Technische Dokumentation Sensoren

Maurizio Tidei, contexagon i.A. Fasnachtsmuseum Schloss Langenstein
(maurizio.tidei@contexagon.com)

Dieses Dokument ist entstanden im Verbundprojekt museum4punkt0 – Digitale Strategien für das Museum der Zukunft, Teilprojekt M4 Kulturgut Fastnacht digital, Modul 4. Weitere Informationen: www.museum4punkt0.de

Fasnachtsmuseum Schloss Langenstein

März 2020

museum4punkt0

Gefördert durch:

Die Beauftragte der Bundesregierung für Kultur und Medien

aufgrund eines Beschlusses des Deutschen Bundestages

Lizenz und Hinweis zur Nachnutzung

Change Log

Date	Version	Author
13.12.2019	0.1 Initiale Version	MT
31.03.2020	1.0 Finalisierung	MT

Inhaltsverzeichnis

1. Einleitung

Die Umsetzung des Moduls 4 «Personalisierte, interaktive und individualisierte Museumsführungen in sensorischen Räumen» bedarf diverser Sensoren, um die Präsenz der Besucher*innen in einem Raum bzw. an einer Station zu ermitteln und sie wiederzuerkennen. Dazu nutzen wir kleine Bluetooth Sender, die in 3D gedruckten Masken eingelassen sind. Die Besucher*innen erhalten diese Masken an der Museumskasse und führen diese über den gesamten Besuch mit.

Für die Erkennung der Bluetooth Sender werden stationäre Bluetooth Sensoren benötigt, die Anhand der eindeutigen Kennung und der Signalstärke ermitteln können, ob ein*e Besucher*in präsent ist und ob er oder sie durch Auflegen der Maske ein interaktives Terminal aktiviert hat.

Stationen, die auch unabhängig eines Bluetooth Senders getriggert werden sollen, arbeiten mit einem Bewegungs- bzw. Präsenzsensor.

Dieses Dokument beschreibt die Hard- und Software, die für die Umsetzung dieser zwei Sensortypen verwendet wurde.

2. Eingesetzte Technologien

2.1. Hardware

Der Raspberry Pi ist ein kompakter, kostengünstiger und energieeffizienter Mini-PC, auf dem ein Linux Derivat zum Einsatz kommt. Er ist mit einem Ethernet-Port und einem Bluetooth Chip ausgestattet. Aufgrund dieser Eigenschaften ist er ideal dazu geeignet, um als Hardwarebasis für unsere vernetzten Sensoren zu fungieren.

Außerdem verfügt der Raspberry Pi über eine GPIO-Schnittstelle (General Purpose Input Output), so dass eine Vielzahl weiterer Hardware-Sensoren verbunden und ausgelesen werden können.

2.2. Software

Als Programmiersprache und -plattform kommt Python zum Einsatz. Das Betriebssystem ist Raspbian. Die für den jeweiligen Sensor benötigten Software-Bibliotheken sind in den folgenden Kapiteln und im Sourcecode dokumentiert.

3. Softwarearchitektur und -design

3.1. Zielsetzung

Die Programme müssen auf einem Raspberry Pi unter Raspbian lauffähig sein. Die Kommunikation mit dem Server soll über das Ethernet/WLAN erfolgen.

3.2. Softwaremodule

3.2.1. Allgemeines

Alle Sensoren kommunizieren mittels einer REST API mit dem Server. Darüber werden Sie initialisiert und melden Ereignisse an den Server. In festen Zeitabständen wird außerdem ein Heartbeat-Signal an den Server gesendet, damit die Bereitschaft aller Sensoren im Server überwacht werden kann.

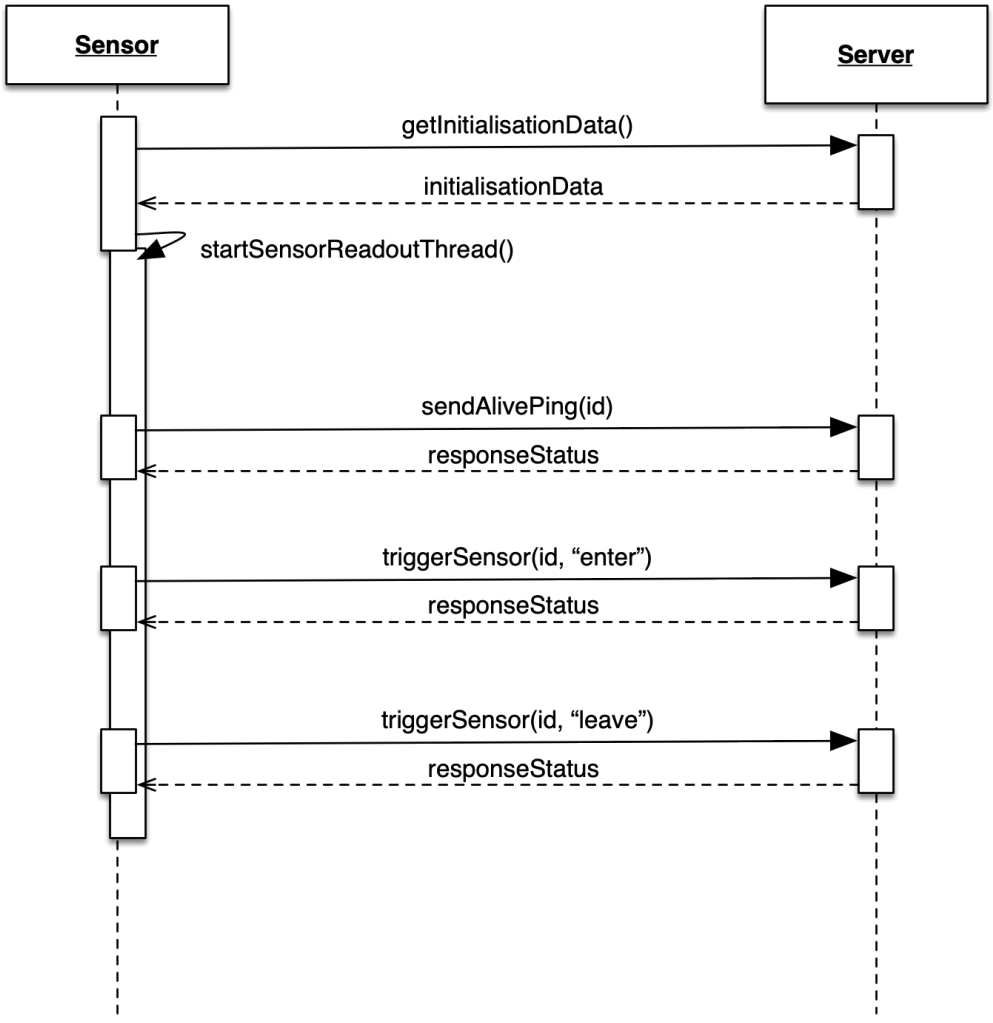

Abbildung 1 - Sequenzdiagramm Client-Server-Kommunikation

Der Quellcode aller Sensoren ist für die Nachnutzung im Projekt-Repository hinterlegt.

3.2.2. Bluetooth iBeacon Sensor

Der Raspberry Pi hat einen Bluetooth 4.0 fähigen Chip onboard, so dass keine separate Hardware benötigt wird, um iBeacon Signale zu empfangen und auszuwerten.

Für Python müssen einige Bibliotheken installiert und konfiguriert werden. Diese können unter Raspbian per apt heruntergeladen und installiert werden. Das folgende Shellskript enthält alle dazu nötigen Befehle:

```
#!/usr/bin/env bash

echo "Installing required bluetooth libraries..."
sudo apt-get update
sudo apt-get install bluetooth
sudo apt-get install bluez
sudo apt-get install python-bluez

sudo apt-get install python-dev libbluetooth-dev libcap2-bin
sudo setcap 'cap_net_raw,cap_net_admin+eip' $(readlink -f $(which python))
pip3 install beacontools[scan]
sudo pip3 install beacontools[scan]


echo "Installing beaconsensor service..."
sudo cp beaconsensor.service /etc/systemd/system/
sudo systemctl daemon-reload
sudo systemctl enable beaconsensor.service
```

Das Python Skript beaconsensor.py enthält die Programmlogik, um auf iBeacon Signale zu horchen und bei der Überschreitung definierter Schwellen ein „Enter“ oder „Leave“ Ereignis an den Server zu schicken. Alle Funktionen zur Kommunikation mit dem Server sind in der Datei servercommunication.py ausgelagert.
Die Serveradresse und der Sensorname werden in der Datei beaconsensor.ini definiert.

Das Programm wird unter Linux als Dienst registriert, so dass es bei Systemstart automatisch ausgeführt und im Falle eines Absturzes neu gestartet wird.

3.2.3. Bewegungs-/Präsenzsensor

Als Bewegungs/Präsenzsensor kommt wahlweise ein PIR (Passive Infrared) oder ein Ultraschall-Sensor zum Einsatz, der mit den GPIO Pins des Raspberry PI verbunden wird.

Die Konfiguration erfolgt über die Datei sensor.ini:

```
[sensor]
; Configuration for sensor.py

; REST API URL
;url = http://192.168.1.52:4000/trigger/sensor/motion1
url = http://192.168.2.222:46000/trigger/sensor/motion1

; Time between retriggering in seconds
time = 10

; Mode PIR or DIST
mode = PIR

; DIST ONLY Trigger if distance falls below x cm
distance = 50

; DIST ONLY PIN Number Trig
trig = 15

; DIST ONLY PIN Number Echo
echo = 16

; PIR ONLY PIN Number PIR
pir = 18
```

Darin können u.a. die GPIO PIN Nummern definiert werden, mit denen die Hardware-Sensoren verbunden sind. Außerdem lässt sich das Triggerverhalten des Sensors über die Parameter „distance“ (nur Ultraschall) und „time“ steuern.

Die Kommunikation mit dem Server erfolgt analog zum iBeacon Sensor.

4. Technischer Ausblick

Auf der Basis der gewählten Infrastruktur und der bereits umgesetzten Sensoren können beliebige weitere Sensortypen in das System integriert werden. Dank der GPIO Schnittstelle kann der Raspberry PI mit vielen Hardwaresensoren kommunizieren, die beispielsweise als Trigger für auf die Besucher*innen zugeschnittene Medien und Texte genutzt werden können.