I'm not a robot

We’ve detected that JavaScript is disabled in this browser. Please enable JavaScript or switch to a supported browser to continue using x.com. You can see a list of supported browsers in our Help Center. Help Center Hy there! I started a new plugin yesterday: GitHub An octoprint plugin for prometheus compatible metrics endpoint - tg44/OctoPrint-Prometheus-Exporter Nearly 0 python knowledge, but it is working so far. In the near future, I will publish a single and a multiprinter grafana dashboard. Questions towards the community: what would you like to see in a single/multiprinter grafana dashboard? Question for other plugin/octoprint devs: The "required" part configured ok in the setup.py? Is there any easy way to test a newly written and committed plugin (like pop up a container click to there do this, check that, etc.)? If not shouldn't we write it into the distribution doc? can somebody point me towards plugins/codesnippets/functions where the eventHandler used? (I watched lot of plugins code, but either they are too simple or too large) I will try to add it to the main repository after some more metrics implemented (maybe tomorrow). Feedbacks are welcome! 3 Likes tg44: can somebody point me towards plugins/codesnippets/functions where the eventHandler used? (I watched lot of plugins code, but either they are too simple or too large) My plugin does things using the event handler plugin here. 1 Like Hi @tg44, most of your questions I didn't understand, but I will try to answer: Unclear: If you have python package dependencies the plugin_requires section is the correct way to do (you already use prometheus_client) Unclear: Do you mean a wizardbased plugin-tutorial after installing a new plugin to show/guide the user all features of the newly installed plugin? Like jneilliii mentioned using the eventhandler is easy: -- include octoprint.plugin.EventHandlerPlugin in your class -- implement the def on_event(self, event, payload): method Now, I have some questions What is the goal of your plugin? From my point of view, you want to provide a Prometheus endpoint so that the Prometheus-Server can pull OctoPrint Informations on a regular timeframe, right? I think there are some drawbacks with this approach (please correct me if I am wrong): You need to define the data with the ui-representation in the OctoPrint-Plugin like Gauge or Counter Also if the octoprint-data is not changed you have always network traffic for the "pulling" and probably the "not changed data-event" is stored in the prometheus database I am asking, because I am also looking for an option to visualize historical data. My goal is to use but I need a storage solution for that. The OctoPrint-InfluxDB plugin looks promising, maybe a more generic/api approach, so that other plugin-developers could reuse the connection and can send data to that storage. (the influx-db and the grafana-server should be installed on a separate raspi) I think developing individual ui-diagramms for special cases like print-history or temp-graphs takes a lot of effort. Also, each user wants to see other informations in such a diagram. Thats the reason why I don't started implementing a "print-history-diagram' for my plugin Other developers like @Stefan_Cohen also asking for a timeseries database solution...maybe we can start kind of concept-phase of what functionalities should be in the "timeseries database connector". What do you or others think? Bye Olli In the meantime I solved a lot of stuff I will do a push probably today, but I get a new hw and try to make it work first I will try the install process in a live setup, I think you answered my main concern. I meant something like a test framework for plugin devs. I created my plugin, fine, it is working on my machine, fine. How hard it is to try out with a fresh setup? I have no idea. (I have better idea bcs I started to go after the dockerized solutions, and found some promising images.) It would be nice to have a small writeup with "what to do steps" if sb with close to zero python ecosystem knowledge, but basic ops knowledge can follow to make sure the developed plugin will work as-is on other machines too. (But probably this is only my insecureness with the lang end ecosystem, and it's not a problem for others.) solved, but thx I definitely want to use grafana. I don't think that we need to add more functionality to the webui. We have good tools and those scales better than reimplementing them again and again in every project. There is a huge debate between pull and push data consumption. Both have pros and cons. I think in general the pull method has better "real time" performance (you don't need to do the modification/send when things change, you only need to inc/dec/set a variable). Also, most of the "problems" you mentioned are already handled. Prometheus is starting to be the de-facto monitoring solution in the docker/k8s world, I had no problem with it so far. I'm not pleased with the concept of influx as a monitoring solution, its a kinda good timeseries db, but you need to write quite a lot machinery if you want to send/get/aggregate data (compared to prometheus, where the whole thing is handled by the server). Also the clients need to know who is consuming the data, they need to handle connection errors, etc. (Compared with prom. where you open a single endpoint and you don't need to know if it is consumed by one or ten nodes with minute or in hour intervals, the only need that you need to do is generating a text output for every request.) Also the duplicated work is not really much I think, I wrote this from the ground (without knowing python but with experience with a bunch of other programming languages) in about 8-12hr (which contains the whole install and "I have no idea what I'm doing" phase with documentation reading, and googleing "how to transform float to string in python"), I can live with it if nobody else thinks that its useful tl;dr: I think I have a close to good output with a lot of new metrics (not commited yet); I will work towards to add my local code, and some infra code to the repo, and test how its actually works with a prom/graf setup (and later on, start a grafana dashboard) I will check back (I hope) shortly! Good news! It's actually working! I uploaded my latest code, added example compose stack for testing and easier bootstraping. I also started a print, and it is geting the data as it should be! The next big thing will be the dashboards! I'll publish this version to the main plugin repo. (Also I bought a "mini server" for octoprint, and its working much better than the orangepis (ok, it has almost 10x the hw but still).) To help others who might have stumbled upon this... From the docs What is Prometheus? Prometheus is an open-source systems monitoring and alerting toolkit originally built at SoundCloud. Since its inception in 2012, many companies and organizations have adopted Prometheus, and the project has a very active developer and user community. It is now a standalone open source project and maintained independently of any company. To emphasize this, and to clarify the project's governance structure, Prometheus joined the Cloud Native Computing Foundation in 2016 as the second hosted project, after Kubernetes. 1 Like Yapp, I included links to the doc. Also we are the Jobs/exporters in this image. So this integration can open up not just graph visualisations (grafana), but also alerting, easily. With service discovery and some hard work this can help a lot to multiprinter setups. I've personally done some work with the opc-ua specification for device availability/capabilities. It's more on the machine- and job-management side of things. en.wikipedia.org OPC Unified Architecture (OPC UA) is a machine to machine communication protocol for industrial automation developed by the OPC Foundation. Distinguishing characteristics are: Although developed by the same organization, OPC UA differs significantly from its predecessor, Open Platform Communications (OPC). The Foundation's goal for OPC UA was to provide a path forward from the original OPC communications model (namely the Microsoft Windows-only process exchange COM/DCOM) that would better meet ... Hi @tg44, this week I have a 2x6h train ride...much time to checkout your docker images I will also try to implement some kind of metric-event-listener in your plugin, so that other plugins could send events with metric-data and your plugin grabs that data, wraps it into a metric-types like gauge, counter, etc. so that the prometheus-server could read it. I have been running a grafana/influxdb container in my homelab for about four years now and I really like it. I use it daily to monitor servers, services and network equipment. I feed influxdb with data using telegraf and a plethora of homegrown scripts. I haven't played with Prometheus but it seems very similar to Influx. OllisGit: I think developing individual ui-diagramms for special cases like print-history or temp-graphs takes a lot of effort. Also, each user wants to see other informations in such a diagram. I've started working on a graphing replacement for temperatures that will also allow for other plugins to submit to and have other data visualized via plotly. I haven't yet got to the second part of opening it up to other plugins, but I do have the temperatures graphing working already. Of course, this is only handling realtime data while the ui is open and not storing the data for analysis. 1 Like Hi @tg44 !! Btw. what is the difference between your plugin and the existing one? and sourcecode location GitHub OctoPrint plugin that creates a prometheus endpoint - sbelectronics/octoprint-prometheus OllisGit: I will also try to implement some kind of metric-event-listener in your plugin, so... Nice idea! Thanks the help! Stefan_Cohen: I haven't played with Prometheus but it seems very similar to Influx. Yapp, as I said there are pros and cons for both. If you played with influx a lot, and has your scripts influx is nice. But I more like the prometheus infra, where I don't need any homegrown scripts, or connect my app to a db, or do anything by hand. If I start a mysql/pgsql/rabbitmq/elastic-search/etc. almost a no-brainer to get them under a prometheus server. jneilliii: I've started working on a graphing replacement for temperatures that will also allow for other plugins to submit to and have other data visualized via plotly. This I think is better for single-node installs. But for multi-node and portability grafana is better in my opinion. But if you already have some code with a metric-stream that would be helpful I think. OllisGit: Btw. what is the difference between your plugin and the existing one? I didn't find this. I checked the community forum, and the plugin repo with prometheus, grafana, graph, and there was no relevant conversation. Also as I checked this repo, it is really not what I think is useful. For ex. as the head moves you update the x,y,z coordinates, but the metric-scraper will only ask you for `/15/30 secs. What is the information in the x,y coordinates in every 30 sec? Also if you have retraction and z-hop the e and z means nothing too. They also start up a new web-server on a new port which is at least questionable. (I think this could be a good and a bad idea too, in a dockerised env definitely bad, in hosted env its a meh...) Cool. I can dump all of the Dasboard plugin metrics directly to your plugin once the API is in place. That should cover a few metrics. //Stefan Hi @tg44, I played around with your plugin, prometheus and grafana. Btw. setting up a fresh docker based infrastructure of the toolstack during a trainride is not the best idea It takes 2-3h or Hanover-Wuerzburg and a lot of restarts, wifi is not the best. Currently, I am not so happy with the prometheus pull approach. The default setup pulls every 5 seconds to get the newest values. If you have value changes between the 5 second frame, there are not covered. Of course I can decrease the intervall or add a queuing function into the plugin. And also if the connection between prometheus and octoprint is broken/not established you don't see it in octoprint. Thats the reason why I setup a docker based influxdb. You can see the differnence. The tool-temp. value of 40 is not shown at 9:33 on the right side. IMHO: Prometheus is an easy to use monitoring solution for server/container with out of the box queries, but not usefull for (near)realtime/or event based monitoring. Maybe I missed something, but currently I will try my luck with the influxdb-plugin. C ya, Olli 1 Like Yes, this realtime event based monitoring is not for prometheus. But you really need realtime data? The connection lost between the prom and octoprint instance is also a questionable con. I think in ideal situations the "app" should know nothing about the monitoring infra, and should work the same if its monitored or not. But thanks the testing I got this up and running yesterday with Docker containers on Windows, both Prometheus and Grafana setup with persistent data mappings. I am happy to share and/or add to the documentation on GitHub if that helps. However, I am certainly not going to be good at making the dashboards and related queries. If you could share Grafana config, that would be extremely helpful. Thank you. Also, I am currently using the PrinterStats plugin on OctoPrint. Do you think it is possible to mine the data out of that plugin before removal? evannadeau: If you could share Grafana config, that would be extremely helpful. Yapp, the problem is, that I don't have a full-featured dashboard currently. I still in between to migrate my one printer setup to a three printer setup, while I have a lot of other stuffs in my to-do queue But: based on this I think if you drop in an issue, and tag davidfeeney he probably can share a board with us. evannadeau: Also, I am currently using the PrinterStats plugin on OctoPrint @OllisGit had an idea with a metric-event-listener stream, but I think it didn't happened yet. I would adopt to such a thing. Until then, if you are a python dev, you can try to adopt some of the PrinterStats code to my lib as a PR Is there an example available for setting up the metric feature using Prometheus for the OPC-UA extractor, similar to what is shown for the PI extractor in the Cognite Academy? I found it on Cognite Extractor OPC-UA GitHub. I'll give it a try.metrics: # Start a metrics server in the extractor for Prometheus scrape server: host: port: 0 # Multiple Prometheus PushGateway destinations: push-gateways: - host: job: username: password: # Configuration to treat OPC-UA nodes as metrics. # Values will be mapped to opcua_nodes_NODE-DISPLAY-NAME in prometheus. Learn more about our cookies.","cookiepolicy.button":"Accept All Cookies","cookiepolicy.button.deny":"Deny all","cookiepolicy.link":"Cookie Settings","cookiepolicy.modal.title":"Cookie settings","cookiepolicy.modal.content":"We use 3 different kinds of cookies. You can choose which cookies you want to accept. We need basic cookies to make this site work, therefore these are the minimum you can select. Learn more about our cookies.","cookiepolicy.modal.level1":"BasicFunctional","cookiepolicy.modal.level2":"NormalFunctional + analytics","cookiepolicy.modal.level3":"CompleteFunctional + analytics + social media + embedded videos + marketing"}}}> Groups keyboard shortcuts have been updatedSkip to first unread messageunread,Jan 28, 2021, 9:30:20 AM1/28/21Sign in to reply to authorYou do not have permission to delete messages in this groupEither email addresses are anonymous for this group or you need the view member email addresses permission to view the original messageHi All,Any one please help me how to monitor OPC UA Server using PrometheusIf OpcUaServer is down I should notified and trigger the alert from alert manager.I have checked the below one but not able to understand,Any one please help me on that This is a Prometheus exporter for the OPC Unified Architecture protocol. It uses gopcua/opcua to communicate with an OPCUA endpoint, subscribes to selected channels, and republishes them as Promtheus metrics on a port of your choice. Usage of opcua_exporter: -buffer-size int Maximum number of messages in the receive buffer (default 64) -config string Path to a file from which to read the list of OPC UA nodes to monitor -config-b64 string Base64-encoded config JSON. Overrides -config -debug Enable debug logging -endpoint string OPC UA Endpoint to connect to. (default "opc.tcp://localhost:4096") -max-timeouts int The exporter will quit trying after this many read timeouts (0 to disable). -port int Port to publish metrics on. (default 9686) -prom-prefix string Prefix will be appended to emitted prometheus metrics -read-timeout duration Timeout when waiting for OPCUA subscription messages (default 5s) -summary-interval duration How frequently to print an event count summary (default 5m0s) You need to supply a mapping of stringified OPC-UA node names to Prometheus metric names. This is nessecary because the OPC-UA node strings use ; and = characters to seperate fields from each other and names from values, while Prometheus metric names must match the regex [a-zA-Z_:][a-zA-Z0-9_:]*. For good advice on Prometheus metric naming, refer to the Prometheus docs. The exporter uses a YAML config file to specify the mapping. An example config file might look like: - nodeName: ns=1;s=Voltmeter metricName: circuit_input_volts - nodeName: ns=1;s=Ammeter metricName: circuit_input_amps - nodeName: ns=1;s=CircuitBreakerStates extractBit: 3 # pull just this bit from a bit-vector channel metricName: circuit_breaker_three_tripped Some of our OPC-UA devices send alarm states as binary bit-vector values, for example, a 32-bit unsinged integer where each bit represents the state of a circuit breaker. If you want to monitor a single bit, you can specify an extractBit in the node configuration file. The exporter will pull just that bit (zero-indexed) from the value of the OPC-UA channel, and export it as a 0.0 or 1.0 Prometheus metric value. Utilizing the datasource plugin framework, this projects allows you to access data from OPC UA servers directly from Grafana.TLDR;To build on any platform:StatusCurrently used in production and under active maintenanceWhat worksBrowsing and adding multiple serversAuthenticated connection with certificate or no securityGraphical query editorData Access (DA)Historical Access (HA / HDA)Alarms & Conditions (AC / AE)What needs to be implementedOPCUA DA Subscriptions: These are the ones where you will not need to hit the refresh button. If you have subscribed to a data point at 500ms, you will get an unsolicited update every 500ms.Password authenticationTwo-way communication with the OPC UA server (currently read only)Description and ArchitectureThis plugin uses GRPC and a C# backend to communicate with Grafana directly. See pkg/dotnet directory for the backend componentBuildingyarn install to install dependenciesyarn build | yarn dev to build the pluginmake build to build the backend componentRestart Grafana and you should have the datasource installed.ConfigurationYou will need to add a plugin specific configuration section to your configuration file (grafana.conf) Copy[plugin.grafana-opcua-datasource] data_dir = "/some/path/to/config/grafana-opcua-datasource"An example of this using common linux default paths would be Copy[plugin.grafana-opcua-datasource] data_dir = "/var/lib/grafana-opcua-datasource"ContributingContributions that addresses the needs above or other feature you’d like to see are most welcome. Fork the project and commit a PR with your requests. is a proud contributor Open Software like this project and open standards like OPC UA fits perfectly with our quest to give our clients the freedom to operate. To know more about our offerings and get in touch, check out has a public test OPC UA Server running at opc.tcp://opcua.prediktor.com:4880. It provides a small information model with Realtime/Historic data and Alarms&Events. Feel free to use itYou can find complementary OPC UA Grafana panel plugins at AQ: Can it read OPC Classic DA/HDA/AE?A: Yes, provided use use the OPC Foundations COMIOP wrapper, which you can find here. You will need to configure IOP to wrap your OPC COM server. Tested against Matrikon OPC Desktop Historian and Matrikon OPC Simulation Server.VisionThe OPC UA datasource should beEasy to useGraphically configurableSmart (ie: Able to use metadata from the browsing of tags to ease configuration in whatever panel the datasource is interacting with)Robust and Reliable (of course!) You can’t perform that action at this time. You can’t perform that action at this time. This is a Prometheus exporter for the OPC Unified Architecture protocol. It uses gopcua/opcua to communicate with an OPCUA endpoint, subscribes to selected channels, and republishes them as Promtheus metrics on a port of your choice. Usage of opcua_exporter: -buffer-size int Maximum number of messages in the receive buffer (default 64) -config string Path to a file from which to read the list of OPC UA nodes to monitor -config-b64 string Base64-encoded config JSON. Overrides -config -debug Enable debug logging -endpoint string OPC UA Endpoint to connect to. (default "opc.tcp://localhost:4096") -max-timeouts int The exporter will quit trying after this many read timeouts (0 to disable). -port int Port to publish metrics on. (default 9686) -prom-prefix string Prefix will be appended to emitted prometheus metrics -read-timeout duration Timeout when waiting for OPCUA subscription messages (default 5s) -summary-interval duration How frequently to print an event count summary (default 5m0s) You need to supply a mapping of stringified OPC-UA node names to Prometheus metric names. This is nessecary because the OPC-UA node strings use ; and = characters to seperate fields from each other and names from values, while Prometheus metric names must match the regex [a-zA-Z_:][a-zA-Z0-9_:]*. For good advice on Prometheus metric naming, refer to the Prometheus docs. The exporter uses a YAML config file to specify the mapping. An example config file might look like: - nodeName: ns=1;s=Voltmeter metricName: circuit_input_volts - nodeName: ns=1;s=Ammeter metricName: circuit_input_amps - nodeName: ns=1;s=CircuitBreakerStates extractBit: 3 # pull just this bit from a bit-vector channel metricName: circuit_breaker_three_tripped Some of our OPC-UA devices send alarm states as binary bit-vector values, for example, a 32-bit unsinged integer where each bit represents the state of a circuit breaker. If you want to monitor a single bit, you can specify an extractBit in the node configuration file. The exporter will pull just that bit (zero-indexed) from the value of the OPC-UA channel, and export it as a 0.0 or 1.0 Prometheus metric value. This is a Prometheus exporter for the OPC Unified Architecture protocol. It uses gopcua/opcua to communicate with an OPCUA endpoint, subscribes to selected channels, and republishes them as Promtheus metrics on a port of your choice. Usage of opcua_exporter: -buffer-size int Maximum number of messages in the receive buffer (default 64) -config string Path to a file from which to read the list of OPC UA nodes to monitor -config-b64 string Base64-encoded config JSON. Overrides -config -debug Enable debug logging -endpoint string OPC UA Endpoint to connect to. (default "opc.tcp://localhost:4096") -max-timeouts int The exporter will quit trying after this many read timeouts (0 to disable). -port int Port to publish metrics on. (default 9686) -prom-prefix string Prefix will be appended to emitted prometheus metrics -read-timeout duration Timeout when waiting for OPCUA subscription messages (default 5s) -summary-interval duration How frequently to print an event count summary (default 5m0s) You need to supply a mapping of stringified OPC-UA node names to Prometheus metric names. This is nessecary because the OPC-UA node strings use ; and = characters to seperate fields from each other and names from values, while Prometheus metric names must match the regex [a-zA-Z_:][a-zA-Z0-9_:]*. For good advice on Prometheus metric naming, refer to the Prometheus docs. The exporter uses a YAML config file to specify the mapping. An example config file might look like: - nodeName: ns=1;s=Voltmeter metricName: circuit_input_volts - nodeName: ns=1;s=Ammeter metricName: circuit_input_amps - nodeName: ns=1;s=CircuitBreakerStates extractBit: 3 # pull just this bit from a bit-vector channel metricName: circuit_breaker_three_tripped Some of our OPC-UA devices send alarm states as binary bit-vector values, for example, a 32-bit unsinged integer where each bit represents the state of a circuit breaker. If you want to monitor a single bit, you can specify an extractBit in the node configuration file. The exporter will pull just that bit (zero-indexed) from the value of the OPC-UA channel, and export it as a 0.0 or 1.0 Prometheus metric value. Telegraf is a plugin-driven agent that collects, processes, aggregates, and writes metrics. It supports four categories of plugins: input, output, aggregator, and processor. In addition to the included plugins, you can run external plugins that integrate with the Telegraf Execd processor plugin.Jump to:Input pluginsOutput pluginsAggregator pluginsProcessor pluginsInput pluginsTelegraf input plugins are used with the InfluxData time series platform to collect metrics from the system, services, or third-party APIs. View View View