



# COOL-WEB

Cool Web.de ► Esp8266 Esp32 ► Esp8266 Mit Mt8808 8x8 Analog Witch Array Als Vc20 C64 Wlan Tastatur Emulator

† Werbung auf dieser Seite ist gestorben - Sie können Ihren AdBlocker abschalten †

Warum? (klicken Sie auf ein Dreieck, um einen Abschnitt aufzuklappen)

- 1. Werbung lohnt sich nicht mehr und wirft mittlerweile lächerlich wenig ab.
- 2. Werbung wird immer betrügerischer.
- 3. Werbung spioniert und ist eine potentielle Gefahr.
- 4. Werbung nervt die Besucher.

Darum wird es hier in Zukunft keine Bannerwerbung mehr geben. Die Mini-Einnahmen stehen in keinem gesunden Verhältnis zu den Nachteilen für die Besucher. Trotzdem muss sich die Site irgendwie finanzieren (ich habe leider kein dickes Bankkonto). Darum überlegen Sie bitte, ob Sie nicht vielleicht...

- Ihren **Adblocker** die für diese Seite **deaktivieren** wollen.
- Bei [amazon.de](#) über einen besonderen Link einkaufen wollen.
- Bei [eBay.de](#) über einen besonderen Link einkaufen wollen.

## ESP8266 mit MT8808 8x8 Analog Switch Array als VC20/C64-Tastatur-Emulator über WLAN

Like & Share

Das direkte Einbinden der Interaktion-Schaltflächen von Anbietern sozialer Netzwerke ermöglicht es diesen, Ihr Surfverhalten über alle Seiten, bei denen solche Schaltflächen eingebunden sind, zu protokollieren und auszuwerten. Wir haben uns darum entschlossen, diese Schaltflächen erst nach einem Klick auf den nachfolgenden Button zu aktivieren, um Sie so vor einer generellen Protokollierung zu schützen.

[Soziale Netzwerke anzeigen](#)

Beim letzten Umzug ist mein Blick auf einen alter Karton mit alter Hardware gefallen, darunter Motherboard zu einem 8086, einem Commodore **VC20** und einem **C64**.

Nachdem ich **VC20** und **C64** getestet hatte, befand ich, mit den beiden Platinen noch etwas anzustellen und ein bisschen in Retro-Nostalgie zu schwelgen.

Der C64 allerdings schien einen Defekt zu haben, denn er meldet beim Einschalten ein "Overflow Error in 0". Da habe ich einen Reparaturversuch des RAMs unternommen, der jetzt allerdings erstmal in eine "361 Basic Bytes Free"-Meldung pausiert.

Um weiter nach der Fehlerursache forschen zu können, müsste ich ein kleines RAM-Testprogramm auf dem C64 eingeben. Aber das ist ohne Tastatur natürlich schlecht. Ich könnte mir jetzt natürlich eine nachkaufen, aber auf [eBay](#) sind die gar nicht mal so billig und außerdem finde ich die Platinen nackt viel platzsparender und einen aufregenderen Anblick.

Außerdem war die Tastatur auf dem **VC20** und **C64** schon immer grauenhaft. Anständiges Zehn-Finger-Tippen kann man darauf vergessen, weil die Tasten so einen langen Hub und einen so schlechten Anschlag haben.

Viel schöner wäre es doch, die PC-Tastatur zu benutzen und die Eingaben über das WLAN an einen **ESP8266** zu funken, der die Tastendrücke dann an die Tastatur-Schnittstelle des VC-20/C64 weiterleitet.

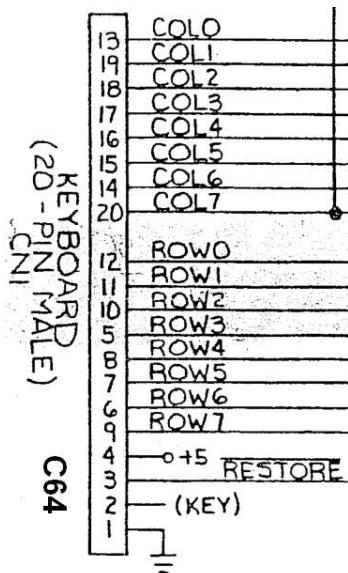
### Die Tastaturschnittstelle des VC20/C64

Dazu muss man wissen, dass die Tastaturschnittstellen von VC20 und C64 im Prinzip gleich sind:





(links VC20, rechts C64)



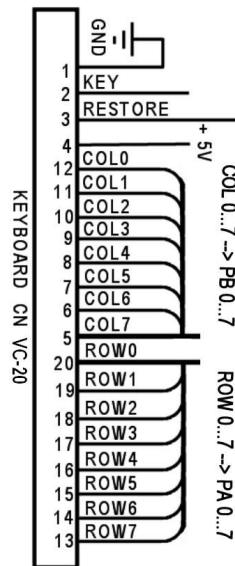
COL 0..7 → PA 0..7    ROW 0..7 → PB 0..7

Beide Keyboard-Konnektoren sind 20-polige Pfostenstecker, von denen 16 Leitungen in Port A und Port B in den Interface Adapter (beim VC20 in den 6522 VIA und beim C64 in den 6526 CIA) führen, wie a nebenstehenden Schaltplänen zu sehen ist.

- Pin 1 führt auf GND
- Pin 2 ist nicht belegt
- Pin 3 führt die Leitung für die RESTORE-Taste
- Pin 4 führt 5 Volt
- Pin 5 bis 20 führen die Leitungen für die 8x8 Schaltmatrix

Die Tastatur ansich ist ziemlich dumm und besteht nur aus einer Matrix von 8 Zeilen (Rows) zu 8 Spalten (Cols). Wird eine Taste auf der Tastatur gedrückt, werden einfach die 2 Leitungen, in dessen Kreuz die Taste liegt, verbunden. Das kann man sich wie das 4x4-Keypad vorstellen, dass ich schon einmal vorgestellt hatte, nur halt mit 8x8 statt 4x4 Tasten.

Für meinem frühen C64 (©1983, Assy No. 250407 Rev. B, Made in Hong Kong) passt der obige, linke C64-



Schaltplan übrigens nicht, sondern vielmehr der von VC-20 rechts. Hier ist der Tastatur Connector genau so durchgängig geschaltet wie beim VC20. Das Verbinden der Kabel bei meinem C64 führt genau zu den gleichen Ergebnissen wie beim VC20: Pin 10 und Pin 19 verbunden führen bei beidem zu einem 'A'. Einen anderen Schaltplan als den oben habe ich allerdings nicht gefunden.

Auf dem C64-Schaltplan sind übrigens zwei Leitungen zu erkennen, die nach oben herausführen. Diese führen zu Joystick-Port 1, der ebenfalls über den Tastatur-Konnektor abgefragt werden. Darum erscheinen auch Buchstaben auf dem Bildschirm, wenn man den Joystick bewegt. Übrigens sind beide Joystick Ports (je 4 Richtungen und Feuerknopf) beim C64 am Tastatur-Port angeschlossen. Um die Joysticks zu steuern, werden allerdings nicht Kreuzpunkte der Matrix miteinander verbunden, sondern eine ganze Zeile oder Spalte auf Ground gezogen. Für die Emulation von Joysticks bräuchte man also einen anderen (wenn auch einfacheren) Ansatz.

Um einen Tastendruck zu emulieren, müssen wir also einfach die beiden richtigen Pins verbinden und schon erscheint der Buchstabe auf dem Bildschirm. Mit einem Jumperkabel direkt am Konnektor kann man das einmal ausprobieren. Nur bitte nicht einen der unteren 4 Pins benutzen, die führen Strom!

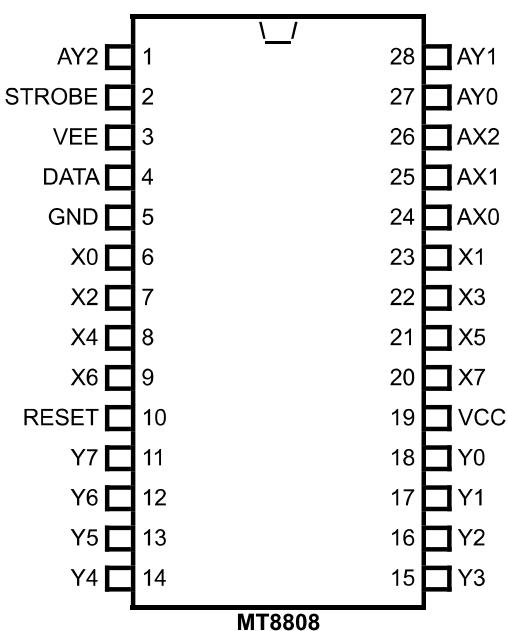
Manchmal muss man natürlich auch eine Taste gedrückt halten, etwa die Shift-Taste, und eine zweite drücken, um zum Beispiel ein großes A eingeben zu können.

Um Tastendrucke zu simulieren bräuchte wir also so etwas wie einen flexibel steuerbaren elektronischen Schalter, die Kreuzpunkte in einer Matrix verbinden und wieder trennen kann. Und auch Verbindungen beibehalten kann, bis sie per Befehl wieder getrennt werden.

### Der MT8808 Analog Crosspoint Switch

Der [MT8808](#) (8 x 8 Analog Switch Array) von Mitel (ab 2001 Zarlink Semiconductor, ab 2011 aufgegangen in Microsemi Corp.) leistet genau das. Er bietet eine 8x8 große Schalter-Matrix, bei dem jeder Kreuzpunkt (Crosspoint) verbunden werden kann. Es können Verbindungen gehalten ([Stichwort Latch](#)) werden und weitere Verbindungen hinzugefügt werden. Eine Verbindung wird erst wieder getrennt, wenn dazu der explizite Befehl kommt, oder wenn Reset ausgelöst wird; dann werden alle bestehenden Verbindungen getrennt. So lässt sich jedes beliebiges Schaltmuster aufbauen, z. B. auch alle Schalter bis auf einen eingeschaltet. Das bräuchte allerdings 63 Schaltvorgänge.

Kommen wir zur Anschlussbelegung des MT8808:



Der **MT8808** hat 28 Pins mit folgender Belegung:

- GND = Ground, VCC = positive Spannung, VEE = negative Spannung
- AX0...AX2 (Input) = X-Position der Verbindung bitkodiert (0...7)
- AY0...AY2 (Input) = Y-Position der Verbindung bitkodiert (0...7)
- DATA (Input) = Low für ausschalten, High für einschalten
- STROBE (Input) = auf High setzen, um konfigurierten Kreuzpunkt X/Y zu schalten
- RESET (Input) = auf High setzen, um alle Verbindungen zu lösen
- X0...X7 (Output) = horizontale Schalter in der Matrix (Cols)
- Y0...Y7 (Output) = vertikale Schalter in der Matrix (Rows)

Die Ansteuerung des **MT8808** funktioniert wie folgt:

- Setzen und halten von RESET auf low
- Setzen und halten von STROBE auf low
- Setzen der Datenbits AX0...AX2
- Setzen der Datenbits AY0...AY2
- Setzen des Schaltzustands in DATA (0=aus, 1=an)
- Strobe für min. 20 ns auf High pulsen
- in typ. 40 ns, max. 100 ns ist der neue Verbindungsstatus hergestellt

Ein Reset führt man gegebenenfalls durch, indem man die RESET-Leitung für min. 40 ns auf High pulst. Ein Reset löst alle Verbindungen. Ein Reset geschieht nicht automatisch nach dem Einschalten, sondern muss explizit zu Anfang ausgeführt werden, um einen definierten Schaltzustand (alles aus) zu setzen.

Der **MT8808** kann eine maximale Schaltfrequenz von 20 MHz leisten.

### Beispiel

Wir wollen Col 3 (X=3), Row 2 (Y=2) schalten. Was ist dafür zu tun?

Zuerst einmal müssen STROBE und RESET auf Low sein. Sonst funktioniert gar nichts.

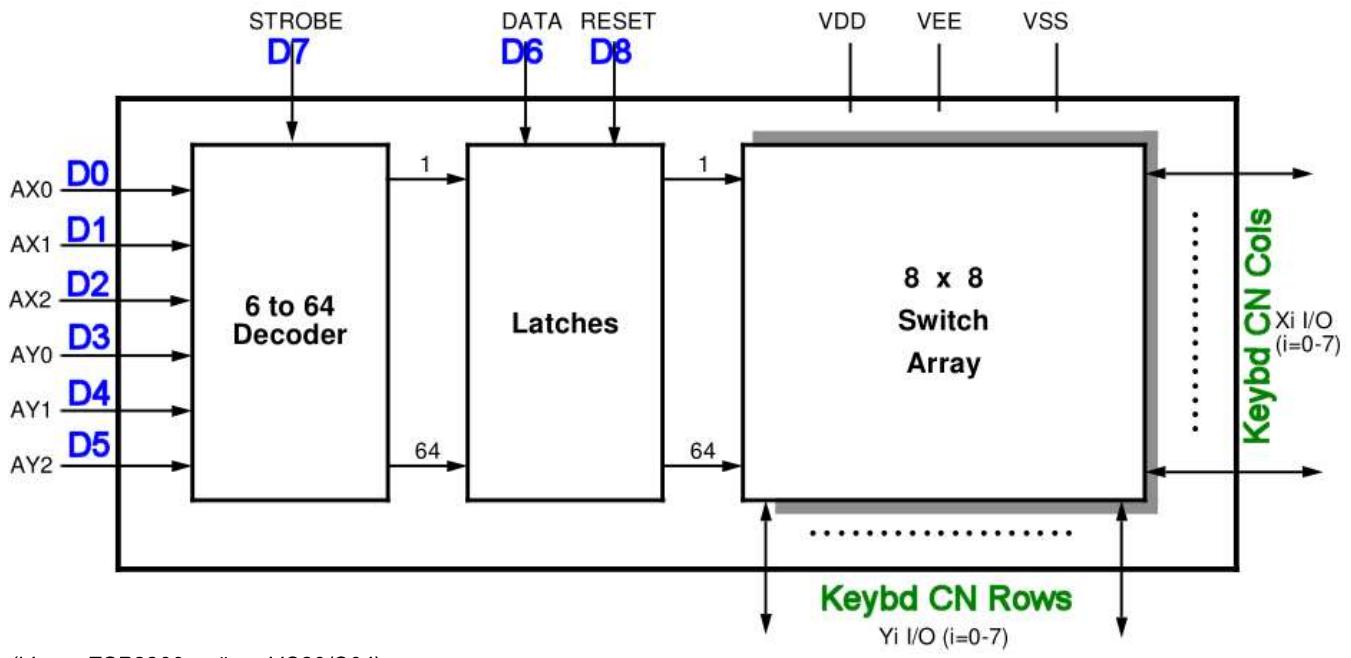
Laut dem Beispiel ist AX 3 (0b011) und AY ist 2 (0b010). Das binär (von rechts nach links) auf die A-Leitungen verteilt heißt: AX0=1, AX1=1, AX2=0; AY0=0, AY1=1, AY2=0.

Wir wollen den Kreuzpunkt verbinden, also setzen wir DATA auf 1 (high). Nun müssen wir nur noch einen STROBE-Puls senden und schon sollte der MT8808 die Verbindung herstellen an den Leitungen, die an X0...X7 und Y0...Y7 angeschlossen sind.

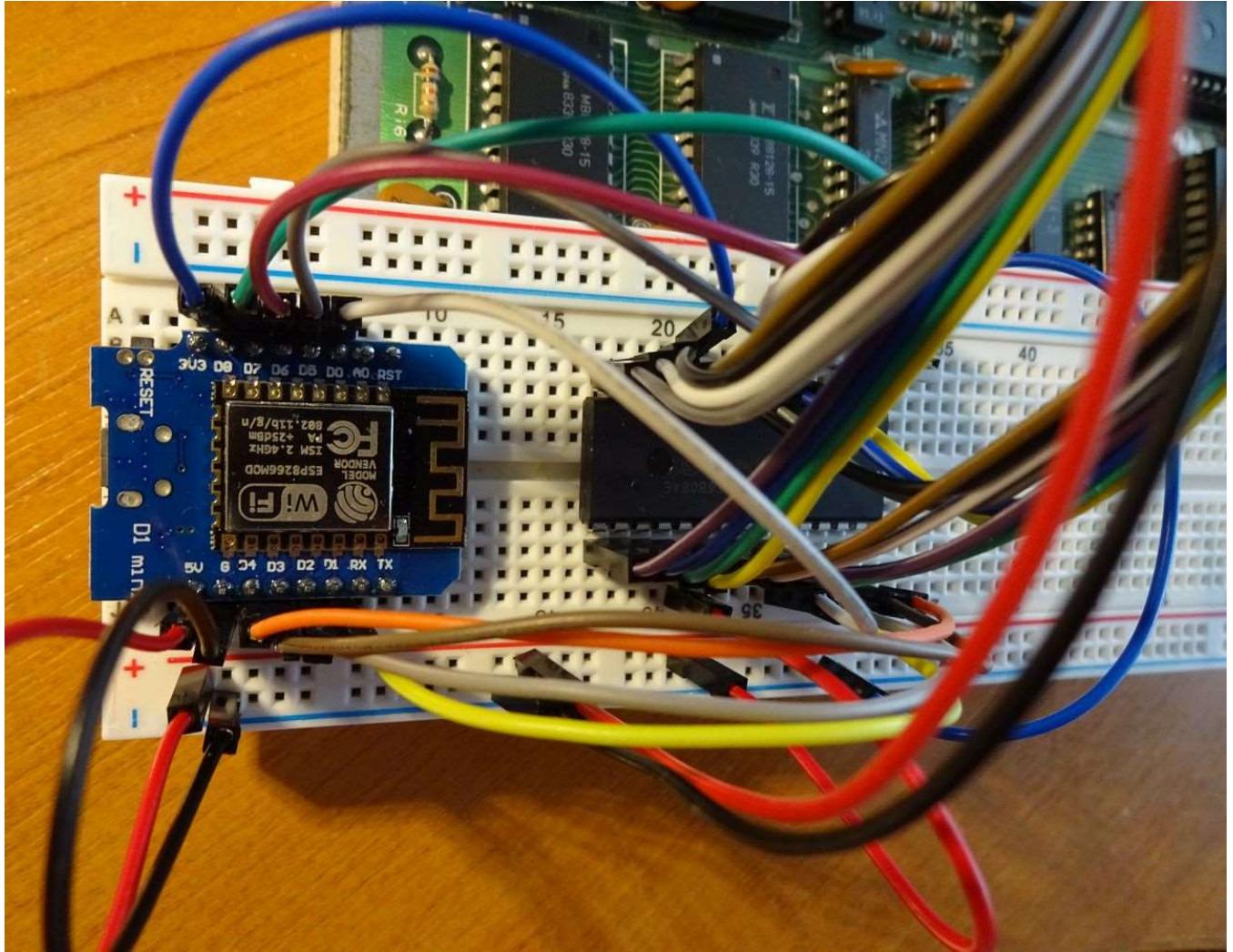
### Anschließen an einen ESP8266 D1 mini

Zur Steuerung des **MT8808** brauchen wir insgesamt 9 Leitungen (AX0...AX2, AY0...AY2, DATA, STROBE, Reset). Diese bietet genau der **ESP8266 D1 Mini** mit seinen neun D0...D8 GPIO-Ports, keinen zuviel und keinen zuwenig.

Die 16 Ausgänge des MT8808 X0...X7 und Y0...Y7 werden an den Tastaturport des C64/VC20 angesteckt. Und zwar X an Cols und Y an Rows. Wenn wir hier einen Verkabelungsfehler machen, ist das halb so schlimm, wir können in der Software immer noch mit den Verbindungen jonglieren. Brauchen wir allerdings mehr als ein Exemplar, sollten wir vielleicht doch Sorgfalt walten lassen, damit die Software auf allen Exemplaren gleich läuft.



(blau = ESP8266, grün = VC20/C64)



Strommäßig können wir uns ebenfalls am Tastatur Konnektor bedienen: Pin 1 führt Ground und Pin 4 führt +5 Volt. GND geht an VSS und VEE. Für VCC müssten wir eigentlich +5 Volt nehmen, da der MT8808 ab 4.5 V Versorgungsspannung spezifiziert ist. Aber bei mir funktioniert er auch sehr gut mit der 3.3V-Leitung des ESP8266. Dann entstehen niedrigere Ausgangsspannungen (um die 5V statt 10V) und es geht nicht so schnell etwas am ESP8266 kaputt, sollte mal was falsch verdrahtet sein.

Bleibt nur noch ein freier Pin (von nicht bestückten Pin 2 einmal abgesehen): Pin 3 / Restore. Die RESTORE-Taste des VC20 / C64 hat eine eigene Leitung und ist zusammen mit der Run / Stop -Taste für einen "kleinen Soft-Reset" der Computer zuständig. Run/Stop-Restore unterbricht Programme, löscht den Bildschirm und setzt die Bildschirmfarben zurück.

Die Restore-Leitung führt beim C64 in einen eigenen kleinen Schaltkreis, der schließlich einen NMI, einen Non-Maskable-Interrupt (nicht zurückverfolgbarer Interrupt) direkt an der 6510 CPU auslöst, um jederzeit sicher erkennen zu können, dass jetzt eine Unterbrechung ausgeführt werden soll.

Beim **VC20** führt die Restore-Leitung hingegen in den **6522** VIA an den dortigen Pin CA1, was ein selektiven, also nachverfolgbaren Interrupt ermöglicht.

Weitere Erklärungen zu Interrupts in meinen Artikeln [8-Bit-Breadboard-Computer auf Basis einer 6502-CPU - Erweiterung um ein Debug-Tool \(NMI\)](#) und [8-Bit-Breadboard-Computer auf Basis einer 6502-CPU - selektives Interrupt Handling über den VIA 6522 \(IRQ\)](#).

Die Restore Leitung liegt normalerweise auf High (das heißt 5V) und muss auf Low gezogen werden, während Run/Stop gehalten wird, um diesen Mini-Reset durchzuführen.

Die 5 Volt wollen wir nicht direkt an einem unserer Pins, denn der **ESP8266** mag nur Eingangsspannungen bis 3.3V. Also müssten wir eine kleine Transistorschaltung, die die Restore-Leitung auf Ground zieht, wenn der Transistor schaltet (High an der Basis bekommt) verbauen.

Es wäre zwar noch der Pin A0 frei, [aber der kann nur als Input, nicht als Output](#) benutzt werden. Auf eine emulierte Restore-Taste müssen wir also leider verzichten. Wir können höchstens einen Taster zwischen GND und Pin 3 des Tastaturkonnektors setzen für ein manuelles Restore.

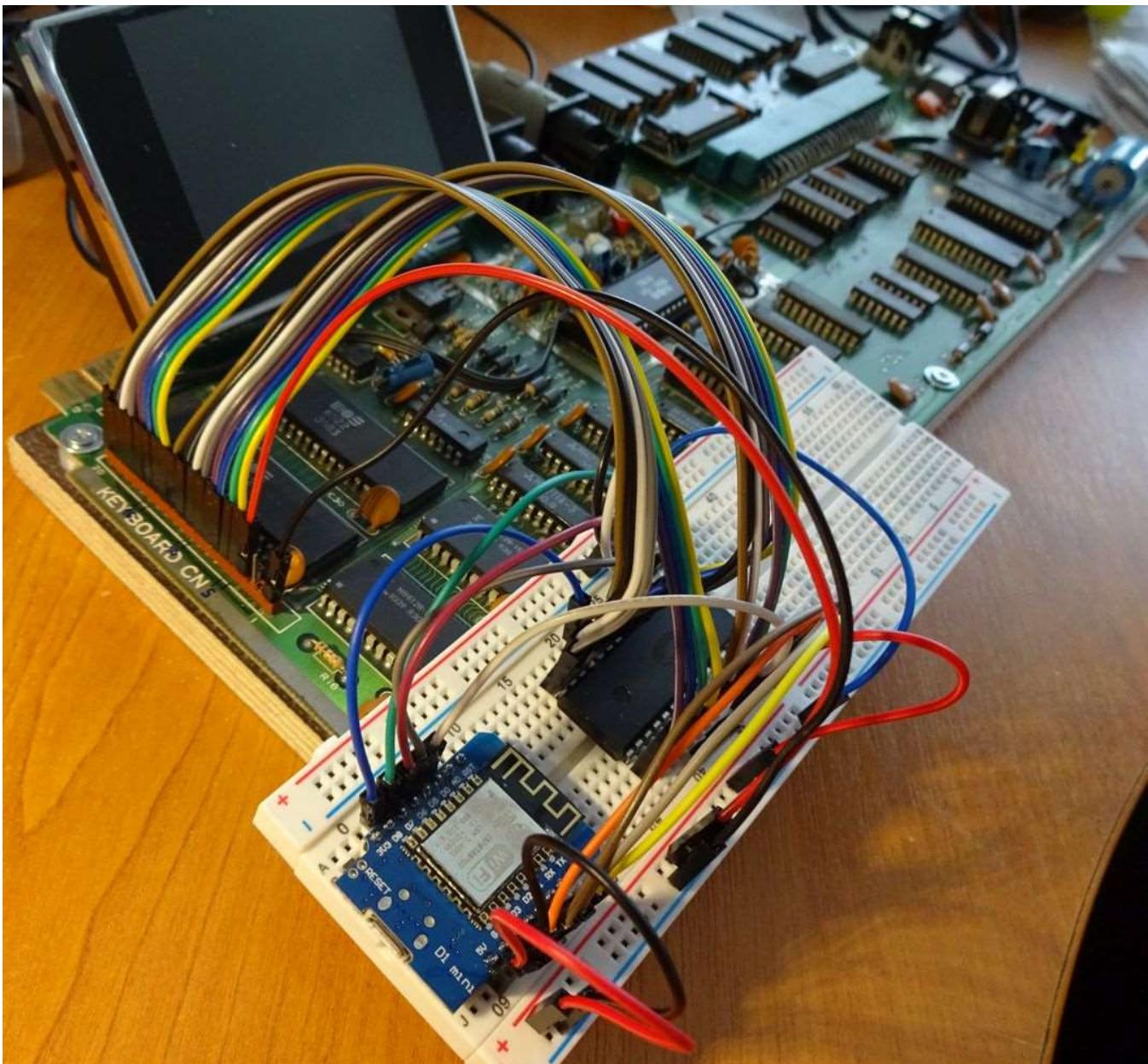
### Anschluss des Tastatur-Konnektors

Die Ausgänge des MT8808 schließen wir einfach der Reihe nach an den Tastatur-Konnektor den C64 bzw. VC20 an:

#### MT8808 20-Pin-Konnektor C64/VC20

X0	Pin 5	(Col 7)
X1	Pin 6	(Col 6)
X2	Pin 7	(Col 5)
X3	Pin 8	(Col 4)
X4	Pin 9	(Col 3)
X5	Pin 10	(Col 2)
X6	Pin 11	(Col 1)
X7	Pin 12	(Col 0)
Y0	Pin 13	(Row 7)
Y1	Pin 14	(Row 6)
Y2	Pin 15	(Row 5)
Y3	Pin 16	(Row 4)
Y4	Pin 17	(Row 3)
Y5	Pin 18	(Row 2)
Y6	Pin 19	(Row 1)
Y7	Pin 20	(Row 0)





### Tastaturmatrix VC20/C64

Natürlich müssen wir wissen, welche beiden Tastatur-Konnektor-Pins wir miteinander verbinden müssen, um eine bestimmte Taste auszugeben. Durch Durchprobieren bin ich zu folgender Tabelle gekommen:

Cols / Rows	Pin13 (Y0)	Pin14 (Y1)	Pin15 (Y2)	Pin16 (Y3)	Pin17 (Y4)	Pin18 (Y5)	Pin19 (Y6)	Pin20 (Y7)	Rows
Pin 5 (X0)	F7	Home	-	0	8	6	4	2	
Pin 6 (X1)	F5	Pfeil hoch	@	O	U	T	E	Q	
Pin 7 (X2)	F3	=	:	K	H	F	S	C= (Fkt.)	
Pin 8 (X3)	F1	Shift (right)	.	M	B	C	Z	Space	
Pin 9 (X4)	Cursor up/dw	/	,	N	V	X	Shift (left)	Run / Stop	
Pin 10 (X5)	Cursor lt/rt	,	L	J	G	D	A	Ctrl	
Pin 11 (X6)	Return	*	P	I	Y	R	W	Pfeil links	
Pin 12 (X7)	Del	Pfund-Zeichen	+	9	7	5	3	1	

2020 by Oliver Kuhlemann, www.cool-web.de

**Cols** zutreffend für: C64 (C)1983 Assy No. 250407 Rev. B, Made in Hong Kong;  
VC20 von 1983

Brauchen wir zum Beispiel ein A müssen wir also Pin 10 (X5) und Pin 19 (Y6) verbinden.

## Die Software

Durch entsprechende Ansteuerung des **MT8808** durch den **ESP8266** können wir jede Taste halten oder kurz drücken (Data erst auf High, dann stroben und dann Data auf Low und stroben).

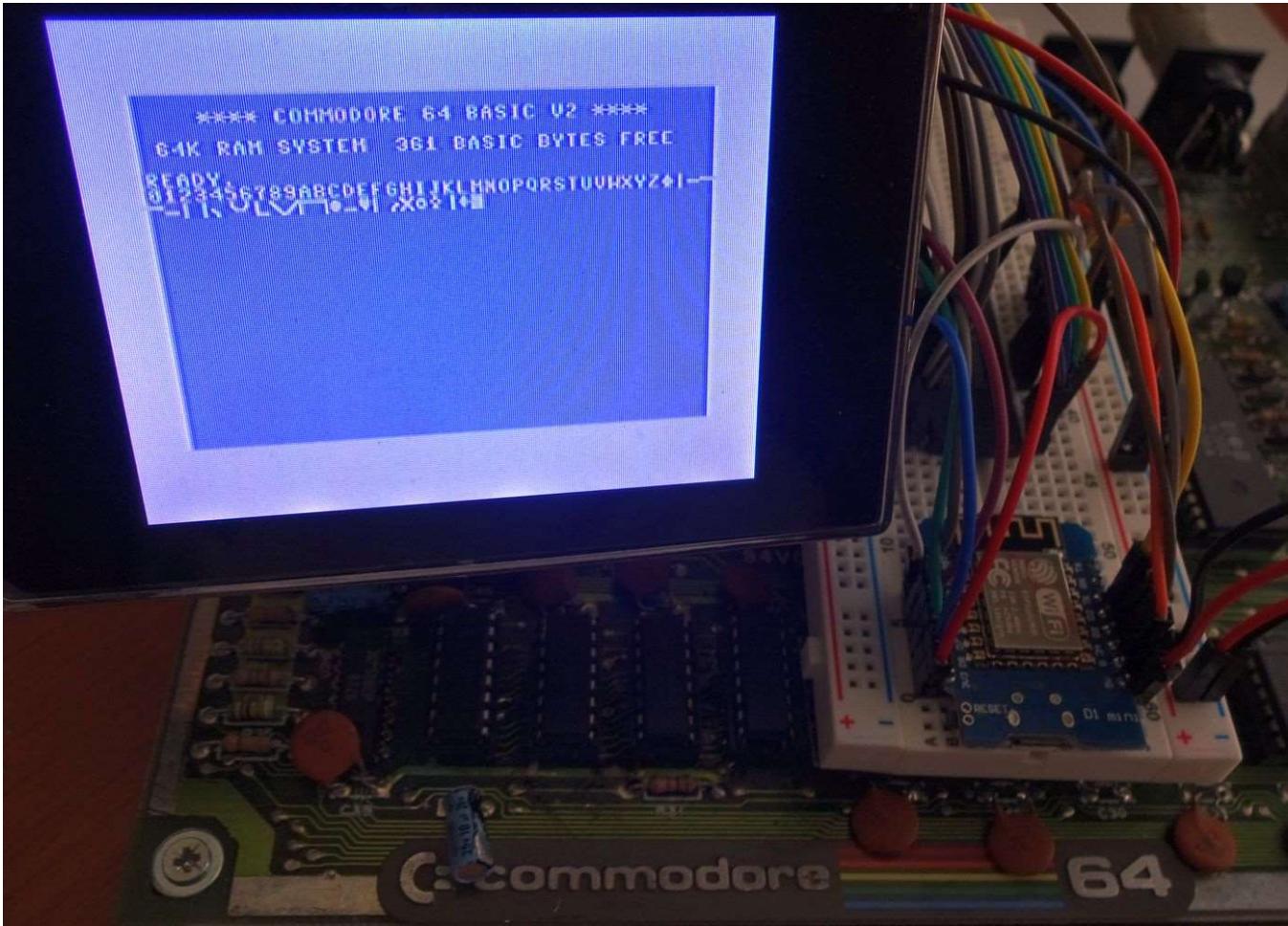
## HTTP Interface

Brauchen wir noch eine Möglichkeit, die WLAN-Tastatur übers WLAN anzusprechen. Am einfachsten ist das wohl zu bewerkstelligen, wenn wir Eingaben über ein HTTP-Interface entgegen nehmen. Dann können wir unser Tastatureingaben über einen Browser in der Adresszeile auf dem PC oder Smartphone oder jedem anderen Gerät mit einem Browser eingeben und der **ESP8266** wertet die URL dann aus und drückt die entsprechenden Tasten.

Da ich mich nicht extra von meinem Haus-WLAN abmelden und beim **ESP8266** anmelden will, lasse ich die WLAN-Tastatur ebenfalls ins Haus-WLAN konnektieren und vergabe ihr per DHCP meies Routers die hübsche IP-Adresse 192.168.0.64 passend zum **C64** bzw. 192.168.0.20 passend zum **VC20**.

Sobald sich die WLAN-Tastatur nach ein paar Sekunden ins WLAN eingeloggt hat, kann sie auch schon Kommandos über den Port 80 / HTTP entgegennehmen.

Ein erster Test sieht auch schon sehr vielversprechend aus:



Die entsprechende URL dazu war

<http://192.168.0.64/0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ>

Wie man sieht, nutze ich die auf dem PC übliche Groß/Kleinschreibung aus. Werden Großbuchstaben übergeben, dann wird neben dem Buchstaben auch die Shift-Taste gedrückt.

Zwei Probleme entstehen allerdings beim HTTP-Protokoll:

### 1. Manche Zeichen werden durch den Browser enkodiert.

Das dient dazu, auch Sonderzeichen über HTTP übertragen zu können. So wird aus dem Leerzeichen zum Beispiel ein "%20", was auch so beim Tastatur-Emulator ankommt. "%20" bedeutet nichts anderes als: "hier kommt Ein Zeichen mit dem HexCode 0x20". Und Hex \$20 sind dezimal 32 und das ist das ASCII-Zeichen für die Leertaste. Das Verhalten gilt auch für andere Zeichen wie Anführungsstriche, Prozentzeichen etc.

Die Lösung ist recht simpel: Ich ersetze ein %20 wieder zu einem Leerzeichen.

### 2. Manche Zeichen kann man so nicht in einer URL eingeben

Schwieriger wird es, wenn man Zeichen übermitteln will, die man nicht so einfach in einer URL eingeben kann, oder die es gar nicht auf einer PC-Tastatur gibt. Wie will man z. B. ein *Return* eingeben, wenn dies dazu führt, dass die URL abgeschickt wird?

Hier habe ich folgende Lösung gefunden: ich ersetze Sonderzeichen durch "%xx" Codes. Bzw. habe ich *Return* durch die Tilde (~) ersetzt, die es auf der PC-Tastatur und nicht auf dem VC20/C64 gibt. Für alle anderen Zeichen habe ich willkürlich ASCII-Zeichen oberhalb von 128 als Platzhalter ausgewählt:

```
//Sondertasten
strg.replace ("%80", (String) '\x80'); // RETURN
strg.replace ("%81", (String) '\x81'); // HOME
strg.replace ("%82", (String) '\x82'); // CLR
strg.replace ("%83", (String) '\x83'); // DEL
strg.replace ("%84", (String) '\x84'); // INST
strg.replace ("%85", (String) '\x85'); // Pi-Zeichen
strg.replace ("%86", (String) '\x86'); // Groß/Klein-Umschaltung
strg.replace ("%87", (String) '\x87'); // RUN
strg.replace ("%88", (String) '\x88'); // STOP

strg.replace ("%90", (String) '\x90'); // CURSOR LEFT
strg.replace ("%91", (String) '\x91'); // CURSOR RIGHT
strg.replace ("%92", (String) '\x92'); // CURSOR UP
strg.replace ("%93", (String) '\x93'); // CURSOR DOWN
strg.replace ("%99", (String) '\x99'); // Run/Stop-Restore
```

```

strg.replace ("%A0", (String) '\xA0'); // C= drücken
strg.replace ("%A1", (String) '\xA1'); // C= loslassen
strg.replace ("%A2", (String) '\xA2'); // SHIFT-L drücken
strg.replace ("%A3", (String) '\xA3'); // SHIFT-L loslassen
strg.replace ("%A4", (String) '\xA4'); // SHIFT-R drücken
strg.replace ("%A5", (String) '\xA5'); // SHIFT-R loslassen
strg.replace ("%A6", (String) '\xA6'); // CTRL drücken
strg.replace ("%A7", (String) '\xA7'); // CTRL loslassen
strg.replace ("%A8", (String) '\xA8'); // Stop halten
strg.replace ("%A9", (String) '\xA9'); // Stop loslassen

// F-Tasten
strg.replace ("%F1", (String) '\xF1'); // F1
strg.replace ("%F2", (String) '\xF2'); // F2
strg.replace ("%F3", (String) '\xF3'); // F3
strg.replace ("%F4", (String) '\xF4'); // F4
strg.replace ("%F5", (String) '\xF5'); // F5
strg.replace ("%F6", (String) '\xF6'); // F6
strg.replace ("%F7", (String) '\xF7'); // F7
strg.replace ("%F8", (String) '\xF8'); // F8

```

Wenn ich also z. B. die F1-Taste drücken will, dann würde ich <http://192.168.0.64/%F1> abschicken. "%F1" würde dann in ASCII hex 0xF1 (dez 241) umgesetzt und als dieses Zeichen gespeichert.

Im Programm ist eine Umsetzungstabelle gespeichert, die pro Zeichen ein Byte definiert. Dies hat folgenden Aufbau:

```

76543210 Bit
^----- Commodore-Taste zusätzlich drücken
^----- Y2
^----- Y1
^----- Y0
^----- Shift-Taste zusätzlich drücken
^----- X2
^----- X1
^----- X0

```

Für ASCII 241 (unser F1) steht dort hex 0x30. die 3 steht für X=3 und die 0 steht für Y=0. Und das entspricht in der Tastaturmatrix der F1-Taste.

So kann ich jede beliebige Taste emulieren. Oder auch kleine Programme eingeben, etwa ein Progrämmchen für den VC20, dass mit Farben und Tönen spielt:

```

1 poke 36878,3~2 c=int(rnd(1)*255)+1~3 s=int(rnd(1)*50)+175~4 poke 36879,c~5 poke 36875,s~6 for t=1
to 100: next t~7 goto 2~

```

oder eines, um ein Labyrinth auf den C64-Bildschirm zu zeichnen:

```

10 print chr$(205.5+rnd(1)); : goto 10~

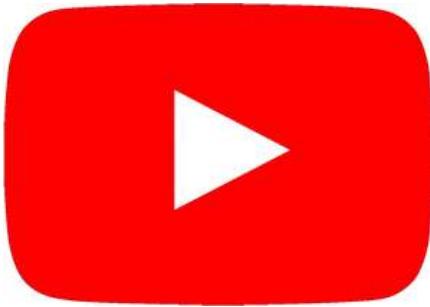
```

Dank der Tilde als Return-Zeichen können die Programme auch aus mehreren Programm-Zeilen bestehen.

Momentan lässt der Emulator zwischen jedem Tastendruck und Loslassen eine Pause von 20 Millisekunden. 10 ms stellten sich beim VC20 als zu schnell heraus. Das heißt, dass etwa 25 Zeichen pro Sekunde eingegeben werden können. Damit lassen sich auch längere Listings in überschaubarem Zeitrahmen eingeben.

### Test und Video

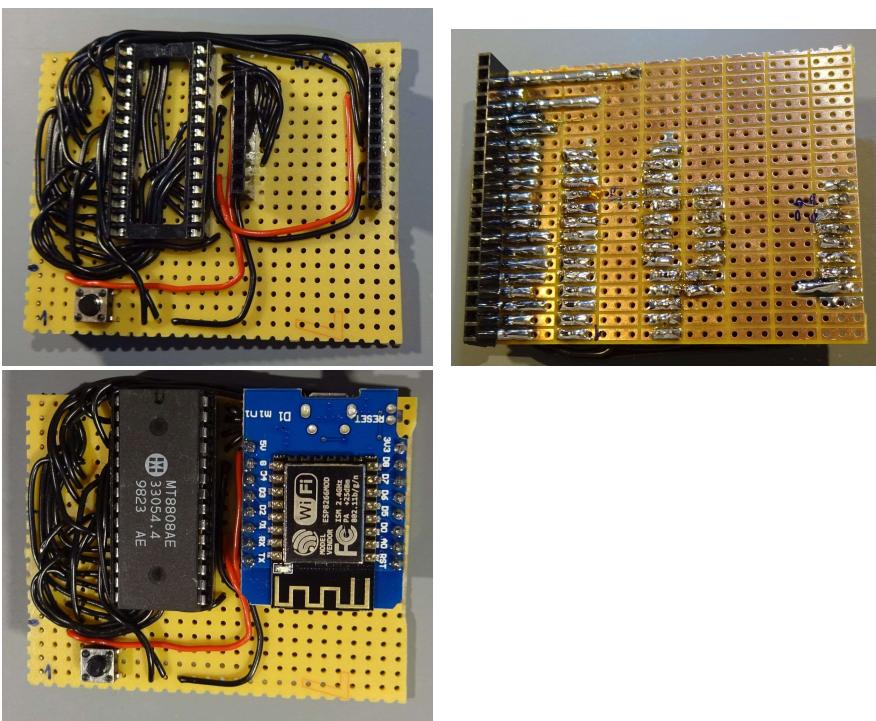
Natürlich habe ich die Schaltung und Software auch einem Test unterzogen, und zwar auf dem VC-20 als auch auf dem C64. Hier ein Video des Tests:



### Lochraster-Platinen-Version

Die fliegende Verkabelung auf dem **Breadboard** ist natürlich nicht das Wahre. Zum einen nimmt sie zuviel Platz weg und zum zweiten bleibt man bei sowas gerne mal an einem Kabel hängen, sieht es aus Versehen raus und sucht dann ewig, wo es denn wieder hingehört. Außerdem ist die Installation/Deinstallation ein Graus: alle Kabel müssen einzeln in der richtigen Reihenfolge eingesteckt bzw. abgezogen werden.

Also habe ich eine **Lochrasterplatine** zurechtgeschnitten und zwei Sockel für den **MT8808** und den **ESP8266** eingelötet. Des **ESP8266** habe ich gesockelt, damit ich ihn einfach herunternehmen und dann flashen kann. Auf der Platine selbst geht das ja nicht wegen der Verbindungen von 5V, GND, RX und TX.



Da ja leider keine Leitung mehr für die Restore-Taste übrig war, ist es dafür ein simpler Taster geworden, der Restore auf Ground zieht. So wichtig ist die Restore-Taste dann auch wieder nicht, dass man sie aus der Ferne bedienen können müsste. Da könnte man genausogut auch einen Hardware-Reset per WLAN-Tastatur auslösen; den braucht man fast öfters.

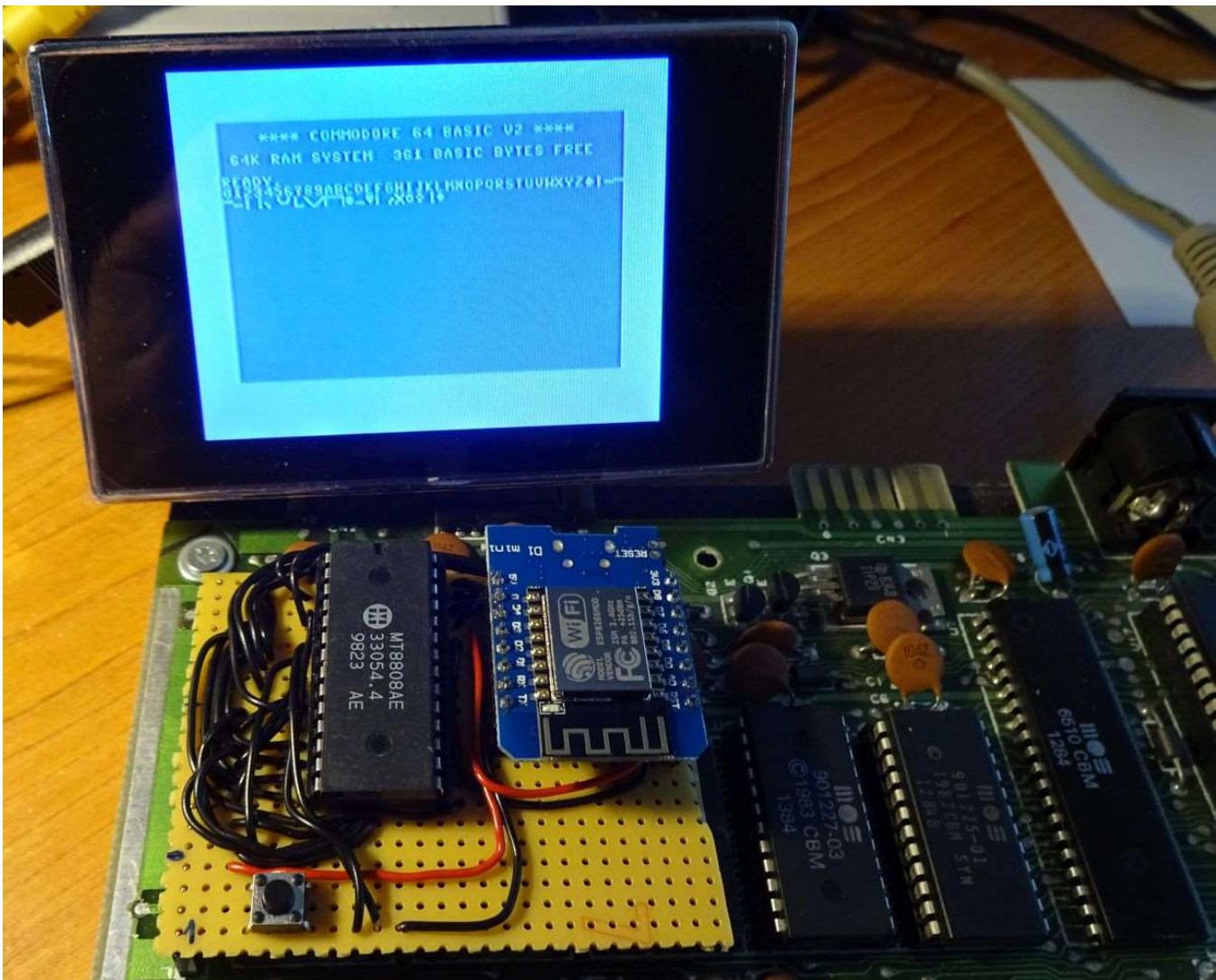
Da ich nur einseitige **Lochrasterplatine** da hatte, war das schwierigste, die Buchsenleiste für den Tastatur-Konnektor einzulöten. Das musste auf der Lötseite geschehen, also unter der Buchsenleiste selbst. Das war eine enge, fummelige Angelegenheit, aber schließlich ergab die Durchgangsprüfung, dass alle Leitungen verbunden waren.

Danach ging es ans Verdrahten: 19 Leitungen von der Tastatur, 28 Pins des **MT8808** und 12 Pins des **ESP8266** wollten richtig miteinander verbunden werden.

Ich habe mich für diesmal für isoliertes Kupferkabel (voll, keine Litze) entschieden. Das ist leichter abzusolieren und man kann sich das verdrillen der Litzen sparen. Auch ist das Durchstecken durch die Platine einfacher. Die Kabel habe ich auf der Oberseite verlegt. Wie man sieht, sind das einige geworden und man muss aufpassen, welchen Lötspalt man wählt, damit man sich nicht selbst den Weg verbaut.

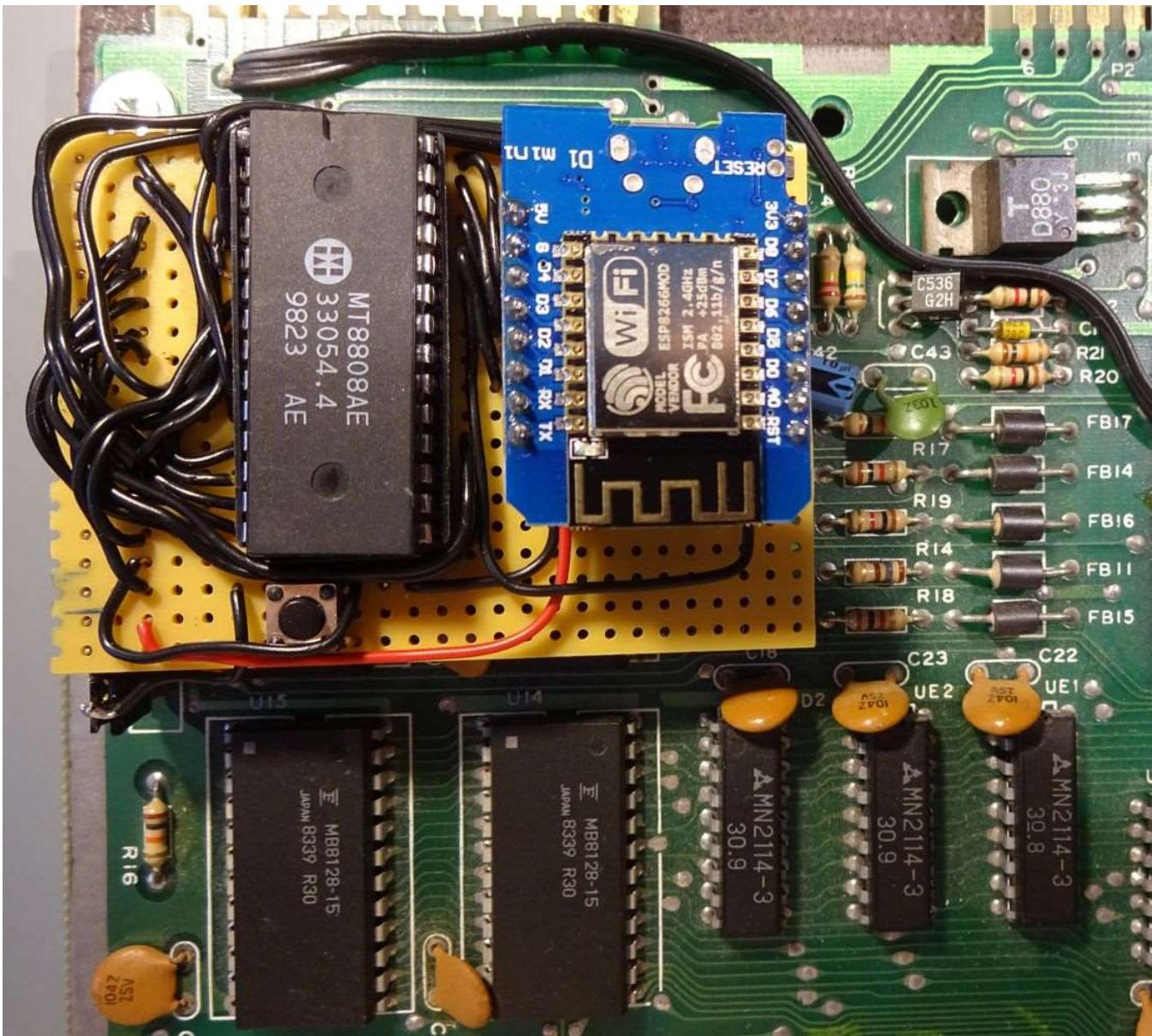
Nach einem abschließenden Durchmessen aller Leitungen und dem Trennen einer versehentlich entstandenen Lötverbindung sah dann alles gut auf und konnte am C64 getestet werden. Läuft. Perfekt.

Nur noch schnell ein paar Gummifüße auf die Unterseite geklebt, als Sicherheitsabstandshalter. Sicher ist sicher.



Nach der mehrstündigen Lötorgie hatte ich nicht gleich Lust, auch noch die Platine für den VC20 zusammenzulöten und habe das auf den nächsten Tag verschoben. Leider war das letzte Stück [Lochrasterplatine](#), das ich hatte, ein kleines Stückchen zu schmal, so dass ich ein wenig knausern musste.

Doch letztlich passierte auch die den Funktionstest, wurde mit [MT8808](#) und [ESP8266](#)-Modul bestückt und in den VC20 eingesetzt.



Ich bin sehr zufrieden mit den WLAN-Tastatur-Emulatoren mit [Lochrasterplatine](#). Die sind jetzt sehr viel kompakter.

Nun kann ich über die IP-Adresse 192.168.0.20 meinen VC20 fernbedienen und unter 192.168.0.64 steht mein C64-Tastatur-Emulator zur Verfügung.

Ich muss aber schon sagen, dass das viel Lötterei war. Gerade so gucken, wo welches Kabel hin muss, das viele Abisolieren, stecken, halten, löten, abzwacken, das zieht sich doch in die Länge mit der Anzahl der Leitungen.

Ich war schon am Überlegen, ob ich mir nicht Platinen entwerfe und ätzen lasse, aber dann wollte ich das doch schnell fertig haben und nicht lange warten, bis die Platinen da sind.

Und schließlich will ich die Tastatur auch möglichst bald benutzen, um [dem RAM-Fehler bei meinem C64 auf den Grund zu gehen](#).