NOTE: This page is a “Work In Progress”. So there might be mistakes, misprints, broken links and some TODO items listed. You are welcome to use it, just be aware of that.

Intro

This guide will help you to understand how to get started using Blynk and give you the most comprehensive overview of all the features.

If you want to jump straight into playing with Blynk, check out Getting Started.

GETTING STARTED

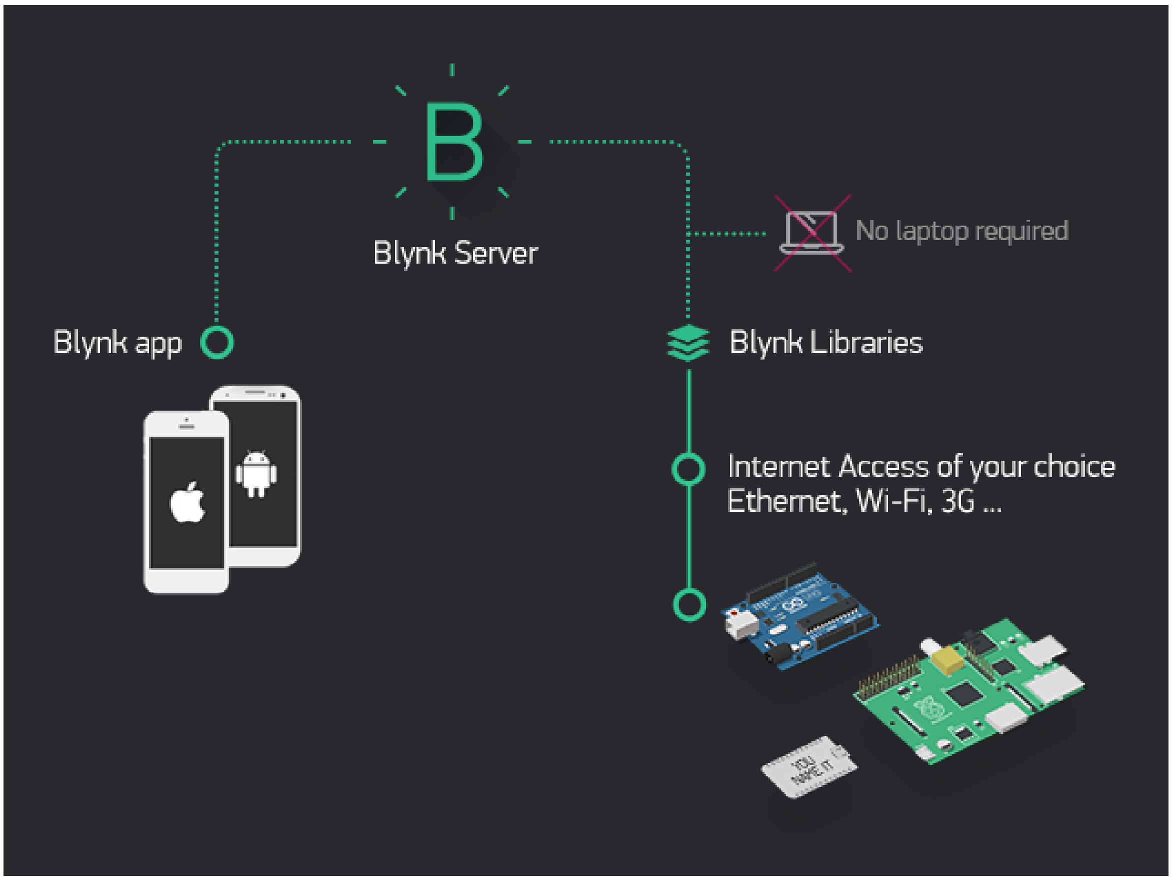
How Blynk Works

Blynk was designed for the Internet of Things. It can control hardware remotely, it can display sensor data, it can store data, analyze and do lot’s of other cool things.

There are three major components in the platform:

- **Blynk Apps** – allow to you create amazing interfaces for the projects from various Widgets we provide.
- **Blynk Server** is responsible for all the communications between the smartphone and hardware. You can use our Blynk Cloud or run your private Blynk server locally. It’s open-source, could easily handle thousands of devices and can be launched even on Raspberry Pi.
- **Blynk Libraries** for all the popular hardware platforms - enable communication with the server and process all the incoming and outcoming commands.

Now imagine: every time you press a Button in the Blynk app, the message travels to ~~space~~ Blynk Cloud, where it magically finds the way to your hardware. It works the same in the opposite direction and everything happens in a blynk of an eye.



What do I need to Blynk?

At this point you might be thinking: **"Ok, I want it. What do I need to start?"** – Just a couple of things, really:

Hardware.

We hope you already have an Arduino or someting similar. If not - come on, get one ASAP!

Blynk works over the Internet. It means that the hardware you choose should be able to connect to Internet. Some of the boards, like Arduino Uno will need an Ethernet or Wi-Fi Shield to communicate, others are already Internet-enabled: like ESP8266, Particle Photon or SparkFun ESP Thing. But even if you don't have a shield, you can connect over the USB to your laptop or desktop (it's a bit more complicated for newbie, but we got you covered). What's cool, is that the list of hardware that works with Blynk is really huge.

A Smartphone.

Blynk App is a well-designed interface builder. It works on both iOS and Android, so no holywars here, ok?

Downloads

Blynk Apps for iOS or Android

Available on the App Store   Google play

Blynk Library

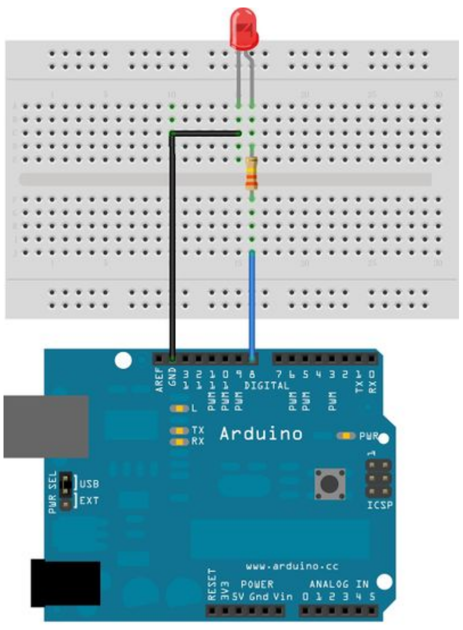
DOWNLOAD BLYNK LIBRARY

In case you don't know, or forgot how to install Arduino libraries check here.

Getting Started

Let's try to get you started in 5 minutes (reading doesn't counts!). We will try to switch an LED connected to your Arduino with the Smartphone.

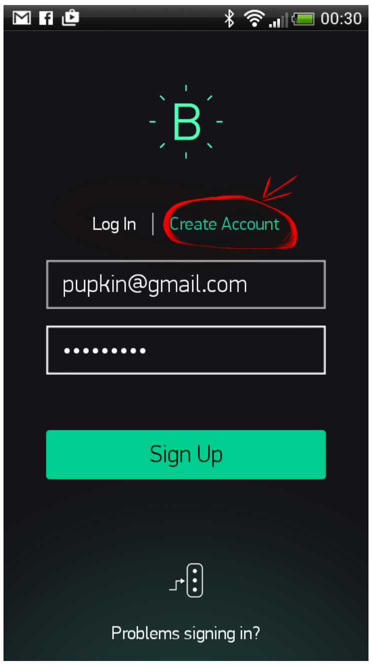
Connect an LED as shown here:



Getting started with application
1. Create Blynk Account

After you download Blynk app, you'll need to create a new Blynk account. It's not connected to our Forum or Kickstarter - just create a New Account.

Use a **real** e-mail address - it will be used later and you'll appreciate that.



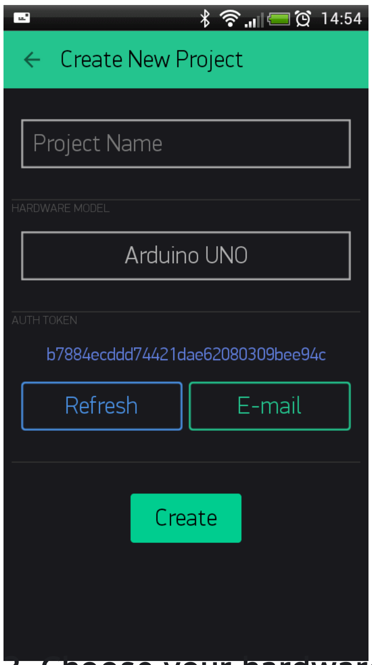
Why do I need to create account?

Account is needed to store your Projects and for you to have access to them from multiple devices. It's also a security measure.

You can always install your Private Blynk Server and have full control.

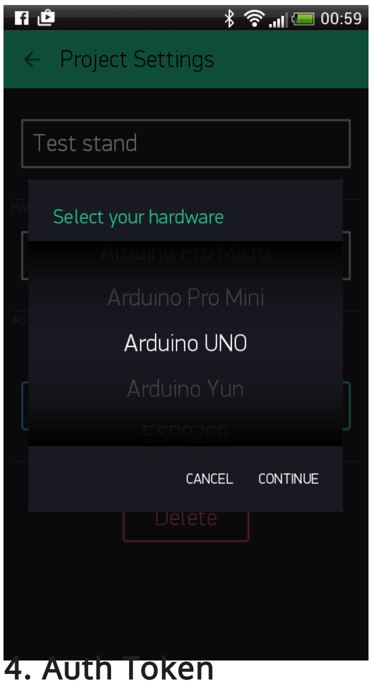
2. Create new project

After you successfully logged into your account, start by creating a new Project. Give it a name.



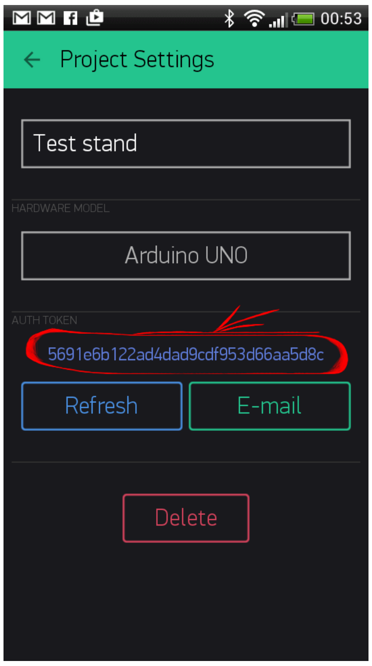
3. Choose your hardware

Select the hardware you will use. Check out how big is the list of supported hardware!



4. Auth Token

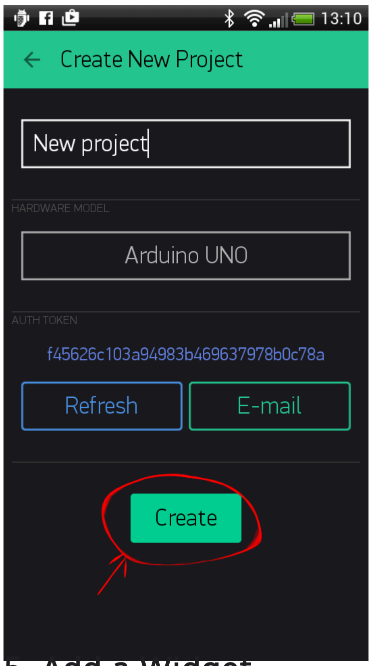
Auth Token is a uniqie identifier which is needed to connect your Arduino or other board to your smartphone. Every new project you create will have an Auth Token.



NOTE: Don't share your Auth Token with anyone, unless you want them to have access to your hardware.

It's very convenient to send it over E-mail. Press E-mail button and token will be sent to the e-mail address you used for registration. You can also tap on the Token itself and it will be copied to the clipboard.

Now press **"Create"** button.

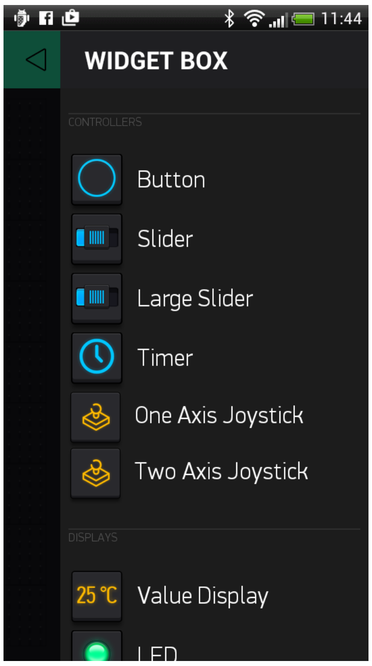


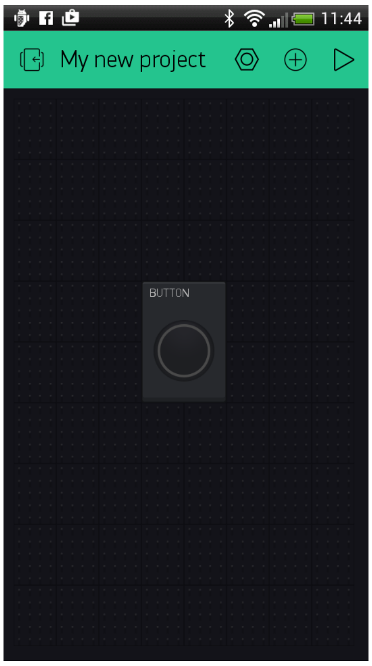
5. Add a Widget

Your project is empty, so let's add a Button to control our LED.

Tap anywhere on empty space - Whoa! you got inside a Widget Box! All the widgets are here. Now pick a Button.

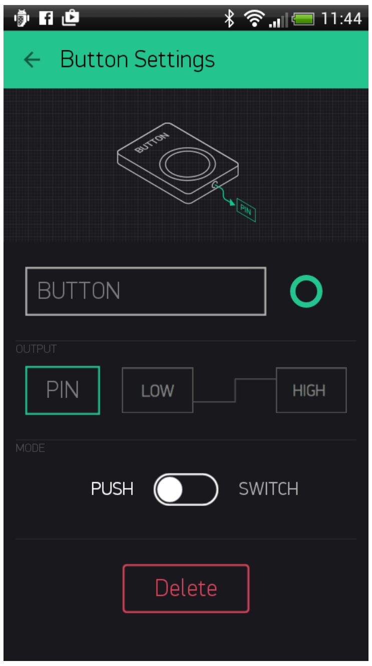
Widget Box



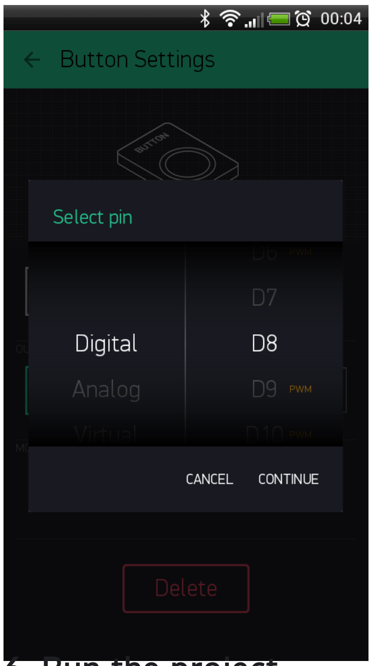


Drag-n-Drop Tap and hold the Widget to drag it to teh new position.

Widget Settings Each Widget has it’s own settings. Tap on the widget to get to them

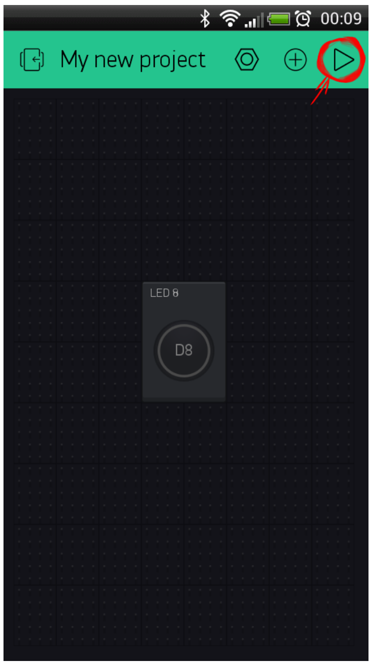


The most important parameter to set is **PIN** . List of pins reflects physical pins defined by your hardware. If your LED is connected to Digital Pin 8 - then select **D8** (**D** - stands for **D**igital).



6. Run the project

When you are done with the Settings - press **PLAY** button. This will switch you from EDIT mode to PLAY mode where you can interact with the hardware. Now you can't drag or set up Widgets, but if you need it: press **STOP** and get back to Edit Mode.

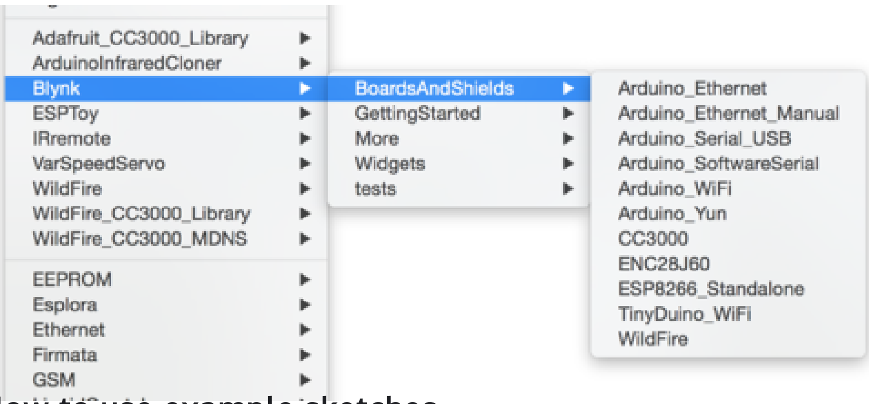


You may get a message "Arduino UNO is offline". Don't worry now. We'll get to that in a minute.

Getting started with hardware

Hope you have Blynk Library installed. If not - check here.

We've prepared example sketches that will get your microcomputer online. Open the example sketch according to the device or shield you are using.



How to use example sketches

Let's take a look at the simple sketch for Arduino UNO + Ethernet shield

```
#define BLYNK_PRINT Serial
#include <SPI.h>
#include <Ethernet.h>
#include <BlynkSimpleEthernet.h>

char auth[] = "YourAuthToken";

void setup()
{
  Serial.begin(9600); // See the connection status in Serial Monitor
  Blynk.begin(auth);  // Here your Arduino connects to the Blynk Cloud.
}

void loop()
{
  Blynk.run(); // All the Blynk Magic happens here...
}
```

Auth Token

In this sketch find this line:

```
char auth[] = "YourAuthToken";
```

This is the Auth Token that you've sent over e-mail recently. Please check your email - it should be there already. Copy it from e-mail and put inside curly brackets.

```
char auth[] = "f45626c103a94983b469637978b0c78a";
```

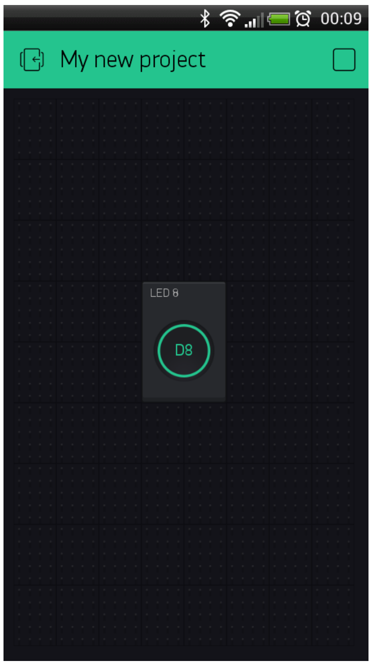
Upload sketch to the board and open Serial Terminal. Wait until you see something like this:

```
Blynk v.1.0.3
Your IP is 192.168.0.11
Connecting...
Blynk connected!
```

Congrats! You are all set! Now your Arduino is connected to Blynk Cloud!

Blynking

Go back to the Blynk app. Push the Button and turn the LED On and Off!



Check out other example sketches! Always feel free to experiment and combine different examples together!

For example, attach an LED to PWM-enabled Pin on your Arduino. Set the Slider Widget to control brightness of an LED. Just use the same steps described above.

Hardware set-ups

Arduino over USB (no shield)

If you don't have any shield and your hardware doesn't have any connectivity, you can still use Blynk – directly over USB :

1. Open Arduino Serial USB example and change Auth Token

```
#include <SoftwareSerial.h>
SoftwareSerial SwSerial(2, 3); // RX, TX
#define BLYNK_PRINT SwSerial
#include <BlynkSimpleSerial.h>

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

void setup()
{
  SwSerial.begin(9600);
  Blynk.begin(auth);
  // Default baud rate is 9600. You could specify it like this:
  //Blynk.begin(auth, 57600);
}

void loop()
{
  Blynk.run();
}
```

2. Run the script which is usually located in `/scripts` folder:

   - Windows: `My Documents\Arduino\libraries\Blynk\scripts`
   - Mac `User$/Documents/Arduino/libraries/Blynk/scripts`

On Windows:

Open cmd.exe

Write your path to blynk-ser.bat folder. For example:

```
cd C:\blynk-library-0.3.1\blynk-library-0.3.1\scripts
```

Run `blynk-ser.bat` file. For example : `blynk-ser.bat -c COM4` (where COM4 is port with your Arduino)

And press "Enter", press "Enter" and press "Enter"

On Linux and Mac:

Navigate to /scripts folder. For example:

```
cd User$/Documents/Arduino/libraries/Blynk/scripts
```

When inside this folder, run:

```
user:scripts User$ ./blynk-ser.sh
```

You may need to run it with `sudo`

```
user:scripts User$ sudo ./blynk-ser.sh
```

This is what you'll see in Terminal app on Mac (usbmodem address can be different):

```
[ Press Ctrl+C to exit ]
/dev/tty.usbmodem not found.
Select serial port [ /dev/tty.usbmodem1451 ]:
```

Copy the serial port address: `/dev/tty.usbmodem1451` and paste it back:

```
Select serial port [ /dev/tty.usbmodem1451 ]: /dev/tty.usbmodem1451
```

After you press Enter, you should see something similar:

```
Resetting device /dev/tty.usbmodem1451...
Connecting: GOPEN:/dev/tty.usbmodem1451,raw,echo=0,clocal=1,cs8,nonblock=1,
ixoff=0,ixon=0,ispeed=9600,ospeed=9600,crtscts=0 <-> openssl-connect:cloud.
blynk.cc:8441,cafile=/Users/.../server.crt,nodelay
```

```
2015/10/03 00:29:45 socat[30438.2046857984] N opening character device "/de
v/tty.usbmodem1451" for reading and writing
2015/10/03 00:29:45 socat[30438.2046857984] N opening connection to LEN=16
AF=2 45.55.195.102:8441
2015/10/03 00:29:45 socat[30438.2046857984] N successfully connected from l
ocal address LEN=16 AF=2 192.168.0.2:56821
2015/10/03 00:29:45 socat[30438.2046857984] N SSL connection using AES128-S
HA
2015/10/03 00:29:45 socat[30438.2046857984] N starting data transfer loop w
ith FDs [3,3] and [4,4]
```

NOTE: Arduino IDE may complain with "programmer is not responding". You need to terminate script before uploading new sketch..

Additional materials:

- Instructables: Control Arduino with Blynk over USB

Raspberry Pi

1. Connect your Raspberry Pi to the Internet and open it's console.
2. Install WiringPi
3. Download and build Blynk:

~~~
```bash
$ git clone https://github.com/blynkkk/blynk-library.git
$ cd blynk-library/linux
$ make clean all target=raspberry
```
~~~

1. Run Blynk:

   ```
   $ sudo ./blynk --token=YourAuthToken
   ```

2. To enable Blynk auto restart for Pi, find `*/etc/init.d/rc.local*` file and add there:

   ```
   /FULL_PATH_TO_LIB/blynk-library/linux/blynk --token=<Auth Token>
   ```

   For example:

   ```
   /home/pi/blynk-library/linux/blynk --token=<my token> &
   ```

We have also provided a build script. You can simply run it inside of the `linux` directory:

```
$ ./build.sh raspberry
```

Additional materials:

- Instructables: Blynk on Javascript for Raspberry Pi, Intel Edison and others
- Instructables: Use DHT11/DHT12 sensors with Raspberry Pi and Blynk
- Blynk Community Topic: How-To Raspberry Pi

- Video tutorial - Setting up Blynk and Raspberry Pi:

ESP8266 Standalone

You can run Blynk directly on the ESP8266!

Install the latest ESP8266 library for Arduino using this guide.

Example Sketch: ESP8266_Standalone

Additional materials:

- Instructables: ESP8266 ESP-12(Standalone)+ Blynk
- Instructables: ESP8266-12 standalone Blynk lm35 temperature sensor

Step-by-Step Tutorial in Russian language

Particle (formely Spark)

Blynk works with the whole family of Particle products: Core, Photon and Electron(soon)

TODO:

1. Open Particle Web IDE.
2. Go to the libraries.
3. Search for **Blynk** in the Community Libraries and click on it
4. Open SparkCore.ino example
5. Click "use this example"
6. Update your auth token, and flash the Particle!

On Android, you can scan this QR code from Blynk App and you'll get a ready-to-test project for **Particle Photon**. Just put your Auth Token to the SparkCore.ino example.

Additional materials:

- Particle core + DHT22

Blynk main operations

Virtual Pins

Blynk can control Digital and Analog IO Pins on you hardware directly. You don't even need to write code for it. It's great for blinking LEDs, but often it's just not enough…

We designed Virtual Pins to send **any** data from your microcontroller to the Blynk App and back.

It opens huge opportunities for you, because anything you plug in to your hardware will be able to talk to Blynk. With Virtual Pins you can send something from the App, process it on Arduino and then send it back to the smartphone based on any logic you want. You can trigger functions, read I2C devices, convert values, control any servo motor and so on.

Virtual Pins can be used to interface with external libraries (Servo, LCD and others) and implement custom functionality. The device may send data to the Widgets to the Virtual Pin like this:

```
Blynk.virtualWrite(pin, "abc");
Blynk.virtualWrite(pin, 123);
Blynk.virtualWrite(pin, 12.34);
```

Send data from app to hardware

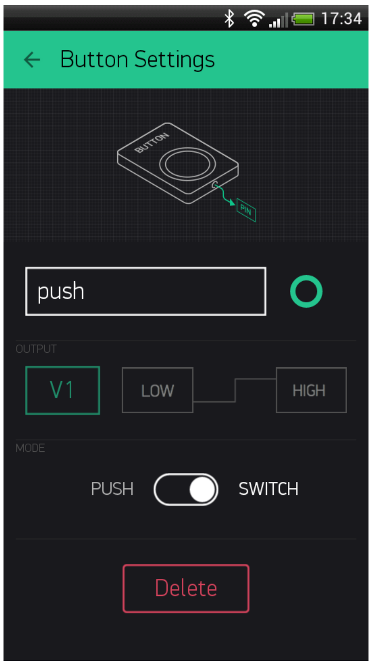
You can send any data from Widgets in the app to your hardware.

All Controller Widgets can send data to Virtual Pins on your hardware. For example, code below shows how to get values from the Button Widget in the App

```
BLYNK_WRITE(V1) //Button Widget is writing to pin V1
{
  int pinData = param.asInt();
}
```

When you press Button, Blynk App sends `1` On second click - it sends `0`

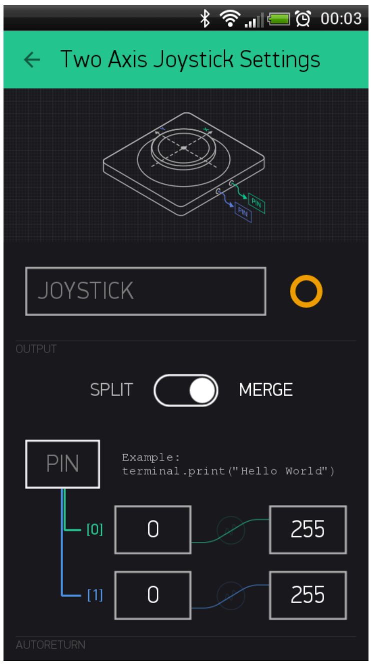
This is how Button Widget is set up:



Full example sketch: Get Data

Sending array from Widget

Some Widgets (e.g Joystick, zeRGBa) have more than one output.



This output can be written to Virtual Pin as an array of values. On the hardware side - you can get any element of the array [0,1,2…] by using:

```
BLYNK_WRITE(V1) // Widget WRITEs to Virtual Pin V1
{
  int x = param[0].asInt(); // getting first value
  int y = param[1].asInt(); // getting second value
  int z = param[N].asInt(); // getting N value
}
```

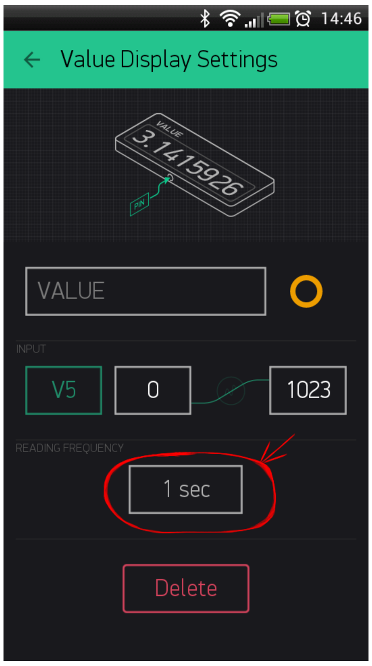
Sketch: JoystickTwoAxis

Get data from hardware

There are two ways of pushing data from your hardware to the Widgets in the app over Virtual Pins.

Perform requests by Widget

- Using Blynk built-in reading frequency while App is active by setting ‘Reading Frequency’ parameter to some interval:



VALUE

V5 0 1023

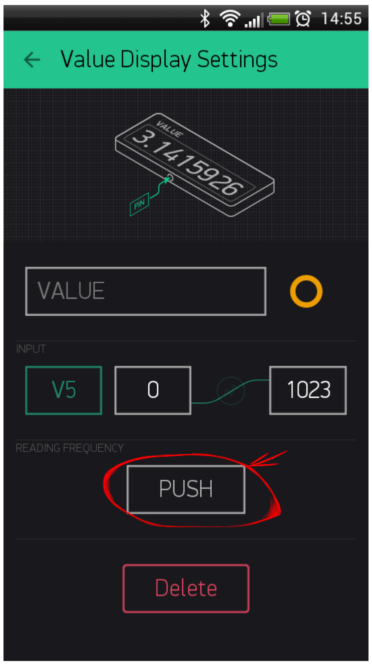
1 sec

```
BLYNK_READ(V5) // Widget in the app READs Virtal Pin V5 with the certain fr
equency
{
  // This command writes Arduino's uptime in seconds to Virtual Pin V5
  Blynk.virtualWrite(5, millis() / 1000);
}
```

Sketch: PushDataOnRequest

Pushing data from hardware

If you need to PUSH sensor or other data from your hardware to Widget, you can write any logic you want. Just set the frequency to PUSH mode



VALUE

INPUT

V5 0 1023

READING FREQUENCY

PUSH

Delete

We recommend sending data in intervals and avoiding Flood Error. For example, this SimpleTimer Library is an Arduino library for timed events. Please read instructions inside this example sketch for more details.

SimpleTimer is included in Blynk's library. Here is how it can work:

```
#include <SPI.h>
#include <Ethernet.h>
#include <BlynkSimpleEthernet.h>
#include <SimpleTimer.h> // here is the SimpleTimer library

char auth[] = "YourAuthToken"; // Put your token here

SimpleTimer timer; // Create a Timer object called "timer"!

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

  timer.setInterval(1000L, sendUptime); //  Here you set interval (1sec) an
d which function to call
}

void sendUptime()
{
  // This function sends Arduino up time every 1 second to Virtual Pin (V5)
  // In the app, Widget's reading frequency should be set to PUSH
  // You can send anything with any interval using this construction
  // Don't send more that 10 values per second

  Blynk.virtualWrite(V5, millis() / 1000);
}

void loop()
{
  Blynk.run(); // all the Blynk magic happens here
  timer.run(); // SimpleTimer is working
}
```

Sketch: PushData

Data types

The actual values are sent as Strings, so there is no practical limits on the data that can be sent. However, remember the limitations of the platform when dealing with numbers. For example, integer values on Arduino are 16-bit, allowing range -32768 to 32767. You can interpret incoming data as Integers, Floats, Doubles and Strings. Use these commands:

```
param.asInt();         // get an integer value
param.asFloat();     // get a float value
param.asDouble();    // get a double value
param.asStr();        // get a string value
```

You can also get the RAW data from the param buffer:

```
param.getBuffer()
param.getLength()
```

Limitations and Recommendations

- Don't put `Blynk.virtualWrite` and any other `Blynk.*` command inside `void loop()` - it will cause lot's of outgoing messages to our server and your connection will be terminated;
- We recommend calling functions with intervals. For example, this SimpleTimer Library is a simple library for timed events. Please read instructions inside this example sketch for more details;
- Avoid using long delays with `delay()` – it may cause connection breaks;
- If you send more than 10-100 (depends on hardware) values per second - you'll cause Flood Error* and connection to your hardware will be terminated;

Widgets

Widgets are interface modules. Each of them performs a specific input/ output function when communicating with the hardware.

There are 4 types of Widgets:

- **Controllers** - they send commands to hardware. Use them to control your stuff;
- **Displays** - used for various visualizations of data that comes from hardware to the smartphone;
- **Notifications** - are various widgets to send messages and notifications;
- **Others** - widgets that don't belong to any category;

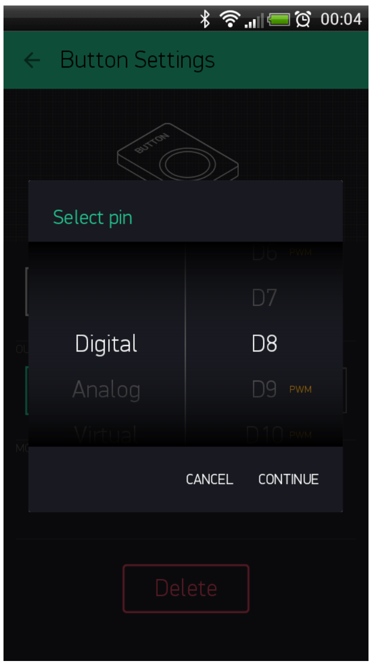
Each Widget has it's own settings.

Some of the Widgets (e.g. E-mail Widget) are used to enable some functionality and they don't have any settings.

Common Widget Settings

Pin Selector

This is one of the main parameters you need to set. It defines which pin to control or to read from.



Digital Pins - represent physical Digital IO pins on your hardware. PWM-enabled pins are marked with the `~` symbol

Digital Pins - represent physical Analog IO pins on your hardware

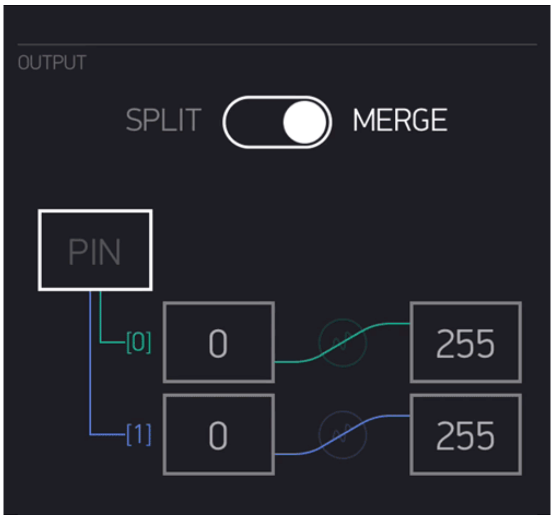
Virtual Pins - have no physical representation. They are used for any data transfer between Blynk App and your hardware. Read more about Virtual Pins here.

Data Mapping

Screenshot

SPLIT/MERGE

Some of the Widgets can send more than one value. And with this switch you can control how to send them.



- **SPLIT**: Each of the parameters is sent directly to the Pin on your hardware (e.g D7). You don't need to write any code.

  NOTE: In this mode you send multiple commands from one widget, which can reduce performance of your hardware.

  Example: If you have a Joystick Widget and it's set to D3 and D4, it will send 2 commands over the Internet:

```
digitalWrite(3, value);
digitalWrite(4, value);
```

- **MERGE**: When MERGE mode is selected, you are sending just 1 message, consisting of array of values. But you'll need to parse it on the hardware.

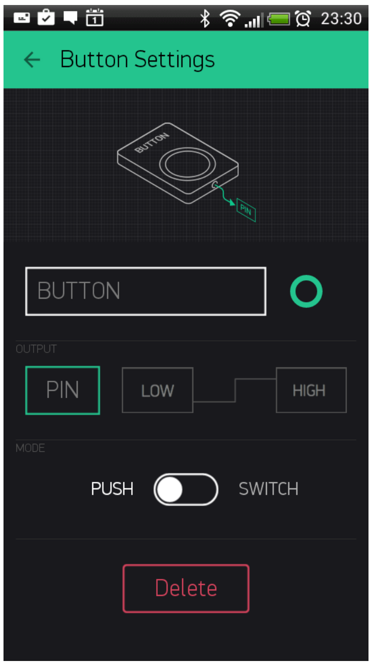
  This mode can be used with Virtual Pins only.

  Example: Add a zeRGBa Widget and set it to MERGE mode. Choose Virtual Pin V1

```
BLYNK_WRITE(V1) // There is a Widget that WRITEs data to V1
{
  int r = param[0].asInt(); // get a RED channel value
  int g = param[1].asInt(); // get a GREEN channel value
  int b = param[2].asInt(); // get a BLUE channel value
}
```

Controllers
Button

Works in push or switch modes. Allows to send 0/1 (LOW/HIGH) values.

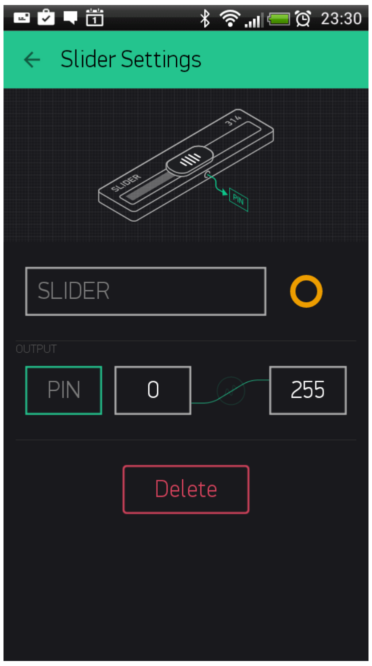

23:30
Button Settings

BUTTON
OUTPUT
PIN
LOW
HIGH
MODE
PUSH
SWITCH
Delete

Sketch: BlynkBlink

Slider

Similar to potentiometer. Allows to send values between MIN and MAX.



23:30

Slider Settings

SLIDER

OUTPUT

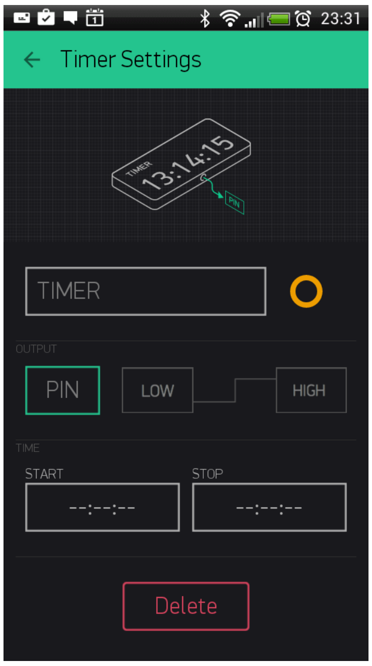
PIN 0 255

Delete

Sketch: BlynkBlink

Timer

Timer triggers actions at a specific time. Even if smartphone is offline. Start time sends 1 (HIGH). Stop time sends 0 (LOW).



23:31

Timer Settings

TIMER

OUTPUT

PIN LOW HIGH

TIME

START --:--:--

STOP --:--:--

Delete

Sketch: BlynkBlink

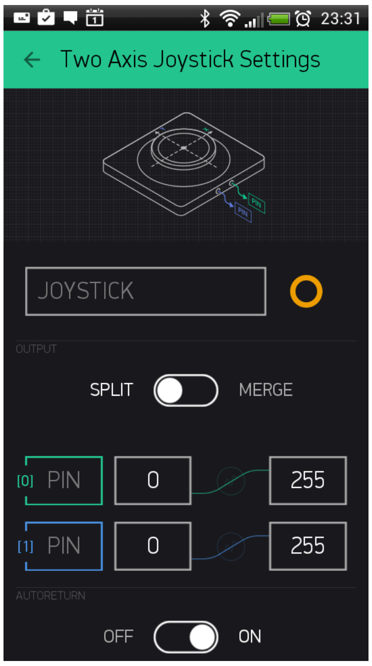
Joystick

Control servo movements in 4 directions

Settings:

- SPLIT/MERGE modes - read here
- **Rotate on Tilt**

When it's ON, Joystck will automatically rotate if you use your smartphone in landscape orientation - **Auto-Return** - When it's OFF, Joystick handle will not return back to center position. It will stay where you left it.


23:31

Two Axis Joystick Settings

JOYSTICK

OUTPUT

[✓ SPLIT] MERGE

[0] PIN 0 255

[1] PIN 0 255

AUTORETURN

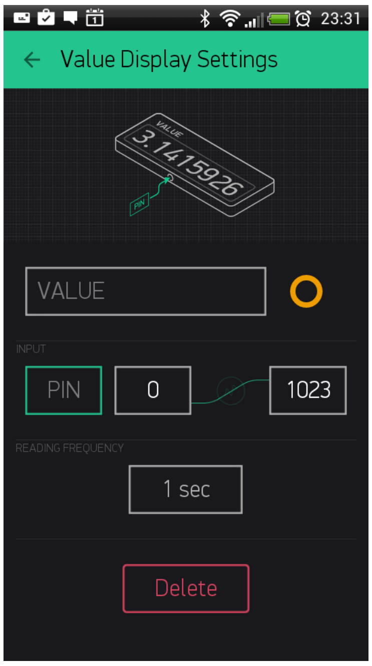
OFF [✓ ON]

Sketch: JoystickTwoAxis

Displays

Value Display

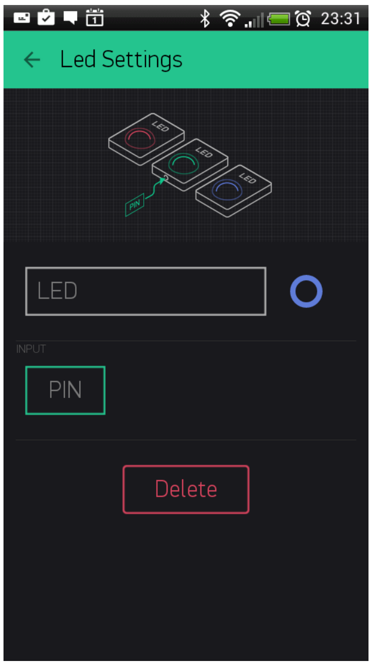
Displays incoming data from your sensors or Virtual Pins.



Sketch: BlynkBlink

LED

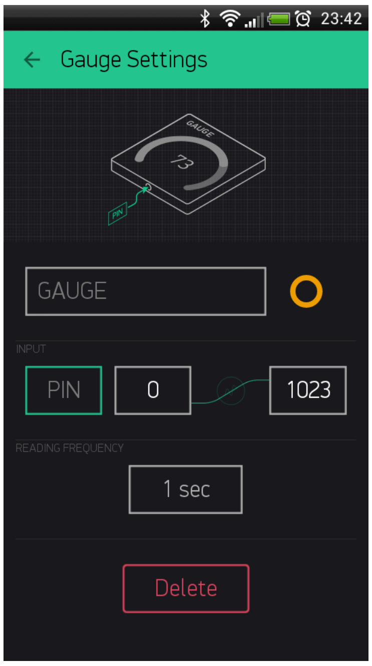
A simple LED for indication. TODO: describe 0-255 / 0-1 and fix in apps (make consistent)



Sketch: LED

Gauge

A great visual way to display incoming numeric values.



GAUGE

INPUT

PIN 0 1023

READING FREQUENCY

1 sec

Sketch: BlynkBlink

LCD

This is a regular 16x2 LCD display made in our secret facility in China.

SIMPLE / ADVANCED MODE

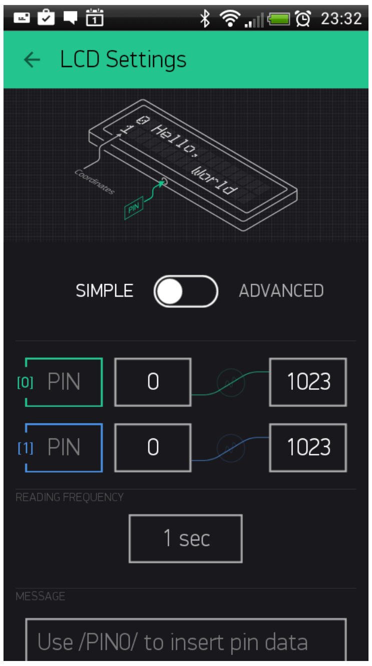
Commands

You need to use special commands with this widget:

```
lcd.print(x,y, "Your Message");
```

Where x is a symbol position (0-15), y is a line number (0 or 1),

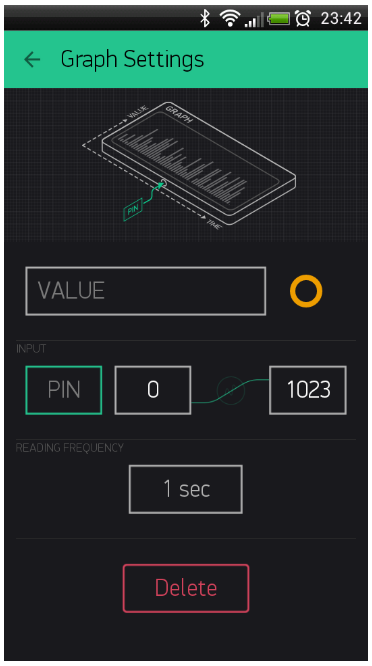
```
lcd.clear();
```



Sketch: LCD

Graph

Easily plot incoming data from your project in various designs.



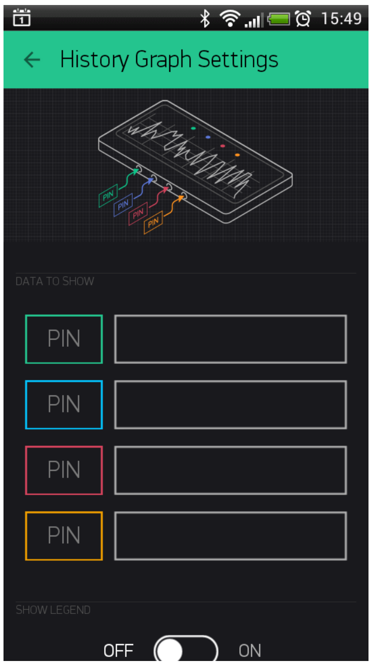
Sketch: BlynkBlink

History Graph

Allows you to see any data your hardware had sent to server previously. History graph has 3 granularities :

- Minute granularity - `1h` , `6h` ;
- Hour granularity - `1d` , `1w` ;
- Day granularity - `1m` , `3m` ;

This means that minimum graph update interval is 1 minute for `1h` and `6h` periods. 1 hour for `1d` and `1w` periods. 1 day for `1m` and `3m` periods.



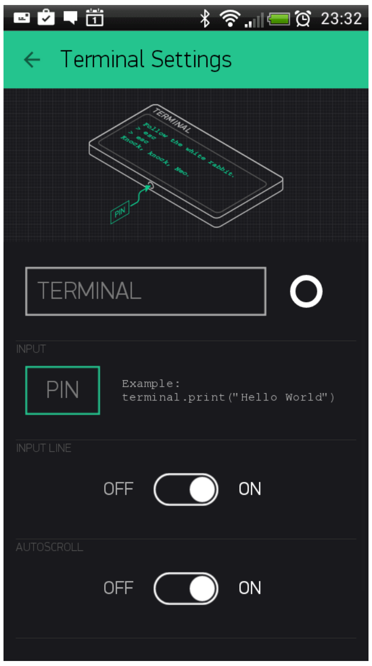
Sketch: PushData

Terminal

Display data from your hardware. Allows also to send any string to your hardware.

You need to use special commands with this widget:

```
terminal.print();
terminal.flush();
```


Terminal Settings

TERMINAL

INPUT

PIN

```
Example:
terminal.print("Hello World")
```

INPUT LINE

OFF ON

AUTOSCROLL

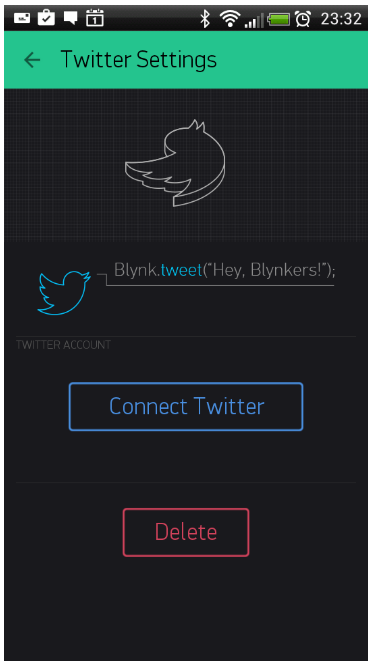
OFF ON

Sketch: Terminal

Notifications

Twitter

Twitter widget connects your Twitter account to Blynk and allows you to send Tweets from your hardware.


Twitter Settings

Blynk.tweet("Hey, Blynkers!");

TWITTER ACCOUNT

Example code:

```
Blynk.tweet("Hey, Blynkers! My Arduino can tweet now!");
```

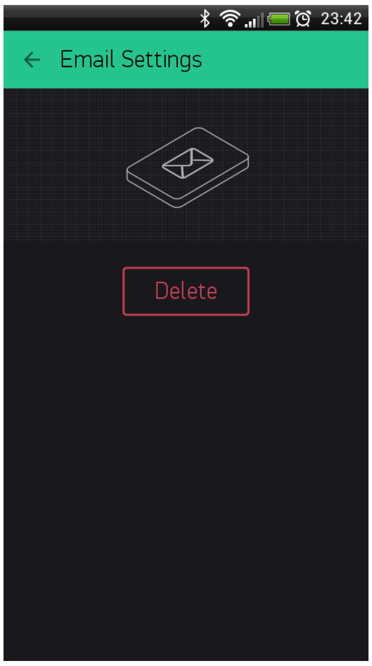
Limitations :

- you cant' send 2 tweets with same message (it's Twitter policy)
- only 1 tweet per minute is allowed

Sketch: Twitter

Email

Email widget allows you to send email from your hardware to any address.



Example code:

```
Blynk.email("my_email@example.com", "Title", "Body");
```

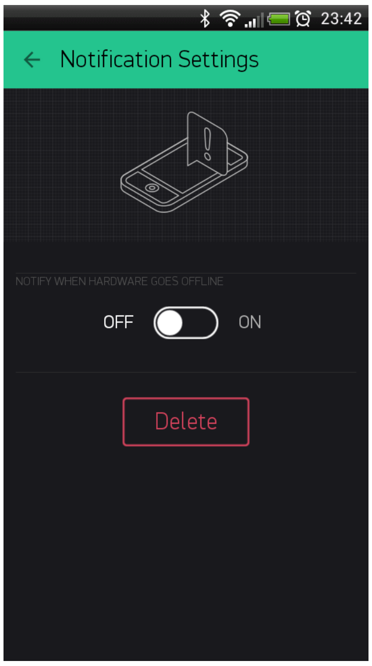
Limitations :

- only 1 email per minute is allowed

Sketch: Email

Push Notifications

Push Notification widget allows you to send push notification from your hardware to your device.



Example code:

```
Blynk.notify("Hey, Blynkers! My hardware can push now!");
```

Limitations :

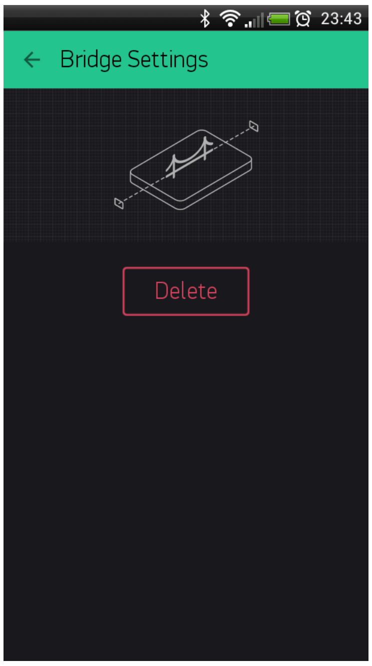
- maximum allowed body length is 255 symbols.
- only 1 notification per minute is allowed

Sketch: PushNotification

Other

Bridge

Bridge can be used for Device-to-Device communication. You can send digital/analog/virtual write commands from one device to another, knowing it's auth token.



Example code:

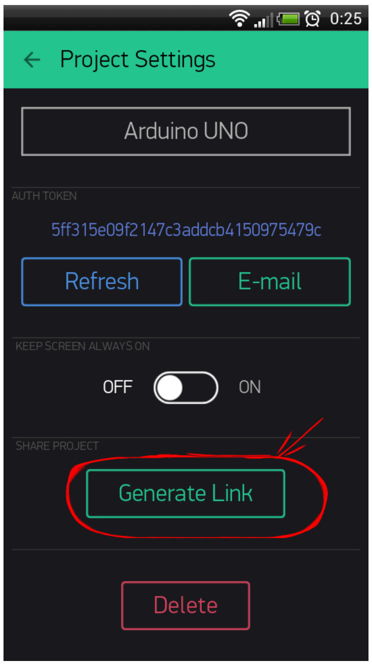
```
WidgetBridge bridge1(1); //Bridge widget on virtual pin 1
...
void setup() {
    bridge1.setAuthToken("OtherDeviceAuthToken");
    bridge1.digitalWrite(9, HIGH);
    bridge1.analogWrite(10, 123);
    bridge1.virtualWrite(1, "hello");
}
```

Sketch: Bridge

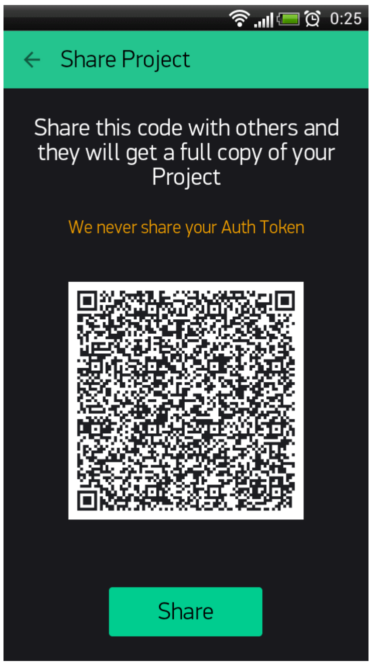
Sharing

Dashboard sharing without Auth Token

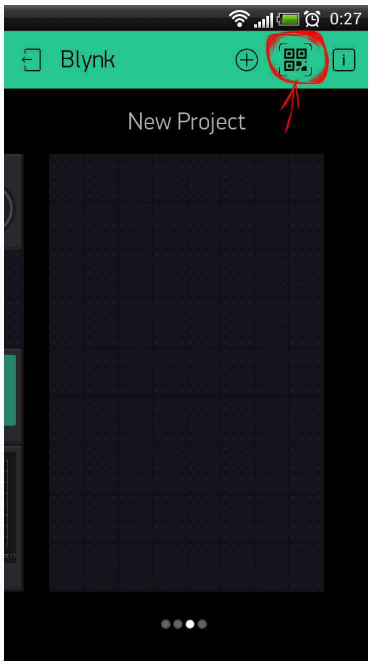
In case you want to share your dashboard without Auth token and thus without access to your device (for example it could be ideal case for instructable) in project setting go to **Generate Link** button.



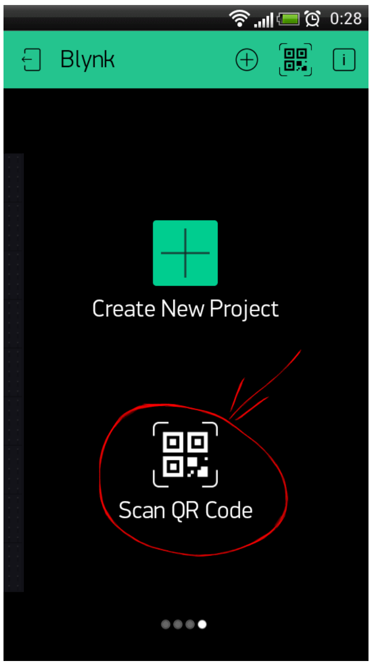
It will generate QR code you could share with anyone you want.


0:25
Share Project
Share this code with others and they will get a full copy of your Project
We never share your Auth Token
Share

For scanning - find QR button on Dashboards List screen.



Or on creating new Dashboard screen.



Supported Hardware

Platforms

- **Arduino** (https://github.com/blynkkk/blynk-library)
    - Uno, Duemilanove (select “UNO” in the list, not the “Due”)
    - Nano, Mini, Pro Mini, Pro Micro
    - Mega
    - YÚN (onboard WiFi and Ethernet, via Bridge)
    - Due - limited support

- **Arduino-like**
    - ESP8266 (running standalone, using https://github.com/esp8266/Arduino)
    - Intel Edison
    - Intel Galileo
    - LinkIt ONE
    - Wicked WildFire (CC3000)
    - TinyCircuits TinyDuino (CC3000)
    - LightBlue Bean *, soon*
- **Energia**
    - Texas Instruments
        - CC3200-LaunchXL
        - Tiva C Connected LaunchPad
        - Stellaris LM4F120 LaunchPad
    - RedBearLab (CC3200, WiFi Mini)
- **Particle** (formerly Spark: https://github.com/vshymanskyy/blynk-library-spark)
    - Core
    - Photon
- **ARM mbed** (soon)
    - Seeed Tiny BLE
    - RedBearLab BLE Nano
    - BBC Micro:bit
- **JavaScript** (Node.js, Espruino) (https://www.npmjs.com/package/blynk-library)
    - Regular PC with Linux / Windows / OS X
    - Raspberry Pi (Banana Pi, Orange Pi, …)
    - BeagleBone Black
    - PandaBoard
    - CubieBoard
    - pcDuino
    - Tessel 2
    - Intel Edison
    - Intel Galileo
    - VoCore (OpenWRT + Espruino package)
    - Espruino Pico
    - …
- **Python** (MicroPython)
    - WiPy

Arduino connection types

- USB (Serial), connected to your laptop or desktop
- Official Ethernet shield (W5100)
- Official Arduino WiFi shield
- Adafruit Bluefruit LE (nRF8001 Breakout, BLE 4.0) *, soon*
- Adafruit CC3000 WiFi Breakout / Shield
- ENC28J60 - based boards
- ESP8266 as WiFi modem (running original firmware)
- SeeedStudio Ethernet Shield V2.0 (W5200)
- RN-XV WiFly

- You can implement your own connection type easily (see this example)!

Made by Community

- WIZnet-W5500-EVB
- LabVIEW
- Node-RED (can be used as bridge to HTTP, TCP, UDP, MQTT, XMPP, IRC, OSC…)

Troubleshooting

Connection

If you experience connection problems, follow these steps:

1. Check that your hardware, wires, cables and power supply are good quality, not harmed or damaged, etc.
   Use high power USB cables and USB ports.
2. Check your wiring using the examples (TCP/HTTP Client or similar) **provided with selected shield and hardware**. Once you have some understanding how to configure connection, it's much easier to use Blynk.
3. Try running command `telnet cloud.blynk.cc 8442` from your PC, connected to the same network as your board. You should see something like `Connected to cloud.blynk.cc.`.
4. Try running Blynk default examples for your platform **without modifications** to see if it is working.
   - Read carefully the example comments and explanations
   - Check that your token is valid (copied from the App and **doesn't contain spaces, etc.**)
   - If it doesn't work, try looking into serial debug prints.
5. Done! Add your modifications and functionality. Enjoy Blynk!

Delay

Your application might be calling a delay() function or sleeps/cycles for a long time inside of the loop(), like this:

```
void loop()
{
  ...
  delay(1000);
  other_long_operation();
  ...
  Blynk.run();
}
```

You should be aware that this can degrade performance of Blynk, or cause connection drops.

Note: This also applies to the BLYNK_READ & BLYNK_WRITE handlers!

If you need periodic actions, consider using some timer library, like shown in this example.

Flood Error

Your application may cause an enormous load on our server, please try avoiding sending data too fast.

For example, in this situation Blynk.run() will quickly finish processing incoming messages, and then new value is sent to the server immediately, causing high traffic:

```
void loop()
{
  Blynk.virtualWrite(1, value);
  Blynk.run();
}
```

You might be thinking about adding a delay(), but this creates another trouble.

If you need periodic actions, consider using some timer library, like shown in this example.

Enable debug

To enable debug prints on the default Serial, add on the top of your sketch **(should be the first line)**:

```
#define BLYNK_DEBUG // Optional, this enables lots of prints
#define BLYNK_PRINT Serial
```

And enable serial in setup():

```
Serial.begin(9600);
```

You can also use spare Hardware serial ports or SoftwareSerial for debug output (you will need an adapter to connect to it with your PC).

Note: enabling debug mode will slowdown your hardware processing speed up to 10 times.

Security

Blynk server has 3 ports open for different security levels.

- **8441** - SSL/TLS connection for hardware
- **8442** - plain TCP connection for hardware (no security)
- **8443** - mutual authentication (mutual SSL) connection for Mobile Apps

Hardware may select to connect to 8441 or 8442, depending on it’s capabilities.

Use SSL gateway

Most platforms are not capable to handle SSL, so they connect to 8442. However, our gateway script can be used to add SSL security layer to communication.

```
./blynk-ser.sh -f SSL
```

This will forward all hardware connections from 8441 port to the server via SSL gateway. You can run this script on your Raspberry Pi, desktop computer, or even directly on your router!

Note: when using your own server, you should overwrite the bundled server.crt certificate, or specify it to the script using `--cert` switch:

```
./blynk-ser.sh -f SSL -s <server ip> -p 8441 --cert=<certificate>.crt
```

Flag `-f SSL` is enabled by default for USB communication so you don't have to explicit declare it.

Note: SSL is supported by the gateway only on Linux/OSX for now

If you want to skip SSL, and connect to TCP, you can also do that:

```
./blynk-ser.sh -t TCP
```

Use Local Blynk Server

In order to gain maximum security you could install Blynk server locally and restrict access to your network, so nobody except you could access it.

Blynk server

Blynk Server is an Open Source Netty-based Java server, responsible for forwarding messages between Blynk mobile application and various microcontroller boards (i.e. Arduino, Raspberry Pi. etc).

Download latest server build:

DOWNLOAD BLYNK SERVER

Requirements

Java 8 required. (OpenJDK, Oracle).

How to install Java.

How to run local Blynk Server

By default, mobile application use 8443 port and is based on SSL/TLS sockets. Default hardware port is 8442 and is based on plain TCP/IP sockets.

Quick Local Server launch

1. Make sure you are using Java 8

```
java -version
Output: java version "1.8.0_40"
```

2. Navigate to the folder where `.jar` file is. For example:

```
cd Users/User/Blynk/Server
```

3. Launch Blynk server.

```
java -jar server-0.11.0.jar -dataFolder /path
```

   By default server uses these ports:

   - Hardware: 8442
   - Blynk App: 8443 (SSL port)

That's it! Your Blynk Server is up and running

You won't see any output, because all the logging is done within same folder in `./logs/blynk.log` file.

Launch Blynk Server on Raspberry Pi

1. Login to Raspberry Pi via ssh;
2. Install Java 8:

```
sudo apt-get install oracle-java8-jdk
```

3. Make sure you are using Java 8:

```
java -version
Output: java version "1.8.0_40"
```

4. Download Blynk server .jar file (or manually copy it to raspberry via ssh and scp command):

```
wget "https://github.com/blynkkk/blynk-server/releases/download/v0.11.0/se
rver-0.11.0.jar"
```

5. Run the server on default `hardware port 8442` and default `application port 8443` (SSL port)

```
java -jar server-0.11.0.jar -dataFolder /home/pi/Blynk
```

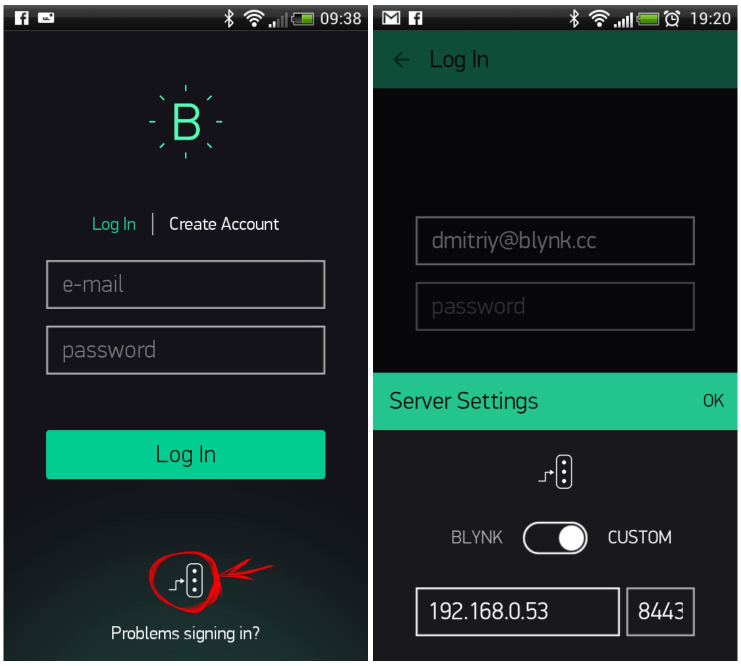
That's it! You won't see any output because all the logging is done within same folder in `./logs/blynk.log file.`

- To enable server auto-restart, find /etc/init.d/rc.local file and add a line:

```
java -jar /home/pi/server-0.11.0.jar -dataFolder /home/pi/Blynk &
```

App and sketch changes

To connect Blynk App to your local server you need to set up a custom server during login. Set up IP address and PORT (should be 8443 by default)



- If you are using Ethernet connection, in the Ethernet example sketch change:

```
Blynk.begin(auth);
```

to

```
Blynk.begin(auth, "your_host"); //or
Blynk.begin(auth, IPAddress(xxx,xxx,xxx,xxx));
```

- If you are connecting over WiFi, make edits in the WiFi example sketch. Change

```
Blynk.begin(auth, SSID, pass));
```

to

```
Blynk.begin(auth, SSID, pass, "your_host"); //or
Blynk.begin(auth, SSID, pass, IPAddress(XXX,XXX,XXX,XXX));
```

- If you are connected over USB when running `blynk-ser.sh` provide `-s` option with address of your local server:

```
./blynk-ser.sh -s you_host_or_IP
```

Advanced local server setup

If you need more flexibility, you can extend server with more options by creating server.properties file in same folder as server.jar. Example could be found here. server.properties options:

- Application port

  ```
  app.ssl.port=8443
  ```

- For simplicity Blynk already provides server jar with build-in SSL certificates, so you have working server out of the box via SSL/TLS sockets. But as certificate and it's private key is in public, this is very unsecure. In order to fix that, you will need to provide your own certificates and change properties with path to your certificates, private key and it's password.

  See how to generate self-signed certificates here

  Points to certificate and key that placed in same folder as running jar:

  ```
  server.ssl.cert=./server_embedded.crt
  server.ssl.key=./server_embedded.pem
  server.ssl.key.pass=pupkin123
  ```

- Hardware port

  ```
  hardware.default.port=8442
  ```

- User profiles folder. Folder in which all users profiles will be stored. By default System.getProperty("java.io.tmpdir")/blynk used. Will be created if not exists

  data.folder=/tmp/blynk

- Folder for all application logs. Will be created if it doesn't exist

  ```
  logs.folder=./logs
  ```

- Log debug level. Possible values: trace|debug|info|error. Defines how precise logging will be. From left to right -> maximum logging to minimum

  ```
  log.level=trace
  ```

- Maximum allowed number of user dashboards.

  ```
  user.dashboard.max.limit=10
  ```

- 100 Req/sec rate limit per user.

  ```
  user.message.quota.limit=100
  ```

- In case user exceeds quota limit - response error returned only once in specified period (in Millis).

  ```
  user.message.quota.limit.exceeded.warning.period=60000
  ```

- Maximum allowed user profile size. In Kb's.

  ```
  user.profile.max.size=128
  ```

- Maximum allowed number of notification queue. Queue responsible for processing email, pushes, twits sending. Because of performance issue - those queue is processed in separate thread, this is required due to blocking nature of all above operations. Usually limit shouldn't be reached

  ```
  notifications.queue.limit=10000
  ```

- Period for flushing all user DB to disk. In millis

  ```
  profile.save.worker.period=60000
  ```

- Specifies maximum period of time when application socket could be idle. After which socket will be closed due to non activity. In seconds. Leave it empty for infinity timeout

  ```
  app.socket.idle.timeout=600
  ```

- Specifies maximum period of time when hardware socket could be idle. After which socket will be closed due to non activity. In seconds. Leave it empty for infinity timeout

  ```
  hard.socket.idle.timeout=15
  ```

- Mostly required for local servers setup in case user want to log raw data in CSV format. See raw data section for more info.

  ```
  enable.raw.data.store=true
  ```

- Comma separated list of users allowed to create accounts. Leave it empty if no restriction required.

  ```
  allowed.users.list=allowed1@gmail.com,allowed2@gmail.com
  ```

Enabling mail on Local server

In order to enable mail notifications on Local server you need to provide own mail credentials. To do that you need to create file "mail.properties" within same folder where server.jar is. Mail properties:

```
mail.smtp.auth=true
mail.smtp.starttls.enable=true
mail.smtp.host=smtp.gmail.com
mail.smtp.port=587
mail.smtp.username=YOUR_EMAIL_HERE
mail.smtp.password=YOUR_EMAIL_PASS_HERE
```

See example here.

NOTE: you'll need to setup Gmail to allow less secured applications. Go here and then click "Allow less secure apps".

Raw data storage

By default, raw data storage is enabled. Every 'write' `Blynk.virtualWrite()` command will be stored on disk. The default path is "data" folder within data.folder property of server properties.

File name format is:

```
dashBoardId_pin.csv
```

Example:

```
data/1_v5.csv
```

It means that raw data from Virtual Pin 5 of dashboard with ID 1 is stored in 1_v5.csv

Data format is:

```
value,timestamp
```

Example:

```
10,1438022081332
```

Where 10 - pin value, and 1438022081332 - difference(in milliseconds) between current time and midnight, January 1, 1970 UTC

Raw data files are rotated every day and gzipped.

WARNING: It will be changed soon.

Generate SSL certificates

- Create key

  ```
    openssl genrsa -out server.key 2048
  ```

- Create new cert request

  ```
    openssl req -new -out server.csr -key server.key
  ```

- Generate self-signed request

  ```
    openssl x509 -req -days 1825 -in server.csr -signkey server.key -out serv
  er.crt
  ```

- Convert server.key to PKCS#8 private key file in PEM format

  ```
    openssl pkcs8 -topk8 -inform PEM -outform PEM -in server.key -out server.
  pem
  ```

WARNING: in case you connect hardware via USB script you have to provide an option '-s' pointing to "common name" (hostname) you did specified during certificate generation.

As output you'll retrieve server.crt and server.pem files that you need to provide for server.ssl properties.

Install Java on Ubuntu

```
sudo apt-add-repository ppa:webupd8team/java
sudo apt-get update
```

```
sudo apt-get install oracle-java8-installer
```

Behind WiFi router

If you want to run Blynk server behind WiFi-router and want it to be accessible from the Internet, you have to add port-forwarding rule on your router. This is required in order to forward all of the requests that come to the router within the local network to Blynk server.

Blynk Firmware

Configuration

The simplest way to configure Blynk is to call `Blynk.begin()`:

```
Blynk.begin(auth, ...);
```

It has various parameters for different hardware, depending on the type of connection you use. Follow the example sketches for your board.

You can also set up shields (WiFi, Ethernet) manually, and then call:

```
Blynk.config(auth);
```

or

```
Blynk.config(auth, server, port);
```

For WiFi connections, you can use a `connectWiFi` (just for convenience).

```
Blynk.connectWiFi(ssid, pass)
```

To connect to open WiFi networks, set pass to an empty string (“”).

Connection management

There are several functions to help with connection management:

```
# This functions will keep connecting to Blynk server. Default timeout is 3
0 seconds
bool result = Blynk.connect();
bool result = Blynk.connect(timeout);
```

To disconnect from Blynk server, use:

```
Blynk.disconnect();
```

To get the status of connection to Blynk Server use:

```
bool result = Blynk.connected();
```

Note: Just after `Blynk.begin(...)` or `Blynk.config(...)`, Blynk is not yet connected to the server. It will try to connect when it reaches first `Blynk.run()` or `Blynk.connect()` call.

If you want to skip connecting to the server, just call `Blynk.disconnect()` right after configuration.

If your shield/connection type is not supported yet - you can craft it yourself easily! Here is an example.

Blynk.run()

This function should be called frequently to process incoming commands and perform housekeeping of Blynk connection. It is usually called in `void loop() {}`.

You can initiate it in other places, unless you run out of heap memory (in the cascaded functions with local memory). For example, it is not recommended to call `Blynk.run()` inside of the `BLYNK_READ` and `BLYNK_WRITE` functions on low-RAM devices.

Digital & Analog pins control

The library can perform basic pin IO (input-output) operations out-of-the-box:

```
digitalRead
digitalWrite
analogRead
analogWrite (PWM or Analog signal depending on the platform)
```

So need to write code for simple things like LED, Relay control and analog sensors.

Virtual pins control

BLYNK_WRITE(vPIN)

BLYNK_READ(vPIN)

BLYNK_WRITE_DEFAULT()

BLYNK_READ_DEFAULT()

Blynk.virtualWrite();

You can send all the formats of data to Virtual Pins

```
Blynk.virtualWrite(pin, "abc");
Blynk.virtualWrite(pin, 123);
Blynk.virtualWrite(pin, 12.34);
```

Debugging

BLYNK_PRINT

TODO:Description how to use

`#define BLYNK_PRINT`

TODO: Output example

BLYNK_DEBUG

TODO: Description how to use

`#define BLYNK_DEBUG`

TODO: Output example

BLYNK_LOG()

To enable debug prints on the default Serial, add on the top of your sketch **(should be the first line)**:

```
#define BLYNK_DEBUG // Optional, this enables lots of prints
#define BLYNK_PRINT Serial
```

And enable Serial Output in setup():

```
Serial.begin(9600);
```

Open Serial Monitor and you'll see: TODO: Output example

You can also use spare Hardware serial ports or SoftwareSerial for debug output (you will need an adapter to connect to it with your PC).

WARNING: Enabling Debug mode will slow down your hardware processing speed up to 10 times!

Blynk Amendments

Tell every maker about Blynk

No pressure. Just do it. Now.

Make your idea work without Blynk

Blynk can be easily integrated in almost any project. But before that - make it work **without** it. After you are sure that you can get all the sensor data or can control things from the code – integrate Blynk and make it even more awesome.

Use search

We are always happy to chat and help, but remember - every time you ask the question that was answered many many times before that, Blynk Team is not building a new widget or new cool feature. So: - google before asking - use search on our forum, it works really well - check Instructables

Always wrap your code

Though shalt not post code without `wrapping it`

FAQ

- I backed Blynk on Kickstarter. Where are my widgets and why the app is free?

  App is free because otherwise you would have to pay to download it. This is how AppStore and Google Play works. Current Blynk release has a limited amount of widgets. We decided to make them free for everyone until we implement store. After that, every widget will be paid. However every backer will get them for free (according to their pledge).

- What is Blynk Cloud?

  Blynk Cloud is a open-source software written on Java using plain TCP/IP and secured TCP/IP (for hardware that supports it) sockets and running on our server. Blynk iOS and Android apps connect to Blynk Cloud by default. Access is free for every Blynk user. We also provide a Private Server distribution for those who want to install it locally.

- How much access to Cloud Blynk Server cost?

  It is free for every Blynk user.

- Can I run Blynk server locally?

  Yes. Those of you, who want extra security or don't have internet connection, can install Local Blynk Server and run it in your own local network. Blynk Server is Open-Source and it takes less than few seconds to deploy. All the instructions and files are here.

- What are the requirements to run Private Blynk Server?

  To run Private Blynk Server, all you need is Java Runtime Environment.

- Can I run Blynk server on Raspberry Pi?

  Yes, surely! Here is instruction.

- Does Blynk app work over Bluetooth?

  No. But it's planned for next releases.

- Does Blynk support Ethernet / Wi-FI / UART?

  Yes, all of them. See full list of supported hardware and shields.

- I don't have any shield. Can I use Blynk with my computer?

  Yes, you can use Blynk just with a USB cable. There is a step-by-step instruction on how to do it.

- Can Blynk handle multiple Arduinos?

  Yes. There 2 ways right now :

  - you may use same Auth Token for different hardware. In that case you can control few hardwares from 1 dashboard.
  - you can do it using Bridge functionality which allows you to send messages from one hardware to another.

- Does Blynk server store sensor data when app goes offline?

  Yes, every command that hardware sends to server is stored. You could use History Graph widget in order to view it.

Links

- Blynk site
- Blynk community
- Facebook

- Twitter
- Blynk Library
- Blynk Server
- Kickstarter campaign

License