IoT - Online Quiz

Internet of Things Sommersemester 2021

Jonas Behling, Fabian Lignitz, Lucien Siani

Projektidee

Unser Projekt soll es ermöglichen, dass mehrere Teilnehmende zusammmen an einem Online-Quiz teilnehmen. Bei diesem Quiz spielen die Spieler:innen gegeneinander und können über einen Buzzer mitteilen, dass sie die Antwort wissen. Je nachdem welche Person zuerst gebuzzert hat, darf die Antwort geben. Entsprechend bedurfte es für die Umsetzung unserer Idee einen Buzzer, welcher eine den Spieler:innen zugeordnete ID an das Spiel (den Spieleserver) überträgt und das Frontend für unser Spiel. Als Buzzer soll der ESP dienen, welcher mittels Taster ausgestattet die ID an das Backend überträgt. Der Spieleserver muss eine graphische (Web)Oberfläche besitzen, die Spieler:innen verwalten, Fragen anzeigen und die jeweiligen IDs der Buzzer auswerten und anzeigen. Zusätzlich ist es nötig, dass der Spielablauf bei allen Teilnehmenden synchron angezeigt wird.
Im Folgenden wollen wir darstellen, wie wir diese Idee in einem kleinen Prototyp umgesetzt haben.

ESP32-Client

Für unseren Buzzer, braucht es einen relativ simplen Aufbau, welcher grundsätzlich nur auf den Druck eines Tasters bzw. auf das Schließen eines Stromkreises reagiert. Dieser simple Aufbau hatte jedoch das Problem, dass das Auslösen oft durch einfache Berührungen des Controllers geschehen ist. Aus diesem Grund haben wir zusätzlich einen Pull-Down Widerstand eingebaut, somit wird der Taster zuverlässig wieder auf den Zustand 'LOW' gebracht. Darüber hinaus haben wir unseren Aufbau um eine LED erweitert, welche optisches Feedback während des Netzwerkaufbaues gibt und bei jedem Tasterdruck einmal aufleuchtet.

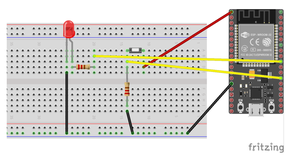

Abbildung 1: Aufbau ESP

Auf Softwareseite haben wir die Bestandteile der Veranstaltungen umgesetzt, so baut unser ESP eine Wifi Verbindung auf und kann OTA Updates einspielen. Darüber hinaus wird eine Verbindung zum Websocket hergestellt und eingehende Nachrichten werden an die Serielleschnittstelle übergeben. Wird der Taster betätigt, wird in einer Nachricht die MAC-Adresse des Gerätes an den Websocket übertragen. Leider mussten wir auf die Übertragung der Uhrzeit verzichten, da lediglich eine Zeitangabe in Sekunden möglich gewesen wäre, in unserem Fall ist dies nicht ausreichend.

Server und Programmablauf

Alle Clients müssen Informationen über den aktuellen Spielzustand erhalten. Damit die unterschiedlichen Clients miteinander kommunizieren können, eröffnet jeder Client eine Websocket Verbindung mit dem Backend-Server. Dabei wird eine bidirektionale Verbindung über das TCP-Protokoll aufgebaut.
Das Backend-System wird in der Docker-Infrastruktur der Hochschule Bremerhaven gehostet. Als Websocket-Server wird das schon vorinstallierte command-line Tool websocketd genutzt. Dieser Websocket-Server startet für jeden Client ein neues Shell-Skript, welches die vom Client eingehenden Nachrichten an alle anderen Clients verteilt. Um dieses Senden und Empfangen von Nachrichten zu realisieren wird die In-Memory Datenbank redis verwendet. Redis ist nicht nur eine NoSQL Datenbank, welche auch zur Speicherung von Spielerinformationen genutzt wird, sondern unterstützt den Publish und Subscribe Standard (kurz: pub/-sub).
Hierbei werden Daten prozessübergreifend an sogenannten Channels veröffentlicht (publish). Andere Prozesse können auf diesen Channeln lauschen bzw. abonnieren (subscribe). Außerdem wird mithilfe asynchronem Javascript und XML (englisch: Äsynchronous JavaScript and XML" , kurz: AJAX) Daten aus der Redis Datenbank nachgeladen. Dies passiert immer dann, wenn einmalig Daten gelesen, verifiziert oder geschrieben werden. Hierfür wird ein PHP-Skript geladen, welches die Verbindung zu Redis ermöglicht.
Gestartet wird die Applikation auf der client.html Seite. Auf dieser Webseite wird der Spieler:innen Name und die MAC-Adresse vom ESP32 eingegeben und validiert. Die MAC-Adresse muss mit der Doppelpunkt-Notation angegeben werden. Ist die Eingabe gültig, werden die Informationen in redis in einer Liste gespeichert. Da dies eine unsortierte Liste von Strings ist, wird MAC-Adresse und der Username zu einem String konkateniert und schließlich der Liste hinzugefügt.
Anschließend wird die Liste ausgewertet und auf der Webseite userfreundlich dargestellt. Somit werden alle schon angemeldete User angezeigt, aber um nicht ständig den Server mit Anfragen nach neuen Usern zu beschäftigen, wird der Redis Channel über den Websocket abonniert. Außerdem wird nicht nur nach neuen Spielern gelauscht, sondern auch auf den Spielbeginn.
Jeder Client kann den Spielbeginn auslösen, welches über den Redis-Channel jedem Client mitgeteilt wird. Ist dies der Fall, wird der Client auf eine andere Seite weitergeleitet. Auch diese Webseite stellt wieder eine Websocket Verbindung zum Server her. Nun beginnt der eigentliche Spielablauf. Jedes Mal wenn eine neue Runde gestartet wird, lädt der Server eine Frage aus der Quiz-API. Jeder Client liest anschließend diese Frage vom Server ein und stellt sie inklusiv der Antwortmöglichkeiten grafisch dar. Sobald eine der Spieler:innen, meint die Antwort zu wissen und der Taster gedrückt wird, veröffentlicht der Client seine MAC-Adresse auf dem Redis-Channel. Dies ist für die anderen Clients der Hinweis die Runde zu sperren und nach 5 Sekunden die richtige Antwort anzuzeigen. Sobald dies geschehen ist, können alle User eine neue Runde starten und der Ablauf beginnt erneut.

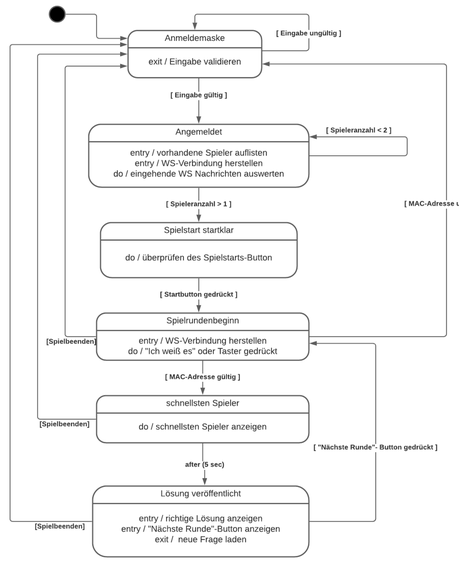

Abbildung 2: Zustandsdiagramm

Analyse und kritische Betrachtung der Ergebnisse

Abschließend lässt sich sagen, dass wir lediglich einen Prototypen bauen konnten. Der ESP übermittelt nicht die genaue Uhrzeit in Millisekunden und somit lässt sich nicht exakt auswerten, welche Person tatsächlich zuerst den Buzzer betätigt hat. Unsere Auswertung erfolgt lediglich auf Basis der zuerst ankommenden Nachricht. Zusätzlich findet zwischen Auswertung der MAC-Adresse und Anzeige im Frontend noch ein AJAX-Request statt, welcher die Ergebnisse verfälschen kann. Ebenso finden sich im Backend bzw. im Zusammenspiel von Backend und Frontend noch einige kleine Bugs, so funktioniert das nachträgliche Beitreten zum Spiel nicht optimal und die Antwortmöglichkeiten könnten als html oder Javascript Code interpretiert werden. Dennoch konnten wir unsere Projektidee umsetzen, ein Quiz welches über Buzzer gesteuert wird und über alle Clients hinweg synchron abläuft. Dabei direkt an der Hardware zu "basteln"hat uns sehr viel Spaß gemacht und gleichzeitig einen Einblick in die IoT Welt ermöglicht.