Projekt 3

# Allgemeines

Wir benötigen eine Universal-Fernbedienung für IR, 433MHz und GET/POST Anfragen. Weitere Steuerungsformen sollen später mit möglichst wenig Aufwand zu integrieren sein.

# Hardware

Die Grundlage wird ein ~~Banana~~-Pi Ein-Platinen Computer sein. Dazu ein entsprechender Touchscreen. Je ein 433MHz Sende- und Empfangs-Modul, eine IR Diode und ein IR Transistor

Nachtrag: Um dem Entwickler etwas entgegen zu kommen wechseln wir auf den Raspberry Pi, da dieser etablierter ist als der Banana Pi.

# Eingabe/Frontend

Als Interface steht ein Touchscreen zu Verfügung. Die Benutzeroberfläche soll als Design Paket mit möglichst wenig Aufwand austauschbar sein und aus benötigten Bildern und einer json-Datei (index.json) bestehen, die die Position und Funktion der einzelnen Elemente angibt. Auch sollen mehrere ‚Fenster‘ unterstützt werden zwischen denen hin und her gewechselt werden kann.

Die dynamischen Fensterinhalte müssen dabei **nicht** zwischengespeichert werden.

Wird länger als 60 Sekunden keine Eingabe gemacht, soll das Display sich abschalten. Der erste Klick auf den deaktivierten Touchscreen soll nur den Touchscreen einschalten, aber keine Objekte triggern.

Beim starten des Programms wird automatisch das main-Menü im Vollbild angezeigt. Das Menü enthält Objekte wie im unten gezeigten JSON-Objekt angegeben.

Enthält ein Objekt einen goto-Eintrag, so wird beim antippen der komplette Bildschirm gelöscht und mit den Objekten gefüllt, die unter dem neuen Menünamen angegeben sind. Der neue Menüname steht in dem goto-Eintrag des angeklickten Objekts

{  
 „main“: //Menüname  
 [ //Objektliste  
 {  
 „id“: 1,  
 „posx“: 50,  
 „posy“: 200,  
 „width“: 100,  
 „height“: 100,  
 „text“: „Exit“,  
 „color“: „#fff“,  
 „beep“: „beep.wav“,  
 „exec“: „shutdown.py“,  
 „answer“: 2  
 }  
 ,  
 …  
 ],  
 „music“:[…], //  
 …  
}

Folgende Attribute gibt es für ein Objekt

## id

Eindeutiger Objektname

## posx, posy

x und y Position des Objektes auf dem Touchscreen

## width, height

breite und höhe des Objekts

## img

Hintergrundbild des Objekts

## text

Text des Objektes

## goto

wenn Objekt geklickt wird wechselt das Frontend die Ansicht zum angegeben Fensternamen

## color

Textfarbe

## bgcolor

Hintergrundfarbe

## beep

Eine Sounddatei die abgespielt wird, wenn das Objekt angeklickt wird

## exec

Befehl oder Script das ausgeführt werden soll wenn das Objekt angeklickt wird

## answer

ID des Objektes, in das der Standardoutput (STDO) des in exec(s.o.) ausgeführten Scripts geschrieben werden soll

# Funktionen

Neue Funktionen sollen schnell integriert werden können. Startet ein Frontend eine nicht existente Funktion, soll ein standardisiertes Fehlerfenster angezeigt werden, welche Funktion nicht existiert.

Die Folgenden Funktionen sollen gespeichert sein:

## send433

Sendet mit dem 433MHz Modul einen ein-/ausschalt Befehl.

## sendIR

Sendet per IR-Diode einen Befehl.

## sendGet

Sendet einen get-Request per Netzwerk und liefert den Status-Code als STDO zurück.

## sendPost

Sendet einen post-Request per Netzwerk und liefert den Status-Code als STDO zurück.

## getData

Holt Daten von einer Netzwerkadresse und verarbeitet sie durch Regex und liefert das Ergebnis als STDO zurück.

## shutdown

Fährt das System herunter.

# Arbeitsanweisung

Vor Beginn sollte das Projekt in bis zu maximal dreitätige Unterprojekte zerlegt werden.

Alle Unterprojekte sollen folgende Fragen behandelt werden:

* Was sollte erreicht werden?
* Was wurde erreicht?
* Warum habe ich mich für diesen Lösungsweg entschlossen?
* Welche Erkenntnis habe ich gezogen?
* Warum habe ich länger/nicht so lange benötigt wie ich ursprünglich geplant habe?

Selbstgeschriebener Code ist ins gitlab zu laden und dort auch aktuell zu halten, für externe Pakete ist ein link und eine Installationsanleitung zu schreiben

# Empfehlung

Ich würde empfehlen, mit Python und der Erweiterung Pygame für Frontend, Signaltöne und Toucheingaben zu arbeiten.

|  |  |  |
| --- | --- | --- |
| Datum | Verfasser | Grund |
| 29.01.2020 | stitz | Raspberry ist günstiger und mehr für Medienanwendungen ausgelegt |
| 03.02.2020 | stitz | Genauere Spezifikation des Aufbaus |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |