Sensordaten zur KI-Wissensbildung

Mit Hilfe geeigneter Algorithmen aus dem Umfeld der Künstlichen Intelligenz (KI) lässt sich die Entwicklung einer Sensorikanwendung erheblich verbessern. Die größte Herausforderung ist allerdings das Bereitstellen geeigneter Sensordaten, die als gelabelte Trainingsdaten zur Modellbildung von den KI-Algorithmen benötigt werden. Mit PyDSlog steht jetzt ein Softwarebaukasten mit einer Python-Bibliothek und C/C++-Code zur Verfügung, der den gesamten Entwicklungs-Workflow deutlich erleichtert.

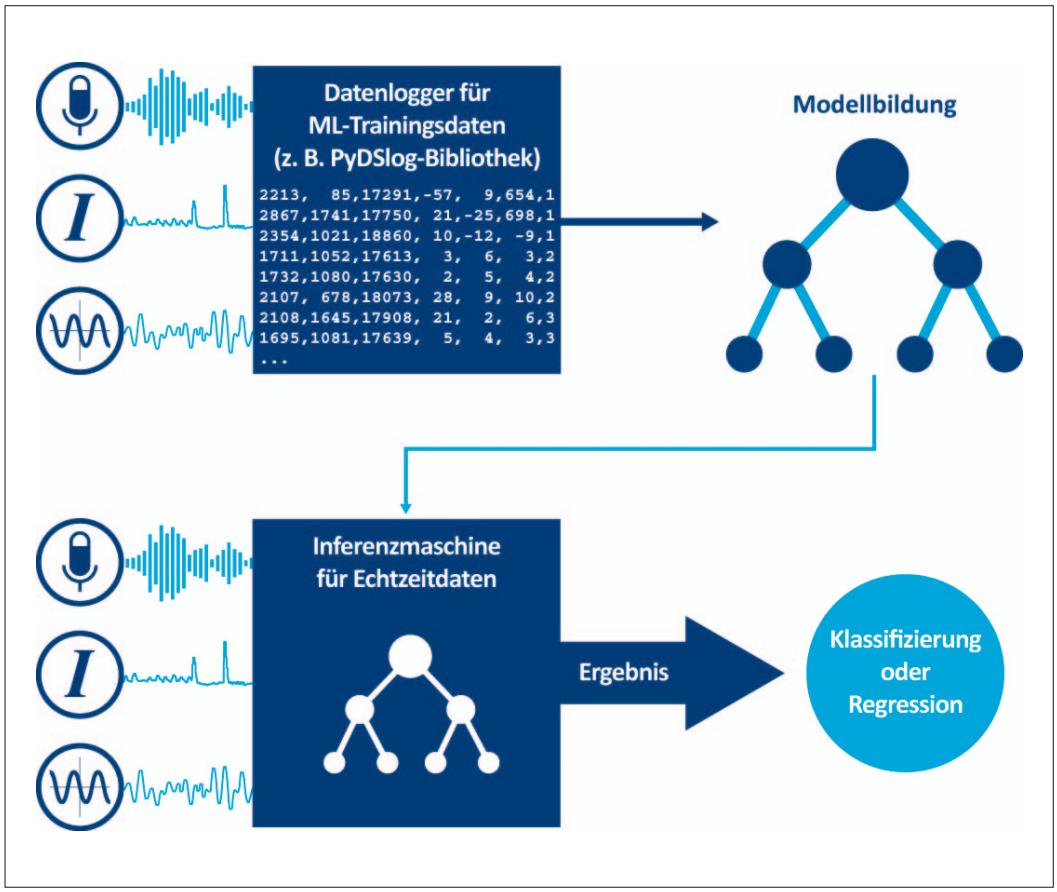

Bild 1: Der Einsatz von Machine-Learning-Algorithmen für eine Sensorikanwendung besteht aus zwei Phasen. In einer Trainingsphase werden aus den zu einer bestimmten Anwendung gehörenden Sensoren zunächst Historiendaten mit Merkmalsvektoren in einer CSV-Datei gesammelt und anschließend zur Modellbildung genutzt. In der anschließenden Inferenzphase wird jeweils ein einzelner Merkmalsvektor mit Echtzeitsensordaten an Hand des mathematischen Modells per Supervised Machine Learning analysiert und dabei z. B. klassifiziert.

Die Problematik

In unzähligen Sensorsystemen wird eine in speziellen Hochsprachen erstellte Firmware genutzt, die den jeweils gewünschten Zusammenhang zwischen Ein- und Ausgangssignalen anhand von statischen Regeln herstellt, die auf lexikalisches Wissen basieren. Ein typischer Anwendungsfall wäre beispielsweise eine komplexe Sensorikapplikation zur Zustandsüberwachung der elektrischen Antriebe einer Maschine mit Hilfe einer Stromsensorik: Die Embedded-Systems-Firmware im Sensorsystem verarbeitet die analogen Rohdaten der einzelnen Sensorelemente und liefert anhand eines programmierten Regel-basierten Messverfahrens (z. B. welche Frequenzen und Amplituden sind jeweils zulässig?) das gewünschte digitale Ausgangssignal.

Klassische Firmware-Entwicklungen sind in der Regel recht aufwändig und über die gesamte Produktlebensdauer betrachtet, relativ inflexibel. Jede noch so kleine Änderung der Anforderungen löst einen neuen Entwicklungszyklus aus. Durch die zahlreichen Weiterentwicklungen im Bereich der künstlichen Intelligenz (KI) und des maschinellen Lernens (ML) ist nun ein weiterer Lösungsansatz möglich: Zwischen die Ein- und Ausgangsdaten eines Mikrorechners wird ein lernfähiger Algorithmus geschaltet und mittels spezieller Trainingsdaten für eine bestimmte Aufgabenstellung konfiguriert. Dabei entsteht ein mathematisches Modell, das den jeweiligen Zusammenhang der Ein- und Ausgänge abbildet. Anforderungsänderungen werden durch eine erneute Trainingsphase und mit Hilfe zusätzlicher Referenzdaten umgesetzt. Grundsätzlich lässt sich mit dieser Vorgehensweise jedes Problem, dessen Zusammenhang zwischen Ein- und Ausgängen durch ein mathematisches Modell beschreibbar ist, lösen.

Input für den Data Scientisten

Ein Supervised-Machine-Learning-Projekt in der industriellen Sensorik, also im Umfeld von Maschinen und Anlagen, ist ein relativ komplexes Vorhaben, für das umfangreiches Spezialwissen aus unterschiedlichen Disziplinen erforderlich ist. Zur Datenanalyse, Datenvorbereitung, Algorithmenauswahl, Modellbildung und Ergebnisverifikation wird vielfach ein Data Scientist aus dem IT-Umfeld in das Projekt eingebunden. Der geht in der Regel davon aus, dass für seine Arbeit bereits ausreichend Daten in der erforderlichen Qualität zur Verfügung stehen. In der Automatisierung ist das aber sehr oft nicht der Fall – hier gibt es entweder überhaupt keine oder in Hinblick auf die zu lösende Aufgabe unbrauchbare Daten. Insofern werden dann schnell mal einige Sensoren oder ein auf den ersten Blick geeignetes Starterkit in einem Webshop bestellt und an einer Maschine installiert, um überhaupt ein paar Daten zu erfassen, mit denen der Datenwissenschaftler loslegen kann. Da sich in ML-Projektteams häufig keiner

Autor:
Klaus-Dieter Walter, CEO
SSV Software Systems GmbH
www.ssv-embedded.de

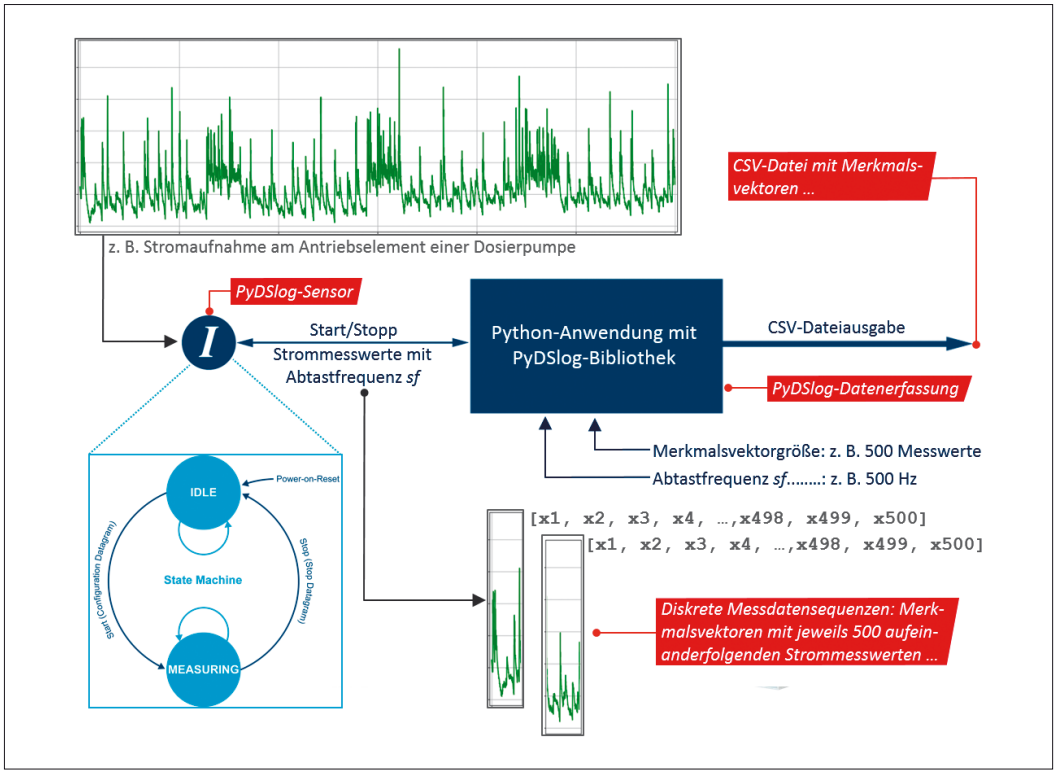

Bild 2: Die primäre Aufgabe von PyDSlog ist das Speichern von Sensorausgangsdaten als Merkmalsvektoren (Feature Vectors) in einer CSV-Datei. Jeder Feature Vector fasst die parametrisierbaren Eigenschaften eines Zustandsmusters in Form eines Arrays einzelner Zahlenwerte zusammen, zum Beispiel die diskreten Messdaten eines Stromsensors x1 bis x500 zu den Abtastzeitpunkten t1 bis t500. Die PyDSlog-Datenerfassung erfolgt in der Regel auf einem Edge-Gateway. In den Sensoren ist eine spezielle State Machine erforderlich.

so richtig mit Sensormerkmalen und -eigenschaften, Signalaufbereitung, Abtasttheoremen, Störgrößen, Filterschaltungen, A/D-Wandlung usw. auskennt, bekommt der Data Scientist meistens völlig „verrauschte“ und bestenfalls bzgl. der Quantität geeignete Maschinendaten geliefert, mit denen er auch nicht so richtig weiterkommt. Viele Vorhaben bleiben derzeit an dieser Stelle „hängen“ und warten auf den „richtigen“ Sensor. Um genau diese Projektphase erfolgreich zu durchlaufen, wurde PyDSlog entwickelt.

PyDSlog

PyDSlog ist ein Baukasten aus Wissens- und Softwarefunktionsbausteinen zur Aufzeichnung von beliebigen Sensormesswerten in CSV-Dateien, um die Sensorrohdaten zum einen für explorative und deskriptive Datenanalysen sowie zum anderen für das Training von ML-Algorithmen zu nutzen. Mit Hilfe des durch die Trainingsphase entstandenen Machine-Learning-Modells kann anschließend eine Inferenzmaschine im Praxiseinsatz bestimmte Muster in Echtzeitmessdaten identifizieren und Ereignisse per Klassifizierung oder Regression vorhersagen.

Für die CSV-Datei wird jede einzelne Sensormessung als Zahlenwert in einem Merkmalsvektor (Feature Vector) eingefügt. Ein solcher Vektor (Array aus diskreten Zahlenwerten) repräsentiert die Messwerte an den jeweiligen Sensorikeingängen zu einem bestimmten Zeitpunkt. Mit anderen Worten: Der einzelne Merkmalsvektor beschreibt ein zeitpunktspezifisches Zustandsmuster der Sensormesswerte. Die per PyDSlog erzeugte CSV-Datei bildet somit ein Datenset aus gleichartigen Merkmalsvektoren, die zu unterschiedlichen Zeitpunkten erfasst wurden.

Die einzelnen Bausteine

Eine PyDSlog-Anwendung besteht grundsätzlich aus zwei elementaren Bausteinen, die über eine einfache Kommunikationsverbindung miteinander gekoppelt sind und jeweils folgende Merkmale aufweisen:

Datenerfassung: Python-basierte Anwendung auf einem Edge-Gateway oder PC, um über die von den Sensoren erhaltenen Daten die erforderlichen Merkmalsvektoren

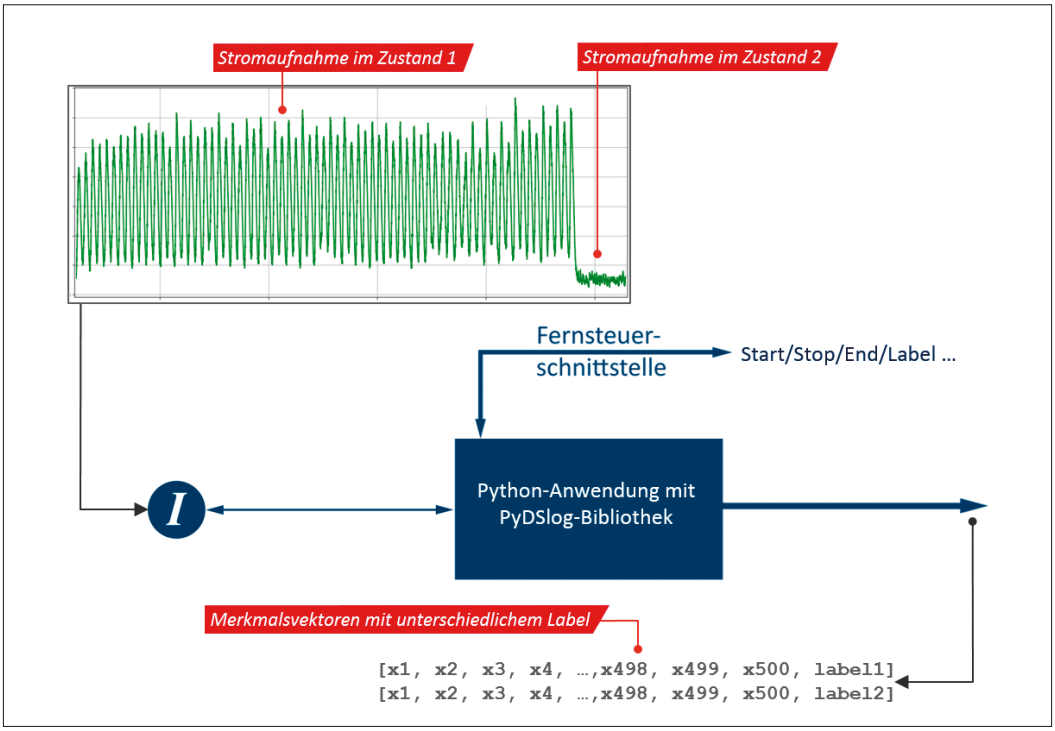

Bild 3: Mit der PyDSlog-Fernsteuerschnittstelle lässt sich die Aufzeichnung von Merkmalsvektoren in einer CSV-Datei jederzeit anhalten und bei einem veränderten Zustand am Eingang der Sensorik wieder fortsetzen. Neben den Verhaltensanweisungen für Start, Stopp und Ende lässt sich ein optionales Zustandskennzeichen (Label) an PyDSlog übermitteln, das an die diskreten Sensormessdaten eines Vektors angefügt wird. Damit kann eine bestimmte Gruppe von Vektoren einem klassifizierten Zustand zugeordnet werden.

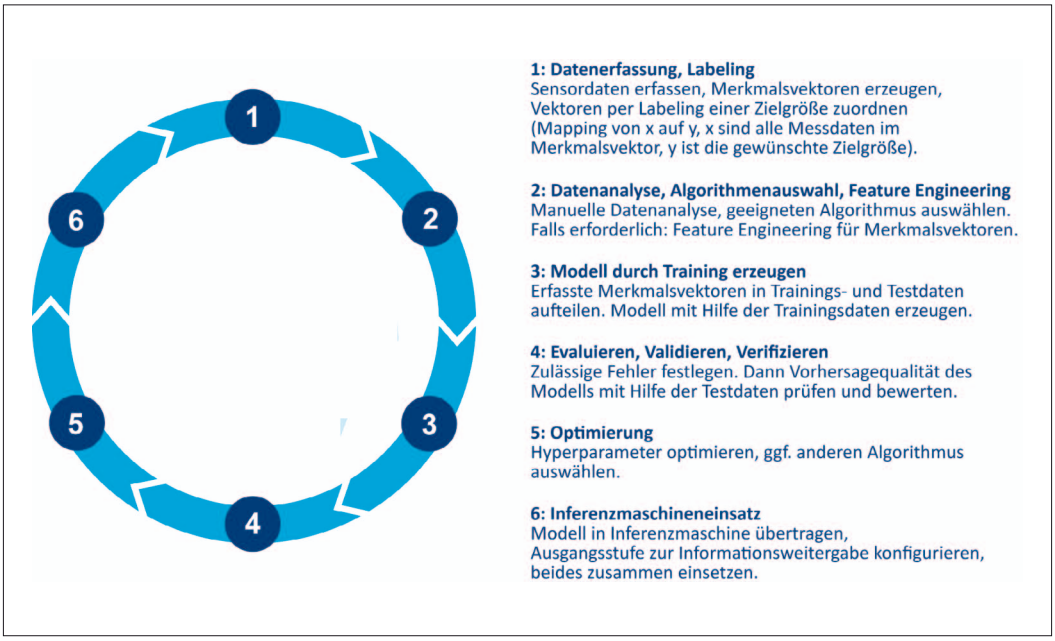

Bild 4: Der gesamte Engineering-Prozess einer Sensorikanwendung mit KI-Algorithmen besteht aus sechs Teilaufgaben. Nach dem Erfassen von Trainings- und Testdaten per PyDSlog und einer explorativen Datenanalyse wird ein Machine-Learning-Modell erzeugt und getestet. Innerhalb des Prozesses sind zwei Iterationsschleifen vorgesehen, um zum einen die Fehlerquote mit verschiedenen Machine-Learning-Algorithmen zu untersuchen und zum anderen die sogenannten Hyperparameter für den gewählten Algorithmus anzupassen.

zu erzeugen und in eine CSV-Datei zu schreiben (Logger für ML-Trainingsdaten). Zwischen der Datenerfassung und den Sensoren existiert eine Hardware-agnostische Kommunikation über drei unterschiedliche Datagramme (siehe Tabelle 1). Jedes Datagramm ist einem bestimmten Zustand einer Sensor-State-Machine zugeordnet. Zum Speichern der CSV-Daten muss die Anwendung einen Schreibzugriff auf ein geeignetes Dateisystem haben.

Sensor: Funktionseinheit mit mindestens einem Sensorelement und einer digitalen Kommunikationsschnittstelle sowie einem Mikrocontroller, in dem sich die PyDSlog-Zustandsmaschine und Datagramm-Kommunikation implementieren lassen. Die Aufgabe des Sensors in einer PyDSlog-Anwendung ist es, die einzelnen Messwerte mit der von der Datenerfassung gewünschten Abtastfrequenz sf zu erfassen und als (Messdaten-) Datagramme an die Datenerfassung zu schicken.

Zusammenhänge zwischen Datenerfassung und Sensor

Bild 2 illustriert an einem Beispiel die wesentlichen Zusammenhänge zwischen Datenerfassung und Sensor. Das Sensorelement ist in diesem Fall ein einfacher Stromsensor, z. B. ein Shunt-Widerstand plus 1-Kanal-A/D-Wandler, der von einem 8-, 16- oder 32-Bit-Single-Chip-Mikrocontroller per I^2C angesteuert wird. Über diesen Mikrocontroller wird auch die Datagramm-Kommunikation zur Datenerfassung gesteuert. Zum Start der CSV-Messdatenerfassung übermittelt die Python-Anwendung mit Hilfe der PyDSlog-Bibliotheksfunktionen ein Konfigurations-Datagramm mit der gewünschten Abtastfrequenz sf (z. B. 500 Hz) an den Sensor. Anschließend überträgt der Sensor fortlaufend mit der jeweiligen Frequenz (bei 500 Hz also alle 2 Millisekunden) 16-Bit-Integer-Strommesswerte als Messdaten-Datagramme an die Datenerfassung. Über die voreingestellte Merkmalsvektorgröße (z. B. 500 Messwerte je Vektor) setzt die entsprechende Bibliotheksfunktion die einzelnen Vektoren zusammen und übergibt jeden vollständigen Vektor als Textzeile im CSV-Format an das Dateisystem, um es an eine Datei anzufügen. Bei sf = 500 Hz und 500 Messpunkten je Vektor repräsentiert jeder einzelne Merkmalsvektor in der CSV-Datei somit eine Zeitspanne von 1 Sekunde.

Der gesamte Datagramm-Dialog zwischen Datenerfassung und Sensor kann sowohl über einfache serielle Schnittstellen (UART, RS232, RS485) als auch per MQTT erfolgen. Für die PyDSlog-Datenerfassung wurde eine Python-Bibliothek entwickelt. Bezüglich der Sensorik existieren zurzeit verschiedene C/C++-Beispielcodes mit Zustandsmaschinen-Template und serieller Datagramm-Kommunikation, die bisher an Arduino und STM32-Nucleo-Boards angepasst wurden. Eine Adaption für das IoT-Betriebssystem RIOT befindet sich in der Entwicklung. Alle Softwarekomponenten stehen über Open Source-Lizenzen jedem interessierten Anwender kostenlos zur Verfügung.

Fernsteuerschnittstelle für das Labeling

Soll ein Supervised-Machine-Learning-Algorithmus in den Echtzeitmessdaten eines Stromsensors auto-

Datagramm-Typ	Sensorzustand	Beschreibung
Konfiguration (Start)	Idle	Wird von der PyDSlog-Datenerfassung an den Sensor übermittelt. Dieses Datagramm erzeugt im Sensor einen Zustandsübergang von Idle nach Measuring und startet somit eine Datenerfassung. In einem Konfigurations-Datagramm werden Parameter für die gewünschte Abtastfrequenz sf und eine Liste der jeweils zu aktivierenden Kanalnummern cn[] (nur bei Sensoren mit mehr als einem Sensorelement) übertragen.
Messdaten	Measuring	Werden mit der Abtastfrequenz sf vom Sensor fortlaufend an die Datenerfassung übertragen. In jedem Messdatendatagramm ist jeweils ein Messwert für jeden aktiven Kanal enthalten.
Stopp	Measuring	Dieses Datagramm erzeugt im Sensor einen Zustandsübergang von Measuring nach Idle und beendet eine laufende Datenerfassung mit der voreingestellten Abtastfrequenz sf und den jeweils aktivierten Kanalnummern cn[].

Tabelle 1: Zwischen dem Sensor und der Datenerfassung einer PyDSlog-Anwendung existiert eine Hardware-agnostische Kommunikationsverbindung zur Übermittlung von einfachen Datagrammen. Die gesamte Kommunikation basiert auf drei unterschiedlichen Datagramm-Typen, die jeweils zwei verschiedenen Sensorik-Systemzuständen zugeordnet sind

matisch unterschiedliche Betriebszustände erkennen, werden für die Trainingsphase entsprechend gekennzeichnete Merkmalsvektoren benötigt. Dafür muss an alle Messdatenvektoren vor dem Training in einem zusätzlichen Arbeitsschritt ein entsprechendes Zustandskennzeichen (Label) angefügt werden. Dieses Label klassifiziert den jeweiligen Betriebszustand, zu dem im Beispiel der Bild 2 die diskreten Strommessdaten des jeweiligen Vektors gehören.

Das Labeling von Zustandsdaten ist in der Praxis häufig mit dem Problem behaftet, dass in vielen Projekten lediglich Daten existieren, die einen einzigen Betriebszustand beschreiben. Damit lässt sich dann beispielsweise ein sogenannter One-Class-Classification-Algorithmus trainieren, der zur Anomalieerkennung dienen kann. Dafür sind zunächst einmal keine zusätzlichen Zustandskennzeichen in den Merkmalsvektoren erforderlich. Da jede Anomalie in diesem Fall aber auch ein spezieller Betriebszustand sein kann, wäre ein Werkzeug zur automatischen Zustandsdatenerfassung im Anomaliefall hilfreich.

Zustandskennzeichnung

Um die Zustandskennzeichnungsaufgaben für die zu erfassenden Merkmalsvektoren zu automatisieren, wurde PyDSlog mit einer Fernsteuerschnittstelle ausgestattet. Dieses Interface besteht im Wesentlichen aus vier Nachrichten mit den Namen Start, Stop, End und Label, die über zwei MQTT-Topics verschickt werden. Damit sind folgende Funktionen möglich:

- Start: Eine neue Datenerfassung mit einer bestimmten CSV-Datei starten oder die zuvor über Stop angehaltene Datenerfassung weiter fortsetzen.
- Stop: Die laufende Datenerfassung für die jeweilige CSV-Datei nach dem vollständigen Schreibzugriff für den (letzten) aktuellen Merkmalsvektor sofort anhalten.
- End: Eine laufende Datenerfassung beenden. Die jeweilige CSV-Datei mit bisher erfassten Merkmalsvektoren wird final geschlossen.
- Label: Vorgabe eines numerischen Werts (Label Value), der als Zustandskennzeichen an das Ende jedes Merkmalsvektors angefügt wird.

Bild 3 illustriert die Anwendungsmöglichkeiten der Fernsteuerschnittstelle am Beispiel der Stromdaten eines Antriebselements, das durch externe Ereignisse in verschiedene Zustände gesetzt wird. Zunächst werden eine Zeit lang Merkmalsvektoren im Zustand 1 gesammelt. Dann wird die Datenerfassung angehalten und auf einen anderen Zustand gewartet, um z. B. Messdatenmerkmalsvektoren für den Antriebszustand 2 zu erfassen. ◀