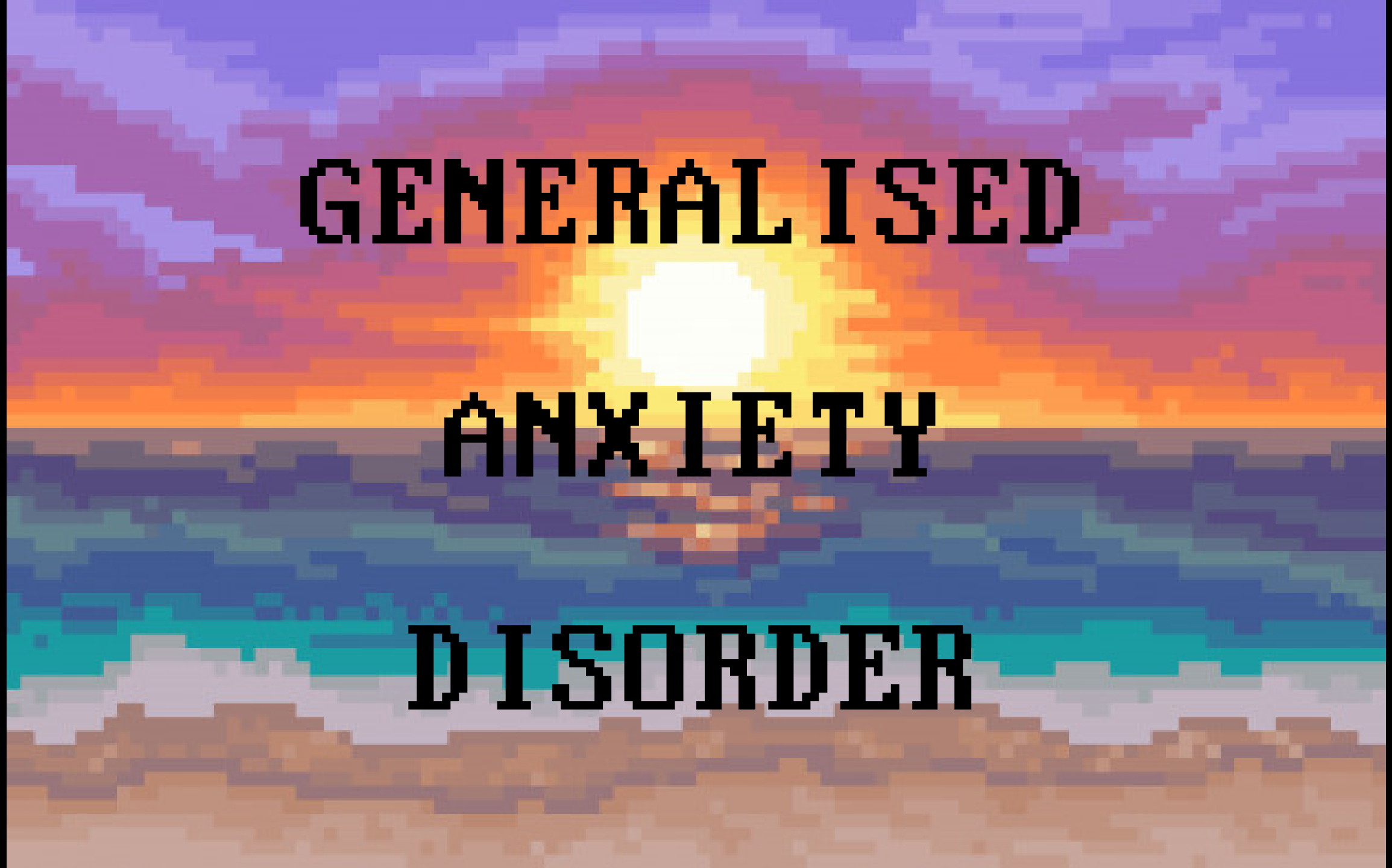
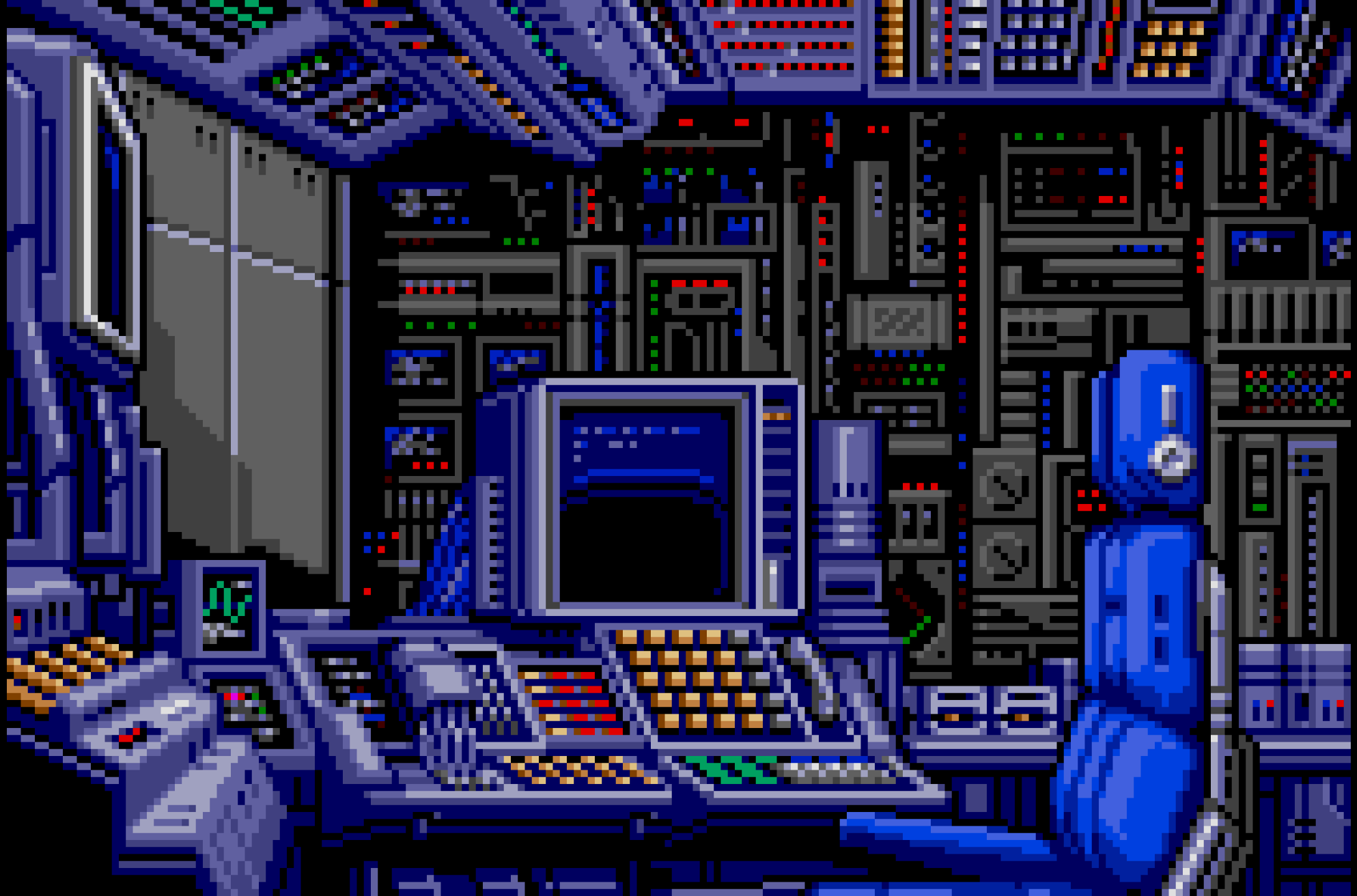

RAI 1

🎓 TFCon presents

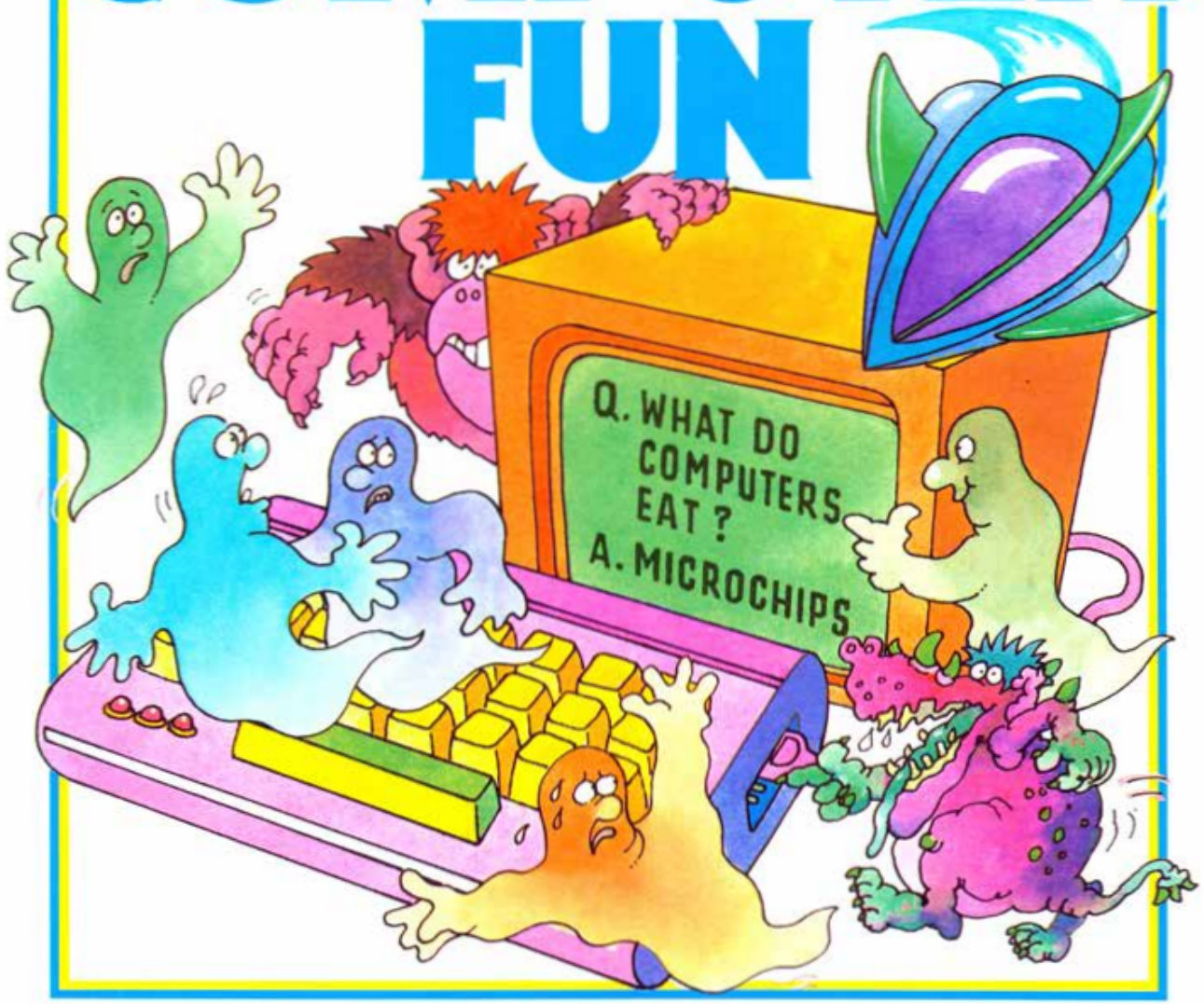
GOOD COMPUTER


GENERALISED
ANXIETY
DISORDER



FIRST COMPUTER LIBRARY

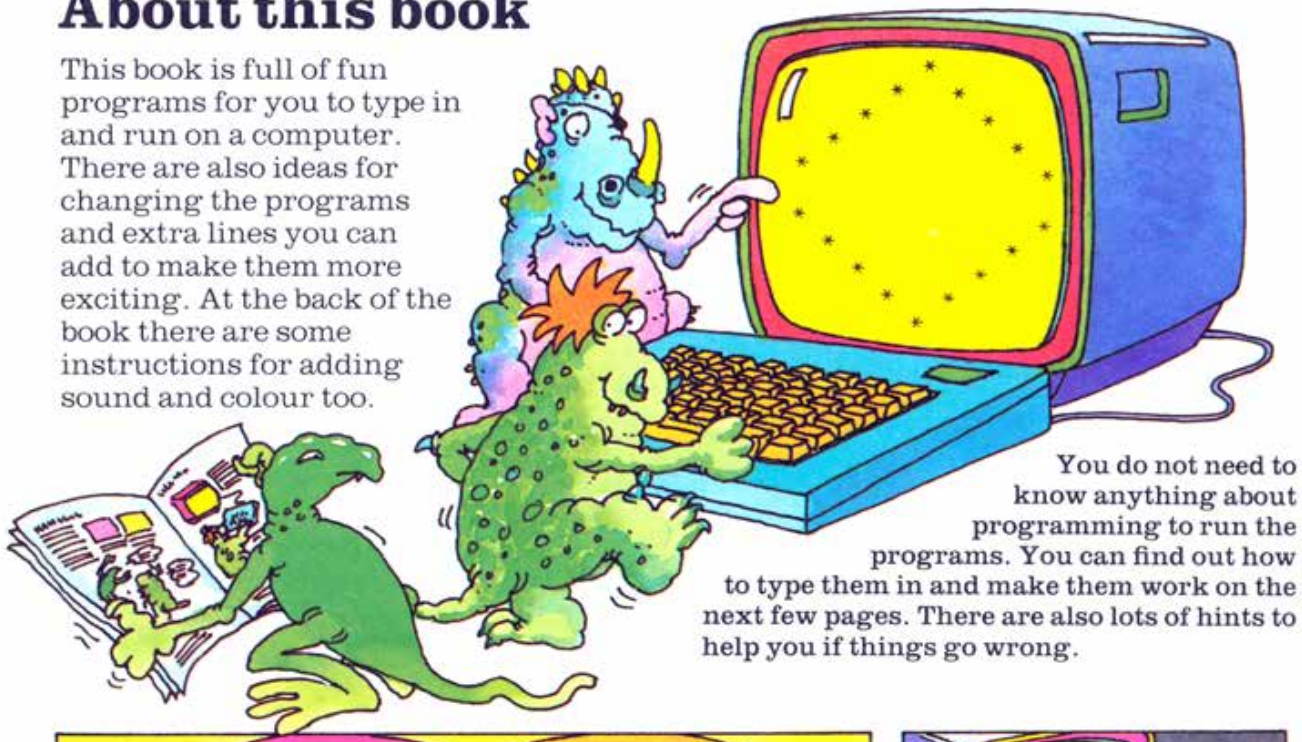
COMPUTER FUN



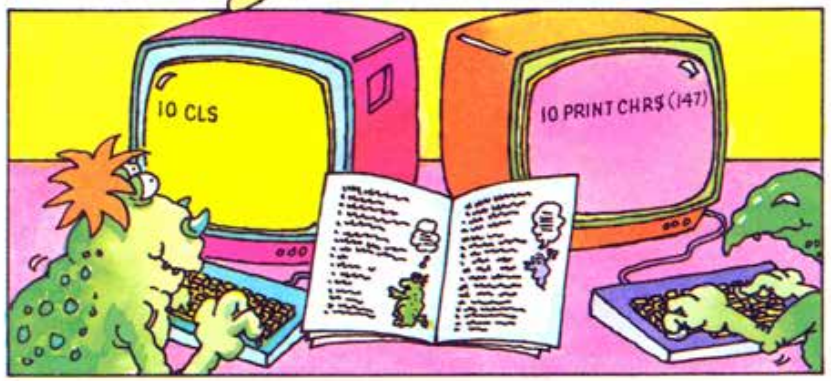
USBORNE

About this book

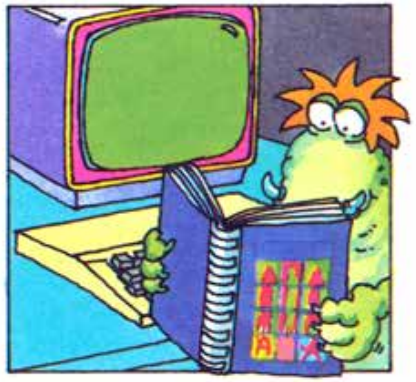
This book is full of fun programs for you to type in and run on a computer. There are also ideas for changing the programs and extra lines you can add to make them more exciting. At the back of the book there are some instructions for adding sound and colour too.



You do not need to know anything about programming to run the programs. You can find out how to type them in and make them work on the next few pages. There are also lots of hints to help you if things go wrong.



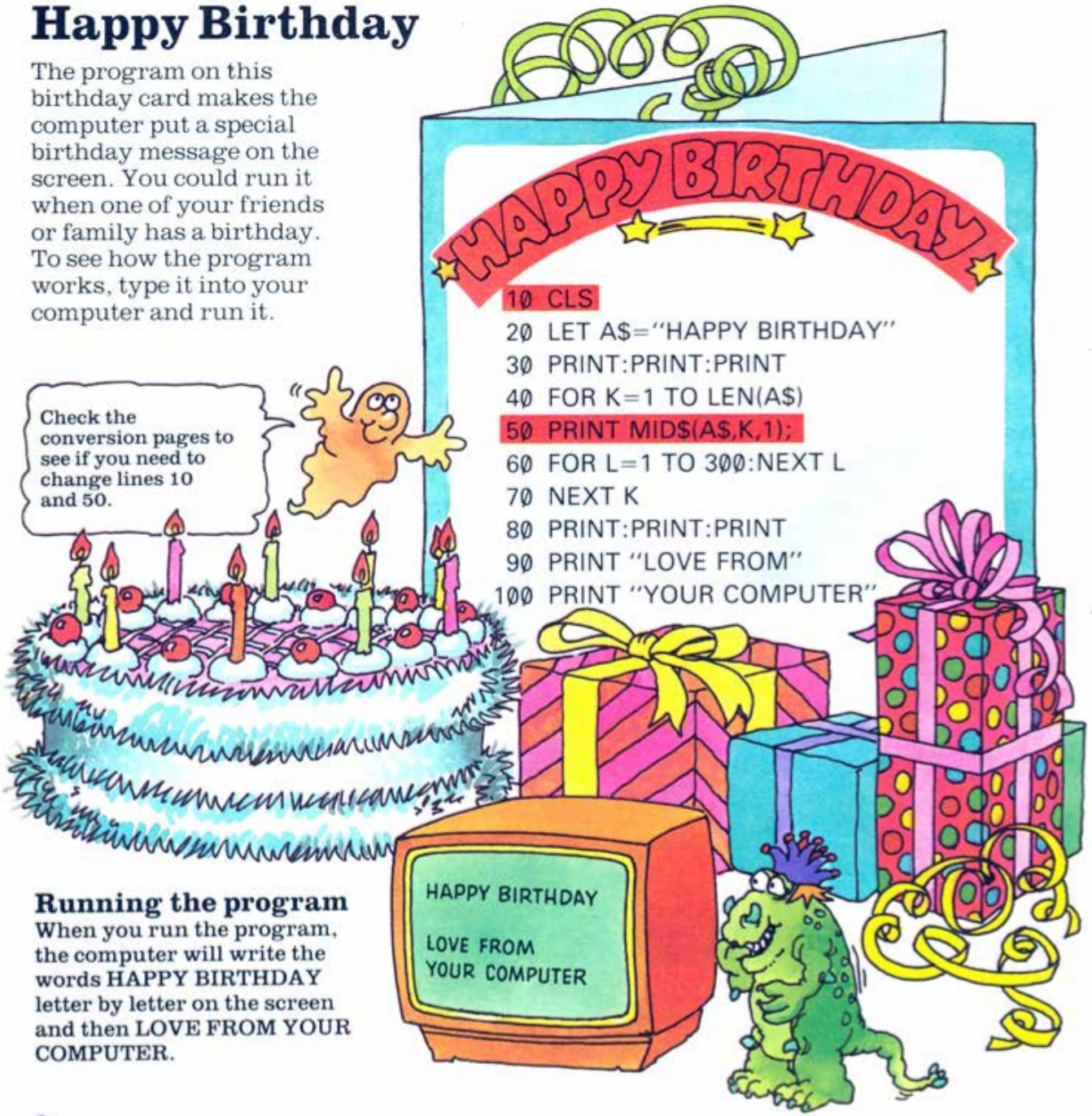
The programs are written in a computer language called BASIC. Each home computer understands a slightly different version of BASIC and you will need to change some program lines to suit your computer. These lines are clearly marked and on pages 40-46 there are lists of line changes for Spectrum, BBC, Electron, Commodore 64, VIC 20 and Apple computers.



If you have a different make of computer, you may be able to work out the changes you need to make from your computer's manual.

Happy Birthday

The program on this birthday card makes the computer put a special birthday message on the screen. You could run it when one of your friends or family has a birthday. To see how the program works, type it into your computer and run it.

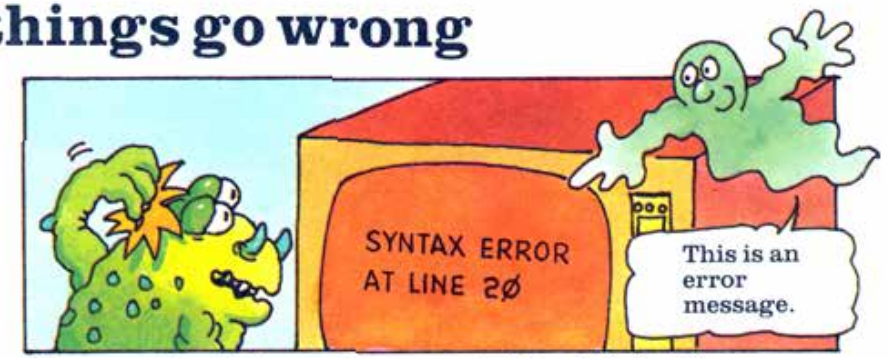


Running the program

When you run the program, the computer will write the words HAPPY BIRTHDAY letter by letter on the screen and then LOVE FROM YOUR COMPUTER.

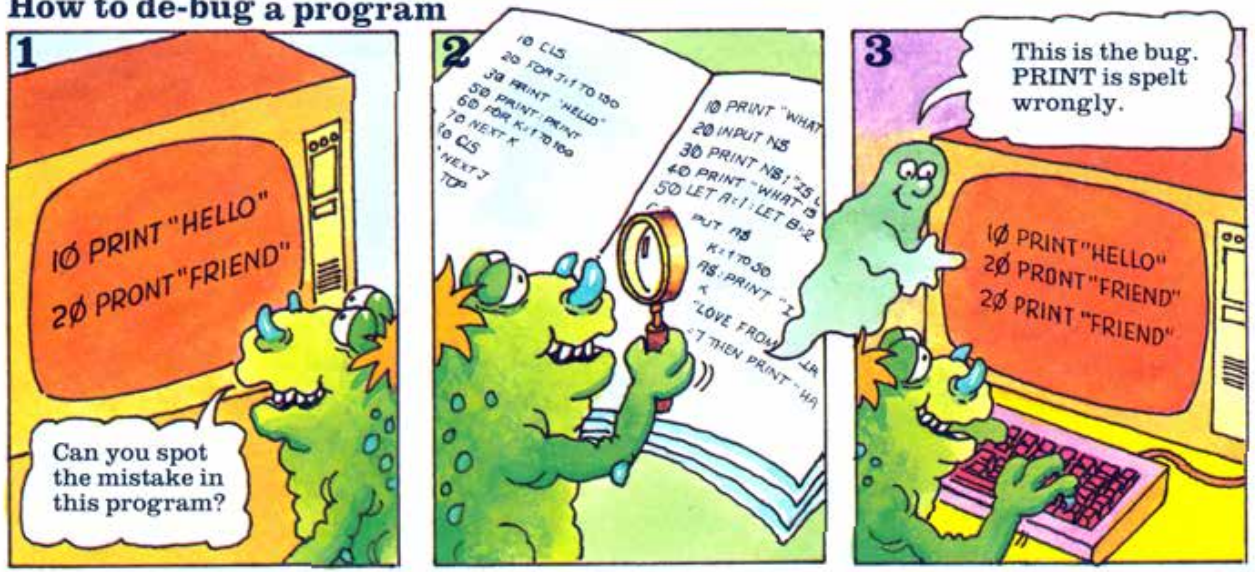
What to do if things go wrong

If a program does not work you have probably made a mistake typing it in. It is very easy to make mistakes when you type in programs. Mistakes in programs are called bugs. Some computers let you know if there is a bug in a program line when you press RETURN or ENTER, but most wait until you try to run the program.



The computer tells you there is a bug in a program by putting a special message on the screen like the one above. This is called an error message. It usually tells you what sort of mistake it is and what line it is in. Error messages are different on different computers and are often written in a sort of code. To find out what they mean, look in your computer's manual.

How to de-bug a program

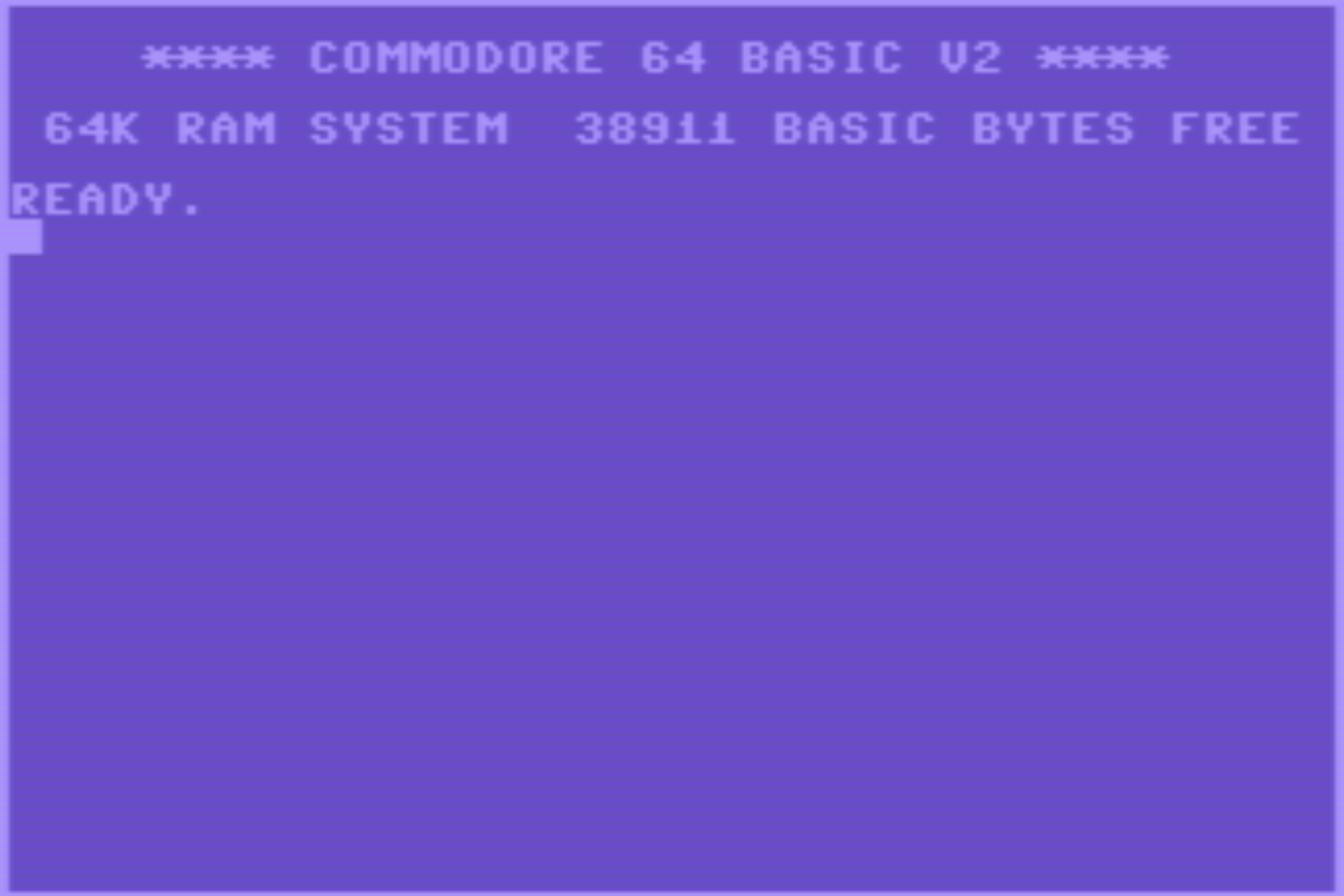


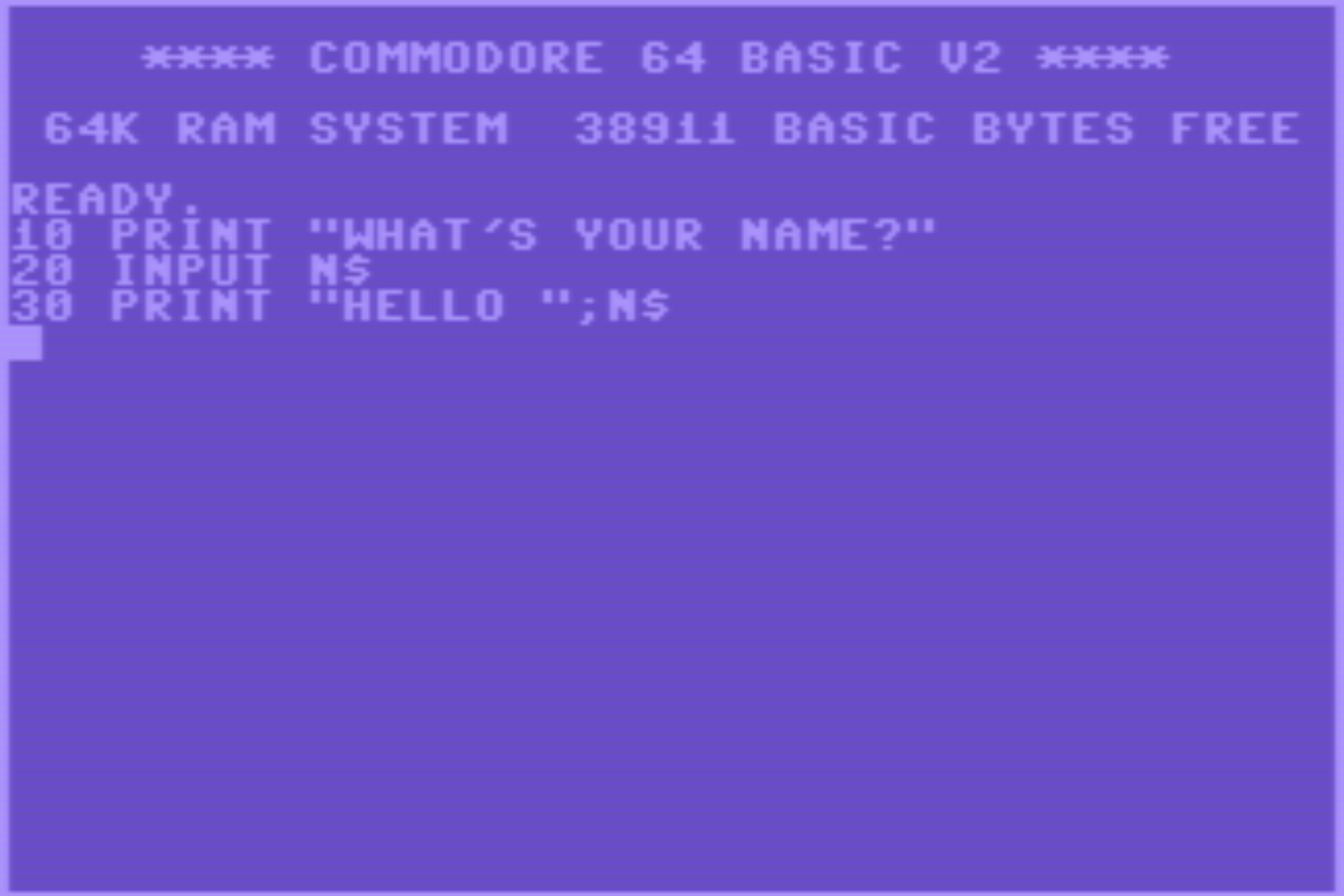
First type LIST and press RETURN* to display the program lines on the screen. Then check through the program lines for mistakes.

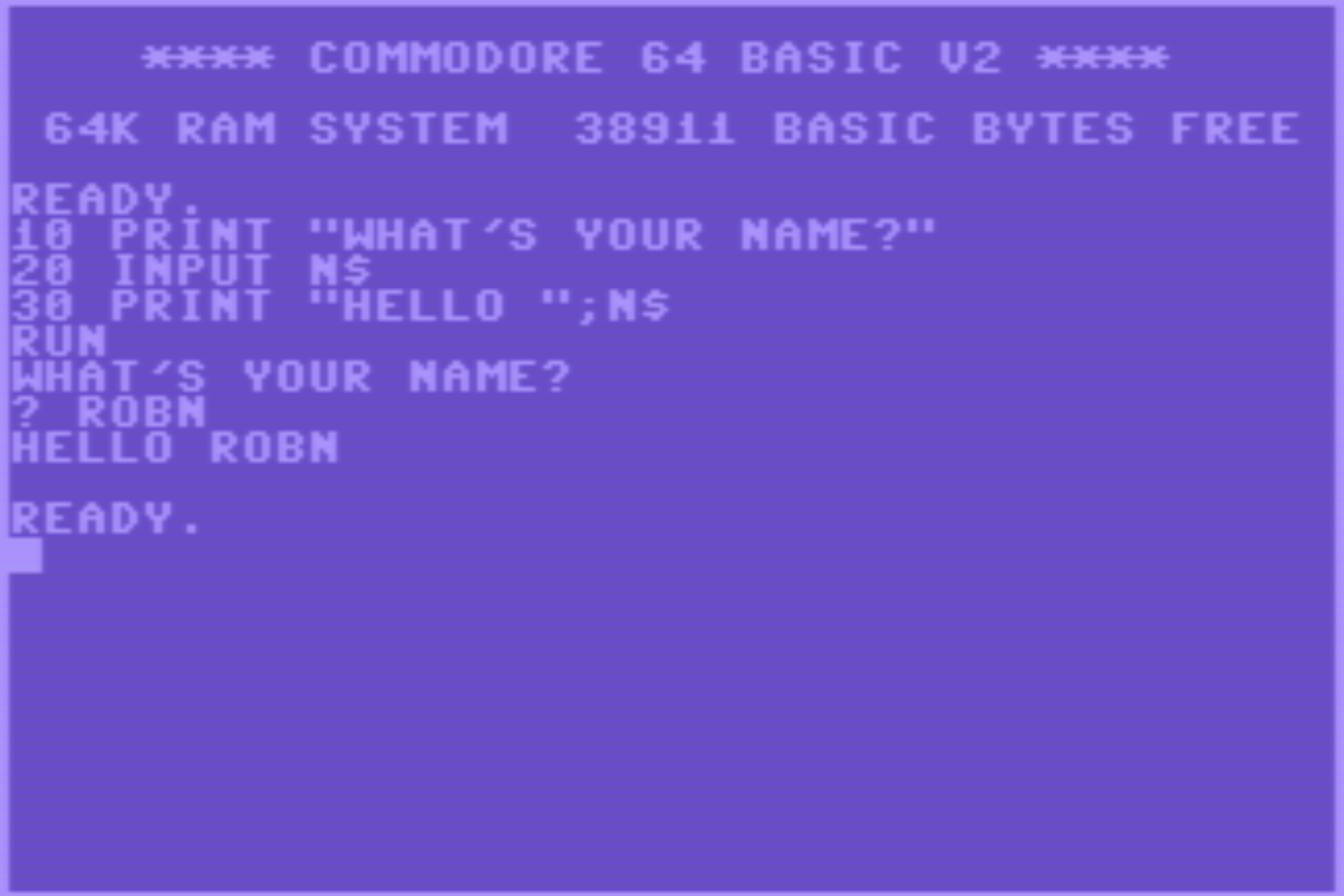
Bugs are not easy to find, but it is easier to spot them if you know what to look for. The bug spotting guide on page 47 gives you some handy tips and hints.

When you spot the bug, type the whole line again and press RETURN. Now if you list the program again, you will see that the new line has replaced the old one.

*If your computer's key is marked ENTER, press it each time you see RETURN from now on.


**** COMMODORE 64 BASIC V2 ****
64K RAM SYSTEM 38911 BASIC BYTES FREE
READY.


**** COMMODORE 64 BASIC V2 ****
64K RAM SYSTEM 38911 BASIC BYTES FREE
READY.
10 PRINT "WHAT'S YOUR NAME?"
20 INPUT N$
30 PRINT "HELLO ";N$


**** COMMODORE 64 BASIC V2 ****
64K RAM SYSTEM 38911 BASIC BYTES FREE
READY.
10 PRINT "WHAT'S YOUR NAME?"
20 INPUT N$
30 PRINT "HELLO ";N$
RUN
WHAT'S YOUR NAME?
? ROBN
HELLO ROBN
READY.

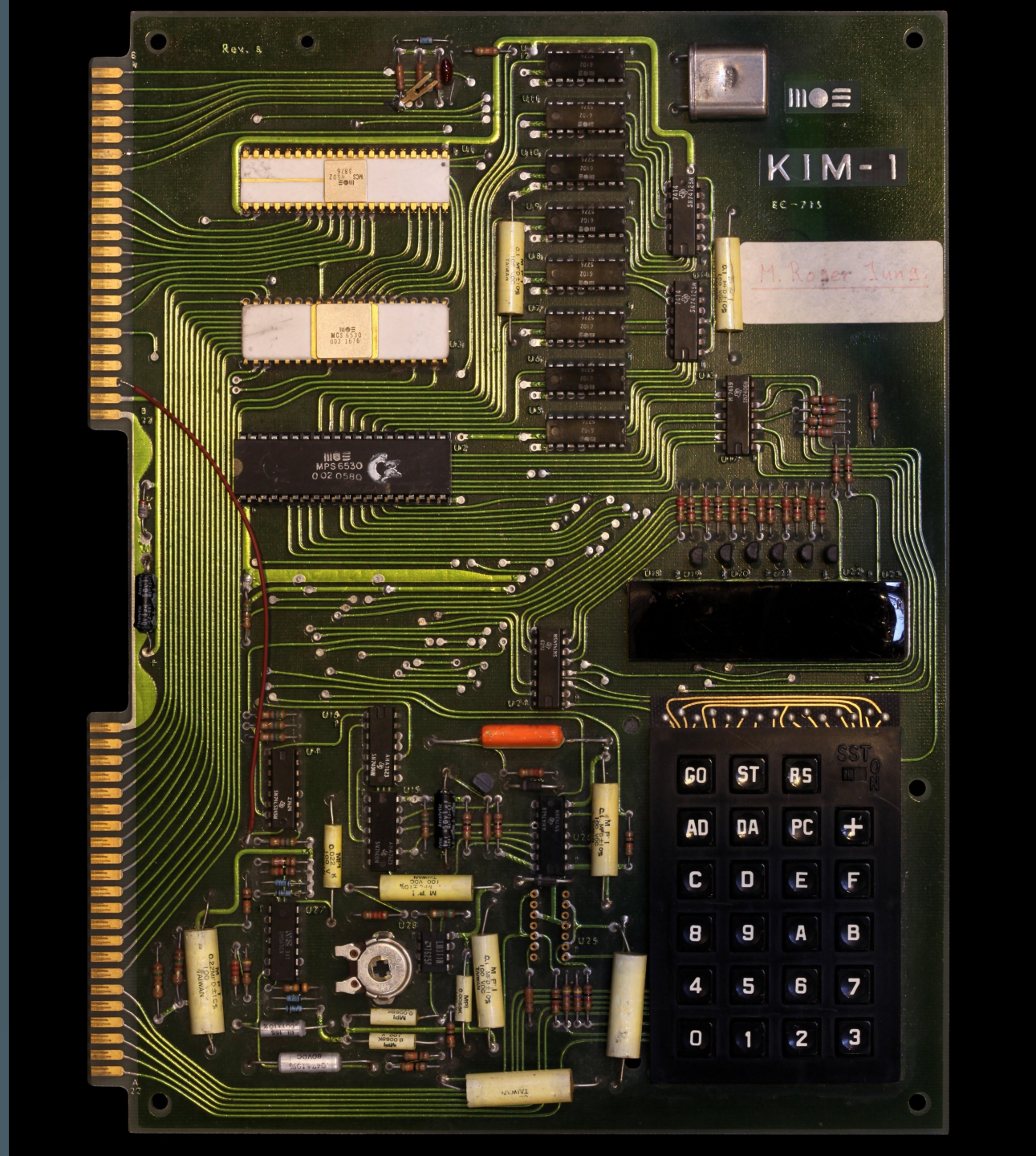
GOOD COMPUTER

A late-70s/early-80s computer, made in 2020

- 8-bit
- BASIC
- Written in assembly
- Easy to connect to other things
- (Pandemic-resistant)

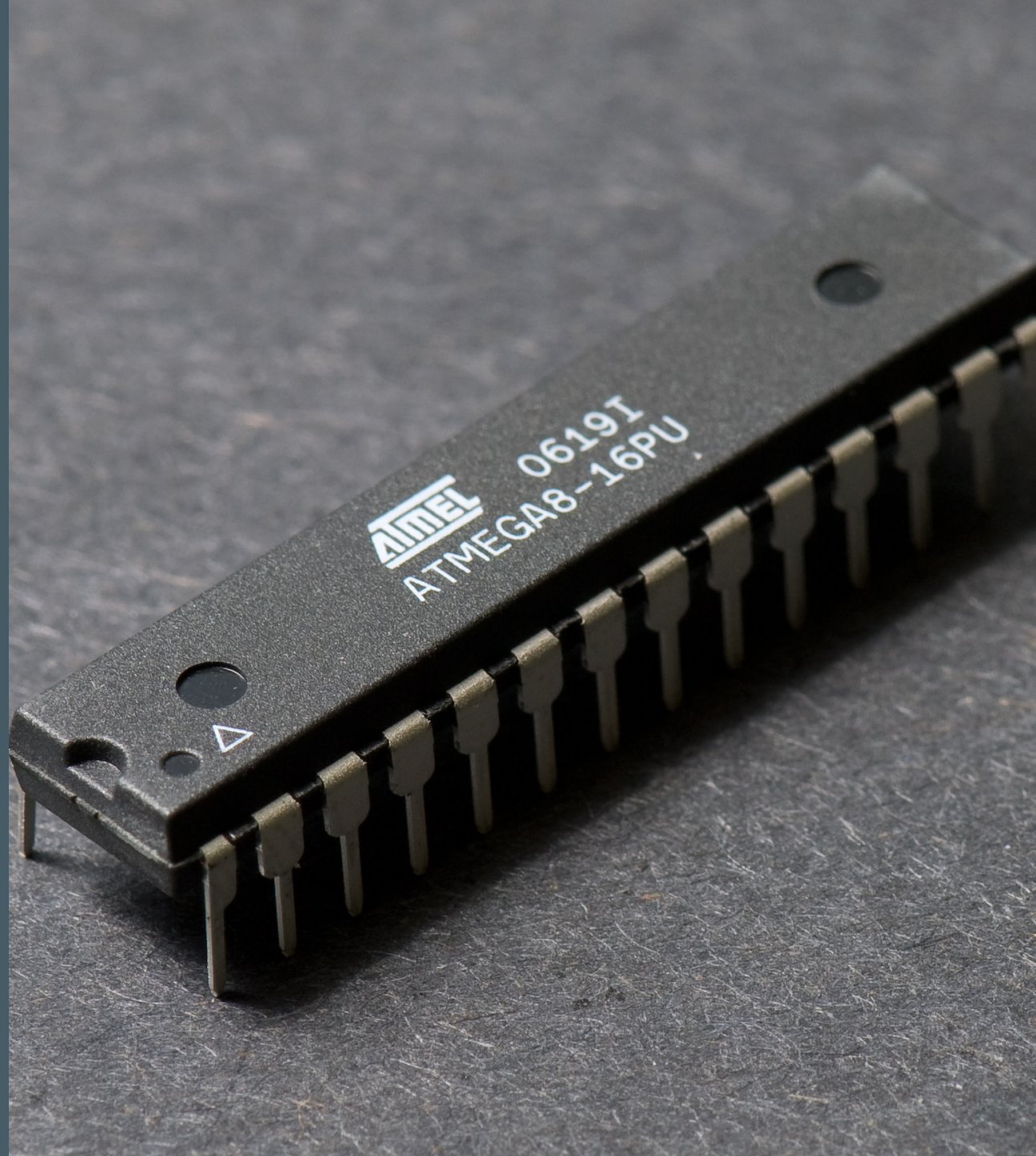
MOS KIM-1

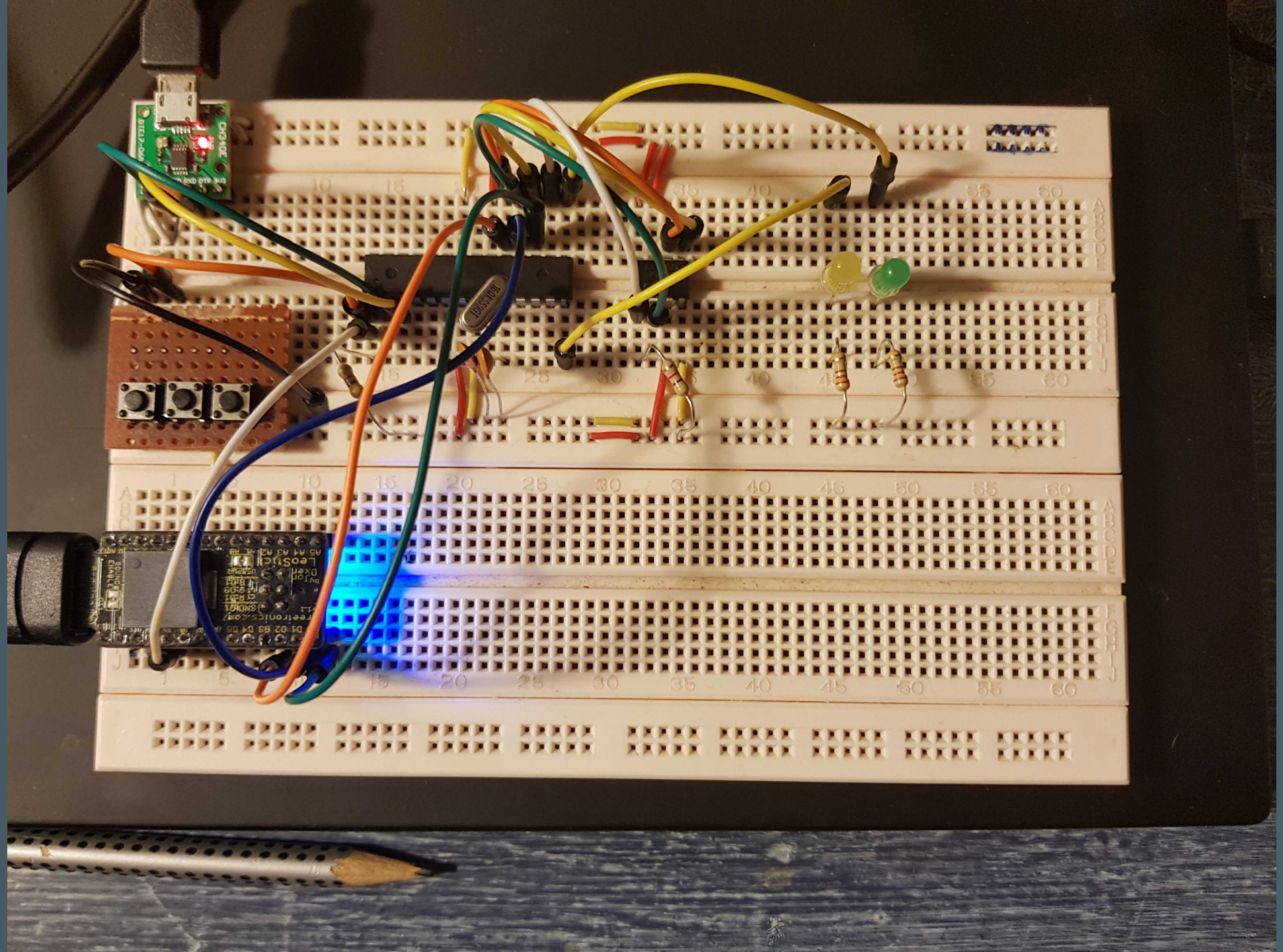
- 6502 CPU
- 8x 6102 SRAM (1024 bytes)
- 2x 6530 RRIOT
  - ROM (1024 bytes)
  - SRAM (64 bytes)
  - 16 IO pins
  - programmable interval timer
- 6x 7-segment LED display
- Keypad

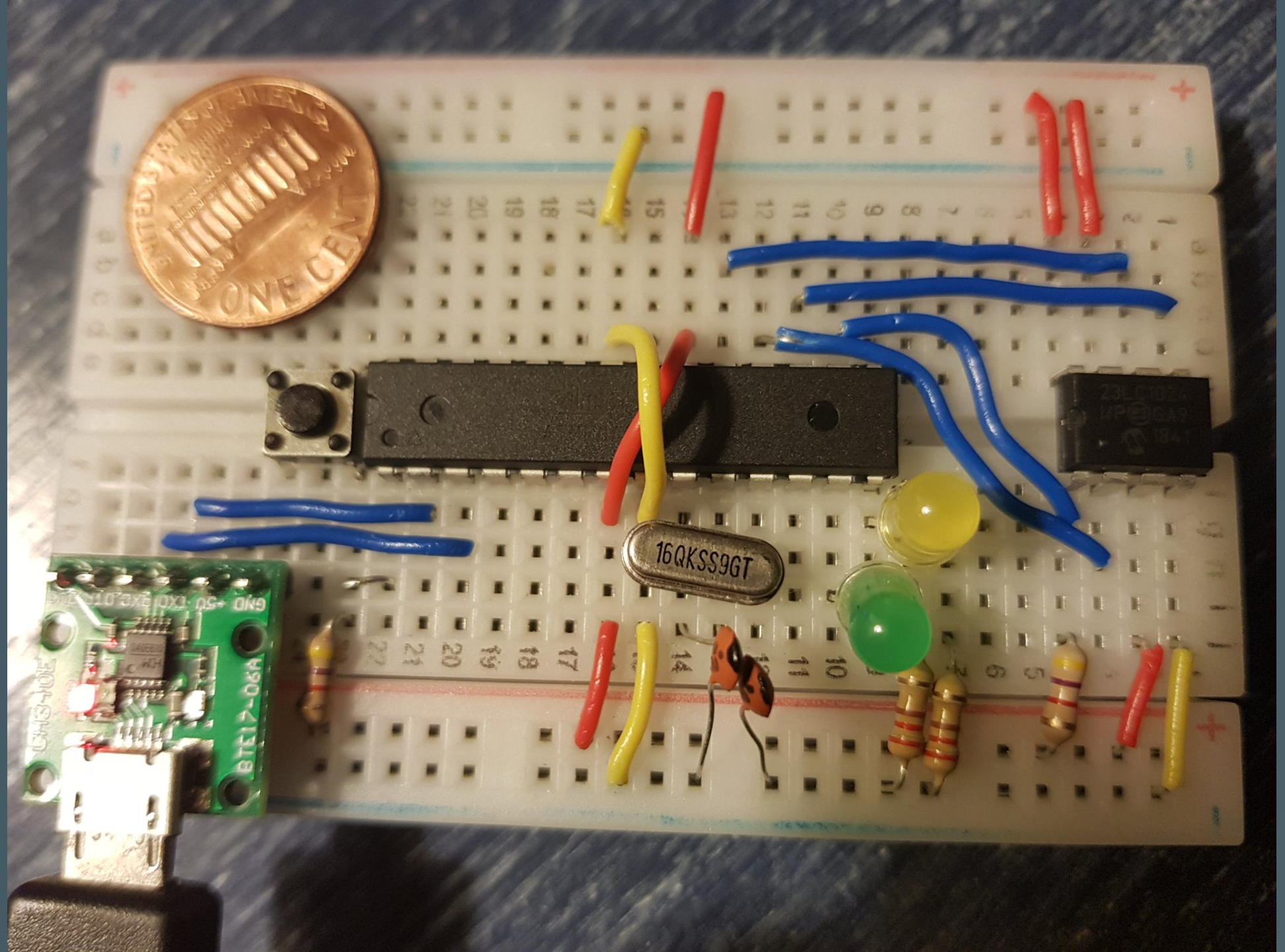


Microchip ATmega8

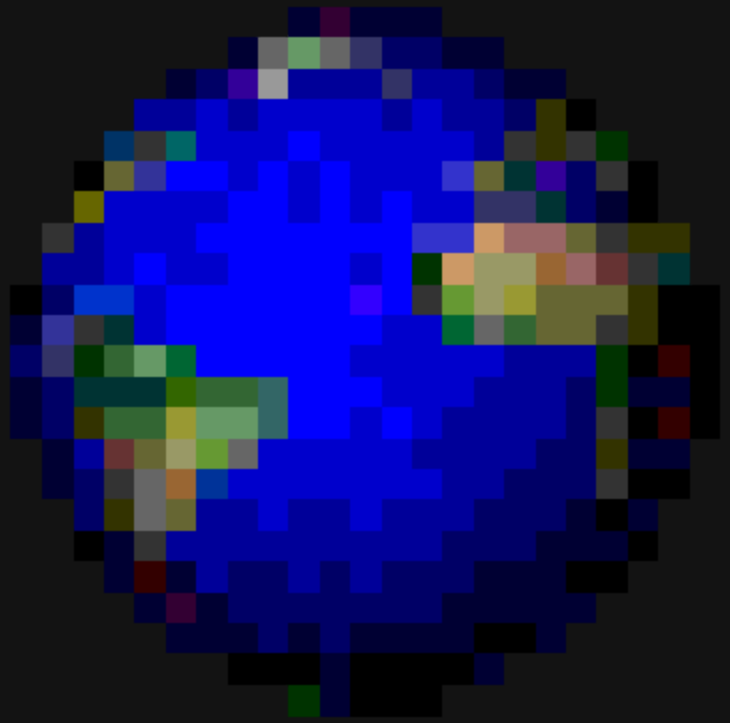
- megaAVR CPU core
- 8KB Flash program memory
- 1KB SRAM
- Multiple peripheral devices
  - SPI/I2C/2wire/1wire buses
  - USART (serial port)
  - Various A/D & D/A converters
  - Various timers






ONE CENT
16QKSS9GT
GND +5V TXD RXD
BTE12-06A

HELLO



WORLD

```
.include "m8def.inc"

.cseg
.org 0x0000

  rjmp reset       ; any reset source
  reti             ; external interrupt request 0
  reti             ; external interrupt request 1
  reti             ; timer/counter2 compare match
  reti             ; timer/counter2 overflow
  reti             ; timer/counter1 capture event
  reti             ; timer/counter1 compare match A
  reti             ; timer/counter1 compare match B
  reti             ; timer/counter1 overflow
  reti             ; timer/counter0 overflow
  reti             ; serial transfer complete
  reti             ; USART Rx complete
  reti             ; USART data register empty
  reti             ; USART Tx complete
  reti             ; ADC conversion complete
  reti             ; EEPROM ready
  reti             ; analog comparator
  reti             ; two-wire serial interface
  reti             ; store program memory ready
```

```
reset:
  ldi r16, low(RAMEND)
  ldi r17, high(RAMEND)
  out SPL, r16
  out SPH, r17
```

```
ldi r16, 1<<PB0
out DDRB, r16
```

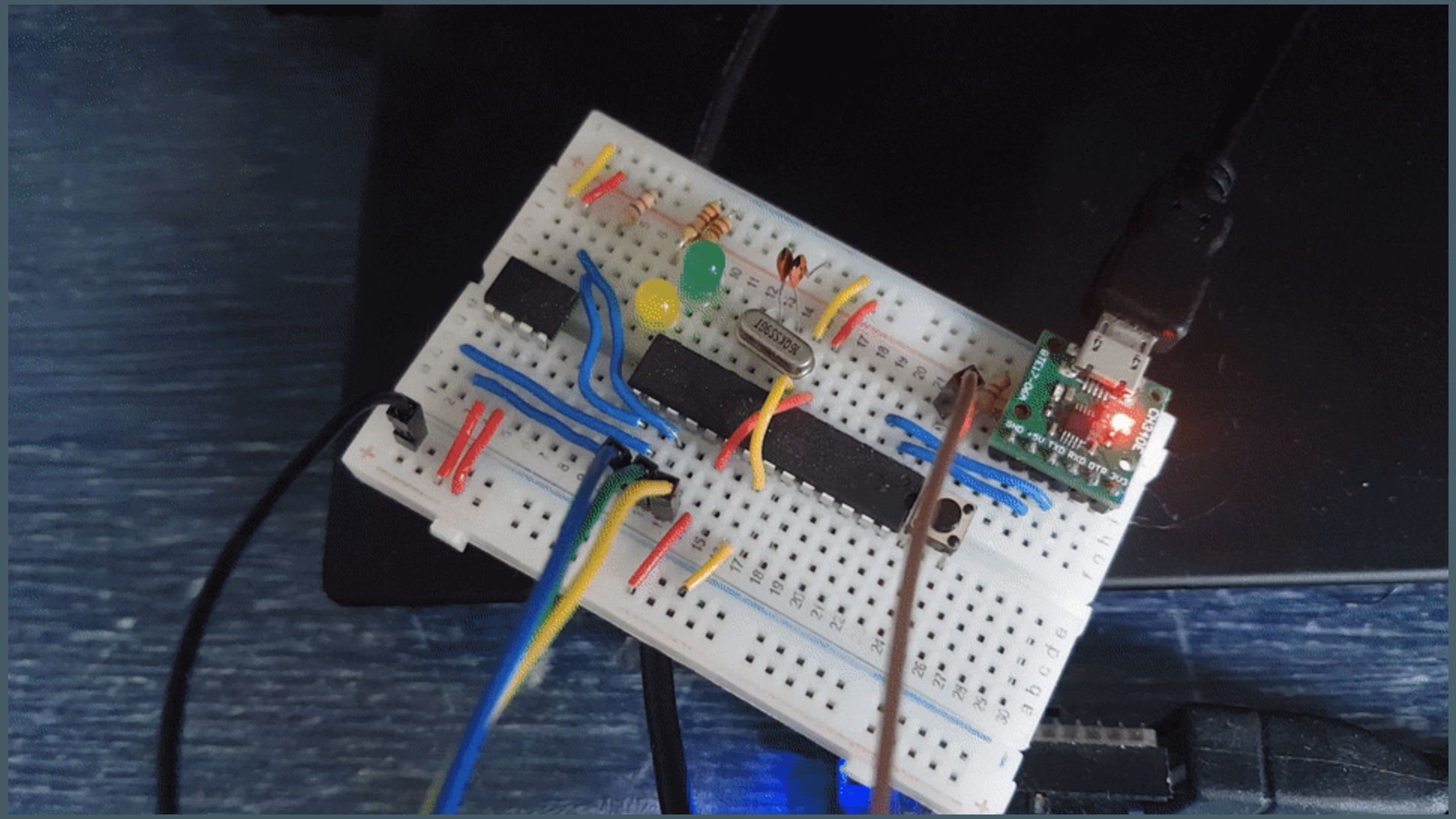
```
loop:
  sbi PORTB, PB0
  rcall wait
  cbi PORTB, PB0
  rcall wait
  rjmp loop
```

```
wait:
  ldi r16, 30
  clr r17
  clr r18
  dec r18
  brne PC-1
  dec r17
  brne PC-3
  dec r16
  brne PC-5
  ret
```

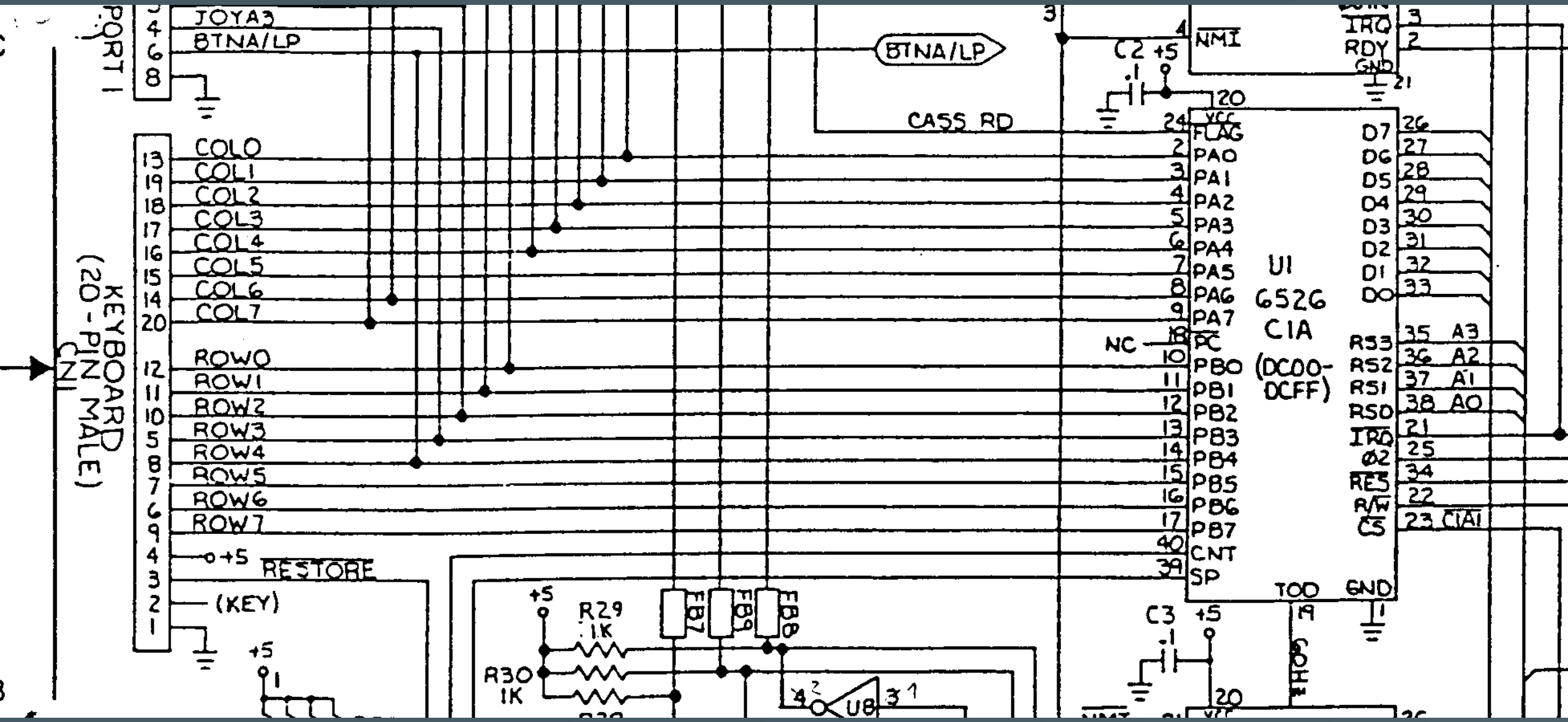
```
wait:
  ldi r16, 30      ;       uint8_t r16 = 30;
  clr r17          ;       uint8_t r17 = 0;
  clr r18          ;       uint8_t r18 = 0;
  dec r18          ; loop: r18--;
  brne PC-1        ;       if (r18 != 0) goto loop;
  dec r17          ;       r17--;
  brne PC-3        ;       if (r17 != 0) goto loop;
  dec r16          ;       r16--;
  brne PC-5        ;       if (r16 != 0) goto loop;
  ret
```

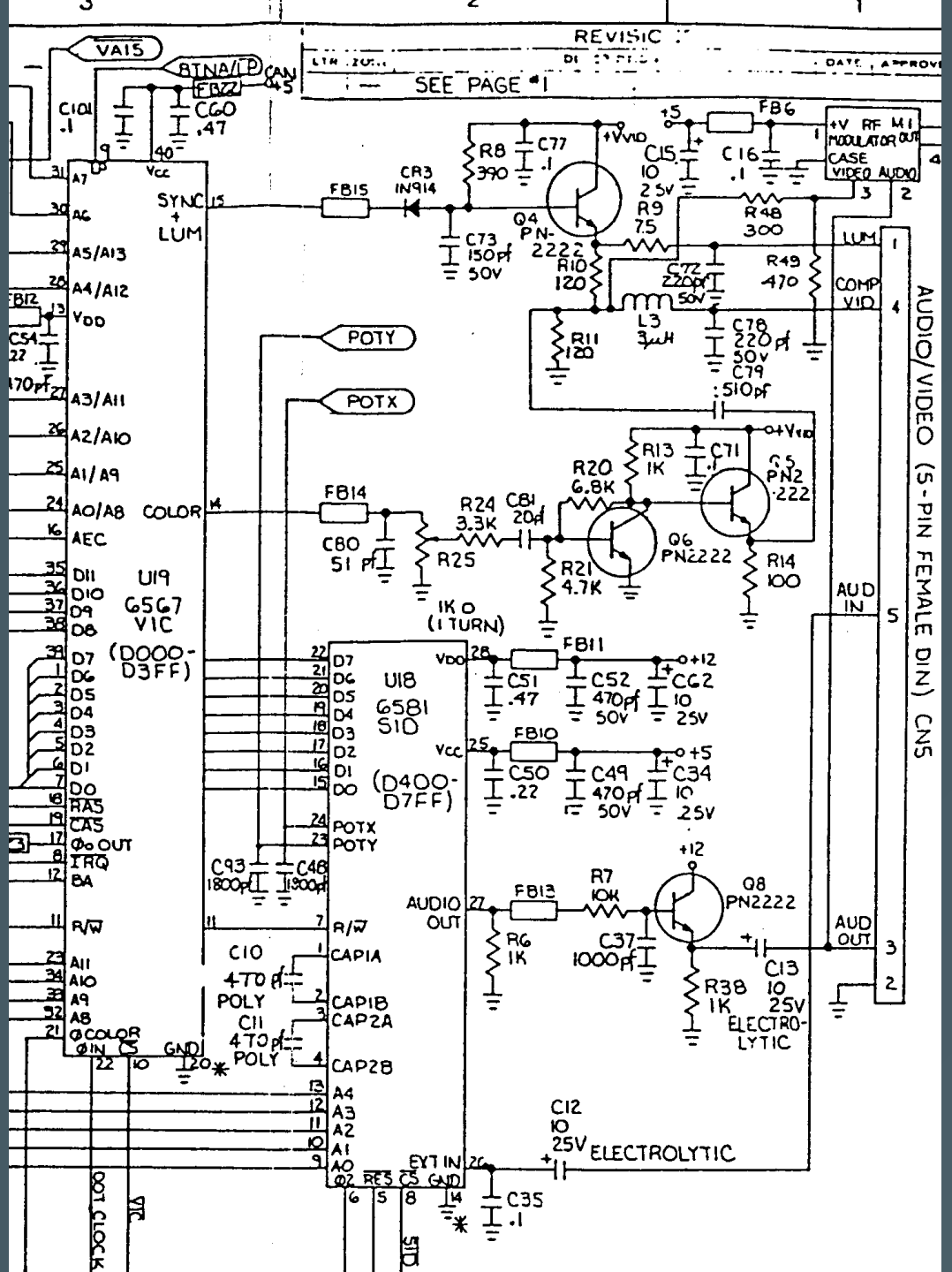
```
$ avra blink.asm
Pass 1...
Pass 2...
done


Assembly complete with no errors.
Segment usage:
   Code      :         51 words (102 bytes)
   Data      :          0 bytes
   EEPROM    :          0 bytes
```

```
$ avrdude -v -c arduino -P /dev/ttyACM0 -p m88p -U flash:w:blink.hex:i
...
avrdude: AVR device initialized and ready to accept instructions

Reading | ################################################## | 100% 0.00s

avrdude: Device signature = 0x1e930f (probably m88p)
avrdude: safemode: lfuse reads as FF
avrdude: safemode: hfuse reads as DF
avrdude: safemode: efuse reads as F9
avrdude: NOTE: "flash" memory has been specified, an erase cycle will be performed
         To disable this feature, specify the -D option.
avrdude: erasing chip
avrdude: reading input file "blink.hex"
avrdude: writing flash (102 bytes):

Writing | ################################################## | 100% 0.17s

avrdude: 102 bytes of flash written
avrdude: verifying flash memory against blink.hex:
avrdude: load data flash data from input file blink.hex:
avrdude: input file blink.hex contains 102 bytes
avrdude: reading on-chip flash data:

Reading | ################################################## | 100% 0.11s

avrdude: verifying ...
avrdude: 102 bytes of flash verified

avrdude: safemode: lfuse reads as FF
avrdude: safemode: hfuse reads as DF
avrdude: safemode: efuse reads as F9
avrdude: safemode: Fuses OK (E:F9, H:DF, L:FF)

avrdude done.  Thank you.
```



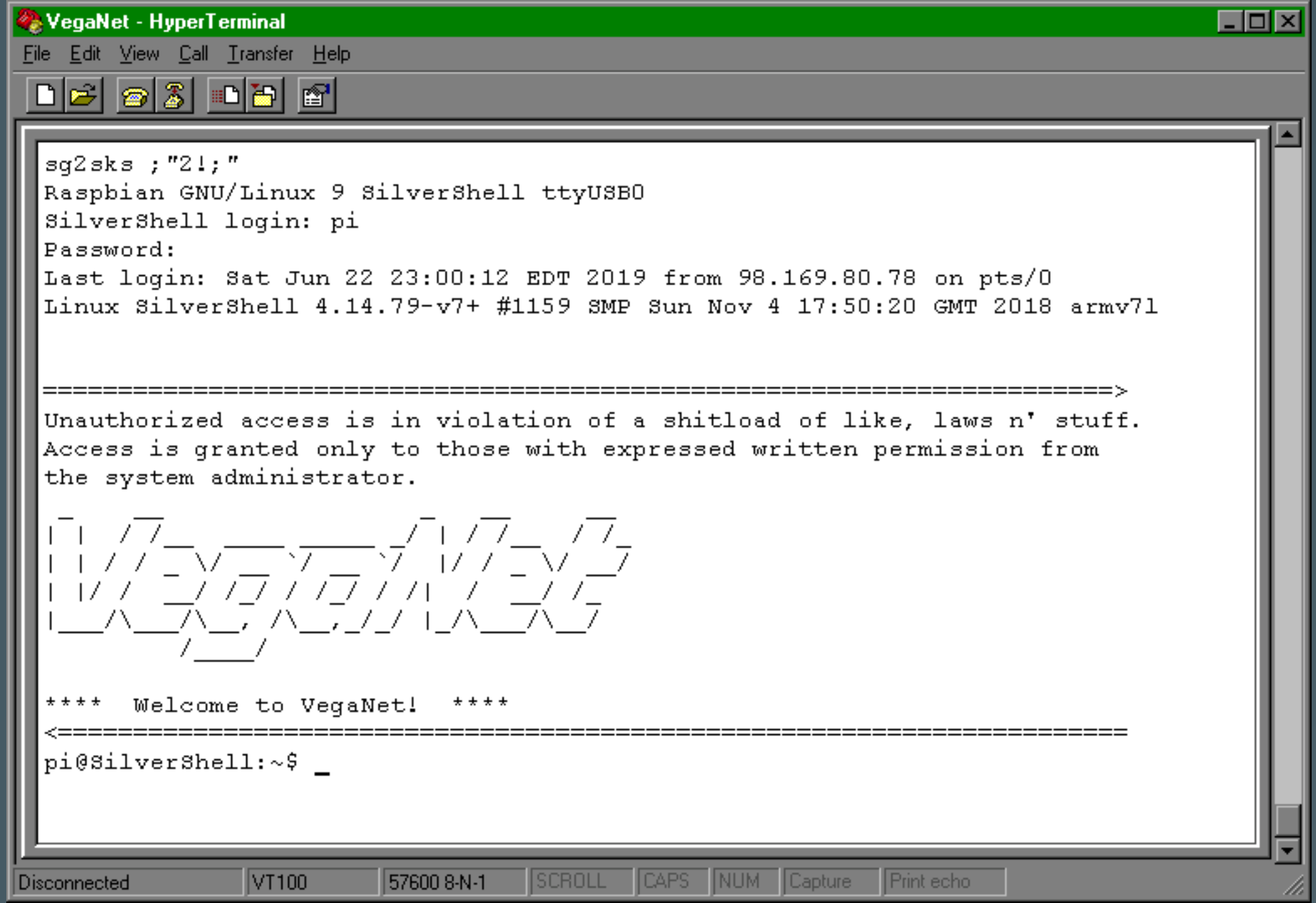

Telecommunications

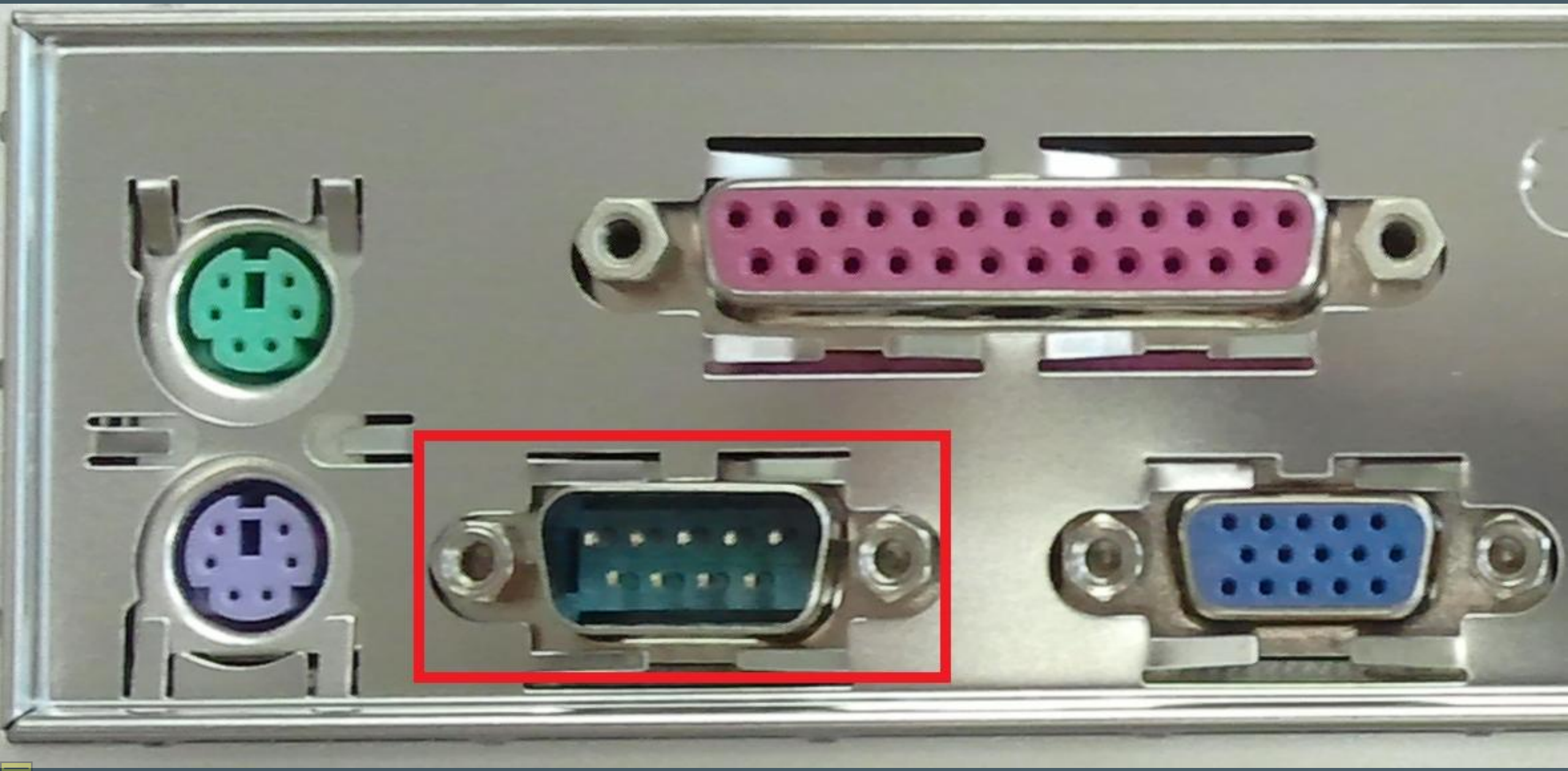

PORT 1
JOYA3
BTNA/LP
BTNA/LP
CASS RD
NMI
IRQ
RDY
GND
C2 +5
.1
KEYBOARD (20-PIN MALE) CN1
COL0
COL1
COL2
COL3
COL4
COL5
COL6
COL7
ROW0
ROW1
ROW2
ROW3
ROW4
ROW5
ROW6
ROW7
+5
RESTORE
(KEY)
U1
6526
CIA
(DC00-DCFF)
FLAG
PA0
PA1
PA2
PA3
PA4
PA5
PA6
PA7
PC
PB0
PB1
PB2
PB3
PB4
PB5
PB6
PB7
CNT
SP
D7
D6
D5
D4
D3
D2
D1
D0
RS3
RS2
RS1
RS0
IRQ
Ø2
RES
R/W
CS
CIAI
A3
A2
A1
A0
VCC
TOD
GND
C3
R29
1K
R30
1K
U8


REVISIC
SEE PAGE #1
VA15
C101 .1
C60 .47
U19 6567 VIC (D000-D3FF)
U18 6581 S10 (D400-D7FF)
SYNC + LUM
COLOR
POTY
POTX
R8 390
C77 .1
CR3 IN914
FB15
C73 150pf 50V
Q4 PN-2222
R9 7.5
R10 120
R11 120
L3 3uH
C72 220pf 50V
C78 220pf 50V
C79 :510pf
R49 470
R48 300
C15 10 25V
C16 .1
FB6
+V RF MODULATOR CASE
VIDEO
AUDIO
LUM
COMP VID
AUD IN
AUD OUT
AUDIO/VIDEO (5-PIN FEMALE DIN) CN5
R13 1K
C71 .1
R20 6.8K
Q5 PN2222
FB14
C80 51 pf
R24 3.3K
C81 20pf
R25 1K Ω (1 TURN)
Q6 PN2222
R21 4.7K
R14 100
FB11
C51 .47
C52 470pf 50V
CG2 10 25V
FB10
C50 .22
C49 470pf 50V
C34 10 25V
C93 1800pf
C48 1800pf
C10 470 pf POLY
C11 470 pf POLY
FB13
R7 10K
Q8 PN2222
R6 1K
C37 1000pf
R38 1K
C13 10 25V ELECTROLYTIC
C12 10 25V ELECTROLYTIC
C35 .1
AUDIO OUT
EXT IN
+12
+5

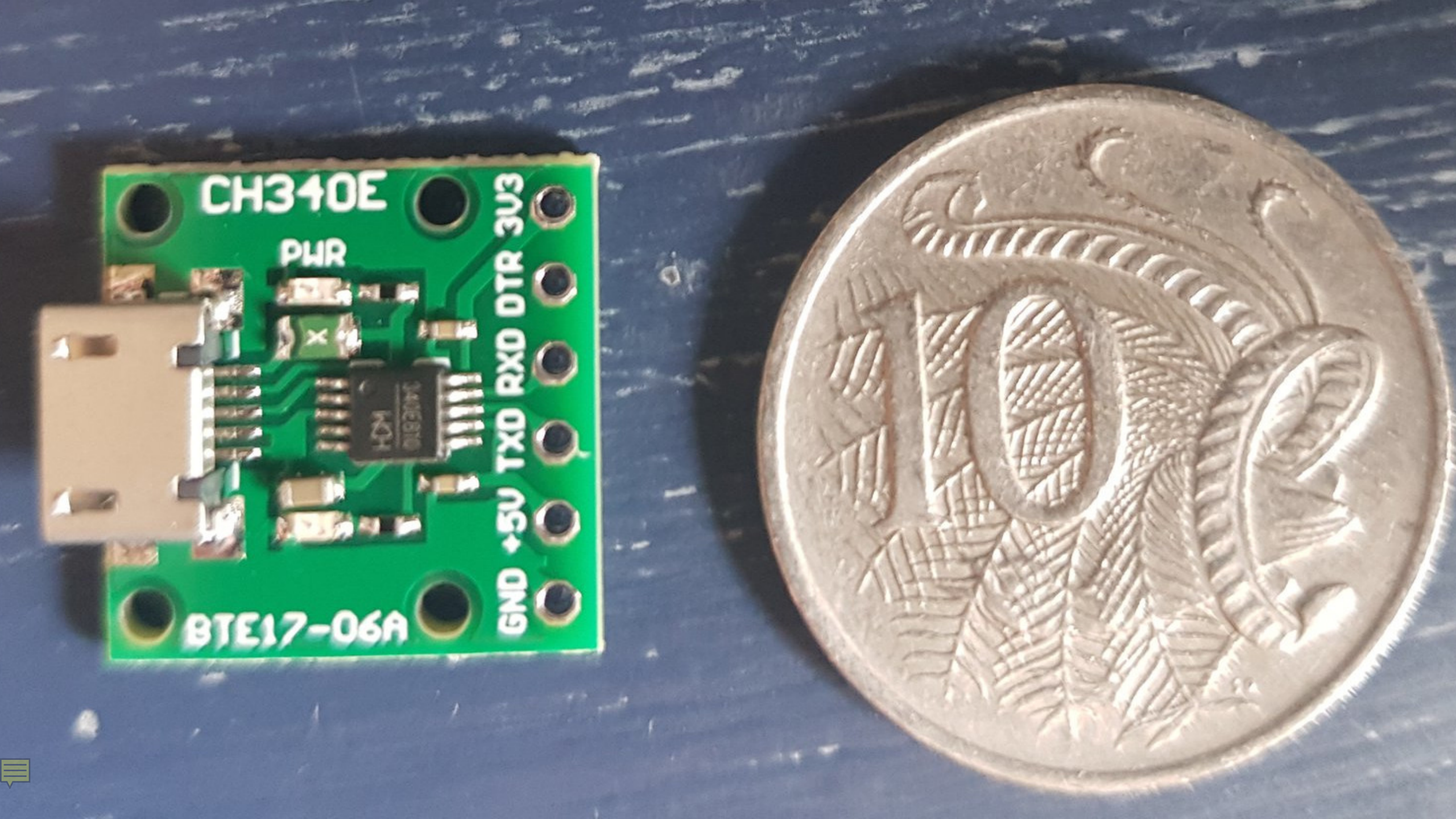

5
Litton
Westrex
501

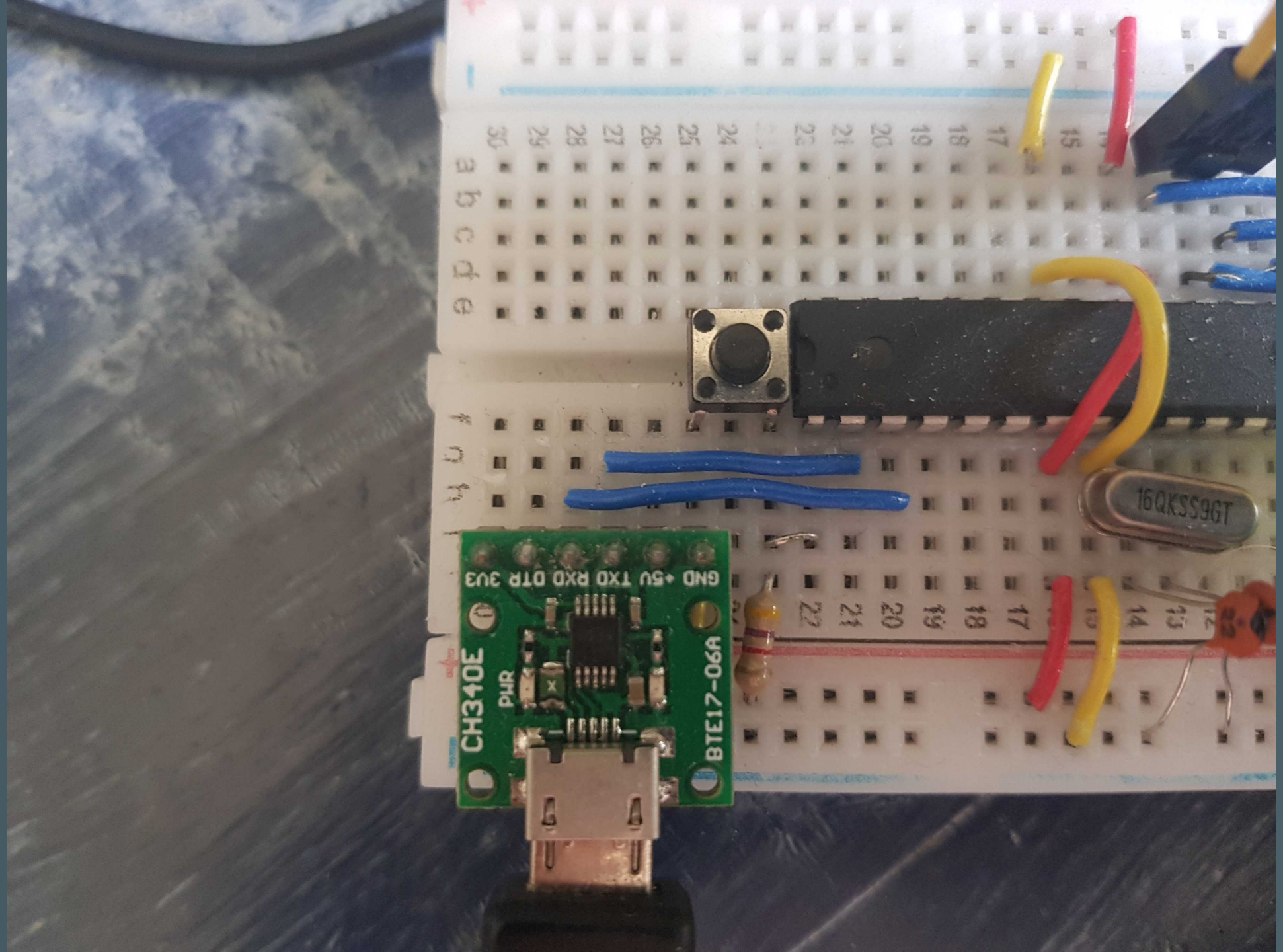

SLMIN .SYS 12P 20-Dec-85
XL .SYS 4P 20-Dec-85
SP .SYS 6P 13-Aug-86
RT11SJ.SYS 78P 13-Aug-86
NL .SYS 2P 13-Aug-86
SQ .SYS 5P 31-May-85
BASIC .SAV 56 24-May-79
DATIME.SAV 4 20-Dec-85
DUMP .SAV 9 20-Dec-85
TSXMOD.SAV 78 27-Nov-92
HARRIS.SAV 41 12-Jun-85
START .P36 2 21-Dec-91
EM1A .STM 19P 02-Feb-93
STAND .LIN 12P 15-Aug-83
EM1B .STM 19 11-Feb-93
JKFLIP.FOR 3 08-Mar-96
ANNOT .SAV 38 18-Apr-93
DUO .DIR 18 16-Jul-96
DIR .DIF 26 16-Jul-96
DEMOBC.OBJ 1 -BAD-
114 Files, 5949 Blocks
14433 Free blocks
.O
VM .SYS 3P 13-Aug-86
LD .SYS 8P 23-Aug-86
ML .SYS 5P 13-Aug-86
DU .SYS 8P 13-Aug-86
TT .SYS 2P 13-Aug-86
RL02DG.SYS 71P 21-Nov-84
BINCOM.SAV 24 20-Dec-85
DIR .SAV 19 20-Dec-85
DUP .SAV 47 20-Dec-85
FORTRA.SAV 206 21-May-85
LET .SAV 5 20-Dec-85
RETRO .OBJ 1519P 16-May-88
TSXUCL.TSX 22 07-Mar-92
TSXV6 .MSG 1P 04-Sep-95
JKFLIP.SAV 30 08-Mar-96
RT11FB.SYS 93P 20-Dec-85
TSXP23.NEW 1200P 27-Nov-92
NL .DIR 7 16-Jul-96
DEMOFC.OBJ 1 -BAD-
EVAN .ID 1
digital VT100


VegaNet - HyperTerminal
File Edit View Call Transfer Help
sg2sks ;"2!;"
Raspbian GNU/Linux 9 SilverShell ttyUSB0
SilverShell login: pi
Password:
Last login: Sat Jun 22 23:00:12 EDT 2019 from 98.169.80.78 on pts/0
Linux SilverShell 4.14.79-v7+ #1159 SMP Sun Nov 4 17:50:20 GMT 2018 armv7l
Unauthorized access is in violation of a shitload of like, laws n' stuff.
Access is granted only to those with expressed written permission from
the system administrator.
**** Welcome to VegaNet! ****
pi@SilverShell:~$ _
Disconnected
VT100
57600 8-N-1
SCROLL
CAPS
NUM
Capture
Print echo




G4409
USB3.0
DVI
HDMI
USB3.0
c/sub
in
rear
out
SPDIF OUT
mic


CH340E
PWR
GND +5V TXD RXD DTR 3V3
BTE17-06A


CH340E
PWR
BTE17-06A
GND +5V TXD RXD OTR 3V3
16QKSS9GT

```
; usart tx/rx enable
ldi r16, (1<<RXEN0) | (1<<TXEN0)
sts UCSR0B, r16

; usart frame format: 8N1 (8 data bits => UCSZ2:0 = 011,
; no parity => UPM1:0 = 00, 1 stop bit => USBS = 0)
ldi r16, (1<<UCSZ00) | (1<<UCSZ01)
sts UCSR0C, r16

; usart 38400 baud at 16MHz => UBRR = 25
ldi r16, 25
ldi r17, 0
sts UBRR0L, r16
sts UBRR0H, r17
```

```
; receive a byte from the usart
; outputs:
;   r16: received byte
usart_rx_byte:
  lds r16, UCSR0A
  sbrs r16, RXC0
  rjmp PC-3
  lds r16, UDR0
  ret

; transmit a byte via the usart
; inputs:
;   r16: byte to send
usart_tx_byte:
  push r16
  lds r16, UCSR0A
  sbrs r16, UDRE0
  rjmp PC-3
  pop r16
  sts UDR0, r16
  ret
```

```
$ grep ^usart_ basic.asm
usart_rx_byte:
usart_rx_byte_maybe:
usart_tx_byte:
usart_print_static:
usart_print:
usart_line_input:
usart_tx_byte_hex:
usart_tx_bytes_hex:
usart_tx_bytes_hex_next:
usart_tx_bytes_hex_done:
```

```
$ picocom -b 38400 /dev/ttyUSB0
picocom v3.1

port is        : /dev/ttyUSB0
flowcontrol    : none
baudrate is    : 38400
parity is      : none
databits are   : 8
stopbits are   : 1
escape is      : C-a
local echo is  : no
noinit is      : no
noreset is     : no
hangup is      : no
nolock is      : no
send_cmd is    : sz -vv
receive_cmd is : rz -vv -E
imap is        :
omap is        :
emap is        : crcrlf,delbs,
logfile is     : none
initstring     : none
exit_after is  : not set
exit is        : no

Type [C-a] [C-h] to see available commands
Terminal ready

GOOD COMPUTER

BASIC>
```

SO BASIC

SO, BASIC

A very BASIC history

- Dartmouth BASIC (1964)
- Microsoft BASIC (1975)
- Tiny BASIC (1975)
- Integer BASIC (1976)
- BBC BASIC (1981)
- GW BASIC (1983)
- QuickBASIC (1985)
- AMOS BASIC (1990)
- Visual Basic (1991)
- Visual Basic .NET (2001)
- Small Basic (2008)

```
10 PRINT "How many stars do you want?"
20 INPUT N
30 LET S$ = ""
40 FOR I = 1 TO N
50 LET S$ = S$ + "*"
60 NEXT I
70 PRINT S$
```

```
BASIC> RUN
How many stars do you want?
INPUT? 5
*****

BASIC> RUN
How many stars do you want?
INPUT? 10
**********
```

GOOD BASIC (2020)

GOOD BASIC (2020)

Keywords

- Variables: LET, FOR
- Conditionals: IF/THEN
- Flow control: FOR/NEXT, GOTO, GOSUB/RETURN
- Program control: NEW, CLEAR, RUN, END
- IO: PRINT, INPUT
- Hardware: ON, OFF, SLEEP, RESET
- Fancy: XLOAD (XMODEM receiver)

GOOD BASIC (2020)

Numeric expressions

- Integers only (no floating point)
- 16-bit two's-complement (-32768 - 32767)
- Correct order of operations
- Comparators
- Functions:
  - ABS: take absolute value of number
  - RND: random number generator

GOOD BASIC (2020)

String expressions

- Concatenation operator
- Ordered comparators
- Functions:
  - LEFT/RIGHT/MID: take substrings
  - LEN: get length of string
  - INKEY: wait for keypress, return character

GOOD BASIC (2020)

Expressions

- Variable expansion
- Type-checked at parse time

GOOD BASIC (2020)

(it's good!)

Code-writer

Here is a program to make a computer code and decode messages. You can use it to send top-secret information, or write private letters to your friends. Nobody will be able to crack the code without this program and the line changes shown opposite.

Code program

```
10 CLS
20 LET C$="":PRINT
30 PRINT "TYPE IN YOUR MESSAGE"
40 INPUT M$:LET L=LEN(M$)
50 IF INT(L/2)<>L/2 THEN LET M$=M$+" "
60 FOR K=1 TO LEN(M$) STEP 2
70 LET C$=C$+MID$(M$,K+1,1)
80 LET C$=C$+MID$(M$,K,1)
90 NEXT K
100 CLS
110 PRINT:PRINT "CODE MESSAGE:"
120 PRINT:PRINT C$
```

Leave no space inside these quotation marks.

Type one space inside these quotation marks.

How to use the program

When you run the program, the computer asks you to type in your message.

You should type it in and then press RETURN.

The computer translates your message into code and puts it on the screen.

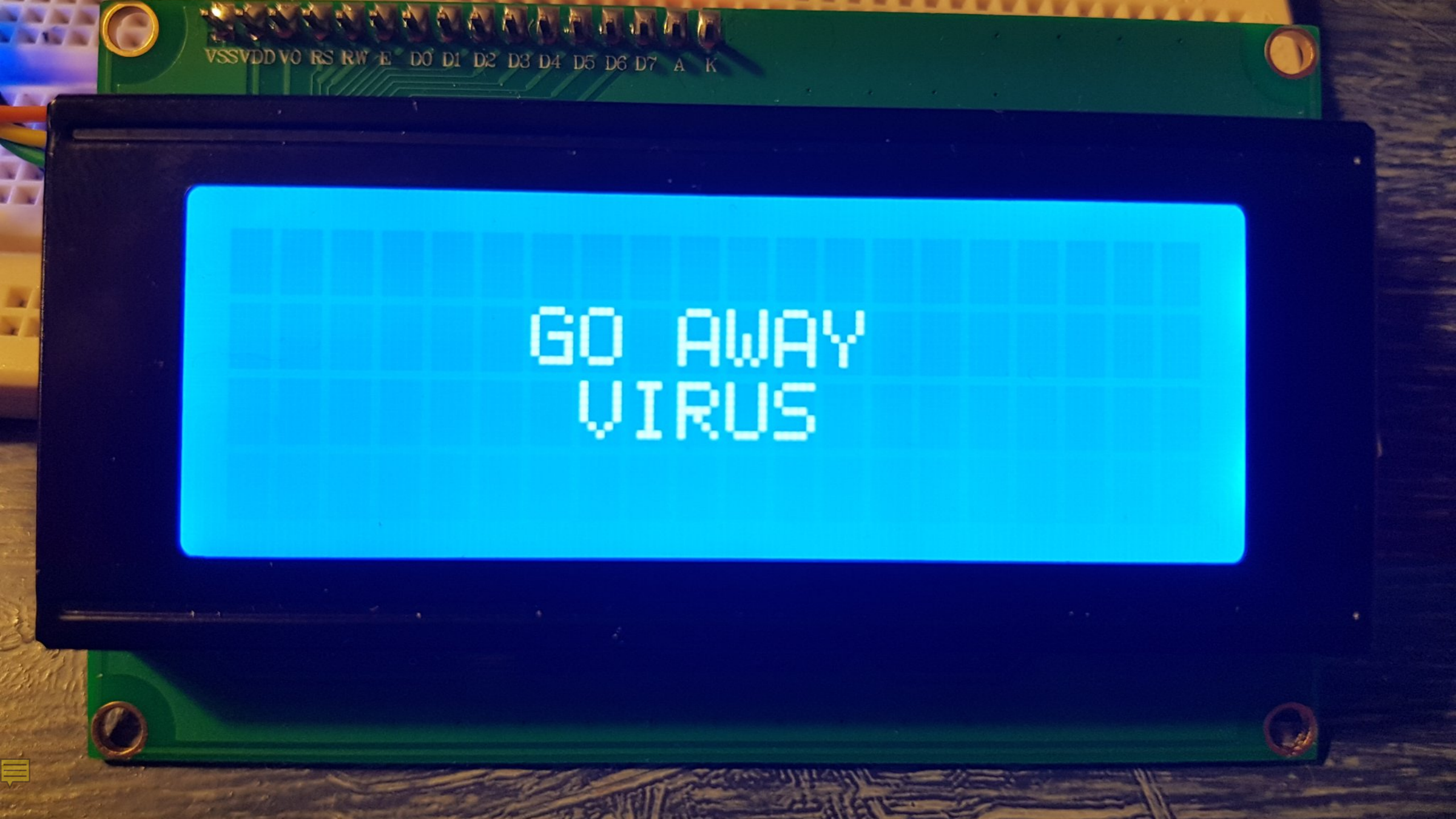
```
GOOD COMPUTER

BASIC> 10 CLS

BASIC> 20 LET C$="":PRINT

BASIC> 30 PRINT "TYPE IN YOUR MESSAGE"

BASIC> 40 INPUT M$:LET L=LEN(M$)

BASIC> 50 IF L/2*2<>L THEN LET M$=M$+" "

BASIC> 60 FOR K=1 TO LEN(M$)

BASIC> 70 LET C$=C$+MID$(M$,K+1,1)

BASIC> 80 LET C$=C$+MID$(M$,K,1)

BASIC> 85 LET K=K+1

BASIC> 90 NEXT K

BASIC> 100 CLS

BASIC> 110 PRINT PRINT "CODE MESSAGE:"

BASIC> 120 PRINT:PRINT C$
```

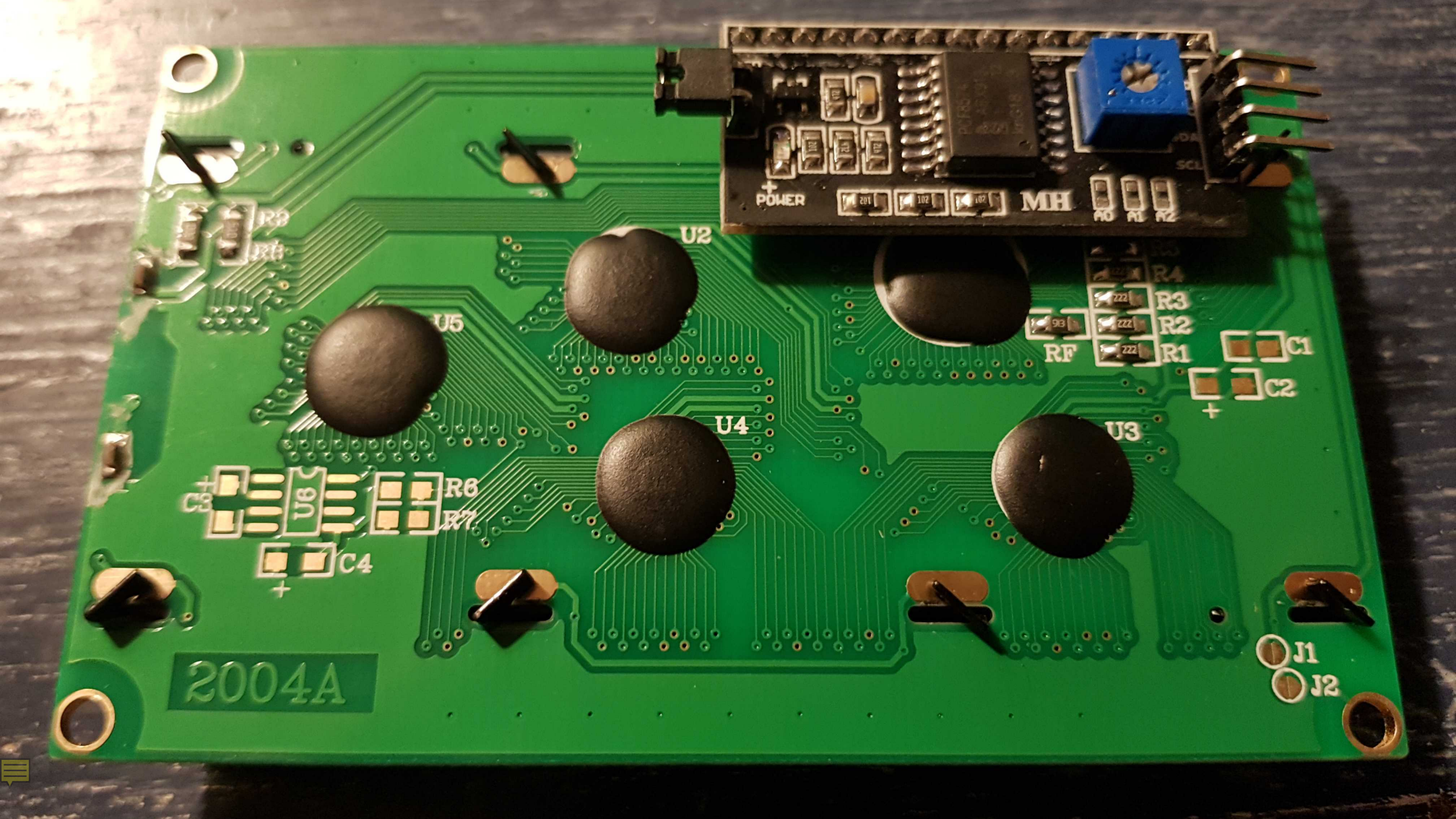
```
TYPE IN YOUR MESSAGE
INPUT? SECRET MEETING WITH AGENT X AT THE HOLLOW TREE TONIGHT
```

```
CODE MESSAGE:

ESRCTEM EEITGNW TI HGANE T XTAT EHH LOOL WRTEET NOGITH
```

Nobody will be able
to crack the code


OFFWORLD LAW
RESTART
BEGIN AGAIN


VSSVDD V0 RS RW E D0 D1 D2 D3 D4 D5 D6 D7 A K
GO AWAY
VIRUS


POWER
MH
A0
A1
A2
SCL
U2
U5
U4
U3
R4
R3
R2
R1
RF
C1
C2
C3
C4
U6
R6
R7
J1
J2
2004A

LCD initialisation

- `0x00` (1000ms)
- `0x30` (4500us)
- `0x30` (4500us)
- `0x30` (4500us)
- `0x20` (50us) (enable 4-bit mode)
- `0x28` (50us) (FUNCTION SET (0x2x), 2-line mode (0xx8))
- `0x0c` (50us) (DISPLAY CONTROL (0x08), display on (0x4))
- `0x01` (2000us) (CLEAR DISPLAY)
- `0x06` (50us) (ENTRY MODESET (0x4), ENTRYLEFT (0x2))
- `0x0a` (2000us) HOME

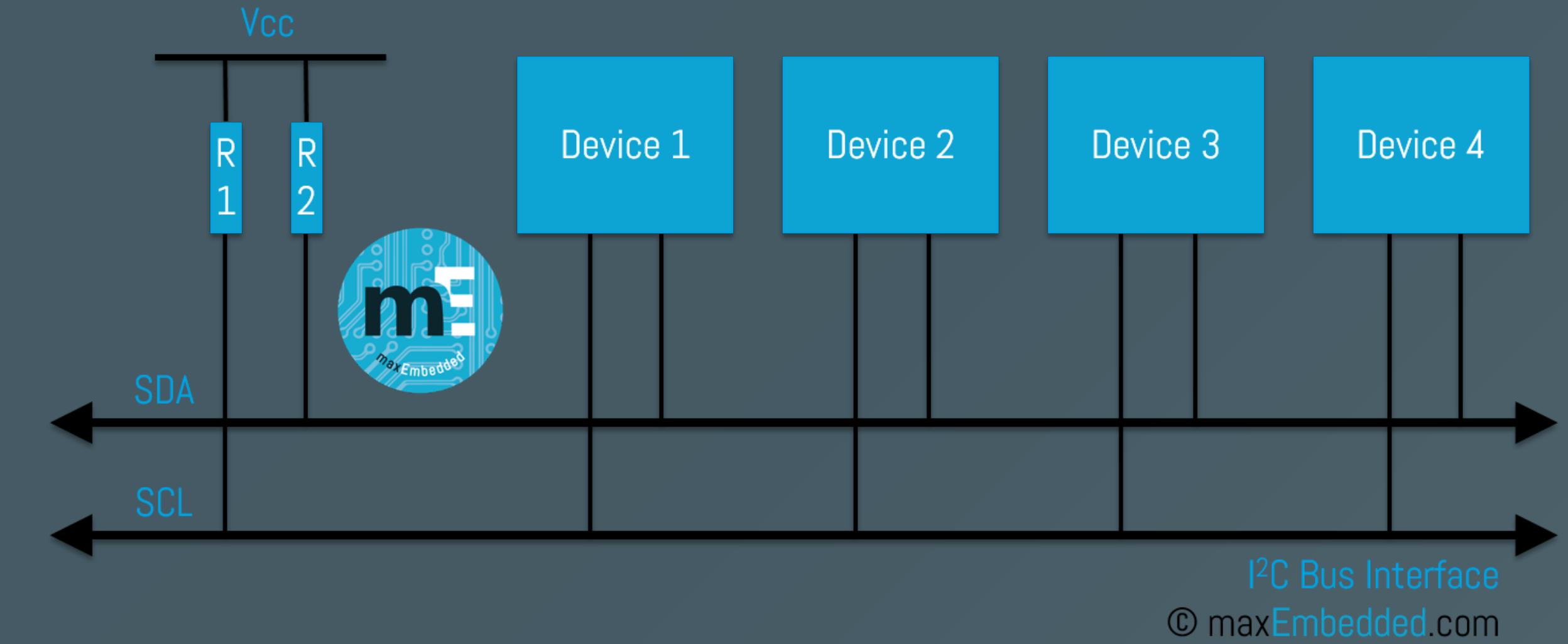
LCD character data

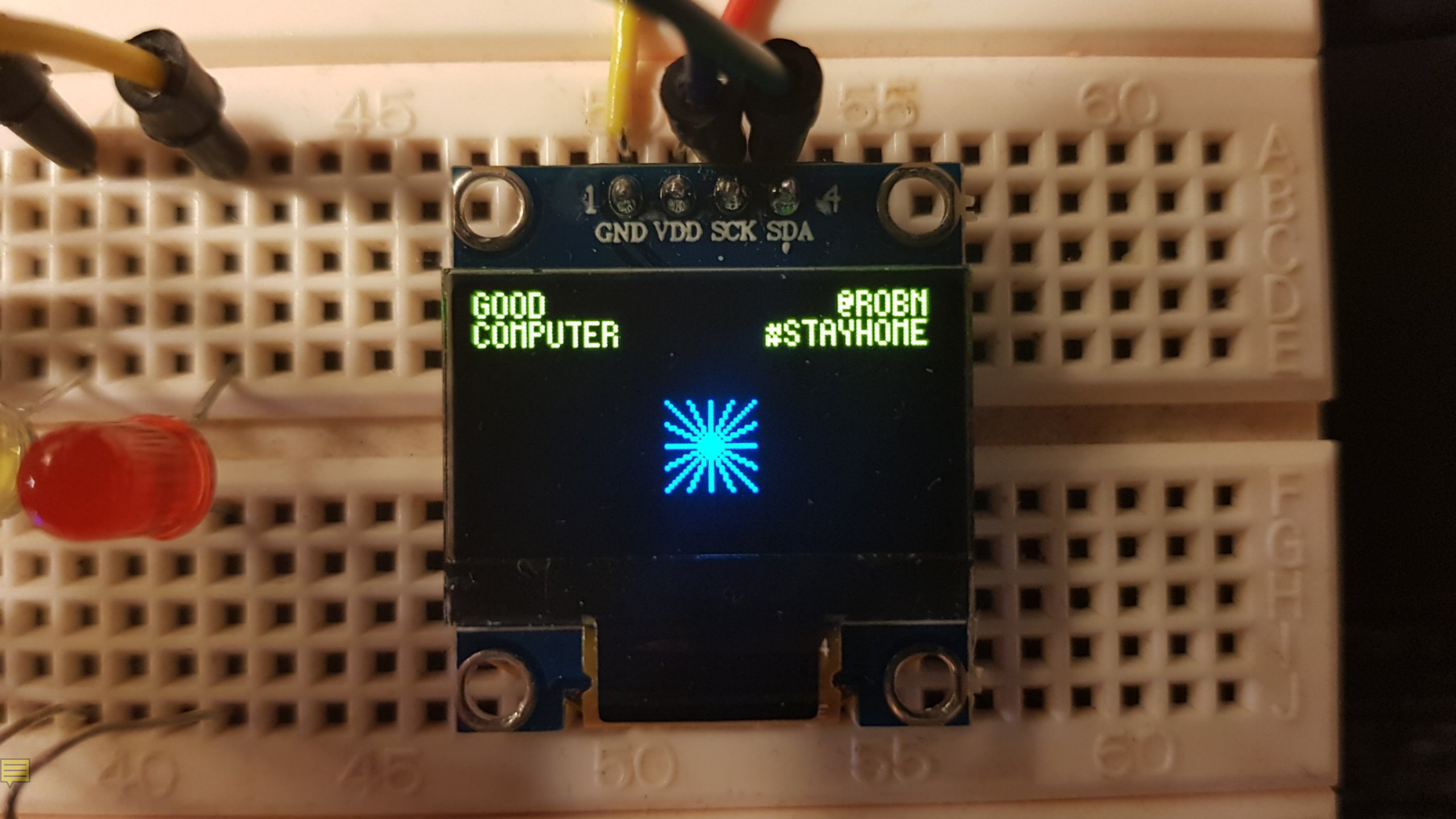
- `0x09` (50us) (select data register)
- `...` (50us) (send ASCII data)

LCD character data

- `0x09` (50us) (select data register)
- `...` (50us) (send ASCII data)

```
AAAAAAAAAAAAAAAAAAAA
CCCCCCCCCCCCCCCCCCCC
BBBBBBBBBBBBBBBBBBBB
DDDDDDDDDDDDDDDDDDDD
```


Vcc
R 1
R 2
Device 1
Device 2
Device 3
Device 4
SDA
SCL
I²C Bus Interface
© maxEmbedded.com


1
4
GND VDD SCK SDA
GOOD
COMPUTER
@ROBN
#STAYHOME

```
0       8       10

0000000000000000...
1111111111111111...
2222222222222222...
3333333333333333...
4444444444444444...
5555555555555555...
6666666666666666...
7777777777777777...

80      88      90

0000000000000000...
1111111111111111...
2222222222222222...
3333333333333333...
4444444444444444...
5555555555555555...
6666666666666666...
7777777777777777...
```

SWIRLY

GRAPHICS

Dots

- Calculate buffer offset for pixel byte
- Read it
- Set the bit within it
- Write the byte back

Dots

- byte row: `(y >> 3) * 128`
- byte column: `x`
- bit: `y & 7`

Dots

- byte row: `(y >> 3) * 128`
- byte column: `x`
- bit: `y & 7`

```
char *p = buffer + ((y >> 3) * 128) + x;
*p |= (1 << (y & 7);
```

Lines

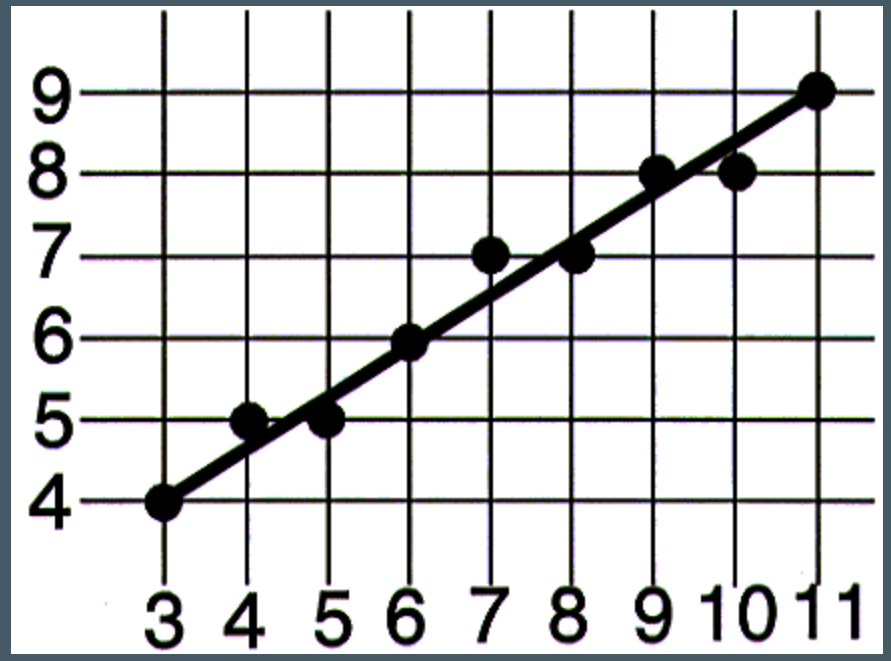
- (x1,y1) - (x2,y2)
- y = mx + c

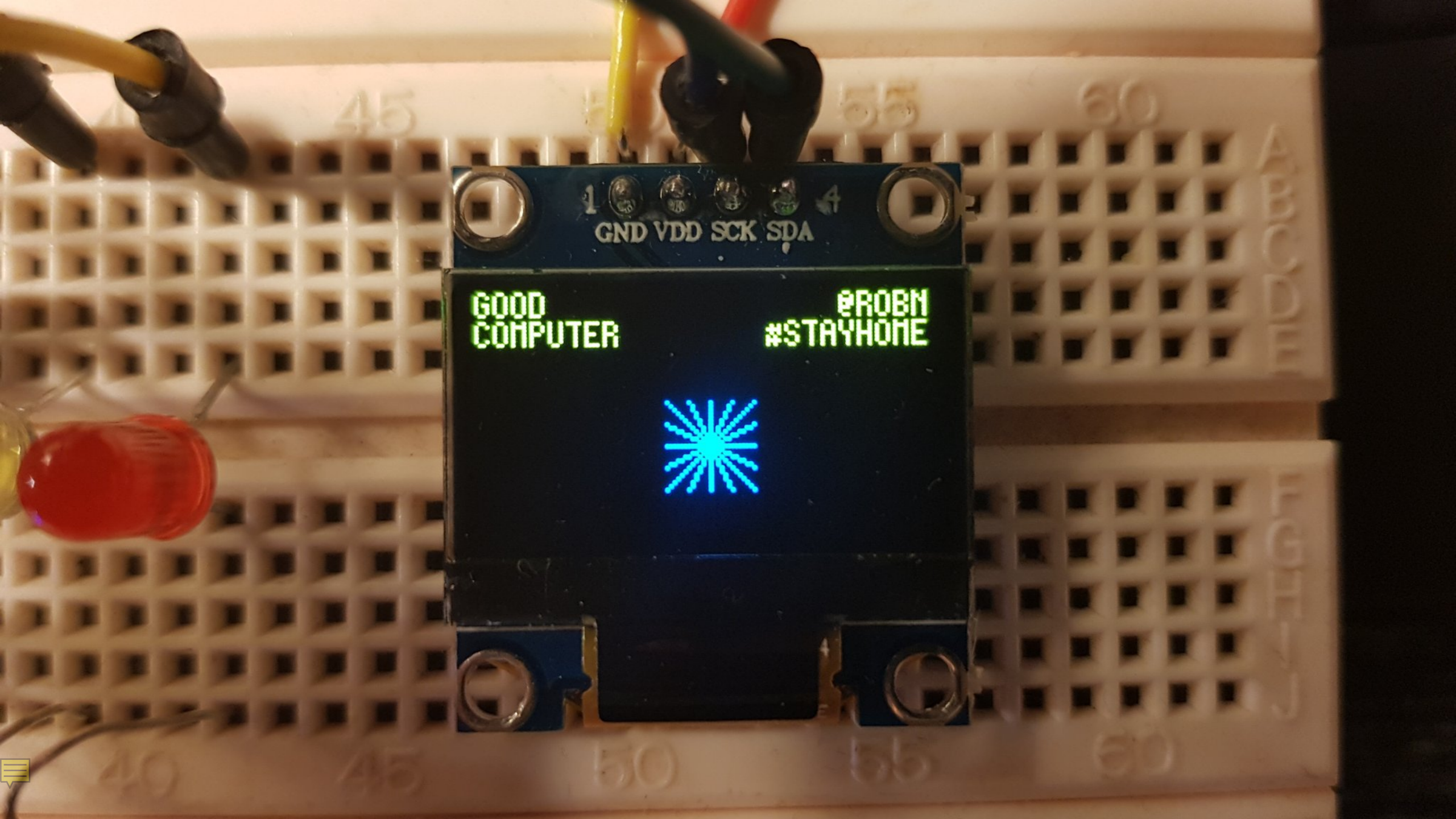
Lines

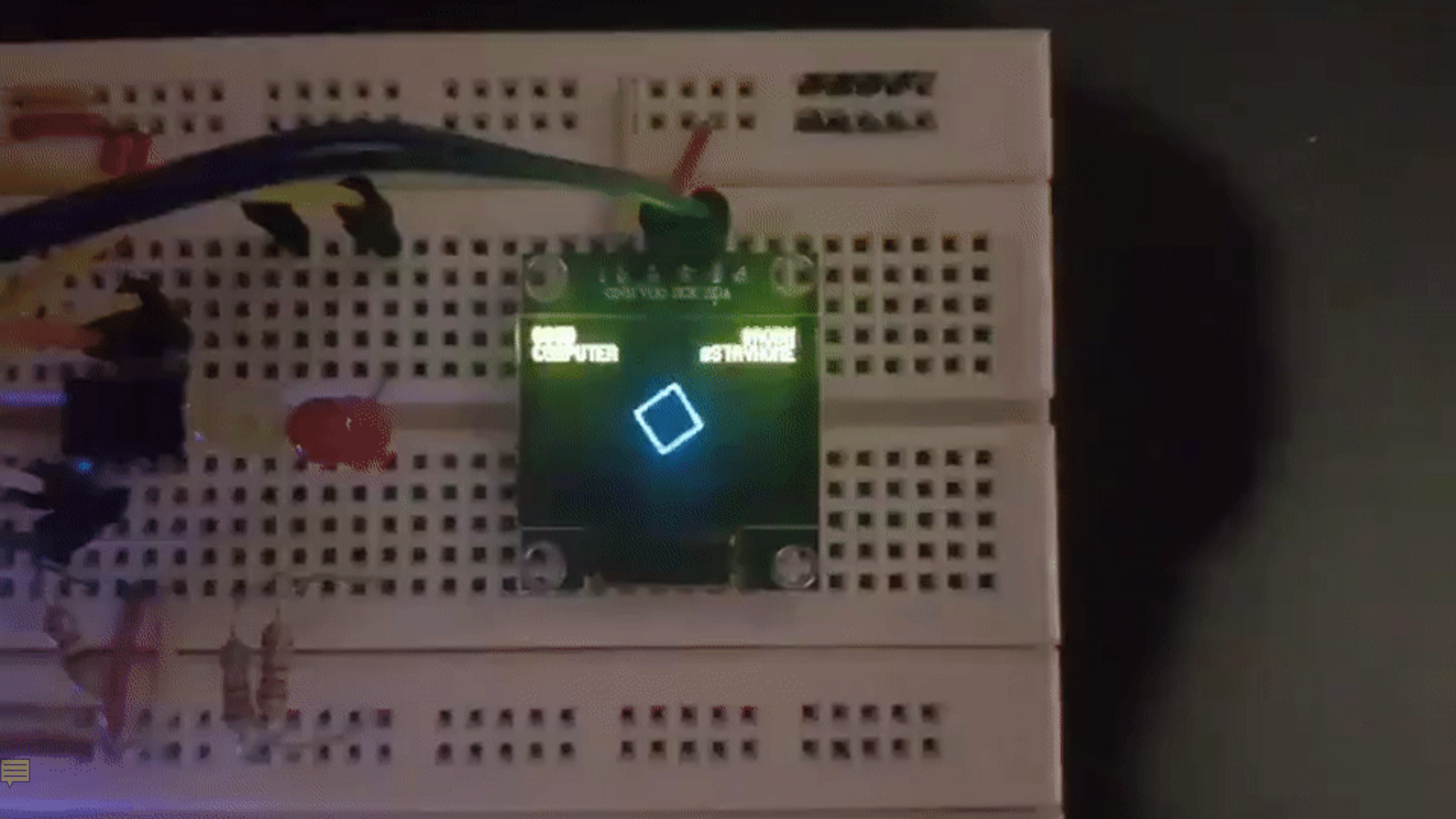
- Bresenham's algorithm

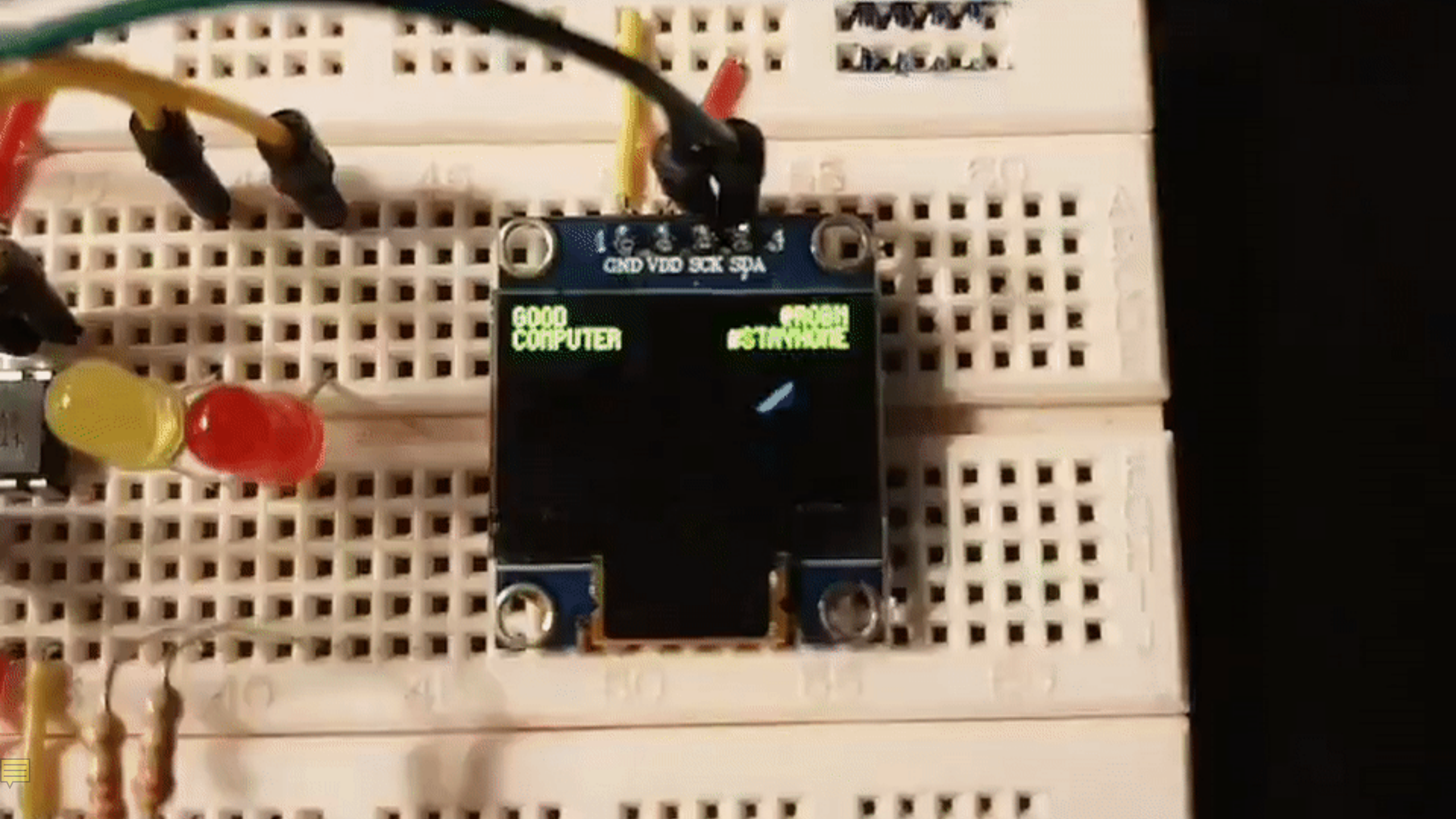
Lines

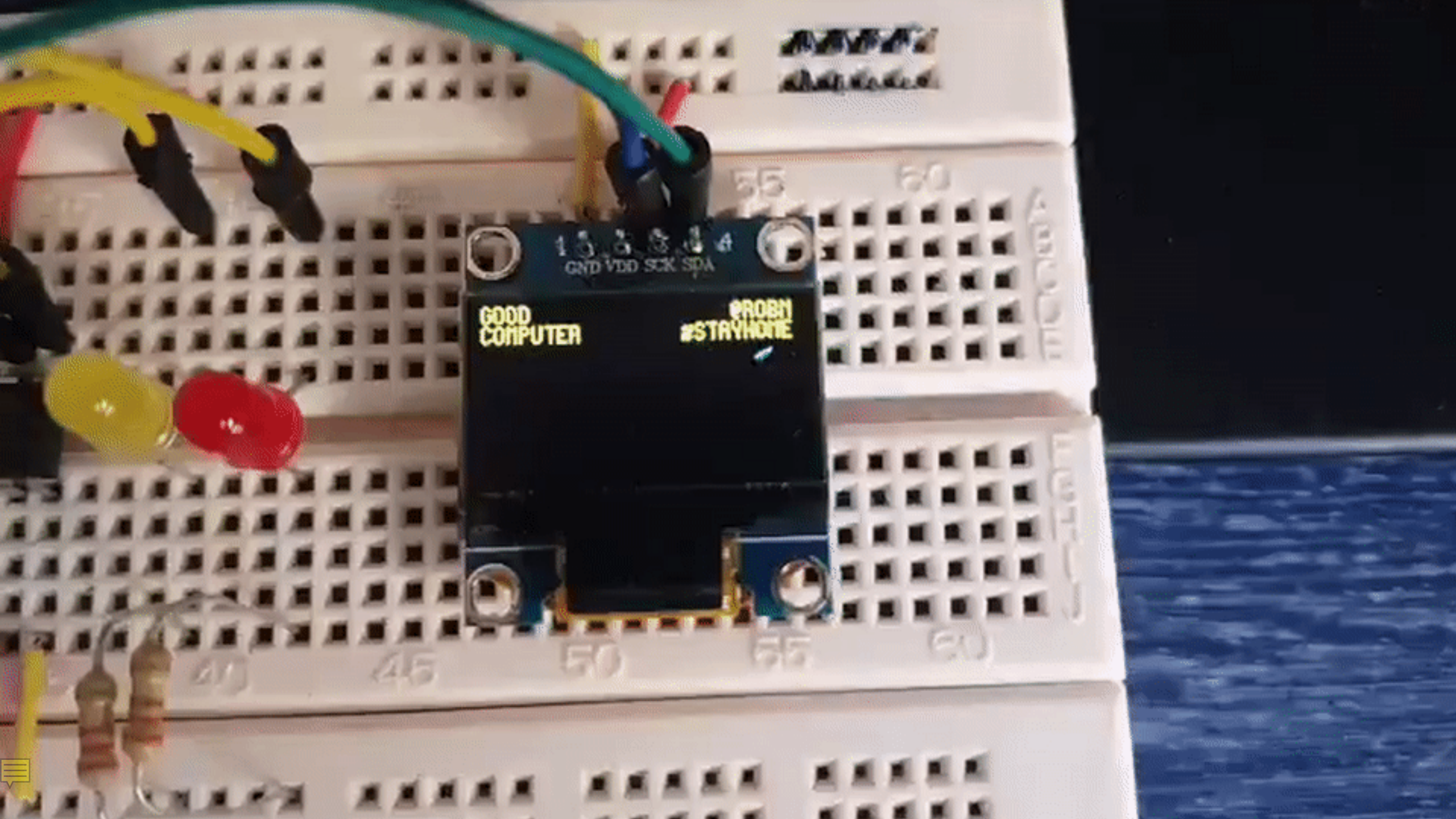
- Bresenham's algorithm




1
4
GND VDD SCK SDA
GOOD
COMPUTER
@ROBN
#STAYHOME


COMPUTER
#STAYHOME


GND VDD SCK SDA
GOOD
COMPUTER


1 2 3 4
GND VDD SCK SDA
GOOD
COMPUTER
PROBN
#STAYHOME

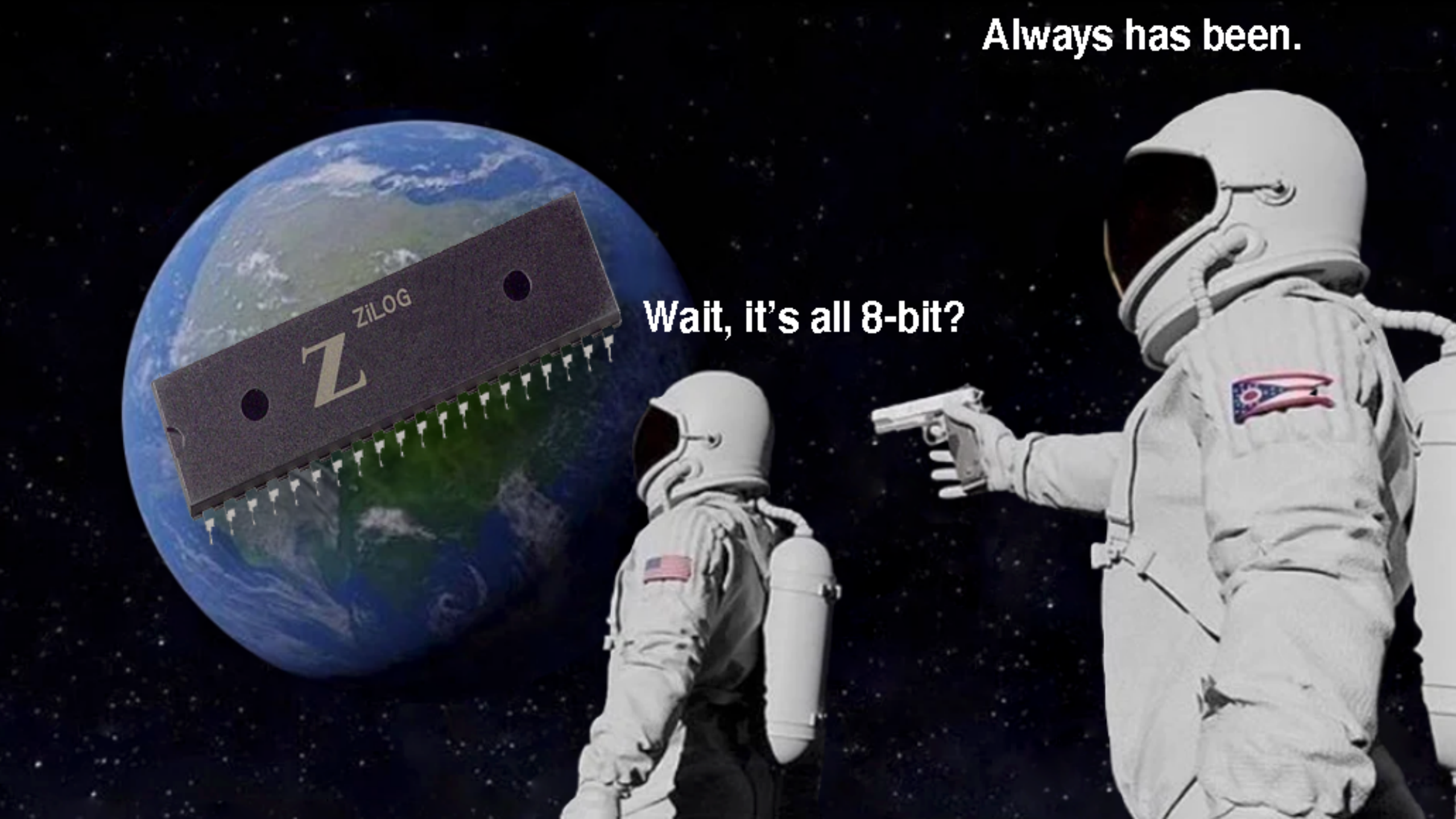
SYSTEM
ACTIVATED

```
ZORK: The Great Underground Empire - Part I
Copyright (c) 1980 by Infocom, Inc. All rights reserved.
ZORK is a trademark of Infocom, Inc.
Release 5 / Serial number

West of House
You are standing in an open field west of a white house, with a boarded front door.
There is a small mailbox here.
>e
The door is boarded and you can't remove the boards.
>n
North of House
You are facing the north side of a white house. There is no door here, and all the windows are boarded up. To the n
orth a narrow path winds through the trees.
>e
Behind House
You are behind the white house. A path leads into the forest to the east. In one corner of the house there is a sma
ll window which is slightly ajar.
>open window
With great effort, you open the window far enough to allow entry.
>w
Kitchen
You are in the kitchen of the white house. A table seems to have been used recently for the preparation of food. A
passage leads to the west and a dark staircase can be seen leading upward. A dark chimney leads down and to the eas
t is a small window which is open.
On the table is an elongated brown sack, smelling of hot peppers.
A bottle is sitting on the table.
The glass bottle contains:
  A quantity of water
>take bottle
Taken.
>i
You are carrying:
  A glass bottle
  The glass bottle contains:
    A quantity of water
>drink water
I'd like to, but I can't get to it.
>open bottle
Opened.
>drink water
Thank you very much. I was rather thirsty (from all this talking, probably).


unimplemented!
     PC: 75d2
 opcode: a9
argtype: bf
  arg 0: 003b


[zap] (r)un (l)oad:
```

VCC
CD
GND
8GB
San


RST
seeed
IIC/I2C
UART
ANALOG
DIGITAL
2
3
4
5
6
7
8
9
10
3V3


@arturo182
UART Demo
Hello World
sym
alt


Always has been.
ZiLOG
Wait, it's all 8-bit?