Wi-Fi IEM – In-Ear Monitoring über WLAN

Sven Thielen, Dieter Leckschat, Christian Epe

Hochschule Düsseldorf / University of Applied Sciences
Institute of Sound and Vibration Engineering – ISAVE
sven.thielen@study.hs-duesseldorf.de

Einleitung

Durch WLAN sind drahtlose Netzwerkverbindungen keine Seltenheit mehr. In der professionellen Audiodomäne, insbesondere im Bereich von Anwendungen, welche eine niedrige Latenz voraussetzen, bspw. In-Ear Monitoring (IEM), findet es hingegen kaum Anwendung. Konventionelle Systeme für In-Ear Monitoring sind meist teuer und daher bei Künstler*innen mit geringem Budget weniger verbreitet.

Das in diesem Beitrag vorgestellte Wi-Fi IEM [1] ist der Prototyp eines kostengünstigen WLAN In-Ear Monitoring Systems auf Basis von Raspberry Pi [2] mit sogenannten HAT-Soundkarten sowie Freier Software im Kontext von Linux Real-Time Kernel [3] und JACK Audio Connection Kit, kurz JACK [4]. Neben den Aspekten des Kostenaufwands und der Mobilität stellt die unterbrechungsfreie Audioübertragung bei minimaler Latenz eine große Herausforderung dar.

Konzeption

Im Hinblick auf die Übertragung von Audiodaten per WLAN ist es erforderlich, die Datenübertragungsrate mit den vsl. Audiobitraten gegenüber zu stellen. Konventionelle IEM-Systeme arbeiten mit einem Zweikanal-Audiosignal, sodass unter dieser Annahme in Tabelle 1 zu erwartende Audiobitraten in Megabit pro Sekunde (MBit/s) für unterschiedliche Wortlängen (w) in Bit und Sampleraten (f) in Kilohertz aufgeführt sind.

Tabelle 1: Audiobitraten in MBit/s bei 2 Audiokanälen

	f / kHz						
w / Bit		**192**	**186,4**	**96**	**88,2**	**48**	**44,1**
	16	6,144	5,965	3,072	2,822	1,536	1,411
	24	9,216	8,947	4,608	4,234	2,304	2,117
	32	12,288	11,930	6,144	5,645	3,072	2,822

Es ist zu erkennen, dass die Kapazitäten aktueller Wi-Fi Spezifikationen wie Wi-Fi 5 mit Übertragungsraten von theoretisch 1,3 GBit/s durch die benötigte Audiobitrate von bspw. 12,288 MBit/s für ein 192 kHz Stereosignal mit 32 Bit Auflösung nur ansatzweise ausgeschöpft werden. Zwar wird die theoretische Datenübertragungsrate von Wi-Fi 5 auf Grund des Protokolloverheads und je nach Signalstärke nur eingeschränkt verfügbar sein, eine Kapazität von ca. 1 % der theoretisch möglichen Datenübertragungsrate sollte jedoch dauerhaft zur Verfügung stehen.

Zur Ermittlung eines Limits für die Verzögerungszeit bei der Audiosignalübertragung wird ein Abstand von 3 Metern zwischen Lautsprecher und Ohr ($d = 3\ m$) analog zu konventionellem Monitoring angenommen. Zusammen mit der Schallgeschwindigkeit ($c_{20} = 343\ m/s$) bei 20 ° Celsius durchschnittlicher Umgebungstemperatur liefert Formel (1) einen Grenzwert von $L_M = 8{,}764\ ms$. Ein Wi-Fi IEM-System sollte diese Verzögerungszeit im Mittel möglichst nicht überschreiten.

$$L_M = d * \frac{1}{c_{20}} \quad [\mathrm{s}] \qquad (1)$$

Für die weitere Betrachtung wird das geplante Wi-Fi IEM-System zunächst abstrahiert und in Module zerlegt. Kern dabei ist die Funkstrecke, welche mit einem Sende- und Empfangsmodul zu implementieren ist. Die A/D- bzw. D/A-Wandlung, als separater Vorgang, stellt jeweils eine eigene Komponente dar. Abbildung 1 zeigt den abstrakten Entwurf in Form eines Blockschaltbildes.

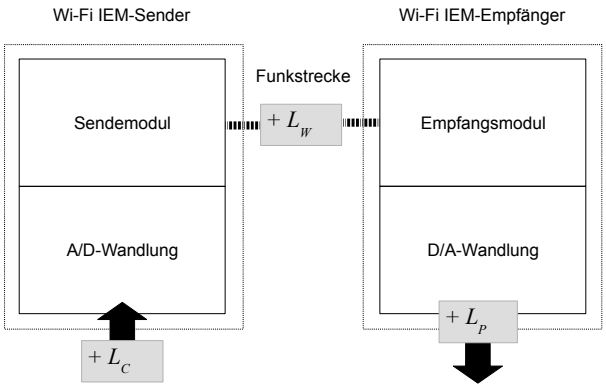

Abbildung 1: Blockschaltbild des Systementwurfes mit den zu erwartenden Latenzen: A/D-Wandlung L_C, Funkstrecke L_W und D/A-Wandlung L_P

Darin wird deutlich, dass beim Übergang von einem zum anderen Modul jeweils eine Verzögerung entsteht. Die Gesamtlatenz des Systems L_A ist die Summe der einzelnen Verzögerungen (2).

$$L_A = L_C + L_W + L_P \quad [\mathrm{s}] \qquad (2)$$

Audiosignalwandlung

Neben der variabel auftretenden Latenz L_W über die Funkstrecke sollte idealerweise keine weitere Verzögerung entstehen. Daher muss ein Wi-Fi IEM-System Komponenten enthalten, welche statische Verzögerungen für die A/D- und D/A-Wandlung während des Betriebs gewährleisten. Freie Software liefert diesbezüglich ein Betriebssystem auf Basis von Raspbian [5] mit dem Namen RealtimePi [6]. Diese Distribution enthält einen mit dem Preempt RT Patch [3] modifizierten Linux-Kernel für Raspberry Pi, welcher durch gleichzeitige Verwendung von JACK als Audio Server die

Audiosignalverarbeitung priorisieren kann, um statische Latenzen unterhalb von 2 bis 3 ms zu erreichen. Dadurch kann ein Szenario geschaffen werden, in welchem ein differenziertes Analysieren und Optimieren der Funkstrecke möglich ist. Die Wiedergabe- bzw. Aufnahmelatenz (playback/capture latency) bei der Wandlung kann theoretisch ermittelt werden. In Formel (3) ist erkennbar, dass die Periods (n) als Faktor auftreten. Dessen Wert muss mindestens 2 betragen und in der Regel nur bei USB- bzw. Firewire-Soundkarten erhöht werden. Zusammen mit den Frames (p) als Zweierpotenz stellt dies den Hardware-Puffer dar [4].

$$L_P = \frac{n * p}{f} \quad [\text{s}] \qquad (3)$$

Unabhängig von der Wortbreite, Kanalanzahl usw. entspricht ein Frame genau einem Sample im Kontext der Advanced Linux Sound Architecture (ALSA) als Treiber für JACK. Der Period-Parameter gibt die Anzahl von Frames zwischen jedem Hardware-Interrupt an. Über einen sogenannten Ring buffer wird Frame für Frame abgespielt. Während die ALSA-Algorithmen versuchen den Hardware-Puffer stets gefüllt zu halten, entsteht ein Interrupt beim Übergang vom ersten zum zweiten Frame, vom zweiten zum dritten usw.. Bei nicht optimalen Einstellungen treten dadurch vermehrt sogenannte xruns (Über- bzw. Unterläufe des Hardware-Puffers) auf, welche sich in Form von hörbaren Aussetzern bemerkbar machen und verhindert werden sollten [7].

Tabelle 2: Latenzen in ms bei 2 Periods (n)

	Samplerate (f) / kHz						
Frames (p)		**192**	**186,4**	**96**	**88,2**	**48**	**44,1**
	32	0,333	0,343	0,667	0,726	1,333	1,451
	64	0,667	0,687	1,333	1,451	2,667	2,902
	128	1,333	1,373	2,667	2,902	5,333	5,805
	256	2,667	2,747	5,333	5,805	10,667	11,610
	512	5,333	5,494	10,667	11,610	21,333	23,220
	1024	10,667	10,987	21,333	23,220	42,667	46,440

In der Tabelle 2 sind Berechnungen mit üblichen Parameterkombinationen aufgeführt. Die Samplerate ist durchgehend von 192 kHz bis 44,1 kHz angegeben. In Bezug darauf wird ersichtlich, dass eine Erhöhung dieser bei gleichbleibenden Frames eine Verringerung der Latenz bewirkt.

Audiosignalübertragung

Mit JACK als Basis stehen verschiedene Audio over IP Lösungen zur Übertragung des Audiosignals über die Funkstrecke zur Verfügung. Bei Wi-Fi IEM erscheint die Client-Anwendung zita-njbridge sinnvoll, da keinerlei Abhängigkeiten von der Netzwerkinfrastruktur bestehen, Multicasting (Adressierung mehrerer Empfänger) implementiert ist und kein dedizierter Taktgeber infolge eines Master/Slave-Designs notwendig ist. Sender und Empfänger können somit unterschiedliche Sampleraten verwenden. Die Anpassung geschieht empfangsseitig mit Adaptive Resampling [8], welches durch die Programmbibliothek zita-resampler zur Verfügung gestellt wird. Zita-njbridge dient als Brücke zwischen zwei oder mehr Computern mit JACK als Audio Server zur Übertragung von bis zu 64 Kanälen per UDP. Ein Sender kann über die Verwendung von Multicast IP-Adressen an mehrere Empfänger streamen. Dabei wird jeweils die Sender- oder Empfänger-Applikation mit den Namen zita-j2n bzw. zita-n2j verwendet.

Implementierung

Aus dem vorangegangenen theoretischen Fundament werden nachfolgend Prototypen mit Raspberry Pi dargestellt. Die Wahl dieser Plattform ist vor allem dem modularen Design mit Do-It-Yourself Ansatz sowie der Mobilität geschuldet und liegt nicht zuletzt auch in dem geringen Preis der Komponenten sowie der maßgeblichen Verwendung von Freier Software begründet.

Raspberry Pi Einplatinencomputer

Beim Raspberry Pi, im Folgenden kurz RPi genannt, handelt es sich um einen Einplatinencomputer mit ARM-Prozessor. Das verwendete Modell 3 B+ ist mit einem ARM Cortex-A53 4-Kern CPU sowie 1 GB Arbeitsspeicher ausgestattet. Neben einem Gigabit-Ethernet Anschluss über den USB 2.0 Bus (ca. 300 MBit/s Datendurchsatzrate) ist eine Drahtlos-Netzwerkschnittstelle mit Wi-Fi 5 Standard (IEEE 802.11ac) integriert. Über einen 40-pin GPIO Header können Erweiterungsmodule (HAT) verwendet werden [9]. Die integrierte Soundkarte hat lediglich einen Audioausgang mit mäßiger Klangqualität und ist nicht für niedriglatente Anwendungen geeignet, sodass die D/A-Wandlung für den Empfänger durch eine 192 kHz HAT-Soundkarte mit Kopfhörerverstärker [10] erfolgt und zur Aufnahme im Sender-RPi eine vergleichbare HAT mit Line-In [11] zum Einsatz kommt.

Der RPi 3 B+ ist bereits ab 38,50 € erhältlich und die HAT-Soundkarte kostet jeweils 35 € bis 50 €. Die mobile Stromversorgung des Empfängers könnte mit einer Powerbank für Smartphones im Bereich von 5 bis 15 € realisiert werden. Zusammen mit einem Plastikgehäuse für durchschnittlich 10 € und einer 16 GB SD-Karte ab 5 € sind sowohl die Sender- als auch Empfänger-Bauteile für weniger als 100 € erhältlich. Als Set mit einem Empfänger kostet ein Wi-Fi IEM-System somit weniger als 200 € und jeder weitere Empfänger ca. 95 bis 100 € (Stand: 24.03.2020).

Für die Inbetriebnahme werden beide SD-Karten mit der RealtimePi-Distribution ausgestattet. Es wird der „Nightly Build“ vom 16.12.2019 (Raspbian Buster) verwendet. Nach der Post-Installationsroutine beim ersten Starten, erfolgen die jeweiligen Konfigurationen der RPis als Empfänger bzw. Sender. Fortan werden die Bezeichnungen „iem-tx“ für den Sender und „iem-rx“ für den Empfänger verwendet.

IEM-Empfänger – iem-rx

Zunächst erfolgt die Deaktivierung der internen Soundkarte sowie des Bluetooths durch das Einfügen entsprechender Zeilen in der Datei */boot/config.txt*. In der *cmdline.txt*-Datei

am selben Speicherort wird der „Low-latency mode“ für das Schreiben auf die SD-Karte durch Anfügen von *sdhci_bcm2708.enable_llm=0* an die Kernelbootparameter deaktiviert. Des Weiteren werden die folgenden nicht benötigten Dienste jeweils mit Hilfe der beiden Befehle *systemctl disable* und *systemctl mask* deaktiviert sowie maskiert, um dessen Ausführung effektiv zu verhindern:

- bluetooth.service
- dbus.service
- hciuart
- triggerhappy.service
- usage-statistics.service

Daraufhin kann die Installation der Software-Pakete jackd1 und zita-njbridge erfolgen. Im Zusammenhang mit dem jackd1-Paket muss der Frage für das Aktivieren der Prozess-Priorisierung in Echtzeit (Enable realtime process priority) zugestimmt werden und die Werte *vm.swappiness=10* sowie *fs.inotify.max_user_watches=524288* in der Systemdatei */etc/sysctl.conf*-Datei gesetzt werden. Dadurch wird das Auslagern von Arbeitsspeicherinhalten auf die Festplatte (Swapping) erst bei gänzlich genutztem Arbeitsspeicher erlaubt und die Überwachung für Verzeichnisänderungen limitiert, sodass keine Performance-Einbußen auftreten [12].

Zuletzt geschieht die Einrichtung als WLAN-Client über das Entfernen der Kommentarzeichen und Einsetzen der SSID sowie PSK (Pre-Shared Key) für ein „WPA/WPA2 secured network“ in der Datei *realtimepi-wpa-supplicant.txt* auf der boot-Partition.

IEM-Sender – iem-tx

Die Konfiguration des RPi-Senders iem-tx ist analog zu iem-rx mit der Ausnahme, dass ergänzende Software-Pakete für den Betrieb als WLAN Access Point (AP) notwendig sind. Zusätzlich zu o.g. Paketen werden dnsmasq und hostapd installiert. Die Einrichtung eines RPis als Access Point mit diesen Software-Komponenten wird in der Dokumentation der Raspberry Pi Foundation ausführlich beschrieben [13]. Im Gegensatz zum Empfänger muss abschließend in der Datei */boot/realtimepi-wpa-supplicant.txt* lediglich der Parameter *country=DE* gesetzt werden.

Zur automatisierten Installation von Sender und Empfänger liegen Bash-Skripte und vorgefertigte Konfigurationsdateien bereit [14]. In diesem GitHub-Projekt sind außerdem kommentierte Bash-Skripte zur Ausführung des JACK Audio Server Daemons jackd und der damit zusammenarbeitenden Programme zita-j2n bzw. zita-n2j mit optimalen Parametern für die Inbetriebnahme der nachfolgenden Prototypen zu finden. Auf Grund des Umfangs finden Details zu Letzteren hier nur in Ansätzen Erwähnung und erheben daher keinerlei Anspruch auf Vollständigkeit.

Latenzmessungen

Durch Messungen der Verzögerungszeit über einen Zeitraum von zwei Stunden in einer interferenzfreien Umgebung können zwei Prototyp-Varianten für Wi-Fi IEM erprobt werden und im Folgenden Erkenntnisse über das Verhalten der Audioübertragung per Wi-Fi 5 gewonnen werden. Beim Alpha-Prototyp sind beide RPis direkt verbunden, indem iem-tx als AP fungiert. Der Prototyp B wird durch einen kostengünstigen WLAN-Router für Heimanwender mit MU-MIMO Technologie [15] erweitert. Dieser übernimmt die Funktion des APs und ist per Gigabit-Ethernet (IEEE 802.3ab) am Sender-RPi iem-tx angeschlossen. Zwar sind durch den Einsatz kabelgebundener Netzwerkverbindungen weitere variable Latenzen zu erwarten, jedoch sind diese im Full-Duplex Modus minimal und werden daher nicht differenziert in die Betrachtung einbezogen.

Prototyp A

Im Alpha-Stadium des Wi-Fi IEM Prototyps werden beide RPis direkt über Wi-Fi 5 mit iem-tx als AP verbunden. In der Abbildung 2 ist der Aufbau für ein solches Wi-Fi IEM-System zu sehen. Die A/D- bzw. D/A-Wandlung erfolgt auf beiden RPis mit 128 Frames und 2 Periods bei 96 kHz als JACK-Parameter. Das Programm zita-j2n sendet mit 16 Bit Auflösung den zweikanaligen Audiostream des iem-tx über das Interface wlan0 an die Multicast-Adresse 224.0.0.3 auf Port 30042 und iem-rx empfängt diesen mit einem Jitter-Puffer von 3 ms über die gleiche Schnittstelle.

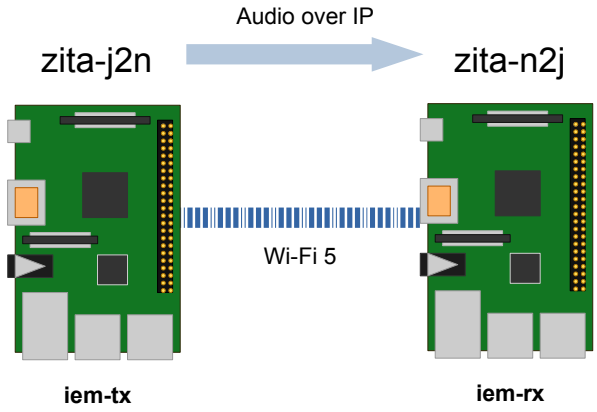

Abbildung 2: Alpha-Prototyp mit sendendem Raspberry Pi „iem-tx“ als Wi-Fi 5 Access Point.

Die Latenzmessung kann durch ein dem JACK Audio Server beiliegenden Programm namens jack_iodelay erfolgen, welches zur Feststellung von „Round-Trip Latency“ entwickelt wurde und auf Grund des flexiblen Routings von Ein- und Ausgängen mit JACK vielseitig benutzt werden kann. Indem der Audioausgang des Empfängers mit dem Eingang des Senders über ein Klinkenkabel verbunden wird und auf diese Weise eine Schleife, zusammen mit der Audio over IP Verbindung entsteht, ist jack_iodelay in der Lage, die durch Verzögerung entstehenden Phasenverschiebungen zu analysieren und den Latenzwert kontinuierlich in Millisekunden anzugeben. Abbildung 3 zeigt das Histogramm einer jack_iodelay-Latenzmessung über einen Zeitraum von zwei Stunden.

Abgesehen von periodisch auftretenden Abweichungen bis hin zu 8,94 ms, welche auf mögliche Aushandlungen im Zusammenhang mit der WLAN-Konfiguration deuten, liegt der Mittelwert der Latenz für Prototyp A mit einer Standardabweichung von 0,103 ms bei 7,754 ms. Auf Grund

der o.g. Abweichungen liegen der Minimal- und Maximalwert mit 7,646 ms und 8,947 ms weit auseinander.

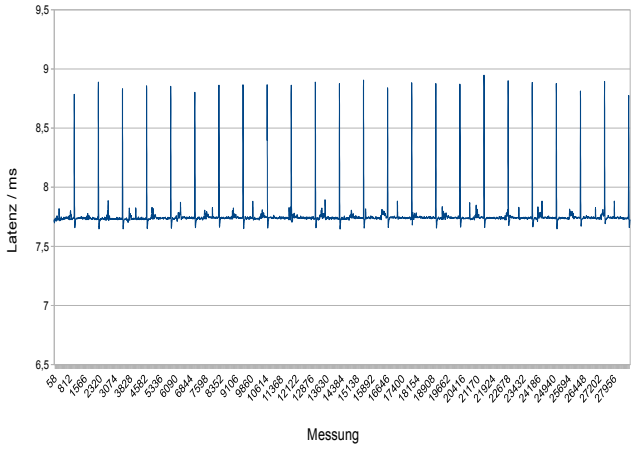

Abbildung 3: Latenzmessung des Prototyps A (iem-tx als Access Point) über einen Zeitraum von zwei Stunden

Prototyp B

Bei Prototyp B erfolgt die Latenzmessung analog, sodass die Parameter für JACK und die zita-njbridge Programme identisch sind. Eine Ausnahme stellt jedoch die Angabe des Interfaces eth0 beim Sender-RPi iem-tx dar, um den Multicast-Audiostream per Ethernet an den WLAN-Router und über diesen schließlich per Wi-Fi 5 zu senden.

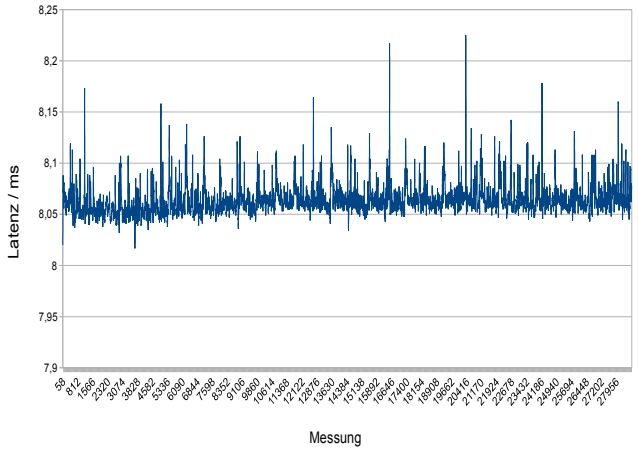

Abbildung 4: Latenzmessung des Prototyps B (WLAN-Router als AP) über einen Zeitraum von zwei Stunden

In der Abbildung 4 ist der Verlauf der Latenz mit diesem Szenario dargestellt. Im Vergleich zum Alpha-Prototyp gibt es hierbei nur Standardabweichungen im Bereich unterhalb von 0,015 ms bei einem leicht erhöhten Mittelwert von 8,065 ms. Minimal- und Maximalwert liegen mit 8,017 ms und 8,225 ms dicht beieinander.

Fazit und Ausblick

Eine Realisierbarkeit von IEM über WLAN mit Freier Software erscheint möglich, benötigt jedoch weitere Forschung. Schwerpunktmäßig wurde bisher die Latenzanalyse und -optimierung vorgenommen. Die Ergebnisse zeigen, dass gewisse Anforderungen, abhängig vom Szenario, erreicht werden können. Im Bereich der gewählten Zielplattform, Raspberry Pis mit HAT-Soundkarten und Real-Time Kernel in der jeweiligen Funktion als A/D- bzw. D/A-Wandler, Audiostreaming-Sender bzw. Empfänger sowie als WLAN Access Point, sollten weitere Untersuchungen erfolgen. Insbesondere ist auch die Performance beim Betrieb mehrerer Empfänger zu testen und Auswirkungen von Interferenzen auf das System zu ermitteln.

Literatur

[1] Thielen, S. In-Ear Monitoring über WLAN. Eine Analyse zur Realisierbarkeit mit Freier Software. Bachelorthesis, HS Düsseldorf, 2019

[2] Raspberry Pi Foundation Homepage, URL: `https://www.raspberrypi.org`

[3] The Linux Foundation – Linux Real-Time, URL: `https://wiki.linuxfoundation.org/realtime/start`

[4] JACK Audio Connection Kit – jackd Manpage, URL: `https://github.com/jackaudio/jackaudio.github.com/wiki/jackd(1)`

[5] Raspbian Homepage, URL: `https://www.raspbian.org`

[6] RealtimePi GitHub, URL: `https://github.com/guysoft/RealtimePi`

[7] alsa-project.org – FramesPeriods, URL: `https://www.alsa-project.org/main/index.php/FramesPeriods`

[8] Adriaensen, F.: Controlling Adaptive Resampling. Linux Audio Conference 2012, URL: `http://lac.linuxaudio.org/2012/papers/23.pdf`

[9] Raspberry Pi 3 Model B+ Product Brief, URL: `https://static.raspberrypi.org/files/product-briefs/Raspberry-Pi-Model-Bplus-Product-Brief.pdf`

[10] IQaudIO Pi-DAC+ Produktseite, URL: `http://iqaudio.co.uk/hats/8-pi-dac.html`

[11] HifiBerry DAC+ ADC Produktseite, URL: `https://www.hifiberry.com/shop/boards/hifiberry-dac-adc`

[12] Wiki.linuxaudio.org – System configuration, URL: `https://wiki.linuxaudio.org/wiki/system_configuration`

[13] Setting up a Raspberry Pi as a Wireless Access Point, Raspberry Pi Documentation, URL: `https://www.raspberrypi.org/documentation/configuration/wireless/access-point.md`

[14] Wi-Fi IEM GitHub, URL: `https://github.com/thisven/Wi-Fi_IEM`

[15] Linksys EA8500 Produktseite, URL: `https://www.linksys.com/de/p/P-EA8500`