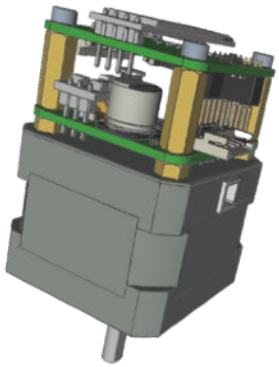
# Richtlinie zum „Software-Sprint“

# Ethersweep

## *Schlussbericht*

Zuwendungsempfänger: Jan Neumann

Das diesem Bericht zugrundeliegende Vorhaben wurde mit Mitteln des Bundesministeriums für Bildung und Forschung unter dem Förderkennzeichen 01IS23S28 gefördert. Die Verantwortung für den Inhalt dieser Veröffentlichung liegt beim Autor.



ethersweep

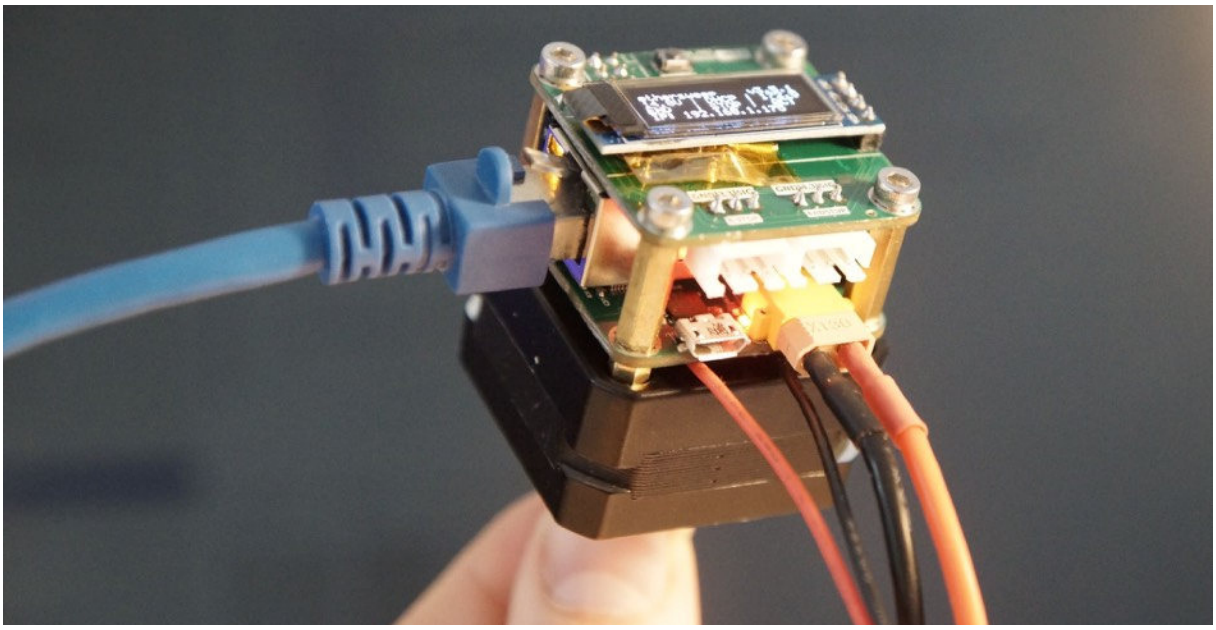
*easy motion control*

## Kurze Darstellung der Aufgabenstellung und Motivation

*Was war Deine Motivation? Welches Problem wolltest Du mit Deinem Projekt lösen? Wie war die geplante Vorgehensweise zur Problemlösung (auch Angabe der wichtigsten Meilensteine)?*

Schrittmotoren sind weit verbreitete, hochgenaue Motoren, welche in einer Vielzahl von Automatisierungsprojekten, Maschinen und Geräten verbaut sind. Die elektronische und softwareseitige Ansteuerung ist jedoch verhältnismäßig aufwändig. Durch den erhöhten Aufwand der Ansteuerung, ist die Integration kostenaufwändiger und zeitintensiver. Es sind zwar kommerzielle Ansteuerungen verfügbar, welche über spezielle Netzwerke angesteuert werden können, doch eine Open-Source Lösung gab es bisher nicht.

Ethersweep soll genau dieses Problem lösen, indem eine Schrittmotorsteuerung entwickelt wird, welche über standard Ethernet mit standard Netzwerkkomponenten (Switches, Hubs, RJ45, LAN, DHCP) angesteuert werden können. Das erleichtert die Integration von diesen hochgenauen Schrittmotoren in Automatisierungsprojekten und Prototypenaufbauten, indem herkömmliche Computer und Netzwerkclients zur Steuerung verwendet werden können. Durch ein einfaches, weit verbreitetes JSON und UDP Interface, was nahezu jede entwickelnde Person verstehen sollte, lassen sich Motorsteuerungen auch von nicht Elektrotechnisch versierten Personen verwenden und steuern. Ethersweep ist ein Bauteil, welches in vielen Bereichen der elektronischen Bewegungssteuerungen verwendet werden kann und erweitert und erleichtert die Verwendung eben dieser.



*Zusammengebauter Ethersweep Controller auf flachem Schittmotor aufgeschraubt*

## Beitrag des Projektes zu den Zielen der Förderinitative „Software-Sprint“

*Welche Bezüge gibt es zu den Themenfeldern „Civic Tech“ und „Data Literacy“ des Software Sprints oder zu weiteren gesellschaftlich relevanten Zielen bzw. Lösungsansätzen?*

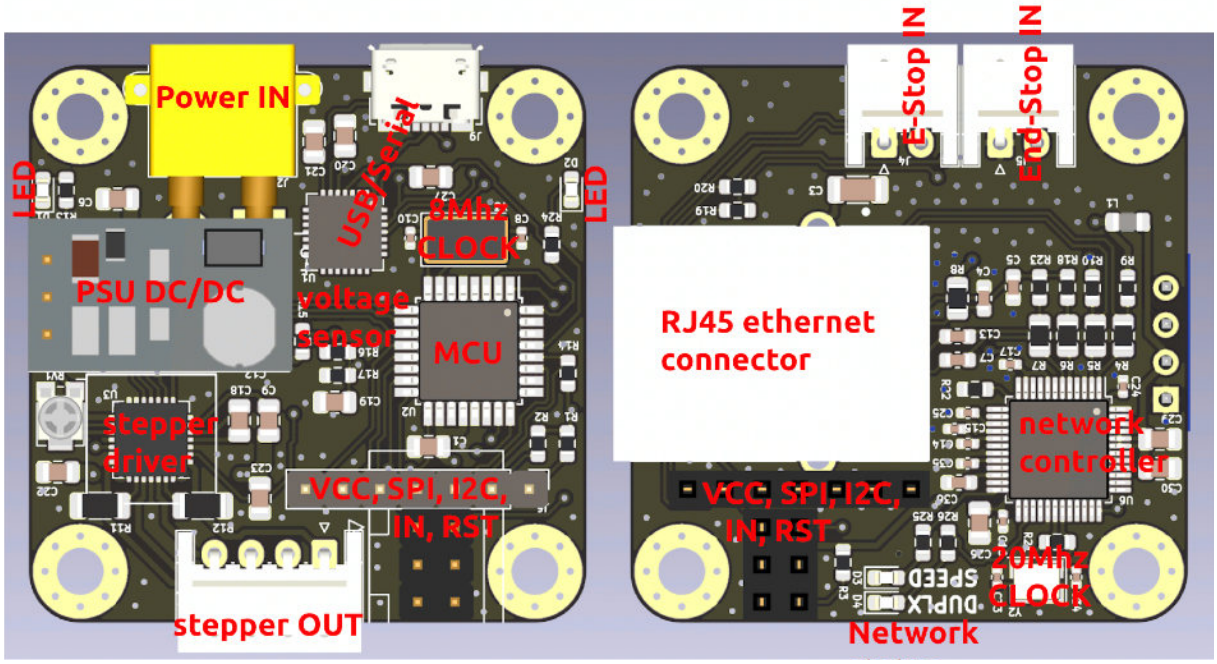
Durch den geringen Anteil an Open-Source im Feld der Automatisierung, sind EntwicklerInnen bislang vermehrt auf kommerzielle und geschlossene Systeme angewiesen. Eine vergleichbare Lösung zu Ethersweep liegt derzeit bei einem 5-10 Fachen des Preises eines Ethersweep controllers. Dadurch bleibt dieser Bereich vielen EntwicklerInnen, KünstlerInnen, Forschenden, BastlerInnen aber auch kleineren Firmen verschlossen oder erschwert den Einstieg erheblich. Ethersweep soll einen Teil dazu beitragen, ein offenes, veränderbares und kostengünstige Konkurrenz aufzubauen, welche von den AnwenderInnen selbst hergestellt und weiterentwickelt werden kann. Dadurch, dass die Software und Hardware zur freien Verfügung steht, können eigene Funktionen mit Relativ wenig Aufwand umgesetzt werden, ohne einen Hersteller bestehender Automatisierungstechnik beauftragen zu müssen. Das ist im Industriellen/Elektronischen Bereich sehr wenig verbreitet.

## Ausführliche Darstellung der Ergebnisse

*Welche konkreten Ergebnisse hast Du erzielt? Konnten alle Meilensteine erreicht werden? Welche zusätzlichen Erkenntnisse hast Du aus der Projektarbeit gewonnen, auch im Hinblick auf die Begleitung durch die Open Knowledge Foundation?*

Die Ethersweep Hardware wurde ebenfalls meinerseits entwickelt, bestand aber bereits vor dem Projektstart. Die meiste Software, die während des Projektes entstanden ist, wird auf der Hardware ausgeführt wird.

Dazu zählt ein grundsätzlicher Neuaufbau der Steuersoftware zu Netzwerkansteuerung, Steuerung und Feedback.




*Darstellung der einzelnen Hardware-Komponenten auf den Leiterplatten*

Die Ethersweep Software läuft auf einem STM32F103 Microcontroller, welcher in C++ programmiert wurde. Dabei sind auf der Hardware verschiedene Bauteile verbaut, die unterschiedliche Aufgaben erledigen. Jedes Bauteil musste auch Softwareseitig angesteuert werden. Dazu zählen: W5500 Netzwerkinterface, TMC2209 Schrittmotortreiber, OLED Display, AS5600 Magnetencoder, CP2102 USB/Serial Chip und passive Komponenten wie ein Spannungssensor, Input/Output für End- und Notausschalter.

Die Entwicklung der Software zur Ansteuerung der einzelnen Komponenten war notwendig, um übergeordnete Funktionen zu ermöglichen. Die Übergeordneten Funktionen sind:

Netzwerkkommunikation über UDP und TCP, IP-Addressenvergabe mittels DHCP, OLED Display Ansteuerung, Spannungsüberwachung, AM5600 Encoder auslesen, TMC2209 Schrittmotortreiberansteuerung und Input/Output überwachung.

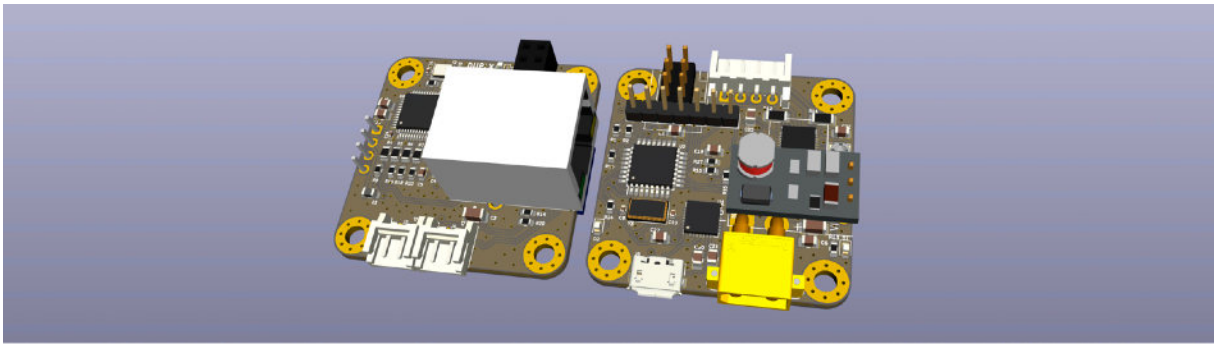
Alle Komponenten zusammen wurde verwendet um die eigentliche Motorsteuerung zu ermöglichen. So können die NutzerInnen nun über UDP, JSON encodete Nachrichten an Ethersweep senden, welche darauf verarbeitet und ausgeführt werden. Durch einige Python Beispiele wurde diese Funktionen implementiert, damit ein einfacher Einstieg ermöglicht wird.

Wird der Controller eingeschaltet (mit Spannung versorgt), besorgt sich der Controller über das DHCP Protokoll eine IP-Addresse, falls verfügbar. Sollte kein DHCP bestehen, nutzt der Controller eine Fallback IP-Addresse. Ist der Controller nun im Netzwerk etabliert, kann er über einen UDP Broadcast Call aufgerufen werden, um sich zurückzumelden. Das ist sinnvoll, sollten mehrere Ethersweeps verwendet werden und bspw. kein Zugang zum Controller bestehen (IP-Adresse steht ebenfalls auf dem OLED Display). Nun kann der Controller von allen Geräten im selben Netzwerk (oder erweiterten Netzwerk, auch per Port-Forwarding über das Internet) Nachrichten empfangen und angesteuert werden. Angesteuert wird über eine simple JSON encodete Nachricht, welche über UDP gesendet wird. Die Nachricht sieht bspw. so aus: {‘mode’ : 4, ‘degrees’: 402.3, ‘speed’: 200} Dieser Befehl lässt den Motor an Position 402,3° (mehr als eine Umdrehung) mit einem Geschwindigkeitswert von 200 drehen. Mode 4 bedeutet Servo-Mode, also dass der Motor Störeinflüsse eigenständig versucht zu überschreiben. Es kann aber ebenfalls ein Befehl gesendet werden, dass der Motor eine bestimmte Anzahl Schritte mit einer bestimmten Geschwindigkeit macht. Oder Feedback zur aufrufenden Stelle (oder anderer, konfigurierbarer IP) gibt (Zustände von Schaltern, Spannung, Motorwinkel,

Betriebszustand, Fehler etc.). Oder ein Befehl zur Identifikation, sodass der Motor ein Pieps- und Lichtsignal gibt. Ebenfalls können Konfigurationsdaten zum Controller gesendet werden, um bspw DHCP und verschiedene Standardparameter zu ändern. Diese Daten werden dauerhaft im internen Speicher gespeichert und bestehen auch nach einen Neustart des Controllers.

Die Dokumentation mit Beispielen gibt einen besseren Eindruck in die Funktionalitäten des Controllers, als hier sinnvollerweise möglich wäre.

Der Controller erfüllt damit aber eine Vielzahl an Funktionen von Servomotoren und ist durch das leicht zu begreifende und zu implementierende Kommunikationsprotokoll äußerst vielseitig anwendbar. Eine iPhone app wurde ebenfalls erstellt, um die Funktionen des Controllers zu Testen und auszuprobieren.



*3D rendering der Leiterplatten*

## Zielgruppe, Nutzen und mögliche Weiterentwicklungen

*Welcher Nutzen ergibt sich für die Zielgruppe aus den Ergebnissen Deines Projekts? Welche weitergehenden Effekte ergeben sich aus der Open-Source-Stellung der Ergebnisse? Gibt es Ideen für die Weiterentwicklung Deiner Lösung und Pläne zu deren Umsetzung?*

*Hat die Arbeit in dem Projekt Dich in Deiner persönlichen, fachlichen Weiterentwicklung unterstützt?*

Nutzer des Projektes verwenden die Motoren bereits für verschiedene Aufgaben. Von Aktuatoren an Robotern und Prototypen über verstellbare Tracking-Antennen, zu Lasersteuerungen in einem Sägewerk oder einem Roboterarm zu Herstellung von Kompositwerkstoffen, aber auch Kunstinstallationen (ähnlich dem Kinetic Rain am Singapurer Flughafen), gibt es ein breites Interesse an den Motoren und der Software.

Durch die freie Verfügbarkeit der Software und der Elektronik, ist die Einstiegsschwelle günstig und niedrig. Durch die Kollaborationsmöglichkeiten auf GitHub und YouTube bspw. wird Feedback zum Projekt gegeben und Wünsche geäußert, welche Teilweise bereits umgesetzt wurden. Erweiterungen können natürlich auch eigenständig umgesetzt werden und der Controller umgenutzt werden.

Einige Veränderungen benötigen neue Hardware, andere lassen sich in der Software umsetzen. Da das Projekt kontinuierlich weiterentwickelt wird, werden einige dieser Vorschläge wahrscheinlich umgesetzt werden. Dazu gehört Power over Ethernet, ArtNET Ansteuerung, bessere

Echtzeitverarbeitung der Encoder-Daten und allgemeine Verbesserungen und Aktualisierungen der Hard- und Software. Ebenso sollen weitere Bibliotheken zur Ansteuerung entwickelt werden.

Persönlich konnte viel aus dem Projekt mitgenommen werden. Das Projekt hat dazu beigetragen ein großes Netzwerk an Interessierten und potentiellen Nutzern aufzubauen. Durch das Feedback der Community konnten neue Ideen und Features umgesetzt werden. Auf GitHub hat das Projekt mittlerweile ca. 450 Stars, was für ein Projekt mit hohem Hardwareanteil, und damit verbundener erhöhten Einstiegsschwelle, viel ist. Auf LinkedIn erreichen Posts häufig Fünfstellige Aufzuzahlen. Auf YouTube erreichte das aktuelle video binnen weniger Wochen 50000 Klicks, was ebenfalls ungewöhnlich viel ist, im vergleich zu anderen meiner Videos und Posts. Daher denke ich, dass das allgemeine Interesse nach wie vor hoch ist, und das Projekt einen Erkennbaren Mehrwert mit sich bringt.

## Kurze Darstellung der Arbeiten, die zu keiner Lösung geführt haben

*Gab es Arbeiten bzw. Lösungsansätze, die nicht weiter verfolgt wurden? Was waren die Hintergründe, und wie bist Du alternativ vorgegangen?*

Während der Entwicklung wurde durch Branches verschiedene Aufgaben verteilt und parallel bearbeitet. Dadurch wurden Arbeiten auch mehrfach gemacht und entsprechend gelöscht, als die Branches zusammengefügt wurden.

Ein größeres Problem ergab sich bei der Verbesserung der Echtzeitfähigkeit. Dazu ist wichtig zu wissen, wie das Problem eigentlich aussieht.

Der Controller versucht permanent verschiedene Dinge gleichzeitig zu erledigen. Echte Gleichzeitigkeit ist aber nicht möglich, da der Prozessor Aufgaben Stück für Stück nacheinander abarbeiten muss. Daher wird ein Scheduler gebaut, der die Aufgaben priorisiert und erst dann abarbeitet, wenn es notwendig ist. Bspw. muss das Display nur alle 100ms aktualisiert werden, der Encoder und Motor aber so häufig wie möglich angesteuert werden, um einen Runden lauf des Motors zu ermöglichen. Das Netzwerkinterface muss ebenfalls häufig abgerufen werden, um neue Steuerdaten zu erhalten. Das Steuern und Abrufen des Displays, des Netzwerkchips und des Encoder dauern relativ lange. Daher wird während der Motor in bestimmten Betriebsmodus läuft, das Display nicht mehr aktualisiert und auch der Netzwerkchip nicht abgefragt.

Das eigentliche Problem liegt derzeit darin, dass der Netzwerkchip eigentlich über einen dedizierten Anschluss dem STM32 Microcontroller signalisieren kann, dass eine neue Nachricht vorliegt und er bereit ist, über das SPI Protokoll Daten auszutauschen. In anderen Tests konnte ich diese Funktion nutzen, doch auf dem STM32 scheint diese Funktion nicht ohne weiteres verfügbar zu sein. Daher konnte ich die geplante verbesserte Echtzeitfähigkeit nicht ganz erfüllen. Bis jetzt wird der Scheduler-Ansatz verwendet, der manchmal ein leichtes Ruckeln in der Motorsteuerung verursacht. Das ist aber vornehmlich in schnellen Bewegungen und besonderen Fällen zu erkennen.

Möglicherweise benötigt die Lösung des Problems Veränderungen der Hardware.

## Kurze Angabe von Präsentationsmöglichkeiten für mögliche Nutzer

*Wo können sich Interessenten detailliert über Deine Projektergebnisse informieren (z.B. Webseite, GHitHub, Veröffentlichungen)?*

Das Projekt wird auf https://github.com/neumi/ethersweep entwickelt und verwaltet. Die Dokumentation und Anwendungsbeispiele befindet sich ebenfalls dort.

Neuigkeiten gibt es dazu auf https://www.linkedin.com/in/neumi und teilweise auf https://www.youtube.com/neumi

## Kurze Erläuterung zur Einhaltung der Arbeits- und Kostenplanung

*Gab es im Projektverlauf Ereignisse, die eine Anpassung der Planung erforderlich machten – z.B. Mehr- oder Minderaufwand bei der Bearbeitung von Teilaufgaben?*

Die Teilaufgaben haben sich Zeitlich ineinander verschoben und dadurch, dass parallel an den Aufgaben gearbeitet wurde, konnte kein genaues Zeittracking gemacht werden. Insgesamt hätte einzelne Features noch weiter entwickelt werden können, doch die geplanten Neuerungen wurde umgesetzt. Insgesamt hat der Zeitplan für die gesetzten Aufgaben gut gepasst. Das gesamte Projekt ist aber noch nicht vollständig und bedarf noch weiterer Entwicklungszeit und Finanzierung für die Zukunft.

## Kurze Darstellung von etwaigen Ergebnissen bei anderen Stellen

*Gab es Entwicklungen anderer Personen oder Institutionen, die Einfluss auf Deine Arbeiten und die Zielsetzung hatten? Wenn ja, worin bestand dieser und wie bist Du damit umgegangen?*

Während der Projektzeit wurde ein sehr ähnlicher Open-Source Controller veröffentlicht, welcher Hardwareseitig sehr vielversprechend aussah. Einzelne Funktionen sind sehr ähnlich, doch die Ansteuerung erfolgt nicht über Ethernet, sonder über RS485. Daher überschneiden sich zwar die Anwendungsfälle, aber das Kernkonzept von Ethersweep ist die Standard-Netzwerkfähigkeit über das normale Ethernet Protokoll. Daher könne beide Projekte parallel existieren und stellen kaum eine Konkurrenz dar. Außerdem wurde zu dem Projekt keine Software veröffentlich, sodass der Controller bislang nicht zu verwenden ist. Ich bin im Austausch mit dem Entwickler, jedoch scheint die Software ein größeres Problem darzustellen. Teile der Ethersweep Software könnten auch dort Verwendung finden.

