(12) NACH DEM VERTRAG ÜBER DIE INTERNATIONALE ZUSAMMENARBEIT AUF DEM GEBIET DES PATENTWESENS (PCT) VERÖFFENTLICHTE INTERNATIONALE ANMELDUNG

(19) Weltorganisation für geistiges Eigentum
Internationales Büro

(43) Internationales Veröffentlichungsdatum
24. September 2020 (24.09.2020)

WIPO | PCT

(10) Internationale Veröffentlichungsnummer
WO 2020/187843 A1

(51) Internationale Patentklassifikation:
G06F 8/34 (2018.01) *G06F 9/50* (2006.01)
G05B 19/05 (2006.01)

(21) Internationales Aktenzeichen: PCT/EP2020/057130

(22) Internationales Anmeldedatum:
16. März 2020 (16.03.2020)

(25) Einreichungssprache: Deutsch

(26) Veröffentlichungssprache: Deutsch

(30) Angaben zur Priorität:
10 2019 203 639.2
18. März 2019 (18.03.2019) DE

(71) Anmelder: **FRAUNHOFER-GESELLSCHAFT ZUR FÖRDERUNG DER ANGEWANDTEN FORSCHUNG E.V.** [DE/DE]; Hansastraße 27c, 80686 München (DE).

(72) Erfinder: **EMMERICH, Jan Sören**; c/o Fraunhofer-Institut für Materialfluss und Logistik IML, Joseph-von-Fraunhofer-Str. 2-4, 44227 Dortmund (DE). **TEN HOMPEL, Michael**; c/o Fraunhofer-Institut für Materialfluss und Logistik IML, Joseph-von-Fraunhofer-Str. 2-4, 44227 Dortmund (DE). **HAMMERMEISTER, Christian**; c/o Fraunhofer-Institut für Materialfluss und Logistik IML, Joseph-von-Fraunhofer-Str. 2-4, 44227 Dortmund (DE).

(74) Anwalt: **KÖNIG, Andreas** et al.; Schoppe, Zimmermann, Stöckeler, Zinkler, Schenk & Partner mbB, Radlkoferstr. 2, 81373 München (DE).

(81) Bestimmungsstaaten *(soweit nicht anders angegeben, für jede verfügbare nationale Schutzrechtsart)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, ZA, ZM, ZW.

(84) Bestimmungsstaaten *(soweit nicht anders angegeben, für jede verfügbare regionale Schutzrechtsart)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), eurasisches (AM, AZ, BY, KG, KZ, RU, TJ, TM), europäisches (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI,

(54) Title: DEVICE AND METHOD FOR REMOTE PROGRAMMING

(54) Bezeichnung: VORRICHTUNG UND VERFAHREN ZUR FERNPROGRAMMIERUNG

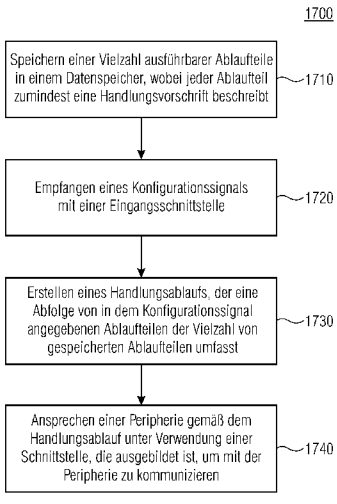

Fig. 17

1710 Saving a plurality of executable process flow elements in a data store, wherein each process flow element describes at least one action specification
1720 Receiving a configuration signal by means of an input interface
1730 Creating an action process flow comprising a sequence of process flow elements of the plurality of stored process flow elements specified in the configuration signal
1740 Addressing a periphery in accordance with the action process flow by using an interface which is designed to communicate with the periphery

(57) Abstract: A device comprises an input interface which is designed to receive a configuration signal; a data store in which a plurality of executable process flow elements is stored, each process flow element describing at least one action specification; a generator which is designed to create an action process flow comprising a sequence of process flow elements of the plurality of stored process flow elements specified in the configuration signal; an interface which is designed to communicate with a periphery; and a processor which is configured to address the periphery in accordance with the action process flow.

(57) Zusammenfassung: Eine Vorrichtung umfasst eine Eingangsschnittstelle, die ausgebildet ist, um ein Konfigurationssignal zu empfangen, einen Datenspeicher, in dem eine Vielzahl ausführbarer Ablaufteile gespeichert ist, wobei jeder Ablaufteil zumindest eine Handlungsvorschrift beschreibt, einen Generator, der ausgebildet ist, um einen Handlungsablauf zu erstellen, der eine Abfolge von in dem Konfigurationssignal angegebenen Ablaufteilen der Vielzahl von gespeicherten Ablaufteilen umfasst, eine Schnittstelle, die ausgebildet ist, um mit einer Peripherie zu kommunizieren, und einen Prozessor, der konfiguriert ist, um die Peripherie gemäß dem Handlungsablauf anzusprechen.

WO 2020/187843 A1

[Fortsetzung auf der nächsten Seite]

SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Veröffentlicht:

— *mit internationalem Recherchenbericht (Artikel 21 Absatz 3)*

Vorrichtung und Verfahren zur Fernprogrammierung

Beschreibung

Die vorliegende Erfindung bezieht sich auf eine Vorrichtung und auf ein Verfahren zur Fernprogrammierung. Die vorliegende Erfindung bezieht sich insbesondere auf ein Fern-Programmierbares Gerät für Regel-basierte Anwendungen bzw. auf ein Fern-Programmierbares Gerät mit Turing-vollständigem Parametersatz für Regel-basierte Anwendungen (FPGRA).

IoT-Devices oder IoT-Vorrichtungen (IoT = Internet of Things, Internet der Dinge) haben verschiedenste, flexible Einsatzmöglichkeiten. Dabei kann ein und dieselbe Hardware-Konfiguration verschiedene Anwendungsfälle besitzen und auch zwischen diesen wechseln. Im Standardfall muss ein Gerät, dessen Anwendung sich ändert, neu programmiert werden. Hierfür wird ein neuer Programmcode auf das Device gespielt und in den Festspeicher (EEPROM) geschrieben. Dieser Prozess ist verhältnismäßig energieaufwendig, da zum einen für das Flashen oder Programmieren des Festspeichers sehr viel Energie benötigt wird, darüber hinaus im Live-Betrieb auch mit einigen Risiken verbunden ist, und zum anderen ein recht großer Programmcode meist über Funk übertragen wird und eine derartige Funkkommunikation ebenfalls recht energieaufwendig ist. Insofern ist diese Vorgehensweise eher ungeeignet für IoT-Devices im Ultra-Low-Power-Bereich (Niedrigenergiebereich).

In diesem Bereich beschränkt man sich bisher darauf, einige wenige vorprogrammierte „Szenarien" im Festspeicher des Devices zu halten, zwischen denen dann gewechselt werden kann. Beispielsweise kann so der Schwellwert einer Abfrage ausgetauscht werden oder der Zeitpunkt/das Intervall, in dem sich das Gerät beim Server meldet, per Funknachricht geändert werden. Hierbei büßt man viel Flexibilität ein, denn wenn ein Anwendungsfall nicht exakt in die vorprogrammierten Szenarien passt, ist es dennoch wieder erforderlich, das Gerät via Funkkommunikation neu zu programmieren.

Wünschenswert wären demnach funkbasierte Vorrichtungen, die mit geringem Energieaufwand flexibel eingesetzt werden können.

Die Aufgabe der vorliegenden Erfindung besteht deshalb darin, Vorrichtungen zu schaffen, die mittels Datenkommunikation unter geringem Energieaufwand flexibel eingesetzt oder programmiert werden können.

Diese Aufgabe wird durch die Gegenstände der unabhängigen Patentansprüche gelöst.

Die Erfinder haben erkannt, dass es von besonderem Vorteil ist, die durch ein Gerät zu implementierenden Funktionalitäten in Form von Ablaufteilen, die jeweils zumindest eine Handlungsvorschrift beschreiben, in dem Gerät zu hinterlegen und vorzuhalten. Eine später von dem Gerät zu implementierende Funktion oder Funktionalität, das bedeutet ein Handlungsablauf, kann durch Empfang eines Konfigurationssignals festgelegt werden, welches lediglich die Reihenfolge und Verknüpfung der bereits im Gerät hinterlegten Ablaufteile angibt. Die Vorrichtung kann somit durch Neusortierung der vordefinierten und bereits hinterlegten Ablaufteile neu programmiert werden, was mit geringem Energieaufwand möglich ist und gleichzeitig eine sehr hohe Flexibilität ermöglicht.

Gemäß einem Ausführungsbeispiel umfasst eine Vorrichtung eine Eingangsschnittstelle, die ausgebildet ist, um ein Konfigurationssignal zu empfangen. Die Vorrichtung umfasst einen Datenspeicher, in dem eine Vielzahl ausführbarer Ablaufteile gespeichert ist, wobei jeder Ablaufteil zumindest eine Handlungsvorschrift beschreibt. Ein Generator ist ausgebildet, um einen Handlungsablauf zu erstellen, der eine Abfolge von in dem Konfigurationssignal angegebenen Ablaufteilen der Vielzahl von gespeicherten Ablaufteilen umfasst. Eine Schnittstelle der Vorrichtung ist ausgebildet, um mit einer Peripherie zu kommunizieren. Ein Prozessor ist ausgebildet, um die Peripherie gemäß dem Handlungsablauf anzusprechen.

Gemäß einem Ausführungsbeispiel umfasst eine Vorrichtung eine Ausgangsschnittstelle, die ausgebildet ist, um ein Konfigurationssignal zu senden, einen Signalgenerator, der ausgebildet ist, um das Konfigurationssignal bereitzustellen, so dass das Konfigurationssignal eine Abfolge von ausführbaren Ablaufteilen einer Vielzahl von ausführbaren Ablaufteilen beschreibt. Die Abfolge von ausführbaren Ablaufteilen beschreibt einen Handlungsablauf mit einer Mehrzahl von Handlungsvorschriften eindeutig.

Weitere Ausführungsbeispiele sind auf ein System mit erfindungsgemäßen Vorrichtungen und auf Verfahren sowie auf ein Computerprogramm gerichtet.

Vorteilhafte Weiterbildungen sind in den abhängigen Patentansprüchen definiert.

Bevorzugte Ausführungsbeispiele der vorliegenden Erfindung werden nachfolgend bezugnehmend auf die beiliegenden Zeichnungen erläutert. Es zeigen:

Fig. 1 ein schematisches Blockschaltbild einer Vorrichtung gemäß einem Ausführungsbeispiel;

Fig. 2 ein schematisches Blockschaltbild eines Datenspeichers gemäß einem Ausführungsbeispiel;

Fig. 3 ein schematisches Blockschaltbild einer Vorrichtung gemäß einem Ausführungsbeispiel, die Schnittstellen aufweist, um sowohl eine drahtlose Kommunikation als auch eine drahtgebundene Kommunikation bereitzustellen;

Fig. 4 eine schematische Repräsentation eines beispielhaften Handlungsablaufs gemäß einem Ausführungsbeispiel;

Fig. 5 ein schematisches Blockschaltbild eines Teils eines Regelbaums gemäß einem Ausführungsbeispiel, wobei beispielhaft zwei ausführbare Ablaufteile und durch Ablaufknoten repräsentiert werden;

Fig. 6a ein schematisches Blockschaltbild einer Verschaltung eines Ablaufknotens in einem Regelbaum gemäß einem Ausführungsbeispiel;

Fig. 6b eine schematische Darstellung einer Verknüpfung eines Ablaufknotens in einem Regelbaum, wobei die implementierte Operation gemäß einem Ausführungsbeispiel eine Arithmetik-Operation ist;

Fig. 6c eine schematische Darstellung einer Verknüpfung des Ablaufknotens in dem Regelbaum, bei dem die implementierte Funktion gemäß einem Ausführungsbeispiel eine Speicheroperation ist;

Fig. 6d eine schematische Darstellung des Ablaufknotens gemäß einem Ausführungsbeispiel, bei dem mittels Verknüpfungen erhaltene Informationseingänge gemäß einem Ausführungsbeispiel in einer auszuführenden Aktion münden;

Fig. 7 eine beispielhafte Darstellung von Computer-implementierten Instruktionen gemäß einem Ausführungsbeispiel zur beispielhaften Darstellung unterschiedlicher durch Knoten implementierter Funktionen;

Fig. 8 eine beispielhafte Darstellung von Code gemäß einem Ausführungsbeispiel, der Instruktionen wiedergibt, die ein Generator einer erfindungsgemäßen Vorrichtung anweisen, ein Konfigurationssignal auf eine Funktions-ID auszuwerten;

Fig. 9a-c beispielhafte Darstellungen von Code gemäß einem Ausführungsbeispiel zur Definition einer Knotenstruktur;

Fig. 10a einen beispielhaften Code gemäß einem Ausführungsbeispiel, bei dem eine maximale Anzahl von Knoten des Handlungsablaufs definiert wird;

Fig. 10b einen beispielhaften Code gemäß einem Ausführungsbeispiel, der anzeigt, dass eine Knotennummer bis zu einer maximalen Anzahl von Knoten oder Aktionen vollständig durchlaufen wird;

Fig. 10c einen beispielhaften Code gemäß einem Ausführungsbeispiel zum Ausführen eines Handlungsablaufs durch einen Prozessor;

Fig. 10d eine beispielhafte Darstellung von Code gemäß einem Ausführungsbeispiel, der Instruktionen wiedergibt, die ein Prozessor einer erfindungsgemäßen Vorrichtung anweisen, einen Knotentyp auszuwerten;

Fig. 10e einen bespielhaften Code gemäß einem Ausführungsbeispiel, der einen Prozessor instruiert, um den nächsten auszuführenden Knoten zu bestimmen;

Fig. 11 ein schematisches Blockschaltbild einer Vorrichtung gemäß einem Ausführungsbeispiel, die Eingänge, Ausgänge und eine innere Logik umfasst;

Fig. 12 ein schematisches Blockschaltbild einer Vorrichtung gemäß einem Ausführungsbeispiel, wobei an den Ausgängen die Peripherie in Form eines Schalters drahtgebunden oder drahtlos angeschlossen ist;

Fig. 13 ein schematisches Blockschaltbild einer Vorrichtung gemäß einem Ausführungsbeispiel, bei der die Eingänge mit Sensoren in beliebiger Anzahl verbunden sind, um Sensorsignale zu empfangen;

Fig. 14 ein schematisches Blockschaltbild einer Vorrichtung gemäß einem Ausführungsbeispiel, die mit der Peripherie verbunden ist und die als Sensor/Aktor-Logik bzw. als Sender und Empfänger, etwa für eine Smart-Home-Einrichtung, konfiguriert ist;

Fig. 15 ein schematisches Blockschaltbild eines Systems gemäß einem Ausführungsbeispiel, das beispielhaft die Vorrichtung aus Fig. 1 umfasst;

Fig. 16 ein schematisches Blockschaltbild eines Systems gemäß einem Ausführungsbeispiel, das Vorrichtungen umfasst, die ausgebildet sind, um einen Datenfluss zum Austausch von Logik und/oder Parametern oder Soll-Werten auszutauschen;

Fig. 17 ein schematisches Ablaufdiagramm eines Verfahrens gemäß einem Ausführungsbeispiel, das beispielsweise von einer hierin beschriebenen fernprogrammierbaren Vorrichtung, etwa der Vorrichtung ausgeführt werden kann;

Fig. 18 ein schematisches Flussdiagramm eines Verfahrens zum Bereitstellen eines Konfigurationssignals gemäß einem Ausführungsbeispiel;

Fig. 19 ein schematisches Blockschaltbild einer Vorrichtung gemäß einem Ausführungsbeispiel, die als Bewegungstracker fungieren kann;

Fig. 20 ein schematisches Blockschaltbild einer Vorrichtung gemäß einem Ausführungsbeispiel, die als Diebstahl-Sicherung fungieren kann; und

Fig. 21 ein schematisches Blockschaltbild einer Vorrichtung gemäß einem Ausführungsbeispiel, die eine intelligente Puls-Uhr bereitstellen kann.

Bevor nachfolgend Ausführungsbeispiele der vorliegenden Erfindung im Detail anhand der Zeichnungen näher erläutert werden, wird darauf hingewiesen, dass identische, funktionsgleiche oder gleichwirkende Elemente, Objekte und/oder Strukturen in den unter-

schiedlichen Figuren mit den gleichen Bezugszeichen versehen sind, so dass die in unterschiedlichen Ausführungsbeispielen dargestellte Beschreibung dieser Elemente untereinander austauschbar ist bzw. aufeinander angewendet werden kann.

Manche der nachfolgend beschriebenen Ausführungsbeispiele sind unter Verwendung oder anhand einer beispielhaften Programmiersprache, d. h. anhand von Code beschrieben. Details dieser Ausführungsbeispiele, die sich beispielsweise aufgrund der Semantik und/oder anhand der konkreten Befehlsimplementierungen ergeben, sind dabei lediglich beispielhaft zu verstehen und zum besseren Verständnis eingefügt. Die gewählten Codeimplementierungen beschränken die Ausführungsbeispiele dabei weder auf die gewählte Programmiersprache noch auf die konkrete Befehlsabfolge oder die Bezeichnung von Variablen, sofern dies nicht explizit anders angegeben ist. Insbesondere beziehen sich die hierin erläuterten Ausführungsbeispiele auf alternative oder zusätzliche Ausführungsbeispiele mit gleicher, ähnlicher und insbesondere äquivalenter technischer Wirkung.

Nachfolgende Ausführungsbeispiele werden im Zusammenhang mit IoT-Vorrichtungen oder IoT-Devices, was ein Synonym ist, beschrieben, wobei „IoT" den Begriff „Internet of Things" (Internet der Dinge) bezeichnet und für Geräte steht, die über einen üblicherweise schmalbandigen Kanal mit anderen Einrichtungen kommunizieren, wobei hierfür sowohl Direktverbindungen (Peer-to-Peer) als auch Infrastruktur-basierte Netzwerke verwendet werden können.

Obwohl im Bereich der schmalbandigen Kommunikation besondere technische Vorteile erhalten werden, sind die hierin beschriebenen Ausführungsbeispiele nicht hierauf beschränkt, insbesondere da die Vorteile der Flexibilität und Energieeinsparung auch bei breitbandigeren Kanälen erhalten werden kann. Daraus wird ebenfalls deutlich, dass die hierin beschriebenen Ausführungsbeispiele zwar im Bereich des klassischen Mobilfunks (etwa 3G/4G/LTE – Long Term Evolution – oder 5G) eingesetzt werden können, dass aber sowohl andere drahtlose Kommunikationsprotokolle implementiert werden können wie etwa lokale Drahtlosnetzwerke (WLAN), großflächige Drahtlosnetzwerke (WMAN) oder beliebige andere drahtlose Kommunikationsprotokolle, dass die beschriebenen Vorteile aber auch im Bereich der drahtgebundenen Kommunikation erhalten werden können.

Nachfolgende Ausführungsbeispiele beziehen sich auf die Verwendung von Integer-Zahlen, d. h., ganzzahliger Werte, bspw. für Ergebnisse von Knoten von Handlungsabläufen bzw. deren graphischer Repräsentanz. Obwohl dies im Hinblick auf Speichereffizienz

vorteilhaft ist, sind die vorliegenden Ausführungsbeispiele nicht hierauf beschränkt. Es ist vielmehr möglich, dass Operationen eines Knoten andere Werte, etwa Fließkomma-Werten (bspw. Datentyp „float") verwenden. Hierfür kann in der Implementierung das ein Datenfeld, das bspw. mit „res" bezeichnet ist, und das Ergebnis des Knotens repräsentieren kann, sogar weiterhin ein Integer bleiben, da die Information (also die Bits) im „res"-Feld durch sogenanntes „casten" (auswerten) einfach als float (Fließkommazahl) interpretieren kann ohne Informationsverlust zu erleiden.

Fig. 1 zeigt ein schematisches Blockschaltbild einer Vorrichtung 10 gemäß einem Ausführungsbeispiel. Die Vorrichtung 10 umfasst eine Eingangsschnittstelle 12, die ausgebildet ist, um ein Konfigurationssignal 14 zu empfangen. Die Eingangsschnittstelle 12 kann eine drahtlose oder drahtgebundene Kommunikationsschnittstelle oder Interface sein, so dass das Konfigurationssignal dementsprechend ein drahtgebundenes Signal oder ein drahtloses Signal sein kann.

Die Vorrichtung 10 umfasst einen Datenspeicher, in dem eine Vielzahl ausführbarer Ablaufteile 18 gespeichert ist, die beispielhaft als „a", „b", „c", „d", oder „e" dargestellt sind, obwohl eine Anzahl beliebig ist, etwa zumindest drei, zumindest vier, zumindest 10, zumindest 50 oder höher, etwa zumindest 100. Jeder Ablaufteil 18_1 bis 18_5 kann als Codesegment oder Codeabschnitt verstanden werden, der zumindest eine Handlungsvorschrift beschreibt, beispielsweise eine Verarbeitung oder Manipulation einer Eingangsgröße hin zu einer Ausgangsgröße, worauf später noch detailliert eingegangen wird. Die Ablaufteile 18_1 bis 18_5 sind jedoch eindeutig voneinander unterscheidbar und stellen beispielsweise Teilprogramme oder Teilcodes dar.

Die Vorrichtung 10 umfasst einen Generator 22, der ausgebildet ist, um einen Handlungsablauf 24 zu erstellen. Der Handlungsablauf 24 kann dabei eine Kombination mehrerer Ablaufteile 18_1 bis 18_5 sein, wobei jeder Ablaufteil 18_1 bis 18_5 optional in dem Handlungsablauf 24 vorkommen kann, was eine Möglichkeit zum mehrfachen Vorkommen miteinschließt. Beispielhaft umfasst der Handlungsablauf 24 eine Anzahl von vier Ablaufteilen 18_3, 18_1, 18_4 und 18_2 in der genannten Reihenfolge, die als c, a, d, b darstellbar ist. Sowohl die gewählte Reihenfolge als auch die Anzahl der Ablaufteile 18_1 bis 18_4 in dem Handlungsablauf 24 sind hierbei beispielhaft gewählt.

Die Anzahl voneinander unterscheidbarer unterschiedlicher Ablaufteile 18_i kann mit $i = 1, \ldots N$ mit $N > 1, > 2, > 3, > 4, > 5, > 10, > 50$ oder mehr, etwa > 100 beliebig skalierbar

sein. Eine Anzahl von M Ablaufteilen in dem Handlungsablauf 24 kann ebenfalls einen beliebigen Wert aufweisen, wobei die Anzahl von M Ablaufteilen 18 in dem Handlungsablauf 24 eine Teilmenge der N in dem Datenspeicher 16 hinterlegten Ablaufteile ist, wobei innerhalb der Teilmenge M ein Ablaufteil mehrfach vorkommen kann. Das bedeutet, die in dem Handlungsablauf implementierten Ablaufteile sind eine Teilmenge der gespeicherten Handlungsabläufe. Der Handlungsablauf 24 kann somit zumindest eine Teilmenge der in dem Datenspeicher gespeicherten Vielzahl von ausführbaren Ablaufteilen umfassen. Der Prozessor 31 kann ausgebildet sein, um zum Ansprechen der Peripherie 32 die Teilmenge von ausführbaren Ablaufteilen gemäß dem Handlungsablauf 24 zu durchlaufen und in einem ersten Ablaufteil zumindest eine Eingangsgröße zu verarbeiten, um eine Ausgangsgröße zu erhalten, und um die Ausgangsgröße als Eingangsgröße eines zweiten Ablaufteils zu verwenden, wie es beispielsweise anhand der Fig. 4 näher erläutert ist.

Der Generator 22 ist ausgebildet, um die für den Handlungsablauf 24 zu verwendenden Ablaufteile aus dem Datenspeicher 16 zu lesen und zu dem Handlungsablauf 24 zu verknüpfen, was sowohl die Implementierung einer Reihenfolge der Ablaufteile 18_1 bis 18_i umfasst als auch eine Verknüpfung der einzelnen Ablaufteile. So kann eine Ausgangsgröße eines ersten Ablaufteils, etwa 18_1, eine Eingangsgröße eines darauffolgenden Ablaufteils, etwa 18_4, sein, so dass durch entsprechende Speicherverweise eine Übergabe der Parameter erfolgen kann. Die anzuwendende Vorschrift bezüglich der Verknüpfungen kann Teil des Konfigurationssignals 14 sein, das beispielsweise als c', a', d' oder b' angedeutete Informationen 26_1 bis 26_4 aufweisen kann.

So kann eine Information 26_3 (c') eine Instruktion zur Verwendung des Ablaufteils 18_3 in dem Handlungsablauf 24 wiedergeben. Ähnlich kann die Präsenz einer Information 26_1 (a') eine Instruktion zur Verwendung des Ablaufteils 18_1 in dem Handlungsablauf 24 wiedergeben. Informationen 26_4 (d') sowie 26_2 (b') können in ähnlicher Weise Instruktionen zur Verwendung der Ablaufteile 18_4 und 18_2 in dem Handlungsablauf 24 aufweisen. Eine Reihenfolge der Ablaufteile 18_1 bis 18_4 in dem Handlungsablauf 24 kann in dem Konfigurationssignal 14 explizit angegeben sein, kann aber auch implizit durch eine Reihenfolge der Information 26_3, 26_1, 26_4 und 26_2 enthalten sein, was eine zusätzliche Reduktion übertragener Daten ermöglicht. Der Generator 22 ist ausgebildet, um das Konfigurationssignal 14 auf die Informationen 26 auszuwerten und um den Handlungsablauf 24 so zu erstellen, dass eine Abfolge von in dem Konfigurationssignal durch die Informationen 26 angegebenen Ablaufteilen 18, die in dem Datenspeicher 16 gespeichert sind, umfasst.

Der Prozessor 22 kann ausgebildet sein, um bei Erhalt eines ggf. ersten Konfigurationssignals einen neuen oder ersten Handlungsablauf zu erstellen. Der Prozessor 22 kann ausgebildet sein, um bei Erhalt eines neuen, weiteren oder anderen Konfigurationssignals 14 den Handlungsablauf 24 durch einen neuen, weiteren bzw. anderen Handlungsablauf zu ersetzen, der in Übereinstimmung mit dem aktualisierten Konfigurationssignal 14 ist. Dadurch wird eine Umprogrammierung der Vorrichtung 10 erhalten, ohne dass hierfür neuer Programmcode erforderlich ist. Gleichzeitig kann bei einer Ausgestaltung des Datenspeichers 34 beispielsweise als flüchtiger Datenspeicher ein geringes Maß an Energie eingesetzt werden, um die Umprogrammierung zu erhalten.

Ausgehend von einem Initialzustand, bei dem beispielsweise alle Knoten des Handlungsablaufs 24 unkonfiguriert oder auf einen Standardwert konfiguriert sind, oder zum Umprogrammieren eines zuvor erstellten Handlungsablaufs 24, kann das Konfigurationssignal 14 eine Abfolge von Informationen aufweisen, die für jeden zu verwendenden Knoten eine Kennung (Knotennummer) sowie einen Typ und Speicherbereiche für Ein- und Ausgänge angibt, so dass durch insgesamt sechs Parameter ein Knoten des Handlungsablaufs 24 eindeutig identifizierbar ist und über die angegebenen Parameter umprogrammierbar ist, sodass mittels des Konfigurationssignals 14 eine Umprogrammierung einzelner Knoten als auch des gesamten Handlungsablaufs möglich ist, um einen neuen, zweiten Handlungsablauf zu erhalten. Der Generator kann ausgebildet sein, um basierend auf dem zweiten Konfigurationssignal eine Neudefinition zumindest eines ausführbaren Ablaufteils des ersten Handlungsablaufs auszuführen, um einen zweiten Handlungsablauf zu erhalten, also einen aktualisierten oder neudefinierten Handlungsablauf. Die Neudefinition des Handlungsablaufs kann dabei ebenso Turing-vollständig sein wie der erste Handlungsablauf. Zur Neudefinition kann der Generator ausgebildet sein, um das zweite Konfigurationssignal auf die erwähnten sechs Parameter auszuwerten, die den zumindest einen ausführbaren Ablaufteil für die Neudefinition bestimmen.

Die Vorrichtung 10 umfasst eine Schnittstelle 28, die eine drahtgebundene oder drahtlose Verbindung mit einer Peripherie 32 ermöglicht. Die Peripherie 32 kann eine beliebige Anzahl, d. h., jeweils null oder mehr, insgesamt jedoch eines oder mehr, von Sensoren, Aktuatoren, Motoren Datenausgabegeräten Datenverarbeitungsgeräten, etwa Server, Schalter und/oder Dateneingabegeräten umfassen. Zu den Sensoren gehören beispielsweise Temperatursensoren, Feuchtigkeitssensoren, Drehratensensoren oder beliebige andere Sensoren. Aktuatoren können beispielsweise Stellglieder sein, die im Bereich der Fluidsteuerung genutzt werden, beliebige andere Bewegungen erzeugen, etwa Schalter oder

Motoren oder dergleichen, während zu Dateneingabegeräten Scanner, Tasten, Tastaturen oder Touchpads gehören. Datenausgabegeräte können beispielsweise Anzeigen, Leuchten oder Displays sein.

Die Vorrichtung 10 umfasst einen Prozessor 32, der ausgebildet ist, um die Peripherie 32 gemäß dem Handlungsablauf 24 anzusprechen. Das bedeutet, der Prozessor 31 ist konfiguriert, um die Ablaufteile 18_1 bis 18_4 in der durch den Handlungsablauf 24 festgelegten Reihenfolge auszuführen, wobei sich der Handlungsablauf 24 auf die Interaktion mit der Peripherie sowie zwischengeschaltete oder zwischengeordnete Verarbeitungsschritte beziehen kann.

Obwohl der Generator 22 und der Prozessor 31 als separate Elemente dargestellt sind, können der Generator 22 und der Prozessor 31 auch Teil einer gemeinsamen Steuereinheit sein, die beispielsweise als Prozessor, Micro-Controller, feldprogrammierbares Gatter-Array (FPGA) oder dergleichen implementiert ist. Alternativ kann jedes der Elemente 22 und 31 für sich derartig implementiert sein.

Der Handlungsablauf 24 kann in einem Datenspeicher 34 hinterlegt sein, der von dem Datenspeicher 16 separat sein kann. Alternativ ist es ebenfalls möglich, dass die Datenspeicher 16 und 34 Speicherbereiche eines gleichen Datenspeichers sind. Bevorzugt ist der Datenspeicher 16 als nicht-flüchtiger Speicher ausgebildet, wobei zu dem nicht-flüchtigen Speichern auch MRAMs gezählt werden (MRAM = magnetoresistive Arbeitsspeicher) und andere Speicher, die einen zumindest kurzfristigen Zustand ohne Energieversorgung ohne Datenverlust überstehen. Der Datenspeicher 34 kann ein flüchtiger oder nicht-flüchtiger Speicher sein, da in Kenntnis des Konfigurationssignals 14 der Handlungsablauf 24 durch Zugriff auf den Datenspeicher wiederherstellbar ist. Ausführungsbeispiele können vorsehen, Informationen des Konfigurationssignals 14 in dem Datenspeicher 16 und/oder dem Datenspeicher 34 zu hinterlegen.

In anderen Worten ermöglicht die Vorrichtung 10 vor der oben genannten Zielsetzung, dass ein hohes Maß an Flexibilität an Anwendungen erhalten werden und für eine Neuprogrammierung der Vorrichtung 10 auf eine Übertragung neuen Programmcodes verzichtet werden kann, je nach Implementierung des Datenspeichers 34, auf ein Schreiben eines Codes in einem Festspeicher wie einem EEPROM, verzichtet werden kann, was insbesondere für den Ultra-Low-Power-Bereich vorteilhaft ist, die Ausführungsbeispiele aber nicht hierauf beschränkt. Ausführungsbeispiele basieren auf der Erkenntnis, dass die

vom Anwender/Programmierer vorgegebene Tätigkeit des Geräts nur ein kleiner Teil des gesamten Programmcodes ist. Ein sehr großer Teil dient, vereinfacht betrachtet, nur der Ansteuerung und Einbindung der Hardware/Peripherie. Außerdem sind die meisten IoT-Vorrichtungen, insbesondere im Ultra-Low-Power-Segment, von recht einfacher Natur, nur in seltenen Fällen werden rekursive Programmaufrufe oder komplex verschachtelte Schleifen eingesetzt, was zum Teil auch an den Hardware-Begrenzungen liegt. Ein gewisser Anteil bzw. ein hoher Anteil lässt sich mit Konzepten gemäß „Wenn X, dann Y, sonst Z" darstellen, wobei hier X durchaus das Ergebnis einer komplexeren Operation sein kann und Y sowie Z ein Funktionsaufruf oder eine weitere Logikabfrage.

Fig. 2 zeigt ein schematisches Blockschaltbild eines Datenspeichers 36, der als gemeinsamer Datenspeicher für die Datenspeicher 16 und 34 wirkt. Ein gemeinsamer Datenspeicher ist jedoch nicht darauf beschränkt, dass die Datenspeicher 16 und 34, die als erster Speicherabschnitt bzw. zweiter Speicherabschnitt wirken, auf den gleichen Speicherprinzipien basieren. Der Speicherabschnitt oder Datenspeicher 16 ist dabei ausgebildet, um die Vielzahl ausführbarer Ablaufteile 18 zu speichern, das bedeutet, die vorgefertigten oder vordefinierten Codeabschnitte. Der Datenspeicher oder Speicherabschnitt 34 kann ausgebildet sein, um den Handlungsablauf 24 zu speichern. Insbesondere kann der Generator 22 ausgebildet sein, um den erstellten Handlungsablauf 24 in dem Speicherabschnitt 34 zu speichern.

Gemäß einem Ausführungsbeispiel ist der Datenspeicher 16 als nicht-flüchtiger Speicher gebildet und der Datenspeicher 34 als flüchtiger Speicher.

Fig. 3 zeigt ein schematisches Blockschaltbild einer Vorrichtung 30 gemäß einem Ausführungsbeispiel, die Schnittstellen 12_1 und 12_2 aufweist, um sowohl eine drahtlose Kommunikation über die Schnittstelle 12_1 als auch eine drahtgebundene Kommunikation über die Schnittstelle 12_2 bereitzustellen. Die Schnittstellen 12_1 und/oder 12_2 können dabei eine bidirektionale Kommunikation ermöglichen, ebenso wie die Schnittstelle 12 der Vorrichtung 10. So kann beispielsweise ein Ergebnis interner Berechnungen oder ausgewerteter Sensoren oder Knoten zurückgemeldet werden.

Die Vorrichtung 30 umfasst eine Steuereinrichtung 38, beispielsweise eine zentrale Recheneinheit (central processing unit – CPU, ein Micro-Controller oder dergleichen), die mit dem beispielhaft nicht-flüchtigen Datenspeicher 16 und dem beispielhaft flüchtigen Datenspeicher 34 verbunden ist und sowohl den Generator 22 als auch den Prozessor 31 der

Vorrichtung 10 implementiert. Die Steuereinrichtung 38 kann mit einer beliebigen Anzahl ≥ 1 Taktgebern (real-time clock – RTC) in Verbindung stehen. Zur Kommunikation mit der Peripherie kann eine Schnitt 28_1 konfiguriert sein, um eine Anzahl von 0, ... v Eingängen zu erfassen, wobei V eine beliebige Anzahl > 0 ist. Die v Eingänge der Schnittstelle 28_1 können beliebige Informationen oder Daten bereitstellen. Eine optionale weitere Schnittstelle 28_2 kann mit einer Anzahl von 0, ..., w Sensoren verbunden sein, so dass bei Konnektierung von Sensoren entsprechende Sensorsignale an die Steuereinrichtung 38 gelangen können, das bedeutet, die Steuereinrichtung 38 kann Zugriff auf die optionalen Eingänge und die optionalen Sensoren aufweisen. Es wird darauf hingewiesen, dass bevorzugt zumindest einer der Werte v oder w > 0 ist und die Vorrichtung 30 bzw. die Peripherie zumindest einen Eingang oder zumindest einen Sensor aufweist. Über eine Schnittstelle 28_3 können andere Systeme angesteuert werden, beispielsweise über einen Kommunikationsbus oder einen Energieübertragungsbus. Die Vorrichtung 30 kann eine optionale Schnittstelle 28_4 zur parallelen (Par) und/oder seriellen (Ser) Kommunikation aufweisen. Eine Schnittstelle 28_5 kann optional vorgesehen sein und eine Anzahl von ≥ 0 Ausgängen bereitstellen, die beispielsweise mit einer Anzeigeeinrichtung und/oder einem Aktuator verbunden sind und/oder Steuersignale für Sensoren oder dergleichen bereitstellen.

Die Vorrichtung 30 kann ausgebildet sein, um das Konfigurationssignal über die Schnittstelle 12_1 und/oder die Schnittstelle 12_2, das bedeutet über ein drahtgebundenes oder drahtloses Kommunikationsnetzwerk zu empfangen. Bevorzugt wird das Konfigurationssignal über ein drahtloses Netzwerk empfangen, da dies ein hohes Maß an Flexibilität für die Vorrichtung 30, insbesondere in Hinblick auf die Positionierung derselben, bietet. Die Schnittstelle 12_2 kann insofern eine Verbindung zu einem drahtlosen oder drahtgebundenen Netzwerk bereitstellen. Das drahtlose Netzwerk kann ein Funknetzwerk umfassen, worunter Mobilfunkanwendungen verstanden werden, beispielsweise 3G, 4G oder 5G.

Nachfolgend wird Bezug genommen auf die Erstellung des Handlungsablaufs aus dem Konfigurationssignal. Das Konfigurationssignal ist dabei so ausgestaltet, dass eine Abfolge von ausführbaren Ablaufteilen in dem Konfigurationssignal angezeigt oder enthalten ist und sowohl die Teilmenge der in dem Datenspeicher 16 hinterlegten Ablaufteile angibt als auch eine Reihenfolge der ausgewählten Ablaufteile in dem Handlungsablauf 24 eindeutig festlegt. Das bedeutet, die Abfolge von ausführbaren Ablaufteilen und damit der Handlungsablauf sind in dem Konfigurationssignal eindeutig festgelegt, auch wenn der auszuführende Programmcode selbst nicht zwingend Teil des Konfigurationssignals ist.

An den Eingängen 28_1 bis 28_4 verbundene Komponenten können ebenso zur Peripherie gehören, wie die an den Ausgängen 28_5 verbundene Komponenten.

Fig. 4 zeigt eine schematische Repräsentation eines beispielhaften Handlungsablaufs 24. Der Handlungsablauf 24 kann einen oder mehrere Regelbäume 44_1 und 44_2 aufweisen oder durch die Regelbäume repräsentiert werden, wobei die Anzahl der Regelbäume 44_1 und/oder 44_2 in dem Handlungsablauf 24 eine beliebige Anzahl von ≥ 1 sein kann. Die Regelbäume 44_1 und 44_2 können dabei parallel oder zeitlich nacheinander, d. h. sequenziell, abgearbeitet werden und sich beispielsweise auf unterschiedliche Funktionen der den Handlungsablauf 24 implementierenden Vorrichtung beziehen. So kann beispielsweise einer der Regelbäume eine Sensorauswertung beschreiben und ein anderer Regelbaum eine Aktuatoransteuerung oder eine Datenanzeige oder Datenübermittlung.

Die Regelbäume 44_1 und 44_2 umfassen zumindest einen Startknoten 46_1 bzw. 46_2 und jeweils zumindest einen Endknoten (NULL) 48_1 und 48_2 bzw. 48_3. Zwischen den Startknoten 46 und den Endknoten 48 sind Ablaufknoten oder Ausführknoten 52 angeordnet, wobei jeder Regelbaum 44_1 und 44_2 zumindest einen Ablaufknoten 52 umfasst. Zumindest jeder der Ausführungsknoten 52, optional der Startknoten 46 und ferner optional der Endknoten 48 kann einen der ausführbaren Ablaufteile 18 implementieren, so dass die dargestellten Knoten 52, 46 und möglicherweise 48 als Repräsentationen der Ablaufteile angesehen werden können. Jeder Knoten kann einen der ausführbaren Ablaufteile implementieren oder repräsentieren, die in dem Handlungsablauf enthalten sind. Alternativ oder zusätzlich kann ein Knoten auch eine Kombination mehrerer ausführbarer Ablaufteil repräsentieren.

Jeder Ausführknoten 52 ist mit zumindest einem vorangehenden Knoten durch eine entsprechende Verknüpfung 52 verknüpft und ist mit zumindest einem nachfolgenden Knoten über eine entsprechende Verknüpfung 54 verknüpft. So ist beispielsweise der Ausführungsknoten 52_1 des Regelbaums 44_1 mit dem Startknoten 46_1 über die Verknüpfung 54_1 verknüpft, was andeutet, dass ein Ausgang des Knotens auf einen nachfolgenden Knoten verweist, etwa unter Verwendung eines Pointers/Zeigers. Dieser nachfolgende Knoten kann seinerseits auf einen Speicherbereich zugreifen, in welchem ein Ergebnis des vorangehenden Knotens hinterlegt werden kann. Unterschiedliche Ausgängen können Werte für unterschiedliche Pointer aufweisen, so dass bspw. in Abhängigkeit davon, welcher Ausgang des Knotens erreicht wird (bspw. wahr oder falsch, größer oder kleiner, etc.)

unterschiedliche Pointer wirksam werden, d. h., unterschiedliche nachfolgende Knoten aufgerufen werden. Ein Knoten weist bspw. zwei Ausgänge auf, wobei jeder Ausgang über eine entsprechende Verknüpfung 54_2 bzw. 54_3 mit einem Ausführungsknoten 52_2 bzw. 52_3 verknüpft sein kann. Beispielsweise können die Ausgänge Ergebnisse einer Oder-Verknüpfung sein, so dass basierend auf dem jeweiligen Ergebnis der Oder-Verknüpfung eine Eingangsgröße an den Ausführungsknoten 52_2 oder 52_3 ausgegeben wird. Anders als Ausführungsknoten 52 weisen Startknoten 46 beispielsweise keine Verknüpfung zu einem vorangehenden Knoten auf, während Endknoten 48 ohne Verknüpfung zu einem nachfolgenden Knoten ausgeführt sein können. Bevorzugt weisen Ausführungsknoten 52 höchstens zwei Eingänge und höchstens zwei Ausgänge auf.

Jedem Ausführungsknoten 52, optional jedem Startknoten 46 und optional jedem Endknoten 48 kann dabei innerhalb des Handlungsablaufs 24 eine vordefinierte Kennung, etwa eine Knotennummer, zugeordnet sein, über die der durch den Knoten 52, 46 bzw. 48 ausführbare Ablaufteil eindeutig adressierbar ist. Der Prozessor der Vorrichtung kann ausgebildet sein, um jeden ausführbaren Ablaufteil der Vielzahl ausführbarer Ablaufteile über diese vordefinierte Knotennummer zu adressieren. Die Knotennummer kann beispielsweise einen Speicherbereich angeben, in dem die zu implementierenden Instruktionen und/oder die zu verarbeitenden Größen hinterlegt sind.

Obwohl die Regelbäume 44_1 und 44_2 so dargestellt sind, dass ein Vorgängerknoten stets einen Nachfolgerknoten referenziert, sind die vorliegenden Ausführungsbeispiele nicht hierauf beschränkt. Vielmehr können Knoten als Nachfolger auch einen direkten oder indirekten Vorgängerknoten eintragen oder mittels einer Verknüpfung referenzieren. Beispielsweise könnte der Ausführungsknoten 52_2 den Knoten 46_1 oder den Ausführungsknoten 52_1 referenzieren, das bedeutet, ein Ausgang des Knotens könnte auf einen Eingang des Knotens weisen oder auf den Knoten selbst. Über solche Konstrukte können sich beispielsweise Schleifen und Zähler (counter) realisieren lassen. Sobald ein Knoten als Ausgang einen Nullpointer aufweist, wie es beispielsweise für die Ausführungsknoten 52_2, 52_3 und 52_5 dargestellt ist, kann die Auswertung des Graphen durch einen Rücksprung aus der Funktion beendet werden. Beispielsweise kann ausgehend von den Knoten 48_1 oder 48_2 zum nächsten Regelbaum 44_2 gesprungen werden.

Der Prozessor kann konfiguriert sein, um die Vielzahl ausführbarer Ablaufteile des Handlungsablaufs von dem Startpunkt des Handlungsablaufs bis zu einem Ende des Handlungsablaufs auszuführen. Hierfür kann das Konfigurationssignal eine Information aufwei-

sen, die den zumindest einen Startknoten des Handlungsablaufs definiert. Der Prozessor kann das Konfigurationssignal auf diese Größen auswerten. Der Startpunkt kann dabei der Startknoten 46_1 oder 46_2 sein, kann aber alternativ auch ein beliebiger anderer Knoten eines Regelbaums 44_1 oder 44_2 bzw. des Handlungsablaufs 24 sein.

In anderen Worten kann der Handlungsablauf durch einen oder mehrere Einzelgraphen oder Regelbäume implementiert sein. Ein einzelner Graph kann dabei einen Startknoten (First-Layer), von dem aus die Auswertung des Regelbaums beginnt. Ein einzelner Knoten weist dabei maximal zwei Ausgänge auf, die in Form von Pointern implementiert werden können, welche auf den nächsten auszuführenden Knoten zeigen. Sobald ein gewählter Ausgang auf 0, etwa ein Nullpointer, zeigt, kann die Auswertung des Graphen beendet sein. Das Programm, der Handlungsablauf, kann dabei aus mehr als einem Graphen bestehen. Jeder Graph kann einen eigenen, dem Graphen zugeordneten, Startknoten aufweisen, wobei die Graphen komplett separat aber auch miteinander verbunden sein können, etwa einen gemeinsamen Teilgraphen aufweisen können. Dies kann alternativ auch so verstanden werden, dass ein Regelbaum mehr als einen Startknoten aufweisen kann. Die Startknoten aller Graphen oder Regelbäume können beispielsweise in einer Art Main-Funktion des als konstant implementierten oder unveränderlichen Programmcodes nacheinander aufgerufen werden.

In anderen Worten kann der Handlungsablauf (Graph) oder ein Teil hiervon, der Regelbaum, aus unterschiedlichen Knoten bestehen, die verschiedene Aufgaben erfüllen und/oder Funktionsweisen repräsentieren. Der Graph muss dabei nicht zusammenhängend sein, vielmehr bilden alle miteinander verknüpften Knoten einen Regelbaum, der abgearbeitet wird. Der vollständige Handlungsablauf kann dabei aus mehreren unabhängig voneinander arbeitenden Regelbäumen bestehen bzw. hiervon repräsentiert werden. Im von der Vorrichtung ausgeführten Programmcode können periodisch, durch gegebenenfalls unterschiedliche Timer getriggert, die sogenannten Startknoten oder First-Layer-Knoten ausgewertet werden. Welche Knoten die Startknoten sind, kann dem Gerät durch ein entsprechendes Interface, etwa die Eingangsschnittstelle 12, mitgeteilt werden. Der weitere Ablauf des Programms kann ereignisbasiert von den Knoten selber bestimmt werden, da jeder einzelne Knoten eindeutig auf den als nächsten auszuwertenden Knoten zeigt und somit den Ablauf vorgibt. Die Auswertung eines Regelbaums terminiert, wenn der Zeiger des nächsten auszuwertenden Knotens auf eine Null-ID gesetzt wird, einen Endknoten 48.

Fig. 5 zeigt ein schematisches Blockschaltbild eines Teils eines Regelbaums 44 gemäß einem Ausführungsbeispiel, wobei beispielhaft zwei ausführbare Ablaufteile 18_1 (a) und 18_2 (b) durch Ablaufknoten 52_1 bzw. 52_2 repräsentiert werden. Der Ausführungsknoten 52_1 weist zwei Eingänge 56_1 und 56_2 auf, über die mittels entsprechender Verknüpfungen 54_1 bzw. 54_2 Eingangsgrößen oder Eingangsinformationen von vorangehenden Knoten erhalten werden können. Ferner weist der Ausführungsknoten 52_1 zwei Ausgänge 58_1 und 58_2 auf, die über entsprechende Verknüpfungen 54_3 und 54_4 mit nachfolgenden Knoten verbunden sein können. So ist beispielsweise der Ausgang 58_2 mittels der Verknüpfung 54_4 mit einem Eingang 56_3 des Ausführungsknotens 52_2 verbunden. Der Ablaufknoten 52_2 kann darüber hinaus einen weiteren Eingang 56_4 aufweisen, der über eine Verknüpfung 54_5 mit einem anderen Knoten verknüpft ist. Der Ablaufknoten 52_2 kann einen oder zwei Ausgangsknoten 58_3 und/oder 58_4 aufweisen, um entsprechend Ausgangsgrößen über Verknüpfungen 54_6 bzw. 54_7 bereitzustellen.

Die Ausgänge 58_1 bis 58_4 können jeweils als Ausgangspointer der jeweiligen Knoten verstanden werden und bspw. als Pointer zu den Index-Nummern nachfolgender Knoten implementiert werden, welche bspw. im, in Zusammenhang mit Fig. 9a erläuterten „next_t/f"-Feld des Knoten gespeichert sind. Die Pointer können den nächsten Auszuführenden Knoten angeben. In den hierein erläuterten Ausführungsbeispielen können deshalb sämtliche Verbindungen, welche den Ausgang eines Knotens repräsentieren als Pointer verstanden werden.

Die Ausführungsknoten 52_1 und 52_2 können voneinander verschiedene aber auch gleiche oder identische Funktionen implementieren, etwa indem im vorliegenden Fall gilt, dass a = b, das bedeutet, die Ablaufteile 18_1 und 18_2 sind von gleicher Art oder dem gleichen Typ zugehörig. Dem Ablaufknoten 52_1 kann eine Knotennummer 62_1 eindeutig zugeordnet sein. Ebenso kann eine Knotennummer 62_2 dem Ablaufknoten 52_2 eindeutig zugeordnet sein, so dass auch bei gleicher oder identischer Funktion der Ablaufknoten 52_1 und 52_2 beide unterscheidbar und separat adressierbar sind.

Es können mehrere Arten von Knoten implementiert sein, die Obergruppen von Knoten definieren. Von jeder Art kann es unterschiedliche Ausführungen mit spezifischen Funktionen geben, die anhand der Fig. 6a bis 6d beispielhaft und nicht erschöpfend beschrieben sind.

Fig. 6a zeigt ein schematisches Blockschaltbild einer Verschaltung eines Ablaufknotens 52 in einem Regelbaum gemäß einem Ausführungsbeispiel. Eingänge 56_1 und 56_2 können mit unterschiedlichen Speicherbereichen 64_1 und 64_2 über Verknüpfungen 54_1 und 54_2 verbunden sein, wobei die Speicherbereiche 64_1 und 64_2 beliebige Bereiche in den Datenspeichern 16, 34 oder 36 sein können, die entsprechende Eingangsgrößen für den Knoten 52 bereitstellen. Knoten eines Graphen können eine Funktion implementieren, die eine Verarbeitung mindestens einer Eingangsgröße des entsprechenden ausführbaren Ablaufteils definiert. Der Ablaufknoten 52 der Fig. 6a ist dabei so ausgeführt, dass die Funktion eine Vergleichsoperation ist, das bedeutet, ein Operator des Knotens 52 kann in den Speicherbereichen 64_1 und 64_2 hinterlegte Werte vergleichen. Der Vergleichsoperator kann beispielsweise einen Größer-Vergleich (x>y), einen Kleiner-Vergleich (x<y), einen Gleich-Vergleich (x=y bzw. x==y), einen Ungleich-Vergleich (x<>y bzw. x!=y), weitere Vergleiche oder Kombinationen hieraus umfassen.

An den Ausgängen 58_1 und 58_2 können Zeiger (Pointer) ausgegeben werden, die mittels der entsprechenden Verknüpfungen 54_3 bzw. 54_4 auf einen nachfolgenden Knoten weisen, für den Fall, dass das Ergebnis des Operators erfüllt ist (true) oder nicht erfüllt ist (false).

In anderen Worten zeigt Fig. 6a einen Vergleichsknoten. Werte in zwei Speicherbereichen können mit logischen Operatoren (UND, ODER, XODER) und/oder mit Negationen derselben oder mathematischen Operatoren, etwa <, ≤, ==, ..., verglichen werden. Der Kontrollfluss, d. h. der nächste auszuwertende Knoten, wird durch das Ergebnis des Vergleichs beeinflusst.

Fig. 6b zeigt eine schematische Darstellung einer Verknüpfung eines Ablaufknotens 52 in einem Regelbaum, wobei die implementierte Operation eine Arithmetik-Operation ist. Hierzu gehören beispielsweise Addition, Subtraktion, Multiplikation oder Division. Mittels des implementierten Operators können die Werte in den Speicherbereichen 64_1 und 64_2 verknüpft werden, so dass ein Ergebnis mittels der Verknüpfung 54_3 einem nachfolgenden Knoten bereitgestellt werden kann. Beispielsweise kann das Ergebnis in einem weiteren Speicherbereich hinterlegt werden und mittels der Verknüpfung 54_3 ein Zeiger auf diesen Speicherbereich übermittelt werden.

In anderen Worten zeigt Fig. 6b einen Arithmetik-Knoten. Mit Werten in zwei Speicherbereichen können arithmetische oder mathematische Operationen ausgeführt werden, z. B.

+, -, x, /, Modulo, Betrag und dergleichen. Das Ergebnis wird in einem separaten, dem Knoten zugeordneten und von außen adressierbaren Speicherbereich bereitgestellt. Ein einzelner Pointer gibt hier den Kontrollfluss, sprich den nächsten auszuwertenden Knoten, an. Es wird noch einmal darauf hingewiesen, dass jedem Knoten, sofern es erforderlich ist, ein Speicherbereich zugewiesen sein kann, wo das zugeordnete Ergebnis gespeichert werden kann, bspw. das „res"-Feld in der Struktur gemäß Fig. 9a. Genau dieser Speicherbereich kann in einem anderen Knoten als Eingang gesetzt werden, um das Ergebnis weiterzureichen, was bspw. dann erfolgt, wenn die Index-Nummer des Knotens als Input eines anderen Knotens gesetzt wird. Die Eingänge und Ausgänge der Knoten können damit vollständig unabhängig voneinander sein.

Fig. 6c zeigt eine schematische Darstellung einer Verknüpfung des Ablaufknotens 52 in dem Regelbaum, bei dem die implementierte Funktion eine Speicheroperation ist, das bedeutet, der Knoten 52 speichert einen Wert des Speicherbereichs 64_1 und/oder 64_2 in einem anderen Speicherbereich und übermittelt mittels der Verknüpfung 54_3 an dem Ausgang 58 einen Zeiger, der auf den Speicherbereich verweist, in dem der Wert gespeichert ist. Eine Eingangsgröße kann in den Speicherbereich der anderen Eingangsgröße oder in den Ausgangsknoten geschrieben werden.

In anderen Worten zeigt Fig. 6c einen Speicherknoten. Der Wert aus dem Speicherbereich 1 wird in einem dedizierten, dem Knoten zugeordneten und von außen adressierbaren Speicherbereich gespeichert. Alternativ kann der Wert aus Speicherbereich 1 an die Adresse von Speicherbereich 2 geschrieben werden, ohne Rücksichtnahme auf den darin enthaltenen Wert. Beide Speicheroperationen können auch kombinatorisch ausgeführt werden. Ein Speichern in dem Speicherbereich 2 kann dafür genutzt werden, um Vergleiche mit alten Werten durchzuführen. Ein einzelner Pointer kann den Kontrollfluss angeben, sprich den nächsten auszuwertenden Knoten.

Fig. 6d zeigt eine schematische Darstellung des Ablaufknotens 52 gemäß einem Ausführungsbeispiel, bei dem mittels Verknüpfungen 54_1 und 54_2 erhaltene Informationseingänge, etwa direkt oder durch Zugriff auf Speicherbereiche 64, in einer auszuführenden Aktion münden, beispielsweise einer Ansteuerung der Peripherie, die über den Ausgang 58 ausgegeben wird.

In anderen Worten zeigt Fig. 6d einen Aktionsknoten. Ein Aktionsknoten kann sich von den anderen Obergruppen an Knoten dadurch unterscheiden, dass hier keine Auswertung

von Operationen auf Speicherbereichen stattfindet. Dieser Knoten kann dazu genutzt werden, Funktionen im Programmcode aus einer Liste von möglichen wählbaren Funktionen aufzurufen, wobei die Auswahl an Funktionen, die das Gerät aufweist, hardwarespezifisch oder gerätespezifisch sein kann. Über das Interface kann sichergestellt werden, dass die wählbaren Aktionsknoten zu den Device-Funktionen passen. Die im Programmcode aufgerufenen Funktionen sind sogenannte normale implementierte Funktionen und können demnach für alle möglichen Funktionen genutzt werden, etwa das Steuern von Aktuatoren, das Setzen von Variablen und Speicher, eine Kommunikation nach außen und dergleichen. Ein einzelner Pointer kann den Kontrollfluss vorgeben, sprich den nächsten auszuwertenden Knoten.

Fig. 7 zeigt eine beispielhafte Darstellung von Computer-implementierten Instruktionen, das bedeutet Code, zur beispielhaften Darstellung unterschiedlicher durch Knoten implementierter Funktionen. So sind beispielhaft 15 unterschiedliche Arithmetik-Funktionen unter Verwendung zweier Eingangsgrößen a, b definiert, die durch eine Funktions-Identifikation 66_1 bis 66_{15} voneinander unterscheidbar sind. Zu den Arithmetik-Funktionen gehört beispielhaft die Addition, die Subtraktion, die Multiplikation, die Division, die Absolut-Summe der Einzelbeträge von a, b, die Halb-Absolut-Summe unter Verwendung des Betragswerts des Operators a oder b, die Absolut-Differenz als Differenz der Betragswerte, die Halb-Absolut-Differenz unter Verwendung lediglich eines Betragswerts, die Ziffer 1-Norm-Summe als Betrag der Addition, die Ziffer 1-Norm-Differenz als Betrag der Differenz, die Absolut-Multiplikation, das bedeutet der Betrag der Multiplikation, die Halb-Absolut-Multiplikation unter Verwendung lediglich eines Betragswerts der Operatoren a oder b, die Absolut-Division, d. h. der Betrag der Division sowie der Modulo-Operator mod.

Es wird deutlich, dass jeder Knoten, unabhängig davon, welche Funktion implementiert wird, eine Handlungsvorschrift beschreibt, wie mit Eingangsgrößen und/oder Ausgangsgrößen umzugehen ist.

Darüber hinaus werden beispielhaft 12 Vergleichsfunktionen mit den Funktions-IDs 16 bis 27 definiert, die einen Gleich-Vergleich, einen Nicht-Gleich-Vergleich (Ungleich), einen Kleiner-Vergleich, einen Kleiner-Gleich-Vergleich, einen Betrags-Gleich-Vergleich, einen Halb-Betrags-Gleich-Vergleich, einen Nicht-Betrags-Gleich-Vergleich, einen Betrag-Nicht-Gleich-Vergleich, einen Betrag-Kleiner-Betrag-Vergleich, einen Betrag-Kleiner-Vergleich,

einen Betrag-Kleiner-Gleich-Betrag-Vergleich und/oder einen Betrag-Kleiner-Gleich-Vergleich umfassen können.

Darüber hinaus können beispielhaft 9 Logik-Funktionen, die auch den Vergleichs-Operationen zugerechnet werden können, definiert werden. Die Funktions-IDs 66_{28} bis 66_{36} (vorher 66_{16} bis 66_{27}) beschreiben dabei Funktionen der Verknüpfung als logisches Und, logisches Oder, logisches Exklusiv-Oder (XOR), einseitig verneintes Und, einseitig verneintes Oder, einseitig verneintes Exklusiv-Oder, beidseitig verneintes Und, beidseitig verneintes Oder sowie beidseitig verneintes Exklusiv-Oder.

Alternativ oder zusätzlich können weitere Funktionen definiert werden. Beispielhaft können, wie es für die Funktions-IDs 66_{37} bis 66_{39} dargestellt ist, eine Negation (!a), das Setzen auf den Wert wahr (true) durch a&&1 bzw. Setzen auf den Wert falsch (false) durch a&&0 ausgeführt werden. Derartige Funktionen oder Logik-Operationen können auch für den Operator b ausgeführt werden. Die Funktions-IDs 66_{37}, 66_{38} und 66_{39} können beispielhaft Funktionen beschreiben, die unter Verwendung lediglich einer Eingangsgröße erfolgen und somit Knoten zugeordnet sind, die lediglich einen Eingang aufweisen. Ebenso können die Arithmetik-Funktionen mit den IDs 66_1 bis 66_{15} lediglich einen Ausgang aufweisen.

Eine Art oder Funktion des ausführbaren Ablaufteils der Vielzahl ausführbarer Ablaufteile kann durch fünf Parameter eindeutig definiert werden. Diese Parameter können in dem Konfigurationssignal enthalten sein. Vorteilhaft stellt sich die Verwendung von Integer-Zahlen dar, da diese einen geringen Speicherbereich aufweisen und gleichzeitig einen ausreichend hohen Wertebereich aufweisen. Gemäß alternativen Ausführungsbeispiele kann zumindest einer der Parameter anders als eine Integer-Zahl definiert werden. Die fünf Parameter können ausreichen, um die Art des ausführbaren Ablaufteils eindeutig zu definieren. Einer der Parameter, etwa die Funktions-ID, kann die Funktion des ausführbaren Ablaufteils definieren. Ein zweiter und ein dritter Parameter kann einen ersten bzw. zweiten Speicherbereich, der mit dem ersten bzw. zweiten Eingang des ausführbaren Ablaufteils assoziiert ist, definieren. Beispielsweise kann in dem Konfigurationssignal angegeben werden, welche der Speicherbereiche 64_1 und/oder 64_2 mit dem jeweiligen durch den Ablaufknoten 52 implementierten Ablaufteil durch die entsprechenden Verknüpfungen 54_1 bzw. 54_2 verknüpft sind, etwa durch Ausgestaltung der Verknüpfungen 54 als Zeiger (Pointer). Ein vierter und fünfter Parameter können entsprechende Speicherbereiche definieren, die mit den Ausgängen 58_1 und/oder 58_2 bzw. 58 assoziiert sind. Der Generator

22 kann ausgebildet sein, um die Integer-Zahlen, die die fünf bzw. sechs Parameter des Knotens wiedergeben, jeweils mit einem Speicherbereich des Datenspeichers eindeutig zu assoziieren. Dies ermöglicht die Sicherstellung, dass jeder Knoten auf einen ihm zugewiesenen Datenspeicherbereich zugreift, um Eingangsgrößen zu erhalten oder Ausgangsgrößen zu hinterlegen

Unbenötigte Parameter, etwa wenn nur einer der Ausgänge benötigt wird, können ausgelassen werden oder zu 0 gesetzt werden. Der Generator kann ausgebildet sein, um das Konfigurationssignal auf die entsprechenden Parameter zu überprüfen. Die entsprechenden Parameter, insbesondere die Integer-Zahlen, können jeweils mit einem Speicherbereich des Datenspeichers 16 eindeutig assoziiert werden, so dass es hinreichend ist, lediglich den Speicherbereich anzugeben, der die weiteren zu implementierenden Informationen angibt. Insofern kann die in dem Konfigurationssignal enthaltene Information 26 eine Sequenz von Integer-Zahlen aufweisen, was eine geringe Datenmenge darstellt, die zu übertragen ist.

In anderen Worten zeigt Fig. 7 eine beispielhafte Zuweisung von Knotentypen oder Funktionen an Funktions-IDs 66.

Fig. 8 zeigt eine beispielhafte Darstellung von Code, der Instruktionen wiedergibt, die den Generator 22 der Vorrichtung 10 oder die Steuereinrichtung 38 der Vorrichtung 30 anweisen, das Konfigurationssignal 14 auf die Funktions-ID 66, die im Zusammenhang mit der Fig. 7 beschrieben ist, auszuwerten. Der Generator kann ausgebildet sein, um die in dem Konfigurationssignal enthaltenen Parameter auszuwerten und abhängig von deren Wert den Graphen zu konfigurieren. So kann bspw. der Knoten mit ID i darauf ausgewertet werden als welcher Knotentyp er konfiguriert wird, hier gibt conf_typ[i] den Typen an, siehe Zeile 6. Weitere Eigenschaften des Knotens, etwa Eingänge und/oder Ausgänge und/oder Speicherbereiche zum Speichern von Ergebnissen können ebenfalls festgelegt werden, siehe Zeile 7ff. Fig. 9a zeigt eine beispielhafte Darstellung von Code zur Definition einer Knotenstruktur und somit einer Struktur, der die ausführbaren Ablaufteile 18 folgen. Mit „int16_t" bzw. „uint16_t" können vorzeichenbehaftete bzw. ohne Vorzeichen versehene 16-Bit-wertige Integer-Zahlen bezeichnet werden, die dazu verwendet werden, ein Ergebnis (res), Eingangsgrößen (inp_1, inp_2) sowie Verweise oder Pointer/Zeiger auf nachfolgende Knoten für den Fall eines positiven (true) oder negativen (false) Ergebnisses (next_t, next_f) zu definieren, worauf bereits im Zusammenhang mit Fig. 5 eingegangen wurde. Alternativ oder zusätzlich kann unter „typ" angegeben werden, um welche Art

von Knoten es sich bei dem vorliegenden Knoten handelt. Es wird deutlich, dass die jeweiligen Speicherbedarfe an die Anzahl unterschiedlicher Werte anpassbar ist, so kann beispielsweise der Parameter typ mit 8-Bit-Integer-Zahlen hinreichend beschreibbar sein, während anderenfalls vorzeichenbehaftete bzw. vorzeichenunbehaftete, d. h., vorzeichenneutrale 16-Bit-Integer-Zahlen verwendet werden können. Prinzipiell kann eine Zuweisung eines Speicherbereichs für den entsprechenden Parameter an den tatsächlichen Bedarf angepasst sein. Obwohl in Fig. 9a insgesamt sechs Eigenschaften definiert werden, kann ein Knoten eindeutig durch lediglich fünf dieser Eigenschaften definiert oder festgelegt werden. Der sechste, für die eindeutige Definition nicht erforderliche Parameter in der Knotenstruktur kann das das Ergebnis der Operation durch den Knoten beschreiben und ist beispielhaft durch das „res"-Feld angegeben. Das Ergebnis muss bei der Definition des Graphen nicht von außen vorgegeben werden, es kann sich bspw. bei der Ausführung des Knotens ergeben. Von daher werden nur fünf Parameter zur Definition benötigt, auch wenn in der Knoten Struktur sechs Variablen vorhanden sind.

In anderen Worten können die Knoten des Graphen (Regelbaums) als Struktur (struct) im Programm repräsentiert werden. Die Struktur kann die Variable „typ" oder gleichbedeutend enthalten, welche die Art des Knotens festlegt. Die Variable „res" speichert beispielsweise das Ergebnis einer Rechenoperation oder eines Vergleichs und die Variablen „input_1/input_2" sowie „next_t/next_f" fungieren im Grunde genommen als Pointer für die Eingangsvariablen und den nächsten auszuführenden Knoten. In der Implementierung kann hier aus Speichereffizienz anstelle eines Pointers ein Integer-Wert verwendet werden. Dies ist möglich, weil die Anzahl der maximal verwendbaren Knoten und Variablen begrenzt ist und a priori festgelegt wird, d. h. vordefiniert ist. Der Nullpointer kann hierbei durch die größte darstellbare Zahl des Datensystems repräsentiert sein.

Der Datentyp „Knoten" kann so definiert werden, dass er sich zusammensetzt aus

- uint typ; eine Zahl, die die Art des Knotens festlegt;
- int res; ein Ergebnis der Berechnung, falls vorhanden oder erzeugt;
- uint inp1; eine Index-Nummer für einen ersten Eingang;
- uint inp2; eine Index-Nummer für den zweiten Eingang;
- uint out_t; eine Index-Nummer für einen ersten Ausgang des Knotens; und
- uint out_f; eine Index-Nummer für den zweiten Ausgang.

Arithmetik, Aktions- und Speicherknoten können lediglich einen der beiden Ausgänge verwenden, etwa out_t, so dass beispielsweise out_f auf NULL-ID gesetzt werden kann.

Da der Speicherbereich der Vorrichtung begrenzt ist, kann eine vordefinierte Anzahl von Knoten, Variablen und Aktionen existieren. Alle können in Form von Arrays organisiert vorliegen und hintereinander im Speicher angeordnet sein. Dies ermöglicht eine eindeutige Adressierung über eine Integer-Zahl, die als Index-Nummer bezeichnet werden kann. Die Index-Nummern lassen sich verwenden wie allgemeine Pointer und können allgemein als solche gesehen werden, diese Art der Implementierung spart lediglich Ressourcen am Adressieren von Speicherbereichen. Analog zu einem Nullpointer kann eine Null-ID existieren.

Bezugnehmend auf Fig. 9b kann beispielsweise die maximale Anzahl von Knoten (MAX_KNOTS), die maximale Anzahl von Variablen (MAX_VARS), die Anzahl von Sensoren (N_SENSOREN), die Anzahl von Aktionen (N_AKTIONEN), die Anzahl von Knoten einer ersten Schicht oder Layer, also bspw. Einsprungsknoten in einen Graphen oder Regelbaum, (N_First_Layer), NULL_ID, d. h. der Nullpointer, und beliebige andere Variablen definiert werden, etwa Unterscheidungen, ob es sich um einen Arithmetikknoten zum Ausführen einer Arithmetikoperation handelt (IS_ARI) und/oder ob es sich um einen Vergleichsknoten zum Ausführen einer Vergleichsoperation (IS_VGL) handelt.

Bei der beispielhaften Verwendung eines Micro-Controller als Prozessor 31 oder Steuereinrichtung 38 in einer 32-Bit-Konfiguration kann ein entsprechender Pointer ebenfalls 32 Bit im Speicher benötigen. Durch die Implementierung als Integer-Wert mit beispielsweise 16 Bit können so in jedem Knoten 4*16 Bit, d. h. 8 Byte eingespart werden. Dies ist insbesondere dann von Vorteil, wenn insgesamt weniger als 2^{16} Knoten, Variablen und Aktionen verwendet werden. Der Vorteil ist jedoch auch dann zu erzielen, wenn eine Datenreduktion anderer Kategorie vorgenommen wird, beispielsweise bei Verwendung eines 64-Bit-Controllers, für den 32-Bit-Variablen verwendet werden. Bei dem Durchnummerieren werden die ersten Eingänge beispielsweise als Knoten dereferenziert (bzw. der Speicherbereich des „res"-Feldes eines Knotens), alles darüber oder danach sind dann beispielsweise die Sensoren, dann Aktionen, alles darüber wird als Variablen behandelt. Dies kann sich aus der in Fig. 9b dargestellten Reihenfolge ergeben, in der nacheinander die Knoten, Variablen (VARS), Sensoren und Aktionen.

Die dargestellte Reihenfolge ist dabei beliebig oder willkürlich. Es ist jedoch erforderlich, dass sie einmal festgelegt wird und anschließend identisch bleibt bzw. eingehalten wird. Der dadurch erhaltene Informationsgewinn kann somit innerhalb des Konfigurationssignals 14 eingespart werden, da die Struktur beidseitig bekannt ist.

Eine derartige Konfiguration ist beispielsweise in der Fig. 9c dargestellt, die einen beispielhaften Code für eine derartige Strukturierung angibt. So kann beispielsweise definiert werden, dass alle Knotentypen unterhalb des zuvor festgelegten Wertes T_SEN Knoten sind. Werte ≥ T_SEN Knoten sind. Werte ≥ T_SEN und < T_AKT sind Sensoren. Von diesem Wert ausgehend sind alle Werte von ≥ T_AKT und < T_THR Aktionen. Alles Darüber hinaus gehende, das bedeutet ≥ T_THR, sind Variablen. T_MAX gibt dabei die maximal mögliche Zahl im System an.

In anderen Worten kann die „typ"-Variable im Knoten-struct (der Definition der Struktur) festlegen, was genau der Knoten zu tun hat, sobald er aufgerufen wird. Hierüber kann auch die Art des Knotens festgelegt werden, etwa als Arithmetik-Knoten, in dem die ID 66_1 bis 66_{15} verwendet wird, die im Zusammenhang mit der Fig. 7 erläutert ist. Die Art der Knoten kann sich in verschiedene Gruppen aufteilen lassen. Zum einen können Arithmetik-Knoten implementiert sein, die für arithmetische Rechenoperationen wie z. B. „+" oder „*" eingerichtet ist. Hierbei werden die Werte der beiden Input-Speicherbereiche, der Eingänge, verwendet und das Ergebnis der Operation im „res"-Feld des Knoten-structs gespeichert. Alternativ oder zusätzlich kann es Vergleichs-Knoten geben, die für Vergleiche wie z. B. „<", „==" und/oder logische Operationen wie z. B. „UND" eingerichtet sind. Auch hier können die Werte der beiden Input-Speicherbereiche verwendet und das Ergebnis der Operation im „res"-Feld gespeichert werden. Dies lässt jedoch auch Ausnahmen zu, beispielsweise eine Negation oder einen Vergleich auf „true" bzw. „false" (wahr/falsch), da diese nur eine Input-Variable verwenden.

Alternativ oder zusätzlich können einer oder mehrere Speicherknoten definiert werden, die den Wert an der Adresse eines der Eingänge verwenden und diesen beispielsweise an die Adresse des anderen Eingangs oder in ein extra hierfür vorgesehenes „res"-Feld schreiben, wie es beispielsweise im Zusammenhang mit der Fig. 6c beschrieben ist. Speicherknoten können beispielsweise benutzt werden, um Sensorwerte (die sich über die Zeit ändern können) zu speichern. Alternativ oder zusätzlich können Aktions-Knoten vorgesehen sein, die dazu verwendet werden können, vorher implementierte Funktionen im Programmcode aufzurufen. Die beiden Input-Variablen können dabei als Übergabepa-

rameter für die aufgerufene Funktion dienen, wobei dies optional ist. Aktions-Knoten können relevant sein, um komplexere Programmstrukturen zu ermöglichen. So kann beispielsweise das Senden eines Werts über einen Aktions-Knoten repräsentiert werden. Die verfügbaren Aktionen sind dabei zum Teil Hardware-abhängig, da eine Vorrichtung zum Senden einer WLAN-Nachricht ein WLAN-Modul erfordert. Die Aktions-Knoten können demzufolge für die verwendete Hardware speziell implementiert oder eingerichtet sein. Sämtliche vom Benutzer gewünschten im Programmcode aufrufbaren Funktionen können mit einem jeweils zugeordneten Aktions-Knoten verknüpft werden und durchnummeriert werden, etwa unter Verwendung einer entsprechenden Kennung, die mit der Funktions-ID vergleichbar ist. Eine Aktion mit einer gewissen Nummer entspricht dann immer derselben Funktion im eigentlichen Programmcode und diese wird aufgerufen, sobald im Graphen ein Aktions-Knoten mit der entsprechenden Kennung, die über das „typ"-Feld repräsentiert werden kann, aufgerufen oder getriggert wird.

Die maximale Anzahl an Knotentypen kann durch den Zahlenbereich der Variable „typ" in der Knotenstruktur begrenzt sein, siehe beispielsweise Fig. 9a. Alle Werte, die nicht für Arithmetik-, Logik- oder Speicherknoten verwendet werden, können mit einer Aktion assoziiert werden. Um zu jedem Zeitpunkt eine Turing-Vollständigkeit zu gewährleisten, können beispielsweise mindestens Knoten für Addition (+), Subtraktion (-), Vergleiche (==) und ein Speicherknoten implementiert sein.

Fig. 10a zeigt einen beispielhaften Code, bei dem in Zeile 2 eine maximale Anzahl von Knoten des Handlungsablaufs definiert wird, was dazu führt, dass die Liste an Knoten eine innerhalb dieser Konfiguration stets gleiche Länge aufweist. In Zeile 5 wird eine Liste von Aktionen für den Knoten definiert. In Zeile 8 wird eine Liste von Knoten in einem ersten Layer (First-Layer) definiert, worauf später eingegangen wird. In Zeile 11 werden unterschiedliche Variablen, etwa Schwellwerte oder Zwischenspeicher, definiert. In Zeile 14 werden Sensorwerte definiert.

Wie bereits erwähnt, ist die Anzahl der maximal möglichen Knoten und Aktionen sowie Sensoren und Variablen (wobei Sensoren effektiv als „read-only" (Nur-Lesen-) Variablen interpretiert werden können), a priori vorgegeben. Deshalb kann vor Beginn der Main-Funktion der Speicherbereich für die entsprechenden Strukturen allokiert, das bedeutet assoziiert werden.

Die gerade erwähnten First-Layer-Knoten oder Startknoten können ein wichtiges Element für die Auswertung der auf der Vorrichtung implementierten Handlungsabläufe sein. Eine Liste an Pointern auf die Knoten-Struktur kann dazu verwendet werden, um die Startknoten festzulegen, wie es in Zeile 8 des Codes in Fig. 10a beschrieben ist. Der Startknoten kann beispielsweise ein Knoten 46_i des jeweiligen Regelbaums 44 sein, wie es im Zusammenhang mit Fig. 4 beschrieben ist.

Fig. 10b zeigt einen beispielhaften Code, der anzeigt, dass eine Knotennummer i bis zu einer maximalen Anzahl N_ACTIVE_FL von Knoten oder Aktionen der Graph vollständig durchlaufen wird und der entsprechende Knoten ausgeführt wird (doKnot).

Der Generator einer erfindungsgemäßen Vorrichtung kann ausgebildet sein, um den Handlungsablauf so zu erstellen, dass dieser zumindest einen Startknoten, an dem der Handlungsablauf durch den Prozessor begonnen wird, umfasst. Für den Fall, dass der Handlungsablauf mehrere Regelbäume umfasst, kann für jeden Regelbaum ein solcher Startknoten vorgesehen sein. Der Prozessor ist ausgebildet, um den zumindest einen Startknoten periodisch auszuwerten, um ein Auswertungsergebnis zu erhalten. Das bedeutet, jeder Regelbaum wird an dem entsprechend zugeordneten Startknoten periodisch begonnen. Der Prozessor kann dabei ausgebildet sein, um die Vielzahl ausführbarer Ablaufteile des Handlungsablaufs von dem Startpunkt, d. h. dem Startknoten des Handlungsablaufs, bis zu einem Ende des Handlungsablaufs auszuführen, wie es beispielsweise in Fig. 4 dargestellt ist.

Auch Fig. 10c, die einen beispielhaften Code zum Ausführen durch den Prozessor darstellt, zeigt dies. Ein empfangener Wert k wird ausgewertet, wobei der Wert k beispielsweise einen Pointer auf den Knoten, welcher gerade ausgeführt werden soll, repräsentiert. Mit dem „->“ Operator können die Werte aus dem zugehörigen Knoten gelesen werden. Der Ausdruck „k->typ“ kann bspw. den Typen des Knoten zurückgeben, d. h., in den Zeilen 3, 7 und 12 erfolgt bspw. eine Auswertung des Typs des Knotens. Der Ausdruck „k->inp_1“ in Zeile 14 steht beispielhaft dafür, die Index Nummer des ersten Eingangs zu enthalten. Der Wert des Eingangs kann durch die Funktion „getInput(...)“ in den Zeilen 14 und 15 bereitgestellt werden, welche die Index-Nummern der Eingänge in die zugehörigen Werte überführt. Analog kann die Funktion „getPointer(...)“ in Zeile 10 funktionieren; diese nimmt die Index-Nummer des Ausgangs („k->next_t“ oder „k->next_f“) und liefert den zugehörigen Pointer auf den nächsten Knoten.

Wird durch die „typ" Variable in „k" festgestellt, dass es sich um eine Aktion handelt, so wird die zugehörige Aktion nicht über den Wert „k" abgerufen, sondern über den Wert „k->typ", da jede Aktion eindeutig über die Typnummern festgelegt ist. Eine entsprechende Ausgestaltung kann bspw. dadurch erhalten werden, dass in Übereinstimmung mit den Ausführungen im Zusammenhang mit Fig. 7 und Fig. 10d Knoten dergestalt definiert werden, dass Werte von k mit 64 und größer durchzuführende Aktionen definieren. Die in der Zeile 7 durchgeführte Überprüfung unterscheidet also, ob es sich um eine Aktion handelt, wenn der Wert k zumindest 64 beträgt, wobei beliebige andere Ausgestaltungen möglich sind. Der Rückgabewert der Funktion (welche mit der Aktion verknüpft ist), wird hier beispielhaft nicht betrachtet. Bei dem Pointer, welcher in der angegebenen Zeile 10 zurückgegeben wird, handelt es sich um den Pointer auf den nächsten Knoten, welcher durch den oben beschriebenen Mechanismus bereitgestellt wird.

Handelt es sich im angegebenen Beispiel um keinen Aktionsknoten, kann eine entsprechende Verarbeitung erfolgen, wie es in den Zeilen 12 ff. beschrieben ist. In Zeile 17 ist dargestellt, dass ein als Speicherknoten vorgesehener Knoten als Sonderfall behandelt werden kann, bei dem der Wert 1 in das „res"-Feld geschrieben wird, siehe Zeile 19, oder in den Speicherbereich des anderen Eingangswerts, siehe Zeilen 21 und 22.

In anderen Worten können die Startknoten in der Main-Funktion des Programmcodes der Reihe nach ausgewertet werden. Diese Routine kann dabei immer identisch sein, unabhängig von den Graphen, die dann die eigentliche Funktionalität des Geräts bereitstellen. Dadurch kann gewährleistet werden, dass der Programmcode im EEPROM, d. h. dem Datenspeicher 16, unverändert bleiben kann, selbst wenn das Gerät umprogrammiert wird. Die Funktion, welche die Knoten auswertet, ist ebenfalls immer identisch und fest im Programmcode vergossen, das bedeutet im Datenspeicher 16 hinterlegt. Sie kann beispielsweise initial von der Main-Funktion für die Startknoten aufgerufen werden und sich dann an den Output-Pointern der einzelnen Knoten entlanghangeln, bis das Ende des Graphen (Nullpointer bzw. Knoten 48) erreicht wird. Dabei kann der Prozessor ausgebildet sein, um einen auszuführenden Knoten danach auszuwerten, um welchen Typen es sich handelt und um dann die entsprechende mit dem Knoten verknüpfte Operation auszuführen, wie es anhand von Fig. 10d beschrieben ist.

Fig. 10d zeigt eine beispielhafte Darstellung von Code, der Instruktionen wiedergibt, die den Prozessor 31 der Vorrichtung 10 oder die Steuereinrichtung 38 der Vorrichtung 30 anweisen, das Ergebnis der Auswertung des Knotentyps (k->typ) aus Fig. 10c weiterzu-

führen. Dort wird bspw. festgestellt, dass es sich um keinen Aktionsknoten handelt, Zeile 12 in Fig. 10c, und mittels „switch (k-> typ) in die konkreten Auswertung gewechselt. Durch die Auswertung (case), welche Funktions-ID 66_1 bis 66_x angegeben ist, wird daraus inhärent angegeben, welche der im Zusammenhang mit Fig. 7 dargestellten Funktionen implementiert wird. Der Ausdruck „k->res" gibt dabei an, dass ein Ergebnis res aufgrund der nachfolgend definierten Funktion erhalten wird.

Bei dem Ausführen des Handlungsablaufs kann die Steuereinrichtung 38 oder der Prozessor 31 den Knoten bzw. die Funktion dahin gehend auswerten, um welchen Typen es sich handelt und dann die entsprechende mit dem Knoten verknüpfte Operation ausführen.

In anderen Worten können Ergebnisse des Knotens bzw. der Funktion im „res"-Feld der Struktur hinterlegt werden. Wird ein Knoten als Eingang von einem anderen Knoten gesetzt, so liest der dahinter liegende Knoten den Speicherbereich des „res"-Feldes von dem Eingangsknoten aus. Dadurch können Ergebnisse durch simple Verknüpfung von Knoten im Graphen weitergereicht werden.

Nachdem die Auswertung eines einzelnen Knotens abgeschlossen ist, kann der nachfolgende Knoten ausgewertet werden. Dabei ist bei einigen Knoten der Nachfolger anhängig vom Ergebnis des „res"-Feldes des Vorgängerknotens, insbesondere bei Vergleichen sowie logischen Operationen, was in Fig. 10e illustriert ist, die einen beispielhaften Code anzeigt, bei dem für den Fall des Knotentyps als arithmetischer Knoten (IS_ARI) der Pointer für den nächsten Knoten abgerufen wird, siehe Zeile 4. Für den Fall eines Vergleichsknotens (IS_VGL) wird der Pointer abhängig von dem erhaltenen Ergebnis true/false erhalten, siehe Zeilen 9 bis 12. Das bedeutet, wird der nächste auszuführende Knoten bestimmt.

Ausführungsbeispiele der vorliegenden Erfindung basieren auf der Erkenntnis, dass der Graph zur Laufzeit der Vorrichtung neu konfiguriert werden kann, um beliebige Funktionen oder Programme zu repräsentieren, das bedeutet, eine Flexibilität der Vorrichtung zu erhalten. Da die Strukturen für die Knoten, Aktionen und Variablen für eine Hardware fest sein können und zur Laufzeit im flüchtigen Speicher liegen, kann es ausreichend sein, diese der Reihe nach richtig zu verknüpfen und zu konfigurieren, um ein neues Programm zu repräsentieren. Derartige Informationen können in dem Konfigurationssignal angegeben werden. Als Initialisierung können beispielsweise alle Knoten auf Typ 0, d. h. nicht

initialisiert, gesetzt werden. Dies kann beispielsweise über ein hierfür geeignetes Event, das bedeutet eine entsprechende Anweisung, etwa von einer zentralen Steuereinrichtung oder einem Server ausgesendet, angesteuert oder getriggert werden. Jeder Knoten eines Regelbaums kann über eine Integer-Zahl adressierbar sein. Jeder Knoten kann darüber hinaus über fünf Integer-Zahlen eindeutig definiert werden. Zwei Werte für Input und Output sowie einer für den Typ. Es können also pro verwendetem Knoten sechs Integer-Werte verwendet werden, nämlich die erwähnten fünf Integer-Zahlen für die Konfiguration des Knotens sowie ein sechster Wert für die Knotennummer. Das Konfigurationssignal kann die sechs Integer-Werte aufweisen. Es ist möglich, jedoch nicht erforderlich, pro Knoten eine Nachricht zu senden. Es ist möglich, sechs hintereinanderliegende Werte in der Nachricht für einen Knoten zu verwenden und eine beliebige Anzahl von Knoten innerhalb eines Konfigurationssignals zu definieren.

In dem Konfigurationssignal oder einem weiteren Konfigurationssignal kann definiert werden, welcher Knoten oder welche Knoten als Startknoten fungieren. Diese Information kann grundsätzlich auch Teil der Konfigurationsnachricht der Knoten sein, beispielsweise an das Ende gesetzt werden, ist aber bevorzugt eindeutig von den anderen Integer-Werten abgetrennt. Als Beispiel kann dienen, dass für die Definition des Graphen mit 256 Knoten lediglich 256*16 Bit (bei Verwendung von 16-Bit-Integern für die Durchnummerierung der Knoten und Variablen) = 512 Byte an Informationen übertragen werden und zusätzlich optional ein paar weitere Byte je nach Menge der verwendeten Startknoten. Nach erfolgreicher Konfiguration durch den Generator kann der Graph sofort verwendet werden.

Fig. 11 zeigt ein schematisches Blockschaltbild einer Vorrichtung 110 gemäß einem Ausführungsbeispiel, die Eingänge (Schnittstelle 28_1), „Ausgänge“ (Schnittstelle 28_5) und eine innere Logik, die den Handlungsablauf 24 umfasst. Als Eingänge können z. B. Sensoren, Schalter/Taster oder andere Bedienelemente dienen. Die innere Logik kann das abzulaufende Programm, das der Benutzer vorher durch das Interface festgelegt hat, implementieren. Dieses kann beispielsweise im Regelbetrieb in einer Dauerschleife ausgeführt werden. Ausgänge können beispielsweise Aktuatoren diverser Art, Schalter, Motoren, eine Kommunikation nach außen, etwa Sendewert XY an Server N, umfassen, kann alternativ oder zusätzlich aber auch das Setzen/Ändern von Timern und Interrupts auf der Hardware des Geräts und/oder das Ändern von Speicherbereichen im flüchtigen Speicher der Hardware umfassen. Durch die innere Logik kann ein Algorithmus oder ein Programm bzw. eine Sequenz maschinenlesbare Instruktionen repräsentiert werden, welche Eingän-

ge und Ausgänge aber auch Variablen, Timer und interne Elemente des Micro-Prozessors verwendet. Durch die erfindungsgemäße Struktur kann ein beliebiges Programm repräsentiert und aufgebaut werden.

Fig. 12 zeigt ein schematisches Blockschaltbild einer Vorrichtung 120 gemäß einem Ausführungsbeispiel, das ähnlich gebildet sein kann wie die Vorrichtung 10 oder die Vorrichtung 30 oder die Vorrichtung 110, wobei an den Ausgängen 28_5 die Peripherie 32 in Form eines Schalters drahtgebunden oder drahtlos angeschlossen ist, der in unterschiedliche Zustände, entweder „an"/"aus" oder dergleichen, steuerbar ist. Über die Eingänge 28_1 können Steuersignal 68_1 und/oder 68_2 von externen Einrichtungen 72_1 und/oder 72_2 erhalten werden, etwa Server oder dergleichen. Die Steuersignale 68_1 und/oder 68_2 können eine Anweisung zum Anschalten oder Ausschalten des Schalters der Peripherie 32 aufweisen. Der Handlungsablauf, der in der Vorrichtung 120 implementiert ist, kann dabei so ausgeführt sein, dass die Peripherie gemäß den Steuersignalen 68_1 und 68_2 angesteuert wird.

In anderen Worten zeigt Fig. 12 ein Beispiel, wie die Logik im Gerät Eingänge und Ausgänge miteinander verknüpft und wozu Ausführungsbeispiele verwendet werden können. Die Vorrichtung 120 ist beispielsweise als reiner Aktuator ausgebildet. Die innere Logik kann so ausgelegt sein, dass im Falle, dass die externe Einrichtung 72_1 instruiert, den Schalter anzuschalten, oder, wenn die externe Einrichtung 72_2 instruiert, den Schalter 32 anzuschalten, die Vorrichtung 120 die Peripherie 32 dann entsprechend ansteuert. Ansonsten wird der Schalter am Ausgang 28_5 auf „aus" geschaltet.

Fig. 13 zeigt ein schematisches Blockschaltbild einer Vorrichtung 130 gemäß einem Ausführungsbeispiel, bei dem die Eingänge 28_1 mit Sensoren 74_1 und 74_2 in beliebiger Anzahl ≥ 1 verbunden sind, um Sensorsignale 76_1 und 76_2 mit entsprechenden Sensorwerten zu empfangen. Die Vorrichtung 130 kann ausgebildet sein, um die Sensorsignale 76_1 und 76_2 auszuwerten und um mittels eines Ausgangssignals 78 die Peripherie anzusteuern, die beispielsweise einen Server umfasst, um die ausgewerteten Daten dort zu speichern.

In anderen Worten zeigt Fig. 13 eine Konfiguration der Vorrichtung 130 als reinen Sensor mit einer Kommunikation. Die innere Logik kann beispielsweise so ausgestaltet sein, dass regelmäßig, etwa alle 60 Minuten ein neuer Wert von dem Sensor 74_1 gespeichert wird. In einem gleichen oder verschiedenen Zeitintervall, etwa alle 120 Minuten, wird ein neuer Wert von dem Sensor 74_2 gespeichert. Die innere Logik kann so ausgestaltet sein, dass,

wenn beispielsweise eine bestimmte Anzahl Messwerte, etwa 100, vorliegen, ein Minimum und ein Maximum der Werte aus Sensor 74_1 und/oder ein Mittelwert der Werte aus Sensor 74_2 berechnet wird. Ein entsprechendes Ergebnis wird an den Server gesendet. Alternativ oder zusätzlich können die alten Werte gelöscht werden und eine neue Messreihe begonnen werden.

Fig. 14 zeigt ein schematisches Blockschaltbild einer Vorrichtung 140, die mit der Peripherie 32 verbunden ist und die als Sensor/Aktor-Logik bzw. als Sender und Empfänger, etwa für eine Smart-Home-Einrichtung (intelligentes Zuhause), konfiguriert ist. Hierfür kann der Sensor 74 beispielsweise eingerichtet sein, um eine Temperatur zu erfassen, ein Server 82 kann eine Steuerung, etwa eines zusätzlichen Schalters oder Ventils bzw. Heizungsventils 84, bewirken, wobei die Vorrichtung 140 ausgebildet ist, um diverse Schalter 86_1 und/oder 86_2 zu steuern, das Heizungsventil 84 zu öffnen oder zu schließen und dabei Informationen von Schaltern 86_1, Sensoren 74 und/oder Servern 82 zu empfangen. Beispielsweise kann der Sensor 74 als Temperatursensor eingerichtet sein, um eine Temperatur zu messen, wobei jede andere Konfiguration ebenfalls implementierbar ist.

In anderen Worten kann die innere Logik z. B. so ausgestaltet sein, dass, wenn der Wert des Schalters 86_1 verändert wird oder die Temperatur veränderlich ist, sich beispielsweise um mehr als 10 Grad geändert hat, dann wird ein neuer Zustand an den Server 82 gesendet und die entsprechenden Werte gespeichert. Wenn beispielsweise der Server 82 die Steuerung 68 auf „an" setzt, so kann beispielsweise die Temperatur geprüft werden und mit einem Soll-Wert verglichen werden, wenn die Temperatur unter dem Soll-Wert ist, dann kann das Heizungsventil 84 geöffnet werden, ansonsten geschlossen. Wenn der Server die Steuerung auf „an" setzt, so kann beispielsweise ein Zustand des Eingangs-Schalters 86_1 an den Ausgangs-Schalter 86_2 weitergegeben werden, sonst kann der Ausgang auf „aus" geschaltet werden.

Die Fig. 12, 13 und 14 beschreiben unterschiedliche Konfigurationen von Vorrichtungen. Es wird darauf hingewiesen, dass diese Konfigurationen von ein und derselben Vorrichtung implementierbar sind, beispielsweise durch Neukonfiguration oder Umkonfiguration oder Re-Konfiguration mittels eines Konfigurationssignals. Ist die Vorrichtung derart implementiert, dass alle Konfigurationen von der Hardware abbildbar sind, kann somit jedes der Beispiele lediglich durch das Konfigurationssignal in die jeweils andere Konfiguration umkonfiguriert werden.

Wenn der Graph, d. h. der Handlungsablauf, geändert werden soll, kann für jeden verwendeten Knoten eine Nachricht mit dem nachfolgend beschriebenen Aufbau gesendet werden und/oder es wird eine kombinatorische Nachricht mit Informationen bezüglich mehrerer oder aller Knoten gesendet, entweder aller Knoten insgesamt oder aller zu ändernden Knoten. Die Länge der Nachricht kann abhängig vom verwendeten Kommunikationsprotokoll ausgestaltet sein. Unter Verwendung von jeweils sechs Zahlenwerten kann ein Knoten neu definiert werden. Es können entweder nur einzelne Knoten geändert/neu definiert werden oder über eine spezielle Lösch-Nachricht dem Gerät mitgeteilt werden, dass es seinen aktuellen Graphen komplett zurücksetzen soll. Die sechs Zahlenwerte, die einen Knoten definieren, können beispielsweise wie folgt definiert angeordnet werden, wobei eine Reihenfolge beliebig ausgestaltet sein kann, solange sie dem Gerät vorbekannt ist, um eine korrekte Decodierung zu ermöglichen:

- Index_Nr.; Index-Nummer des zu ändernden Knotens;
- Typ_Nr.; Neuer Typ des zu ändernden Knotens;
- indx_inp1; Index-Nummer des Speicherbereichs 1 des ersten Eingangs;
- indx_inp2; Index-Nummer des Speicherbereichs 2 des zweiten Eingangs;
- indx_out1; Index-Nummer des ersten Ausgangs, gegebenenfalls True-Ausgangs; und
- indx_out2; Index-Nummer des zweiten Ausgangs, beispielsweise des Falls-Ausgangs.

Da alle Knoten und Variablen eindeutig durch ihre Index-Nummer adressiert werden können, können vier Zahlen ausreichend sein, um die Struktur des Graphen, d. h. die Verknüpfung der Knoten untereinander und mit Variablen, zu ändern, indem die Index-Nummern der Speicherbereiche der Eingänge und Ausgänge verändert werden. Eine weitere Index-Nummer kann angeben, welcher Knoten neu definiert wird und mit einem zusätzlichen, sechsten Integer-Wert kann die Art des Knotens geändert werden.

Bekannt sind verschiedene Möglichkeiten, wie Geräte mit neuen Aufgaben/Programmen von außen bespielt werden können.

Die naivste Methode ist, das komplette Programm über die Kommunikationsschnittstelle zu senden. Dieses Verfahren bietet die größte Flexibilität, da sich hier selbst Hardware-Treiber austauschen lassen. Der größte Nachteil besteht darin, dass zum einen Programmcode sehr groß ist und zum anderen dafür der Mikroprozessor aufwendig neu pro-

grammiert bzw. geflashed werden muss. Der Flash-Vorgang im laufenden Betrieb ist enorm aufwendig und sehr energieintensiv (was das Verfahren gerade auf Hardware mit begrenzten Ressourcen/Energy-Harvesting unbrauchbar macht) und stellt des Weiteren ein Risiko dar, falls er nicht vollständig abgeschlossen werden kann (in diesem Fall ist das Gerät dann ggf. „gebrickt" und von außen nicht mehr ansprechbar, es sei denn, man verbindet den Prozessor wieder über einen Debugger für die Prozessorfamilie mit einem PC).

Die Variante der Szenarien/Parametrierung versucht den Ansatz, eine Auswahl an möglichen Programmen, die der Benutzer verwenden will, „vorzuprogrammieren" – d. h., es existieren mehrere kleine Unterprogramme auf dem Gerät, welche alle zur Auslieferung kompiliert auf der Hardware existieren. Soll das Gerät seine Funktionalität ändern, kann ihm mitgeteilt werden, dass es jetzt ein anderes Szenario/Programm auswählen soll.

Durch dieses Verfahren ist man allerdings am Ende sehr unflexibel – es können nur die Szenarien gewählt werden, die der Hersteller bereitgestellt hat. Eine Verwendung der Hardware für andere als die vorgesehenen Szenarien oder eine Änderung einer kleinen Regel in einem Szenario ist nicht vorgesehen. Ein weiterer kleiner Nachteil ist, dass der Speicherbedarf im Gerät durch das wesentlich größere Gesamt-Programm etwas erhöht ist.

Ein Interpreter ist ein Programm, welches in der Lage ist, Code in der sog. „Interpretersprache" in ein Hochsprachliches Äquivalent (z. B. C++) umzuwandeln. Die Umwandlung erfolgt dabei Zeile für Zeile; d. h., es muss nicht das gesamte zu interpretierende Programm permanent vorliegen und zusätzlich ist keine Aufbauphase wie bei einem Compiler notwendig. Interpreter sind tendenziell sehr aufwendige Programme, die aber durch ihre 1-zu-1-Umwandlung in Hochsprachlichen Programmcode maximale Flexibilität bereitstellen. Der Nachteil ist hier analog zu dem Verfahren des Neuflashens, dass der übertragene Programmcode (trotz Interpretersprache) sehr groß ist. Des Weiteren wird bei dieser Variante sehr viel Speicher benötigt, da permanent der zu interpretierende Code im Speicher gehalten werden muss sowie auch die Elemente des auszuführenden Programmes. Interpreter werden allgemein auf „fester Hardware" (Desktop PCs etc.) eingesetzt.

Die Vorteile der erfindungsgemäßen FPGRA-Lösung liegen dabei darin, dass zur Änderung der gesamten Funktionalität nur ein relativ kleiner Code gesendet werden muss, wodurch sich eine große Flexibilität ergibt.

Grundsätzlich ermöglichen erfindungsgemäße FPGRA-Ausführungsbeispiele eine hohe Flexibilität, wenig Übertragung und benötigen keinen energieaufwendigen Flash-Vorgang. Diese Vorteile können möglicherweise einhergehen mit einer etwas schlechteren Ausführungseffizienz des Programmes, da simpel gesagt immer der Umweg über den Graphen und den Knoten-Datentyp gegangen wird, was allerdings für den Einsatzbereich weniger wichtig ist, da z. B. das Senden oder Empfangen einer Nachricht über Funk weit mehr Energie verbraucht als das Ausführen von Code auf dem Haupt-Prozessor – vom Flash-Vorgang und darüber hinaus auch weitere Vorteile erhalten werden. Dadurch ist das beschrieben Konzept hervorragend dazu geeignet, auf Hardware mit begrenzten Ressourcen in Form von Energie sowie Geräten mit Energy-Harvesting eingesetzt zu werden.

Für den Beweis der Turing-Vollständigkeit wird gezeigt, dass mit der Syntax, welcher von den Knoten im Graphen aufgespannt wird, GOTO-Programme ausführbar sind/die GOTO-Language vollständig abgebildet wird („GOTO-Berechenbarkeit"). Dazu wird gezeigt, dass die komplette Syntax der GOTO-Language durch die Knoten repräsentiert werden kann.

Jede Funktion, die GOTO-berechenbar ist, ist Turing-vollständig (und umgekehrt).

- Die Speicheradressen/Index-Nummern der Knoten sind äquivalent zu den Sprungmarken im GOTO-Programm
- Die Zuweisung einer Variable durch eine andere vermehrt um eine Konstante ist durch den Arithmetikknoten gegeben
- Die Zuweisung einer Variable durch eine andere (ohne Veränderung des Wertes) ist durch den Speicherknoten gegeben
- Die Erhöhung einer Variablen um eine Konstante ist durch einen Arithmetikknoten gegeben
- Die Sprunganweisung GOTO steckt implizit in den Pointern jedes Knotens
- Die bedingte Sprunganweisung wird vollständig durch die Vergleichsknoten repräsentiert
- Die STOP-Anweisung wird durch das Setzen des „next-pointers" auf die „Null-ID" repräsentiert

Es kann also jedes GOTO-Programm durch Knoten repräsentiert werden, was heißt, dass jedes Turing-vollständige Programm durch die Knoten-Struktur repräsentiert werden kann.

Einige Ausführungsbeispiele sind nachfolgend beschrieben. So bezieht sich ein Ausführungsbeispiel auf ein IoT-basiertes Heizungsthermostat, bei dem die Peripherie einen Temperatursensor und ein Stellglied oder Aktuator um Stellen eines Heizungsventils aufweist. Ein Generator oder ein Teil einer Steuereinrichtung ist ausgebildet, um basierend auf einem von dem IoT-basierten Heizungsthermostat empfangenen Konfigurationssignal einen Handlungsablauf zu erstellen, dessen graphische Repräsentation einen ersten Graphen aufweist, der eine widerholte, ggf. zyklische Abfrage des Temperatursensors bewirkt. Die ausgelesenen Sensorsignale können lokal gespeichert und/oder an einen Server übertragen werden. Der Handlungsablauf bewirkt im selben Graphen oder als Teil eines weiteren Graphen ein Steuern des Aktuators in Abhängigkeit der festgestellten Temperatur, um etwa ein Ventil zu schließen sofern eine Solltemperatur erreicht oder überschritten ist und/oder um das Ventil zu öffnen, sollte die Temperatur unterschritten sein. Gemäß Ausführungsbeispielen lässt sich eine derartige Vorrichtung auch für Kühlgeräte, etwa Klimaanlagen einsetzen, wobei dann das Ventil geöffnet wird, wenn eine Solltemperatur überschritten ist und geschlossen wird, wenn die Solltemperatur erreicht oder unterschritten ist.

Fig. 15 zeigt ein schematisches Blockschaltbild eines Systems 150 gemäß einem Ausführungsbeispiel, das beispielhaft die Vorrichtung 10 umfasst, alternativ oder zusätzlich aber auch andere hierin beschriebene Vorrichtungen umfassen kann, beispielsweise die Vorrichtung 30, 110, 120, 130 oder 140.

Das System 150 umfasst ferner eine Vorrichtung 155, die eine Ausgangsschnittstelle aufweist, die ausgebildet ist, um das Konfigurationssignal 14 an die Vorrichtung 10 zu senden. Die Vorrichtung 155 umfasst einen Signalgenerator 92, der ausgebildet ist, um das Konfigurationssignal, zumindest in einer Basisband-Version hiervon, bereitzustellen. Hierbei wird das Konfigurationssignal 14 bevorzugt so durch den Signalgenerator 92 bereitgestellt, dass es eine Abfolge von ausführbaren Ablaufteilen einer Vielzahl von ausführbaren Ablaufteilen beschreibt, also eine Teilmenge der in der Vorrichtung 10 hinterlegten ausführbaren Ablaufteilen selektiert sowie deren Verknüpfung untereinander angibt. Die angegebene Abfolge von ausführbaren Ablaufteilen beschreibt einen Handlungsablauf, etwa den Handlungsablauf 24, eindeutig. Die Vorrichtung 155 kann ausgebildet sein, um eine Benutzereingabe 94 zu erhalten, die den zu erstellenden Handlungsablauf beschreibt. Hierfür kann beispielsweise eine graphische Repräsentation 96 des Handlungsablaufs 24 vorgesehen sein, die der Benutzer an einer hierfür vorgesehenen Vorrichtung 98 erstellen oder konfigurieren kann. Die Vorrichtung 98 kann dabei Teil der Vorrichtung 155 sein,

kann aber auch Zugriff auf ein Web-Interface 102 der Vorrichtung 155 nutzen, so dass die Benutzereingabe 94 über eine Netzwerkverbindung von der über die Netzwerkverbindung verbundenen Vorrichtung 98 empfangen werden kann. Die graphische Repräsentation 96 ist dabei lediglich eine von mehreren möglichen Ausgestaltungen, die alternativ oder zusätzlich auch eine Programmierung in Textform beinhalten kann.

In anderen Worten kann ein Arbeitsablauf Folgendes umfassen:

a) Zusammenstellen der abzulaufenden Logik/des Graphen oder des Handlungsablaufs an einem Computer über beispielsweise ein Web-Interface oder ein Programm;
b) Komprimierung und Codierung der Logik/des Graphen;
c) Senden der codierten Logik über eine Kommunikationsschnittstelle an das Gerät, das bedeutet Senden des Konfigurationssignals;
d) Decodieren der Nachricht und Aufbau der inneren Logik im Gerät, wodurch dieses nun einsatzfähig ist und in Dauerschleife das decodierte Programm ausführt.

Fig. 16 zeigt ein schematisches Blockschaltbild eines Systems 160 gemäß einem Ausführungsbeispiel. Das System 160 kann die Vorrichtungen 98 (Block B) und 155 (Block C) umfassen, die ausgebildet sind, um einen Datenfluss zum Austausch von Logik und/oder Parametern oder Soll-Werten auszutauschen, wie es durch den Buchstaben a angedeutet ist. Die Vorrichtung 155, das heißt der Codierer, kann mit einer oder mehreren Vorrichtungen 10_1 und/oder 10_2 über ein Kommunikationsnetzwerk (Block D), etwa dem Internet, in Verbindung stehen, ebenso wie die Vorrichtung 98. Die Peripherie 32 (Block E) kann mit den Vorrichtungen 10_1 und/oder 10_2 ebenfalls über das Kommunikationsnetzwerk 104 in Verbindung stehen. An der Vorrichtung 98 zur Programmierung des Graphen oder der Logik kann auch Metainformation für die Peripherie erstellt werden, etwa eine Benamung oder virtuelle Adressen, wie es durch den Buchstaben e dargestellt ist. Zwischen den Vorrichtungen 10_1 und 10_2 und der Vorrichtung 98 können weitere Informationen, etwa hardwarespezifische Informationen wie IDs (Identifikationen), Spezifikationen und/oder hardwarenahe Belegungen ausgetauscht werden, wie es durch den Buchstaben b dargestellt ist. Darüber hinaus können die Vorrichtungen 10_1 und 10_2 mit der Peripherie 32 unter Verwendung des Kommunikationsnetzwerks 104 Ausgabewerte, Sensorwerte oder Ist-Parameter (Buchstabe c) und/oder Soll-Parameter (Buchstabe b) austauschen. Ein Austausch von Logikinformationen oder Parametrierungen kann zwischen den Vorrichtungen

98 und 155 sowie unter Verwendung des Kommunikationsnetzwerks 104 mit den Vorrichtungen 10_1 und 10_2 ausgetauscht werden, wie es durch den Buchstaben a dargestellt ist. Dies beinhaltet beispielsweise das Konfigurationssignal.

Die Vorrichtungen 10_1 und 10_2 können untereinander (Buchstabe f) und mit dem Kommunikationsnetzwerk 104 unter Verwendung eines ressourcenbeschränkten Mediums 106 kommunizieren, beispielsweise schmalbandige Kanäle eines Mobilfunknetzwerks (schmalbandig = narrowband Nb), etwa für Nb-LTE zum Implementieren von Nb-IoT-Vorrichtungen. Die Peripherie 32 kann jedwede Art von Datenempfängern/Senken, Aktuatoren, Steuer-Devices zur Parametrierung anpassen, darunter auch mobile Geräte wie etwa Smartphones.

In anderen Worten: kann ein Zahlencode, der bspw. eine Sequenz der Parameter widergibt, (im Datenflussgraphen (a) – in den Zahlen sind die Parameter codiert, welche später den Programmablauf steuern) in eine ausführbare Programmlogik umgewandelt werden, die dann in Dauerschleife auf dem Gerät (im Datenflussgraphen (A)) ausgeführt wird. Man kann sich dies in etwa wie bei einem Compiler vorstellen, welcher Programmcode (z. B. C++) in Maschinensprache übersetzt, welche dann auf einer Hardware ausgeführt werden kann.

Der Unterschied zwischen dem FRGRA und einem Compiler ist, dass hierbei kein ausführbarer Maschinencode erzeugt wird. Das FPGRA ist auch kein Interpreter, da der Code nicht direkt in ein Hochsprachliches Äquivalent umgewandelt und ausgeführt wird. Es ist analog zum Compiler keine „Aufbauphase“ nötig.

Es wird im bestehenden Programmcode durch geschickte Pointer-Arithmetik und Klassenstrukturen, welche in der Lage sind, den Programmablauf zu steuern, eine Logik aufgebaut, die im flüchtigen Speicher gehalten wird.

Der eigentliche Programmcode, welcher auf dem Gerät permanent ausgeführt wird, bleibt dabei unangetastet und unverändert (vereinfacht gesagt läuft die ganze Zeit das gleiche Programm, nur mit anderen Parametern).

Durch dieses Vorgehen kann auf den energieaufwendigen Flash-Vorgang zum Ändern von Strukturen im nicht-flüchtigen Speicher (EEPROM) verzichtet werden und es besteht trotzdem die Möglichkeit, das Gerät Turing-vollständig umzuprogrammieren.

Der Benutzer setzt hier den Programmablauf fest, dem das Gerät später folgen soll. Dies kann über das „Zusammenklicken" eines Graphen geschehen, der die Regeln und Handlungsabläufe beinhaltet; aber auch z. B. über eine Art von Skriptsprache, mit der sich die Ablaufregeln definieren lassen („Wenn X, dann Y, sonst Z -> Danach A").

Das Interface legt neben in der inneren Logik auch die möglichen Aktionen des Gerätes bzw. die Ausgänge fest. Eine Aktion kann beispielsweise eine Hardware-Funktion sein („Sende XY", „Setze Time N auf X"), aber auch einfach ein Funktionsaufruf in der Software. Diese Vorauswahl ist nötig, da zum einen technisch die Menge der Aktionsknoten (welche die verfügbaren Aktionen im Gerät zur Verfügung stellen im Gerät begrenzt ist – da in der Realität der verfügbare Speicher immer begrenzt ist – und zum anderen die Ausgänge und ausführbaren Aktionen hardwarespezifisch sind (beispielsweise darf ein Gerät, welches über keinen Aktor verfügt, nicht mit einem Regelwerk versehen werden, welches am Ende einen Aktor bedienen will).

Selbiges gilt für die möglichen Eingänge/Sensoren und Kommunikationsschnittstellen (beispielsweise darf ein Graph, der im Eingang einen Temperatur-Sensor auswertet, auch nur auf Geräte, die einen solchen Sensor besitzen, gebracht werden oder WiFi-Kommunikation nur auf WiFi-fähige Geräte.

Ausführungsbeispiele basieren darauf, den für den konkreten Anwendungsfall der Vorrichtung relevanten Teil auszuführender Instruktionen (Programmcode) als Regelbaum darzustellen, das bedeutet, die implementierten Funktionen sind als Regelbaum darstellbar. Ein derartiger Graph kann durch eine spezielle Struktur im Programmcode repräsentiert werden und kann jederzeit durch eine Konfigurationsnachricht, die wesentlich weniger Information zur Übertragung umfasst als ein normaler Programmcode, neu zusammengebaut werden, um ein anderes Programm zu repräsentieren. Wie es beschrieben ist, kann diese Art der Repräsentation Turing-vollständig sein, wodurch effektiv jedes beliebige Programm dargestellt werden kann, solange es in dem Speicher speicherbar ist. Knoten des Graphen können als so etwas wie die kleinstmöglichen Unterprogramme verstanden werden, die auf der Hardware möglich sind. Hierzu zählen logische und/oder arithmetische Funktionen („UND", „ODER", „+", „*", ...), aber auch komplexere Programmabläufe/-funktionen wie z. B. Instruktionen gemäß „Sende den Wert an Adresse x zurück an den Server" oder „Lege dich für x Minuten schlafen". Beispielhafte Konfigurationen, die die Vielfältigkeit demonstrieren, jedoch nicht einschränkend wirken sollen, sind hierin be-

schrieben. Es ist anzumerken, dass eine einzelne Vorrichtung zwischen unterschiedlichen Konfigurationen, die beschrieben sind, mit gegebenenfalls lediglich einer Konfigurationsnachricht wechseln kann, d. h. umprogrammiert werden kann, ohne dass dabei der ausgeführte Programmcode in einem Festspeicher (EEPROM) geändert werden muss.

Fig. 17 zeigt ein schematisches Ablaufdiagramm eines Verfahrens 1700 gemäß einem Ausführungsbeispiel, das beispielsweise von einer hierin beschriebenen fernprogrammierbaren Vorrichtung, etwa der Vorrichtung 10, 30, 110, 120, 130 oder 140, ausgeführt werden kann. Ein Schritt 1710 umfasst ein Speichern einer Vielzahl ausführbarer Ablaufteile in einem Datenspeicher, wobei jeder Ablaufteil zumindest eine Handlungsvorschrift beschreibt. Ein Schritt 1720 umfasst ein Empfangen eines Konfigurationssignals mit einer Eingangsschnittstelle. Ein Schritt 1730 umfasst ein Erstellen eines Handlungsablaufs, der eine Abfolge von in dem Konfigurationssignal angegebenen Ablaufteilen der Vielzahl von gespeicherten Ablaufteilen umfasst. Ein Schritt 1740 umfasst ein Ansprechen einer Peripherie gemäß dem Handlungsablauf unter Verwendung einer Schnittstelle, die ausgebildet ist, um mit der Peripherie zu kommunizieren.

Fig. 18 zeigt ein schematisches Flussdiagramm eines Verfahrens 1800 zum Bereitstellen eines Konfigurationssignals, das beispielsweise unter Verwendung der Vorrichtung 155 ausführbar ist. Ein Schritt 1810 umfasst ein Bereitstellen eines Konfigurationssignals unter Verwendung eines Signalgenerators, so dass das Konfigurationssignal eine Abfolge von ausführbaren Ablaufteilen aus einer Vielzahl von ausführbaren Ablaufteilen beschreibt, wobei die Abfolge von Ablaufteilen einen Handlungsablauf mit einer Mehrzahl von Handlungsvorschriften eindeutig beschreibt. Ein Schritt 1820 umfasst ein Senden des Konfigurationssignals mit einer Ausgangsschnittstelle.

Fig. 19 zeigt ein schematisches Blockschaltbild einer Vorrichtung 190 gemäß einem Ausführungsbeispiel, bei der die Eingängen 28_1 zur Verbindung mit einem IoT Modem 108 eingerichtet sind, etwa zur Kommunikation über das resourcenbeschränkte Medium 106. Das IoT Modem 108 kann alternativ auch eine interne Vorrichtung sein. Ferner können die Eingänge eingerichtet sein, um mit einem oder mehreren Sensoren 74 verbunden zu werden, etwa ein MEMS-Sensor. Dieser kann bspw. als Beschleunigungssensor eingerichtet sein. Der Ausgang 28_5 kann zur Kommunikation mit einem Server der Peripherie 32 eingerichtet sein. Das Gerät kann ausgebildet sein, um ansprechend auf ein empfangenes Konfigurationssignal den Handlungsablauf so zu erstellen, dass bei entsprechendem Signal des Sensors 74, was eine Bewegung oder eine ausbleibende Bewegung oder

eine entsprechende Beschleunigung sein kann, eine Nachricht an den Server gesendet wird, die eine Position der Vorrichtung 190 angibt, etwa unter Angabe einer durch das Modem 108 verwendeten Funkzelle, in welcher sich die Vorrichtung gegenwärtig befindet.

In anderen Worten kann die Vorrichtung 190 als Bewegungstracker fungieren, etwa für Transportgut eines Logistik-Unternehmens. Als Eingänge dienen hier ein Beschleunigungs-Sensor („MEMS") und die aktuell Empfangene Funkzelle des NB-IoT Modems (dieses ist bspw. kein extra Gerät, sondern ggf. im Gerät vorhanden). Der Benutzer könnte hier beispielsweise das Gerät so konfigurieren, dass es nachdem initial eine Bewegung detektiert wurde über den MEMS, das Gerät in einen „Transport-Zustand" fällt (was über eine Interne Variable gelöst ist). Im „Transport-Zustand" prüft das Gerät dann in einem Regelmäßigen Intervall, ob es sich noch in Bewegung befindet. Sobald n-Mal keine Bewegung festgestellt wurde, wird der „Transport-Zustand" beendet und das Gerät sendet die aktuell Verbundene Funkzelle an einen Server. So kann ein Unternehmer bspw. Feststellen, welche Fracht wann und wohin Transportiert wurde. Die Schwellwerte, ab wann das Gerät in Bewegung ist und in welchem Intervall das Gerät wieder auf Bewegung prüft sind selbstverständlich frei wählbar und mittels des Konfigurationssignals festlegbar bzw. mittels erneuter Konfigurationssignale veränderbar. Ebenso sind komplexere Verschachtelungen mit weiteren Zuständen möglich.

Fig. 20 zeigt ein schematisches Blockschaltbild einer Vorrichtung 200 gemäß einem Ausführungsbeispiel, bei der die Eingängen 28_1 zur Verbindung mit einer mobilen Vorrichtung 112, etwa ein Smartphone/Mobiltelefon, ein Tabletcomputer oder dergleichen eingerichtet sind. Die mobile Vorrichtung 112 kann eine Positioniervorrichtung 114 aufweisen über die eine Position feststellbar ist, etwa mittel Global Positioning System (GPS), Galileo oder Glonass. Alternativ kann eine entsprechende Positioniervorrichtung auch als eigenständige Vorrichtung vorgesehen sein und über die Eingänge 28_1 mit der Vorrichtung 200 verbunden sein. Die Vorrichtung 200 kann ausgebildet sein, um ansprechend auf ein Konfigurationssignal den Handlungsablauf so zu erstellen, dass über die Ausgänge 28_5 ein oder mehrere Vorrichtungen oder Komponenten angesprochen werden. So kann ein Derver 32_1 eine Nachricht mit einer Position der Vorrichtung 200 empfangen, etwa in vorgegebenen Intervallen und/oder wenn die mittels der Positioniervorrichtung 114 empfangene Position von einer vorgegebenen und in der Vorrichtung 200 hinterlegten oder von extern empfangenen Position oder Bereich abweicht. Alternativ oder zusätzlich kann die mobile Vorrichtung 112 entsprechend informiert werden, etwa eine Anwendung (Application;

App), die dort ausgeführt wird. Alternativ oder zusätzlich die Vorrichtung 200 ausgebildet sein, um eine Anzeigeeinrichtung 32_2 anzusteuern, um einen Status anzeigen.

In anderen Worten kann die Vorrichtung 200 als Diebstahl-Sicherung fungieren, bspw. für bspw. Fahrräder oder Fahrzeuge. Die Sicherung wird bspw. über eine App im Handy 112 aktiviert (der Zustand in der App wird an das Gerät gesendet und ist intern in einer Variable gespeichert, welche als Eingang des Handlungsablaufs gesetzt werden kann), dies wird zusätzlich über eine angeschlossene LED visualisiert. Im aktivierten Modus merkt sich das Gerät seine aktuelle Position über ein GPS-Modul 114. Diese wird in einem regelmäßigem, vom Anwender vorgegebenen Intervall aktualisiert. Bei einer Änderung meldet das Gerät sofort seine aktuelle Position in einem regelmäßigen Intervall zurück an das Smartphone 112 des Benutzers und/oder einen Server 32_1.

Fig. 21 zeigt ein schematisches Blockschaltbild einer Vorrichtung 210 gemäß einem Ausführungsbeispiel, die konfiguriert ist, um über die Eingängen 28_1 mit unterschiedlichen Einrichtungen oder Vorrichtungen verbunden zu werden, um Eingangssignale zu erhalten. Hierzu gehören bspw. eine Drahtlosschnittstelle 116, etwa Bluetooth ® oder eine andere Drahtlostechnologie, ein Zeitgeber oder Uhr 118, eine Signalgenerator einer Benutzereingabe, etwa ein Schalter oder Knopf 122 und/oder andere Sensoren wie ein Pulsmesser 124. Die Daten können verarbeitet werden und entsprechende Ergebnisse an eine Anzeigeeinrichtung 126, etwa ein LCD (LCD = Liquid Chrystal Display), ein LED-Display, eine einzelne LED oder dergleichen zum Darstellen einer Information. Die Vorrichtung 210 kann ausgebildet sein, um bspw. basierend auf der Benutzereingabe zwischen zwei oder mehreren Betriebsmodi umzuschalten, in denen an der Anzeigeeinrichtung 126 unterschiedliche Informationen dargestellt werden, etwa über die Schnittstelle 116 empfangene Informationen, eine Zeit des Zeitgebers 118 und/oder Informationen des Sensors 124 und/oder hieraus abgeleiteter Informationen. Diese Informationen können über die Ausgänge 28_5 sowohl übermittelt werden, etwa unter Verwendung der Schnittstelle 116, als auch alternativ oder zusätzlich dargestellt werden.

In anderen Worten kann die Vorrichtung 210 eine Art intelligente Puls-Uhr bereitstellen. Über einen Knopf 122 kann umgeschaltet werden, was die LCD-Anzeige 126 darstellen soll. Über Bluetooth 116 kann das Gerät Informationen von anderen Devices (bspw. dem zugehörigen Brustgurt bei differentieller Pulsmessung oder dem Tachometer eines Fahrrads oder ein Smartphone) erhalten bzw. an andere Devices senden. Die über Bluetooth

116 erhaltenen Informationen können z.B. dafür genutzt werden, um einen Alarm auf dem Display 126 anzuzeigen, wenn ein vom Benutzer eingestellter Wert überschritten wird.

Die hierin beschriebenen Vorrichtung können unter Verwendung von Konfigurationssignalen programmiert oder umprogrammiert werden. Hierbei können entsprechende Teil-Handlungsabläufe in den Vorrichtung vorhinterlegt sein und mittels des Konfigurationssignals neu verknüpft oder konfiguriert werden. In Vorrichtung unterschiedlicher Ausgestaltung können die Ablaufteile entsprechend der zur Vorrichtung gehörenden oder vorgesehenen oder möglichen Hardware unterschiedlich sein, die Ausgestaltung des Generators kann jedoch gleich oder gleichwirkend sein.

Hierin beschriebene Ablaufteile können als vordefinierte und/oder parametrierbare und/oder anpassbare Codebausteine verstanden werden, die zu einem Gesamtcode, dem Handlungsablauf, in durch das Konfigurationssignal bestimmter Auswahl und/oder Eigenschaft und/oder Reihenfolge zusammengefügt werden können.

Obwohl manche Aspekte im Zusammenhang mit einer Vorrichtung beschrieben wurden, versteht es sich, dass diese Aspekte auch eine Beschreibung des entsprechenden Verfahrens darstellen, sodass ein Block oder ein Bauelement einer Vorrichtung auch als ein entsprechender Verfahrensschritt oder als ein Merkmal eines Verfahrensschrittes zu verstehen ist. Analog dazu stellen Aspekte, die im Zusammenhang mit einem oder als ein Verfahrensschritt beschrieben wurden, auch eine Beschreibung eines entsprechenden Blocks oder Details oder Merkmals einer entsprechenden Vorrichtung dar.

Je nach bestimmten Implementierungsanforderungen können Ausführungsbeispiele der Erfindung in Hardware oder in Software implementiert sein. Die Implementierung kann unter Verwendung eines digitalen Speichermediums, beispielsweise einer Floppy-Disk, einer DVD, einer Blu-ray Disc, einer CD, eines ROM, eines PROM, eines EPROM, eines EEPROM oder eines FLASH-Speichers, einer Festplatte oder eines anderen magnetischen oder optischen Speichers durchgeführt werden, auf dem elektronisch lesbare Steuersignale gespeichert sind, die mit einem programmierbaren Computersystem derart zusammenwirken können oder zusammenwirken, dass das jeweilige Verfahren durchgeführt wird. Deshalb kann das digitale Speichermedium computerlesbar sein. Manche Ausführungsbeispiele gemäß der Erfindung umfassen also einen Datenträger, der elektronisch lesbare Steuersignale aufweist, die in der Lage sind, mit einem programmierbaren Com-

putersystem derart zusammenzuwirken, dass eines der hierin beschriebenen Verfahren durchgeführt wird.

Allgemein können Ausführungsbeispiele der vorliegenden Erfindung als Computerprogrammprodukt mit einem Programmcode implementiert sein, wobei der Programmcode dahin gehend wirksam ist, eines der Verfahren durchzuführen, wenn das Computerprogrammprodukt auf einem Computer abläuft. Der Programmcode kann beispielsweise auch auf einem maschinenlesbaren Träger gespeichert sein.

Andere Ausführungsbeispiele umfassen das Computerprogramm zum Durchführen eines der hierin beschriebenen Verfahren, wobei das Computerprogramm auf einem maschinenlesbaren Träger gespeichert ist.

Mit anderen Worten ist ein Ausführungsbeispiel des erfindungsgemäßen Verfahrens somit ein Computerprogramm, das einen Programmcode zum Durchführen eines der hierin beschriebenen Verfahren aufweist, wenn das Computerprogramm auf einem Computer abläuft. Ein weiteres Ausführungsbeispiel der erfindungsgemäßen Verfahren ist somit ein Datenträger (oder ein digitales Speichermedium oder ein computerlesbares Medium), auf dem das Computerprogramm zum Durchführen eines der hierin beschriebenen Verfahren aufgezeichnet ist.

Ein weiteres Ausführungsbeispiel des erfindungsgemäßen Verfahrens ist somit ein Datenstrom oder eine Sequenz von Signalen, der bzw. die das Computerprogramm zum Durchführen eines der hierin beschriebenen Verfahren darstellt bzw. darstellen. Der Datenstrom oder die Sequenz von Signalen kann bzw. können beispielsweise dahin gehend konfiguriert sein, über eine Datenkommunikationsverbindung, beispielsweise über das Internet, transferiert zu werden.

Ein weiteres Ausführungsbeispiel umfasst eine Verarbeitungseinrichtung, beispielsweise einen Computer oder ein programmierbares Logikbauelement, die dahin gehend konfiguriert oder angepasst ist, eines der hierin beschriebenen Verfahren durchzuführen.

Ein weiteres Ausführungsbeispiel umfasst einen Computer, auf dem das Computerprogramm zum Durchführen eines der hierin beschriebenen Verfahren installiert ist.

Bei manchen Ausführungsbeispielen kann ein programmierbares Logikbauelement (beispielsweise ein feldprogrammierbares Gatterarray, ein FPGA) dazu verwendet werden, manche oder alle Funktionalitäten der hierin beschriebenen Verfahren durchzuführen. Bei manchen Ausführungsbeispielen kann ein feldprogrammierbares Gatterarray mit einem Mikroprozessor zusammenwirken, um eines der hierin beschriebenen Verfahren durchzuführen. Allgemein werden die Verfahren bei einigen Ausführungsbeispielen seitens einer beliebigen Hardwarevorrichtung durchgeführt. Diese kann eine universell einsetzbare Hardware wie ein Computerprozessor (CPU) sein oder für das Verfahren spezifische Hardware, wie beispielsweise ein ASIC.

Die oben beschriebenen Ausführungsbeispiele stellen lediglich eine Veranschaulichung der Prinzipien der vorliegenden Erfindung dar. Es versteht sich, dass Modifikationen und Variationen der hierin beschriebenen Anordnungen und Einzelheiten anderen Fachleuten einleuchten werden. Deshalb ist beabsichtigt, dass die Erfindung lediglich durch den Schutzumfang der nachstehenden Patentansprüche und nicht durch die spezifischen Einzelheiten, die anhand der Beschreibung und der Erläuterung der Ausführungsbeispiele hierin präsentiert wurden, beschränkt sei.

Patentansprüche

1. Vorrichtung mit:

einer Eingangsschnittstelle (12), die ausgebildet ist, um ein Konfigurationssignal (14) zu empfangen;

einem Datenspeicher (16; 36), in dem eine Vielzahl ausführbarer Ablaufteile (18) gespeichert ist, wobei jeder Ablaufteil (18) zumindest eine Handlungsvorschrift beschreibt;

einem Generator (22), der ausgebildet ist, um einen Handlungsablauf (24) zu erstellen, der eine Abfolge von in dem Konfigurationssignal (14) angegebenen Ablaufteilen (18) der Vielzahl von gespeicherten Ablaufteilen umfasst;

eine Schnittstelle (28), die ausgebildet ist, um mit einer Peripherie (32) zu kommunizieren; und

einem Prozessor (31), der konfiguriert ist, um die Peripherie (32) gemäß dem Handlungsablauf (24) anzusprechen.

2. Vorrichtung gemäß Anspruch 1, wobei der Datenspeicher (36) einen ersten Speicherabschnitt (16) und einen zweiten Speicherabschnitt (34) umfasst.

3. Vorrichtung gemäß Anspruch 2, wobei der erste Speicherabschnitt (16) ausgebildet ist, um die Vielzahl ausführbarer Ablaufteile (18) zu speichern und der Generator (22) ausgebildet ist, um in dem zweiten Speicherabschnitt (34) den Handlungsablauf (24) zu speichern.

4. Vorrichtung gemäß einem der Ansprüche 2 oder 3, wobei der erste Speicherabschnitt (16) einen nicht-flüchtigen Speicherabschnitt umfasst und der zweite Speicherabschnitt (34) einen flüchtigen Speicherabschnitt umfasst.

5. Vorrichtung gemäß einem der Ansprüche 1 bis 4, wobei das Konfigurationssignal (14) eine Information (26) aufweist, die eine Abfolge von ausführbaren Ablaufteilen

(18) der Vielzahl ausführbarer Ablaufteile beschreibt und so den Handlungsablauf (24) eindeutig festlegt.

6. Vorrichtung gemäß Anspruch 5, wobei die Information (26) eine Sequenz von Integer-Zahlen aufweist.

7. Vorrichtung gemäß einem der Ansprüche 1 bis 6, wobei der Generator (22) ausgebildet ist, um den Handlungsablauf (24) so zu erstellen, dass dieser durch zumindest einen Regelbaum (44_1, 44_2) ausführbarer Ablaufteile (18) der Vielzahl ausführbarer Ablaufteile repräsentiert wird, wobei der Regelbaum (44_1, 44_2) eine Mehrzahl von Knoten (46, 48, 52) und eine Mehrzahl von Verknüpfungen (54) zwischen Knoten (46, 48, 52) umfasst, und ein ausführbarer Ablaufteil (18) der Vielzahl ausführbarer Ablaufteile durch einen Knoten (46, 48, 52) in dem Regelbaum (44_1; 44_2) repräsentiert wird;

wobei zumindest eine Verknüpfung (54) der Mehrzahl von Verknüpfungen von einem ersten Knoten (52_1) auf einen nächsten, zweiten Knoten (52_2) des Regelbaumes zeigt.

8. Vorrichtung gemäß einem der Ansprüche 1 bis 7, wobei der Generator (22) ausgebildet ist, um den Handlungsablauf (24) so zu erstellen, dass dieser zumindest einen Startknoten (46), an welchem der Handlungsablauf durch den Prozessor (31) begonnen wird, umfasst;

wobei der Prozessor (31) ausgebildet ist, um den zumindest einen Startknoten (46) periodisch auszuwerten, um ein Auswertungsergebnis zu erhalten.

9. Vorrichtung gemäß Anspruch 8, wobei der Prozessor (31) konfiguriert ist, um die Vielzahl ausführbarer Ablaufteile (18) des Handlungsablaufs (24) vom dem Startknoten (46) des Handlungsablaufs (24) bis zu einem Ende (48) des Handlungsablaufs (24) auszuführen.

10. Vorrichtung gemäß Anspruch 8 oder 9, wobei das Konfigurationssignal (14) eine Information aufweist, die den zumindest einen Startknoten (46) des Handlungsablaufs (24) definiert.

11. Vorrichtung gemäß einem der Ansprüche 1 bis 10, wobei der Handlungsablauf (24) zumindest eine Teilmenge der in dem Datenspeicher (16) gespeicherten Vielzahl ausführbarer Ablaufteile (18) umfasst; und

wobei der Prozessor (31) ausgebildet ist, um zum Ansprechen der Peripherie (32) die Teilmenge von ausführbaren Ablaufteilen (18) zu durchlaufen und in einem ersten Ablaufteil (18_1) zumindest eine Eingangsgröße zu verarbeiten, um eine Ausgangsgröße zu erhalten, und um die Ausgangsgröße als Eingangsgröße eines zweiten Ablaufteils (18_2) zu verwenden.

12. Vorrichtung gemäß einem der Ansprüche 1 bis 11, wobei ein erster ausführbarer Ablaufteil des Handlungsablaufs zwei Eingänge (56) und zwei Ausgänge (58) aufweist; und

wobei mindestens ein Ausgang (58) des ersten ausführbaren Ablaufteils (18_1) mit einem nächsten, zweiten ausführbaren Ablaufteil (18_2) des Handlungsablaufs (24) durch eine Verknüpfung (54) verknüpft ist.

13. Vorrichtung gemäß einem der Ansprüche 1 bis 12, wobei jedem ausführbarem Ablaufteil (18) der Vielzahl ausführbarer Ablaufteile eine Funktion, die eine Verarbeitung mindestens einer Eingangsgröße des ausführbaren Ablaufteils definiert, zugeordnet ist; und

wobei die Funktion eine Vergleichsoperation, eine Arithmetikoperation, eine Speicheroperation oder einen Aktionsoperation umfasst.

14. Vorrichtung gemäß einem der Ansprüche 1 bis 13, wobei eine Art eines ausführbaren Ablaufteils der Vielzahl ausführbarer Ablaufteile durch fünf Parameter eindeutig definiert ist.

15. Vorrichtung gemäß Anspruch 14, wobei ein erster Parameter der fünf Parameter die Funktion des ausführbaren Ablaufteils der Vielzahl ausführbarer Ablaufteile (18) definiert;

wobei ein zweiter Parameter der fünf Parameter einen ersten Speicherbereich, der mit einem ersten Eingang des ausführbaren Ablaufteils assoziiert ist, definiert;

wobei ein dritter Parameter der fünf Parameter einen zweiten Speicherbereich, der mit einem zweiten Eingang des ausführbaren Ablaufteils assoziiert ist, definiert;

wobei ein vierter Parameter der fünf Parameter einen dritten Speicherbereich, der mit einem ersten Ausgang des ausführbaren Ablaufteils assoziiert ist, definiert; und

wobei ein fünfter Parameter der fünf Parameter einen vierten Speicherbereich, der mit einem zweiten Ausgang des ausführbaren Ablaufteils assoziiert ist, definiert.

16. Vorrichtung gemäß einem der Ansprüche 14 oder 15, wobei der Generator (22) ausgebildet ist, um das Konfigurationssignal (14) auf Integer-Zahlen zu überprüfen, die jeden der fünf Parameter wiedergeben.

17. Vorrichtung gemäß Anspruch 16, wobei der Generator (22) ausgebildet ist, um die Integer-Zahlen, die die fünf Parameter wiedergeben, jeweils mit einem Speicherbereich des Datenspeichers eindeutig zu assoziieren.

18. Vorrichtung gemäß einem der Ansprüche 1 bis 17, wobei jedem ausführbaren Ablaufteil (18) der Vielzahl ausführbarer Ablaufteile oder des Handlungsablaufs (24) eine vordefinierte Knotennummer (62) zugeordnet ist, über die der ausführbare Ablaufteil (18) eindeutig adressierbar ist;

wobei der Prozessor (31) ausgebildet ist, jeden ausführbaren Ablaufteil (18) der Vielzahl ausführbarer Ablaufteile über die vordefinierte Knotennummer (31) zu adressieren.

19. Vorrichtung gemäß einem der Ansprüche 1 bis 18, wobei die Vorrichtung ausgebildet ist, um das Konfigurationssignal (14) an der Eingangsschnittstelle (12) über ein Kommunikationsnetzwerk (104) zu empfangen.

20. Vorrichtung gemäß Anspruch 19, wobei das Kommunikationsnetzwerk (104) ein drahtloses Netzwerk umfasst.

21. Vorrichtung gemäß Anspruch 20, wobei das drahtlose Netzwerk (104) ein Funknetzwerk umfasst.

22. Vorrichtung gemäß einem der Ansprüche 1 bis 21, wobei die Peripherie (32) zumindest einen Sensor (74) und/oder einen Server (82) und/oder einen Schalter (86) und/oder einen Aktuator und/oder einen Motor umfasst.

23. Vorrichtung gemäß einem der Ansprüche 1 bis 22, wobei der Handlungsablauf (24) ein erster Handlungsablauf ist, der von dem Prozessor basierend auf einem ersten Konfigurationssignal (14) bereitgestellt ist;

wobei die Vorrichtung ausgebildet ist, um zur Änderung des ersten Handlungsablaufes in einen zweiten Handlungsablauf ein zweites Konfigurationssignal (14) zu empfangen;

wobei der Generator (22) ausgebildet ist, um basierend auf dem zweiten Konfigurationssignal (14) eine Neudefinition zumindest eines ausführbaren Ablaufteils des ersten Handlungsablaufes auszuführen;

wobei die Neudefinition des Handlungsablaufes Turing-vollständig ist;

wobei der Generator (22) ausgebildet ist, um das zweite Konfigurationssignal (14) auf sechs Parameter auszuwerten, die den zumindest einen ausführbaren Ablaufteil (18) für die Neudefinition bestimmen.

24. Vorrichtung gemäß Anspruch 23, wobei ein erster Parameter der sechs Parameter eine Knotennummer (62) des ausführbaren Ablaufteils der Vielzahl ausführbarer Ablaufteile ist;

wobei ein zweiter Parameter der sechs Parameter eine Funktion des ausführbaren Ablaufteils neu definiert;

wobei ein dritter Parameter der sechs Parameter einen ersten Speicherbereich, der mit einem ersten Eingang des ausführbaren Ablaufteils assoziiert ist, neu definiert;

wobei ein vierter Parameter der sechs Parameter einen zweiten Speicherbereich, der mit einem zweiten Eingang des ausführbaren Ablaufteils assoziiert ist, neu definiert;

wobei ein fünfter Parameter der sechs Parameter einen dritten Speicherbereich, der mit einem ersten Ausgang des ausführbaren Ablaufteils assoziiert ist, neu definiert; und

wobei ein sechster Parameter der sechs Parameter einen vierten Speicherbereich, der mit einem zweiten Ausgang des ausführbaren Ablaufteils assoziiert ist, neu definiert.

25. Vorrichtung gemäß einem der Ansprüche 23 oder 24, wobei jeder der sechs Parameter in dem Konfigurationssignal als eine Integer-Zahl enthalten ist.

26. Vorrichtung gemäß einem der Ansprüche 1 bis 25, die eine IoT-Vorrichtung ist.

27. Vorrichtung (155) mit:

einer Ausgangsschnittstelle (88), die ausgebildet ist, um ein Konfigurationssignal (14) zu senden;

einem Signalgenerator (92), der ausgebildet ist, um das Konfigurationssignal (14) bereitzustellen, so dass das Konfigurationssignal (14) eine Abfolge von ausführbaren Ablaufteilen (18) einer Vielzahl von ausführbaren Ablaufteilen beschreibt;

wobei die Abfolge von ausführbaren Ablaufteilen (18) einen Handlungsablauf (24) mit einer Mehrzahl von Handlungsvorschriften eindeutig beschreibt.

28. Vorrichtung gemäß Anspruch 27, die ausgebildet ist, um eine Benutzereingabe (94) zu erhalten, die den Handlungsablauf (24) beschreibt.

29. Vorrichtung gemäß Anspruch 27 oder 28, die ein Web-Interface (102) bereitstellt, um die Benutzereingabe (94) von einer über eine Netzwerkverbindung verbundenen Vorrichtung (98) zu empfangen.

30. System (150) mit:

einer Vorrichtung gemäß einem der Ansprüche 1 bis 26; und

einer Vorrichtung gemäß einem der Ansprüche 27 bis 29.

31. Verfahren (1700) mit folgenden Schritten:

Speichern (1710) einer Vielzahl ausführbarer Ablaufteile in einem Datenspeicher, wobei jeder Ablaufteil zumindest eine Handlungsvorschrift beschreibt;

Empfangen (1720) eines Konfigurationssignals mit einer Eingangsschnittstelle;

Erstellen (1730) eines Handlungsablaufs, der eine Abfolge von in dem Konfigurationssignal angegebenen Ablaufteilen der Vielzahl von gespeicherten Ablaufteilen umfasst; und

Ansprechen (1740) einer Peripherie gemäß dem Handlungsablauf unter Verwendung einer Schnittstelle, die ausgebildet ist, um mit der Peripherie zu kommunizieren.

32. Verfahren (1800) mit folgenden Schritten:

Bereitstellen (1810) eines Konfigurationssignals unter Verwendung eines Signalgenerators, so dass das Konfigurationssignal eine Abfolge von ausführbaren Ablaufteilen aus einer Vielzahl von ausführbaren Ablaufteilen beschreibt, wobei die Abfolge von Ablaufteilen einen Handlungsablauf mit einer Mehrzahl von Handlungsvorschriften eindeutig beschreibt; und

Senden (1820) des Konfigurationssignals mit einer Ausgangsschnittstelle.

33. Computerprogramm mit einem Programmcode zur Durchführung des Verfahrens nach Anspruch 31 oder 32, wenn das Programm auf einem Computer läuft.

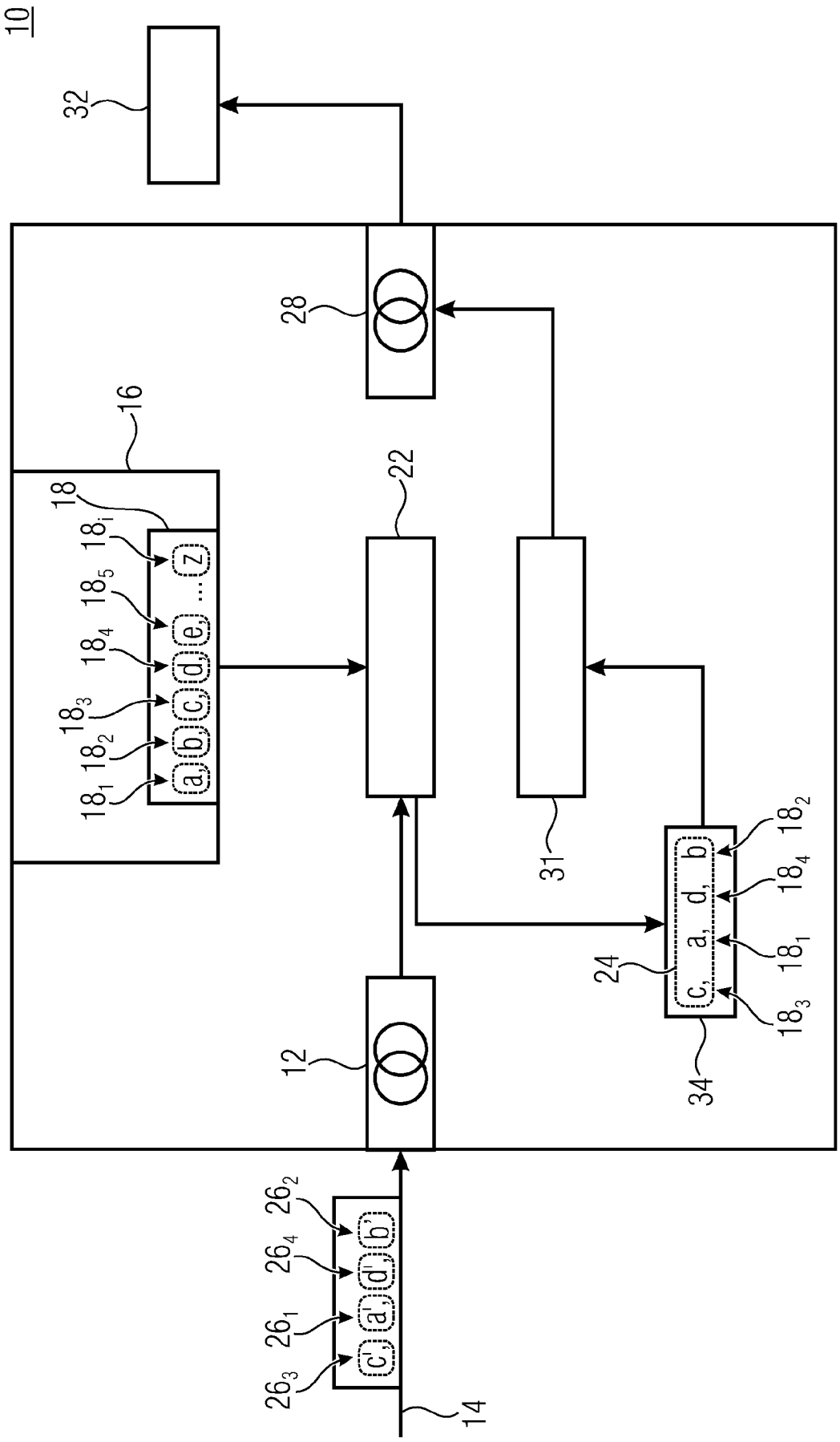
10
32
28
16
18
18_i
18_5
18_4
18_3
18_2
18_1
a b c d e ... z
22
31
24
c, a, d, b
18_2
18_4
18_1
18_3
34
12
26_2
26_4
26_1
26_3
c' a' d' b'
14

Fig. 1

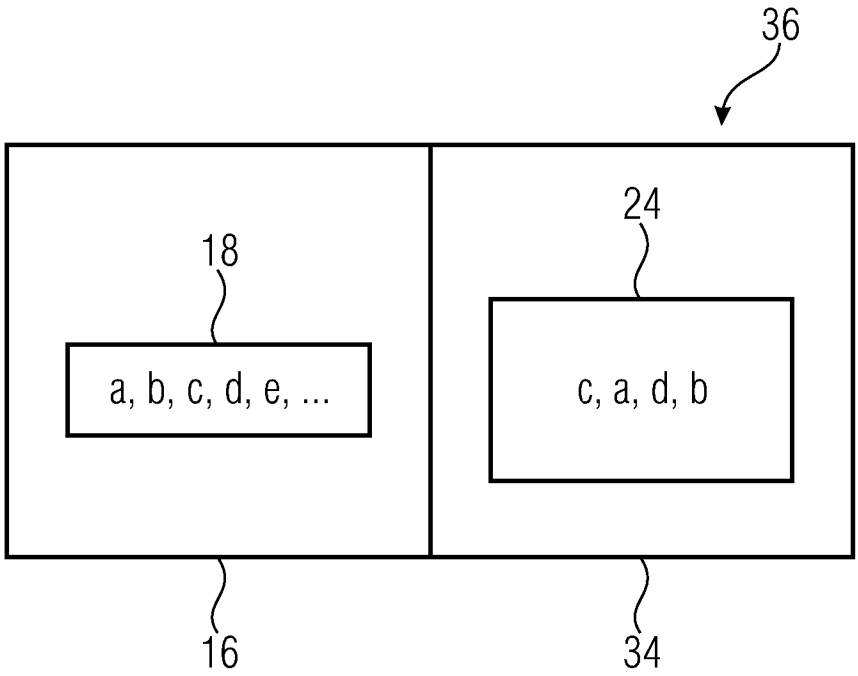

Fig. 2

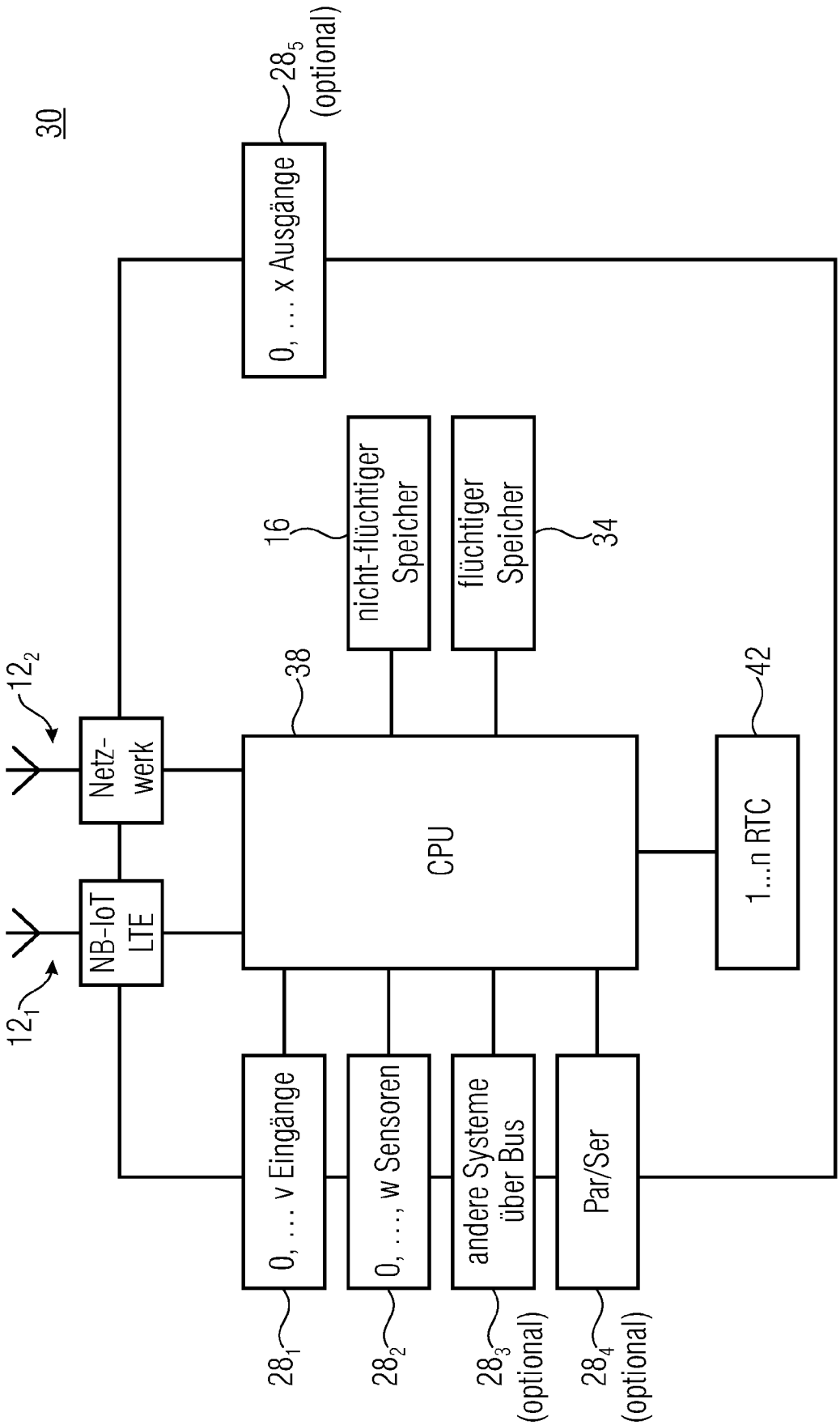
30
28_5 (optional)
0, ... x Ausgänge
16
nicht-flüchtiger Speicher
flüchtiger Speicher
34
12_2
38
42
Netz-werk
CPU
1...n RTC
12_1
NB-IoT LTE
0, ... v Eingänge
0, ..., w Sensoren
andere Systeme über Bus
Par/Ser
28_1
28_2
28_3 (optional)
28_4 (optional)

Fig. 3

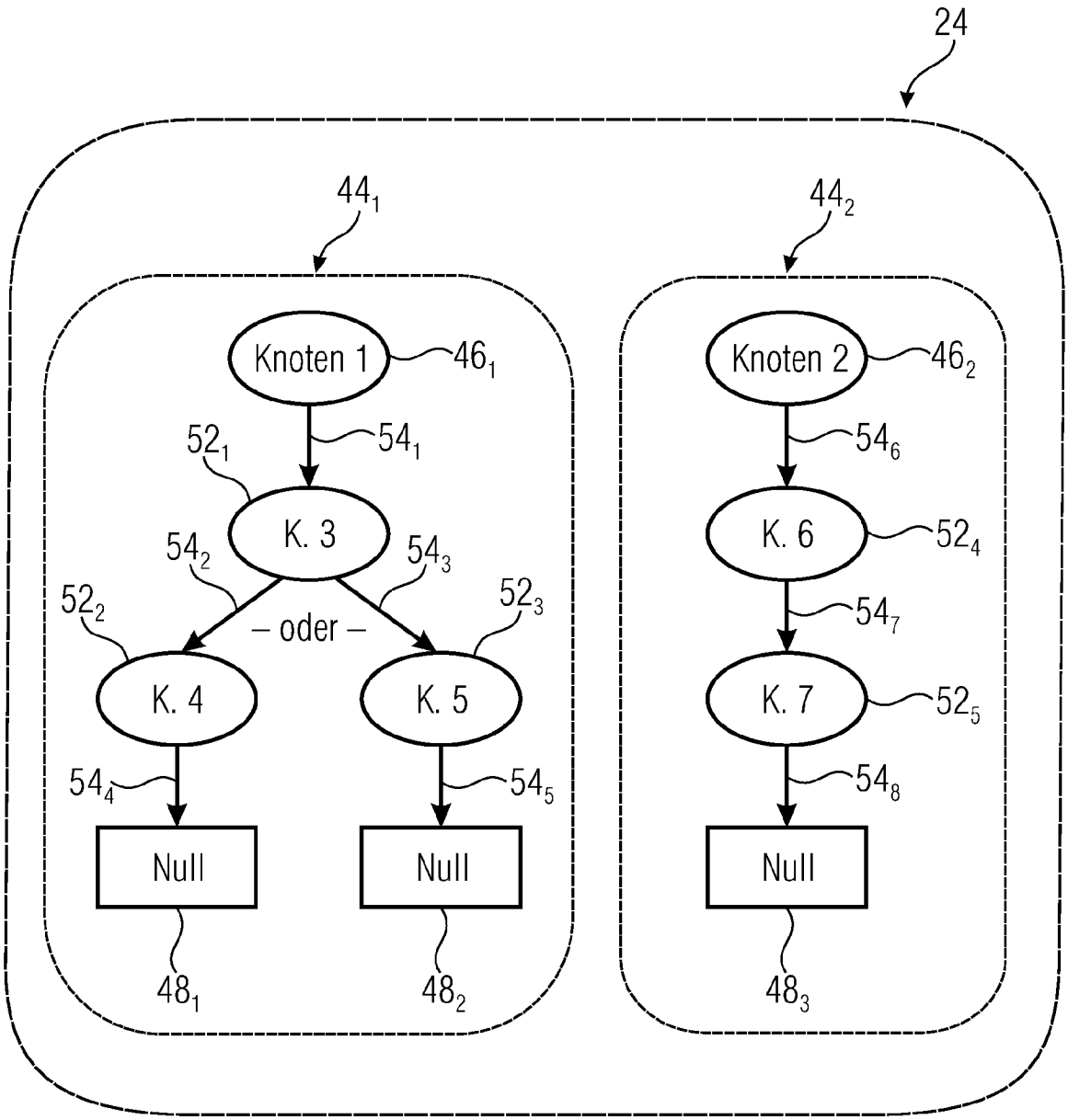
24
44 1
44 2
Knoten 1
46 1
52 1
54 1
K. 3
54 2
54 3
52 2
52 3
– oder –
K. 4
K. 5
54 4
54 5
Null
Null
48 1
48 2
Knoten 2
46 2
54 6
K. 6
52 4
54 7
K. 7
52 5
54 8
Null
48 3

Fig. 4

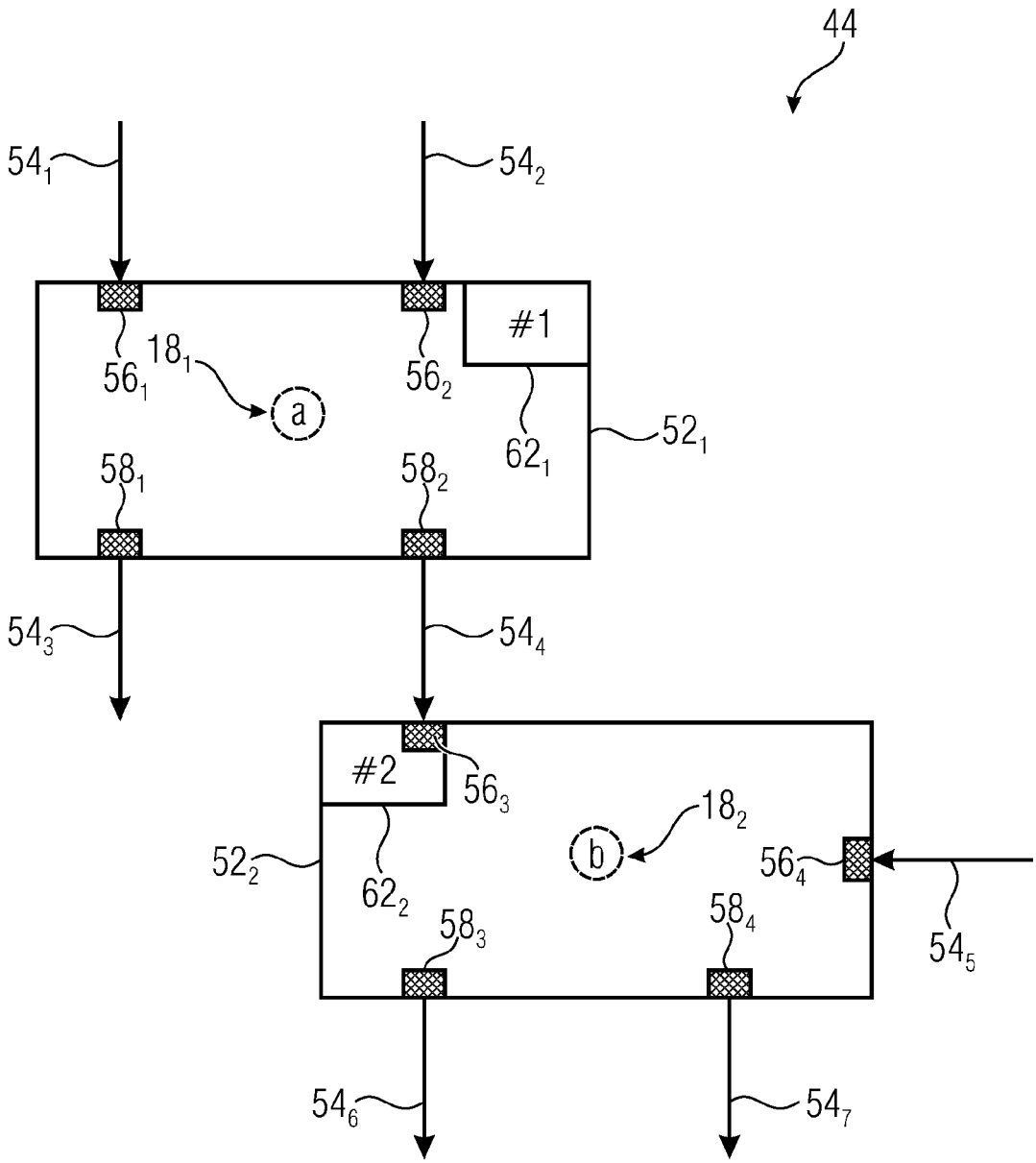
44
54_1
54_2
56_1
18_1
a
56_2
#1
62_1
52_1
58_1
58_2
54_3
54_4
#2
56_3
18_2
b
52_2
62_2
56_4
58_3
58_4
54_5
54_6
54_7

Fig. 5

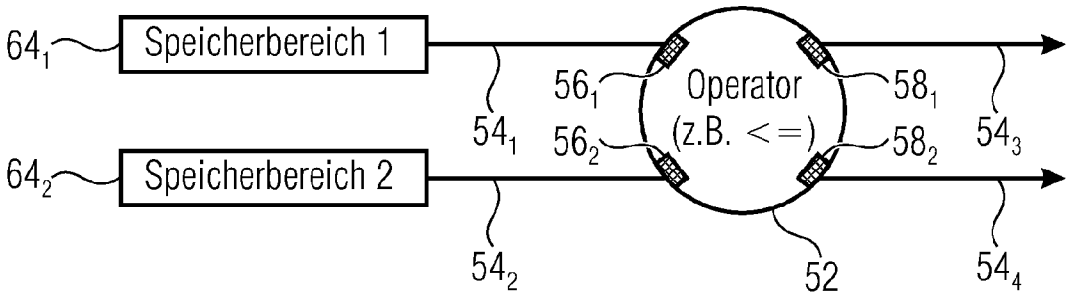

Fig. 6a

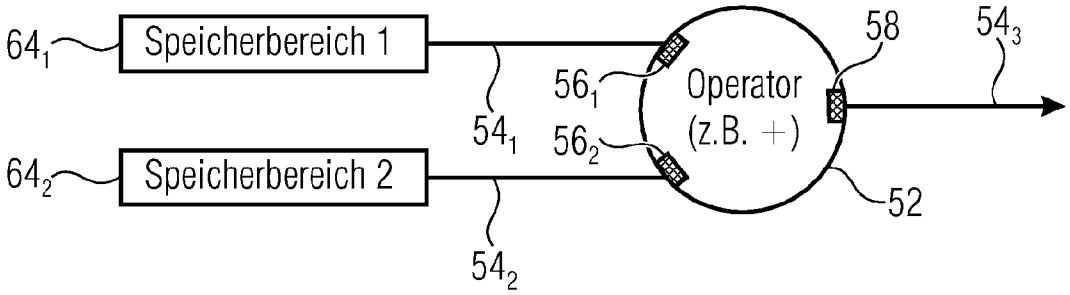

Fig. 6b

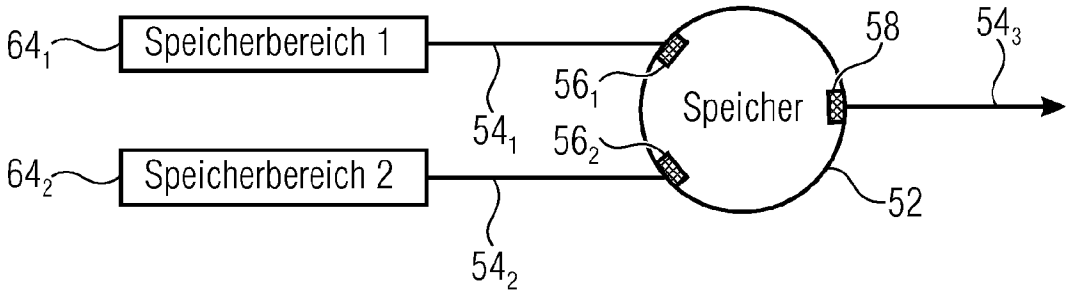

Fig. 6c

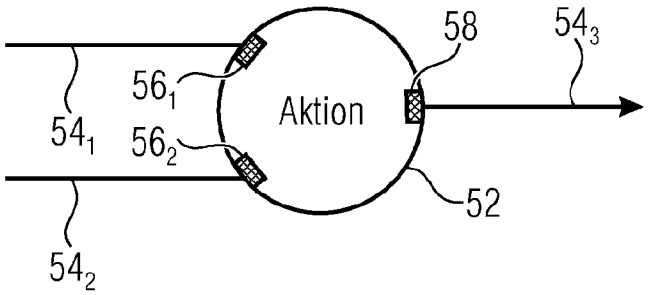

Fig. 6d

```
/* Definition der Typzahlen
   0: Nicht-Definierter Knoten
Arithmetik (a, b):
66_1 ->  1:    a + b          Addition
         2:    a - b          Substraktion
         3:    a * c          Multiplikation
         4:    a / c          Division
         5:    |a| + |b|      Absolut-Summe
         6:    |a| + b        Halbe Absolut-Summe
         7:    |a| - |b|      Absolut-Differenz
         8:    |a| - b        Halbe Absolut-Differenz
         9:    |a + b|        1-Norm-Summe
        10:    |a - b|        1-Norm-Differenz
        11:    |a * b|        Absolut-Multiplikation
        12:    |a| * b        Halb-Absolut-Multiplikation
        13:    |a / b|        Absolut-Division
        14:    |a| / b        Halb-Absolut-Division
66_15 -> 15:    a mod b        Modulo
Vergleich (a, b):
66_16 -> 16:    a == b         Gleich
         17:    a != b         Nicht-Gleich
         18:    a < b          Kleiner
         19:    a <= b         Kleiner-Gleich
         20:    |a| == |b|     Betrags-Gleich
         21:    |a| == b       Halb-Betrags-Gleich
         22:    |a| != |b|     Nicht-Betrags-Gleich
         23:    |a| != b       Betrag-Nicht-Gleich
         24:    |a| < |b|      Betrag-Kleiner-Betrag
         25:    |a| < b        Betrag-Kleiner
         26:    |a| <= |b|     Betrag-Kleiner-Gleich-Betrag
66_27 -> 27:    |a| <= b       Betrag-Kleiner-Gleich
Logik {a,b):
66_28 -> 28:    a && b         Logisches Und
         29:    a || b         Logisches Oder
         30:    a xor b        Logisches Exklusiv-Oder
         31:    a && !b        Einseitig Verneintes Und
         32:    a || !b        Einseitig Verneintes Oder
         33:    a xor !b       Einseitig Verneintes Exklusiv-Oder
         34:    !a && !b       Beidseitig Verneintes Und
         35:    !a || !b       Beidseitig Verneintes Oder
66_36 -> 36:    !a xor !b      Beidseitig Verneintes Exklusiv-Oder
Logik {a):
66_37 -> 37:    !a             Negation
         38:    a && 1         a == true
66_39 -> 39:    a && 0         a == false
```

Fig. 7

```
// -- Inhalt der conf_* Variablen kommt aus Funktion, welche die Funk-Nachricht entschlüsselt --
// Vergleichsknoten konfigurieren:
    uint16_t i ;
for ( i =0 ; i<used_knots; i++){
    // Typ setzen
    vertex[i]. typ = conf_typ[i] ;
    // Input 1 Verknüpfen
    if(conf_in1[i] < T_SEN){
        // Anderer Knoten
        vertex[i] .inp_1 = conf_in l[i] ;
    } else if(conf_in1[i] < T_AKT){
        // Sensor
        vertex[i].inp_1 =  conf_in1[i] ;
    } else if(conf_in1[i] < T_THR){
        // Aktionen
        sprintf((char *)ui8TxBuf, "Aktionen können nicht als Input gesetzt werden!\n“);
        SendString();
    } else if(conf_in1[i] < ( T_MAX )){
        // Schwellwerte
        vertex[i].inp_1 =  conf_in1[i] ;
    } else {
        sprintf((char *)ui8TxBuf, "Fehler bei Input 1. Wert: %d.\n", conf_in1[i] );
        SendString();
    }
    // Input 2 Verknüpfen
    if(conf_in2[i] < T_SEN){
        // Anderer Knoten
        vertex[i].inp_ 2 =  conf_in2[i] ;
    } else if(conf_in2[i] < T_AKT){
        // Sensor
        vertex[i].inp_2 = conf_in2[i] ;
    } else if(conf_in2[i] < T_THR){
        // Aktionen
        //v[i].inp_2 = &(a[ conf_in2[i] - T_AKT ]);
        sprintf((char *)ui8TxBuf, "Aktionen können nicht als Input gesetzt werden!\n");
        SendString();
    } else if(conf_in2[i] < ( T_MAX )){
        // Schwellwerte
        vertex[i].inp_2 = conf_in2[i] ; //&(var[ conf_in2[i] - T_THR ]);
```

Fig. 8

```
typedef struct {
    int16_t res; // Ergebnis des Knotens
    // Pointer für die Logik und den Aufbau der State-Machine:
    uint16_t inp_1; // Pointer auf Input-Variable
    uint16_t inp_2;
    uint16_t next_t; // Pointer "true"
    uint16_t next_f; // Pointer "false"
    //
    uint8_t typ; //Gibt an um was für einen Knoten es sich handelt
} Knoten;|
```

Fig. 9a

```
const uint8_t MAX_KNOTS   = 128;
const uint8_t MAX_VARS    = 128;
const uint8_t N_SENSOREN = 10;
const uint8_t N_AKTIONEN   = 28;
const uint8_t N_First_Layer  = 32;

const uint16_t NULL_ID      =INT16_MAX;

const uint8_t IS_ARI        = 16;
const uint8_t IS_VGL        = 32;
```

Fig. 9b

```
// Alles kleiner T_SEN sind Knoten
const uint16_t T_SEN = MAX_KNOTS;
// Alles >= T_SEN und < T_AKT sind Sensoren
const uint16_t T_AKT  = T_SEN + N_SENSOREN;
// Alles >= T_AKT  und < T_THR sind Aktionenen
const uint16_t T_THR = T_AKT + N_AKTIONEN;
// Alles >= T_THR sind Variablen
const uint16_t T_MAX = T_THR + MAX_VARS; // Maximal mögliche Zahl
```

Fig. 9c

```
1   // Liste an Knoten hat jetzt immer ein feste Länge:
2   Knoten v[MAX_KNOTS];
3
4   // Liste an Aktionen
5   Knoten a[N_AKTIONEN];
6
7   // Liste von Knoten im ersten Layer
8   Knoten* fl[N_First_Layer];
9
10  // Variablen (Schwellwerte, Zwischenspeicher)
11  int16_t var[MAX_VARS];
12
13  // Sensor-Werte
14  int16_t sen[N_SENSOREN];
15
```

Fig. 10a

```
1   for(uint8_t i=0; i<N_ACTIVE_FL; i++)
2       {
3           uint8_t i = 0;
4           while(fl[i] !=0 ) {
5               doKnot(fl[i]);
6               i++;
7           }
8   }
```

Fig. 10b

```
1   while(k != 0){// && iter <MAX_KNOTS){
2           iter++;
3           if(k ->typ == 0){
4                   // Leerer Knoten
5                   return; // Sofort raus hier!
6           }
7           if(k ->typ >= 64){      // Ab 64 gehen ie Aktionen los
8                   aktion(k ->typ);   // Führe die Aktion aus
9                   //k = k ->next_t; // idR. Nullpointer
10                  k = getPointer(k ->next_t);
11                  continue;
12          } else { // Kein Aktionsknoten
13                  // Wenn pointer valid, dereferenziere und setze wert
14                  wert1P = getInput(k ->inp_1);
15                  wert2P = getInput(k ->inp_2);
16                  wert1 = (wert1P ? *(wert1P) : 0);
17                  if( k->typ== 38) // Speicherknoten sonderfall
18                  {
19                          k ->res = wert1;
20                          // Nur wenn Pointer != 0 speichern
21                          if(wert2P)
22                                  *(wert2P) = wert1; // ganz gefährlich ;D
23
24                          k = getPointer( k->next_t);
25                          continue; // Sofort nächster Knoten!
26                  }
27                  // Da der Speicherknoten wert2 nicht dereferenziert
28                  // Kommt die Abfrage vor der eintlichen Abfrage des Types
29                  wert2 = (wert2P ? *(wert2P) : 0);|
30                  switch( k ->typ ){
```

Fig. 10c

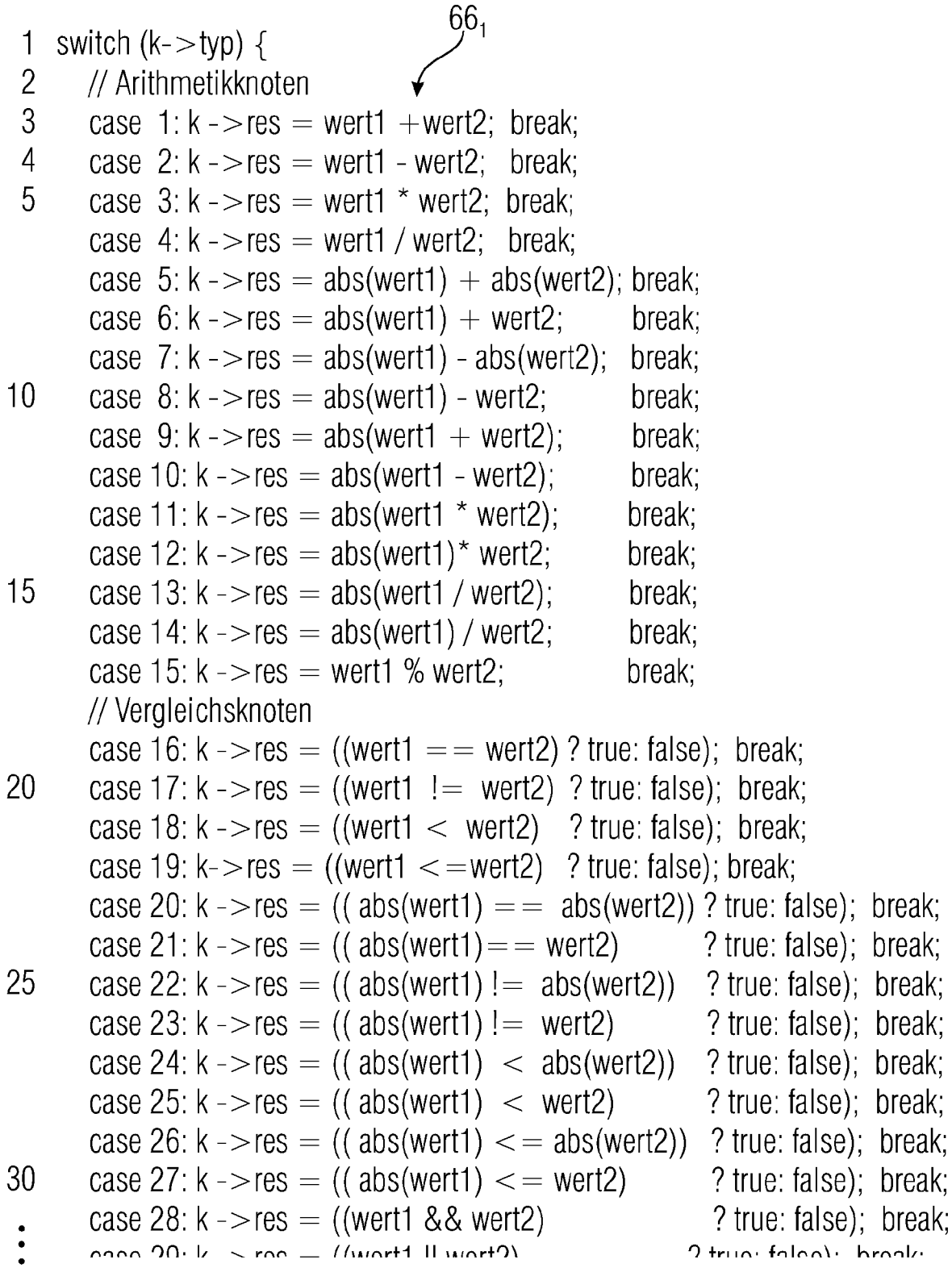
66₁
1 switch (k->typ) {
2 // Arithmetikknoten
3 case 1: k ->res = wert1 +wert2; break;
4 case 2: k ->res = wert1 - wert2; break;
5 case 3: k ->res = wert1 * wert2; break;
case 4: k ->res = wert1 / wert2; break;
case 5: k ->res = abs(wert1) + abs(wert2); break;
case 6: k ->res = abs(wert1) + wert2; break;
case 7: k ->res = abs(wert1) - abs(wert2); break;
10 case 8: k ->res = abs(wert1) - wert2; break;
case 9: k ->res = abs(wert1 + wert2); break;
case 10: k ->res = abs(wert1 - wert2); break;
case 11: k ->res = abs(wert1 * wert2); break;
case 12: k ->res = abs(wert1)* wert2; break;
15 case 13: k ->res = abs(wert1 / wert2); break;
case 14: k ->res = abs(wert1) / wert2; break;
case 15: k ->res = wert1 % wert2; break;
// Vergleichsknoten
case 16: k ->res = ((wert1 == wert2) ? true: false); break;
20 case 17: k ->res = ((wert1 != wert2) ? true: false); break;
case 18: k ->res = ((wert1 < wert2) ? true: false); break;
case 19: k->res = ((wert1 <=wert2) ? true: false); break;
case 20: k ->res = ((abs(wert1) == abs(wert2)) ? true: false); break;
case 21: k ->res = ((abs(wert1)== wert2) ? true: false); break;
25 case 22: k ->res = ((abs(wert1) != abs(wert2)) ? true: false); break;
case 23: k ->res = ((abs(wert1) != wert2) ? true: false); break;
case 24: k ->res = ((abs(wert1) < abs(wert2)) ? true: false); break;
case 25: k ->res = ((abs(wert1) < wert2) ? true: false); break;
case 26: k ->res = ((abs(wert1) <= abs(wert2)) ? true: false); break;
30 case 27: k ->res = ((abs(wert1) <= wert2) ? true: false); break;
case 28: k ->res = ((wert1 && wert2) ? true: false); break;

Fig. 10d

```
1       // Abfrage auf Knotentyp zur weiteren Auswertung
2       if( k ->typ < IS_ARI){
3           // Zum nächsten Knoten Springen:
4           k = getPointer(k ->next_t);
5           continue;
6       // Sonst Vgl.-Knoten
7       } else if(k ->typ < IS_VGL) {
8           if(k ->res == true){
9               k = getPointer(k ->next_t);
10              continue;
11          } else {
12              k = getPointer(k ->next_f);
13              continue;
14          } // k ->res == true - else
15      } //  k ->typ < IS_VGL
16  } // else k ->typ >= 64
```

Fig. 10e

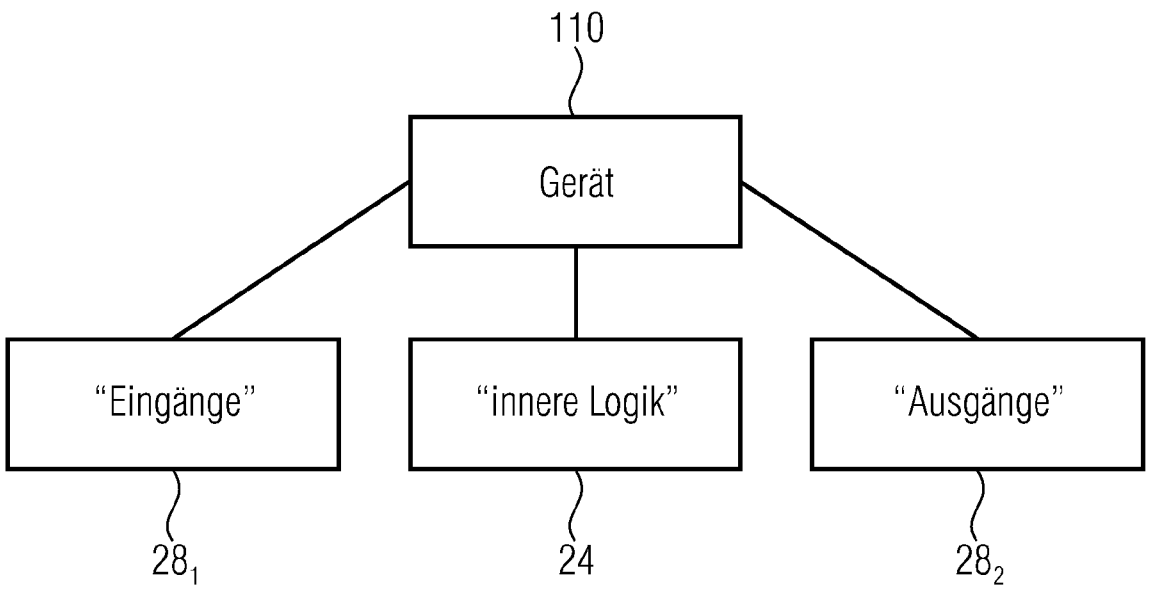

Fig. 11

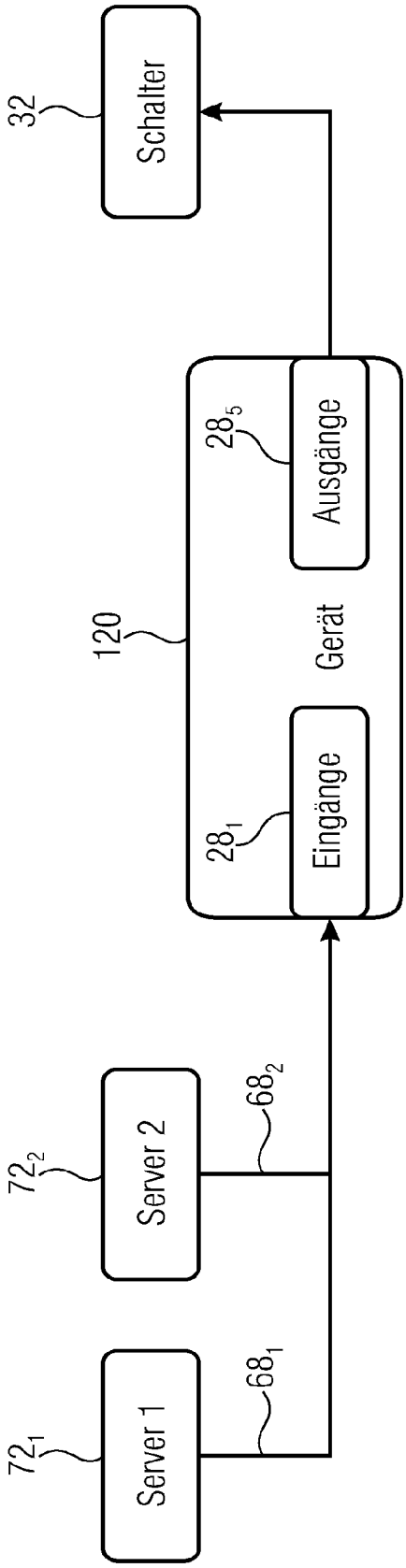
Schalter
32
Gerät
120
Ausgänge
28_5
Eingänge
28_1
Server 1
72_1
Server 2
72_2
68_1
68_2

Fig. 12

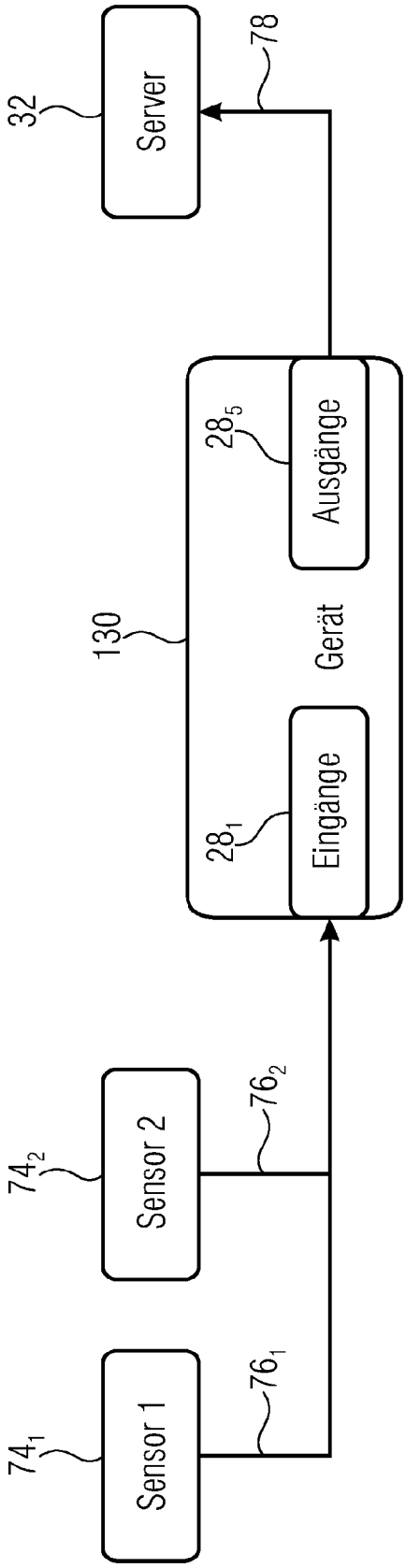
Server
78
32
Gerät
130
Ausgänge
28_5
Eingänge
28_1
Sensor 1
74_1
Sensor 2
74_2
76_1
76_2

Fig. 13

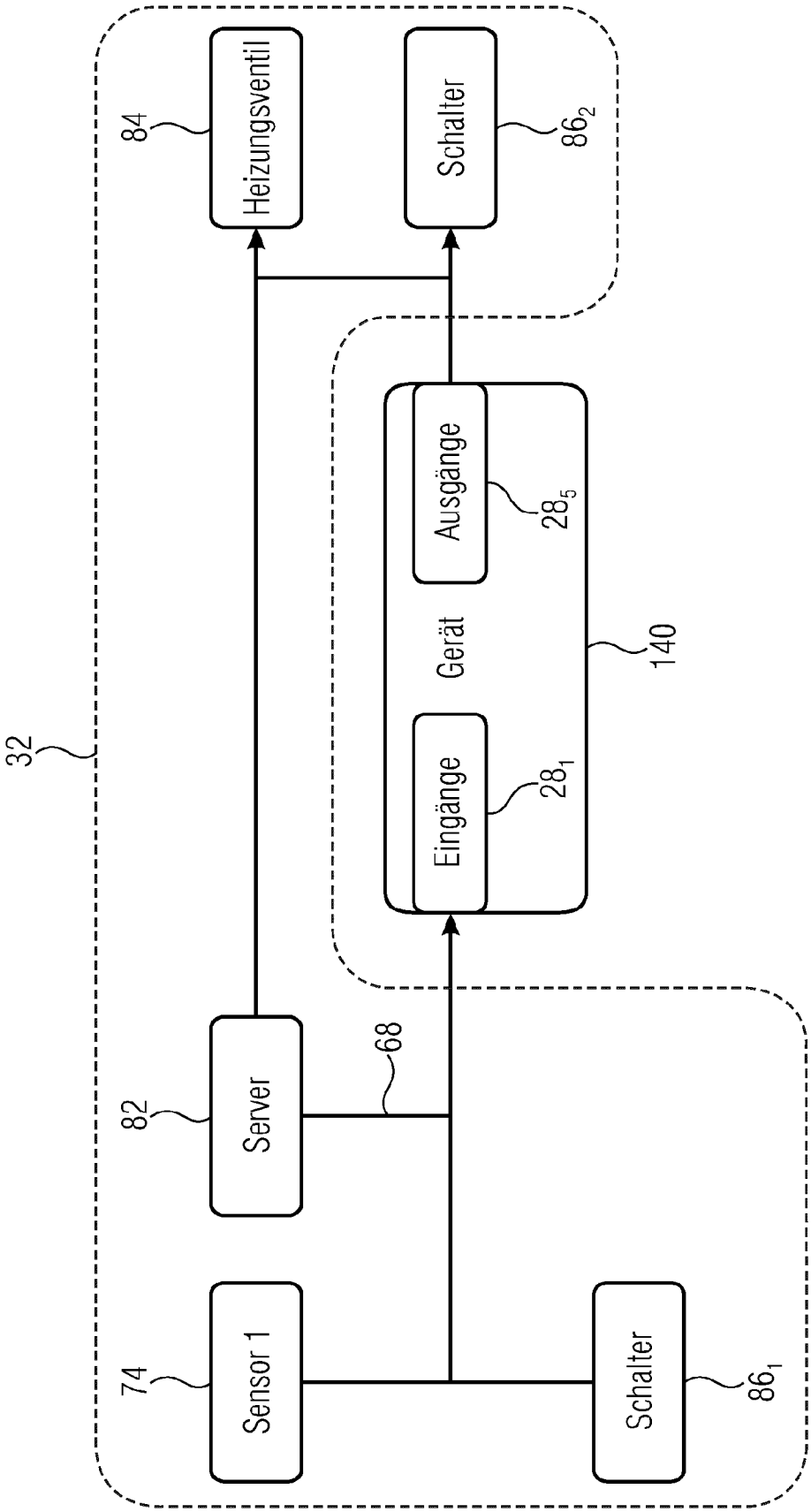
32
84
Heizungsventil
Schalter
86_2
Ausgänge
28_5
Gerät
140
Eingänge
28_1
68
82
Server
74
Sensor 1
Schalter
86_1

Fig. 14

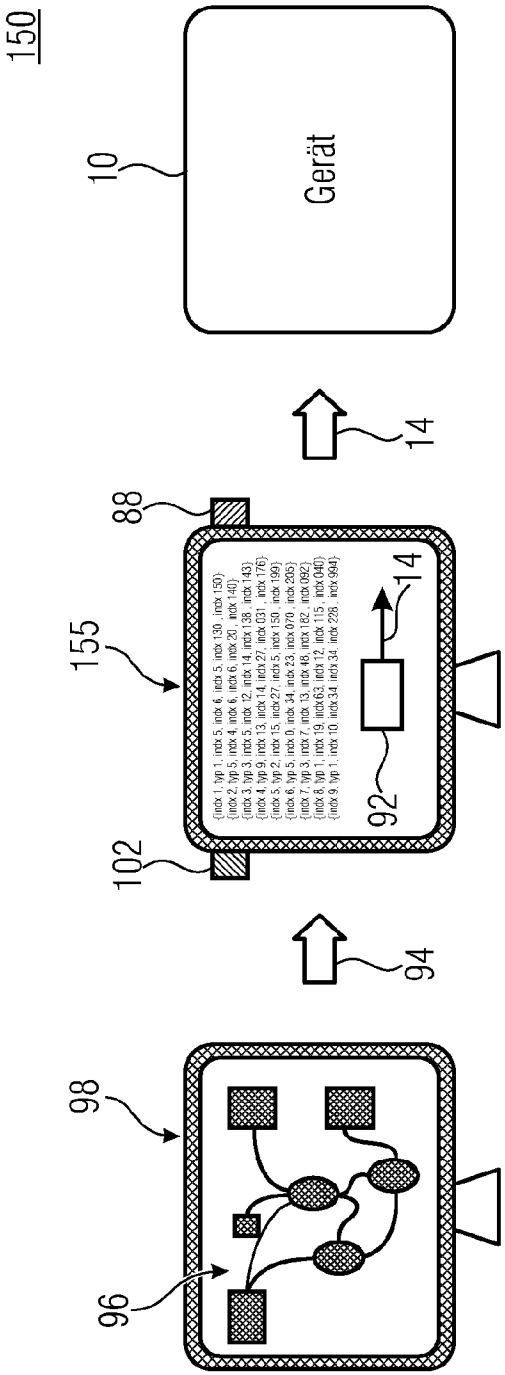

Fig. 15

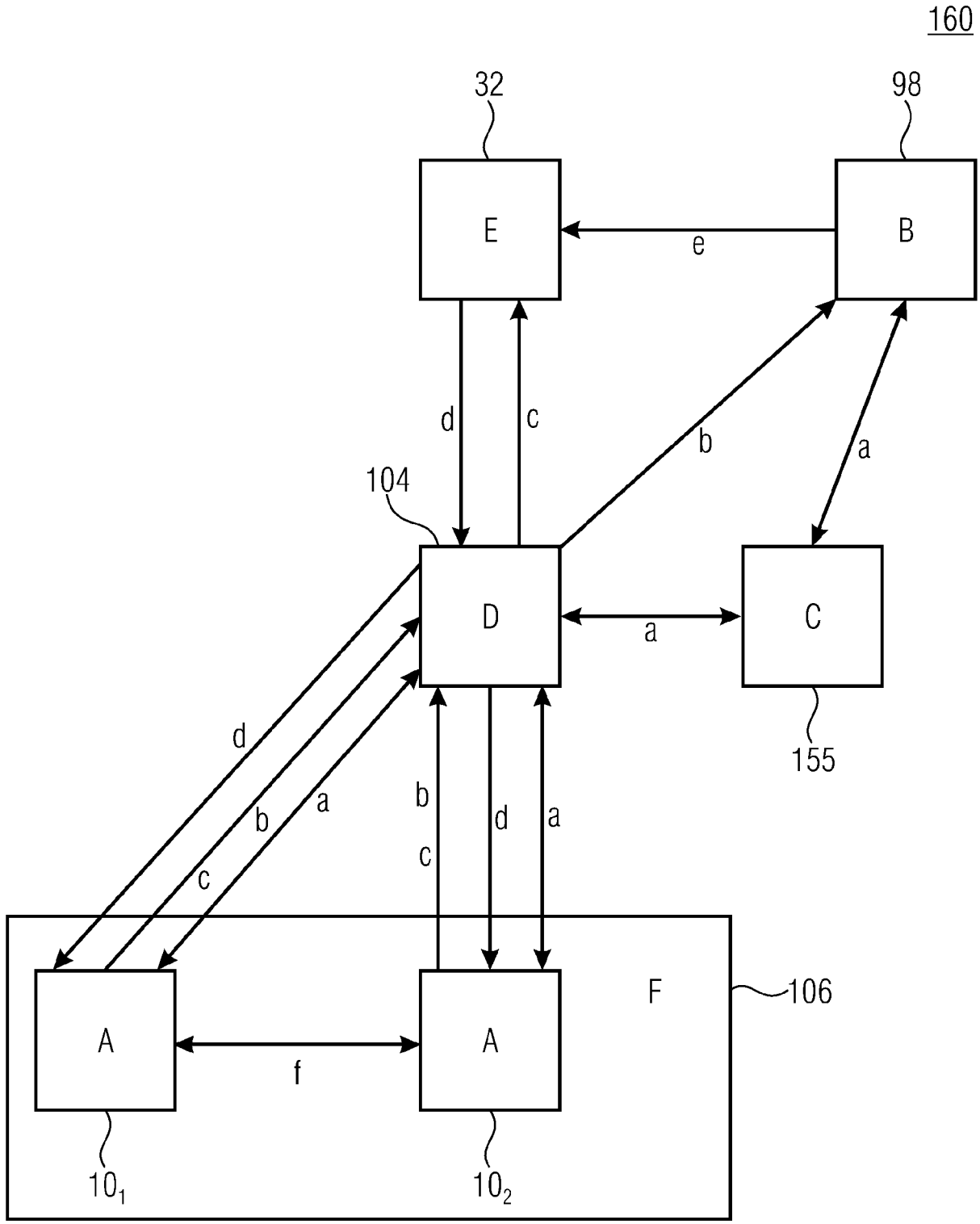

Fig. 16

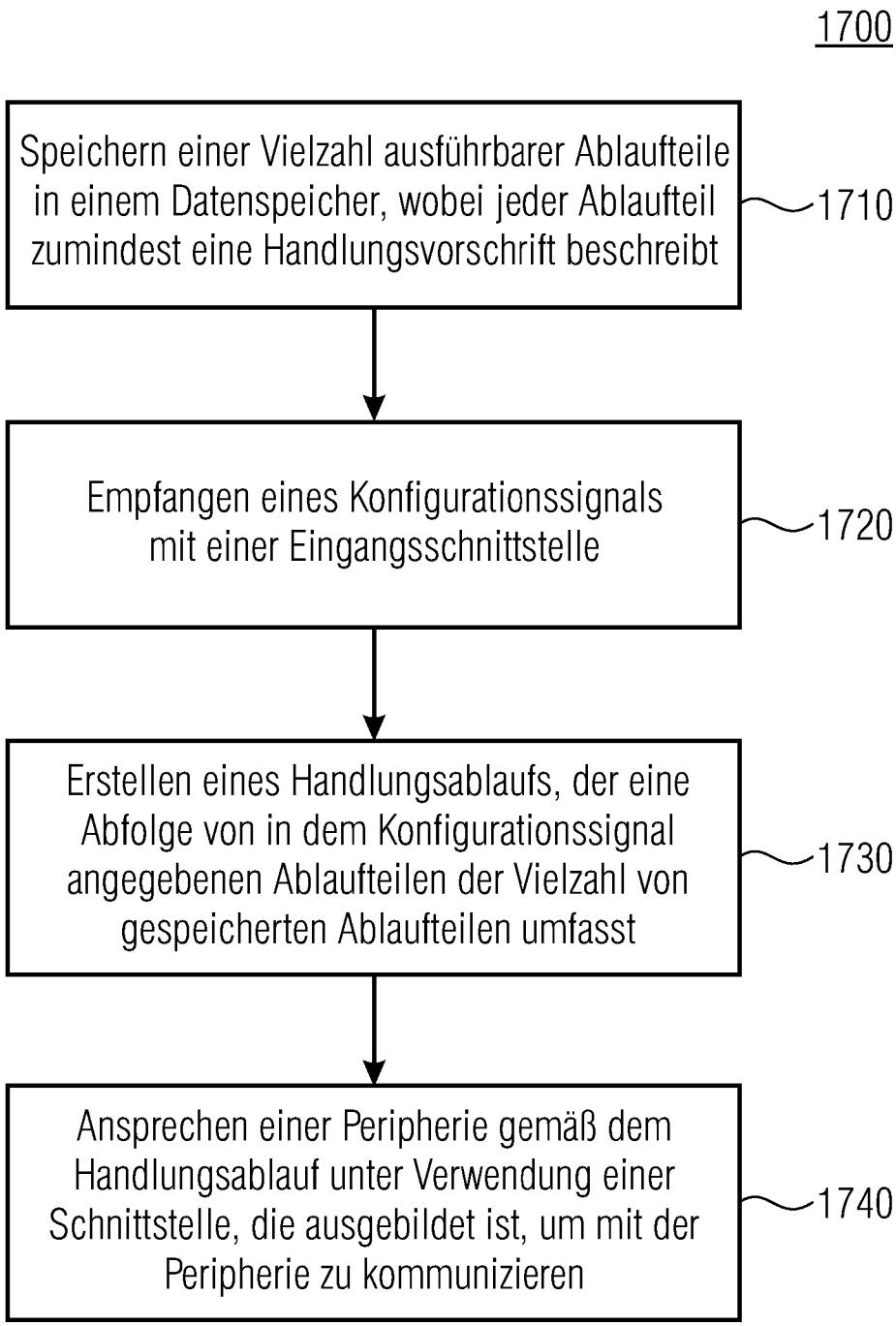
1700
Speichern einer Vielzahl ausführbarer Ablaufteile in einem Datenspeicher, wobei jeder Ablaufteil zumindest eine Handlungsvorschrift beschreibt
1710
Empfangen eines Konfigurationssignals mit einer Eingangsschnittstelle
1720
Erstellen eines Handlungsablaufs, der eine Abfolge von in dem Konfigurationssignal angegebenen Ablaufteilen der Vielzahl von gespeicherten Ablaufteilen umfasst
1730
Ansprechen einer Peripherie gemäß dem Handlungsablauf unter Verwendung einer Schnittstelle, die ausgebildet ist, um mit der Peripherie zu kommunizieren
1740

Fig. 17

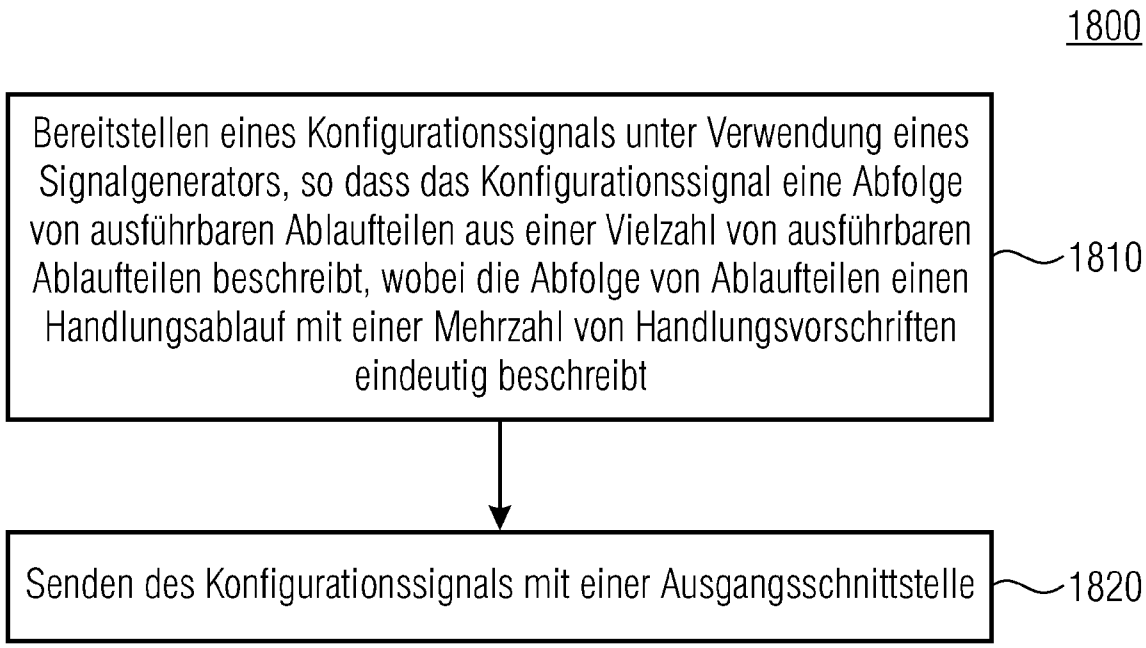
1800
Bereitstellen eines Konfigurationssignals unter Verwendung eines Signalgenerators, so dass das Konfigurationssignal eine Abfolge von ausführbaren Ablaufteilen aus einer Vielzahl von ausführbaren Ablaufteilen beschreibt, wobei die Abfolge von Ablaufteilen einen Handlungsablauf mit einer Mehrzahl von Handlungsvorschriften eindeutig beschreibt
1810
Senden des Konfigurationssignals mit einer Ausgangsschnittstelle
1820

Fig. 18

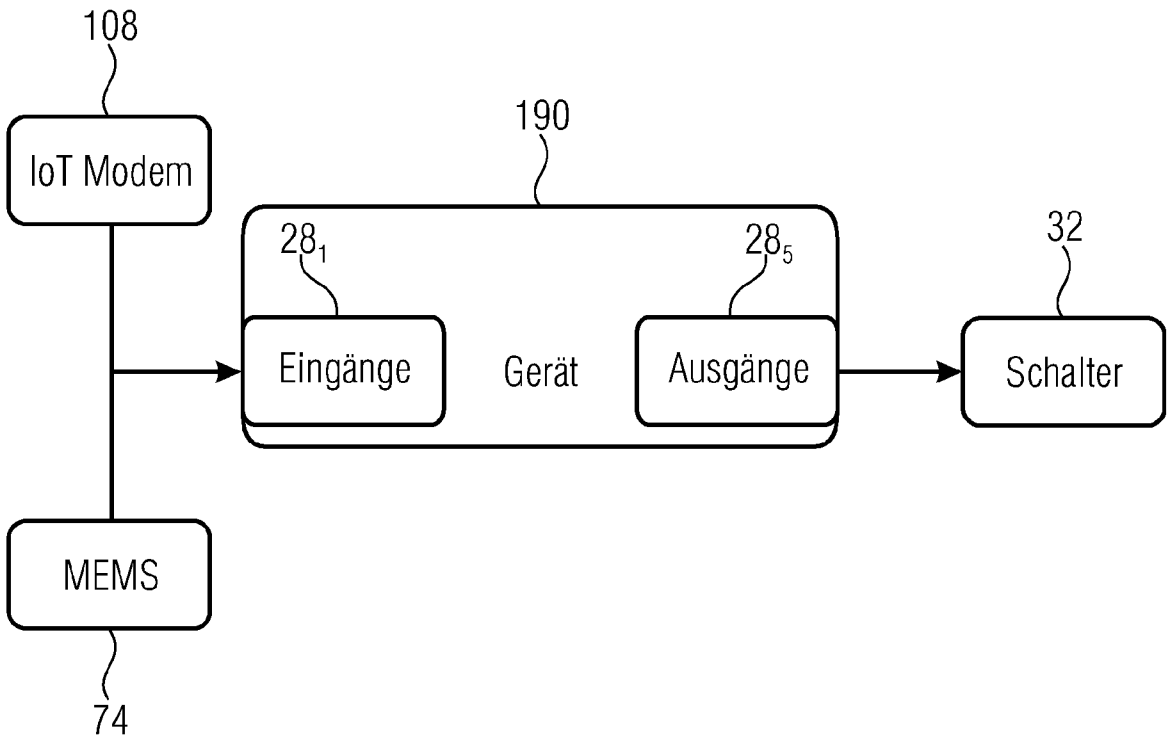
108
IoT Modem
190
28_1
28_5
32
Eingänge
Gerät
Ausgänge
Schalter
MEMS
74

Fig. 19

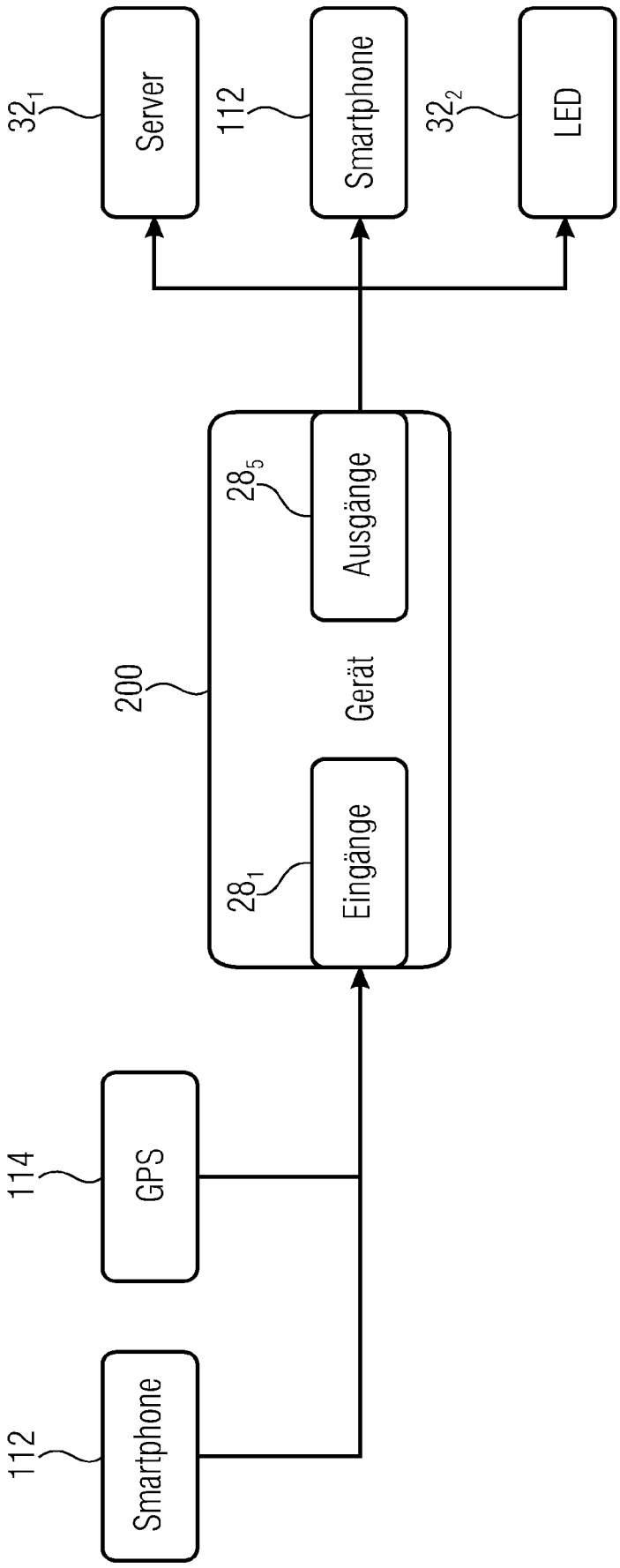
Server
Smartphone
LED
Ausgänge
Gerät
Eingänge
GPS
Smartphone
32_1
112
32_2
28_5
200
28_1
114
112

Fig. 20

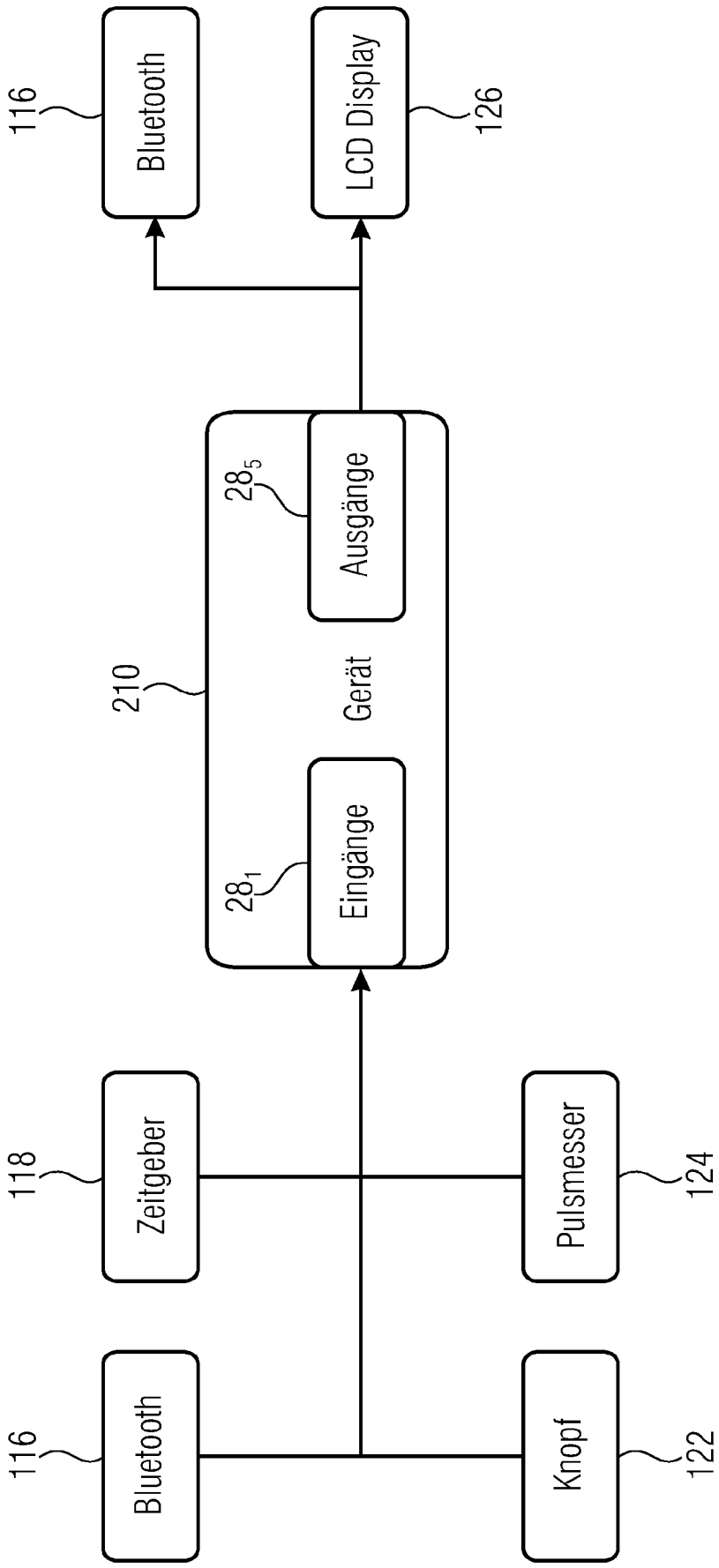
116
Bluetooth
126
LCD Display
210
Gerät
28_5
Ausgänge
28_1
Eingänge
118
Zeitgeber
124
Pulsmesser
116
Bluetooth
122
Knopf

Fig. 21

INTERNATIONAL SEARCH REPORT

International application No.
PCT/EP2020/057130

A. CLASSIFICATION OF SUBJECT MATTER

G06F 8/34(2018.01)i; ***G05B 19/05***(2006.01)i; ***G06F 9/50***(2006.01)i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06F; G05B

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

EPO-Internal, WPI Data, IBM-TDB

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	EP 2357541 A1 (ROCKWELL AUTOMATION TECH INC [US]) 17 August 2011 (2011-08-17) figures 2-6 paragraph [0043] - paragraph [0056] paragraph [0104] - paragraph [0105]	1-33
X	EP 1315055 A2 (ROCKWELL AUTOMATION TECH INC [US]) 28 May 2003 (2003-05-28) paragraph [0002] - paragraph [0009] paragraph [0012] - paragraph [0016] paragraph [0040] - paragraph [0043] paragraph [0050]	1-33
X	EP 2790076 A1 (SIEMENS AG [DE]) 15 October 2014 (2014-10-15) paragraph [0003] - paragraph [0016]	1-33

☐ Further documents are listed in the continuation of Box C. ☑ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
02 June 2020	**12 June 2020**

Name and mailing address of the ISA/EP	Authorized officer
European Patent Office **p.b. 5818, Patentlaan 2, 2280 HV Rijswijk** **Netherlands** Telephone No. **(+31-70)340-2040** Facsimile No. **(+31-70)340-3016**	**Milasinovic, Goran** Telephone No.

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/EP2020/057130

Patent document cited in search report	Publication date (day/month/year)	Patent family member(s)	Publication date (day/month/year)
EP 2357541 A1	17 August 2011	EP 2357541 A1	17 August 2011
		US 2011202688 A1	18 August 2011
EP 1315055 A2	28 May 2003	DE 60219575 T2	27 December 2007
		EP 1315055 A2	28 May 2003
		US 2003100958 A1	29 May 2003
EP 2790076 A1	15 October 2014	NONE	

INTERNATIONALER RECHERCHENBERICHT

Internationales Aktenzeichen
PCT/EP2020/057130

A. KLASSIFIZIERUNG DES ANMELDUNGSGEGENSTANDES
INV. G06F8/34 G05B19/05 G06F9/50
ADD.

Nach der Internationalen Patentklassifikation (IPC) oder nach der nationalen Klassifikation und der IPC

B. RECHERCHIERTE GEBIETE

Recherchierter Mindestprüfstoff (Klassifikationssystem und Klassifikationssymbole)
G06F G05B

Recherchierte, aber nicht zum Mindestprüfstoff gehörende Veröffentlichungen, soweit diese unter die recherchierten Gebiete fallen

Während der internationalen Recherche konsultierte elektronische Datenbank (Name der Datenbank und evtl. verwendete Suchbegriffe)
EPO-Internal, WPI Data, IBM-TDB

C. ALS WESENTLICH ANGESEHENE UNTERLAGEN

Kategorie*	Bezeichnung der Veröffentlichung, soweit erforderlich unter Angabe der in Betracht kommenden Teile	Betr. Anspruch Nr.
X	EP 2 357 541 A1 (ROCKWELL AUTOMATION TECH INC [US]) 17. August 2011 (2011-08-17) Abbildungen 2-6 Absatz [0043] - Absatz [0056] Absatz [0104] - Absatz [0105] -----	1-33
X	EP 1 315 055 A2 (ROCKWELL AUTOMATION TECH INC [US]) 28. Mai 2003 (2003-05-28) Absatz [0002] - Absatz [0009] Absatz [0012] - Absatz [0016] Absatz [0040] - Absatz [0043] Absatz [0050] -----	1-33
X	EP 2 790 076 A1 (SIEMENS AG [DE]) 15. Oktober 2014 (2014-10-15) Absatz [0003] - Absatz [0016] -----	1-33

☐ Weitere Veröffentlichungen sind der Fortsetzung von Feld C zu entnehmen ☒ Siehe Anhang Patentfamilie

* Besondere Kategorien von angegebenen Veröffentlichungen :
"A" Veröffentlichung, die den allgemeinen Stand der Technik definiert, aber nicht als besonders bedeutsam anzusehen ist
"E" frühere Anmeldung oder Patent, die bzw. das jedoch erst am oder nach dem internationalen Anmeldedatum veröffentlicht worden ist
"L" Veröffentlichung, die geeignet ist, einen Prioritätsanspruch zweifelhaft erscheinen zu lassen, oder durch die das Veröffentlichungsdatum einer anderen im Recherchenbericht genannten Veröffentlichung belegt werden soll oder die aus einem anderen besonderen Grund angegeben ist (wie ausgeführt)
"O" Veröffentlichung, die sich auf eine mündliche Offenbarung, eine Benutzung, eine Ausstellung oder andere Maßnahmen bezieht
"P" Veröffentlichung, die vor dem internationalen Anmeldedatum, aber nach dem beanspruchten Prioritätsdatum veröffentlicht worden ist
"T" Spätere Veröffentlichung, die nach dem internationalen Anmeldedatum oder dem Prioritätsdatum veröffentlicht worden ist und mit der Anmeldung nicht kollidiert, sondern nur zum Verständnis des der Erfindung zugrundeliegenden Prinzips oder der ihr zugrundeliegenden Theorie angegeben ist
"X" Veröffentlichung von besonderer Bedeutung; die beanspruchte Erfindung kann allein aufgrund dieser Veröffentlichung nicht als neu oder auf erfinderischer Tätigkeit beruhend betrachtet werden
"Y" Veröffentlichung von besonderer Bedeutung; die beanspruchte Erfindung kann nicht als auf erfinderischer Tätigkeit beruhend betrachtet werden, wenn die Veröffentlichung mit einer oder mehreren Veröffentlichungen dieser Kategorie in Verbindung gebracht wird und diese Verbindung für einen Fachmann naheliegend ist
"&" Veröffentlichung, die Mitglied derselben Patentfamilie ist

Datum des Abschlusses der internationalen Recherche	Absendedatum des internationalen Recherchenberichts
2. Juni 2020	12/06/2020

Name und Postanschrift der Internationalen Recherchenbehörde	Bevollmächtigter Bediensteter
Europäisches Patentamt, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Milasinovic, Goran

1

INTERNATIONALER RECHERCHENBERICHT

Angaben zu Veröffentlichungen, die zur selben Patentfamilie gehören

Internationales Aktenzeichen
PCT/EP2020/057130

Im Recherchenbericht angeführtes Patentdokument	Datum der Veröffentlichung	Mitglied(er) der Patentfamilie	Datum der Veröffentlichung
EP 2357541 A1	17-08-2011	EP 2357541 A1	17-08-2011
		US 2011202688 A1	18-08-2011
EP 1315055 A2	28-05-2003	DE 60219575 T2	27-12-2007
		EP 1315055 A2	28-05-2003
		US 2003100958 A1	29-05-2003
EP 2790076 A1	15-10-2014	KEINE	