Mobile Systeme / ITS – LED-Rotor

Nora Panhuis (2202136), Friedrich Schimmel (1995911), Ann-Kathrin Schaack (1945062), Tobias Schweisfurth (2169629)

Betreuer: Prof. Dr. Torsten Edeler, Prof. Dr. Andreas Plaß

14. Juli 2016

Media Systems (B.Sc.)

Medientechnik (B.Sc.)

Hochschule für angewandte Wissenschaften Hamburg /
Hamburg University of Applied Sciences

Department Medientechnik
Fakultät Design, Medien und Information

Inhaltsverzeichnis

Projektidee

Der LED-Rotor stellt als eine Art des POV-Displays einzelne LED Zustände in hoher Frequenz dar, die für das menschliche Auge zu einem Text oder Bild verschmelzen. Dem Nutzer soll es ermöglicht werden, den Text oder das Bild während der Laufzeit über eine Webapplikation zu verändern.

Inspiration

Den Effekt der Nachbildwirkung (persistence of vision) auszunutzen, haben sich schon viele Projekte zur Aufgabe gemacht. Grundsätzlich geht es immer darum, dass mehrere diskret voneinander unterscheidbare Bilder bei genügend schneller Bildfrequenz für den Menschen zu einem Bild verschmelzen. Dies ist die physiologische Grundlage der Laufbildfotografie und somit des Films und des Fernsehens.

Projekte, die diesen Effekt ausnutzen, sind beispielsweise der POV-Globe, POV-Wand und das POV-Display. Gemeinsam haben sie alle, dass sie mit einer voreingespeicherten Nachricht oder einem voreingespeicherten Bild starten und dieses während der Laufzeit nicht mehr verändern.

Genau dieser Nachteil soll mit diesem LED-Rotor, welcher auch eine Form des POV-Displays darstellt, geändert werden.

Planung

Grundsätzlich gibt es zwei Möglichkeiten einen LED-Rotor dieser Art zu betreiben. Die erste Variante ist ein Zusammenspiel zweier Mikrokontroller, wobei einer auf dem Rotor selbst verbaut ist und der andere stationär danebenliegt. Dabei sorgt der Mikrokontroller, der auf dem Rotor verbaut ist, dafür, dass die LEDs in der richtigen Frequenz an- und ausgeschalten werden. Weiterhin muss er in der Lage sein, Daten von dem zweiten Mikrokontroller zu empfangen. Der zweite Mikrokontroller sorgt für die Datenverarbeitung. Er empfängt Bilder oder Texte vom Server und muss die wichtigsten Informationen herausfiltern, die an den ersten Mikrokontroller auf den Rotor weitergesendet werden.

Die zweite Variante teilt die Aufgaben nicht auf zwei Mikrokontroller auf. Ein einziger stationärer Mikrokontroller sorgt sowohl für die Datenverarbeitung, als auch für die richtige Steuerung der LEDs.

In beiden Fällen benötigt man eine Art der Datenübertragung, die es ermöglicht, dass der erste Mikrokontroller bzw. die LEDs die Daten bei Rotation erhalten können. Zur Auswahl stehen dabei Funk- oder Lichtübertragung oder Schleifkontaktringe.

Tatsächlich

Für das Projekt wurde sich für die erste Variante, die in der Planung angesprochen wurde, entschieden. Als stationärer Mikrokontroller kommt der RaspberryPi zum Einsatz. Auf dem Rotor wird ein Arduino Nano verbaut, der mit seinen kleinen Maßen für diese Aufgabe überzeugen konnte. Für die Datenübertragung zwischen den beiden Mikrokontrollern wurde sich für Funk entschieden. Dafür wurde ein Funkempfänger zusätzlich auf dem Rotor verbaut und der RaspberryPi mit einem Funksender verbunden.

Technik

Neben dem Arduino sind auf dem Rotor zwei Expander (I^2C), ein Funkempfänger und eine Spannungsversorgung mit Schalter verbaut. Die LEDs sind dabei an die Expander angeschlossen (8 pro Expander, obwohl 16 pro Expander möglich wären).

Die Spannungsversorgung führt zum 5V Eingang des Arduino und kann mit einem Schalter unterbrochen werden. Dieses war vorgesehen, um die Knopfzellen nicht unnötig zu entladen.

Auf der Rückseite des Rotors ist ein Reed-Kontakt angebracht, der an einem Input-Pin des Arduino angeschlossen ist. Dieser setzt einen Pegel auf HIGH, wenn er in die Nähe eines Magneten kommt. Der Magnet ist über dem Motor auf dem Gerüst angebracht. Mit dieser Konstruktion ist es möglich, die Zeit, die pro Umdrehung benötigt wird, zu messen. Mit dieser Zeit wiederum kann eine genaue Steuerung der Frequenz zum An- und Ausschalten der LEDs vorgenommen werden.

Zuletzt ist der Arduino mit dem Funkempfänger verbunden. Dieser empfängt auf einer Frequenz von 433MHz das Signal vom RaspberryPi.

Software

Die Software lässt sich in zwei Bereiche aufteilen: Software, die für jegliche Arbeit rund um Server und Website zuständig ist, und Software, die für das Anzeigen des Motivs auf dem Arduino zuständig ist. Dazwischen sind noch jeweilige Python-Programme, die auf dem RaspberryPi liegen und für die Vermittlungsarbeit zuständig sind.

Server

Der Server wird mithilfe des Frameworks Node.js geschrieben. Speziell liegt dabei der Fokus auf SocketIO, damit mehrere Nutzer zeitnah Aktionen ausführen können, die direkt übernommen werden.

Nach Anforderungen des mobile.mt.haw-hamburg-Servers ist der Server in die Unterordner „server“ und „public“ aufgeteilt worden. In „public“ liegen alle HTML-, CSS- und JavaScript-Dateien, die für die Webapplikation benötigt werden und für das Aussehen der Website zuständig sind.

Die Website sorgt für die Auswahl eines voreingespeicherten Motivs, welches über Mehrheitsentscheid zu festen Zeitpunkten ausgewählt wird. Der User hat hier über Checkboxen grundsätzlich die Möglichkeit mehrere Motive gleichzeitig auszuwählen, alle führen zur einer Erhöhung des Vote-Counters um 1.
Zu dem fest eingespeicherten Intervall – in der Praxis sollten es drei Minuten sein – wird ausgelesen, welches Motiv die meisten Votes erhalten hat und dementsprechend ein Code an den RaspberryPi versendet, welcher daraufhin das Signal an den Arduino weitergeben kann.

Auf dem RaspberryPi liegt dafür ein Python-Programm, welches über eine Socket-Verbindung auf ein Signal wartet und bei Erhalten eines Signals kurzzeitig den Motor stoppt und das Signal per Funk übertragt.

Website

Der Nutzer greift auf eine Website zu, die mithilfe von HTML und CSS aufgebaut ist. Sie ist dabei so aufgebaut, dass sie gut auf dem mobilen Endgerät darstellbar ist und alle Funktionen problemlos erreichbar sind.

Funktionalität wird über JavaScript hergestellt. Es wird eine Socket-Verbindung zum Server aufgebaut, woraufhin man auf bestimmte Signale reagieren kann.

Python-Programme

Mit den Daten, die vom Server kommen, kann der Arduino auf dem Rotor vorerst nichts anfangen. Sie müssen vorher noch verarbeitet bzw. komprimiert werden. Dies ist die Hauptaufgabe vom RaspberryPi.

Das Python-Programm für die Bildauswahl auf der Startseite nimmt über eine Socket-Verbindung einen Code entgegen, der stellvertretend für das Bild steht. Daraufhin stoppt er den Motor und wartet bis der Rotor aufgehört hat sich zu rotieren. Dieses geschieht über das Setzen eines Output-Pins, welcher mit der Motor-Steuerung verbunden ist, auf LOW. Dann kann der Code über Funk übertragen und der Motor wieder gestartet werden.

Arduino

Im Git-Repository gibt es mehrere Arduino Programme, deren Zweck einzig das Austesten verschiedener Funktionen ist. Dazu gehören Empfangen von Funksignalen, Steuerung der LEDs, die mit den Expandern verbunden sind, Testen von Timer-Interrupts, Geschwindigkeitserkennung über Reed-Kontakt und Darstellen eines eingespeicherten Musters.

Sie sind möglichst so aufgebaut, dass man sie zum Schluss miteinander verbinden kann, sodass das fertige Programm entsteht.

Grundsätzlich laufen zwei Aufgaben parallel zueinander ab: In der main-Loop des Arduino wird der Input-Pin überprüft, der mit dem Reed-Kontakt verbunden ist, und daraufhin eine Umdrehungszeit ermittelt. Außerdem werden in der main-Loop mögliche Funksignale entgegengenommen.
Parallel dazu läuft ein Timer, der zu festen Zeitpunkten, welche unabhängig von der main-Loop sind, ein Interrupt auslöst, auf welchem eine vordefinierte Funktion gelegt wird, die daraufhin ausgeführt wird. Diese Funktion steuert die Output-Pins des Expanders und somit das An- und Ausschalten der LEDs abhängig von dem momentanen Winkelschritt.