Gadget Entwicklung

Gesehen habe ich die Dinger während des Aufwärmtrainings der Roten Raben, als Timo Lippuner dort noch Cheftrainer war. 2, 3 blinkende Dinger, auf die die Libera reagieren musste, bevor sie einen Ball abwehren musste. „Will haben, aber was ist das?“. Die Startpage-Suche war nicht sehr ergiebig, und so rückten sie in den Hinterkopf, bis Melanie Cina im 22. Trainertalk davon erzählte. Blaze Pods heißen die Dinger, leider unerschwinglich. Also reifte ziemlich schnell die Idee: Selber bauen. Hier ist die Geschichte.

Designkonzept

Die Bedienbarkeit über Mobilgerät (Tablet, Handy, Android oder iOS, PC) erfordert eigentlich zwangsläufig eine Browserlösung mit Javascript. Das hat zwar ein paar hässliche Implikationen, weil die meisten Browserhersteller inzwischen der Meinung sind, lokal auf dem Gerät gespeicherte Dateien sind gefährlich, die von Servern aber nicht.

Von der Hardware Seite war gefordert: Robustes Gehäuse, integrierte Stromversorgung, WiFi oder BLE, leichte Programmierbarkeit. Bei der Wahl des Steuergerätes wurde mir der ESP32 empfohlen, im Rückblick eine gute Wahl. Ich habe mich für das Developmentboard entschieden, das erfordert nicht so ganz gute Lötfähigkeiten. Die ESP32 haben etliche IO-Pins, sind über C++ programmierbar, es gibt etliche Bibliotheken, und ganz wichtig: Viele Tutorials.

Hardwarekonzept

Als Gehäuse verwende ich formschöne HT-Rohr Doppelmuffen mit Deckeln aus dem Baumarkt, die in einem ansprechenden Dunkelgrau gehalten sind. Meine sind DN75, also 75mm Durchmesser, und etwa 110mm hoch. Für die Halterung der LEDs habe ich mit einer Lochsäge Scheiben aus einer alten Korklaminat-Bohle geschnitten, die noch bei uns im Keller rumlag. Gegenüber MDF-Platten hat die den Vorteil, nur 11mm dick zu sein. Mit einem 6mm Holzbohrer einen 9er-Kranz für die LEDs gebohrt und zentriert auf den einen Deckel geklebt. Um das zentrale Loch zu verdecken, wird das ganze noch mit einer Linsenkopfschraube und Mutter auf der Rückseite ergänzt. Der ESP32 kann auch Touch-Ereignisse reagieren, deswegen habe ich die Oberseite mit Alu-Folie beklebt. Mit der Linsenkopfschraube wird die elektrische Verbindung zur Oberseite hergestellt. Das war leider ein Detail, in dem das Internet mich auf die falsche Fährte gelockt hat. Alle Tutorials zeigen das mit einem kleinen Draht, prima, klapp super. Im Kleingedruckten habe ich dann später gefunden, wie das elektronisch funktioniert. Der Controller misst Kapazitätsänderungen. Leider funktioniert das mit dem von mir gewählten Aufbau nicht. Die Alufolie bildet keinen Kondensator, der sich durch die Anwesenheit eines Fingers ändert, sondern lediglich eine Antenne. Signal-to-Noise wird grottig. Da war die Folie aber schon drauf. Also habe ich die Linsenkopfschraube isoliert, sie ist die einzig sensitive Fläche, aber das war elektrisch OK.

Im Deckel ist umlaufend ein 1.5mm² Kabel, das später die Versorgungsspannung heranschafft. Die LEDs sind „fliegend“ verlötet, jeweils drei einer Farbe parallel. Meine Töchter konnte ich damit überraschen, dass ich die drei Anschlusslitzen+Touchleitung in einen Zopf flechten konnte. Der führt zu einer kleinen Lochrasterplatine, auf die drei Schalttransistoren mit Vorwiderständen und

Lastwiderstände gelötet sind. Der ESP32 kann eine LED treiben, aber keine 3, deswegen musste das hier entkoppelt werden. Die Transistoren sind stinknormale 2N2222. Eine 5-polige Buchsenleiste ermöglicht den steckbaren Anschluss an den ESP32.

Zur Stromversorgung habe ich eine DC-Einbaubuchse in die Zylinderwand der Doppelmuffe eingebaut. Vorgesehen ist ein 5V Netzteil, das etwa 1A liefern können sollte. Von der Buchse geht es auf eine Ladeplatine für 3,7V LiIon/LiPo Akkus. An die kommt auf der anderen Seite ein 18650 Li-Ion Akku mit 3.6V/2600mAh. Die letzten beiden Anschlüsse versorgen die restliche Schaltung mit Spannung. Das tun sie auch, während geladen wird. Wer in der Lage ist, SMD-Widerstände aus- und wieder einzulöten, kann den Ladestrom begrenzen. Netzteile, mit denen man 5V/4A abrufen kann, sind dünn gesät, Deswegen sind 250mA eigentlich schöner.

Fast fertig, einen Schönheitsfehler hat das Ganze noch. Der ESP32 kann mit 5V oder 3.3V versorgt werden. Bei den 5V ist er so semi-tolerant, bei den 3.3V eher pingelig. Deswegen wird das ganze noch mit einer weiteren Mini-Platine ergänzt. Auf der sind ein LDO MCP1700-3302E, ein 1000µF Elko und ein 100nF Keramikkondensator verbaut. Der MCP ist ein Low-Drop-Out Spannungsregler, der 3.3V liefert. Der Elko stabilisiert die Spannung, wenn der ESP32 gerade im WiFi funkt, da zieht er gerne mal über 200mA, der 100nF dämpft HF-Oszillationen. Komplettiert wird der ganze Aufbau durch einen einfachen Schalter neben der DC-Buchse, die die 3.3V auf den ESP durchschaltet.

Zum Einbau werden die Platine mit Isolierband umwickelt (ich hab vergessen Schrumpfschlauch zu bestellen 😕) und einigermaßen sicher mit Verpackungschips aufgefüllt. Wie robust das im Trainingsbetrieb ist, wird sich erst nach Corona zeigen.

Software auf dem ESP32

Als Entwicklungsumgebung habe ich Arduino mit der ESP-Erweiterung unter Linux genutzt. Im wesentlichen hat die Firmware auf den ESPs drei Funktionen:

1. Kontakt mit einem WLAN aufnehmen
2. Updates über die OTA entgegennehmen
3. Kommunikation mit den Clients und Management der LEDs und des Touch-Eingangs

Kontakt mit einem WLAN aufnehmen

Beim Start wird das WLAN zweigleisig initialisiert. Zum einen wird versucht, sich als Client im letzten bekannten WLAN einzuloggen. Gleichzeitig wird auch ein eigenes WLAN als Accesspoint aufgespannt. Das ist komplett offen. Falls das Einloggen als Client erfolgreich ist, wird der AP abgeschaltet. Falls der Client nicht innerhalb von 60s in ein WLAN einloggen kann, wird der Versuch aufgegeben. Der AP bleibt dann insgesamt 10 Minuten aktiv. In der Zeit kann man sich mit diesem offenen WLAN verbinden, und die URL http://192.168.4.1/scan aufrufen. Beim ersten mal sieht man, dass die Verbindung erfolgreich war, beim zweiten mal sieht man alle aktiven Netzwerke in der Umgebung. Von denen kann man sich dann eins aussuchen und konfigurieren. Ja, dass man diese Seite zweimal aufrufen muss ist ein Bug.

Updates über OTA

Eine großartige Funktion der ESPs, die leider erst spät entdeckt habe, ist das Update über WiFi. Den ESP einmal richtig einbauen, WLAN konfigurieren und gut. Der realisierte Code ist direkt Copy&Paste aus dem Internet. Es gibt noch eine Version, die besser auf den verwendeten Web-Server zugeschnitten ist, aber da war am Anfang nicht ganz klar, welche Teile für den kleinen Bruder (8266) und welcher für die ESP32 gilt. Wenn ich mal Zeit habe…

Kommunikation mit den Clients

Die Kommunikation erfolgt auf zwei Ebenen. Zum einen steht ein WebServer bereit, der statische Seiten serviert. Die Ursprungsidee war, dort nur die Konfiguration abzuhandeln und die eigentlichen LED-Programme lokal auf den Clients zu speichern. Das funktioniert auch prima mit dem FireFox auf dem Desktop, aber so gar nicht mit mobilen Clients unter Android (Safari unter iOS habe ich dann gar nicht mehr probiert…). Dort sind die Browserhersteller der Meinung, lokale Dateien sind viel gefährlicher als Web-Seiten, deswegen muss das unbedingt verboten sein. Kann man sich gar nicht ausdenken, so was.

Also braucht man entweder einen externen Web-Server oder die Seiten müssen auf den ESP. Wenn man als AP einen RasPi 4 nimmt, kann man auch noch gleich einen nginx oder lighttpd draufsetzen und gut. Das verkompliziert aber den Aufbau, also kommen die Seiten auf die Micros. Praktischerweise sind von den 4MB Speicher 1.2MB für ein Dateisystem reserviert (Das war früher man die Kapazität einer Floppy…). Die einzelnen Dateien werden „minified“ und gezippt. Beim Ausliefern muss lediglich noch der Header für „Inhalt kommt komprimiert“ gesetzt werden, das Entpacken übernimmt der Client auf dem PC/Smartphone.

Der Web-Server liefert HTML, Javascript und ein bischen CSS aus. Für die Kommunikation baut der Client eine Web-Socket Verbindung zu den ESPs auf, über die er Kommandos schicken kann.

Das einfachste Kommando ist „STATE“. Damit kommt als Antwort der Zustand der drei LED-Farben, der Zeit in Millisekunden seit dem Start des Micros, und der Wert des Toucheingangs. Die Farben werden als 0 und 1 ausgegeben, Eine typische Antwort wäre also „STATE001 231298 32“. Fast genauso sieht auch die Nachricht vom Micro aus, wenn eine Berührung erkannt wird: „TOUCH001 234124 21“.

Die weiteren Kommandos von Clientseite sind:

- setX, wobei X zwischen 0 und 2 liegt: Schaltet Farbe X an (Rot, Grün, Blau)
- SETxyz: x, y, z jeweils 0 oder 1: Setzt den Zustand Rot auf x, Grün auf y, Blau auf z.
- CLR: Schaltet alle Farben dunkel

Als Antwort schickt der Micro jeweils den neuen Zustand, Zeit und Toucheingang zurück.

Software auf dem Browser

Es braucht nicht viel, ein Browser, der JavaScript kann. Getestet habe ich mit dem Firefox unter Linux und Windows, Safari unter iOS, sowie Fennec und Bromite unter Android. Etwas nervig ist die Angewohnheit moderner Browser, `https` zu verlangen und bei jeder Gelegenheit darauf zu wechseln. Dieses an sich positive Privacy Feature brauchen wir an der Stelle nicht. In die Adresszeile muss man also explizit `http://esp00/portal.html` eingeben. Am besten als Bookmark. Etwas sperrig stellen sich manche Browser an, den der Web-Host mal ein Nickerchen macht, insbesondere auf den Seiten,

in denen die LEDs angesteuert werden. Am besten die ESPs aufwecken, zur Portalseite zurück gehen und neu laden.

Bislang gibt es drei Seiten, die man tatsächlich im Training gebrauchen kann:

demo01.html: Lässt den Trainer manuell jeweils eine LED ein- oder ausschalten. Gut zum Testen der Kommunikation und für Aufgaben, in denen es auf ein individuelles Timing ankommt.

demo02.html: Die LEDs der eingebundenen Micros werden in einer zufälligen Reihenfolge eingeschaltet und bleiben kurz an und gehen wieder aus. Los geht die Sequenz, wenn man auf Start drückt. Bis zum Aufleuchten der ersten LED dauert es die Startverzögerung plus einer zufälligen Zeitspanne von 0 bis jitter. Die Pause zwischen den nächsten Schritten dauert auch wieder Delay plus zufälliges Jitter-Intervall. An bleibt die LED für die „On time“. Alle Angaben sind in ms, wobei man die leider nicht so 100% ernst nehmen kann.

demo03.html: Macht fast das selbe wie demo02, aber die LEDs bleiben so lange an, bis der Touch-Interrupt ausgelöst wird. Die Dauer wird auf dem Client angezeigt.

Anwendung in der Praxis

Ich habe mir auf einen Raspi einen Accesspoint installiert, der mir im Training ein WLAN für die ESPs und mein Tablet aufspannt, und es kann losgehen. Soft- und Hardware haben beim Training zuverlässig funktioniert, aber es gab einen unschönen Effekt. Die LEDs sind etwas mager. Wenn es hell genug ist in der Halle, sind sie manchmal schwer zu sehen. Insbesondere die grünen LEDs waren vor unserem, leider grünlichen Hallenboden, schwer zu sehen (ich könnte schwören, dass der vor Corona grau war…). Deswegen sollte man beim Aufbau des LED-Rings darauf achten, dass die LEDs aus dem Deckel herausspitzen, ansonsten ist es wirklich schwer zu sehen. Außerdem sollte der Vorwiderstand für die drei grünen LEDs verringert werden, damit man nächer an die 20mA/LED herankommt, und die Dinger etwas heller leuchten. Rot und Blau funktionieren prima.

Ich habe mit den Mädels Reaktionstest gemacht (demo03), vier Stück im Quadrat, möglichst schnell berühren, hat gut funktioniert. demo02 war nicht so wirklich gut, weil das Grün schwer zu erkennen war. Die habe ich kombiniert mit der Aufgabe: Rot=Blocksprung, Blau=Pritschbewegung, Grün=Bagger mit Ausfallschritt. Ansonsten haben wir festgestellt, dass ein Quadrat von etwa 2x2m schon verdammt schwer im Auge zu behalten ist. Mir fehlen da noch die Vergleichswerte, und spannend ist, ob sich das im Laufe von weiteren Trainings verbessert.

Auf dem Plan habe ich noch, einen Spieler oder eine Halterung beim Einschlagen von der IV auf die I und die V zu stellen. Zwischen Anspiel zum Zuspieler und Schlag leuchtet einer der Pods auf, und entsprechend soll Longline oder Diagonal geschlagen werden. Aber da ist erst mal noch ausgiebiges Stellertraining vor…

Weitere Ideen von außerhalb nehme ich natürlich gerne auf.

From:
https://www.tvms-volleyball.de/volleywiki/ - **Volleyball Trainer-Wiki**

Permanent link:
https://www.tvms-volleyball.de/volleywiki/doku.php?id=gadget_entwicklung

Last update: **2021/06/13 14:28**