Send Data From Hardware To Blynk

How to send any data from any sensor

Getting Started -> Send Data From Hardware To Blynk

How Data Flows From Device to Blynk

With Blynk you can send raw or processed data from any sensor or actuator connected to the MCU board

When you send data to Blynk it flows through a Datastream using Blynk protocol. Then every value is automatically timstamped and stored in the Blynk.Cloud database (you can also send batches of timestamped data if needed).

Datastream is a channel that tells Blynk what type of data is flowing through it.

Blynk can work with any sensor

With Blynk you can send any raw or processed data from any sensor or actuator.

First, make sure you can read your sensor data without Blynk.

There are thousands of different sensors in the world. Some of them can be read simply from Analog Pins, others require special libraries to interpret the data correctly.

Before you try to send data to Blynk **you should be able to print the sensor reading to Serial Monitor.**

- Search for tutorials on how to read your specific sensor;
- Find a library that works with your sensor;
- Install the library for your sensor to Arduino IDE;

- Print the sensor data to Serial;

If you can't get readings from the sensor without Blynk, you won't be able to move further

Virtual Pins Datastream

You should be already familiar with Digital and Analog pins which are used on your hardware to transfer data from connected sensors.

Virtual Pins are a Blynk abstraction designed to exchange **any data** between your hardware and Blynk. Anything you connect to your hardware will be able to talk to Blynk. With Virtual Pins you can send something from the App, process it on the microcontroller, and then send it back to the smartphone. You can trigger functions, read I2C devices, convert values, control servo and DC motors etc.

Virtual Pins can be used to interface with external libraries (Servo, LCD, and others) and implement custom functionality.

Why Use Virtual Pins To Send Data To Device?

- Virtual pins are hardware-independent. This means that it’s far easier to port your code from one hardware platform to another in the future (when you realize that the NodeMCU is far better than the Arduino Uno + ESP-01 that you started with, for example).
- You have far more control over what your widgets do when using Virtual Pins. For example, if you want a single app button to switch multiple relays on or off at the same time then that’s simple with virtual pins, but almost impossible using digital pins.
- Virtual pins are more predictable (stable if you like) than manipulating digital pins.

How do Virtual Pins **relate to the GPIO pins on my hardware?**

Virtual Pins are really just a way of sending a message from the app to the code that’s running on your board (via the Blynk server).
There is no correlation between Virtual Pins and any of the physical GPIO pins on your hardware. If you want a Virtual Pin to change the state of one of your physical pins then you have to write the code to make this happen.

Sending And Storing Data

Depending on the plan you choose, the data can be stored as-is (Raw data) or will be averaged to a one-minute average. Averaging means that if you sent 60 values per minute, Blynk will only store one value. You can still see the data flowing in in real-time.

Before you start sending data, we need to prepare some place to store it and visualize. Let's use Chart Widget in Blynk.Console for that and plot the noise coming in from Analog Pin A0 on the hardware.

1. Go to Blynk.Console -> Templates -> Create New Template
2. Go to Web Dashboard Tab -> Add Chart Widget, then open Widget Settings
3. Press Create New Datastream -> Virtual Pin

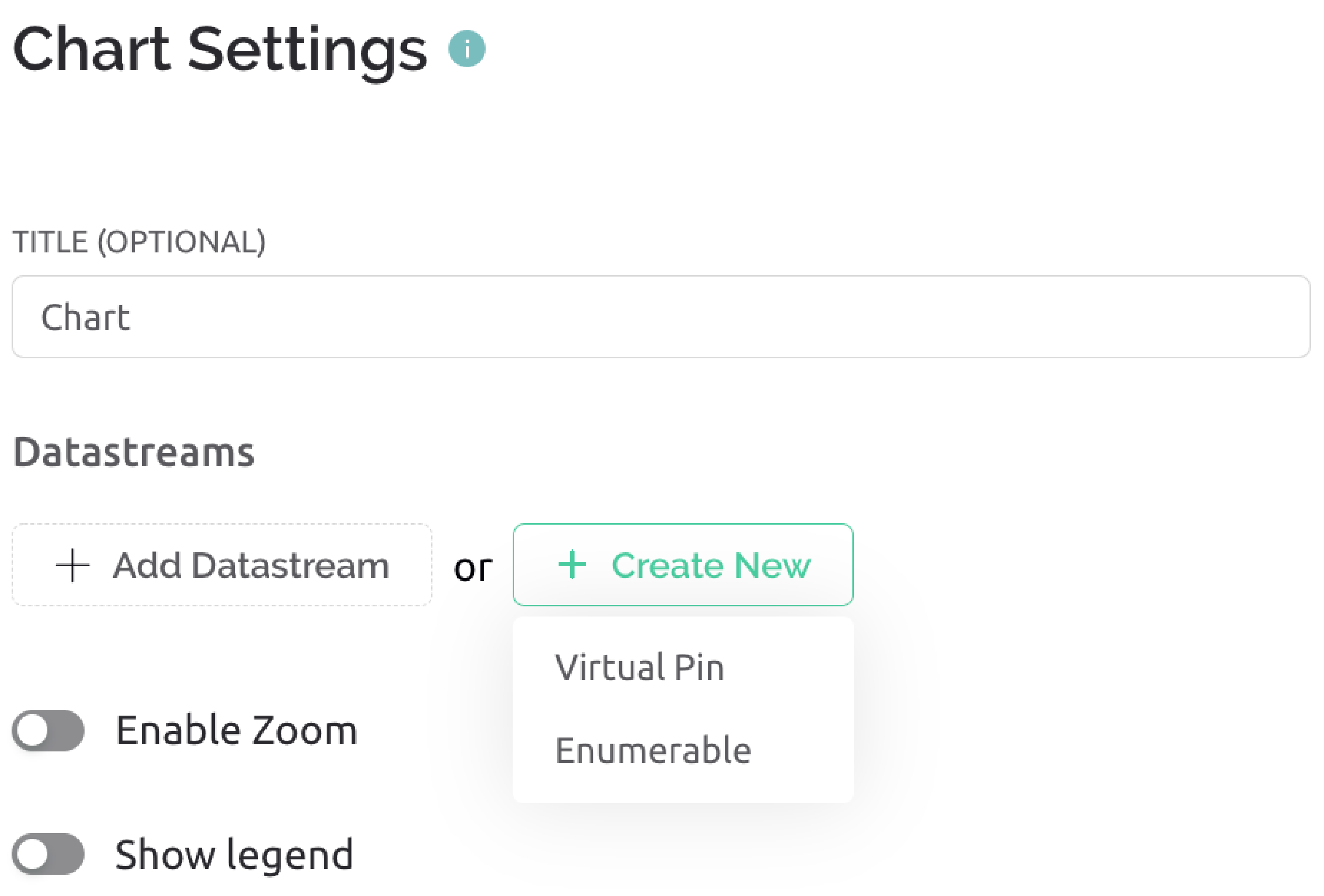


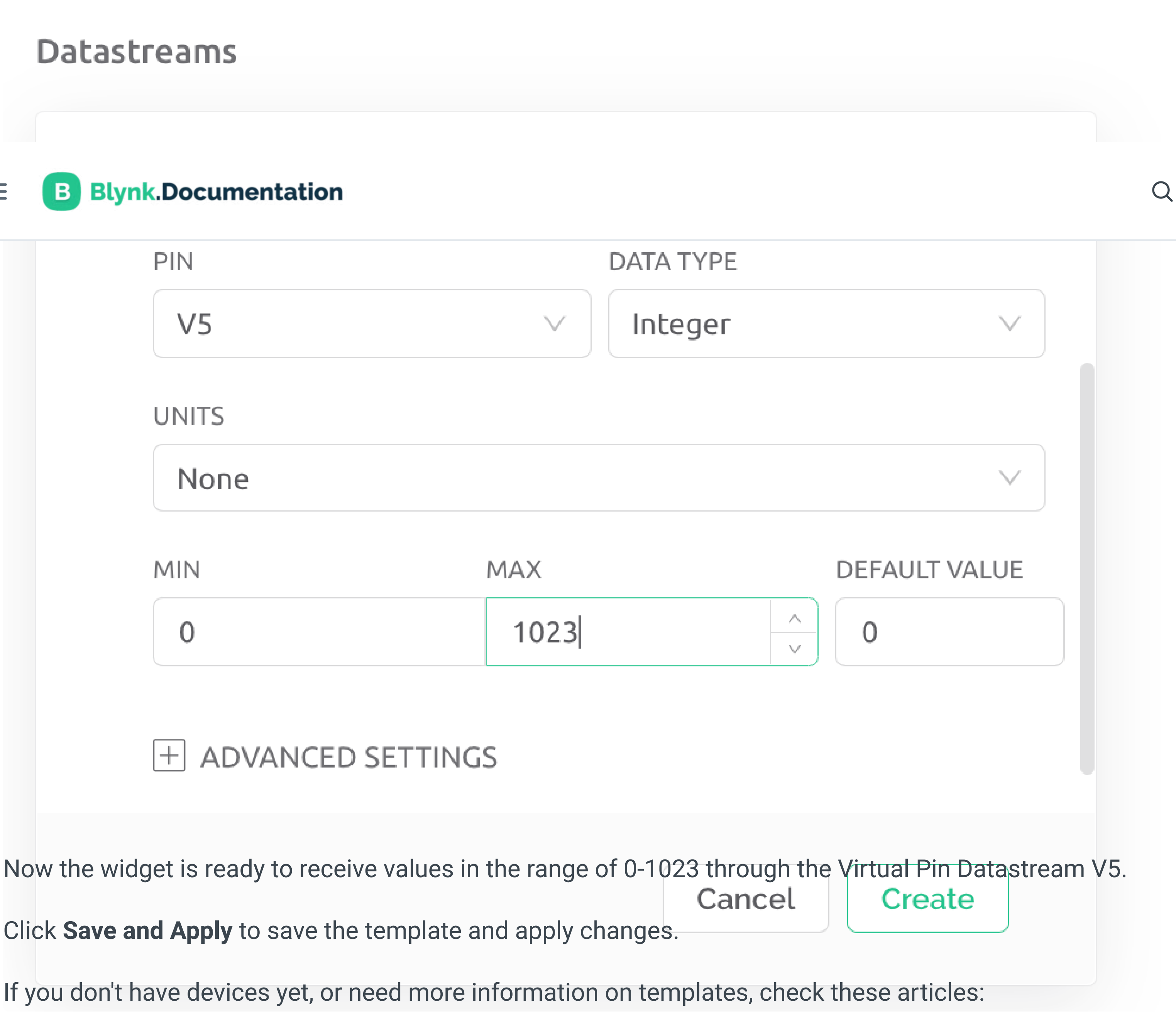
Chart Settings

TITLE (OPTIONAL)

Chart

Datastreams

Now set up the Datastream like this and press create



Datastreams

PIN	V5
DATA TYPE	Integer
UNITS	None
MIN	0
MAX	1023
DEFAULT VALUE	0

Now the widget is ready to receive values in the range of 0-1023 through the Virtual Pin Datastream V5.

Click **Save and Apply** to save the template and apply changes.

If you don't have devices yet, or need more information on templates, check these articles:

- **How to create a device from Template**
- **Quick Template setup**

Now you are ready to send the data from your device. Depending on the chosen hardware and connectivity method you can choose between two main methods of sending data:

- Blynk Library Firmware API: for devices that can be constantly connected to the internet. For example: WiFI or Ethernet
- HTTPS API: for cellular devices or any other cases when you need to use standard HTTP protocol

Send Data With Blynk Library Firmware API

This method utilizes Blynk Protocol and it's the most common and easy-to-use method when you need to send data in real-time.

First you need to do is setup a template with a datastream to configure what type of data your hardware will be sending.

When you have the datastream set, use its Virtual Pin number further.

```
sensorData = analogRead(A0); // this is an example of reading sensor data
Blynk.virtualWrite(V5, sensorData);
```

Hardware may send data to the Widgets over the Virtual Pin like this:

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

Use timers!

> It's important to understand that if you put such code into a `void loop()` it will execute "gazillion" times. This could spam the Blynk.Cloud with thousands of messages from your hardware. When this happens, Blynk.Cloud will cut off the connection between hardware and server.

To avoid spamming the server, send data only when it's needed (event-based) or use timers to send data in controlled intervals. Blynk Library offers a built-in timer for your convenience.

This is an example code on how to send data every second with a timer:

```
// Declaring a global variabl for sensor data
int sensorVal;

// This function creates the timer object. It's part of Blynk library
BlynkTimer timer;

void myTimer()
{
  // This function describes what will happen with each timer tick
  // e.g. writing sensor value to datastream V5
  Blynk.virtualWrite(V5, sensorVal);
}

void setup()
{
  //Connecting to Blynk Cloud
  Blynk.begin(auth, ssid, pass);

  // Setting interval to send data to Blynk Cloud to 1000ms.
  // It means that data will be sent every second
  timer.setInterval(1000L, myTimer);
}

void loop()
{
  // Reading sensor from hardware analog pin A0
  sensorVal = analogRead(A0);

  // Runs all Blynk stuff
  Blynk.run();

  // runs BlynkTimer
  timer.run();
}
```

Send Data Using HTTPs RESTful API

If you prefer or need to use HTTP protocol, use it with any device that supports HTTP client functionality.

With HTTP API you can send individual values, multiple values, and even upload batches of already timestamped data

Read more about available API methods here:

HTTPs REST API

Previous
Test your Template

Next - Getting Started
Control Devices (GPIOs and beyond)