Asvin Documentation

Release 0.0.1

Asvin

02.05.2022

Inhaltsverzeichnis

Bemerkung: Bitte achten Sie darauf, dass Sie die zur Software passende Version der Dokumentation ansehen. Die eingestellte Version finden Sie oben im Navigationsbereich. Ändern können Sie diese über das Auswahlmenü rechts unten im Navigationsbereich.

Ein Platform-as-a-Service (PasS) in Enterprise-Qualität zur Verwaltung, Verteilung und Überwachung von Firmware-Updates und zur Erstellung einer Echtzeit-Bedrohungslandschaft für registrierte IoT-Geräte.

KAPITEL 1

Einleitung

Es gab eine Zeit, in der die Zubereitung von Kaffee ein breites Spektrum an Aufgaben erforderte. Zuerst ging man auf den Markt, um frisch geröstete, aromatische Kaffeebohnen zu kaufen, dann mahlte man sie und während der Kaffee brühte, holte man die Zeitung von der Haustür und begann seinen Tag, während man darauf wartete, dass der Kaffee „aufholte“. Spulen Sie ins Jahr 2021 vor, und unsere „intelligente“ Kaffeemaschine überwacht unser Armband, um festzustellen, ob wir wach sind, so dass zu dem Zeitpunkt, an dem wir die Küche erreichen, unsere handwerklich hergestellten Kaffeebohnen bereits frisch geröstet, gemahlen und genau nach unseren Vorlieben gebrüht wurden. Dazu kommt, dass diese Kaffeemaschine nun überwachen kann, wie viel Kaffee noch im Trichter ist, unsere Lieblingsbohnen bei einem bevorzugten Verkäufer bestellen kann und die Bezahlung automatisch mit den im Gerät gespeicherten Kreditkartendaten vornimmt

Wir sind von vielen intelligenten Geräten wie unserer Kaffeemaschine umgeben. Ausgestattet mit drahtlosen Kommunikationsprotokollen wie WiFi, Bluetooth, NFC, LoRaWAN usw. sprechen diese intelligenten Geräte miteinander und sind mit dem Internet verbunden und werden als Geräte des Internet-of-Things (IoT). IoT-Geräte sind heute in unserem Leben omnipräsent- ob in unseren Häusern, Städten, Arbeitsplätzen oder in Krankenhäusern. Wir leben in einer Zeit, in der intelligente Geräte häufiger mit uns kommunizieren und interagieren als andere Menschen. Bis zum Jahr 2030 werden die Anzahl der IoT-Geräte und die Größe des IoT-Marktes voraussichtlich 125 Milliarden bzw. 1,3 Billionen USD erreichen.

IoT-Geräte vereinfachen Prozesse und machen unser Leben bequemer. Andererseits sind diese Geräte anfällig für Cyber-Sicherheitsbedrohungen. Wenn beispielsweise jemand unsere harmlos aussehende smarte Kaffeemaschine hackt, könnte sie unsere WLAN-Anmeldeinformationen, Kreditkartendaten, Schlafgewohnheiten und viele weitere persönliche Daten preisgeben, aber sie kann auch als Bot verwendet werden, um Cybersicherheitsangriffe auf andere digitale Ressourcen zu starten. Mirai-2016 beispielsweise verwandelte über sechs Millionen vernetzte Geräte in ein Botnet, um DDoS-Angriffe zu starten. Diese Arten von Angriffen haben das exponentielle Wachstum von IoT-Geräten aus einer Vielzahl von Gründen überholt, darunter die Verwendung von Standard- oder schwachen Passwörtern, kleinen Encryption Keys oder das Versäumnis, Sicherheitspatches und Software-Updates herunterzuladen.

Eines der größten Sicherheitsprobleme ist, dass IoT-Geräte keine eingebauten Over-the-Air (OTA)-Firmware-Update-Mechanismen haben. IoT-Geräte werden mit einer „Set and Forget"-Politik hergestellt und verwaltet, d. h. IoT-Geräte werden zu Beginn ihres Lebenszyklus konfiguriert und dann allein gelassen, um sich gegen Sicherheitsangriffe zu schützen. Technologien entwickeln sich so schnell weiter, dass es von größter Bedeutung ist, einen gut definierten, sicheren Prozess zu haben, um die Bedrohungslandschaft von IoT-Geräten kontinuierlich zu bewerten und die Firmware regelmäßig mit den neuesten Sicherheits-Patches zu aktualisieren.

In den letzten paar Jahren gab es große Entwicklungen im Bereich der Update-Plattformen für IoT-Geräte. Einige Lösungen, wie z. B. Eclipse hawkBit, Mbed und Mender, adressieren das bestehende Problem, aber ihre Lösungen sind auf eine bestimmte Mikrocontroller-Architektur beschränkt, Software-Stacks und Branchen sind als zentralisierte Lösungen konzipiert und entwickelt, die anfällig für Single-Point-Failure sind und nicht sehr gut zu skalieren. In den folgenden Abschnitten wird beschrieben, wie sich die asvin-Plattform von anderen IoT-Update-Lösungen unterscheidet.

1.1 Asvin Platform

Die asvin-Plattform ist eine Cloud-Plattform der Enterprise-Klasse, die mit Distributed Ledger Technology (DLT) entwickelt wurde, um Patches zu verteilen und eine Vertrauenskette für IoT-Geräte zu erzeugen. Die asvin-Plattform basiert auf **verteilten** und **dezentralen** Technologien und nutzt das Interplanetary Filesystem (IPFS) zur Speicherung und Verwaltung der Firmware-Dateien von IoT-Geräten. Alle Firmware-Dateien und Patches sind über mehrere IPFS-Knoten verteilt, während die kritischen Metadaten-Informationen der jeweiligen Geräte in einem unveränderlichen Distributed Ledger (DL) gespeichert werden, ebenso wie jedes Ereignis, das auf der asvin-Plattform stattfindet. Diese Technologien erhöhen die Fehlertoleranz der Plattform und machen sie sowohl leichter zugänglich als auch ausfallsicherer.

Es ist eine Sache, einen funktionierenden Prototyp eines IoT-Produkts zu haben und eine andere, Millionen von intelligenten Geräten zu verwalten und zu aktualisieren. Die wünschenswerteste Eigenschaft jeder IoT-Lösung ist Skalierbarkeit. Asvin bietet diese Lösung - effektiv und effizient. Die asvin-Plattform verfügt über eine hochgradig **anpassbare** und **modulare** Architektur und wurde so konzipiert und entwickelt, dass sie mit ihren steckbaren Modulen, die für den jeweiligen Anwendungsfall optimiert sind und so IoT-Anwendungen in verschiedenen Branchen unterstützt. Im Gegensatz zu anderen FUOTA-Lösungen ist die asvin-Plattform eine **universelle** Lösung, die keine Einschränkungen für beliebige Hardware- und Software-Stacks mit sich bringt. asvin ermöglicht die Innovation und Vielseitigkeit jeder IoT-Anwendung, die mit Atmel AVR bis ARM Cortex M7 gebaut wurde, in Anwendungen von der Landwirtschaft bis zu Finanzdienstleistungen.

KAPITEL 2

Key Concepts

2.1 Überblick

Die asvin-Plattform basiert auf mehreren dezentralen Technologien zur Verteilung von Patches für IoT-Geräte mit einem höheren Grad an Sicherheit, Skalierbarkeit, Vertraulichkeit und Ausfallsicherheit und unterstützt die gesamte Bandbreite von IoT-Geräten und deren Anwendungen. Die Plattform verbirgt die Komplexität und Feinheiten von IoT-Geräten, um Firmware-Updates sicher bereitzustellen und gleichzeitig die Bedrohungslandschaft zu identifizieren und die Firmware und Software innerhalb einer sicheren Vertrauenskette zu schützen.

2.1.1 Was ist ein Firmware-Update?

IoT-Gerät

Ein IoT-Gerät ist ein Stück Hardware, das typischerweise mit Mikrocontrollern, Sensoren, Kommunikationsmodulen, Ein-/Ausgabe-Peripheriegeräten und Software ausgestattet ist, um nützliche Daten aus verschiedenen Prozessen zu sammeln und sie mit anderen angeschlossenen Geräten über das Internet auszutauschen. IoT-Geräte werden praktisch überall eingesetzt - in smarten Häusern, Produktionsanlagen, Städten, Krankenhäusern, beim Transport von Waren, an Arbeitsplätzen und im Weltraum.

Firmware

Eingebettete Software, die auf einem IoT-Gerät läuft, wird Firmware genannt. IoT-Geräte haben eine sehr kleine Grundfläche und können daher nicht mit vollwertigen Betriebssystemen betrieben werden. IoT-Geräte, die auf Arduino-, ESP-, TTGO-, WeMoS-Entwicklungsboards basieren, laufen als eigenständiges Programm. Deterministische Multithreading-Programme können auf leistungsfähigeren Boards wie Nucleo, NuMaker, PSoC unter Verwendung von Embedded-RTOS wie Mbed, FreeRTOS ausgeführt werden.

Firmware-Aktualisierung

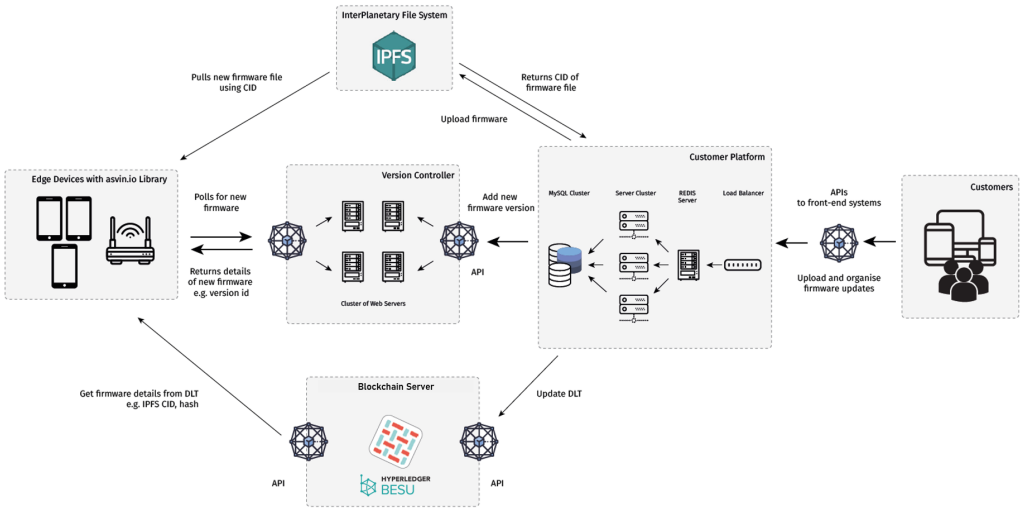
Ein Firmware-Update ist der Prozess, bei dem die Software auf IoT-Geräten aktualisiert wird. Dies kann durch Aktualisierung der kompletten Firmware oder durch Anwendung von Delta-Änderungen erfolgen.

Warum ist ein Firmware-Update notwendig?

Wir leben in einer Welt, in der Technologie sowohl ein Segen als auch ein Fluch ist. Auf der einen Seite vereinfacht Technologie unsere Prozesse, während die Ausnutzung ihrer Schwachstellen durch BlackHats und staatlich gesponserte Akteure zu enormen finanziellen Verlusten führt und die Lebensqualität von Millionen Menschen beeinträchtigt. Eine proaktive Cybersicherheitspolitik, die regelmäßige Updates und Sicherheits-Patches, sowie die Fähigkeit zur Isolierung von anfälligen IoT-Geräten umfasst, ist entscheidend.

2.2 Asvin Modell

In diesem Abschnitt werden wir einen kurzen Blick auf die Architektur von Asvin werfen und seine Komponenten kurz erklären. Einige relevante Konzepte und Schlüsselwörter werden ebenfalls erläutert.



2.2.1 Kundenplattform

asvin bietet eine benutzerfreundliche Kundenplattform, um die Komplexität der Hintergrundprozesse zu abstrahieren, um Geräte zu registrieren, Firmware-Rollouts einzurichten und zu planen, Gruppenaktionen für registrierte Geräte zu definieren und Geräte auf eine Blacklist zu setzen. Die Kundenplattform zeigt auch Informationen zu Firmware-Updates an und liefert die aktuelle Version der Edge-Geräte.

2.2.2 Version controller

Der Versions-Controller hält aktuelle Informationen über die verfügbare Firmware für ein bestimmtes Gerät oder eine Gerätegruppe auf der asvin.io-Plattform bereit. Er ist eine der Kernkomponenten der asvin Architektur.

2.2.3 Interplanetary FileSystem

Interplanetary FileSystem oder IPFS ist ein verteiltes Peer-to-Peer-Hypermedia-Protokoll, welches zum Speichern und Teilen von Daten in einem verteilten Dateisystem entwickelt wurde. IPFS dient als eine der Kernkomponenten der asvin Architektur. asvin nutzt das IPFS. Protokoll um Firmware, Updates und Patches zu speichern.

Content Identifier

Der Datenaustausch auf dezentralen Plattformen wie IPFS hängt von der inhaltsbasierten Adressierung ab, anstatt von der lokalen Adressierung, um Daten sicher zu lokalisieren und zu identifizieren. Ein Content Identifier (CID) ist ein selbstbeschreibender Inhaltsadressbezeichner, kryptographischer Hash, typischerweise SHA-256, der 32 Bytes lang ist. Der CID für eine binäre Bilddatei, die in einem IPFS Netzwerk gespeichert ist, könnte **QmcRD4wkPPi6dig81r5sLdrtd1gDCL4zgpEj9CfuRrGbzFsein**.

2.2.4 Blockchain

Die asvin-Plattform nutzt die Blockchain-Technologie, um alle Transaktionen zu speichern, die im asvin.io-Netzwerk ausgeführt werden, einschließlich: Geräteregistrierung, Firmware-Upload, Geräte-Update, Firmware-Update, Benutzerregistrierung, etc. All diese Transaktionen sind mit Hashs verbunden und in Blöcken gespeichert, die wiederum mit einem gesicherten Hash verbunden sind. Für die asvin Plattform nutzen wir die Hyperledger Fabric und Hyperledger besu blockchains.

Distributed Ledger

Distributed Ledger Technology (DLT) ist ein digitales System zur Aufzeichnung der Transaktion von Assets, in dem die Details von bestimmten Transaktionen an mehreren Stellen aufgezeichnet werden. Im Gegensatz zu traditionellen Datenbanken gibt es bei Distributed Ledger keine zentrale Datenspeicherung oder Verwaltungsfunktion. Die oben erwähnte Blockchain ist eine Art von DLT.

Smart Contracts

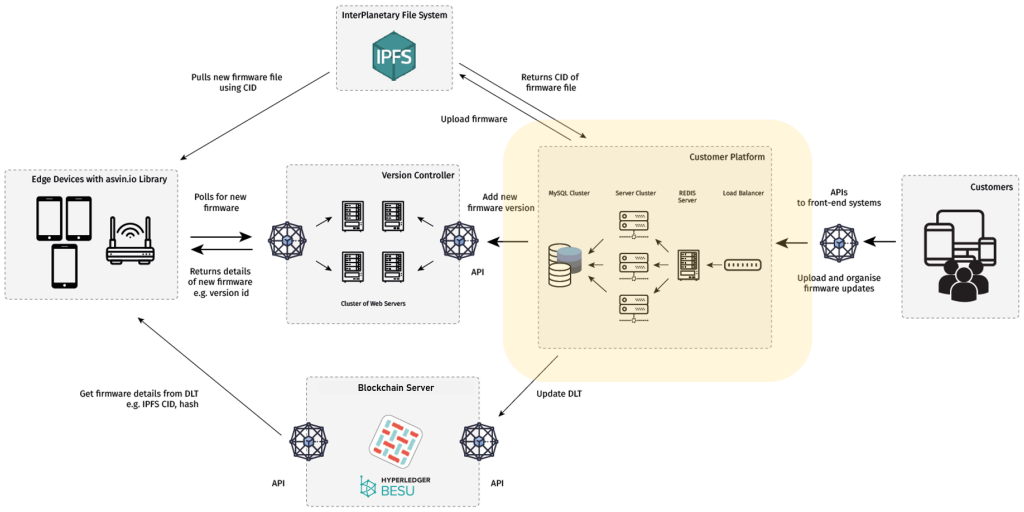
Ein Smart Contract ist ein Computerprogramm oder ein digitales Transaktionsprotokoll, das dazu gedacht ist, rechtlich relevante Ereignisse und Handlungen gemäß den Bedingungen eines Vertrages oder einer Vereinbarung automatisch auszuführen, zu kontrollieren oder zu dokumentieren. Das Ziel eines Smart Contracts ist es, den Bedarf an vertrauenswürdigen Vermittlern, Schlichtung, Durchführungskosten, Betrugsverluste sowie die Risikominderung von schädlichen Angriffen und unbeabsichtigten Störungen zu reduzieren.

2.2.5 Endgeräte oder Edge Devices

Endgeräte sind Endpunkte in der asvin.io-Architektur und in einem IoT-Netzwerk, die eine bestimmte physikalische Aufgabe steuern, verwalten und lösen: Edge Devices schalten zum Beispiel eine smarte Waschmaschine in einem Haus ein, überwachen Temperatur und Luftfeuchtigkeit in einer Chemiefabrik oder einen Luftqualität in einer Stadt. Diese Geräte haben als Herzstück Mikrocontroller und Sensoren und mit ihrem geringen Platzbedarf sind diese Edge Devices leicht in abgelegenen Gebieten unter extremen Umweltbedingungen zu betreiben. Beispiele für Edge Devices im Industrial Internet of Things (IIoT) sind Prozessüberwachungssensoren, Smart Meter, Lora-Knoten, Rauchmelder, etc.

2.3 Beehive - Plattform

In diesem Abschnitt erläutern wir die **Plattform beehive** als Bestandteil der asvin Architektur.



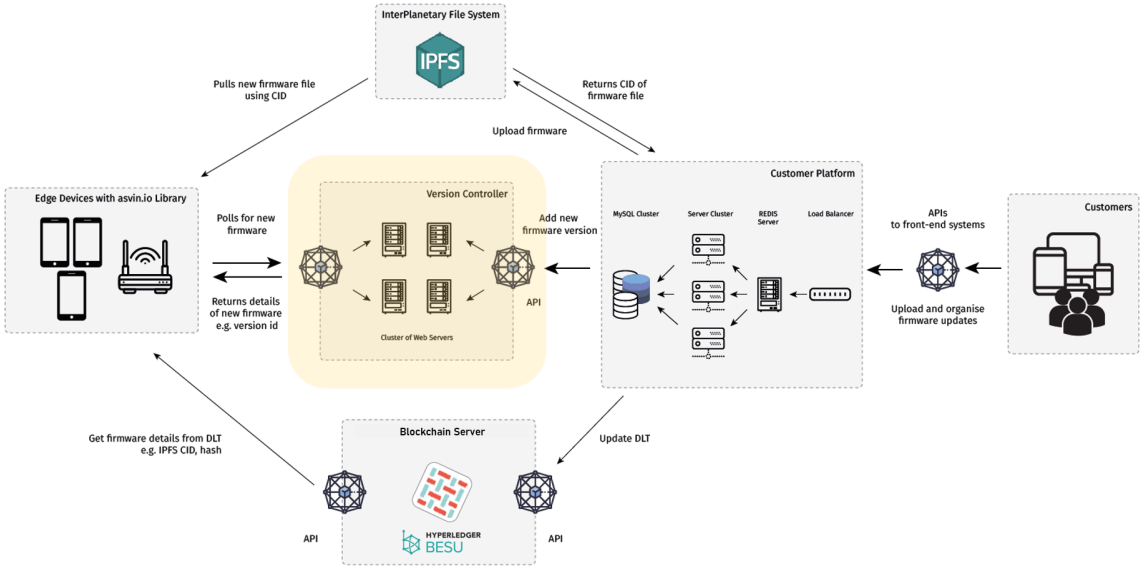
Wir bei asvin glauben an eine einfache und benutzerfreundliche Lösung. Die asvin.io Kundenplattform bietet eine Abstraktionsschicht, um die Vielschichtigkeit und die Komplexität der Distributed Ledger Technologie zu verbergen. Diese Abstraktion wird durch einen Cluster von Servern ermöglicht, der von einem Cluster an Servern unterstützt wird. Ein Load Balancer sorgt dafür, dass die Menge an IoT-Geräten, die auf der asvin.io-Plattform verwaltet werden und, dass es keine Latenz in der Verbindung zwischen dem Netzwerk des Kunden und seinen IoT-Edge-Geräten und dem asvin.io-Server gibt.

Beehive - Die Kundenplattform beinhaltet folgende Funktionen:

1. **Service Dashboard** Es dient als Portal für die IoT-Gerätebetreiber, um direkt zu interagieren und den Update- und Patch-Status ihrer Edge-Geräte zu kontrollieren. Das Portal steuert die Funktionen zum Hochladen von Firmware, Löschen von Firmware, Verwalten von Edge-Geräten, Überprüfen der Gesundheitsstatistiken von Edge-Geräten und vieles weitere.
2. **Upload Firmware** Lädt die vom Kunden bereitgestellte Firmware auf den IPFS Server hoch
3. **Update Ledger** Interagiert mit dem Hyperledger-Blockchain-Server, um die Firmware-Datenbank zu aktualisieren.
4. **Update Version Control Server** Hält den Versionskontrollserver mit Informationen über die neueste Firmware auf dem aktuellen Stand.

2.4 Version Controller

In diesem Abschnitt werden wir einen kurzen Blick auf die Komponenten des **Version Controllers** der asvin Architektur werfen. Vereinfacht ausgedrückt enthält der Version Controller (VC) aktuelle Informationen über die verfügbare Firmware für ein bestimmtes Edge-Gerät oder eine Gerätegruppe auf der asvin.io Plattform.



Um Milliarden von IoT-Edge-Geräten zu verwalten, setzt asvin.io.io eine verteilte Service-Cluster-Infrastruktur ein. Für die Edge-Geräte ist diese Komplexität nicht ersichtlich, sie interagieren mit dem Cluster wie mit einer einzelnen Maschine. Der Version Controller Server besteht aus mehreren Knoten, die eine Kopie des Webservers haben und identische Webservices hosten. Jeder Knoten im Cluster ist ein voll funktionsfähiger Webserver mit einer eindeutigen IP-Adresse und kann jede Anfrage unabhängig bedienen, aber sie sind für die Edge-Devices nicht sichtbar. Um diese Komplexität zu verbergen, wird eine Abstraktionsschicht, bestehend aus einer Round-Robin DNS2 Technik für Lastausgleich, Fehlertoleranz und Lastverteilung, wird über dem Cluster verwendet. Der Server nimmt DNS-Anfragen an und leitet sie an eine Rechenmaschine im Cluster weiter. Eine Maschine aus dem Cluster wird dann im Round-Robin-Verfahren ausgewählt.

Schlüsselfunktionen des Version-Controllers:

Ausfallsicherheit

Die zweischichtige Architektur des VC-Servers bietet Schutz vor Single Point Failure. Falls ein Knoten im Cluster ausfällt, bleibt das asvin.io Framework betriebsbereit.

Skalierbarkeit

Der Cluster ist hoch skalierbar und stabil. Um die Leistung des VC-Servers zu erhöhen, kann einfach ein neuer Knoten im Cluster installiert werden.

Effizienz

Der Version Controller verteilt die Arbeitslast auf mehrere Webserver im Cluster, was die Netzwerkleistung verbessert und Latenz und Kollisionen in Zeiten hoher Nachfrage reduziert. Der Version Controller bietet Ausfallsicherheit für das asvin.io Netzwerk.

Funktionen des Servers:

1. **Abfrage nach Updates/ Firmware**

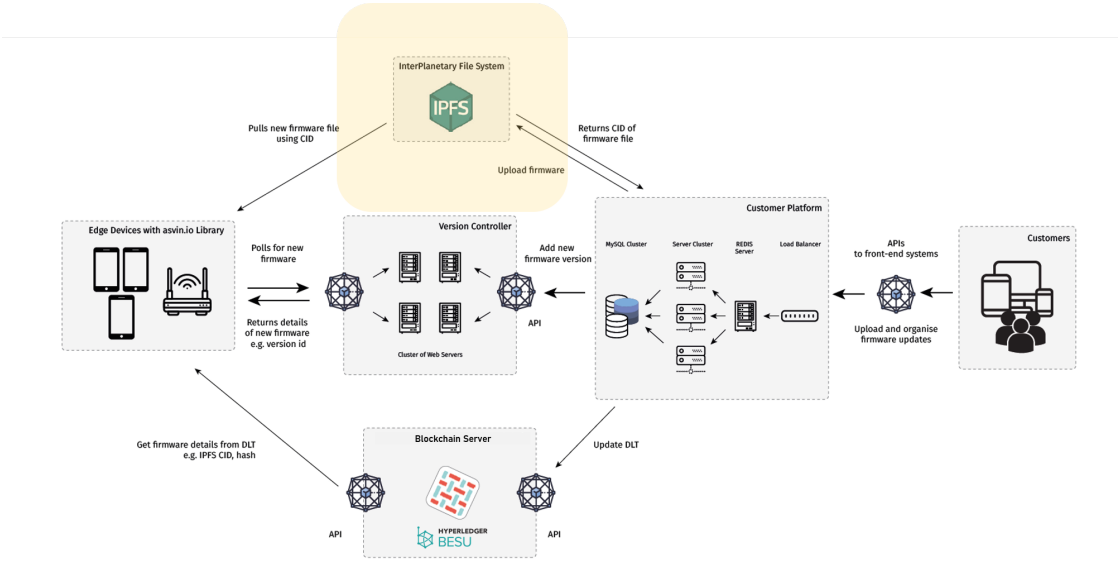
Die Edge-Devices fragen den Versionskontroller ab, um auf ein neues Update zu prüfen. Der VC und der Server antworten dem Edge-Device mit der Information über eine gültige Firmware.

2. **Verwaltung der Firmware-Versionen**

Der Version Controller verwaltet Echtzeit-Informationen über verschiedene Versionen der auf den Servers verfügbaren Firmware. Außerdem verfügt er über eine Liste der verfügbaren Firmware für alle Edge-Devices auf der asvin.io-Plattform. Er hält die Liste durch Interaktion mit dem beehive-Server aktualisiert.

2.5 Interplanetary FileSystem

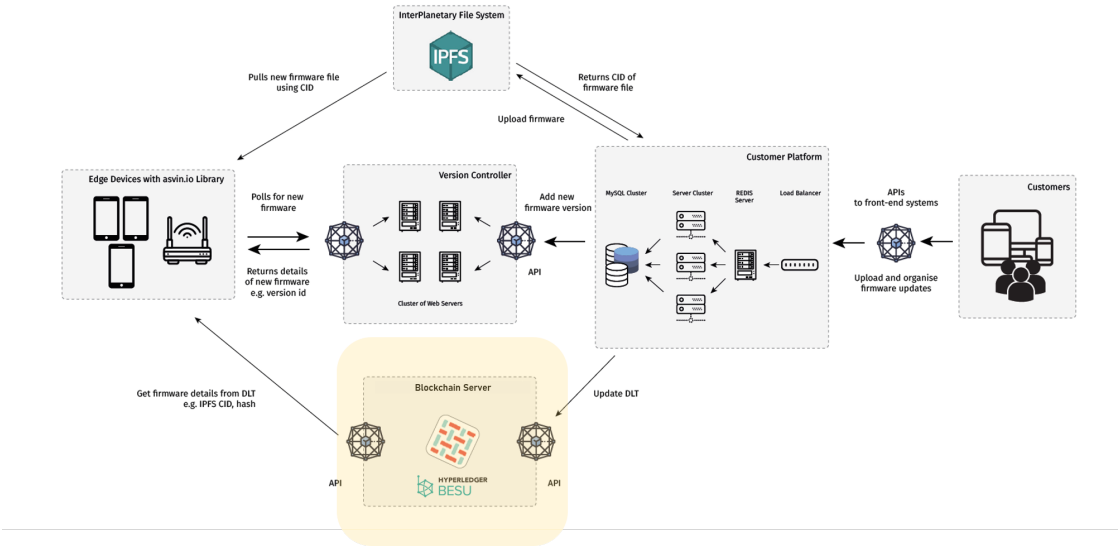
In diesem Abschnitt wird das **Interplanetary FileSystem (IPFS)** als Komponente der asvin Architektur kurz vorgestellt und erklärt.



asvin.io nutzt das Interplanetary File System Protokoll, um Firmware und Patches zu speichern. IPFS ist eine inhaltsadressierbare Peer-to-Peer-Methode (P2P) zum Speichern und Teilen von Hypermedia in einem verteilten Dateisystem, wobei doppelte Dateien und Redundanz vermieden werden. Es nutzt eine Netzwerkinfrastruktur, welche es asvin.io ermöglicht, unveränderliche Firmwaredaten zu speichern. Wenn eine Firmware-Datei im Netzwerk gespeichert wird, wird ein Hash basierend auf dem Inhalt der Firmware generiert und im Blockchain-Netzwerk gespeichert. Später wird derselbe Hash von Edge-Geräten verwendet, um die Firmware aus dem Datenspeicher zu extrahieren. Dies bildet einen verallgemeinerten Merkle gerichteten azyklischen Graphen (DAG). Jeder Knoten des Merkle DAG ist mit einem gesicherten Hash verbunden. Wenn ein Knoten im DAG hinzugefügt wird, wird sein Hash basierend auf dem Hash seines lokalen Inhalts und den Hashes der Namen seiner Unterknoten berechnet, anstatt deren Inhalt. Sobald er erstellt wurde, ist es unmöglich, einen Knoten im Netzwerk zu verändern. IPFS hat keinen Single Point of Failure. asvin.io bietet ein verteiltes Content-Delivery-System, welches eine zusätzliche Sicherheitsebene bietet und DDoS-Attacken verhindert. asvin.io's SDK, das auf Edge-Devices ausgeführt wird, ermöglicht es, mit Datenspeichern zu interagieren. Sobald ein Edge-Device vom Versions-Controller Informationen über die neu verfügbare Firmware erhält, nutzt es diese Informationen, um die entsprechende Firmware aus dem verteilten CDN herunterzuladen.

2.6 Blockchain

In diesem Abschnitt werden wir die **Blockchain** -Komponente der asvin Architektur erläutern.



Blockchain ist eine Art Distributed Ledger Technology (DLT), die eine wachsende Liste von Einträgen enthält, die als **Blöcke** bezeichnet werden. Jeder Block enthält den kryptographischen Hash des vorherigen Blocks. Durch ihr Design sind Blockchains beständig gegen Veränderungen der gespeicherten Daten.

In der asvin Architektur ist das Blockchain-Netzwerk auf Basis von Hyperledger Fabric und Hyperledger besu aufgebaut. Kunden können wählen, welche der beiden Blockchains sie für ihre Einsätze nutzen möchten. Diese Blockchain-Netzwerke stellen ein Ledger zur Verfügung, das alle Transaktionen speichert, die im asvin.io Netzwerk ausgeführt werden, z.B.: Geräteregistrierung, Firmware-Upload, Geräte-Update, Firmware-Update und Benutzerregistrierung. Jede dieser Transaktionen ist mit eindeutigen Hashes gepaart und wird in Blöcken gespeichert, die wiederum mit einem gesicherten Hash verbunden sind. Dieser Prozess bietet Sicherheit und Unveränderlichkeit für den Ledger. Die asvin.io Plattform auf eine unbegrenzte Skalierbarkeit mit einem Cluster von Servern ausgelegt, die Fabric & Besu Netzwerke betreiben, um die Anforderungen zu verwalten um so die Funktionalität für Millionen von IoT-Geräten aufrecht zu erhalten.

2.6.1 Hyperledger Fabric

Die steckbaren Module des Fabric-Netzwerks erlauben der asvin.io-Infrastruktur maximale Flexibilität, sodass asvin.io maßgeschneiderte Lösungen für seine IoT-Kunden entwickeln kann. Der Cluster wird mit Docker swarm. entwickelt.

Die Fabric hat folgende Module und Dienste: 1. Peer:

eine Entität im Fabric-Netzwerk, die Anfragen von Anwendungen empfängt, einen Chaincode ausführt, Transaktionen validiert und den Ledger verwaltet.

2. Orderer: Dieser Dienst ordnet alle Transaktionen, die im Fabric-Netzwerk stattfinden, und leitet sie an die Peers weiter, um sie zu validieren.
3. **MSP:** stellt jedem Mitglied des Fabric Netzwerks digitale Identitäten zur Verfügung.
4. **Storage:** Das Fabric-Netzwerk nutzt CouchDB, um den aktuellen Stand der Ledger-Daten zu speichern.

2.6.2 Hyperledger Besu

Hyperledger Besu ist ein Open Source Ethereum Client. Er kann auf dem öffentlichen Ethereum-Netzwerk oder auf privaten, zugelassenen Netzwerken betrieben werden.

Hyperledger Besu's Features

1. Die Ethereum Virtual Machine (EVM) ist eine vollständige virtuelle Turing-Maschine, die das

   Deployment und die Ausführung von Smart Contracts über Transaktionen innerhalb einer Ethereum-Blockchain ermöglicht.

2. Konsens-Algorithmen sind an der Validierung von Transaktionen, der Validierung von Blöcken und

   der Produktion von Blöcken (d.h. dem Mining in Proof of Work) beteiligt.

3. Zur Speicherung wird die Key-Value-Datenbank RocksDB verwendet, um die Daten der Kette lokal

   zu persistieren.

4. P2P Networking über Ethereums devp2p Netzwerkprotokolle für die Kommunikation zwischen den Clients

   und ein zusätzliches Unterprotokoll für IBFT2.

5. Benutzerfreundliches Mainnet Ethereum und EEA JSON-RPC APIs über HTTP und WebSocket Protokolle sowie eine GraphQL API.

6. Die Node Performance wird mit Prometheus oder der debug_metrics JSON-RPC API Methode überwacht.

   Die Netzwerk-Performance wird mit Alethio-Tools wie dem Block Explorer und EthStats Network Monitor überwacht.

7. Privacy in Hyperledger Besu bezieht sich auf die Fähigkeit, Transaktionen privat zwischen den beteiligten

   Parteien zu halten. Andere Parteien können nicht auf den Inhalt der Transaktion, die sendende Partei oder die Liste der beteiligten Parteien zugreifen. Besu verwendet einen Private Transaction Manager, um Privatsphäre zu implementieren.

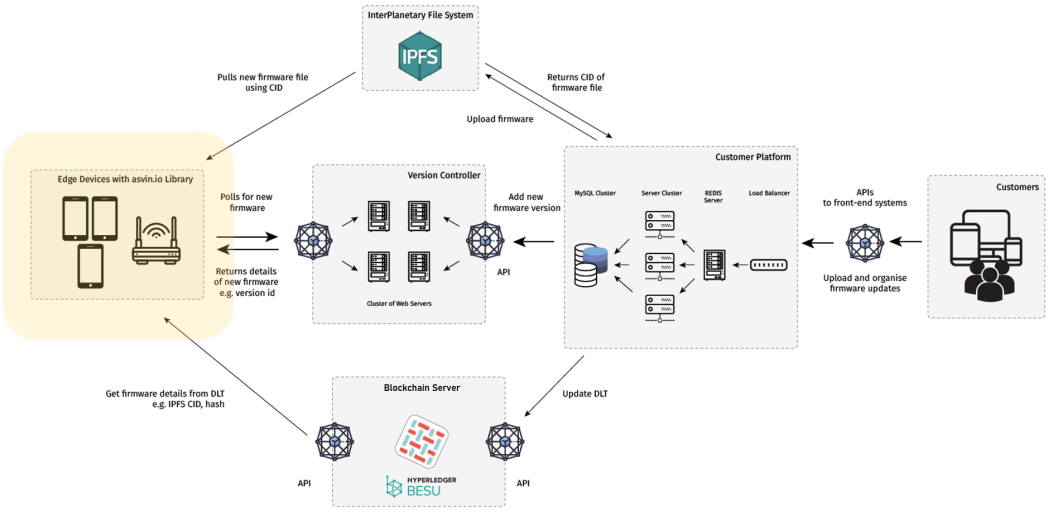
8. Ein berechtigtes Netzwerk erlaubt nur bestimmten Knoten und Accounts die Teilnahme, indem es Node

   Permissioning und/oder Account Permissioning im Netzwerk aktiviert.

Die Virtualisierung auf Betriebssystemebene wird auf dem Blockchain-Server mit Docker erreicht. Jeder Dienst im Netzwerk läuft in einem separaten Docker-Container und diese Container werden auf mehreren Maschinen des Clusters gehostet. Die Kommunikation zwischen den Containern wird mit Docker Swarm erreicht. Das gesamte Blockchain-Netzwerk wird mit der Docker Swarm Technologie entwickelt und betrieben.

2.7 Endgeräte oder Edge Devices

Es folgt eine kurze Erklärung der Funktion **Edge Devices** innerhalb der asvin Architektur.



Die Edge Devices sind Endpunkte in der asvin.io-Architektur und haben in einem IoT-Netzwerk die Aufgabe, eine physikalische Aufgabe zu steuern, zu verwalten und zu lösen. In der Regel sollen Edge-Device einen kleinen Footprint haben und müssen in abgelegenen Gebieten unter oft extremen Umweltbedingungen einfach zu verwalten sein. Zum Beispiel: industrielle Prozessüberwachungssensoren, Smart Meter, Lora-Knoten etc. asvin stellt Bibliotheken für eine Vielzahl von Edge-Geräten zur Verfügung, um die technischen Anforderungen zu erleichtern, die notwendig sind, um sicher und effizient mit der asvin.io-Infrastruktur zu interagieren. Die asvin Bibliotheken fungieren als Schnittstelle zwischen den Edge Devices und der asvin.io Plattform. Das asvin.io SDK ermöglicht die einfache Nutzung von APIs zur Interaktion mit der asvin.io Update-Infrastruktur. Die SDKs sind anwenderfreundlich und ermöglichen es neuen IoT-Geräten, sich schnell mit der asvin.io-Infrastruktur zu verbinden, um direkt von einem einsatzbereiten Update-Verteilungsnetzwerk zu profitieren. Das Hauptmerkmal des asvin SDKs ist, dass es sich um eine generische Lösung handelt und nicht auf eine bestimmte Hardwareplattform wie z.B. Arduino, ESP, STM, Arm Cortex-M3, M4 etc. und Softwareprotokolle beschränkt ist.

The asvin.io libraries installed on edge device provides following functionalities:

1. **Device Update Management**
    - Securely register devices on asvin.io platform
2. **Check for Firmware Updates**
    - Regularly poll for updates in configurable time intervals
3. **Download and Install Firmware**
    - Download and store updated versions of firmware
    - Install the firmware based on traffic and availability of edge device
4. **Collect Health Statistics**
    - Send timely health updates of edge device on asvin.io platform e.g. firmware version in use
    - asvin.io platform generates insights from the data which can be used to extend life cycle of IoT devices and for preventive maintenance.

Funktionen der auf dem Edge Device installierten asvin.io Bibliotheken:

1. **Device Update Management** - Sichere Registrierung von Geräten auf der asvin.io Plattform
2. **Prüfen auf Firmware Updates** - Regelmäßige Abfrage nach Updates in konfigurierbaren Zeitintervallen
3. **Herunterladen und Installieren von Firmware** - Herunterladen und Speichern von aktualisierten Versionen der Firmware - Installiert die Firmware basierend auf dem Traffic und der Verfügbarkeit des Edge Devices.

4. **Erfassen von Health Statistics** - Sendet zeitnahe Health Statistics des Edge Devices an die asvin.io Plattform, z.B. die verwendete Firmware Version - Die asvin.io Plattform generiert Erkenntnisse aus den Daten, die zur Verlängerung des Lebenszyklus von IoT-Geräten und zur vorbeugenden Wartung genutzt werden können.

KAPITEL 3

Getting Started

3.1 Prerequisites

Before your embark on a journey to explore asvin getting started and tutorials sections, it is necessary to install the following tools on your machine.

3.1.1 Postman

Download and install the latest version of postman. The postman will be used to make http requests to asvin servers. We have a postman collection of asvin APIs. You could create a test workspace and import the collection. It is required for the getting started documentation. The following video shows the procedure.

The Postman is enough to execute all the steps listed in the getting started section. If you want to try any of the tutorials then install the tools mentioned below.

3.1.2 Git

Download and install the latest version of git.

3.1.3 cURL

Download and install the latest version of cURL tool.

3.2 Account Registration

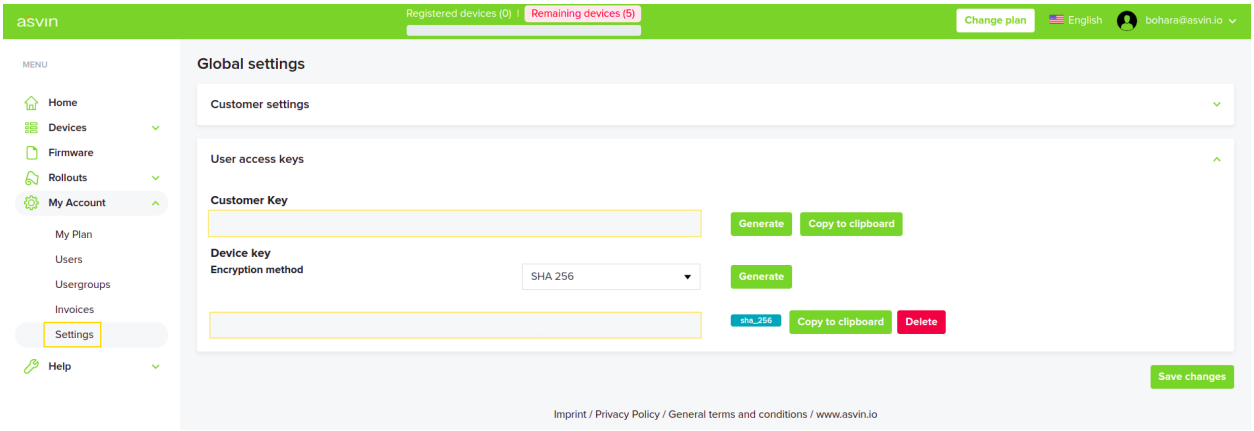
asvin Beehive provides an interactive dashboard to visualize and manages devices, firmware, and rollouts. It is necessary to create an account on the asvin Beehive to use asvin services. The process is quite simple. All you need is a valid email address. Go to the register page, input your email address, select couple of check boxes and click on Register now. It is shown below.

When everything goes well, you will receive an email for setting up password for your account. Click on Account Activation, it will take you to a web page. There you can create your password. You need to take care of asvin's password guidelines.Your password should contain

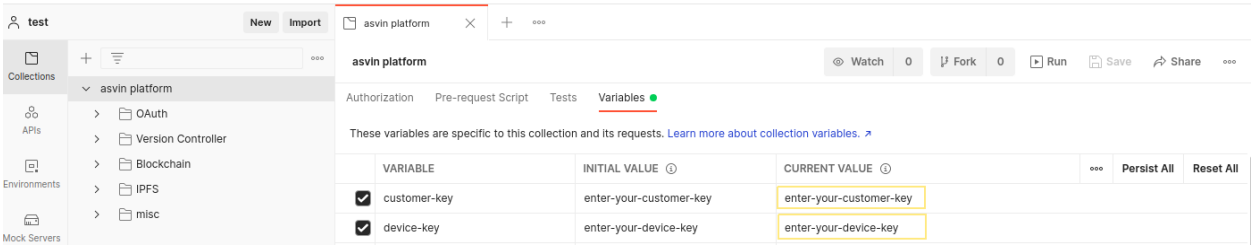
- at least 10 characters
- at least one big letter and one small letter
- at least one number
- at least one special character

Once you enter valid password and click on Create a password button, it will land you to the home page of the dashboard. The process is shown below.

asvin uses access keys to manage unauthorized access to the api end-points. Theses access keys are customer key and device key. The keys are generated automatically when an account is created for the first time. You can access the keys in My Account -> Settings section as shown in the picture below.



The keys will be used to get Jason Web Token (JWT) from OAuth server. Copy the access keys and set the respective collection variables in the Postman as shown in the image below.



3.3 Device Registration

At this stage, you have created an account on the asvin Beehive, received the customer and device keys, and imported asvin api collection on postman application. It is time to register your first device. Go to you postman collection and open Register Device POST request located in the Version Controller folder. In the request, enter desired device name, current firmware version, and valid MAC address. Next, all you have to do is click on big blue button named Send. You will received device inserted message if the request is accepted. You can go the dashboard and check it in Devices menu. It will appear under Lobby sub-menu. The process is shown in video below.

Next, you need to create a device group and add the device in it to complete the registration. You will not be able to create rollout for the device until you add it in a device group. Click on New device group button under Devices->Grouped menu to add a device group. Once you have a device group, go to Lobby, select the device and click on Group to add it in a desired device group. The procedure to add a device group and grouping a device is shown below.

At the end of this task, you will have a device registered, grouped in a device group and stored on distributed ledger.

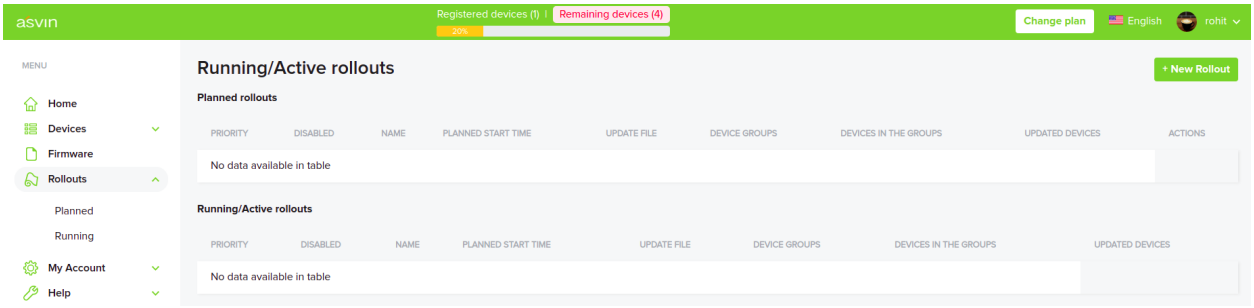
3.4 Firmware Upload

If you have completed the device registration process, then you must have a device in your account. Next, you need a firmware for the device. This can be achieved in two steps. Firstly, create a firmware group by clicking on create new file group button under Firmware menu. When you have a firmware group with a desired name then click on show button or on the name. It will show you list of firmware in the group. Obviously, it is empty now. Secondly, create a simple firmware and click on Upload new file button to upload it. The steps are shown in the video below.

You will have a firmware group and firmware in it at completion of the task. The firmware metadata will be stored on the distributed ledger and firmware will be saved on the private distributed network powered by IPFS protocol.

3.5 Rollout Creation

Once you have a you registered a device and upload a firmware successfully, it is time to create and start a rollout. You can access all your rollouts under Rollouts menu. It contains list of planned and running rollouts. Initially, the list will be empty as shown in the figure below.

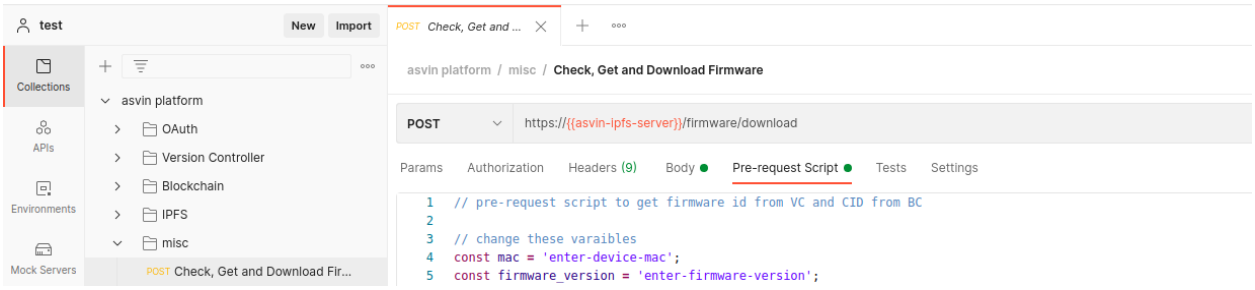


Click on New Rollout button to create a rollout. It will open a dialog box where you need to select relevant device group, firmware file and start time. If everything goes well, you will have a rollout in the Planned list. You can click on Start button to make the rollout active. The procedure is shown below in the video.

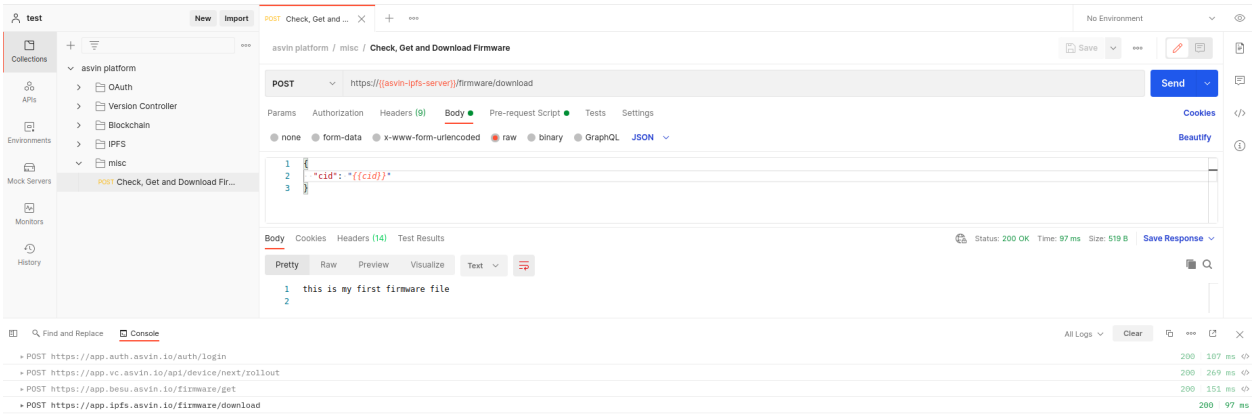
An active rollout is the result of the task. The devices can use it to get firmware details which are utilized to download firmware from IPFS server.

3.6 Firmware Download

After you have an active rollout in your account, you can download the associated firmware. We have created a request in misc folder in the asvin postman collection. The request is named Check, Get and Download Firmware. Firstly, open the request and go to Pre-request Script tab. Secondly, enter valid MAC address and firmware version as shown in the image below. Lastly, click on Send button to send the request.



The one click firmware download request is more complicated than it seems. Under the hood, it contains 4 more requests. First of all, it sends login request to OAuth server to get JWT. The JWT is used in all subsequent requests for authorization. After that it checks for an active rollout. If a rollout exist the Version Controller returns firmware and rollout ids. The firmware id is utilized to get firmware CID from the Blockchain Server. Finally, it sends download firmware request with CID to the IPFS server. You can see the details of all requests in the Postman Console. It is depicted in the image below.

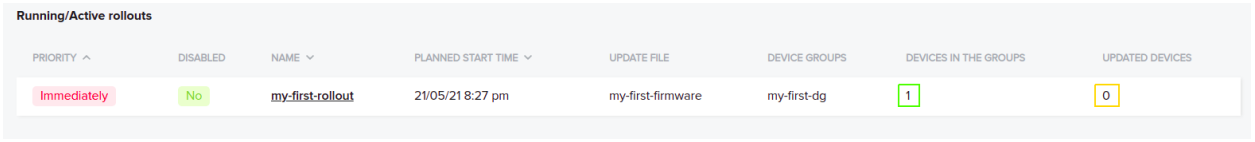


You can also make the requests manually one by one. It is illustrated in the video below.

You will have your first firmware downloaded at the completion of the task.

3.7 Rollout Success

Now that you have downloaded a firmware, you can install it on the device. Once the installation is completed and device works as expected, you should send rollout success request to the Version Controller. A rollout is associated with a device group. Therefore, an active rollout shows the number of devices in the groups and updated devices as shown in the picture below.

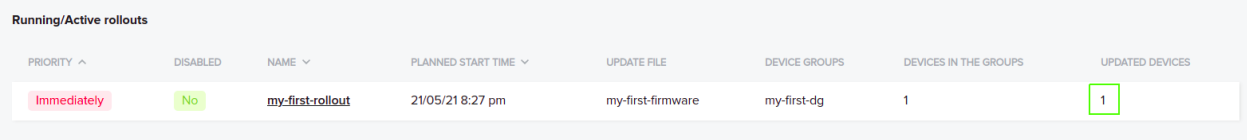


In the beginning, an active rollout has 0 updated devices. It gets updated when a rollout success request is sent with valid parameters. You can send the request using postman collection. It is in the Version Controller folder with the name Rollout Success. The request requires 4 parameters in the body. They are device mac address, firmware version and rollout ID. You must have received rollout ID in response to the next rollout request. The response looks similar to the following.

```
{
  "rollout_id": "63",
  "rollout_name": "my-first-rollout",
  "priority": "1",
  "start_date": "2021-05-21 18:27:31",
  "version": "1.0",
  "firmware_id": "57"
}
```

The rollout success request is shown in the video below.

Once the rollout success request is completed, the updated device count is changed in the rollout as shown in the image below.



Before we start, it is required that you install the *Prerequisites* on the machine on which you will execute the instructions given in the getting started documentation. Once you are done with the prerequisites, you are well equipped to work with and experience asvin platform. Next, follow the steps given below.

- *Create and activate your account* on Customer Platform and get global access keys
- *Register* your first device and add it in a device group
- *Make a file group* and add your first firmware in it
- *Plan a rollout* for your device with the first firmware
- *Check next rollout, get CID and download firmware* from VC, BC and IPFS server respectively
- Send *Rollout success* request

KAPITEL 4

Tutorials

Dieser Abschnitt enthält Tutorials für die Verwendung der Asvin-Plattform.

4.1 Over the air updates mit ESP8266 (nodemcu) boards

In diesem Tutorial sehen wir die Demonstration von OTA-Updates unter Verwendung der asvin IoT-Plattform und des nodemcu-Boards basierend auf dem ESP8266-Chipsatz.

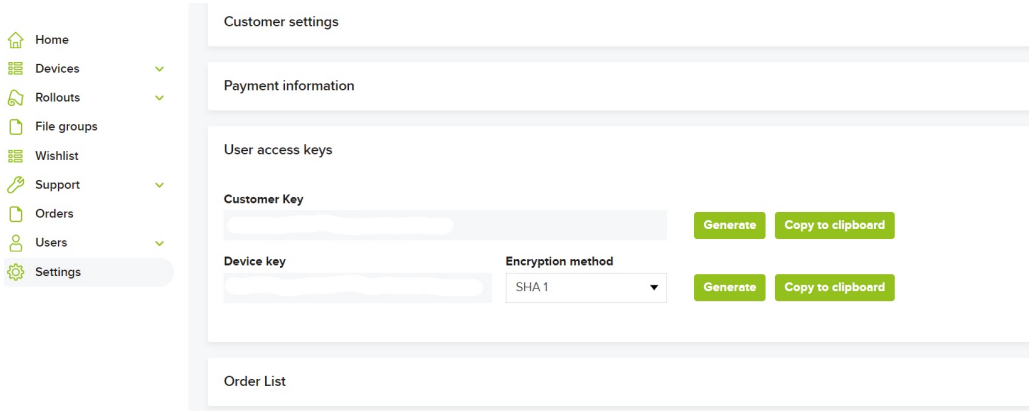
4.1.1 Anforderungen

1. nodemcu esp8266-Platine
2. Micro USB Kable
3. asvin platform Zugang
4. PlatformIO VScode extension

4.1.2 Erste Schritte

Gehen Sie als als erstes zum Github repository von asvin und klonen Sie es. Der Code ist unter PlatformIO geschrieben. Importieren Sie das Projekt mit der PlatformIO-Erweiterung von VScode.

1. Um fortzufahren, müssen Sie einige der Parameter im Code bearbeiten
   - Öffnen Sie die Datei main.cpp im Editor und fügen Sie die Zugangsdaten für Ihr Gerät hinzu.

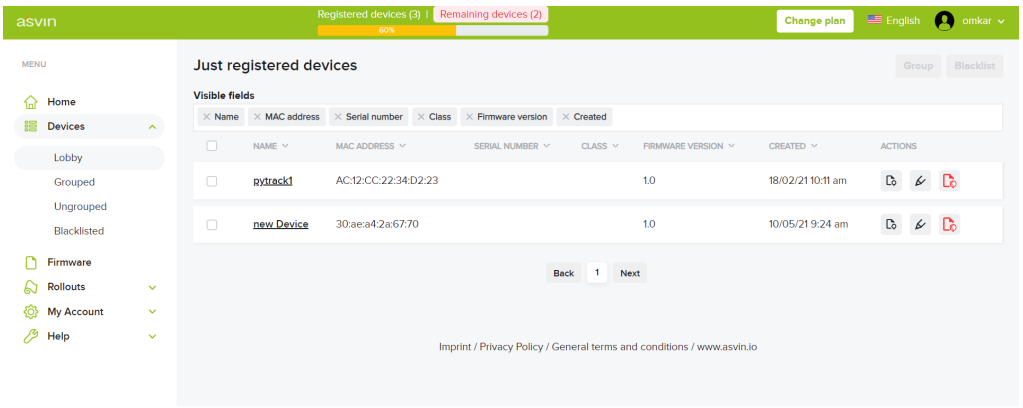


   - Geräte- und Kundenschlüssel: Diese finden Sie auf der asvin-Plattform unter der Registerkarte „Einstellungen“. Diese Schlüssel werden verwendet, um ein Auth-Token zu generieren.
2. Nach diesem Schritt erstellen Sie den Code für Arduino oder PlatformIO und laden ihn auf das ESP8266-Board hoch
3. Dieser Sketch verwendet die populäre WifiManager-Bibliothek , um die WiFi-Berechtigungsnachweise zu verwalten. Nach dem Booten wird der ESP8266 einen WiFi-Hotspot mit dem Namen „AutoConectAP“ starten. Der Benutzer sollte sich mit seinem Handy/Laptop mit diesem verbinden und seine Zugangsdaten eingeben. Diese Zugangsdaten werden im Flash-Speicher des ESP als Standard gespeichert. Diese Zugangsdaten können später geändert werden.
4. **OTA einrichten**

   Die asvin IoT Plattform bietet sichere OTA Updates für IoT Geräte. Folgend wird gezeigt, wie Updates eingerichtet werden können.

   1. *Gerät registrieren*:

      Wenn sie den Sketch starten und auf das ESP8266 Board hochläden, beginnt das Board mit der Ausführung und ruft die definierten API Routen auf. Die erste API, die es aufruft, ist *Register Device*, Nachdem diese API erfolgreich aufgerufen wurde, finden Sie ihr Gerät auf der Plattform unter Geräte>Just registered Devices.

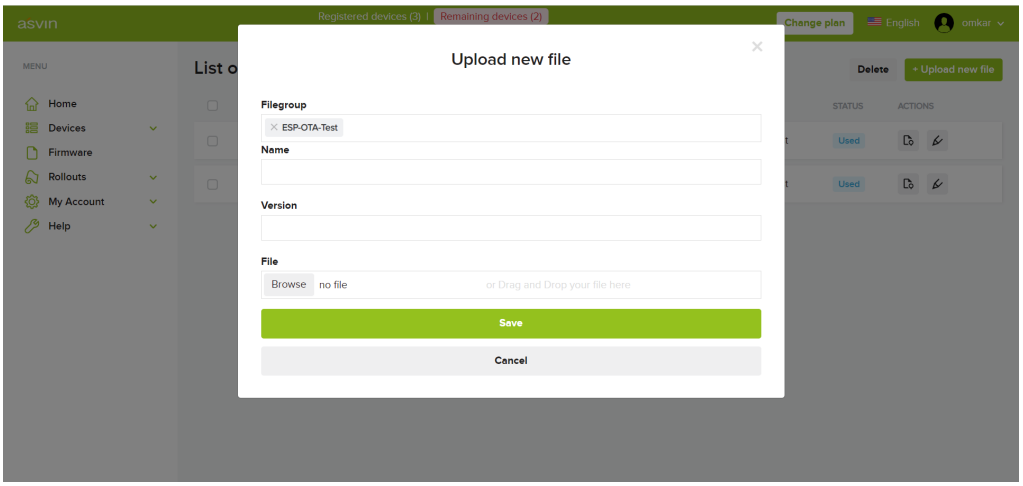


2. *Gerätegruppen*:

   asvin's IoT Plattform bietet Updates für eine Gruppe von Geräten. Das ESP Gerät kann zu dieser Gruppe hinzufügt werden. Gehen Sie hierzu auf Geräte>Gerätegruppe auf *Neue Gerätegruppe*. Anschließend navigieren Sie zurück zu *Lobby*, klicken auf Geräte gruppieren und fügen das Gerät zurneu erstellten Gerätegruppe hinzu.

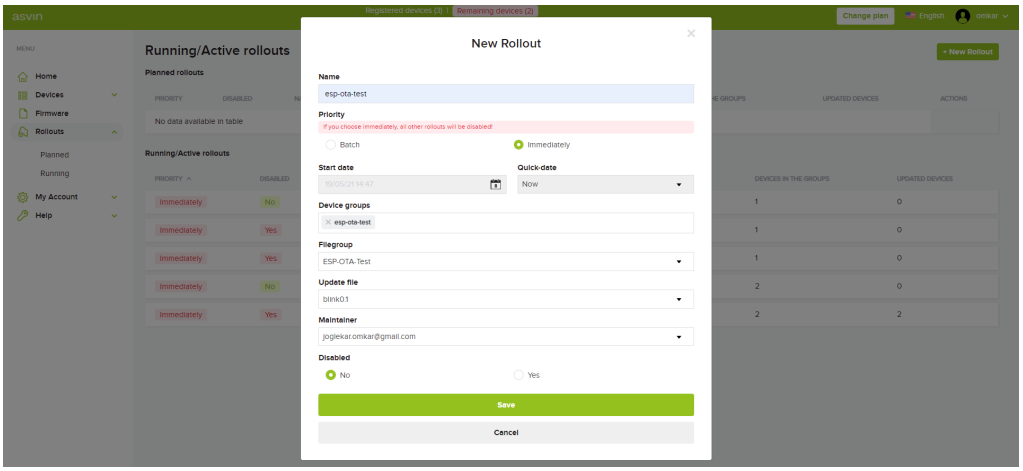
3. *Dateigruppen*:

   Sobald das Gerät einer Dateigruppe zugewiesen ist, kann die Datei hochgeladen werden, die wir als OTA-Update zur Verfügung stellen wollen. Normalerweise handelt es sich um eine Datei mit dem Namen *<Dateiname>.bin*. In diesem Beispiel wird die Datei esp-ota-blink.bin in die Dateigruppe ESP_OTA_Test hochgeladen.



4. *Rollout*:

   In diesem Schritt werden wir einen Rollout einrichten, um ein OTA-Update der oben angegebenen Datei an unser ESP8266-Gerät zu liefern. In der Rollout Sektion beginnen wir mit der Erstellung eines Rollouts. Fülle die Optionen wie im Screenshot gezeigt aus. Wähle entweder Batch oder sofortiges Update. Es gibt eine Option, eine Zeit zu wählen oder ein Update sofort durchzuführen. Wähle die Datei aus, die als Update ausgerollt werden soll und klicke auf *Speichern*.



5. Das Rollout ist nun aktiviert. Wenn das Gerät das nächste Mal die *Next Rollout* API abfragt, wird der Rollout verfügbar sein und weitere APIs werden im ESP-Gerät aufgerufen. Das ESP-Gerät wird sich danach mit der Datei aktualisieren, die wir zuvor hochgeladen haben. In diesem Fall werden wir die LED auf unserem ESP-Board blinken sehen.

6. Sobald der Rollout abgeschlossen ist, wird die neue Datei auf dem Board ausgeführt. In diesem Fall haben wir eine Blink-LED-Datei ausgerollt. Das Board wird die *Rollout Success* API aufrufen, die Teil der Datei esp-ota-blink.bin ist, die wir zuvor hochgeladen haben.
7. Die Änderung der Firmware-Version des Geräts wird auch auf der asvin platform aktualisiert.

Damit haben wir den OTA-Rollout erfolgreich abgeschlossen. Den vollständigen Code und die Dateien finden Sie in asvins Github repository .

4.2 Over the air updates mit Pycom Gpy boards

In diesem Tutorial sehen wir die Demonstration von OTA-Updates mit der asvin IoT-Plattform und dem *Gpy* Board von Pycom.

4.2.1 Requirements

1. Pycom Gpy board
2. Pycom Erweiterungskarte
3. Micro USB Kabel
4. Asvin plattform Zugang
5. Pymakr VScode-Erweiterung

4.2.2 Erste Schritte

Um zu beginnen, gehe zu asvin's Github repository und klone es. Öffne den Ordner *pycom-ota-updates* in VScode. Achte darauf, dass Sie die aktualisierte Firmware auf deinem Pycom *Gpy* und dem *Expansion* Board haben.

1. *Beschreibung der Dateien*:

   Dieses Repository enthält ein paar Python Sketches. Unten ist eine kurze Beschreibung von ihnen

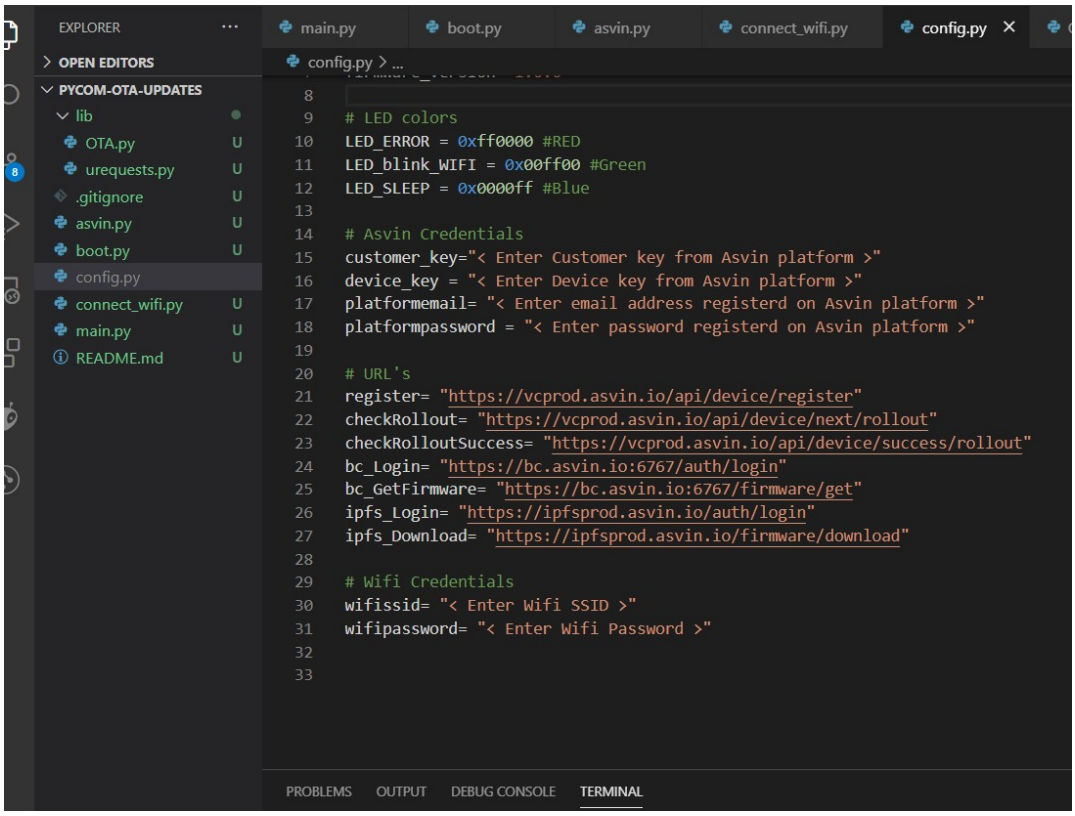
   lib/OTA.py Diese Bibliothek ermöglicht OTA-Updates. Sie wird unten im Detail besprochen.

   connect_wifi.py Dieses Skript ist ein Wrapper um die pycom Wlan() Bibliothek

   asvin.py Diese Datei enthält Funktionen, um verschiedene API's der Asvin Plattform aufzurufen.

   config.py Diese Datei enthält verschiedene Benutzerkonfigurationsoptionen und wird im nächsten Abschnitt besprochen

2. Als nächstes müssen einige Parameter in der config.py Datei überarbeitet werden.
   - Öffne die config.py Datei im Editor und füge die Anmeldedaten für dein Gerät hinzu



- **Ergänzen Sie unter "Asvin Credentials" folgende Informationen**
  - customer_key: siehe asvin Plattform
  - device_key: siehe asvin Plattform
- Ergänzen Sie unter „WiFi Credentials" folgende Angaben SSID und Passwort
- Zusätzlich kann die Farbe der LED in der Konfigurationsdatei eingestellt werden.

3. Laden Sie nun das Projekt aufs Pycom Board hoch
4. **Der Code führt folgende Schritte aus: :**
   - **WiFi-Verbindung**
     - Prüfung, ob der vorherige *Rollout* erfolgreich ausgeführt wurde
   - Registrierung des Gerätes durch Aufruf der *Register Device* API.
   - Prüfung auf neuen Rollout Rollout vorhanden!
   - Herunterladen und Ausführung des Updates.
5. **Einrichtung des OTA**

   Follow the steps below along with the ../getting-started/customer-platform guide.

   1. ***Gerät registrieren*:** Die Registrierung erfolgt automatisch beim Booten
   2. ***Gerätegruppe*:** Bitte richten Sie eine Gerätegruppe auf der asvin Plattform ein.
   3. *Dateigruppen*:

      Bevor Dateien zum Rollout in eine Dateigruppe hochgeladen werden können, müssen bestimmte Änderungen an Dateien vorgenommen werden: Integrieren der folgenden zwei Zeilen zu Beginn des Codes

      ```
      path="/flash/config.py"
      version = "0.0.1"
      """
      Asvin OTA Config File
      """
      ```

      In diesem Fall ist die Path-Variable der Pfad der Variable im Dateisystem von Pycom. Die Version ist die benutzerdefinierte Versionsnummer der vorhandenen Datei.

   4. ***Rollout*:** Setup the rollout as mentioned in the Getting Started guide. In this case it is important to follow the guidelines mentioned under *File Groups*.

      Richten Sie das Rollout wie in der Anleitung Erste Schritte beschrieben ein. In diesem Fall ist es wichtig, die unter Dateigruppen genannten Richtlinien zu befolgen.

Der OTA-Rollout für das Pycom Gpy-Board wurde erfolgreich abgeschlossen. Den vollständigen Code und die Dateien finden Sie in asvin's Github repository.

4.3 Over-The-Air update of ESP32

This tutorial demonstrates the process of securely updating ESP32 device firmware Over-The-Air with asvin platform.

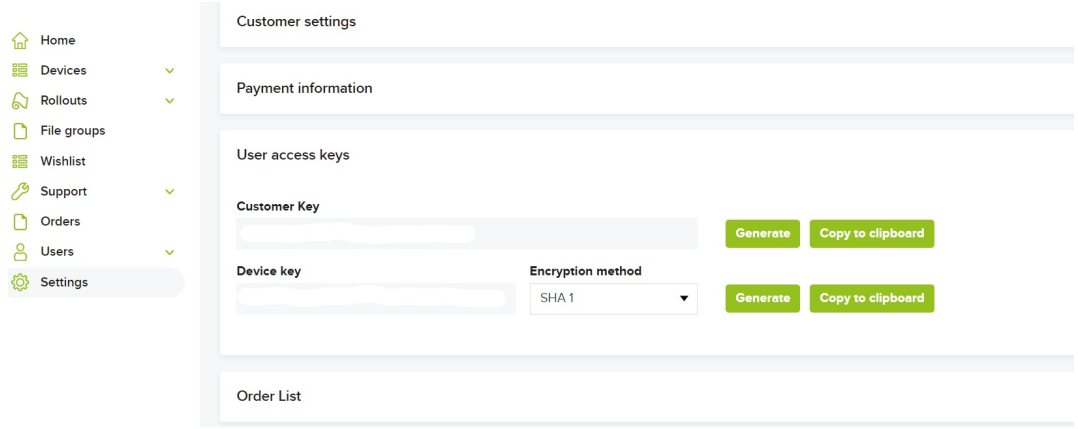
4.3.1 Requirements

1. ESP32 board
2. Micro USB cable
3. asvin platform subscription
4. PlatformIO VScode extension

4.3.2 Setup

To get started head to asvin's Github repository and clone it. Open the folder ESP32-OTA in Visual Studio code. The code is writen under PlatformIO.

1. Rename credentials_copy.h to credentials.h in the src folder.
2. Update customer_key, device_key which can be obtained from asvin platform as shown below.

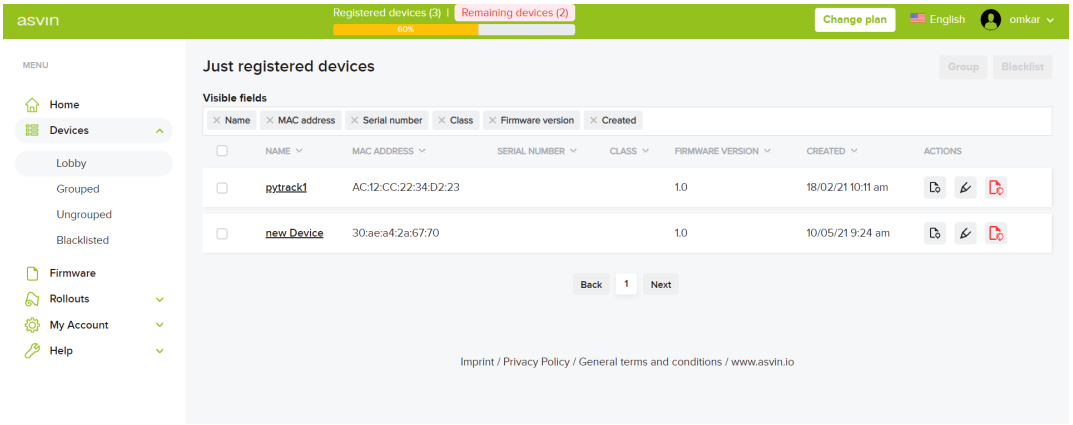


3. Then build the application and flash it to ESP32.
4. This sketch uses the popular WifiManager library to manage WiFi credentials. Upon booting the ESP32 will start a WiFi hotspot with the name „AutoConectAP". The user should connect to it with cellphone/laptop and enter in their WiFi credentials. These credentials will be stored in the ESP flash memory and will be stored as the default. These credentials can be changed later on.

4.3.3 OTA update procedure

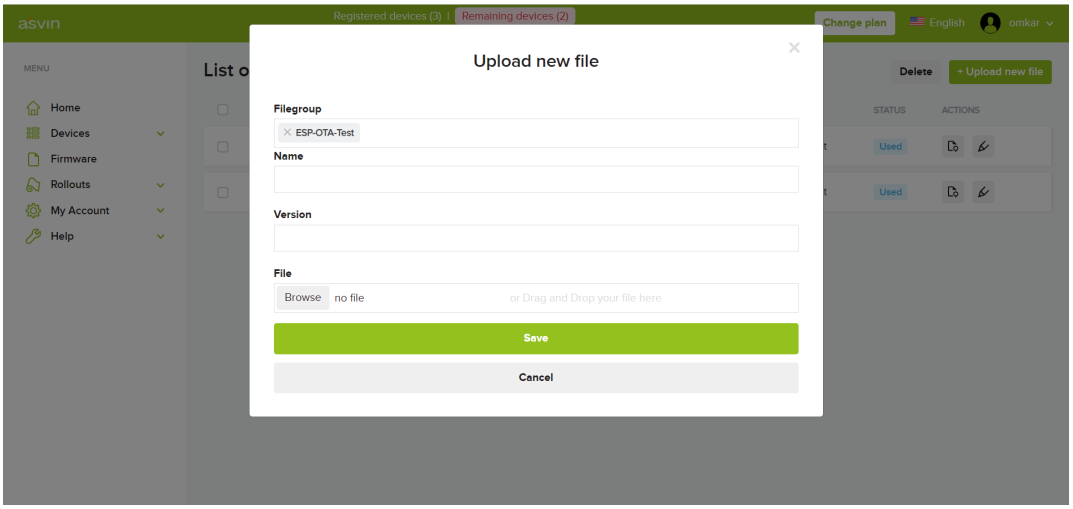
The asvin platform provides secure OTA updates for IoT devices. The user can follow the below easy steps to update their IoT edge devices.

1. *Register Device*: When you upload your sketch on the ESP32 board and start it, the device will start executing and calling set of defined API routes. The device first obtains the oauth token from the asvin oauth API. This token will be valid for next 10 mins. After obtaining the oauth token, the device calls the API *Register Device*, Once this API call is successful, you will see your device appear under the „Lobby" subsection of the „Devices" section in the platform.

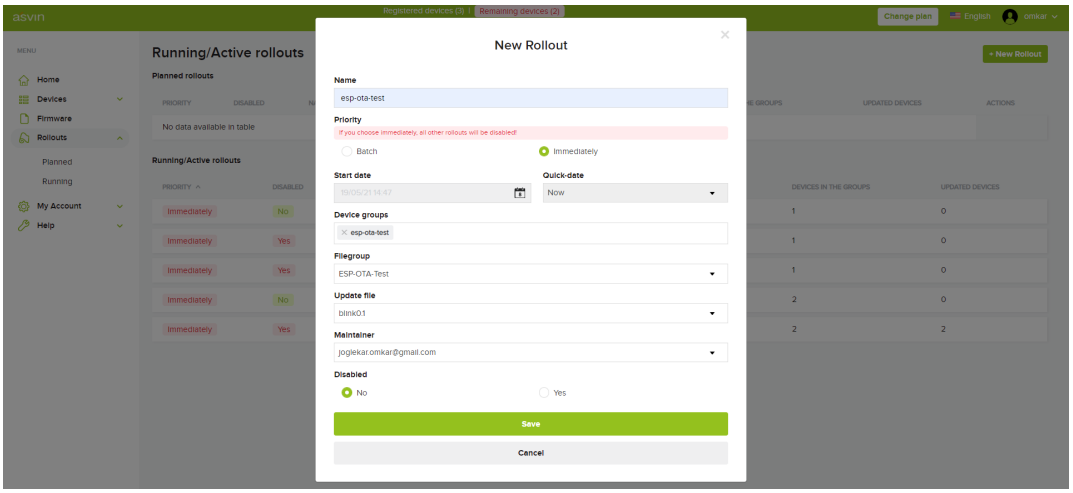


2. *Device Groups*: asvin's IoT platform provides updates for a group of devices. Let us create a group called ESP32Devices. We can add our ESP device to this group . Under *Devices -> Grouped* click on *„New Device Group"*. Then give the group name and save it. After this navigate back to the „Lobby", click Device Grouping and add the device to the newly created device group.

3. *File Groups*: Now we have to upload the file we want to provide as an OTA update. Usually this is *<file_name>.bin*. Let us upload esp-ota-blink.bin file to the filegroup ESP_OTA_Test.



4. *Rollout*: In this step we will setup a rollout to deliver OTA update of the file specified above to our ESP32 device. In the *Rollouts* section let's start by creating a rollout. Fill in the options as shown in the below figure. Choose either batch or immediate update. There is an option to choose a time or to do an update immediately. Select the file to be rolled out as an update and click *Save*.



5. The rollout is now enabled. Next time our device queries the *Next Rollout* API, the rollout will be available and further API's will be called inside the ESP device. The ESP device will update itself by downloading the file from asvin IPFS server. After successful update, we will see the LED blinking on the ESP board.
6. Once the rollout is completed the new application will be running on the board. In this case we rolled out a Blink LED application. The board will call the *Rollout Success* API, which is the part of the esp-ota-blink.bin file that we uploaded earlier.
7. The change in the firmware version of the device is also updated on the asvin platform .

Thus we have sucessfully completed the OTA rollout. The Complete code and files can be found at asvin's Github repository Github repository

4.4 Fed4FIRE+ Experimente

Das Tutorial demonstriert die Schritte, die zum Starten und Verwalten der Fed4FIRE+-Experimente für die Durchführung von Stresstests der asvin.io-Plattform erforderlich sind, indem virtuelle Endgeräte mit dem asvin API-Stack simuliert werden. Diese Vorgehensweise erleichtert die detaillierte Analyse der Skalierbarkeit und Zuverlässigkeit der Plattform.

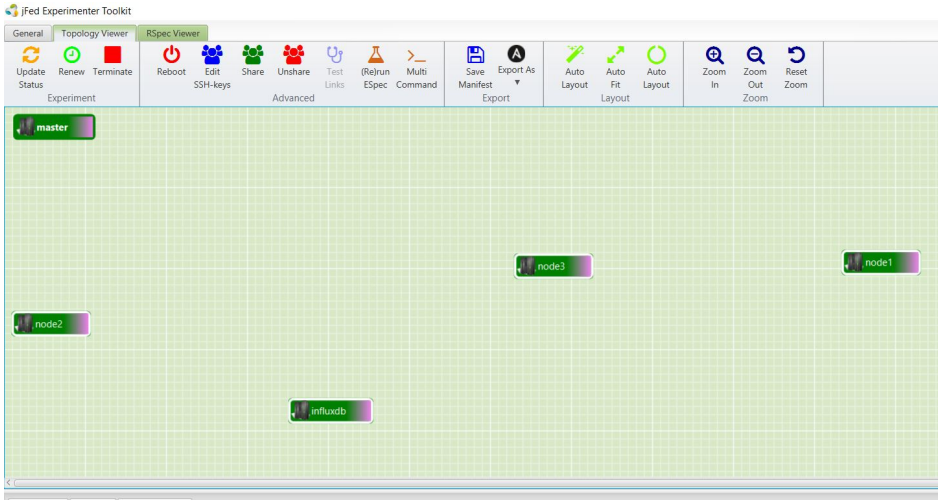
4.4.1 Voraussetzungen

1. jFed Experimenter Toolkit
2. ESpec Generator
3. asvin API stack image
4. Grafana json file

4.4.2 Erste Schritte

1. Start des Experiments

   Zum Start und zum Überwachen der Fed4FIRE+ Experimente findet das jFed Experimenter Toolkit Anwendung. Es ist ein benutzerfreundliches Tool mit einer einfachen und selbsterklärenden Oberfläche. Loggen Sie sich mit Ihrem Fed4FIRE+ Konto in das Experimenter Tool ein. Zum Erstellen und Starten der Experimente kann einfach die Drag and Drop Funktion genutzt werden. Alternativ können die Experiment-Spezifikationen (ESpec) erstellt und in das Toolkit hochgeladen werden.



   Zur Erstellung der ESpec wird das Tool ESpec Generator Github repository, das im Github-Repository ESpec Generator verfügbar ist. Mit diesem Tool kann ESpec nur für die Testbeds Virtual Wall1 und Wall2 generiert werden. Es enthält alle Installationsskripte des Kubernetes-Clusters, der Influx-DB und anderer Tools, die für den Stresstest der Plattform erforderlich sind.

```
generate-espec> python2.exe .\main.py --help
usage: main.py [-h] [--nodes [NODES]] [--no-control-server] [--gateway]
               [--wall {wall1,wall2}]

Generate espec for kubernetes

optional arguments:
  -h, --help            show this help message and exit
  --nodes [NODES]       amount of nodes in the generated espec, not including
                        the master node
  --no-control-server   Do not include the code to provision and setup a
                        control server with influx, grafana, private docker
                        registry and control website
  --gateway             add a gateway + apache server for delay testing
  --wall {wall1,wall2}  Target Virtual Wall, defaults to wall2
generate-espec> python2.exe .\main.py --wall wall2 --node 100
```

   Vor dem Ausführen des ESpec-Generators müssen die Python-Voraussetzungen durch Ausführen der folgenden Zeile im Generatorordner installiert werden

```
pip install -r requirements.txt
```

   Mit der generierten ESpec kann nun das Experiment auf Fed4FIRE+ Testbeds gestartet werden. Das Experiment besteht aus einem Master-Knoten, einen influxdb-Knoten und weiteren Arbeitsknoten Wenn die ESpec für 5 Knoten generiert wurde, dann besteht das Experiment aus einem Master-Knoten, einem influxdb-Knoten und 5 Worker-Knoten.

   Wenn Experimente mit mehr als 15 Knoten durchgeführt werden sollen, kontaktieren Sie bitte das Fed4FIRE+-Team, bevor Sie das Experiment starten.

2. Einrichten des Controll-Servers

   Nach dem erfolgreichen Start des Experiments mit einem Kubernetes-Cluster muss der Benutzer den Kontroll-Server aus dem Experiment-webserver-Github-Repository herunterladen und auf dem Master-Knoten des Kubernetes-Clusters bereitstellen.

   Befolgen Sie die im Repository angegebenen Schritte. Anschließend ist die Website des Kontrollservers über die öffentliche IPv6-Adresse des Servers erreichbar.

3. Bereitstellen des asvin API-Stack-Images

Der Beispiel-Python-Code, mit dem der API-Stack zur Simulation des Edge-Geräts ausgeführt wird, wird im asvin Github repository. von asvin bereitgestellt. Der Benutzer muss die Anmeldeinformationen für den Blockchain-Server und IPFS-Login, Benutzerschlüssel und Geräteschlüssel in der Datei „UserDetails.json“ angeben.

Das Image benötigt 2 Benutzereingaben:

- Anzahl der auszuführenden Threads
- Der Server (Produktion oder Staging)

Standardmäßig startet es mit 1 Thread und verwendet Staging-Server-Details

Die Dateien asvincurl.py und Dockerfile werden zusammen nach .tar.gz gezippt.

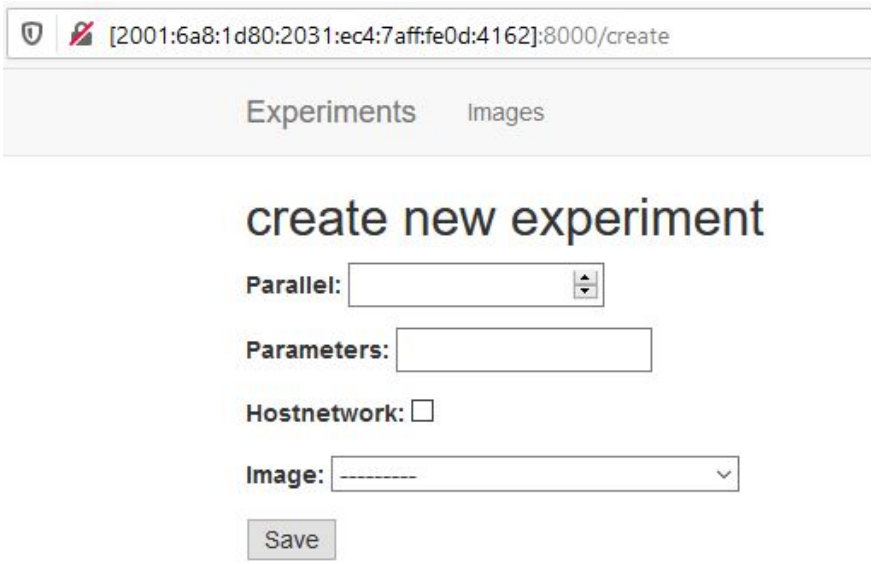
```
tar cvfz asvin_stage2.tar.gz asvincurl.py Dockerfile
```

Der Steuerserver verfügt über eine Weboberfläche, über die der Benutzer mithilfe der erzeugten tar-Datei ein Docker-Image erstellen kann, das dann in der Docker-Registry bereitgestellt wird.

4. Monitoring der Experimente

   In der Schnittstelle zur Experimentüberwachung kann ein neues Experiment mit einem der Docker-Images aus der Docker-Registry erstellt werden.

   Beim Erstellen der Experimente sollten Sie die Laufzeitparameter für den Python-Code angeben. Andernfalls wird der Code mit den Standardparametern ausgeführt. Außerdem sollten Sie die Anzahl der Pods (Parallelen) angeben, die auf dem Kubernetes-Cluster ausgeführt werden sollen.



Die Anzahl der parallelen Pods, die auf dem Cluster laufen, kann jederzeit während des laufenden Experiments geändert werden.

5. **Analyse der Daten in Grafana** Das im Experiment laufende asvin API-Stack-Image speichert die folgenden Werte im influxdb-Server.

    1. Gesamte Anfragen an Versionskontroller, Blockchain- und IPFS-Server
    2. Gesamtzahl der erfolgreich bedienten Anfragen von Versionskontroller, Blockchain- und IPFS-Servern
    3. Summe der fehlgeschlagenen Anfragen von Versionskontroller-, Blockchain- und IPFS-Servern
    4. Antwortzeiten der einzelnen Anfragen an alle 3 Server
    5. Erfolgreiche Firmware-Updates

    In Grafana werden diese Werte vom influxdb-Server geholt und als Zeitseriendiagramme visualisiert, um die Robustheit der asvin-Plattform zu analysieren. Die Beispiel-Json-Datei kann zur Erstellung eines Grafana-Dashboards verwendet werden.

KAPITEL 5

Security Principles

Dieser Abschnitt enthält die Sicherheitsprinzipien der asvin Plattform.

5.1 PUF Based Device ID

Wikipedia " A physical unclonable function or PUF, is a physical object that for a given input and conditions (challenge), provides a physically-defined "digital fingerprint" output that serves as a unique identifier "

PUF takes advantage of submicron variations that occur naturally during semiconductor fabrication. These variations occur in the physical parameters such as length, width, thickness, etc of semiconductor materials. a detailed blog is posted here.

For this demo we have used the E1 devlopment board from OKDO. This development board is based on LPC55S69xx microcontroller from NXP. The microcontroller contains onboard Physical Unclonable Function (PUF) using dedicated SRAM. Follow this blog here for a detailed setup guide.

Later on we also describe the process of key generation here based on PUF technology. This generated key can be used in your applications in various ways. We at asvin are using the generated ID as a unique device ID while making requests to the asvin BeeHive IoT update platform.

5.2 API Endpoint Security

asvin-Komponenten stellen ihre Dienste über RESTful-API-Endpunkte zur Verfügung. Sie werden mit Jason Web Token (JWT) gesichert. Es ist erforderlich, ein JWT vom OAuth-Server zu erhalten. Erst danach kann auf die Endpunkte erfolgreich zugegriffen werden. Der *Login* -API-Endpunkt wird verwendet, um JWT von OAuth zu erhalten.

5.2.1 Geräte-Signatur

Die in der Die *Login* -API verwendete `device_signature` ist ein hashed-basierter Message Authentication Code (MAC). Er besteht aus einer kryptografischen Hash-Funktion (HMAC-SHA256) und einem geheimen Schlüssel. Im Pseudocode kann er als `HMAC-SHA256(key, message)`. dargestellt werden. Hier ist die Nachricht `timestamp+device_key` und der Schlüssel ist `customer_key`. Die `device_signature` wird also berechnet als

```
device_signature = HMAC-SHA256(customer_key, timestamp+device_key)
```

The customer_key and device_key are acquired from Customer Platform. One needs to make a account there. The code block below shows the `device_signature` generation.

Der customer_key und device_key werden von der Customer Platform bezogen. Man muss dort ein Konto anlegen. Der Codeblock unten zeigt die Erzeugung der `device_signature`

Bash

Python

JavaScript

```
#!/bin/bash
customer_key="my-customer-key"
device_key="my-device-key"
timestamp=$(date +%s)
device_signature=$(echo -n $timestamp$device_key | openssl dgst -sha256 -hmac $customer_
↪key)
echo $device_signature
```

```
import hmac
import hashlib
from time import time
customer_key = "my-customer-key"
device_key = "my-device-key"
timestamp = str(math.floor(time()))
device_signature = hmac.new(customer_key, msg=timestamp+device_key, digestmod=hashlib.
↪sha256).hexdigest().upper()
print device_signature
```

```
const CryptoJS = require("crypto-js");
const dateNow = new Date();
const customerKey = "my-customer-key";
const deviceKey =  "my-device-key";
const timestamp = Math.floor(dateNow.getTime() / 1000);
const deviceSignature = CryptoJS.HmacSHA256(timestamp + deviceKey, customerKey).
↪toString(CryptoJS.digest);
console.log(deviceSignature)
```

KAPITEL 6

Architecture Reference

KAPITEL 7

REST API

Dieser Abschnitt definiert die REST-APIs, die von allen Komponenten von Asvin unterstützt werden. Um die API-Endpunkte zu testen, verwenden Sie das Swagger API Doc Doc oder die Postman Collection .

7.1 OAuth APIs

Dieser Abschnitt zeigt die Rest-API-Endpunkte von OAuth Server.

Inhaltsübersicht

7.1.1 Login

Dieser API-Endpunkt gibt ein Token zurück, das dem **x-access-token** Header für alle anderen API-Anfragen hinzugefügt werden sollte.

`POST /auth/login`

Der `device_key` und der `customer_key` werden aus dem asvin Dashboard bezogen. Der The `timestamp` ist eine Unix-Epoche, die `device_signature` ist HMAC-SHA256. Der Abschnitt Gerätesignatur enthält Details zur Berechnung.

Request Headers

- Content-Type – application/json

Example request:

cURL

JavaScript

Python

PHP

```
$ curl --location --request POST 'https://oauth-server/auth/login' \
 --header 'Content-Type: application/json' \
 --data-raw '{
     "device_key": "your-device-key",
     "timestamp": 1620045991
     "device_signature": "your-device-signature"
 }'
```

```
var request = require('request');
var options = {
  'method': 'POST',
  'url': 'https://oauth-server/auth/login',
  'headers': {
    'Content-Type': 'application/json'
  },
  body: JSON.stringify({"device_key":"your-device-key","timestamp": 1620045991,
↪"device_signature":"your-device-signature"})

};
request(options, function (error, response) {
  if (error) throw new Error(error);
  console.log(response.body);
});
```

```
import requests
url = "https://oauth-server/auth/login"
payload="{\"device_key\":\"your-device-key\",\"timestamp\":1620045991,\"device_
↪signature\":\"your-device_signature\"}"
headers = {
  'Content-Type': 'application/json'
}
response = requests.request("POST", url, headers=headers, data=payload)
print(response.text)
```

```
<?php
$client = new http\Client;
$request = new http\Client\Request;
$request->setRequestUrl('https://oauth-server/auth/login');
$request->setRequestMethod('POST');
$body = new http\Message\Body;
$body->append('{
    "device_key": "your-device-key",
    "timestamp": 1620045991,
    "device-signature": "your-device_signature",
}');
$request->setBody($body);
$request->setOptions(array());
$request->setHeaders(array(
  'Content-Type' => 'application/json'
));
$client->enqueue($request)->send();
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
$response = $client->getResponse();
echo $response->getBody();
```

Example response:

```
{
  "token": "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.
↪eyJpYXQiOjE2MDYunDk4ODYsImV4cCI6MTYwNjMxMDQ4Nn0.CCWvzR124OGf5FFOFAObQDPNRlmtI_
↪kaObtu0X-
↪eNFpJUaHv5kfjfGzZl4PUVXTOidSC4SJXFLACqOgyY7gb1UiHI3S47KvhIdCLgte8BvEIyIWLLj4rD4mdWT4NeRkP67-
↪AXUG9IVM7_6XaGB-xmVLD-cLKFimlH7wANeDxO51gOgbcO5CP-1LQKuc2ApYPnDwtJMbkLIcQ-
↪f7k81ouiiOWKOsB-cXq8yqt85WV4BJADhTDbvm3kjAQ5AEOpi7cU_
↪sxh4JG4RaFKz7mNAanvHTw7LbZmP6tcvcf-bvcqTkkb0nkstXCD6300mBe4D44gY-
↪7OehM1HF7xUS6nYpnIw"
}
```

Response Headers

- Content-Type – application/json
- *X-RateLimit-Limit* – 10
- *X-RateLimit-Remaining* – 9
- *X-RateLimit-Reset* – 1617352926

Status Codes

- 200 OK – OK
- 429 Too Many Requests – Zu viele Anfragen im gewünschten Zeitraum
- 500 Internal Server Error – Error on Server

7.2 Version Controller APIs

Dieser Abschnitt zeigt die Rest-API-Endpunkte von Version Controller.

Inhaltsübersicht

7.2.1 Register Device

POST /api/device/register

Register a device.

Request Headers

- Content-Type – application/json
- *X-Access-Token* – JWT-TOKEN

Example request:

cURL

JavaScript

Python

PHP

```
$ curl --location --request POST 'https://vc-server/api/device/register' \
 --header 'Content-Type: application/json' \
 --header 'X-Access-Token: <JWT-TOKEN>' \
 --data-raw '{
   "mac": "your-device-mac",
   "firmware_version": "1.0",
   "name": "device-name",
 }'
```

```
var request = require('request');
var options = {
  'method': 'POST',
  'url': 'https://vc-server/api/device/register',
  'headers': {
    'Content-Type': 'application/json',
    'X-Access-Token': '<JWT-TOKEN>'
  },
  body: JSON.stringify({"mac":"your-device-mac","firmware_version":"1.0","name":
↪"device-name"})

};
request(options, function (error, response) {
  if (error) throw new Error(error);
  console.log(response.body);
});
```

```
import requests
url = "https://vc-server/api/device/register"
payload = "{\n\t\"mac\": \"your-device-mac\",\n\t\"firmware_version\": \"1.0\",\n\t\
↪"name\": \"device-name\"\n}"
headers = {
  'Content-Type': 'application/json',
  'X-Access-Token': '<JWT-TOKEN>'
}
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
response = requests.request("POST", url, headers=headers, data = payload)
print(response.text.encode('utf8'))
```

```
<?php
$client = new http\Client;
$request = new http\Client\Request;
$request->setRequestUrl('https://vc-server/api/device/register');
$request->setRequestMethod('POST');
$body = new http\Message\Body;
$body->append('{
  "mac": "your-device-mac",
  "firmware_version": "1.0",
  "name": "device-name",
}');
$request->setBody($body);
$request->setOptions(array());
$request->setHeaders(array(
  'Content-Type' => 'application/json',
  'X-Access-Token': '<JWT-TOKEN>'
));
$client->enqueue($request)->send();
$response = $client->getResponse();
echo $response->getBody();
```

Example response:

```
{
  "message": "Device inserted!"
}
```

```
{
  "message": "Device exists!"
}
```

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – No error
- 404 Not Found – Not Found
- 401 Unauthorized – JWT is not valid

7.2.2 Delete Device

DELETE /api/device/

Delete a device.

Request Headers

- Content-Type – application/json
- *X-Access-Token* – JWT-TOKEN

Example request:

cURL

```
$ curl --location --request DELETE 'https://vc-server/api/device/' \
 --header 'Content-Type: application/json' \
 --header 'X-Access-Token: <JWT-TOKEN>' \
 --data-raw '{
   "mac": "your-device-mac",
 }'
```

Example response:

```
{
  "message": "Device deleted!"
}
```

```
{
  "message": "Device does't exist!"
}
```

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – No error
- 404 Not Found – Not Found
- 401 Unauthorized – JWT is not valid

7.2.3 Get Device

GET /api/device/:mac

Get a device.

Request Headers

- Content-Type – application/json
- *X-Access-Token* – JWT-TOKEN

Example request:

cURL

```
$ curl --location --request GET 'https://vc-server/api/device/<device-mac>' \
 --header 'X-Access-Token: <JWT-TOKEN>'
```

Example response:

```
{
  "mac": "device-mac",
  "name": "device-name",
  "firmware-version": "device-firmware-version"
}
```

```
{
  "message": "Device does't exist!"
}
```

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – No error
- 404 Not Found – Not Found
- 401 Unauthorized – JWT is not valid

7.2.4 Get All Devices

GET /api/device/

Get all devices.

Request Headers

- Content-Type – application/json
- *X-Access-Token* – JWT-TOKEN

Example request:

cURL

```
$ curl --location --request GET 'https://vc-server/api/device/' \
 --header 'X-Access-Token: <JWT-TOKEN>'
```

Example response:

```
[
  {
  "mac": "device-mac",
  "name": "device-name",
  "firmware-version": "device-firmware-version"
  },
  {
  "mac": "device-mac",
  "name": "device-name",
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
  "firmware-version": "device-firmware-version"
  }
]
```

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – No error
- 404 Not Found – Not Found
- 401 Unauthorized – JWT is not valid

7.2.5 Get Report

GET /api/report/:type

Get report in json, xml, and pdf format.

reqheader Content-Type application/json

reqheader X-Access-Token JWT-TOKEN

Example request:

cURL

```
$ curl --location --request GET 'https://vc-server/api/report/<json/xml/pdf>
↪' \
 --header 'X-Access-Token: <JWT-TOKEN>'
```

Example response:

```
[
  {
  "mac": "device-mac",
  "name": "device-name",
  "firmware-version": "device-firmware-version"
  },
  {
  "mac": "device-mac",
  "name": "device-name",
  "firmware-version": "device-firmware-version"
  }
]
```

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <element>
      <firmware-version>device-firmware-version</firmware-version>
      <mac>device-mac</mac>
      <name>device-name</name>
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
  </element>
  <element>
      <firmware-version>device-firmware-version</firmware-version>
      <mac>device-mac</mac>
      <name>device-name</name>
  </element>
</root>
```

```
:resheader Content-Type: application/json

:statuscode 200: No error
:statuscode 404: Not Found
:statuscode 401: JWT is not valid
```

7.2.6 Next Rollout

POST /api/device/next/rollout

Check next rollout

reqheader Content-Type application/json

reqheader X-Access-Token JWT-TOKEN

Example request:

cURL

JavaScript

Python

PHP

```
curl --location --request POST 'https://vc-server/api/device/next/rollout' \
--header 'Content-Type: application/json' \
--header 'X-Access-Token: <JWT-TOKEN>' \
--data-raw '{
  "mac": "your-device-mac",
  "firmware_version": "1.0"
}'
```

```
var request = require('request');
var options = {
  'method': 'POST',
  'url': 'https://vc-server/api/device/next/rollout',
  'headers': {
    'Content-Type': 'application/json',
    'X-Access-Token': '<JWT-TOKEN>'
  },
  body: JSON.stringify({"mac":"your-device-mac","firmware_version":"1.0"})

};
request(options, function (error, response) {
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
  if (error) throw new Error(error);
  console.log(response.body);
});
```

```
import requests
url = "https://vc-server/api/device/next/rollout"
payload = "{\n\t\"mac\": \"your-device-mac\",\n\t\"firmware_version\": \"1.
↪0\"\n}"
headers = {
  'Content-Type': 'application/json',
  'X-Access-Token': '<JWT-TOKEN>'
}
response = requests.request("POST", url, headers=headers, data = payload)
print(response.text.encode('utf8'))
```

```
<?php
$client = new http\Client;
$request = new http\Client\Request;
$request->setRequestUrl('https://vc-server/api/device/next/rollout');
$request->setRequestMethod('POST');
$body = new http\Message\Body;
$body->append('{
  "mac": "your-device-mac",
  "firmware_version": "1.0"
}');
$request->setBody($body);
$request->setOptions(array());
$request->setHeaders(array(
  'Content-Type' => 'application/json',
  'X-Access-Token': '<JWT-TOKEN>'
));
$client->enqueue($request)->send();
$response = $client->getResponse();
echo $response->getBody();
```

Example response:

```
{
  "rollout_id": "84",
  "rollout_name": "new-demo-rollout",
  "priority": "1",
  "start_date": "2021-04-02 09:30:00",
  "version": "2.0",
  "firmware_id": "11"
}
```

If no rollout exists:

```
{}
```

resheader Content-Type application/json

statuscode 200 No error

statuscode 404 Not Found

statuscode 401 JWT is not valid

7.2.7 Rollout Success

POST /api/device/success/rollout

Inform rollout status

Request Headers

- Content-Type – application/json
- *X-Access-Token* – JWT-TOKEN

Example request:

cURL

JavaScript

Python

PHP

```
curl --location --request POST 'https://vc-server/api/device/success/rollout' \
--header 'Content-Type: application/json' \
--header 'X-Access-Token: <JWT-TOKEN>' \
--data-raw '{
  "mac": "your-device-mac",
  "firmware_version": "2.0"
  "rollout_id": "84"
}'
```

```
var request = require('request');
var options = {
  'method': 'POST',
  'url': 'https://vc-server/api/device/success/rollout',
  'headers': {
    'Content-Type': 'application/json',
    'X-Access-Token': '<JWT-TOKEN>'
  },
  body: JSON.stringify({"mac":"your-device-mac","firmware_version":"2.0","rollout_id
↪":"84"})

};
request(options, function (error, response) {
  if (error) throw new Error(error);
  console.log(response.body);
});
```

```
import requests
url = "https://vc-server/api/device/success/rollout"
payload = "{\n\t\"mac\": \"your-device-mac\",\n\t\"firmware_version\": \"2.0\",\n\t\
↪"rollout_id\": \"84\"\n}"
headers = {
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
  'Content-Type': 'application/json',
  'X-Access-Token': '<JWT-TOKEN>'
}
response = requests.request("POST", url, headers=headers, data = payload)
print(response.text.encode('utf8'))
```

```
<?php
$client = new http\Client;
$request = new http\Client\Request;
$request->setRequestUrl('https://vc-server/api/device/success/rollout');
$request->setRequestMethod('POST');
$body = new http\Message\Body;
$body->append('{
  "mac": "your-device-mac",
  "firmware_version": "2.0",
  "rollout_id": "84"
}');
$request->setBody($body);
$request->setOptions(array());
$request->setHeaders(array(
  'Content-Type' => 'application/json',
  'X-Access-Token': '<JWT-TOKEN>'
));
$client->enqueue($request)->send();
$response = $client->getResponse();
echo $response->getBody();
```

Example response:

```
{
  "message": "Successfully inserted!"
}
```

```
{
  "message": "Existing Record"
}
```

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – No error
- 404 Not Found – Not Found
- 401 Unauthorized – JWT is not valid

7.3 Blockchain APIs

This section shows the Rest API end-points of Blockchain.

Table of contents

7.3.1 Get Device

GET /device/:id

Get a device.

Request Headers

- Content-Type – application/json
- *X-Access-Token* – JWT-TOKEN

Example request:

cURL

JavaScript

Python

PHP

```
$ curl 'https://bc-server/device/:id' \
-H 'Content-Type: application/json' \
-H 'X-Access-Token: <JWT-TOKEN>
```

```
var request = require('request');
var options = {
  'method': 'GET',
  'url': 'https://bc-server/device/:id',
  'headers': {
    'X-Access-Token': '<JWT-TOKEN>'
  })
};
request(options, function (error, response) {
  if (error) throw new Error(error);
  console.log(response.body);
});
```

```
import requests

url = "https://bc-server/device/:id"

headers = {
  'X-Access-Token': '<JWT-TOKEN>'
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
}
response = requests.request("GET", url, headers=headers)
```

```
<?php
$client = new http\Client;
$request = new http\Client\Request;
$request->setRequestUrl('https://bc-server/device/:id');
$request->setRequestMethod('GET');

$request->setOptions(array());
$request->setHeaders(array(
  'X-Access-Token' => '<JWT-TOKEN>'
));
$client->enqueue($request)->send();
$response = $client->getResponse();
echo $response->getBody();
```

Example response:

```
{
  "mac": "AC:AC:CC:CC:34:34",
  "fwId": "3",
  "dType": "ESP"
}
```

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – OK
- 404 Not Found – Not Found

7.3.2 Get Firmware

`GET /firmware/:id`

Get a device.

Request Headers

- Content-Type – application/json
- *X-Access-Token* – JWT-TOKEN

Example request:

cURL

JavaScript

Python

PHP

```
$ curl 'https://bc-server/device/:id' \
-H 'X-Access-Token: <JWT-TOKEN>
```

```
var request = require('request');
var options = {
  'method': 'GET',
  'url': 'https://bc-server/firmware/:id',
  'headers': {
    'X-Access-Token': '<JWT-TOKEN>'
  })

};
request(options, function (error, response) {
  if (error) throw new Error(error);
  console.log(response.body);
});
```

```
import requests

url = "https://bc-server/firmware/:id"

headers = {
  'X-Access-Token': '<JWT-TOKEN>'
}
response = requests.request("GET", url, headers=headers)
print(response.text)
```

```
<?php
$client = new http\Client;
$request = new http\Client\Request;
$request->setRequestUrl('https://bc-server/firmware/:id');
$request->setRequestMethod('GET');
$request->setOptions(array());
$request->setHeaders(array(
  'X-Access-Token' => '<JWT-TOKEN>'
));
$client->enqueue($request)->send();
$response = $client->getResponse();
echo $response->getBody();
```

Example response:

```
{
  "md": "6f5902ac237024bdd0c176cb93063dc4",
  "cid": "QmWATWQ7fVPP2EFGu71UkfnqhYXDYH566qy47CnJDgvs8u",
  "version": "1.9",
}
```

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – OK
- 404 Not Found – Not Found

7.4 IPFS APIs

This section shows the Rest API end-points of IPFS.

Table of contents

7.4.1 Download Firmware

GET /firmware:id

Get a device.

Request Headers

- Content-Type – application/json
- *x-access-token* – JWT-TOKEN

Example request:

cURL

JavaScript

Python

PHP

```
$ curl -X GET 'https://ipfs-server/firmware/:cid' \
-H 'x-access-token: <JWT-TOKEN>' \
--header 'Content-Type: application/json'
```

```
var request = require('request');
var options = {
  'method': 'GET',
  'url': 'https://ipfs-server/firmware/:cid',
  'headers': {
    'x-access-token': '<JWT-TOKEN>',
  },
};
request(options, function (error, response) {
  if (error) throw new Error(error);
  console.log(response.body);
});
```

```
import requests

url = "https://ipfs-server/firmware/:cid"
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
headers = {
  'x-access-token': '<JWT-TOKEN>',
}
response = requests.request("GET", url, headers=headers)
print(response.text)
```

```
<?php
$client = new http\Client;
$request = new http\Client\Request;
$request->setRequestUrl('https://ipfs-server/firmware/:cid');
$request->setRequestMethod('GET');
$body = new http\Message\Body;

$request->setOptions(array());
$request->setHeaders(array(
  'x-access-token' => '<JWT-TOKEN>',
));
$client->enqueue($request)->send();
$response = $client->getResponse();
echo $response->getBody();
```

Example response:

```
```

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – OK
- 404 Not Found – Not Found

KAPITEL 8

Beteiligung am Projekt ist gewünscht!

Das asvin-Projekt wurde von einigen wenigen begonnen, aber von vielen entwickelt und gepflegt. Wir bei asvin sind ein großer Unterstützer von Open-Source-Projekten und haben auch die gesamte asvin-Plattform für die Community offengehalten. Wir begrüßen Ihren Beitrag zum Projekt. Lassen Sie uns gemeinsam die asvin Plattform für alle aufbauen.

Voraussetzung für eine fairen und respektvollem Umgang miteinander ist das Befolgen des asvin *Verhaltenskodex von asvin*. Bevor Sie einen Bug melden oder eine Anfrage starten, lesen Sie bitte vorab unseren asvin-Verhaltenskodex durch.

Die Quelldateien der asvin-Plattform werden in den GitHub-Repositories gespeichert. Folgen Sie den *GitHub-Beitragsrichtlinien*, um einen Beitrag zum Projekt zu leisten.

Bemerkung: Please do not modify without first getting approval from m.ross@asvin.io

8.1 Verhaltenskodex von asvin

Die Asvin GmbH entwickelt eine Plattform zur Verteilung von Firmware-Updates für IoT-Geräte. Wir ermutigen auch externe Entwickler, sich an dem Projekt zu beteiligen. Das bedeutet, dass es sich um ein Open-Source-Projekt handelt, bei dem die Teilnehmer zusammenarbeiten können und dabei Unterschiede in Bezug auf Sprache, Standort, Nationalität und Erfahrung erfahren. In einer solch vielfältigen Umgebung, passieren Missverständnisse und Meinungsverschiedenheiten, die in den meisten Fällen informell gelöst werden können. In seltenen Fällen kann es jedoch zu Verhaltensweisen kommen, die eine oder mehrere Personen in der Gemeinschaft einschüchtern, belästigen oder anderweitig stören, was Asvin nicht tolerieren wird.

Ein Verhaltenskodex ist nützlich, um akzeptierte und akzeptable Verhaltensweisen zu definieren und hohe Standards der beruflichen Praxis zu fördern. Er bietet auch einen Maßstab für die Selbstbewertung und dient als Vehikel für eine bessere Identität der Organisation.

Dieser Kodex (CoC) gilt für jedes Mitglied der Asvin-Gemeinschaft - Entwickler, Teilnehmer an Meetings, Telekonferenzen, Mailinglisten, Konferenzen oder Veranstaltungen usw. Beachten Sie, dass dieser Kodex die gesetzlichen Rechte und Pflichten, die sich auf eine bestimmte Situation beziehen, nicht ersetzt, sondern ergänzt.

8.1.1 Absichtserklärung

Asvin verpflichtet sich, ein positives *Arbeitsumfeld* zu schaffen. Diese Verpflichtung erfordert einen Arbeitsplatz, an dem sich alle *Beteiligten* auf allen Ebenen nach den Regeln des folgenden Kodex verhalten. Ein grundlegendes Konzept dieses Kodex ist, dass wir alle gemeinsam für unser Arbeitsumfeld verantwortlich sind.

8.1.2 Kodex

1. Behandeln Sie sich gegenseitig mit *respekt*, Professionalität, Fairness und Sensibilität für

   unsere vielen Unterschiede und Stärken, auch in Situationen mit hohem Druck und Dringlichkeit.

2. *Belästigen* oder *schikanieren* Sie niemanden verbal, körperlich oder *sexuell*.
3. *Diskriminieren* Sie niemanden aufgrund von persönlichen Eigenschaften oder Gruppenzugehörigkeit.
4. Kommunizieren Sie konstruktiv und vermeiden Sie *erniedrigendes* oder *beleidigendes* Verhalten oder Sprache
5. Suchen, akzeptieren und bieten Sie objektive Arbeitskritik und *erkennen* Sie die Beiträge anderer angemessen an.
6. Seien Sie ehrlich in Bezug auf Ihre eigenen Qualifikationen und auf alle Umstände, die zu Interessenkonflikten führen könnten.
7. Respektieren Sie die Privatsphäre anderer und die Vertraulichkeit der Daten, auf die Sie zugreifen.
8. Seien Sie in Bezug auf kulturelle Unterschiede konservativ in dem, was Sie tun, und liberal in dem, was Sie von anderen

   akzeptieren, aber nicht bis zu dem Punkt, dass Sie respektloses, unprofessionelles oder unfaires oder unwillkommenes Verhalten oder Annäherungen akzeptieren.

9. Fördern Sie die Regeln dieses Kodex und ergreifen Sie Maßnahmen (insbesondere, wenn Sie eine Führungsposition innehaben), um die Diskussion wieder auf eine zivilere Ebene zu bringen, wenn unangemessenes Verhalten beobachtet wird.
10. Bleiben Sie beim Thema: Stellen Sie sicher, dass Sie im richtigen Kanal posten und vermeiden Sie Off-Topic-Diskussionen. Denken Sie daran, dass Sie, wenn Sie ein Thema aktualisieren oder auf eine E-Mail antworten, potenziell an eine große Anzahl von Personen senden
11. Treten Sie rücksichtsvoll zurück: Mitglieder jedes Projekts kommen und gehen, und das ist bei Asvin nicht anders. Wenn Sie das Projekt ganz oder teilweise verlassen, bitten wir Sie, dies so zu tun, dass das Projekt so wenig wie möglich gestört wird. Das bedeutet, dass Sie den Leuten sagen sollten, dass Sie gehen und die richtigen Schritte unternehmen sollten, um sicherzustellen, dass andere dort weitermachen können, wo Sie aufgehört haben.

8.1.3 Glossar

ist ein Verhalten, das die Würde, das Selbstwertgefühl oder den Respekt einer anderen Person innerhalb der Gemeinschaft herabsetzt.

Erniedrigendes Verhalten

ist die Behandlung einer anderen Person mit Verachtung oder Respektlosigkeit.

Diskriminierung

ist die nachteilige Behandlung einer Person aufgrund von Kriterien wie: körperliches Aussehen, Rasse, ethnische Herkunft, genetische Unterschiede, nationale oder soziale Herkunft, Name, Religion, Geschlecht, sexuelle Orientierung, familiäre oder gesundheitliche Situation, Schwangerschaft, Behinderung, Alter, Bildung, Vermögen, Wohnsitz, politische Einstellung, Moral, Beschäftigung oder gewerkschaftliche Tätigkeit.

Beleidigendes Verhalten

is treating another person with scorn or disrespect.

Quellenangabe

ist eine Angabe über die Herkunft(en) und den/die Autor(en) eines Beitrags.

Harassment

ist jedes Verhalten, verbal oder physisch, das die Absicht oder Wirkung hat, eine Person zu beeinträchtigen, oder das eine einschüchternde, feindselige oder beleidigende Umgebung schafft. beleidigendes Umfeld schafft.

Führungsposition

umfasst Gruppenvorsitzende, Projektbetreuer, Mitarbeiter und Vorstandsmitglieder.

Beteiligten

includes the following persons:

- schließt die folgenden Personen ein:
- Entwickler
- Vertreter der Mitglieder
- Mitarbeiter
- Jeder aus der Öffentlichkeit, der an der Arbeitsumfeld von Asvin teilnimmt (z. B. Code beitragen, unseren Code oder Spezifikationen kommentieren, uns eine E-Mail schicken, an unseren Konferenzen und Veranstaltungen teilnehmen, usw.)

Respekt

ist die aufrichtige Rücksicht, die Sie auf jemanden nehmen (und sei es nur aufgrund seines Status als Teilnehmer in Asvin, wie Sie selbst), und die Sie zeigen, indem Sie ihn höflich und freundlich behandeln.

Sexuelle Belästigung

umfasst die visuelle Darstellung entwürdigender sexueller Bilder, sexuell anzügliches Verhalten, beleidigende Bemerkungen sexueller Natur, Bitten um sexuelle Gefälligkeiten, unerwünschten Körperkontakt und sexuelle Übergriffe.

Unerwünschtes Verhalten

Schwer zu definieren? Einige Fragen, die Sie sich stellen sollten, sind:

- Wie würde ich mich fühlen, wenn ich in der Position des Empfängers wäre?
- Würde es meinem Ehepartner, Elternteil, Kind, Geschwister oder Freund gefallen, auf diese Weise behandelt zu werden?
- Würde ich es begrüßen, wenn ein Bericht über mein Verhalten im Newsletter der Organisation veröffentlicht würde?
- Könnte mein Verhalten andere Mitglieder der Arbeitsgruppe beleidigen oder verletzen?
- Könnte jemand mein Verhalten als absichtlich schädlich oder belästigend missverstehen?
- würde ich meinen Chef oder eine Person, die ich bei der Arbeit respektiere, so behandeln?

Zusammenfassung: Wenn Sie sich nicht sicher sind, ob etwas willkommen oder unerwünscht sein könnte, tun Sie es nicht.

Unerwünschte sexuelle Annäherung umfasst:

Bitten um sexuelle Gefälligkeiten und anderes verbales oder körperliches Verhalten sexueller Natur, wenn:

- die Unterwerfung unter ein solches Verhalten entweder ausdrücklich oder stillschweigend zu einer Bedingung für die Beschäftigung einer Person gemacht wird,
- die Unterwerfung unter ein solches Verhalten oder die Ablehnung eines solchen Verhaltens durch eine Person als Grundlage für Beschäftigungsentscheidungen verwendet wird, die diese Person betreffen,
- ein solches Verhalten den Zweck oder die Wirkung hat, die Arbeitsleistung einer Person unangemessen zu beeinträchtigen oder ein einschüchterndes, feindseliges oder beleidigendes Arbeitsumfeld zu schaffen.

Mobbing am Arbeitsplatz

ist eine Tendenz von Einzelpersonen oder Gruppen, anhaltend aggressives oder unangemessenes Verhalten (z. B. verbale oder schriftliche Beschimpfungen, beleidigendes Verhalten oder jegliche Einmischung, die die Arbeit untergräbt oder behindert) gegen einen Mitarbeiter oder eine berufliche Beziehung anzuwenden.

Arbeitsumfeld

ist die Gesamtheit aller verfügbaren Mittel der Zusammenarbeit, einschließlich, aber nicht beschränkt auf Nachrichten an Mailing-Listen, private Korrespondenz, Webseiten, Chat-Kanäle, Telefon- und Video-Telekonferenzen und jede Art von persönlichen Treffen oder Diskussionen.

Verfahren bei Vorfällen

Um Vorfälle zu melden oder um gegen Berichte über Vorfälle Einspruch zu erheben, senden Sie bitte eine E-Mail an Mirko Ross (m.ross@asvin.io). Bitte fügen Sie alle verfügbaren relevanten Informationen bei, einschließlich Links zu allen öffentlich zugänglichen Materialien, die sich auf die Angelegenheit beziehen. Es werden alle Anstrengungen unternommen, um ein sicheres und kollegiales Umfeld zu gewährleisten, in dem die Zusammenarbeit in Bezug auf das Projekt stattfindet. Um die Gemeinschaft zu schützen, behält sich das Projekt das Recht vor, angemessene Maßnahmen zu ergreifen, die möglicherweise auch den Ausschluss einer Person von jeglicher Teilnahme am Projekt beinhalten können. Das Projekt wird im Falle eines Missverständnisses auf eine gerechte Lösung hinarbeiten.

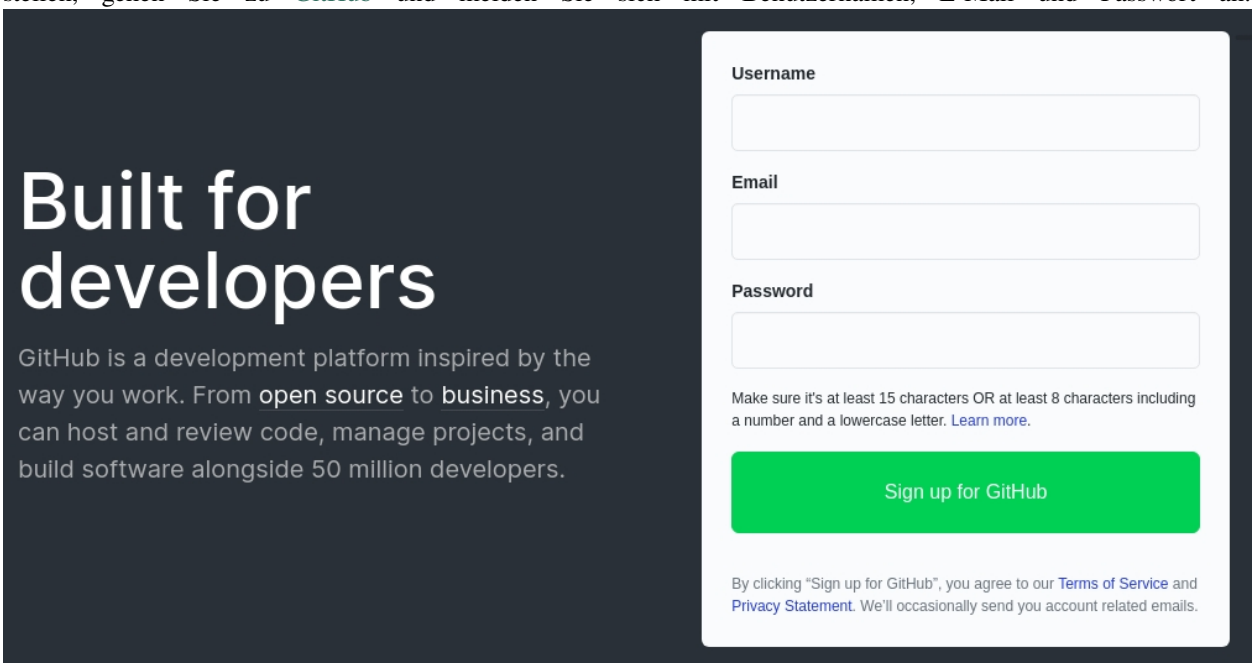
8.1.4 Credits

Dieser Kodex basiert auf dem Hyperledger Code of Conduct.

8.2 GitHub-Beitragsrichtlinien

8.2.1 Github-Konto erstellen

Die Codebasis der Asvin-Plattform wird auf GitHub entwickelt und gepflegt. Um einen Beitrag für das Asvin-Projekt leisten zu können, muss man ein GitHub-Konto erstellen. Um das Konto zu erstellen, gehen Sie zu GitHub und melden Sie sich mit Benutzernamen, E-Mail und Passwort an.

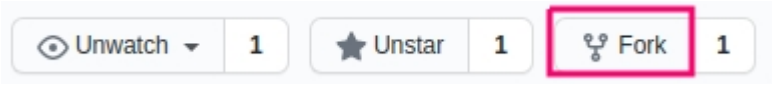


8.2.2 Fork Repository

Asvin besteht aus mehreren Komponenten, die in entsprechenden Repositories entwickelt werden. Wenn Sie anfangen, etwas beizutragen, ist es empfehlenswert, das Repository zu forken, Ihre Änderungen vorzunehmen und einen Pull-Request einzureichen. Dies hilft dabei, den Quellcode im Master-Zweig wartbar, sauber und stabil zu halten. Das Forken eines Projektarchivs führt zu einer identischen Kopie des Projektarchivs in Ihrem persönlichen Konto. Damit haben Sie die volle Kontrolle über das Projektarchiv, um Änderungen am Quellcode vorzunehmen. Sobald Sie mit Ihren Änderungen zufrieden sind, können Sie einen Pull-Request an das offizielle Projektarchiv von Asvin senden.

Wie forked man eine Repository?

- Öffnen Sie Ihren Browser und gehen Sie zu dem Repository von Asvin, das Sie forken möchten
- Klicken Sie auf die Schaltfläche fork in der oberen rechten Ecke



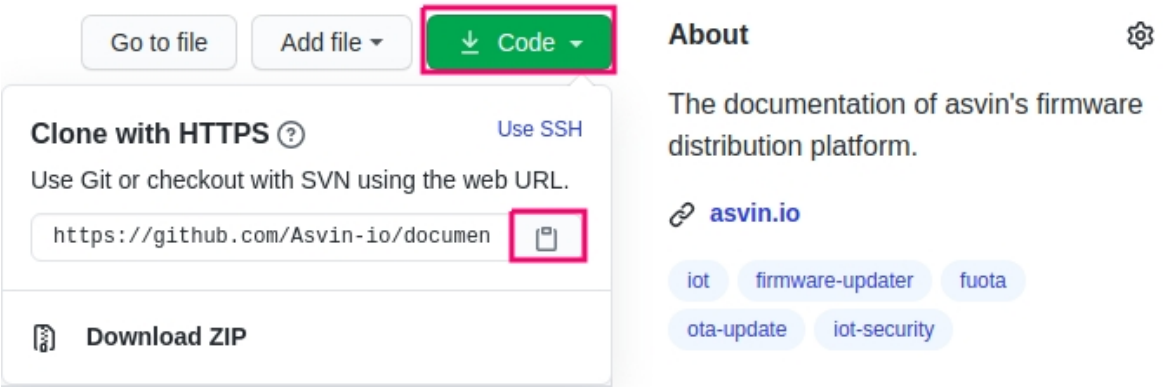
- Wählen Sie Ihren GitHub-Account aus und das Repository wird automatisch geforkt und landet im geklonten Repository in Ihrem Account

8.2.3 Repository klonen

Nach dem Forking-Prozess haben Sie ein Repository in Ihrem Konto. Der nächste Schritt besteht darin, das Repository auf Ihr lokales System zu klonen. Nach dem Klonprozess haben Sie die Quelldateien auf Ihrem lokalen Rechner und können sie in Ihrer bevorzugten IDE zur Entwicklung öffnen.

Wie klont man ein Repository?

- Öffnen Sie Ihren Browser und gehen Sie zu dem Projektarchiv in Ihrem Konto, das Sie klonen möchten.
- Klicken Sie auf die Schaltfläche Code in der Mitte rechts und kopieren Sie die Web-URL clone



- Öffnen Sie Ihr Terminal und wechseln Sie in das Verzeichnis, in das Sie klonen möchten
- Verwenden Sie git clone `git clone <web-url>` z. B. `git clone https://github.com/b-rohit/documentation.git`
- Gehen Sie zum Repository-Verzeichnis und fügen Sie das ursprüngliche Asvin-Repository als entferntes Upstream-Repository hinzu

  Für ein Beispiel-Framework-Repository:

```
cd documentation
git remote add upstream https://github.com/Asvin-io/documentation.git
```

- Sie können alle entfernten Repositorys mit folgendem Befehl überprüfen

```
git remote -v
```

Ein Beispiel für die Ausgabe des Befehls für das Framework-Repository

```
origin        https://github.com/b-rohit/documentation.git (fetch)
origin        https://github.com/b-rohit/documentation.git (push)
upstream        https://github.com/Asvin-io/documentation.git (fetch)
upstream        https://github.com/Asvin-io/documentation.git (push)
```

Jetzt ist alles erledigt, sodass Sie aktiv an der asvin Plattform mitarbeiten können.

8.2.4 Feature-Branch erstellen

Alle asvin-Komponenten-Repositories haben einen Hauptzweig namens `master`. Der `Master-Branch` enthält den stabilen Code der Komponente. Die Hauptidee hinter der Verwendung eines Feature-Branches für die Entwicklung ist, den Master-Branch von fehlerhaftem Code in einem neuen Feature unberührt zu lassen. Ein Entwickler sollte einen Funktionszweig in seinem Fork-Repository erstellen, um eine neue Funktion zu entwickeln, ein Problem zu beheben, Änderungen vorzunehmen usw. Ein Funktionszweig bietet eine Abkapselung von der Haupt-Codebasis. Das bedeutet, dass der `Master-Branch` von Fehlern, die durch fehlerhaften Code eines neuen Commits verursacht werden, isoliert ist. Er hält die Hauptcodebasis sauber und stabil.

Wie erstellt man einen Feature-Branch?

- Holen Sie Commits, Dateien und Refs aus dem Upstream-Repository

```
git fetch upstream
```

- Auschecken in den `Master` Branch

```
git checkout master
```

- Upstream-Änderungen in den lokalen Master-Branch einbinden

```
git merge upstream/master
```

- Änderungen auf den entfernten Master-Zweig im geforkten Repository übertragen

```
git push origin master
```

- Jetzt haben Sie Ihren origin/master und upstream/master synchronisiert. Dieser Prozess stellt sicher, dass es keine Diskrepanzen zwischen den beiden Zweigen gibt und der neue Feature-Branch eine exakte Kopie davon ist.

```
git checkout -b <feature_branch_name>
```

Damit haben Sie einen neuen Feature-Branch, in dem Sie Änderungen vornehmen können.

8.2.5 Änderungen in das Forked Repository übertragen

Wenn Sie mit dem Erstellen einer neuen Funktion oder dem Beheben eines Problems in Ihrem Funktions-Branch fertig sind, ist es an der Zeit, die Änderungen zu übertragen und in das geforkte Repository zu schieben. Dieser Prozess speichert den Status im Remote-Branch im Forked Repository. Später wird dieser Status verwendet, um eine Pull-Anfrage für das Asvin-Komponenten-Repository zu erstellen.

Wie wird gepusht?

- Fügen Sie Ihre geänderten, gelöschten und neuen Dateien in den Index ein

```
git add <file1> <file2>
```

- Sie können nun den aktuellen Inhalt des Indexes übertragen. Dadurch wird ein Schnappschuss der aktuellen Änderungen des Projekts erstellt. Ihre Commit-Nachricht muss die folgenden Informationen enthalten:
  - eine einzeilige Zusammenfassung der Änderungen in diesem Commit als Titel, gefolgt von einer Leerzeile
  - Erläutern Sie im Nachrichtentext, warum diese Änderung notwendig ist, und wie Sie sie angegangen sind. Dies hilft den Reviewern, Ihren Code besser zu verstehen und beschleunigt oft den Review-Prozess.

```
git commit -s
```

- Verschieben Sie die Änderung in Ihr geforktes Repository

```
git push origin <feature_branch_name>
```

Am Ende dieses Prozesses haben Sie eine neue Funktion entwickelt oder einen Fehler behoben und in Ihr Forked-Repository verschoben. In diesem Moment sind Ihre Änderungen im Forked Repository bereit, in das asvin-Komponenten-Repository integriert zu werden. Dies kann durch einen Pull-Request an das Asvin-Repository geschehen.

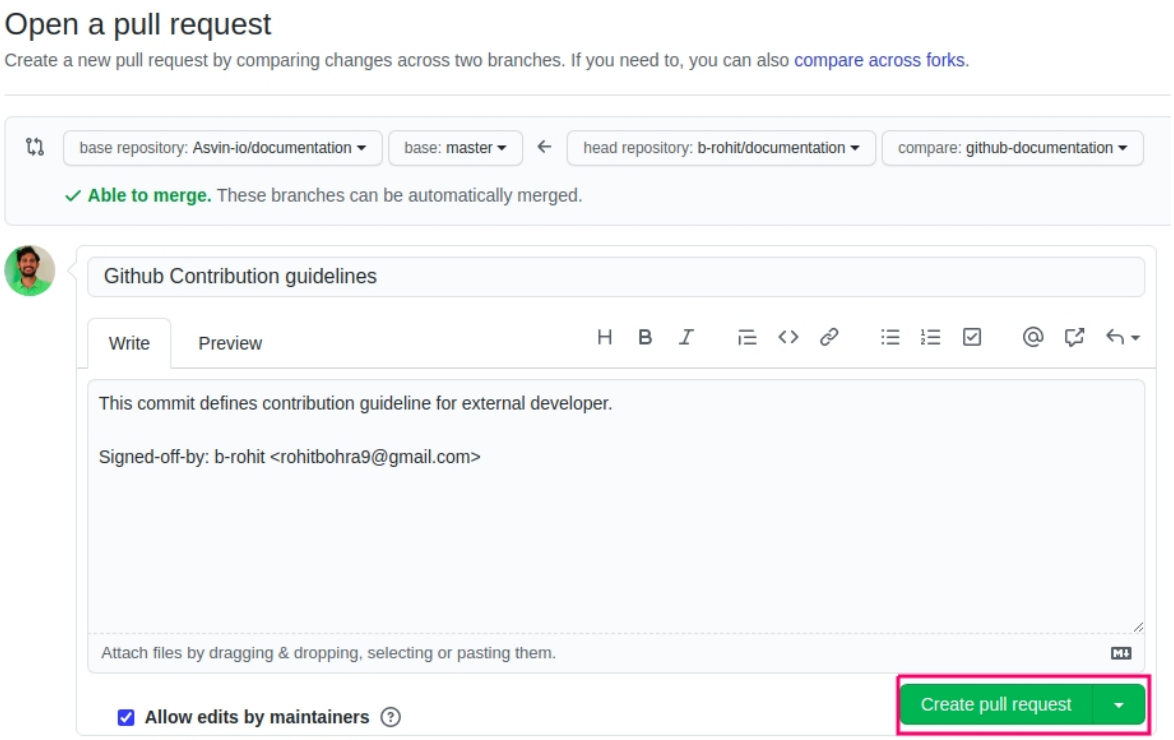
8.2.6 Eröffnen eines Pull-Requests in GitHub

Sobald Sie Ihre Änderungen in den Feature-Branch Ihres Forked Repositories gepusht haben, können Sie nun einen Pull-Request gegen das ursprüngliche Asvin-Komponenten-Repository öffnen. Dies wird die einfachste aller Aufgaben sein, die Sie bisher in diesem Thread erledigt haben.

Wie öffnet man eine Pull-Anfrage?

- Navigieren Sie zu Ihrem geforkten Repository `https://github.com/<Benutzername>/<forked_repository>`.
- Bei neuen Änderungen wird automatisch angezeigt, dass es einige Unterschiede zwischen dem forked und dem ursprünglichen Repository gibt. Klicken Sie auf `Compare & pull request`.

- Sie werden zum ursprünglichen Asvin-Repository navigiert. Hier können Sie den Titel und die Kommentarnachricht ändern, wenn Sie möchten, und auf `Create Pull Request` klicken. .



Sehr gut! Sie öffnen gerade Ihren ersten Pull Request im Asvin Projekt. Der Pull-Request wird geprüft und in das ursprüngliche Asvin-Repository zusammengeführt.

8.2.7 Feature-Branch löschen

Wenn Sie mit Ihrem Feature fertig sind und Ihr Pull-Request akzeptiert und in das ursprüngliche asvin-Repository zusammengeführt wurde, ist es an der Zeit, Ihr geforktes Repository zu bereinigen. Sie müssen den Feature-Branch aus Ihrem lokalen und entfernten geforkten Projektarchiv löschen.

Wie werden lokale und entfernte Verzweigungen gelöscht?

- Löschen Sie Ihren lokalen Funktionszweig

```
git branch -d <feature_branch_name>
```

- Verschieben Sie Ihre Änderungen in den entfernten Zweig

```
git push --delete origin <feature_branch_name>
```

8.2.8 Synchronisierung der Forked Repository mit dem Original

Asvin ist ein vollständig gemeinschaftsbasiertes Projekt. Daher wird die Hauptcodebasis ständig Änderungen von anderen Entwicklern erhalten. Das bedeutet, dass Sie Ihr Forked Repository mit dem originalen Asvin Repository synchronisieren müssen. Auf diese Weise vermeiden Sie Merge-Konflikte.

Wie wird synchronisiert?

- Ziehen Sie die Änderungen aus dem Upstream-Repository

```
git fetch upstream
```

- Rebasieren Sie das lokale origin/master mit upstream/master

```
git rebase upstream/master
```

- Verschieben Sie die Änderungen in das geforkte Repository

```
git push origin master
```

8.3 Styleguide für die Dokumentation

Dieser Abschnitt beschreibt die Stilvorgaben für das Schreiben der Dokumentation. Er basiert auf dem Sphinx-basierten Dokumentations-Styleguide.

8.3.1 Dateinamen

Verwenden Sie nur Kleinbuchstaben und das Symbol - (Minus).

8.3.2 Whitespaces (Leerzeichen)

Einrückung

Einrücken mit 2 Leerzeichen.

Except:

- `toctree` direktive erfordert eine Einrückung mit 3 Leerzeichen.

Leerzeilen

Eine Leerzeile vor jedem Abschnitt (z. B. H1, H2, ...). Siehe *Überschriften* für ein Beispiel.

Eine Leerzeile zur Trennung von Direktiven.

```
Some text before.

.. note::

  Some note.
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
  Ausnahme: Direktiven können ohne Leerzeilen geschrieben werden,
  wenn sie nur eine Zeile lang sind.
```

```
.. note:: A short note.
```

8.3.3 Zeilenlänge

Die Maximale Zeilenläge beträgt 145 Zeichen.

8.3.4 Überschriften

Use the following symbols to create headings:

1. # with overline
2. * with overline
3. =
4. -
5. ^
6. "

Beispiel:

```
##################
H1: document title
##################

Introduction text.


*********
Sample H2
*********

Sample content.


**********
Another H2
**********

Sample H3
=========

Sample H4
---------

Sample H5
^^^^^^^^^
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
Sample H6
"""""""""""

And some text.

Wenn Sie mehr als die Überschriftsebene 4 (d. h. H5 oder H6) benötigen, sollten Sie ein␣
↪neues Dokument erstellen.

In einem Dokument sollte es nur eine H1 geben.
```

Bemerkung: See also Sphinx's documentation about sections[1].

8.3.5 Code-Blöcke

Verwenden Sie die Direktive code-block **und** geben Sie die Programmiersprache an. Als Beispiel:

```
.. code-block:: python

  import this
```

8.3.6 Links und Referenzen

Verwenden Sie Fußnoten für Links und Referenzen mit der Direktive `target-notes`. Als Beispiel:

```
#############
Some document
#############

Some text which includes links to `Example website`_ and many other links.

`Example website`_ can be referenced multiple times.

(... document content...)

And at the end of the document...

********
Verweise
********

.. target-notes::

.. _`Example website`: http://www.example.com/
```

[1] http://sphinx.pocoo.org/rest.html#sections

8.3.7 References

8.4 Wie Sie beitragen können:

8.4.1 Als Benutzer:

- Feature-Vorschlag
- Verbesserungsvorschlag
- Fehlerbericht
- Testen

8.4.2 Als Entwickler

- Offene Probleme beheben
- Funktions-/Erweiterungsvorschläge machen und diese implementieren
- Verbessern der Dokumentation

8.5 Betreuer

Das Asvin-Projekt wird von einem dynamischen Team geleitet. Alle Entwicklungsaktivitäten, wie z.B. das Überprüfen und Zusammenführen von Pull-Requests, Vorschläge für neue Funktionen und die Veröffentlichung der Roadmap, werden von den Hauptbetreuern. geleitet.

Bitte nehmen Sie keine Änderungen vor, ohne vorher die Genehmigung von Mirko Ross von (m.ross@asvin.io) einzuholen.

KAPITEL 9

Glossary

KAPITEL 10

Releases

HTTP Routing Table

/api

/auth

/device

/firmware

/firmware:id