Anleitung für Seuchen Beacons

Tobias Hofbaur *

24. Juli 2023

1 Überblick

Die Beacons wurden 2016 für den Kurs Seuchen und Epidemien in Urspring angeschafft um ein Seuche über die gesamte Akademie simulieren zu können, ohne auf die aktive Mitarbeit der TN angewiesen zu sein. Jeder Akademieteilnehmer und teilweise auch das Personal vor Ort bekommt mit dem Namensschild ein Beacon. Die Grundidee ist, dass die Beacon ständig ihre ID und ihren aktuellen Status senden. Befindet sich ein suszeptibler Beacon nahe und lang genug an einem infektiösen, wird er selber infektiös und sendet fortan den infektiösen Status.

Für die Akademie Torgelow 2017 wurde die Anwendung zusätzlich um die Aufzeichnung des sozialen Netzwerkes ergänzt.

Auf der Akademie Grovesmühle 2018 wurden die Beacons im Rahmen eines Kurses zur Graphentheorie zur Aufzeichnung des sozialen Netzes benutzt.

Auf der Akademie Grovesmühle 2022 wurden die Beacons im Rahmen eines Kurses zur Modellierung der Realität zur Simulation der Ausbreitung einer Seuche und zur Aufzeichnung des sozialen Netzes benutzt.

1.1 Software-Architektur

Das Gesamtsystem ist aus fünf Teilen aufgebaut:

- Firmware für Seuchensimulation und Netzwerkerfassung; Ordner "Network_Beacon"; Kann auf Beacons oder DK aufgespielt werden
- Firmware für Administration (Ordner Network_Control; Veränderung von Parametern im Betrieb; Neuinfektionen; etc.); Kann auf Beacons oder DK aufgespielt werden
- Firmware zum Aufzeichnen (Ordner Network_Base); liest die auf den Beacons gespeicherten Daten aus und sendet sie über eine UART Schnittstelle an den PC.

*tobias@hofbaur.eu

Aufzeichnung am PC kann als einfache Log Datei mit z.B. ExtraPutty erfolgen; kann nur auf DK aufgespielt werden.

- Matlab Skript (Log_Auswertung, R2017b)) das aus den Log Dateien eine csv Datei mit den Kontakten erstellt und eine mat Datei mit dem ausgewerteten Graphen (inklusive Seuchendaten)
- Matlab Skript (Seuchen_GUI, R2017b)) das den ausgewerteten Graphen einliest und auf einem gewählten Abschnitt grafisch darstellt.

1.2 Hardware

- $(117 - x)$ nRF51822 Bluetooth Smart Beacon Kit Rev 1.3 (32kB RAM) (Nordic Boardbezeichnung PCA10028), davon zwei neuangeschafft und noch mit der Default Firmware; Aktuell ist auf den Rev1.3 Chips (bis auf die 2 neuen) eine Software geflasht, die Seuchensimulation und Erfassung der sozialen Kontakte ermöglich. Darunter ist ein Bootloader geflasht für Update ohne Kabel. (Details unten; aktuelle Schätzung 2018 ist $x = 3$)
- 1 nRF51822 Bluetooth Smart Beacon Kit Rev 1.2 (16kB RAM)
- 4 nRF51 DK (32kB RAM) (Nordic Boardbezeichnung PCA20006)
- TC2030-CTX-NL DebugKabel mit Clip zur Fixierung

2 Allgemeines

2.1 Stromversorgung

Das DK kann über ein USB Kabel mit micro USB Stecker mit Strom versorgt werden. Alternativ ist eine Stromversorgung über eine CR2032 Knopfzelle möglich. Es dürfen niemals beide Stromquellen gleichzeitig verwendet werden! (Bei versehentlicher Verwendung ist es bisher aber auch noch nicht explodiert...) Die Beacons müssen über CR1632 Knopfzellen mit Strom versorgt werden. Das Debugkabel liefert keinen Strom, so dass auch bei Programmierung die Knopfzelle nötig ist.

2.2 Stromverbrauch

Der geringe Stromverbrauch von BLE basiert zu einem großen Teil in der sparsamen Aktivierung der BL Signals. Im Empfangmodus (Rx) verbraucht der Chip ca. 13 mA. Im Sendemodus je nach Leistungsstärke und Paketgröße 5mA. Ein Adv. Paket benötigt ca. 15ms Sendezeit. Da die Batterien nur 130mAh Kapazität haben muss sehr genau überlegt werden, welche Sende- und Empfangszyklen benötigt werden. Mehr dazu im Beispiel der SeuchenApp weiter unten.

2.3 Programmierung

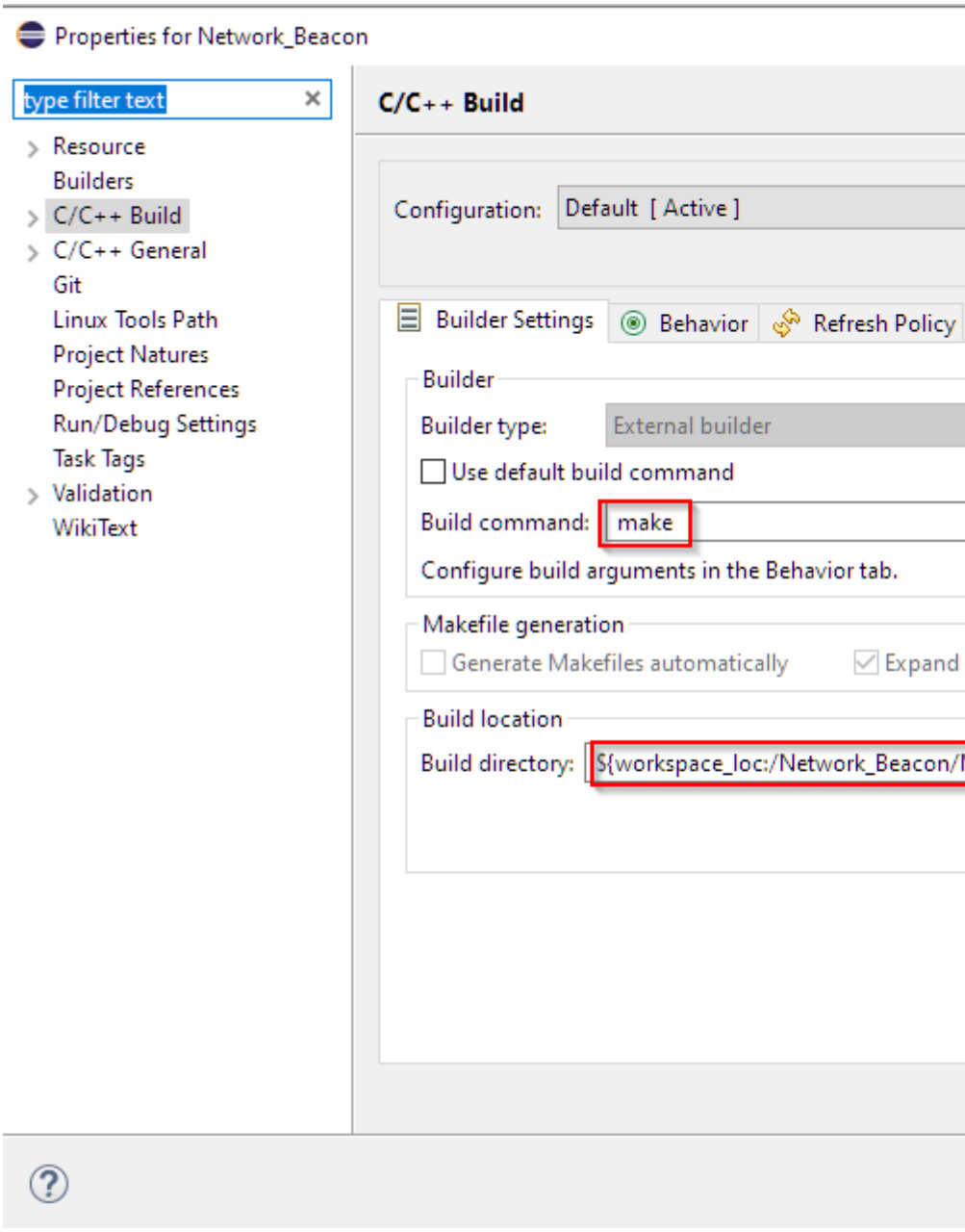
Die Dokumentation durch NordicSemiconductor ist hervorragend, ebenso der Support sowohl im Forum, als auch in bei expliziter Erstellung eines "Support Cases".

Für die Programmierung sieht Nordic mehrere Optionen vor. Der offizielle Weg verwendet die kostenpflichtige Entwicklungsumgebung (IDE) Keil. Die inoffiziell unterstützte Variante ist mit Makefiles über die OpenSource IDE Ecplise. Ein sehr gutes Tutorial ist unter `https://devzone.nordicsemi.com/tutorials/7/` (Stand 15.8.2016) zu finden. Das beschriebene Vorgehen hat bei mir auch bei späteren Versionen von Eclipse einwandfrei funktioniert. Lediglich für die Debug Einstellungen musste der Befehl des Debuggers explizit angegeben werden. Für den reinen Betrieb der Beacons ist keine Entwicklungsumgebung nötig, es reicht eine Toolchain (Compiler + make + Nordic-Kram); Details dazu in Abschnitt 2.3.3.

2.3.1 Windows

Download and install: Nordic Semi nrf51 SDK 12.3 NRF Command line tools from Nordic semi conductor Install the latest GNU Tools for ARM Embedded Processors available at https://launchpad.net/gcc-arm-embedded/+download -¿ adapt values in "C:/nRF5_SDK_12.3.0_d7731ad/components/toolchain/gcc/Makefile.windows"Core Utility from http://gnuwin32.sourceforge.net/packages/coreutils.htm Make utility from http://gnuwin32.source
Latest Segger J-link drivers from https://segger.com/jlink-software.html Latest Eclipse for Embedded C/C++ Install EMSysRegView from Eclipse Marketplace

New project from makefile The project should now be visible in your workspace. Then right-click on the project folder and enter properties-¿C/C++ Build and change the build directory to the gcc folder. Unmark Üse default build commandänd type "makeïf it is not



the default build command.
Erstelle Build target wie benötigt.

NordicSemiconductor stellt einen fertigen Bluetooth Stack zur Verfügung der die komplette Ansteuerung des BLE Senders übernimmt. Hierzu muss das Softdevice auf den Chip geflasht werden. Im Betrieb verbraucht der Softdevice ca. 8 kB RAM. NordicSemiconductor stellt den Softdevice und Code Beispiele in einem SDK zur Verfügung. Für die aktuelle Software wurde SDK 12.3 (`https://www.nordicsemi.com/eng/Products/Bluetooth-low-energy/nRF5-SDK#Downloads`) verwendet. Beim Wechsel der Versionen werden die Schnittstellen und Funktionen von NordicSemi teilweise verändert, so

dass ein Upgrade nur erfolgen sollte falls unbedingt nötig.

Die Code Beispiele in den Software Delevopment Kits (SDK) bieten einen guten Startpunkt.

Auf den Beacons (bis auf den zwei neuen) wurde ein Bootloader aufgespielt der ein drahtloses Update der Firmware ermöglicht. Das Vorgehen dazu ist der Dokumentation von Nordice beschrieben. Der private Schlüssel ist im Ordner "Vault". Der Bootloader sollte weiterhin aufgespielt werden da im Falle eines Verlust oder Defekts des teuren J-Tag Kabels trotzdem die Firmware aktualisiert werden kann.

Bekannte Probleme:

- Nach Flashen muss ein Pin-Reset oder Stromunterbrechung vorgenommen werden. Ansonsten bleibt der Chip im Debug Modus und verbraucht wesentlich mehr Strom. In Seuchenprogramm ist der Pin reset im Makefile integriert. Bei den makefiles der Beispiele von nordic ist das (für nRF5_SDK_11.0.0_89a8197) nicht implementiert.
- Falls nicht aufgespielt werden kann, kann ein kompletter "erase" im nrfGoStudio helfen.

2.3.2 Bootloader

Für die Erstellung des Bootloaders die Anleitung von Nordic Semiconductors befolgen (wichtig: kein Python 3 (confirm?)). Es wird das python package nrfutil benötigt. `https://devzone.nordicsemi.com/b/blog/posts/getting-started-with-nordics-secure-dfu-bootloader` Kommando um settings für Bootloader zu generieren.

```
nrfutil settings generate --family NRF51 --application nrf51822_xxac_s130.hex
     --application-version 0 --bootloader-version 0 --bl-settings-version 1
     bootloader_setting.hex

mergehex --merge s130_nrf51_2.0.1_softdevice.hex bootloader_nrf51822_xxac_s130.hex
--output BL_SD.hex

mergehex --merge BL_SD.hex nrf51822_xxac_s130.hex
--output BL_SD_APP.hex

mergehex --merge BL_SD_APP.hex bootloader_setting.hex
--output BL_SD_APP_SET.hex

nrfjprog  -f NRF51 --program BL_SD_APP_SET.hex  --chiperase  --verify
```

Zum updaten ein neues package erzeugen mit

```
nrfutil pkg generate --hw-version 51 --application-version 0
--application nrf51822_xxac_s130.hex --sd-req 0x87
--key-file priv.pem app_dfu_package.zip
```

Upload des package über

```
nrfutil dfu ble -ic NRF51 -pkg app_dfu_package.zip -p COM6 -f
```

Falls das Update wegen Error 13 fehlschlägt, kann es sein, dass das DK nicht richtig geflasht ist. Dann mit

```
nrfjprog --eraseall
```

das DK komplett löschen. Die Update Firmware wird automatisch aufgespielt.

2.3.3 Minimales Entwicklungsumgebung

Der Verzicht auf eine echte Entwicklungsumgebung bietet sich insbesondere unter Linux an, wo die Toolchain per Shell zugänglich ist. Dazu befolgen wir wie oben beschrieben die Anleitung von Nordicsemi, hören aber direkt vor der Installation von Eclipse auf. Außerdem führen viele Linux-Distributionen die jlink-Software von Segger in ihren Repositories, sodass diese bequem per Paketmanager installiert werden kann.

Zum Anpassen der Funktionalität können nun der Quellcode in einem beliebigen Editor angepasste werden (wesentliche Datei wird hier Network_Control/main.c sein). Danach wird das entsprechende Gerät per USB-Kabel an den Rechner angeschlossen und der Befehl `make` im entsprechenden Verzeichnis (bspw. Network_Control/Makefile_Linker) mit dem gewünschten Ziel (diese heißen `flash_*` mit jeweils passender Endung) aufgerufen.

3 Wirkprinzip

Mit BLE lassen sich verschiedene Betriebsmodi realisieren. Im Peripheral-Betrieb werden regelmäßig "Advertisments" ausgesendet, die Beaconspezifisch angepasst werden können. Nutzlast hierbei sind maximal 29 Byte. Im Central-Betrieb wird kontinuierlich oder regelmäßig nach anderen BLE Sendern im Peripheral-Betrieb in der Nähe gesucht. Dabei wird das komplette Advertisments empfangen und kann verarbeitet werden. Hier können auch weitere Daten angefordert werden (Scan Request) die vom Peripheral in einer Scan-Response gesendet werden. Hierbei können weitere 29 Byte übermittelt werden. Ein Central kann auch eine Verbindung zu einem Peripheral herstellen (connect) um die kompletten Daten auslesen zu können.

3.1 Der einzelne Beacon

In der aktuellen Version wechselt die Network_Beacon ständig zwischen Sende und Empfangsmodus. Sie kann auf Beacons und DK aufgespielt werden.

Im Sendemodus meldet sich jedes Beacon mit dem Namen "DSA" und sendet als Advertisment 3 Bytes Nutzlast.

1. Seine ID (Nr von 1 bis 120, eigentlich bis 128 möglich, aber aktuell können beim Auslesen maximal 15 Bytes = 120 Bit übertragen werden)

2. Infektionsstatus: Aktueller Zustand im SIRV Model und die Version der aktuell aufgespielten Seuchenparameter

   Genauer: `status_infect | (inf_rev << 5)` mit `status_infect`$\in [0, 6]$ sowie `inf_rev`$\in [0, 7]$

3. Batteriestatus und Menge an aufgezeichneten sozialen Kontakten.

   Genauer: `(status_batt << 4) | (status_data << 5)` mit `status_batt`$\in [0, 1]$ sowie `status_data`$\in [0, 7]$

Im Empfangsmodus wird nach Advertisements gescannt die von Geräten mit Namen DSA kommen. Ein solches Paket wird dann in verschiedenen Stufen ausgewertet.

- Ist das Paket von der Zentrale (Network_Control) werden die übermittelten Parameter eingestellt. Die Parameter werden aktuell nur im RAM gespeichert und werden nach einem Reset etwa durch Batteriewechsel oder Wackelkontakt wieder auf die Default werte gesetzt.
- Ist die Signalstärke über dem (parametrisierbar) eingestellten Limit wird zum einen der soziale Kontakttrigger für diese ID gesetzt und zum anderen überprüft ob das Beacon aufgrund seines Seuchenzustandes eine Auswirkung hätte. In diesem Fall wird ebenfalls ein Trigger gesetzt.

Die Trigger werden in der zyklischen Routine ausgewertet. Hier wird jeweils überprüft, ob ein Kontakt (sozial oder Seuche) lange genug bestand um registriert zu werden. Die Limits hierfür sind separat parametrisierbar. Für einen sozialen Kontakt wird bei Gültigkeit ein entsprechender Eintrag in einem Ringspeicher gemacht, der die ID des Kontakts enthält, Start und Endzeit des Kontakts in sekundengenauer Auflösung in einem internen Timer. Die Beacons haben keine absolute Zeit, sondern nur einen internen Sekunden Timer der ab reset läuft) Kontakte sollten idealerweise auf beiden Seiten aufgezeichnet werden.

3.2 Das Seuchenmodell

Bei einem Seuchenereignis wird ein entsprechender Zustandsübergang angestoßen (siehe Abbildung 1, die einzelnen Zustände sind in Tabelle 1 beschrieben) und die Daten ebenfalls in einem weiteren Ringspeicher vermerkt. Hierbei wird die aktuelle Zeit, der neue Zustand und die Verursacher gespeichert. Verursacher sind alle IDs in der relevanten Zeit Kontakt mit dem Beacon hatten und somit zur Infektion beitragen konnten. Bei internen Übergängen wie von Exponiert zu Infektiös wird 0 vermerkt.

Die Zeit für Scanning und Advertising sind aktuell fest eingestellt auf:

- #define ADV_INTERVAL 110 // Advertisement interval in milliseconds
- #define SCAN_WINDOW_MS 125 //scan window in milliseconds
- #define SCAN_INTERVAL_MS 10000 // scan interval in milliseconds

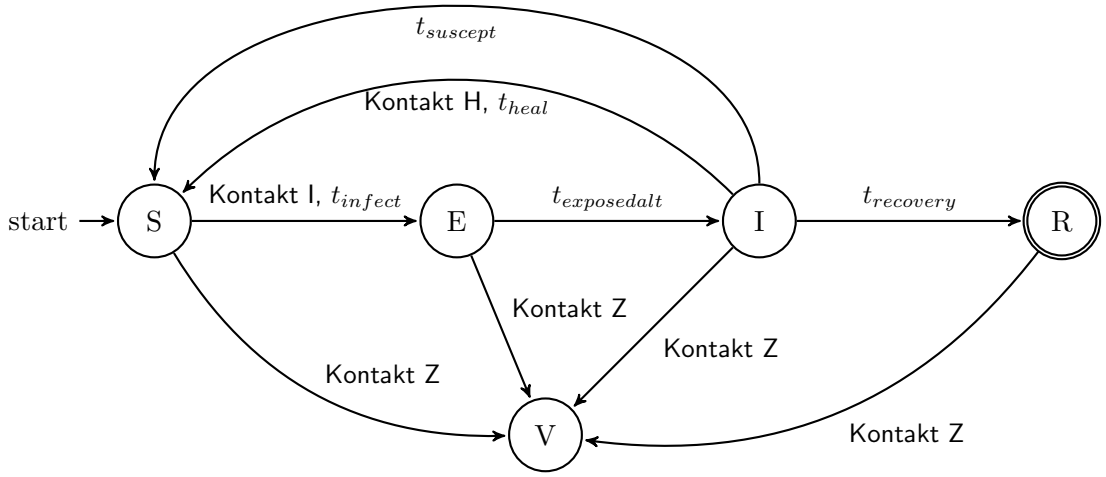


Abbildung 1: Übergänge der Infektionszustände der Seuchenfunktionalität

Die veränderbaren Parameter sind in Network_Beacon/main.h defaultmäßig wie folgt eingestellt:

- #define LIMIT_RECOVERY 21600 // Time to Recovery in seconds 6h (SIR-Modell)
- #define LIMIT_LATENCY 5400 // in seconds 1500 / Suggestion 30 min
- #define LIMIT_HEAL 300 // Time to heal in seconds TODO 300
- #define LIMIT_SUSCEPT 1400000 // Time to suscept in seconds TODO (SIS-Modell)
- #define LIMIT_RESET 3600 // immunity after reset in seconds 1800
- #define LIMIT_RSSI -80 // entspricht ca. 1-2m Abstand

3.3 Auslesen der Daten

Das Auslesen der Daten erfolgt über die Firmware im Ordner Network_Base. Diese wird auf ein DK augespielt das über ein USB Kabel an einen PC angeschlossen ist. Die Anwendung scannt laufend nach allen Paketen von DSA-Beacons. Falls der Datenstand des Ringspeichers für die sozialen Kontakte über dem eingestellten limit ist, wird eine Connection aufgebaut und folgendes ausgelesen:

- Aktueller interner Timer
- Vollständiger Seuchen Ringspeicher

Kürzel	Name	ID	Beschreibung	LED-Farbe
S	suszeptibel	0	gesunder Träger	grün
E	exponiert	1	infizierter Träger, noch ohne Symptome und nicht infektiös	grün
I	infiziert	2	infizierter Träger, der andere infizieren kann	rot
R	recover	3	gesunder Träger, der überlebt hat und nun immun ist (alternativ tot)	blau
V	vaccinated	4	Geimpfte, immun gegen Infektion	blau und grün
VT	vaccinated temporary	5	Temporär geimpft	blau und grün
H	Healer	6	Heilt Status I zu Status S	blau und rot

Tabelle 1: Beschreibung der Infektionszustände der Seuchenfunktionalität

- Netzwerk Ringspeicher ab dem letzten Auslesen

Die Daten werden anschließend über die UART Schnittstelle bereitgestellt und können dort ausgelesen werden. Bewährt hat sich unter Windows hierbei ExtraPutty, da dies mit Zeitstempel loggen kann. Die Verbindung erfolgt über COM. Unter Linux kann das folgende Shell-Skript verwendet werden. Dieses muss mit root-Rechten ausgeführt werden um von `/dev/ttyACM0` lesen zu dürfen.

```
#!/bin/bash
stty -F /dev/ttyACM0  115200
stdbuf -i0 -o0 -e0 \
  awk '{print strftime("%Y-%m-%d %H:%M:%S",systime()) " " $0}' /dev/ttyACM0 \
  | tee -a /beacon.log
```

Das Suchen nach Beacon startet auf dem DK erst nach Druck auf Button 1. Button 2 deaktiviert das Auslesen wieder. Dies soll verhindern, dass Daten ins Nirvana geschickt werden, falls noch kein PC logbereit ist. Aktuell scheint das Auslesen nur 2h lang aktiv zu sein, danach beendet "etwas" das Auslesen, Vermutlich wird die Verbindung vom PC deaktiviert. Dies sollte noch untersucht werden.

3.4 Datenformat der ausgelesenen Daten

In der Logdatei finden sich im Wesentlichen Zeilen von drei verschiedenen Sorten.

- Generelle Informationen zu dem Beacon. Diese Zeile wird direkt am Anfang des Auslesens einmal übertragen. Das Format ist `General: ID, STATUS, TIME`.

  Dabei ist `ID` die ID des Beacons. Weiter ist `STATUS` die folgende Kombination: `status_infect | (status_batt << 4) | (inf_rev << 5)` wobei `status_infect` der Zustand im Seuchenmodell ist (mit Werten in $[0, 6]$), `status_batt` angibt ob die Batteriespannung fällt (1 falls die Spannung niedrig ist und 0 sonst) und `inf_rev`

die Revisionsnummer des Seuchenmodells, mit dem der Beacon zuletzt programmiert wurde (mit Werten in $[0, 7]$). Schließlich ist `TIME` der Zeitzähler, der die Sekunden seit Einschalten des Beacons angibt.

- Angaben zu Zustandsübergängen im Seuchenmodell. Das Format der Zeile ist `Infect: ID, STATUS, TIME, SOURCE`.

  Dabei ist `ID` die eigene ID und `STATUS` der neu angenommene Status. In `TIME` steht der Zeitstempel (in Beacon-lokalen Sekunden seit Einschalten) des Zustandswechsels. Schließlich ist `SOURCE` eine Bitmaske der zum Zeitpunkt der Zustandsänderung in Reichweite befindlichen Beacons; diese gibt in 15 Gruppen zu je 8 Bit mit einer 1 die Anwesenheit an (andernfalls 0).

- Aufzeichnung des sozialen Netzes. Das Format ist `Netz: ID1, ID2, START, DURATION`.

  Dabei ist `ID1` die eigene ID und `ID2` die ID des gesehenen Beacons. Weiter ist `START` der Zeitpunkt des Starts der Interaktion (in Beacon-lokalen Sekunden seit Einschalten) und `DURATION` die Dauer des Kontakts in Sekunden.

3.5 Auswertung mit MATLAB

Das geloggte Skript kann mit dem Matlab Skript Log_Auswertung verarbeitet werden. Das Skript erzeugt eine CSV-Datei in der die sozialen Kontakte im Format (Node_start,Node_end,Time_start,Time_end) aufgezeichnet werden Zusätzlich wird ein .mat Datei erzeugt in der zum einen aus den Kontakten der entsprechende zeitabhänge Graph erstellt wird und zum anderen für jeden Knoten relevante Daten notiert werden. Diese Daten umfassen zum einen die Infektionsdaten, zum anderen Daten wie aktueller Timer, letztes Auslesen, Anzahl der Reset.

Die Funktion Aufräumen säubert doppelte Daten und Überschneidungen. Aufgrund der großen Datenmengen kann dies einige Zeit in Anspruch nehmen.

Die .mat Datei kann im Skript Seuchen_GUI eingelesen werden. Hier kann der zu betrachtende Zeitabschnitt gewählt werden und verschiedene Plot erstellt und abgespeichert werden. Zusätzlich kann eine rudimentäre Simulation auf dem bekannten Graphen ausgeführt werden. Es kann eine gexf Datei für Gephi erstellt werden, die den zeitlichen Verlauf des Netzwerks und den Verlauf der Seuche grafisch anschaulich macht.

3.6 Mögliche Erweiterungen

- Generelles Aufräumen und Optimieren - Done
- Stromsparmaßnahmen:
  - Ausführen des Codes aus dem RAM;
  - Verringerung der Paketgrößen und Übermittlung von weiteren Daten nur auf scanresponse.
- Speichern von veränderten Parametern im Flash

- ? Implementierung der Seuchenparameter im Sender, d.h. Sender übermittelt wie ansteckend seine Seuche ist.