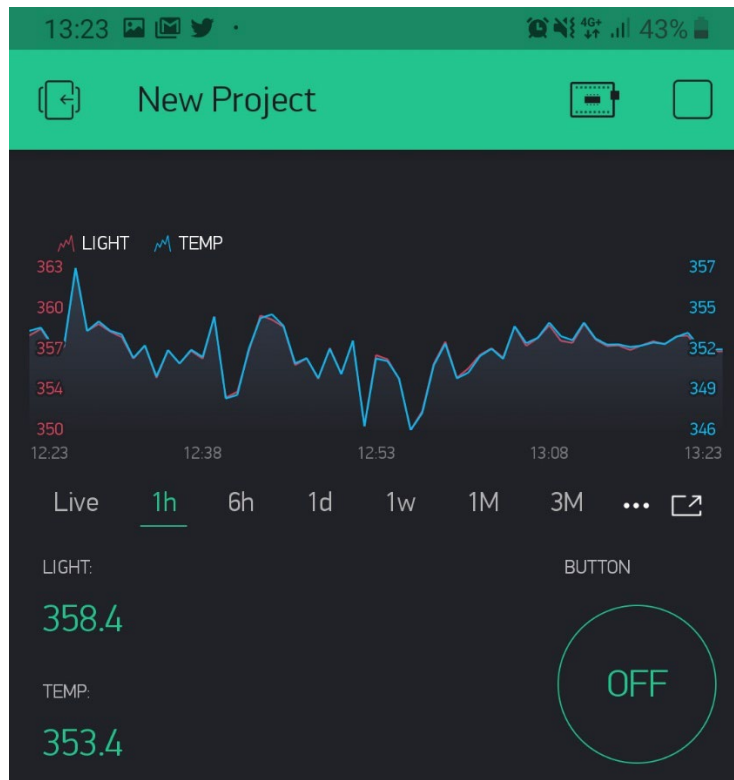
We can add various *data streams* in the same chart to visualize more than one kind of data. Each *data stream* can be configured independently. For example, we will create a *datastream* called *light* linked to a *Virtual Pin 6*, V6. This way, when the Arduino sends the light values to the Blynk, it uses this identifier V6 for this purpose.

In a similar way, data can be shown in a *Value Display* representing the last value read by the Arduino. The widget has to be configured as shown in the following picture where the user can configure the virtual.

Finally, to configure a button to control remotely our Arduino can be done easily using the *Button* widget. For today's session, we will create a button linked to a virtual pin V1. This button will be configured as a Switch, and each time its value changes, the App Blynk will send this information to our Arduino to act in consequence.



Once the canvas is completed, the user can push the *play* button and the application will start running collecting data as shown below:





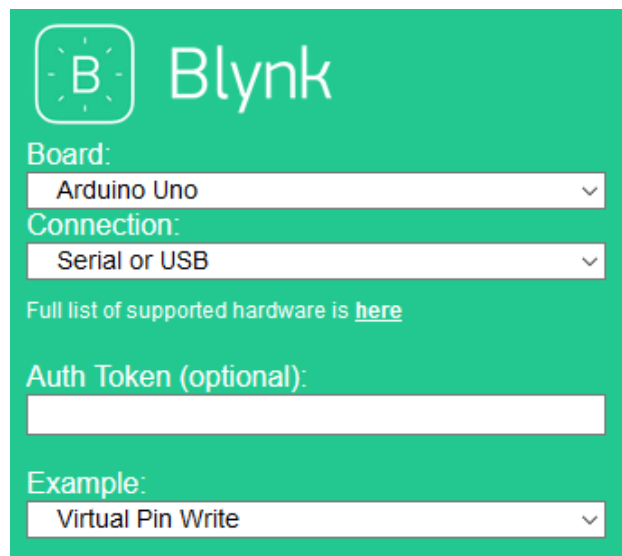
## Part 3: Programming the Arduino IDE

At this point, the Blynk App is already ready, but we have now to program the Arduino to establish a communication with the designed App.

In the following link, Blynk offers an interface to download multiple examples to program different boards.

<https://examples.blynk.cc/>

In the interface, the user can choose the board and the type of connection. One of the most basic and useful examples is the *Virtual Pin Write*. The Arduino code that will be used is based on this sketch.

The image shows a web interface for Blynk. At the top, there is a green header with the Blynk logo (a stylized 'B' in a square) and the word 'Blynk'. Below the header, there are two dropdown menus. The first is labeled 'Board:' and has 'Arduino Uno' selected. The second is labeled 'Connection:' and has 'Serial or USB' selected. Below these, there is a link that says 'Full list of supported hardware is [here](#)'. Then, there is a text input field labeled 'Auth Token (optional):'. At the bottom, there is another dropdown menu labeled 'Example:' with 'Virtual Pin Write' selected.

The code will be divided into 4 parts:

- Definition of variables
- Setup function
- Main function
- Blynk functions

### Definition of variables

In this part, the user has to include all the needs variables and objects.

The variables for the Blynk applications are the following ones:

- The Blynk library.
- A char variable with the *token* that will link the Arduino code with the mobile app.
- A *timer* object. This object will allow us to execute different functions with the desired time-frequency.

For our personal application:

- A digital port number to control an LED and its state variable.
- Three variables. Two of them will be used to store the measurement itself to calculate the average. The third one will be used as a counter to have the total number of measurements.

```

#include <BlynkSimpleStream.h>

// You should get Auth Token in the Blynk App.
// Go to the Project Settings (nut icon).
char auth[] = "XXXXXXXXXXXXXXXXXXXXXXXXXXXX";

BlynkTimer timer;

// Set your LED pins here
const int ledPin = 13;
int ledState = LOW;

// Set variables to calculate averages values
float sum_avg1 = 0, sum_avg2 = 0;
unsigned int count_avg = 0;

```

### Setup Function

The setup function configures the digital pin to control the LED. After that, configures the serial port communication with a baud rate of 9600 bps, and links the Blynk library with this serial port communication and the token.

Finally, it configures two timer functions. The first one, *myData*, will be configured to be executed each second, this function will read 2 values from the ADC and accumulate them in the *sum\_avg1* and *sum\_avg2* variables.

The second one, *myTimerEvent*, is configured to be executed every 5 seconds and will send the measured average values to the mobile App using the serial port.

```

void setup()
{
  // Debug console
  SwSerial.begin(9600);

  // Setup LED pin
  pinMode(ledPin, OUTPUT);
  digitalWrite(ledPin, ledState);

  // Blynk will work through Serial
  // Do not read or write this serial manually in your sketch
  Serial.begin(9600);
  Blynk.begin(Serial, auth);

  // Setup a function to be called every 5 seconds
  timer.setInterval(5000L, myTimerEvent);

  // Setup a function to acquire every 1 second
  timer.setInterval(1000L, myData);
}

```

### Main function

The main or loop function calls the Blynk program, which will be executed when we have an order from the mobile app. Otherwise, it will execute the two configured timers at the indicated time.

```
void loop()
{
  Blynk.run();
  timer.run(); // Initiates BlynkTimer
}
```

After these two functions, you can add all your own code without problems. Take into account two things:

- The serial port is used by the Blynk process so you can't use it for other purposes.
- Don't use delay functions, since it will generate problems with the timers.

### Blynk Function

The Blynk function coordinates four different functions:

- 2 previously defined timers
- 2 events:
  - o Event 1: When the connection between the mobile app and the Arduino is established
  - o Event 2: Every time the Button status changes on the mobile app (we click on it)

The function *myData()* explained before, reads each second two variables to accumulate their values.

```
// This function read from the ADC each second to calculate an average value
void myData()
{
  // Accumulate the ADC measurements each second,
  // to calculate latter and average value each 5 seconds
  sum_avg1 += analogRead(A0);
  sum_avg2 += analogRead(A1);
  // Count to calculate the number of accumulated measurements
  count_avg++;
}
```

*myTimerEvent()* function → This function is where the information is written in the corresponding *Virtual Pins* using the Blynk library.

```
// This function sends Arduino's up time every second to Virtual Pin (5).
// In the app, Widget's reading frequency should be set to PUSH. This means
// that you define how often to send data to Blynk App.
void myTimerEvent()
{
    // You can send any value at any time.
    // Please don't send more that 10 values per second.
    Blynk.virtualWrite(V5, millis() / 1000);
    Blynk.virtualWrite(V6, sum_avg1/count_avg);
    Blynk.virtualWrite(V7, sum_avg2/count_avg);
    sum_avg1 = 0;
    sum_avg2 = 0;
    count_avg = 0;
}
```

The event BLYNK\_WRITE(V1) is executed when the mobile App changes the value of the Virtual Pin V1. At this moment, we assign the new value to the *ledState* variable and write the LED digital pin.

```
// This function will be called every time Slider Widget
// in Blynk app writes values to the Virtual Pin V1
BLYNK_WRITE(V1)
{
    ledState = param.asInt(); // assigning incoming value from pin V1 to a variable
    digitalWrite(ledPin, ledState);
}
```

Finally, for robustness control, each time the Arduino and the phone App are reconnected, it's interesting to update the state of the LED.

```
// Every time we connect to the cloud...
BLYNK_CONNECTED() {
    // Request the latest state from the server
    Blynk.syncVirtual(V1);
}
```

Once the code is already programmed, upload this to the Arduino UNO. Furthermore, verify the Serial Port where the Arduino UNO is connected.

## Part 4. Connect the Arduino to the mobile App

At this point, the Arduino is connected to the Serial Port, sending the collected information. However, it needs to open a tunnel to bypass this information to the cloud.

For this, you can follow the next manual:

<http://help.blynk.cc/en/articles/605484-usb-serial>

- 1) In windows, open a terminal running the `cmd.exe` as administrator and go to the library script folder:

```
C:\...\libraries\Blynk\scripts
```

- 2) Run `blynk-ser.bat` file.

For example `blynk-ser.bat -c COM4` (where COM4 is port with your Arduino).  
And press "Enter".

This will open this tunnel, from this moment, the Arduino and the App will be connected.

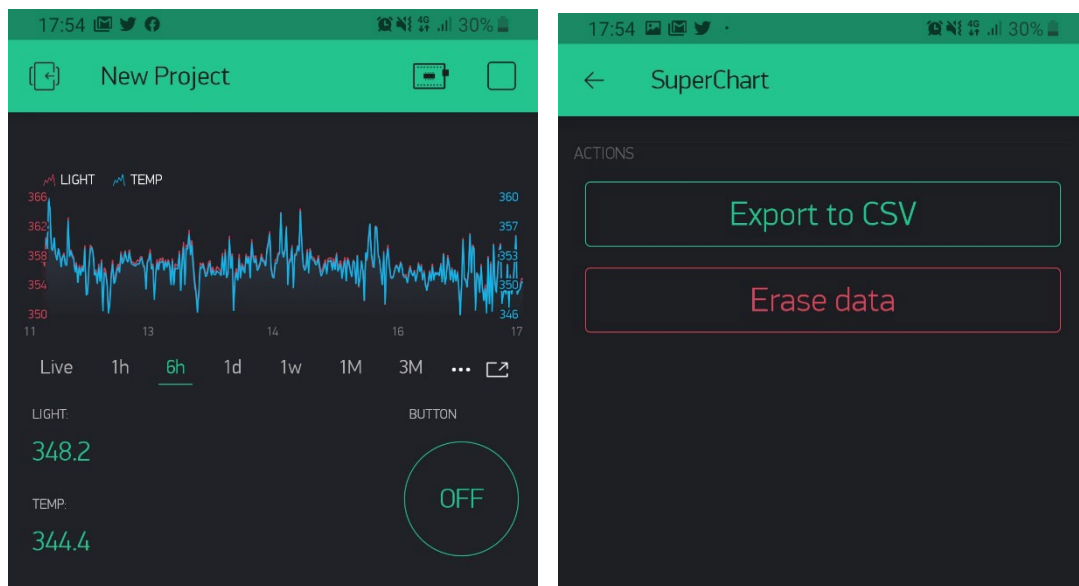
If you want to reprogram again the Arduino, the Arduino IDE may complain with "programmer is not responding". You need to close out the script before uploading new sketch. Remember that to close out a script you have to press *Ctrl + C*.

There is a video tutorial with all the process explained in the following link:

[https://youtu.be/fgzvoan\\_3\\_w](https://youtu.be/fgzvoan_3_w)

## Part 5. Exporting data to CSV

The *SuperChart* widget allows exporting its data with a time resolution of 60 seconds. To do this, press the "... " button and select *Export to CSV*. Blynk will send at your mail the CSV with the historical data.



The CSV file is like the one shown below, the first parameter is the data value and the second one is the time expressed in UNIX time format.

	A	
1	356.955555555555,1574852880000,0	
2	358.0333333333336,1574852940000,0	
3	356.4833333333334,1574853000000,0	
4	360.4333333333334,1574853060000,0	
5	358.8499999999997,1574853120000,0	
6	365.7666666666665,1574853180000,0	
7	364.15000000000003,1574853240000,0	
8	358.95,1574853300000,0	
9	358.5833333333333,1574853360000,0	
10	360.2333333333335,1574853420000,0	
11	358.05,1574853480000,0	
12	357.3333333333333,1574853540000,0	
13	358.9166666666667,1574853600000,0	
14	358.05,1574853660000,0	
15	356.7333333333335,1574853720000,0	
16	357.7999999999995,1574853780000,0	
17	358.3,1574853840000,0	
18	356.95,1574853900000,0	
19	356.8833333333334,1574853960000,0	
20	363.0666666666666,1574854020000,0	
21	358.1499999999999,1574854080000,0	

To work with the data in excel, the user can import the data using the CSV file to separate the values. Adding a new column, it's possible to calculate the data in a more classical.