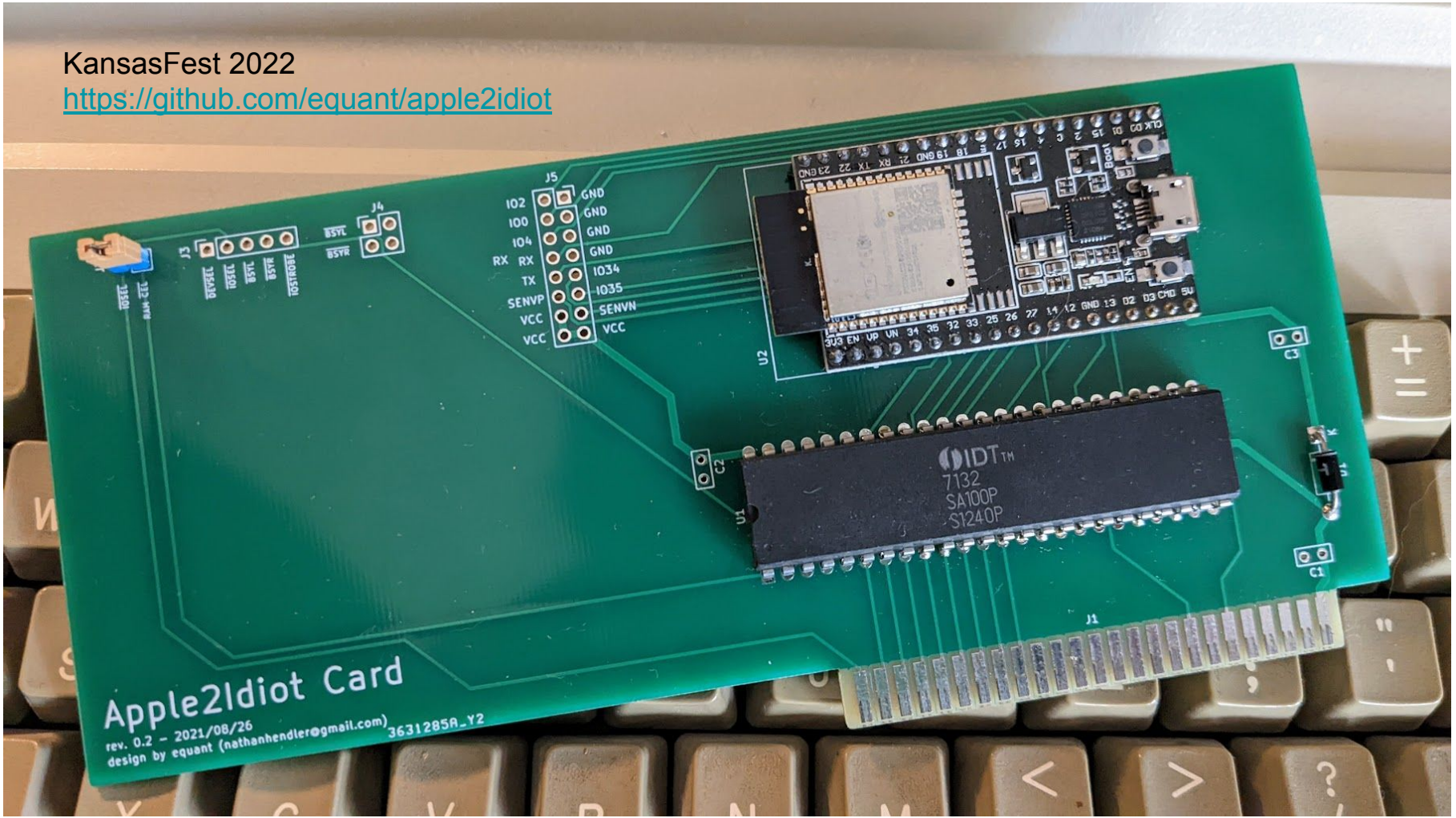
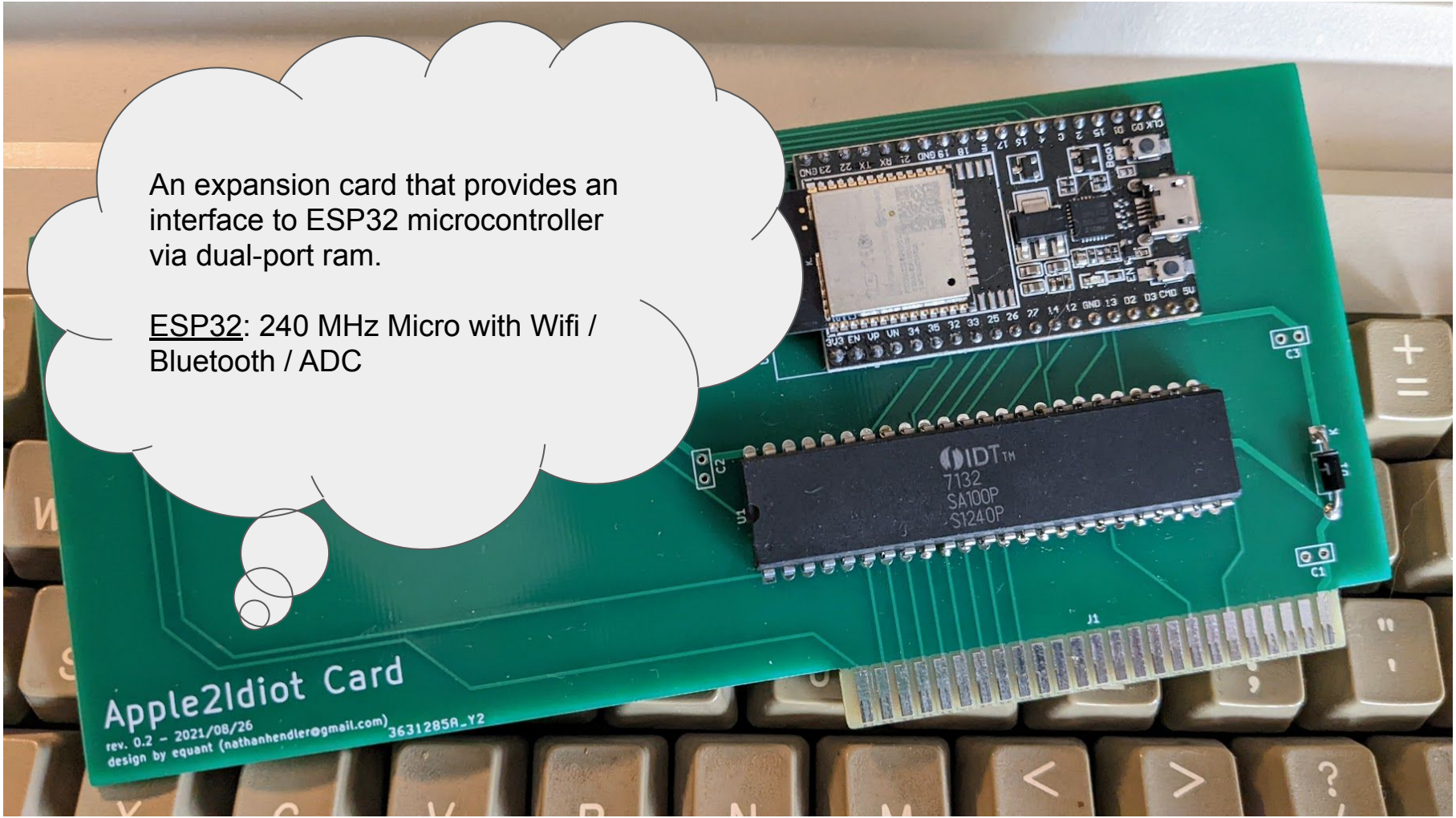
KansasFest 2022
https://github.com/equant/apple2idiot




An expansion card that provides an interface to ESP32 microcontroller via dual-port ram.
ESP32: 240 MHz Micro with Wifi / Bluetooth / ADC
IDT TM
7132
SA100P
S1240P
Apple2Idiot Card
rev. 0.2 - 2021/08/26
design by equant (nathanhendler@gmail.com)
3631285A_Y2

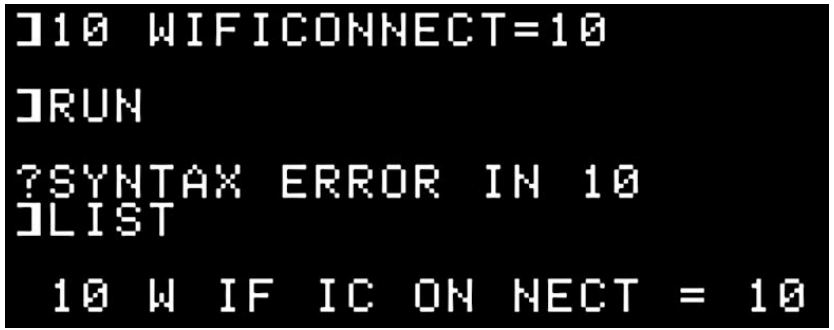
About me / Full disclosure

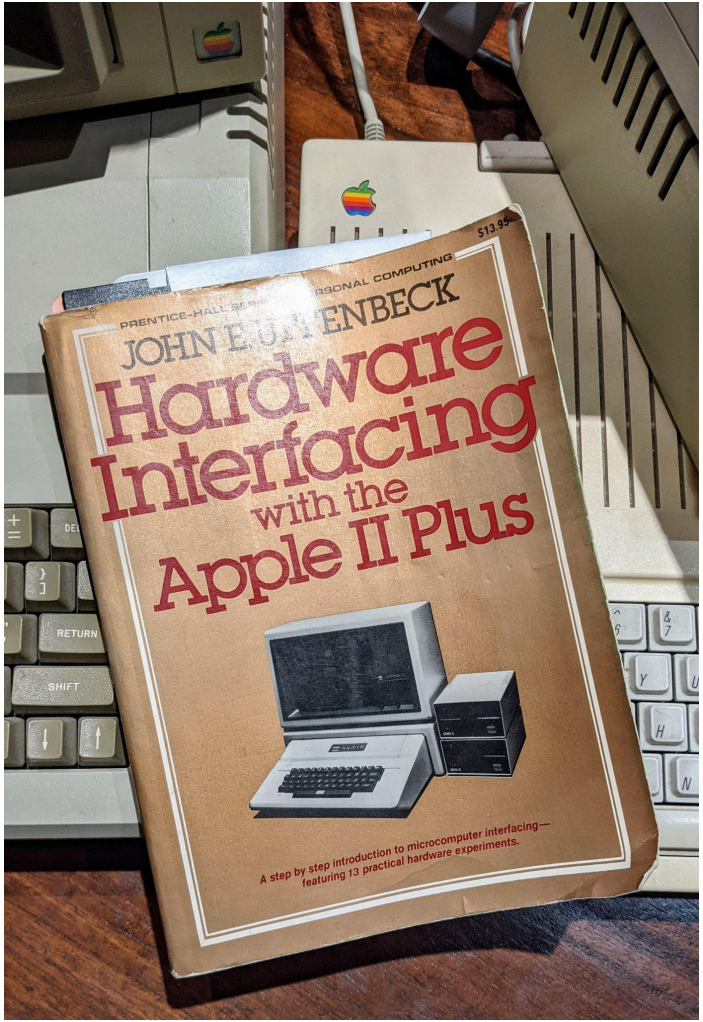
- Macintosh 68k System 6/7 background
  - Grackle 68k
  - mpd client
  - Robotfindskitten port for Mayan Apocalypse Edition
  - RetroBridgeBBS
- Started Apple II stuff with this project.
  - I mostly don’t know what I’m doing
  - I couldn’t have done any of this without the Apple II community.

About me / Full disclosure

- Macintosh 68k System 6/7 background
  - Grackle 68k
  - mpd client
  - Robotfindskitten port for Mayan Apocalypse Edition
  - RetroBridgeBBS
- Started Apple II stuff with this project.
  - I mostly don’t know what I’m doing
  - I couldn’t have done any of this without the Apple II community.

Proof I don’t know what I’m doing




PERSONAL COMPUTING
$13.95
PRENTICE-HALL
JOHN E. UFFENBECK
Hardware Interfacing with the Apple II Plus
A step by step introduction to microcomputer interfacing—
featuring 13 practical hardware experiments.

Hardware Interfacing with the Apple II Plus

Are you tired of game playing on your Apple computer?

Are you bored running other peoples' canned software?

Well, in this book, **John Uffenbeck** shows you how to accomplish such projects as programming your Apple to turn on a room light, your furnace, or your air conditioner; or to display the exponential charging curve of voltage on a capacitor; or the temperature versus time of day.

Hardware Interfacing with the Apple II Plus

Are you tired of game playing on your Apple computer?

Are you bored running other peoples' canned software?

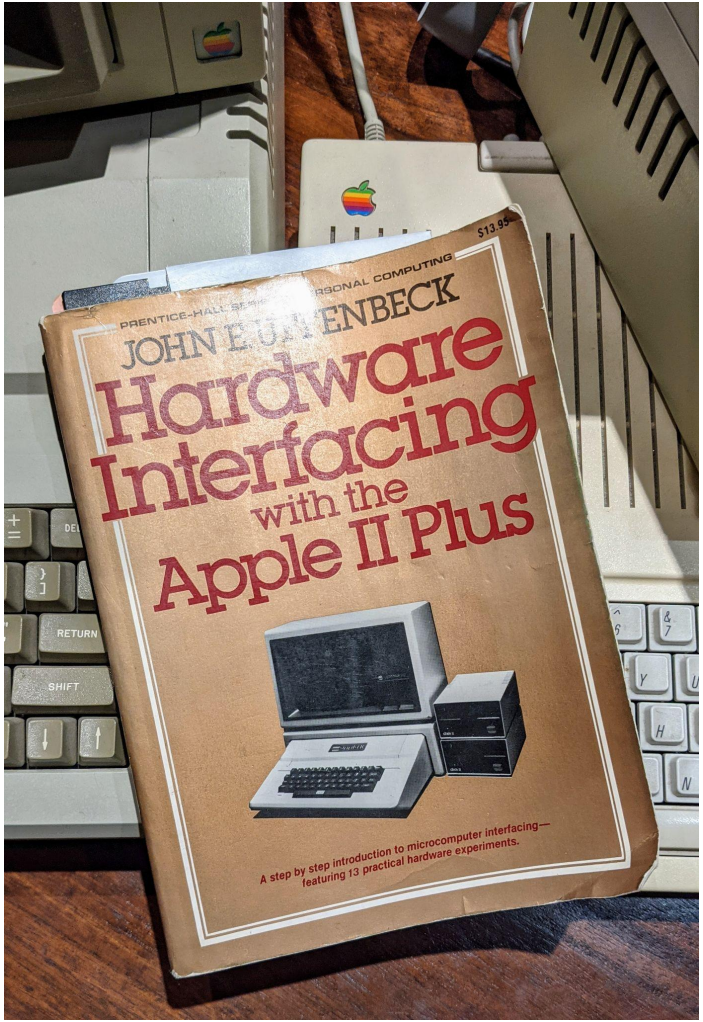
Well, in this book, **John Uffenbeck** shows you how to accomplish such projects as programming your Apple to turn on a room light, your furnace, or your air conditioner; or to display the exponential charging curve of voltage on a capacitor; or the temperature versus time of day.

Hardware Interfacing with the Apple II Plus

Are you tired of game playing on your Apple computer?

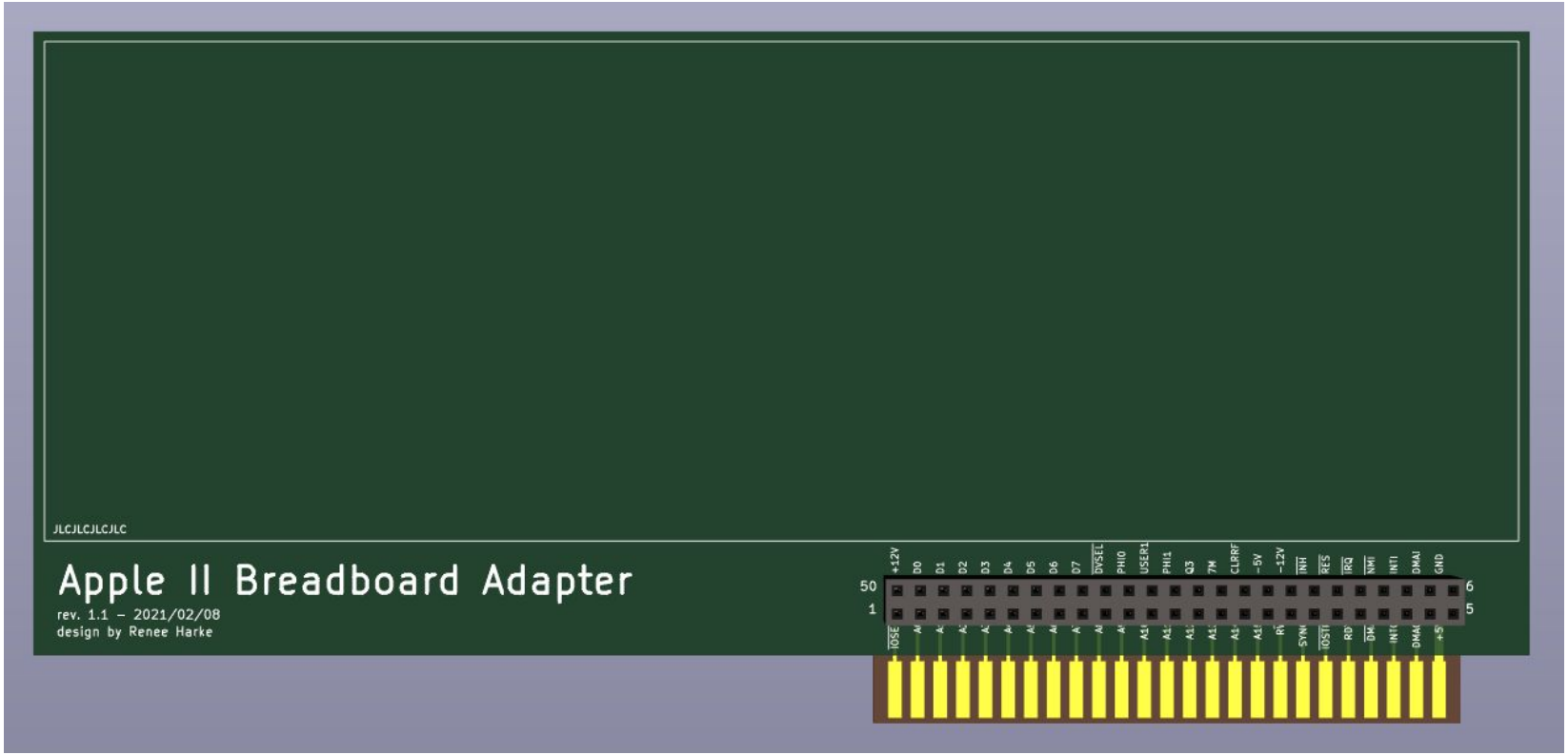
Are you bored running other peoples' canned software?

Well, in this book, **John Uffenbeck** shows you how to accomplish such projects as programming your Apple to turn on a room light, your furnace, or your air conditioner; or to display the exponential charging curve of voltage on a capacitor; or the temperature versus time of day.

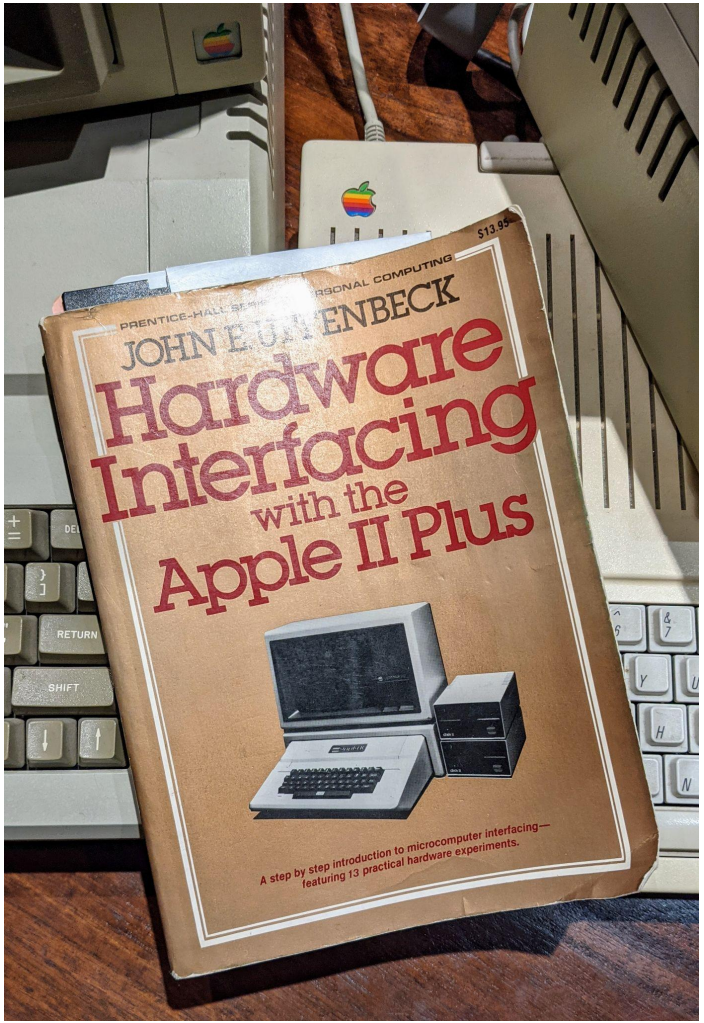


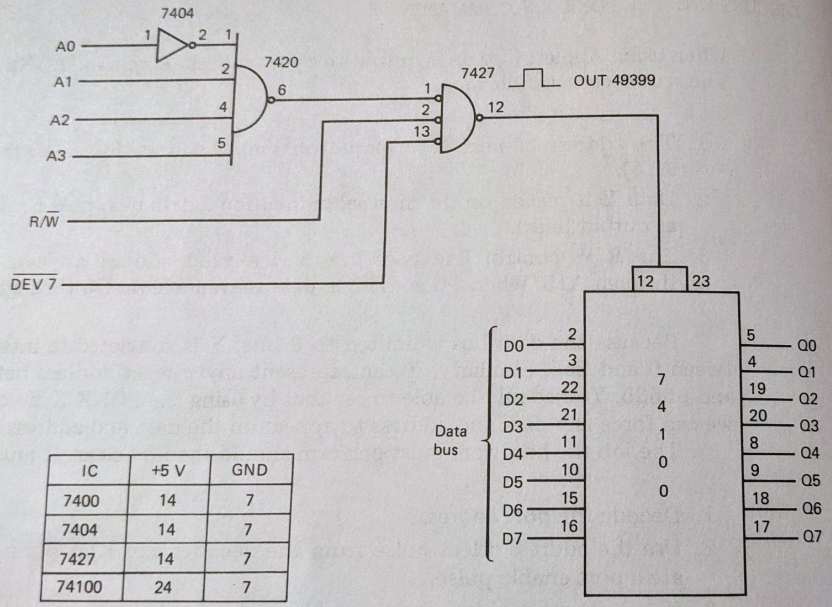
Project Priorities

- Develop hardware
- Custom PCB
- Easy
- Hand solderable
  - Through-hold components
  - Low component count
- Do something neat



https://github.com/rharke/apple-ii-breadboard-card

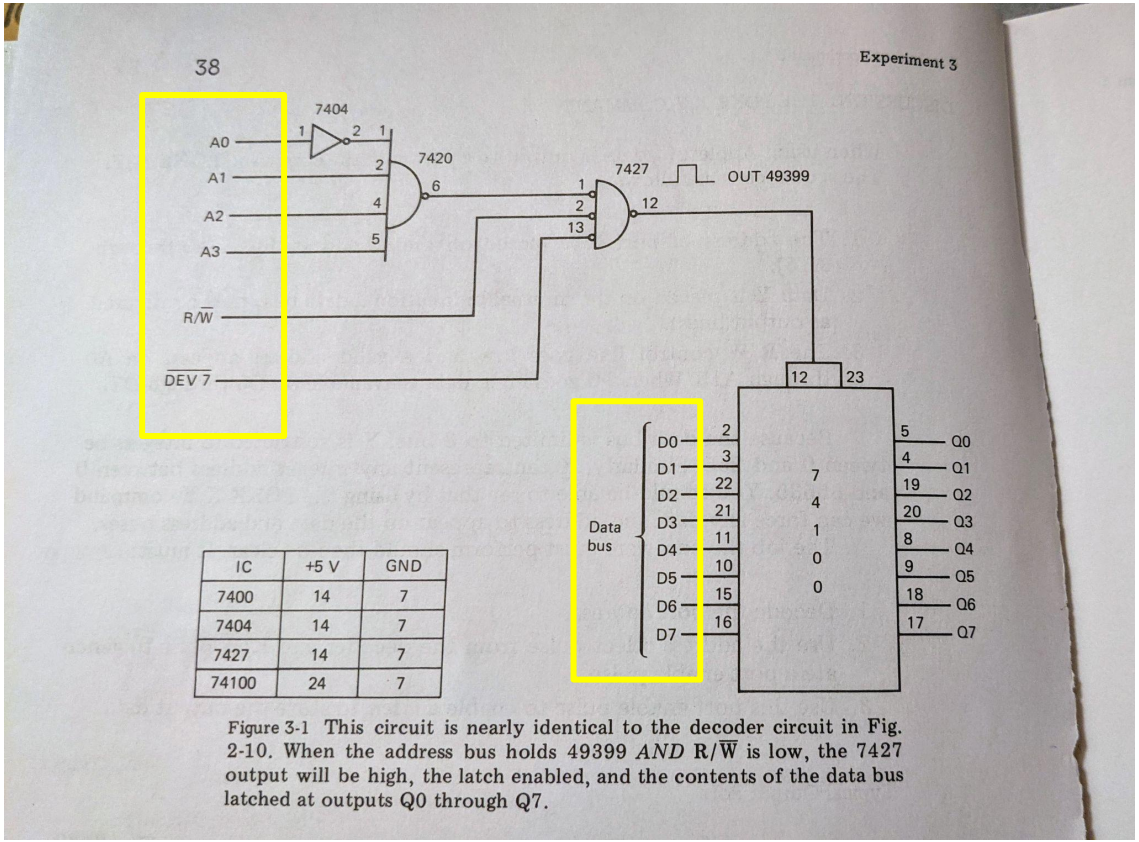




IC	+5 V	GND
7400	14	7
7404	14	7
7427	14	7
74100	24	7

Figure 3-1 This circuit is nearly identical to the decoder circuit in Fig. 2-10. When the address bus holds 49399 *AND* R/$\overline{W}$ is low, the 7427 output will be high, the latch enabled, and the contents of the data bus latched at outputs Q0 through Q7.

Stuff in yellow comes from Apple II (via card edge)



IC	+5 V	GND
7400	14	7
7404	14	7
7427	14	7
74100	24	7

Figure 3-1 This circuit is nearly identical to the decoder circuit in Fig. 2-10. When the address bus holds 49399 *AND* R/$\overline{W}$ is low, the 7427 output will be high, the latch enabled, and the contents of the data bus latched at outputs Q0 through Q7.

Stuff in yellow comes from Apple II (via card edge)

Only 4 LSB from Address Bus

All 8 Bits from Data

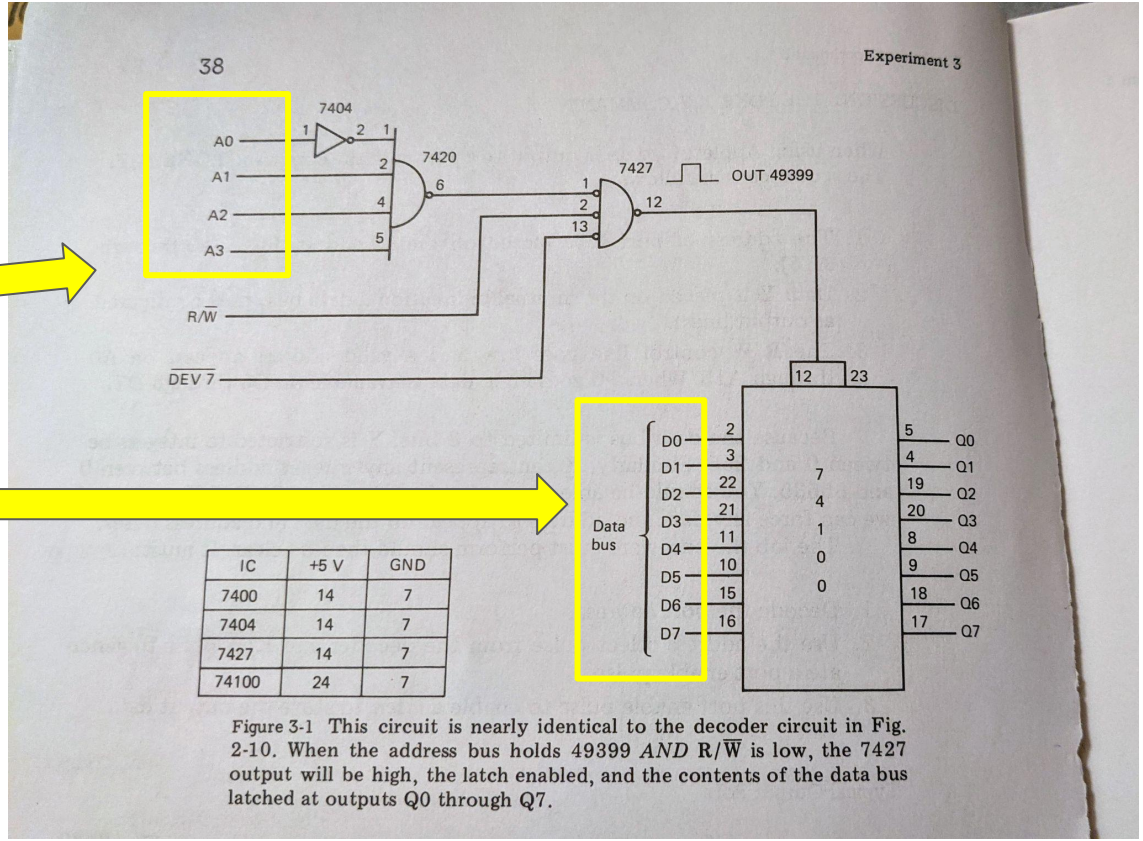
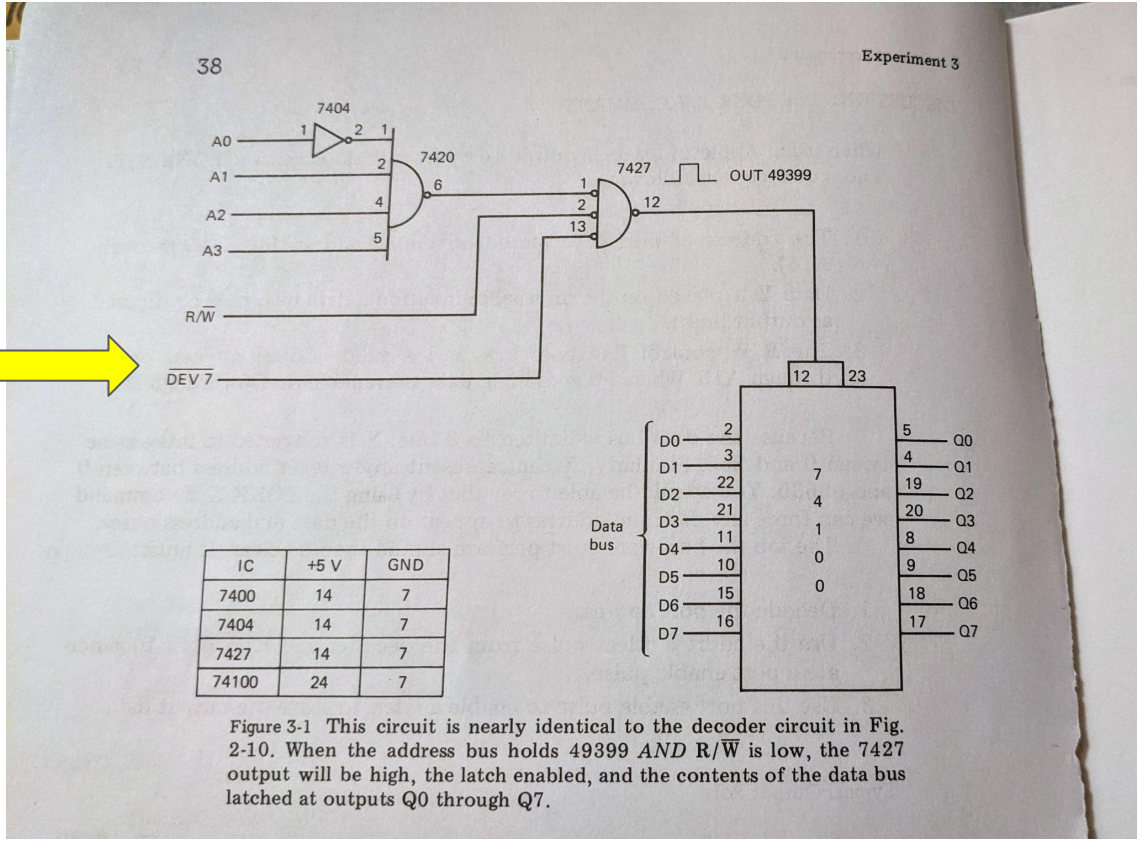


Figure 3-1 This circuit is nearly identical to the decoder circuit in Fig. 2-10. When the address bus holds 49399 *AND* R/$\overline{W}$ is low, the 7427 output will be high, the latch enabled, and the contents of the data bus latched at outputs Q0 through Q7.

Low when card is “selected”

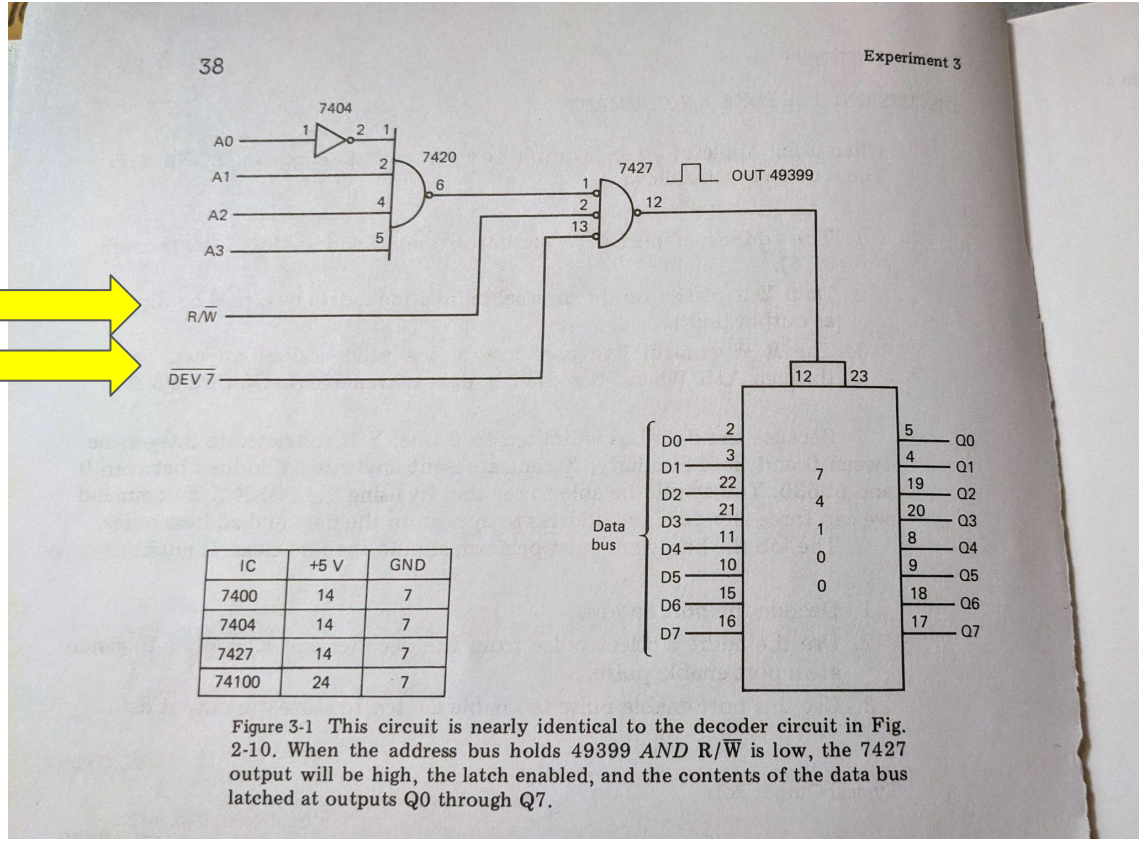


IC	+5 V	GND
7400	14	7
7404	14	7
7427	14	7
74100	24	7

Figure 3-1 This circuit is nearly identical to the decoder circuit in Fig. 2-10. When the address bus holds 49399 *AND* R/$\overline{W}$ is low, the 7427 output will be high, the latch enabled, and the contents of the data bus latched at outputs Q0 through Q7.

Low when Apple II is writing data

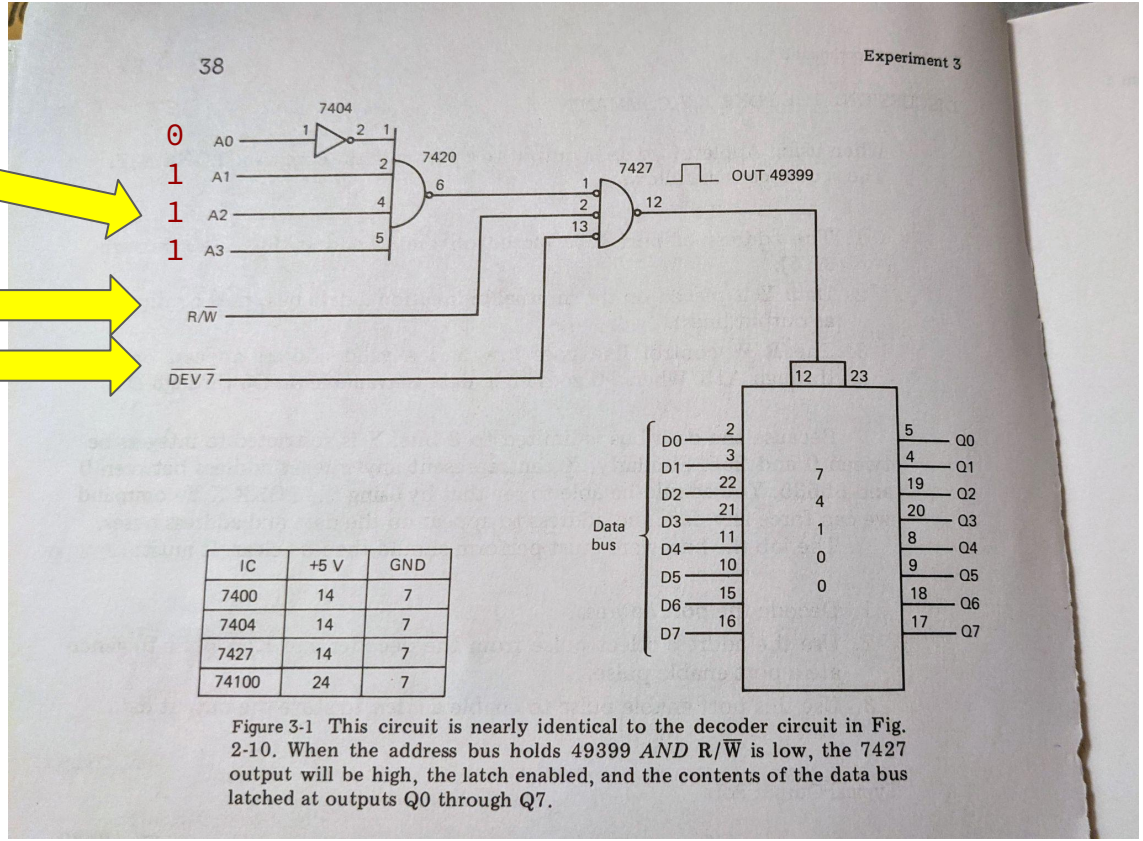
Low when card is “selected”



```
499399 == 0b1100000011110111
`POKE 499399, 42`
```

And Apple II is writing data

When low card is “selected”



CARD #1

DEVSEL

D0
D1
D2
D3
D4
D5
D6
D7

R/W

A0
A1
A2
A3

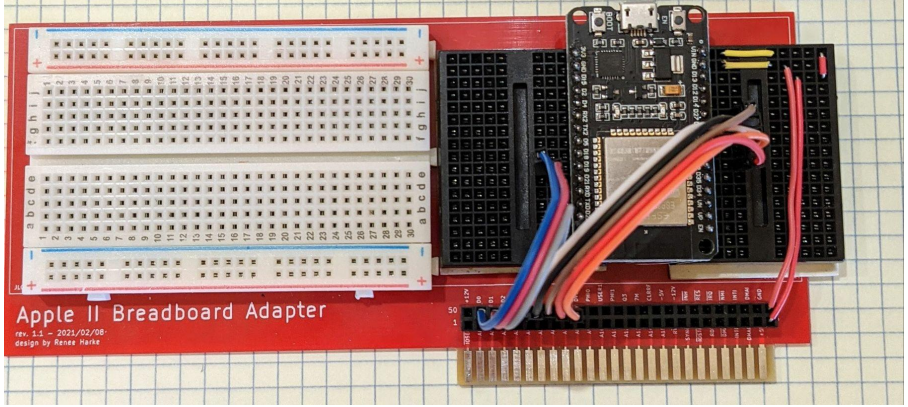
ESP32 PINS

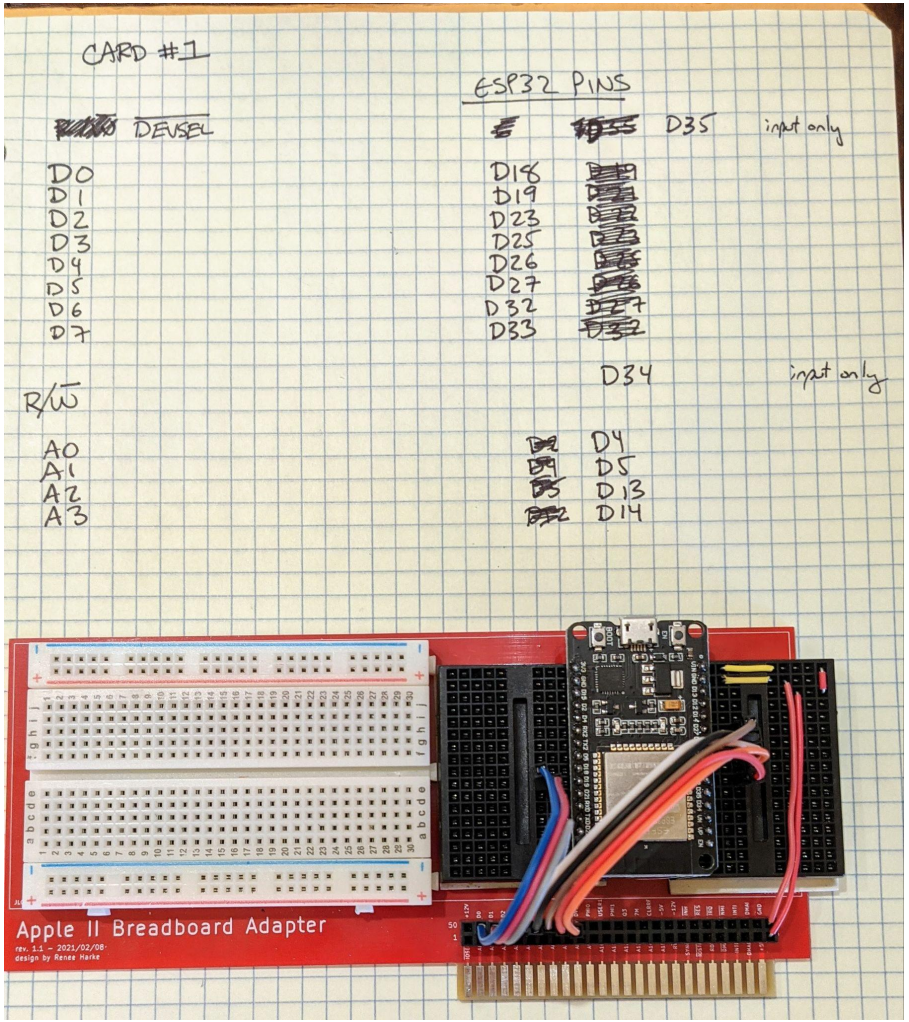
D35 input only

D18
D19
D23
D25
D26
D27
D32
D33

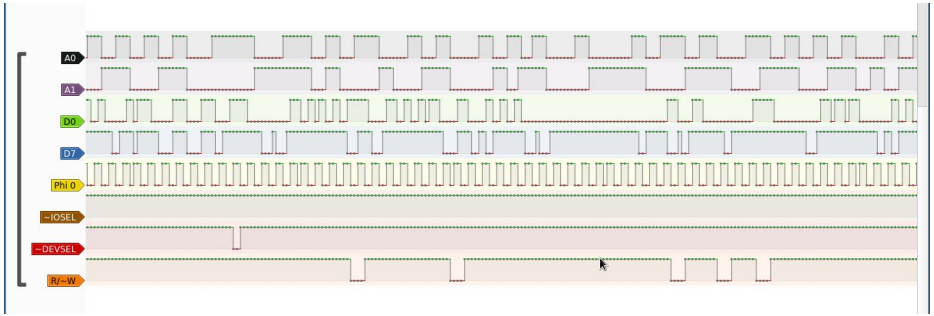
D34 input only

D4
D5
D13
D14



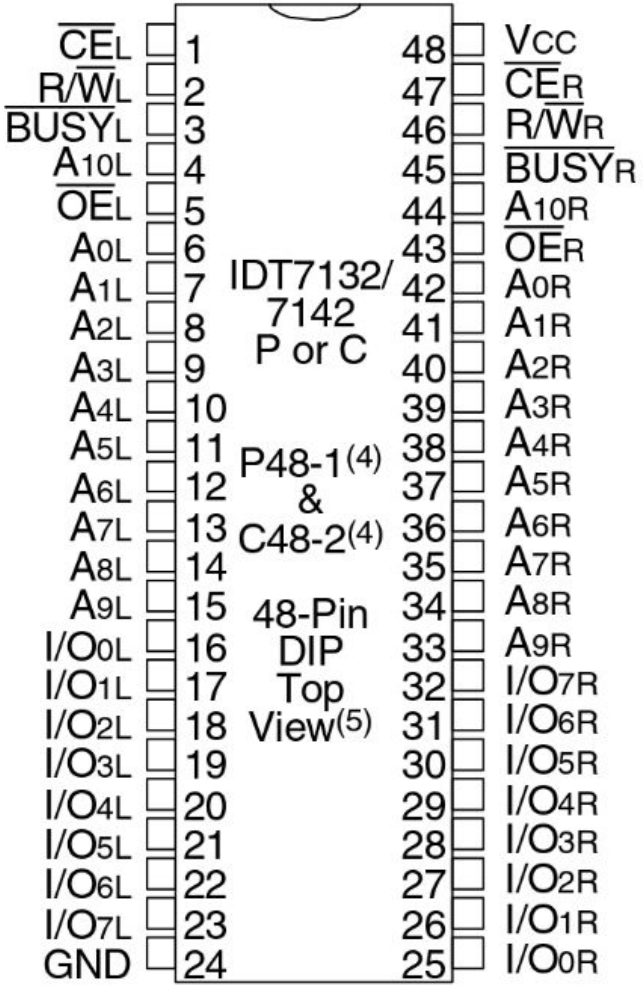


Didn’t work. ~2us interrupt latency



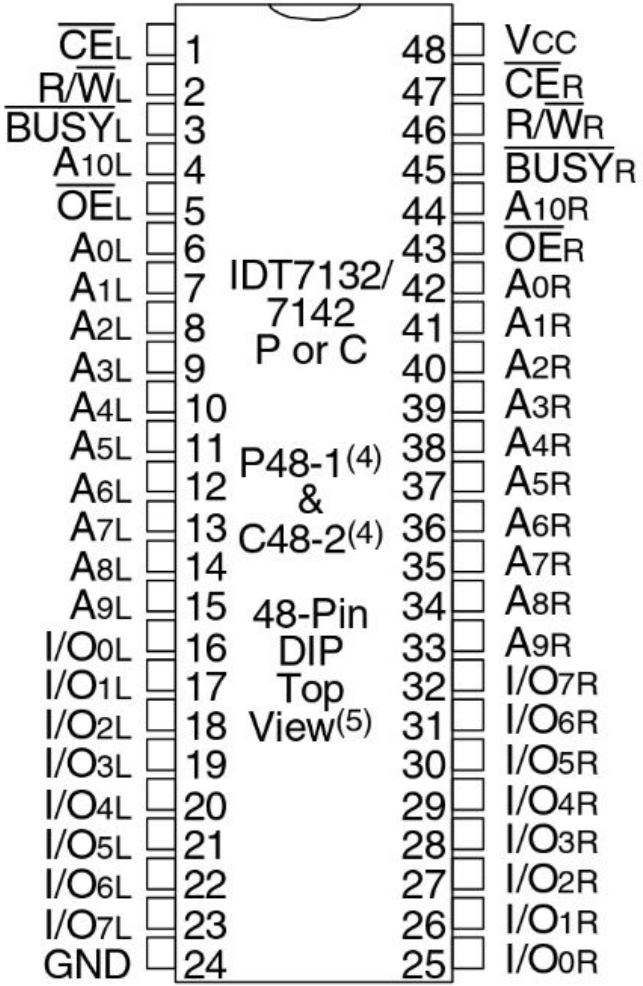
IDT7132

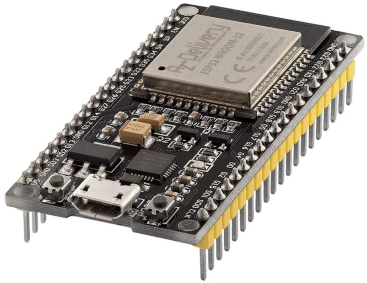
HIGH SPEED
2K x 8 DUAL PORT
STATIC RAM



IDT7132

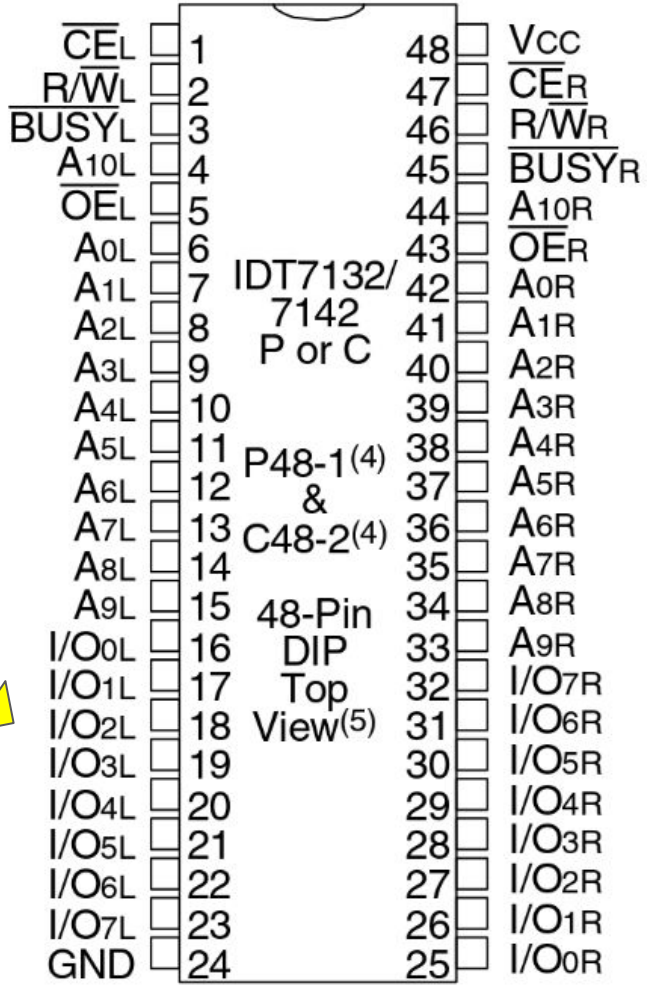
HIGH SPEED
2K x 8 DUAL PORT
STATIC RAM

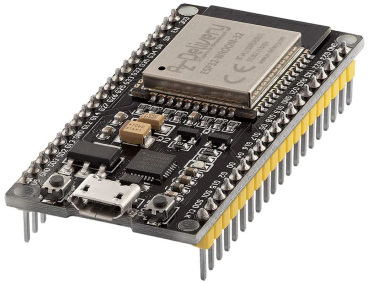




IDT7132

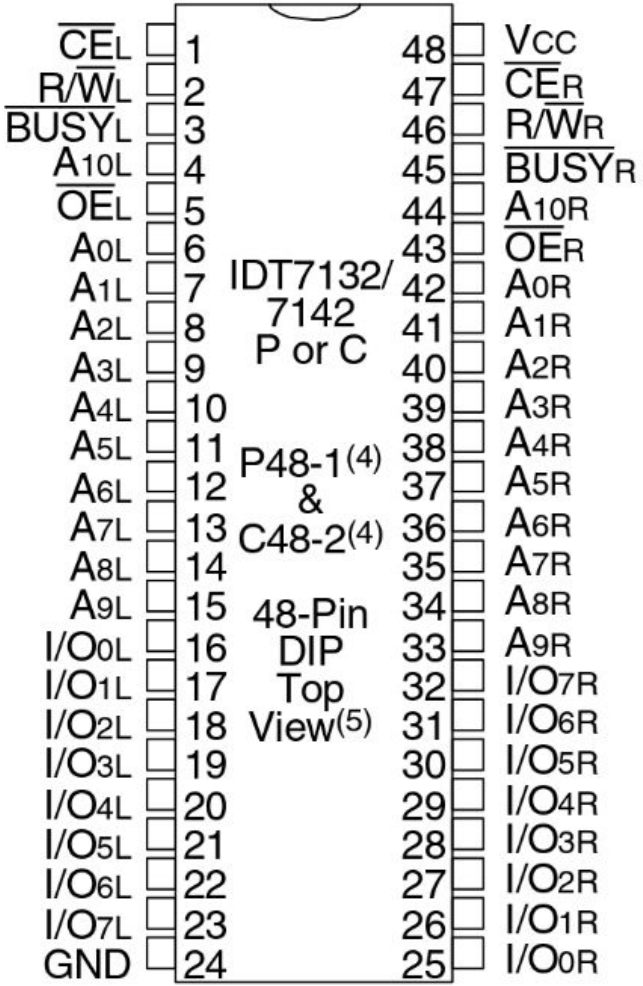
HIGH SPEED
2K x 8 DUAL PORT
STATIC RAM

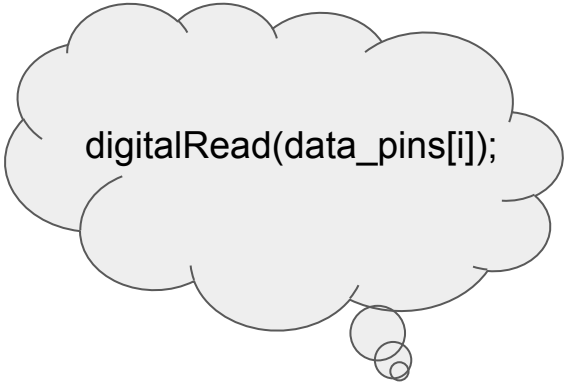


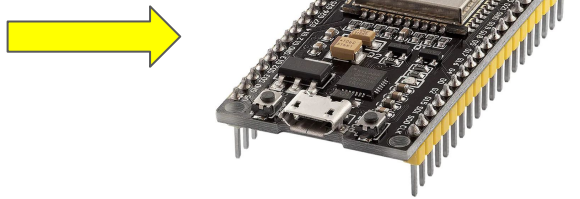


IDT7132

HIGH SPEED
2K x 8 DUAL PORT
STATIC RAM

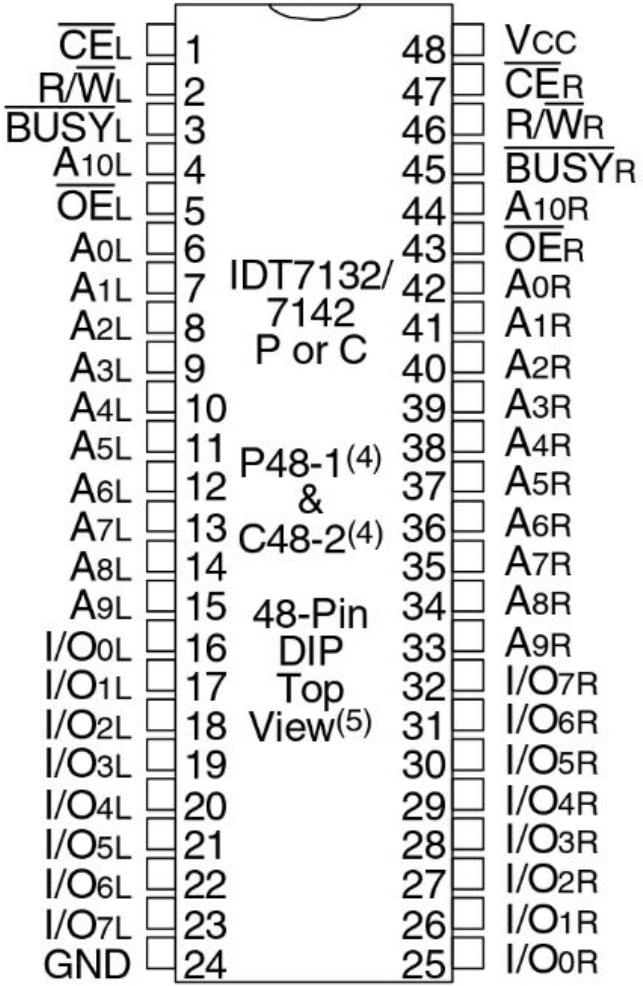




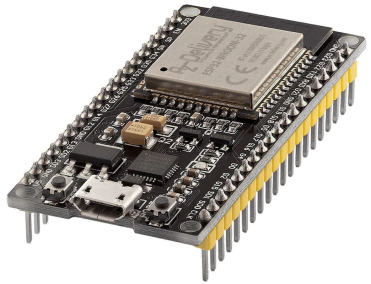


IDT7132

HIGH SPEED
2K x 8 DUAL PORT
STATIC RAM

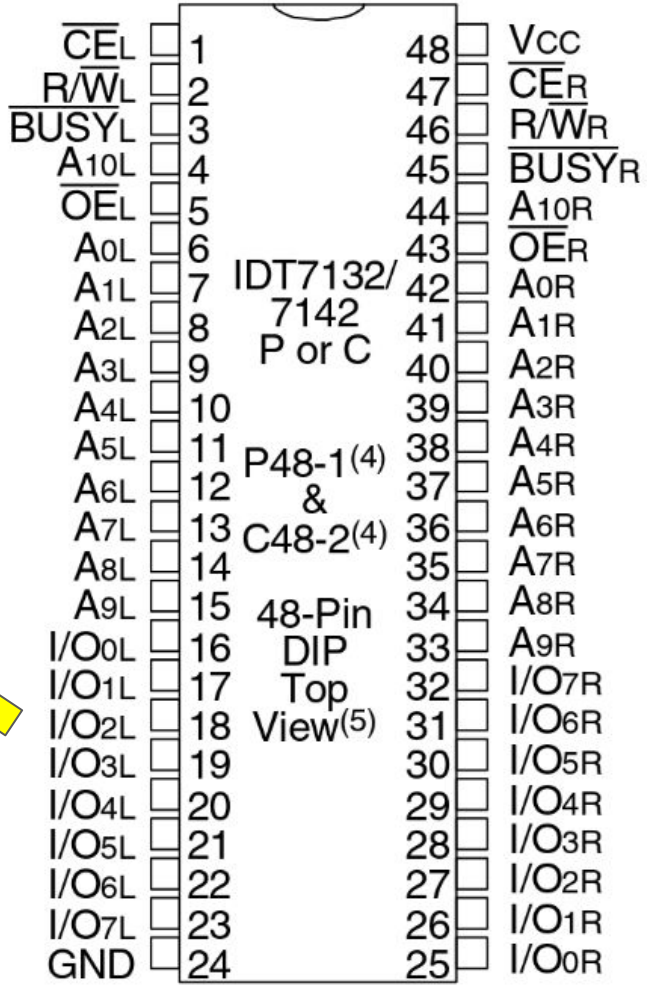


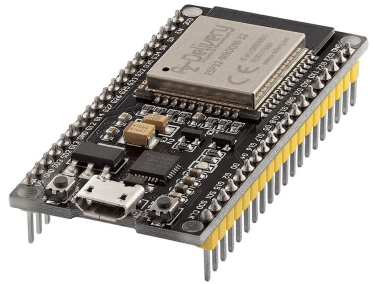
digitalWrite(data_pins[i]);

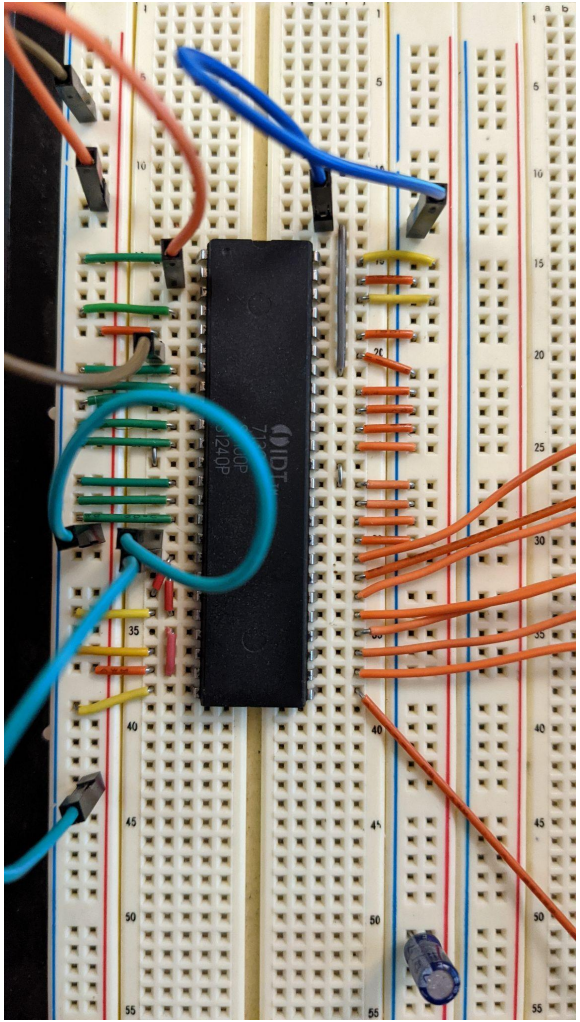


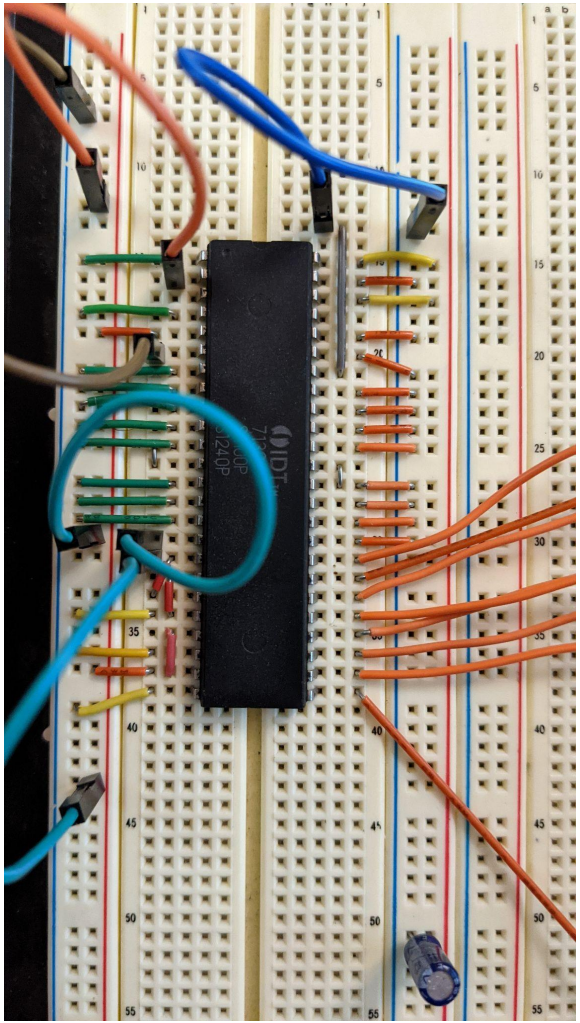
IDT7132

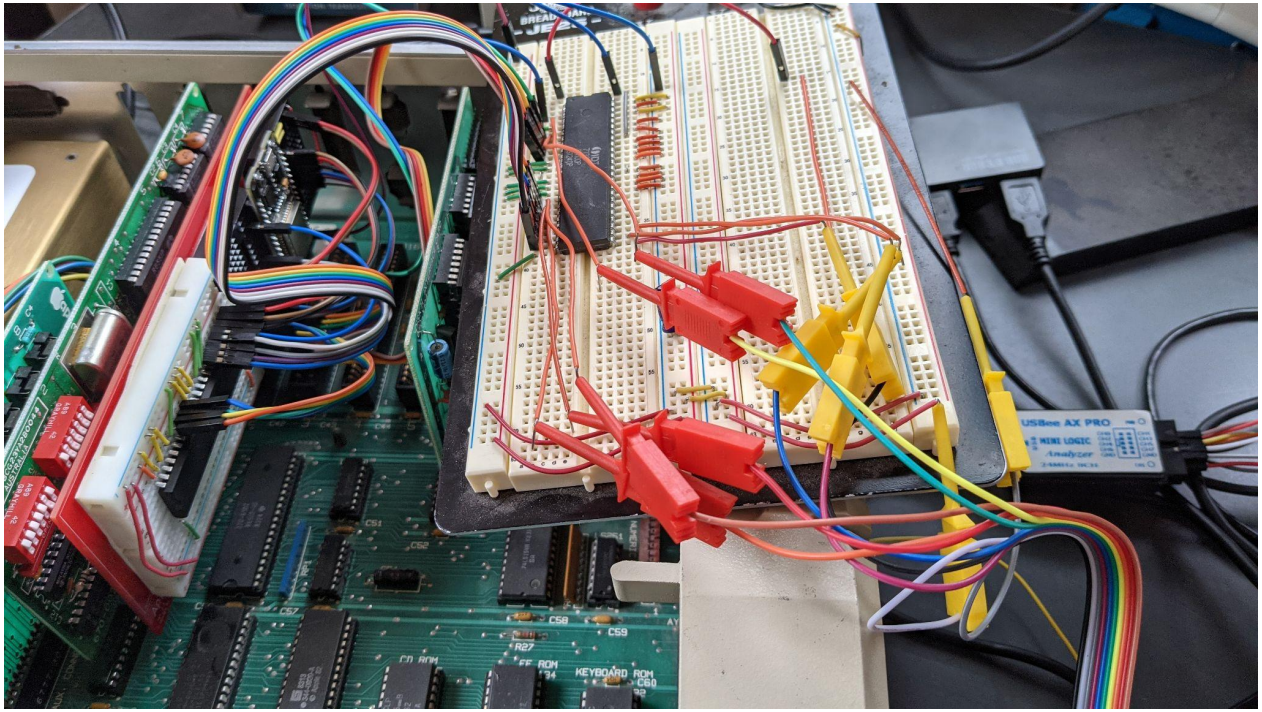
HIGH SPEED
2K x 8 DUAL PORT
STATIC RAM

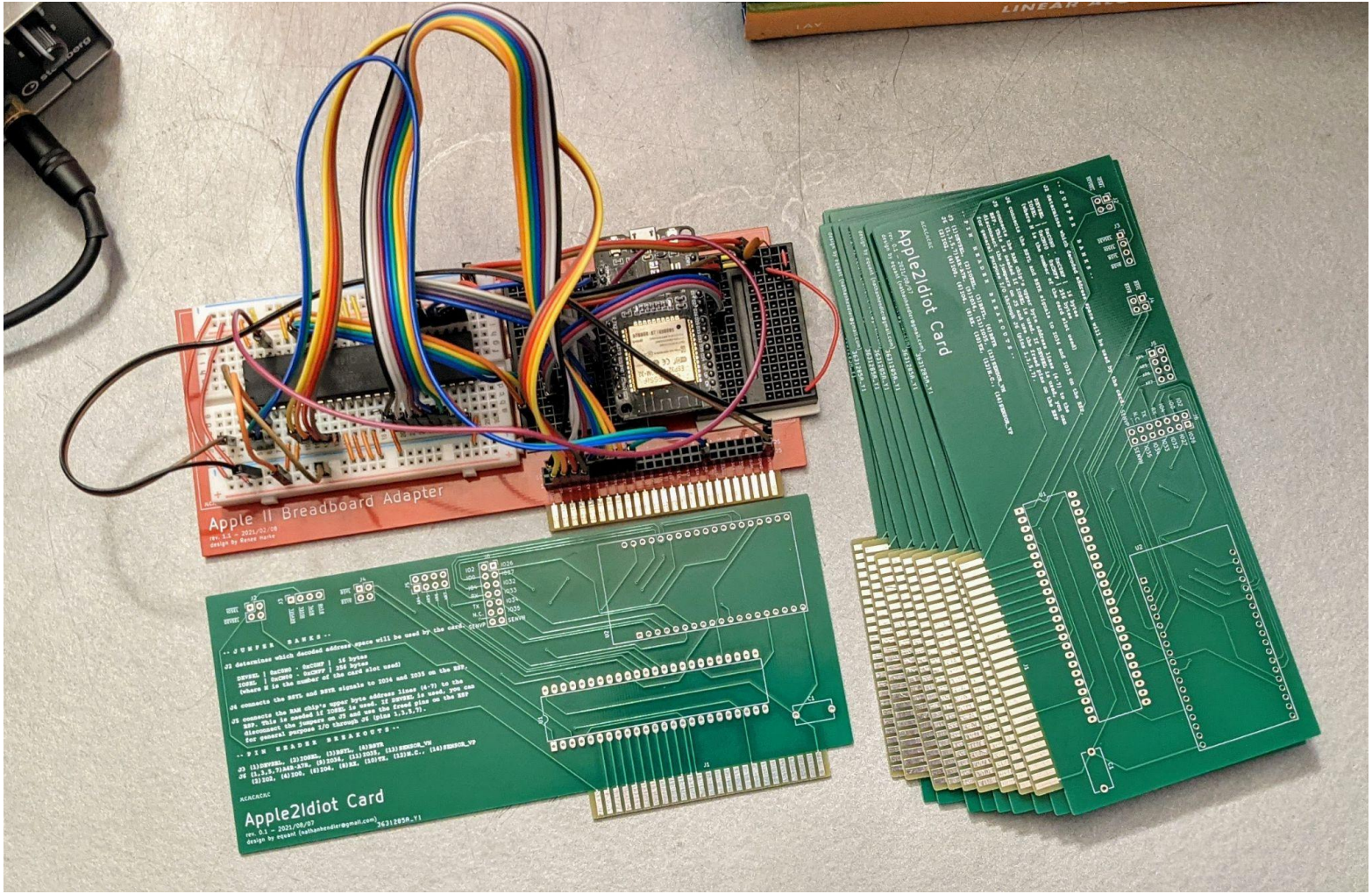


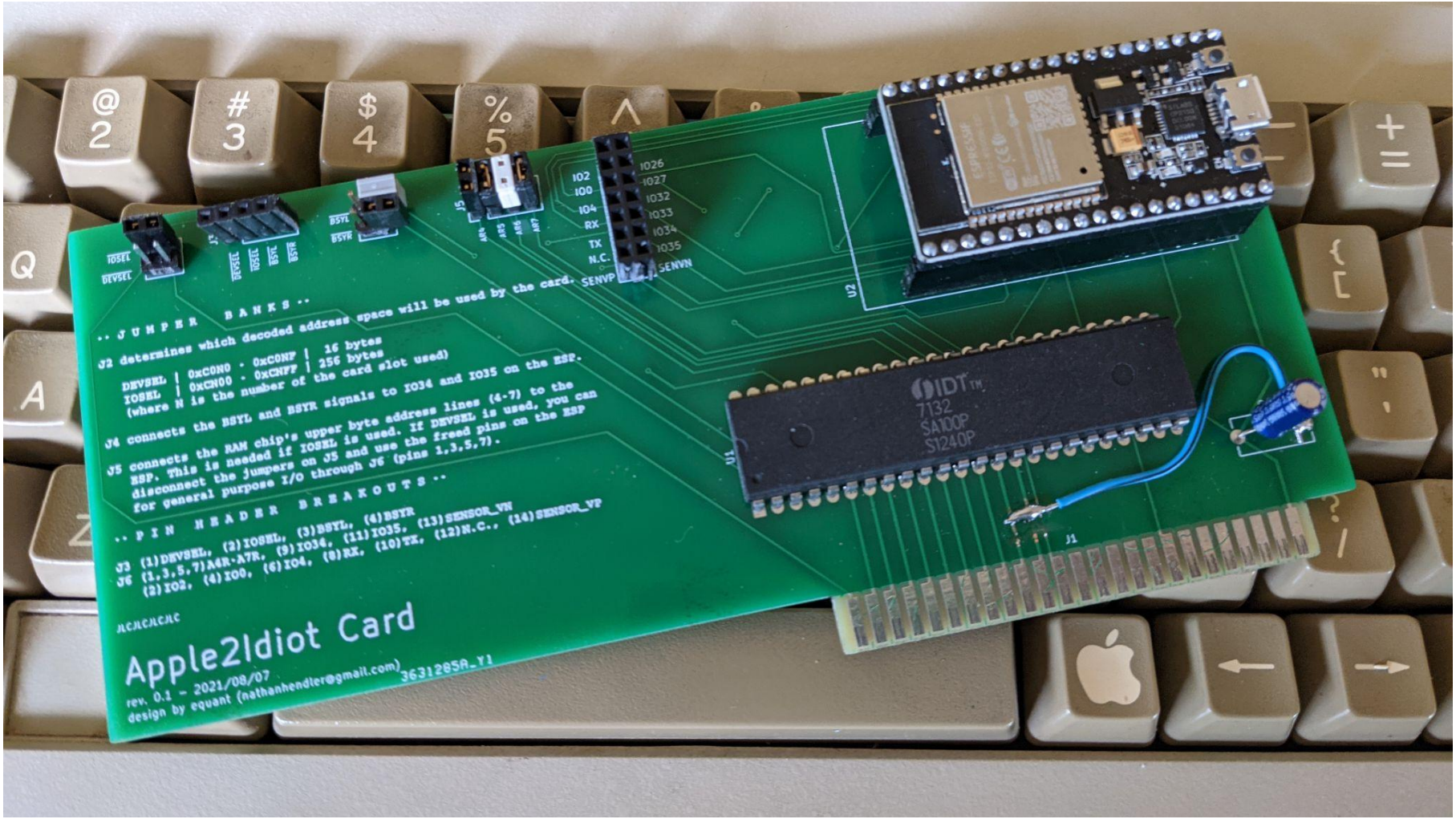


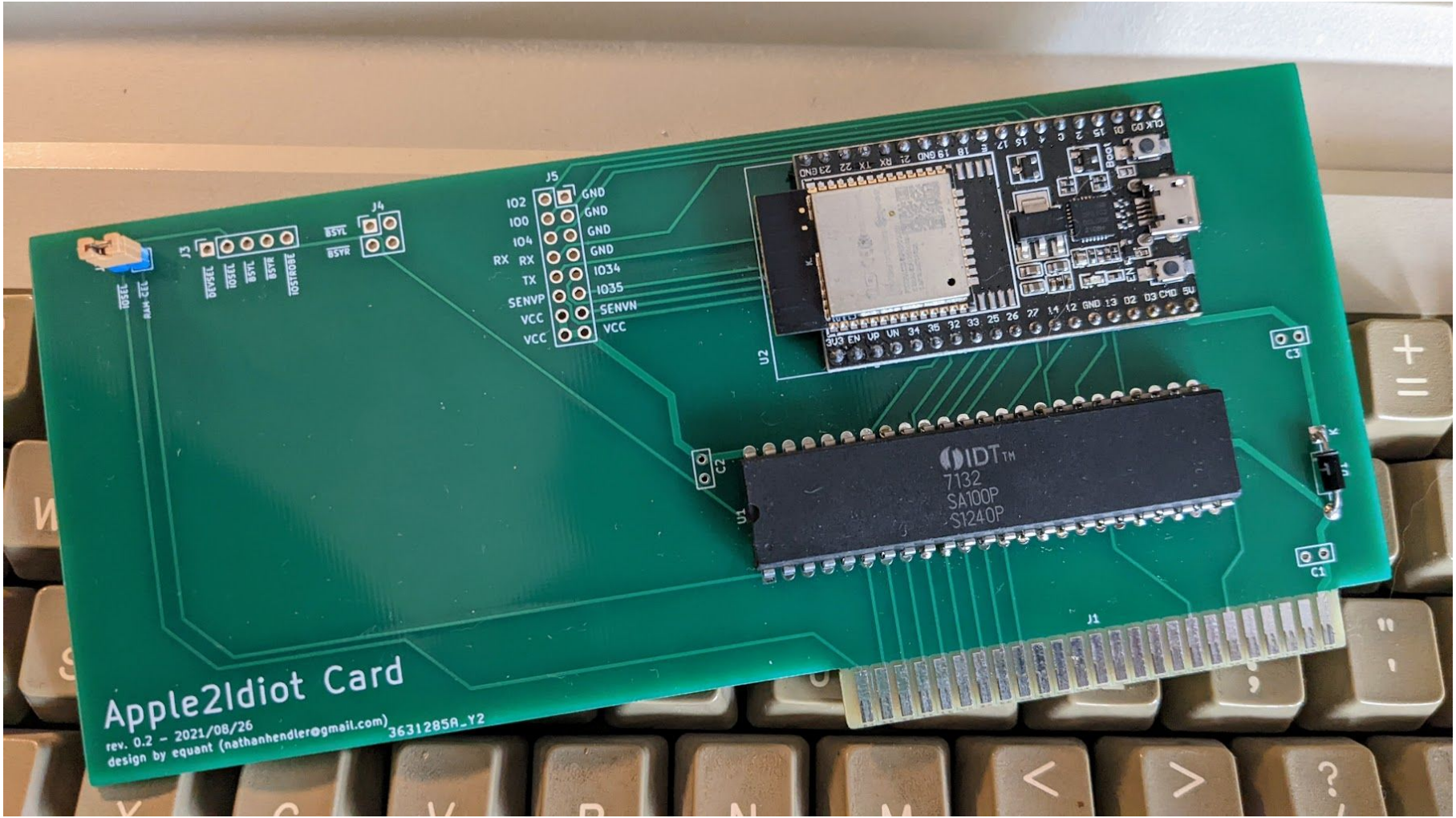


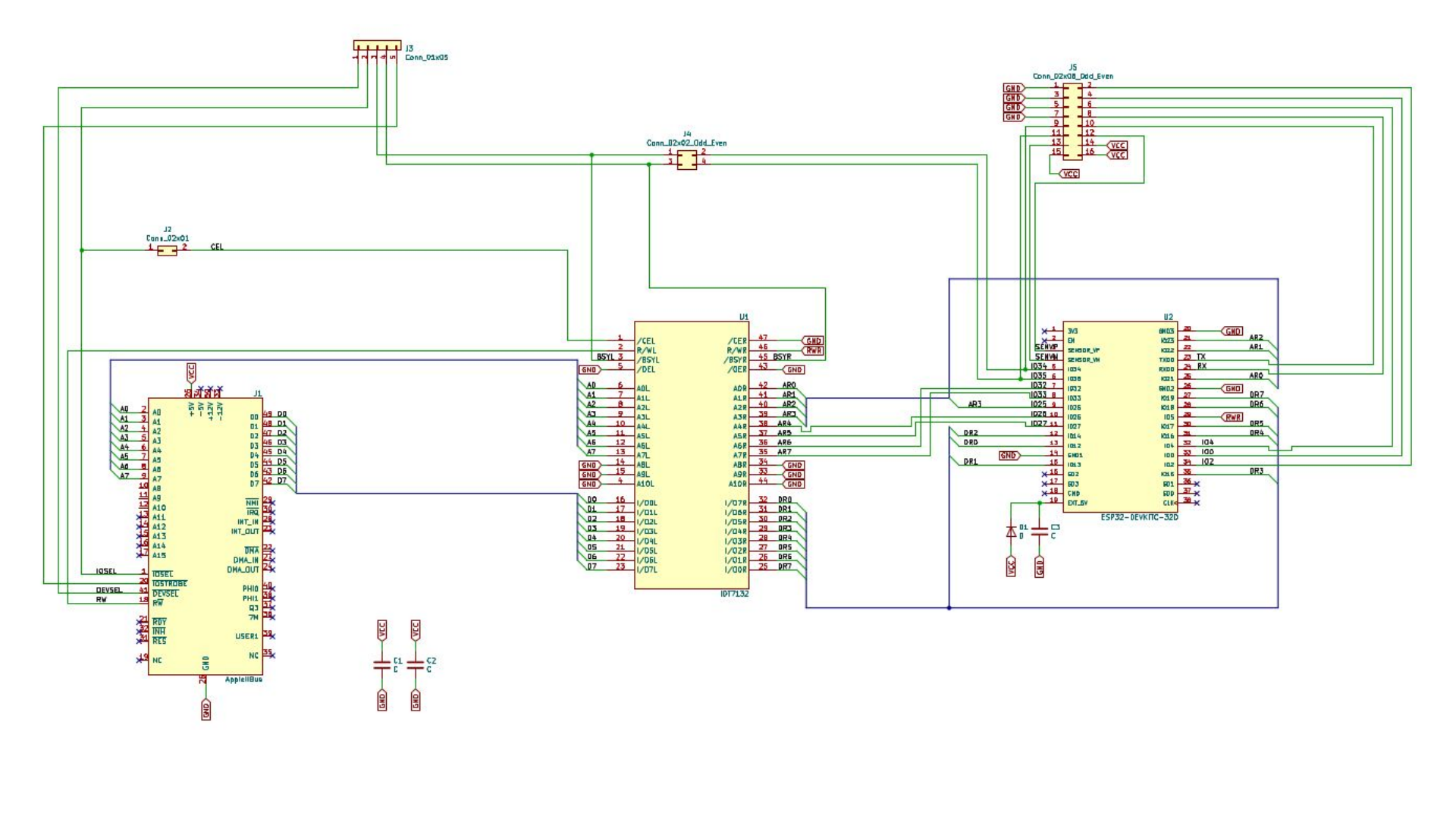


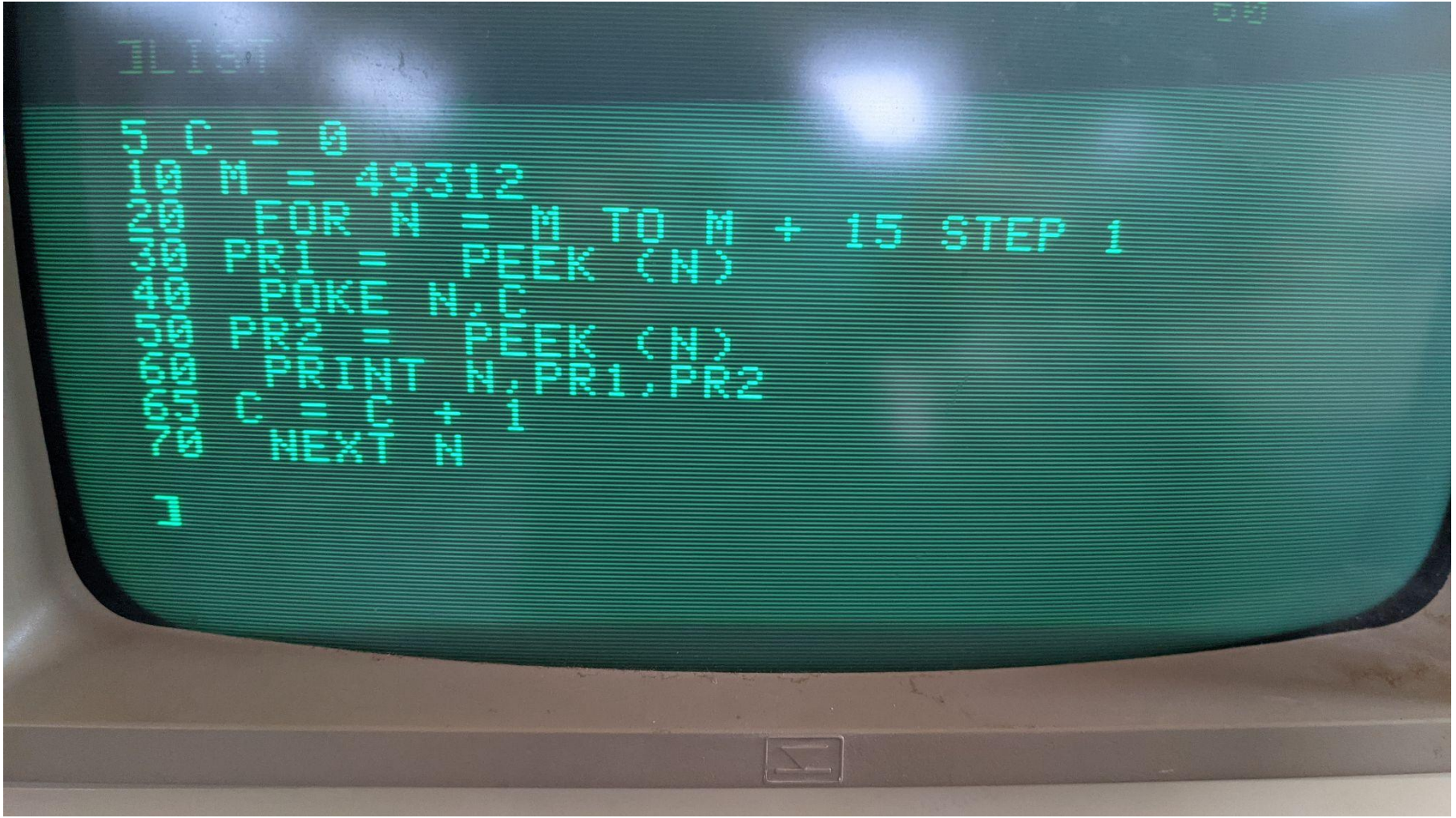

USBee AX PRO
MINI LOGIC
Analyzer
KEYBOARD ROM


Apple II Breadboard Adapter
rev. 1.1 – 2021/02/08
design by Renee Haine
Apple2Idiot Card
rev. 1.1 – 2021/08/07
design by equant (nathanhendler@gmail.com)
Apple2Idiot Card


.. J U M P E R B A N K S ..
J2 determines which decoded address space will be used by the card.
DEVSEL | 0xC0N0 - 0xC0NF | 16 bytes
IOSEL | 0xCN00 - 0xCNFF | 256 bytes
(where N is the number of the card slot used)
J4 connects the BSYL and BSYR signals to IO34 and IO35 on the ESP.
J5 connects the RAM chip's upper byte address lines (4-7) to the ESP. This is needed if IOSEL is used. If DEVSEL is used, you can disconnect the jumpers on J5 and use the freed pins on the ESP for general purpose I/O through J6 (pins 1,3,5,7).
.. P I N H E A D E R B R E A K O U T S ..
J3 (1)DEVSEL, (2)IOSEL, (3)BSYL, (4)BSYR
J6 (1,3,5,7)A4R-A7R, (9)IO34, (11)IO35, (13)SENSOR_VN
(2)IO2, (4)IO0, (6)IO4, (8)RX, (10)TX, (12)N.C., (14)SENSOR_VP
Apple2Idiot Card
rev. 0.1 - 2021/08/07
design by equant (nathanhendler@gmail.com)
IDT
7132
SA100P
S1240P


Apple2Idiot Card
rev. 0.2 – 2021/08/26
design by equant (nathanhendler@gmail.com)
3631285A_Y2
J5
IO2
IO0
IO4
RX
TX
SENVP
VCC
VCC
GND
GND
GND
GND
IO34
IO35
SENVN
VCC
J4
J3
U2
U1
C1
C2
C3
J1
IDT TM
7132
SA100P
S1240P


J3
Conn_01x05
J4
Conn_02x02_Odd_Even
J5
Conn_02x08_Odd_Even
J2
Conn_02x01
CEL
U1
IDT7132
U2
ESP32-DEVKITC-32D
J1
AppleIIBus
C1
C
C2
C
D1
D
C3
C
IOSEL
DEVSEL
RW
BSYL
BSYR
AR0
AR1
AR2
AR3
AR4
AR5
AR6
AR7
DR0
DR1
DR2
DR3
DR4
DR5
DR6
DR7
VCC
GND
TX
RX


LIST
5 C = 0
10 M = 49312
20 FOR N = M TO M + 15 STEP 1
30 PR1 = PEEK (N)
40 POKE N,C
50 PR2 = PEEK (N)
60 PRINT N,PR1,PR2
65 C = C + 1
70 NEXT N
]

```
90  NEXT N

]50 POKE N,R1+R1

]RUN
ADDRESS          DATA          DATA'
 49312           0             0
 49313           1             2
 49314           2             4
 49315           3             6
 49316           4             8
 49317           5             10
 49318           6             12
 49319           7             14
 49320           8             16
 49321           9             18
 49322           10            20
 49323           11            22
 49324           12            24
 49325           13            26
 49326           14            28
 49327           15            30

]%
```

On to the Software

apple2idiot

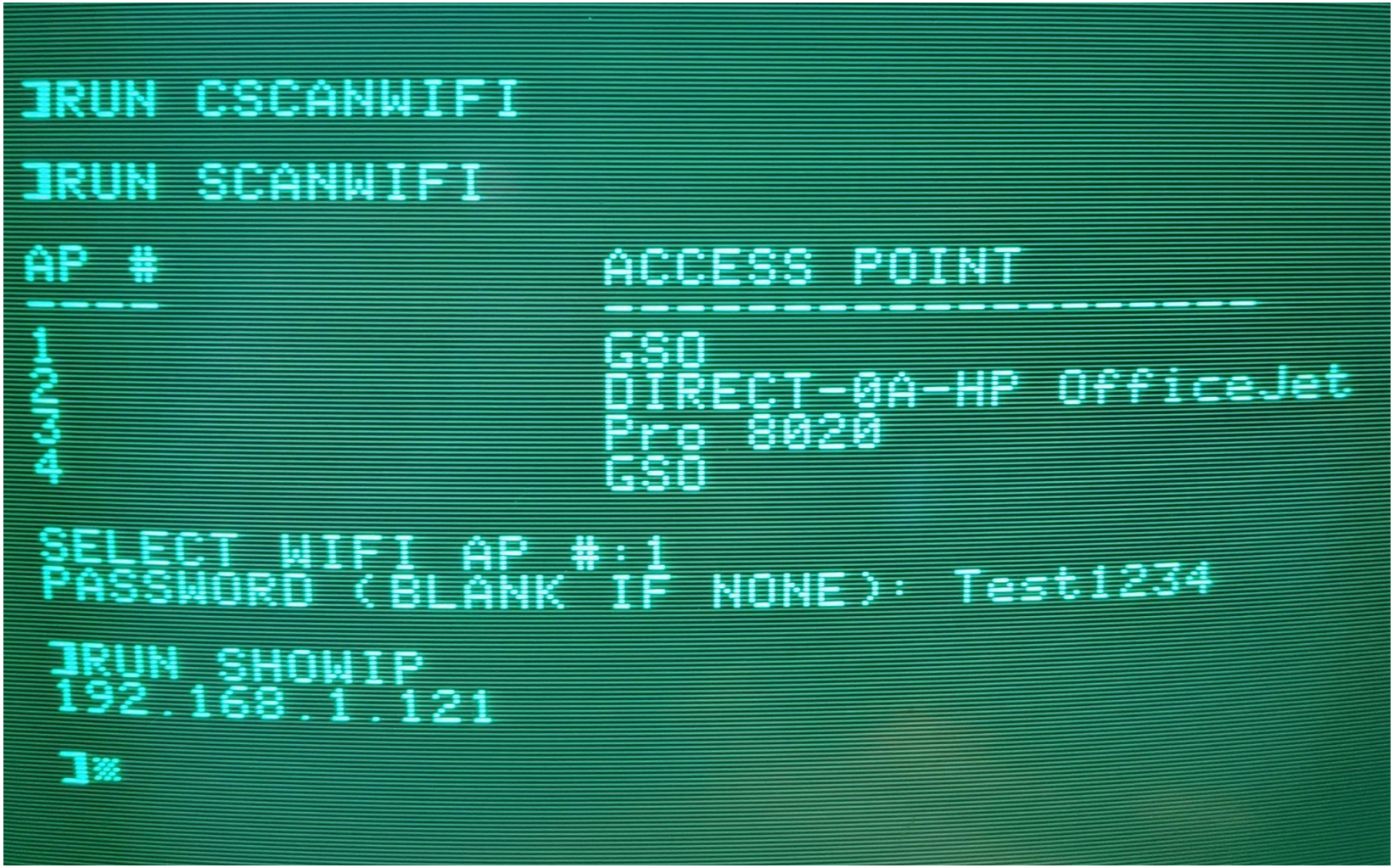
IDT7132 RAM

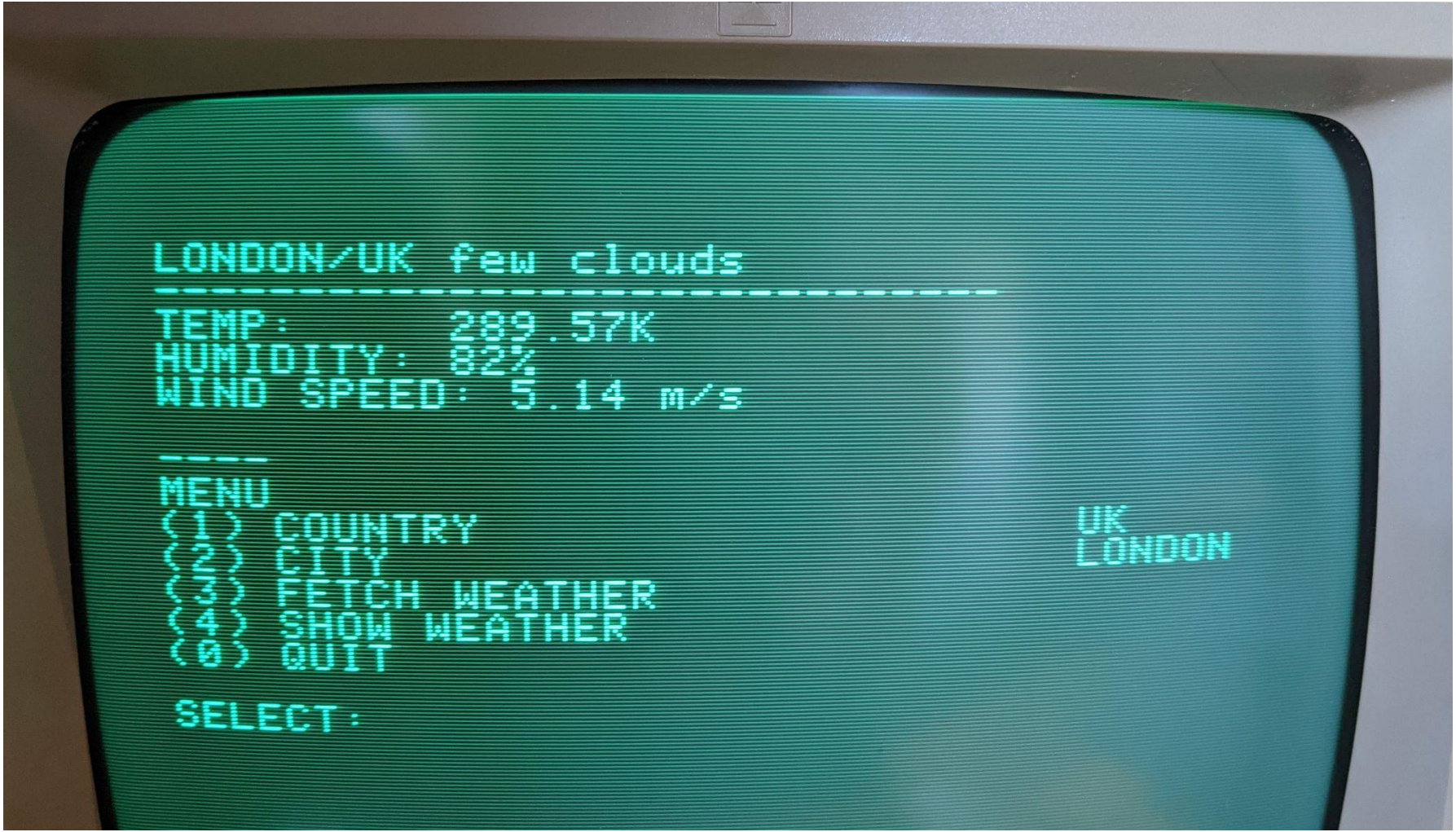
Write Data

Address: 0 Data: 255 Option: Notify Apple II Submit

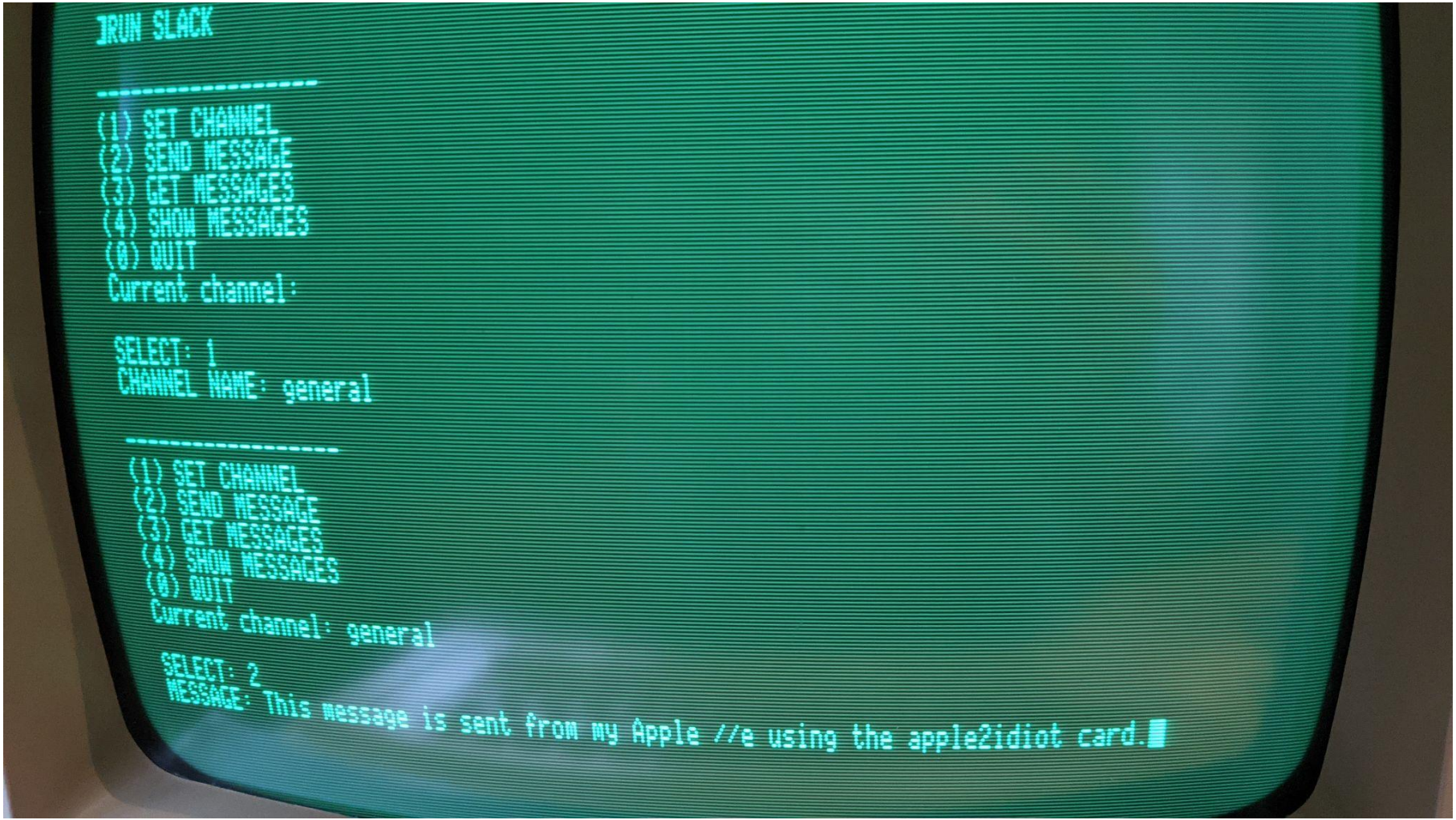
Memory Map

ADDRESS	DATA
49312	255
49313	0
49314	1
49315	2
49316	3
49317	0
49318	42
49319	27
49320	111
49321	188
49322	199
49323	55
49324	0
49325	0


]RUN CSCANWIFI
]RUN SCANWIFI
AP #
ACCESS POINT
1
GSO
2
DIRECT-0A-HP OfficeJet
3
Pro 8020
4
GSO
SELECT WIFI AP #:1
PASSWORD (BLANK IF NONE): Test1234
]RUN SHOWIP
192.168.1.121
]%


LONDON/UK few clouds
TEMP: 289.57K
HUMIDITY: 82%
WIND SPEED: 5.14 m/s
MENU
(1) COUNTRY
(2) CITY
(3) FETCH WEATHER
(4) SHOW WEATHER
(0) QUIT
SELECT:
UK
LONDON


1COUNTRY2CITY3FETCH4DISPLAY5QUIT
COUNTRY: MX CITY:MEXICO CITY
MEXICO CITY (MX)
TEMP: 294.23 K
HUMIDITY: 47%
WIND SPEED: 1.91 M/S
WIND DIR: 21
SUMMARY: CLOUDS, OVERCAST CLOUDS
MENU SELECTION:


]RUN SLACK
(1) SET CHANNEL
(2) SEND MESSAGE
(3) GET MESSAGES
(4) SHOW MESSAGES
(0) QUIT
Current channel:
SELECT: 1
CHANNEL NAME: general
(1) SET CHANNEL
(2) SEND MESSAGE
(3) GET MESSAGES
(4) SHOW MESSAGES
(0) QUIT
Current channel: general
SELECT: 2
MESSAGE: This message is sent from my Apple //e using the apple2idiot card.

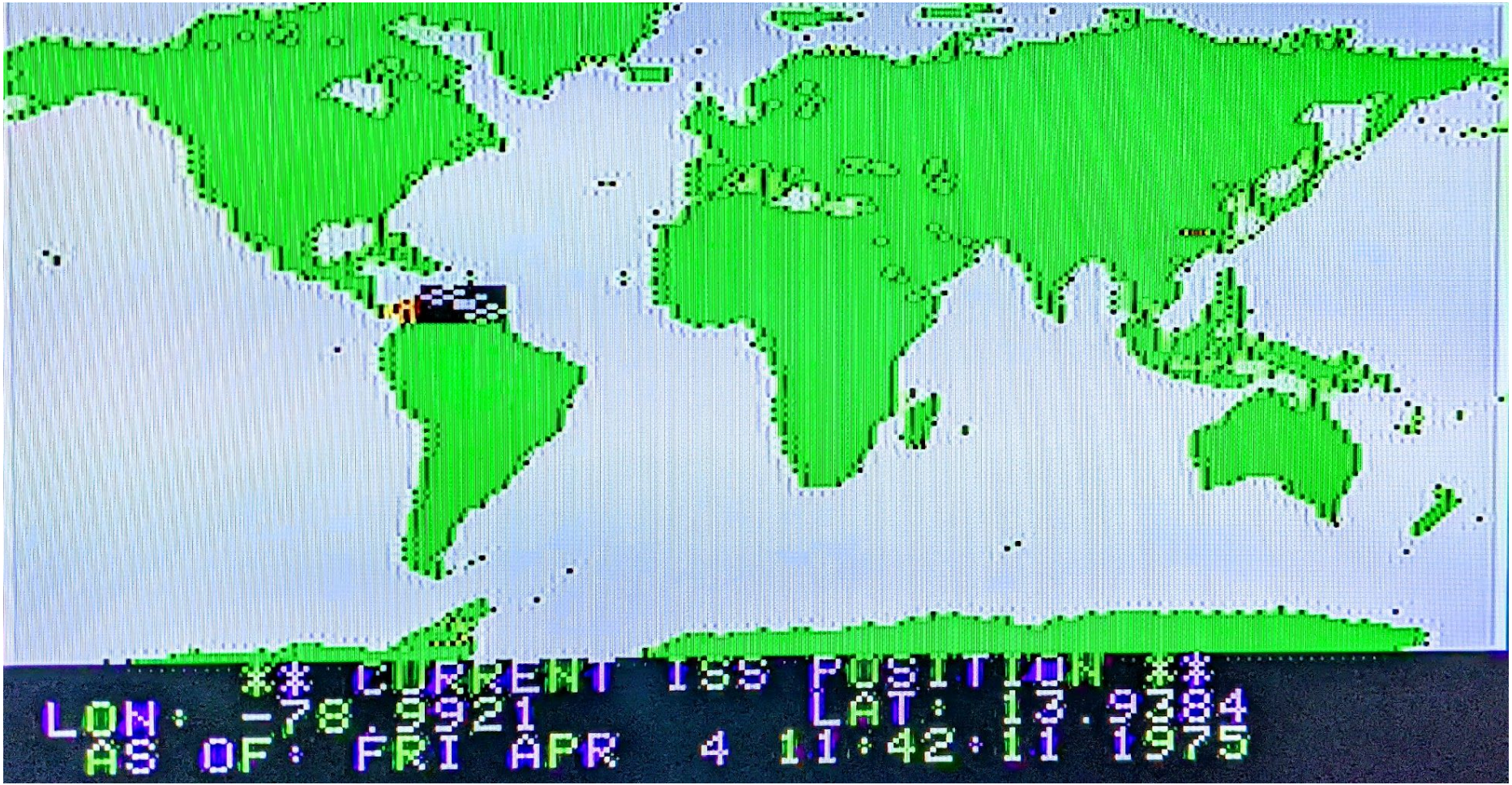
```
FROMESP:   \ LOCAL GAME \ SETTINGS \ QUIT
AI MOVE:  D8F6




           8 R . B . K . . R
           7 P P P P . P P P
           6 . . N . . Q . .
           5 . . . . N . . .
           4 . B . . N . . .
           3 . . . . . . . .
           2 P . P P N P P P
           1 R . B Q K B . R
             A B C D E F G H

+----------------------------------------+
!  YOU LOSE.
+----------------------------------------+
(PRESS A KEY)

YOUR MOVE:c1b2   /
```


** CURRENT ISS POSITION **
LON: -78.9921 LAT: 13.9784
AS OF: FRI APR 4 11:42:11 1975

```
App loop: 0
Checking ID: 202
App loop: 1
Checking ID: 200
Received a command for ISS()
ISS() handleCommand
ISS() COMMAND_GET_ISS
WRITE: 6 -> 0
ISS() before fetch_iss()
    http://api.open-notify.org/iss-now.json
    After GET()
    httpCode:200
++++++++++++++++++++++++
{"iss_position": {"longitude": "-120.4488", "latitude": "-10.6903"}, "timestamp": 1658442399, "message": "success"}
++++++++++++++++++++++++
----------------------
{
  "iss_position": {
    "longitude": "-120.4488",
    "latitude": "-10.6903"
  },
  "timestamp": 1658442399,
  "message": null
}----------------------
{
  "longitude": "-120.4488",
  "latitude": "-10.6903"
}----------------------
...command for ISS() handled
```

So what is it?

- A very **general purpose** microcontroller expansion card…
  - Web APIs
    - Weather, Chess, D&D API, Plex controller, Google sheets importer for visicalc
  - Webserver or serial output for Apple II software debugging
  - Coprocessor for encryption
  - Sensors, ADC, MQTT/IOT
  - Adhock network chat/games
  - Bluetooth
  - NTP -> Prodos time

- …That becomes a very **specific purpose** card once you put the firmware on the ESP32.

Pros and Cons

- Pros
  - Easy and inexpensive to build at home. Low chip count. Through hole components.
  - Card handles hard stuff (e.g. JSON parsing, https)
  - Easily hackable
- Cons
  - To do something, you have to write code for Apple II and ESP32.
  - Card has to have the correct firmware for the software you want to run.
- Maybe you should look at:
  - Fujinet
  - Uthernet II

Future?

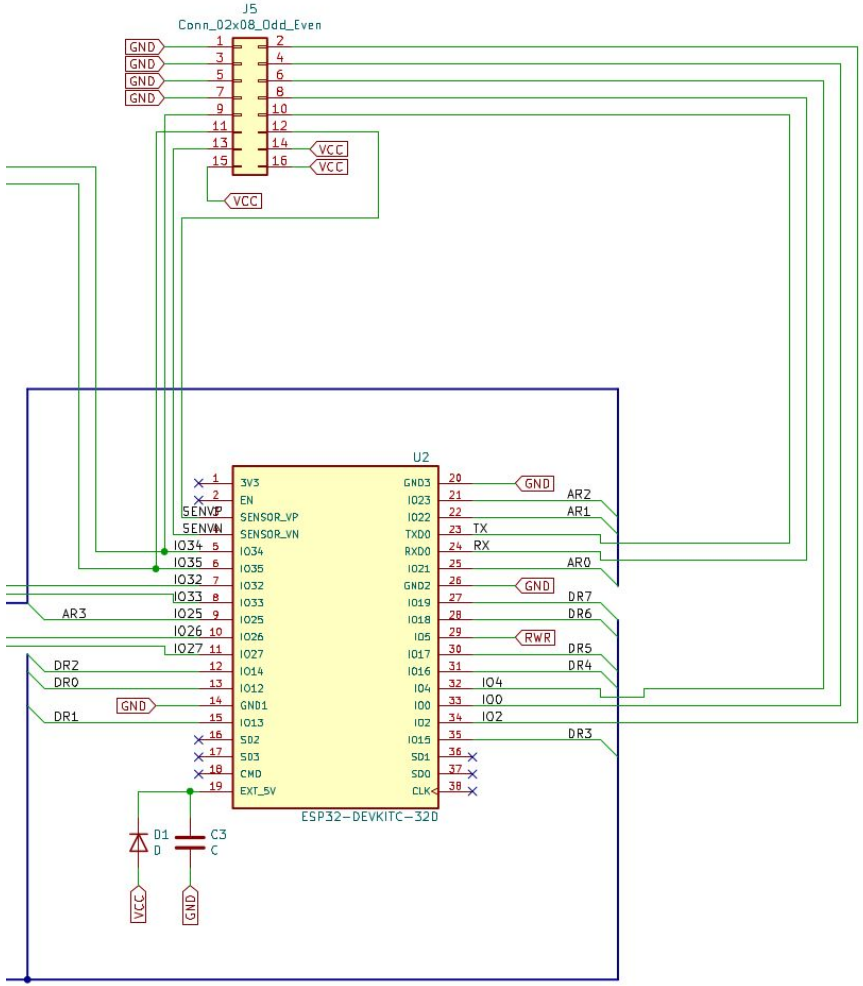
- Multiplayer
- Eigengrau’s Town generator + D&D API dynamic adventure game.

A bit about the ESP32

- Can be programmed with Arduino IDE… but also can be programmed with Espressif’s c/asm toolchains, or, NodeMCU LUA.
- Pins can be a nightmare…

A bit about the ESP32

No.	Name	Type	GPIO	True GPIO	RTC_GPIO	ADC	TOUCH	DAC	I2C	SPI	Controller	UART	QSPI	JTAG	SDIO Slave
25	IO00	I/O	GPIO00	No	RTC_GPIO11	ADC2_CH01	TOUCH1								
35	TXD0	I/O	GPIO01	No								U0TXD			
24	IO02	I/O	GPIO02	No	RTC_GPIO12	ADC2_CH02	TOUCH2				HS2_DATA0		HSPIWP		
34	RXD0	I/O	GPIO03	No								U0RXD			
26	IO04	I/O	GPIO04		RTC_GPIO10	ADC2_CH0	TOUCH0				HS2_DATA1		HSPIHD		
29	IO05	I/O	GPIO05							SS/CS			VSPICS0		
20	SCK/CLK*	I/O	GPIO06	No	UNUSABLE !!! (Flash memory)							U1CTS	SPICLK		SD_CLK
21	SDO/SD0*	I/O	GPIO07	No	UNUSABLE !!! (Flash memory)							U2RTS	SPIQ		SD_DATA0
22	SDI/SD1*	I/O	GPIO08	No	UNUSABLE !!! (Flash memory)							U2CTS	SPID		SD_DATA1
17	SHD/SD2*	I/O	GPIO09	No	UNUSABLE !!! (Flash memory)							U1RXD	SPIHD		SD_DATA2
18	SWP/SD3*	I/O	GPIO10	No	UNUSABLE !!! (Flash memory)							U1TXD	SPIWP		SD_DATA3
19	SCS/CMD*	I/O	GPIO11	No	UNUSABLE !!! (Flash memory)							U1RTS	SPICS0		
14	IO12	I/O	GPIO12	No	RTC_GPIO15	ADC2_CH05	TOUCH5				HS2_DATA2		HSPIQ	MTDI	
16	IO13	I/O	GPIO13		RTC_GPIO14	ADC2_CH04	TOUCH4				HS2_DATA3		HSPID	MTCK	
13	IO14	I/O	GPIO14		RTC_GPIO16	ADC2_CH06	TOUCH6				HS2_CLK		HSPICLK	MTMS	
23	IO15	I/O	GPIO15		RTC_GPIO13	ADC2_CH03	TOUCH3				HS2_CMD		HSPICS0	MTDO	
27	IO16	I/O	GPIO16	Not available on WROVER								U2RXD			
28	IO17	I/O	GPIO17	Not available on WROVER								U2TXD			
30	IO18	I/O	GPIO18							SCLK			VSPICLK		
31	IO19	I/O	GPIO19							MISO		U0CTS	VSPIQ		
33	IO21	I/O	GPIO21						SDA1				VSPIHD		
36	IO22	I/O	GPIO22						SCL1			U0RTS	VSPIWP		
37	IO23	I/O	GPIO23							MOSI			VSPID		
10	IO25	I/O	GPIO25		RTC_GPIO06	ADC2_CH08		DAC_1							
11	IO26	I/O	GPIO26		RTC_GPIO07	ADC2_CH09		DAC_2							
12	IO27	I/O	GPIO27		RTC_GPIO17	ADC2_CH07	TOUCH7								
8	IO32	I/O	GPIO32		RTC_GPIO09	ADC1_CH04	TOUCH9								
9	IO33	I/O	GPIO33		RTC_GPIO08	ADC1_CH05	TOUCH8								
6	IO34	I	GPIO34	No	RTC_GPIO04	ADC1_CH06									
7	IO35	I	GPIO35	No	RTC_GPIO05	ADC1_CH07									
4	SENSOR_VP	I	GPIO36	No	RTC_GPIO00	ADC1_CH0									
5	SENSOR_VN	I	GPIO39	No	RTC_GPIO03	ADC1_CH03									


J5
Conn_02x08_Odd_Even
GND
GND
GND
GND
VCC
VCC
VCC
U2
3V3
EN
SENSOR_VP
SENSOR_VN
IO34
IO35
IO32
IO33
IO25
IO26
IO27
IO14
IO12
GND1
IO13
SD2
SD3
CMD
EXT_5V
GND3
IO23
IO22
TXD0
RXD0
IO21
GND2
IO19
IO18
IO5
IO17
IO16
IO4
IO0
IO2
IO15
SD1
SD0
CLK
GND
RWR
GND
TX
RX
AR3
DR2
DR0
DR1
AR2
AR1
AR0
DR7
DR6
DR5
DR4
IO4
IO0
IO2
DR3
ESP32-DEVKITC-32D
D1
D
C3
C
VCC
GND