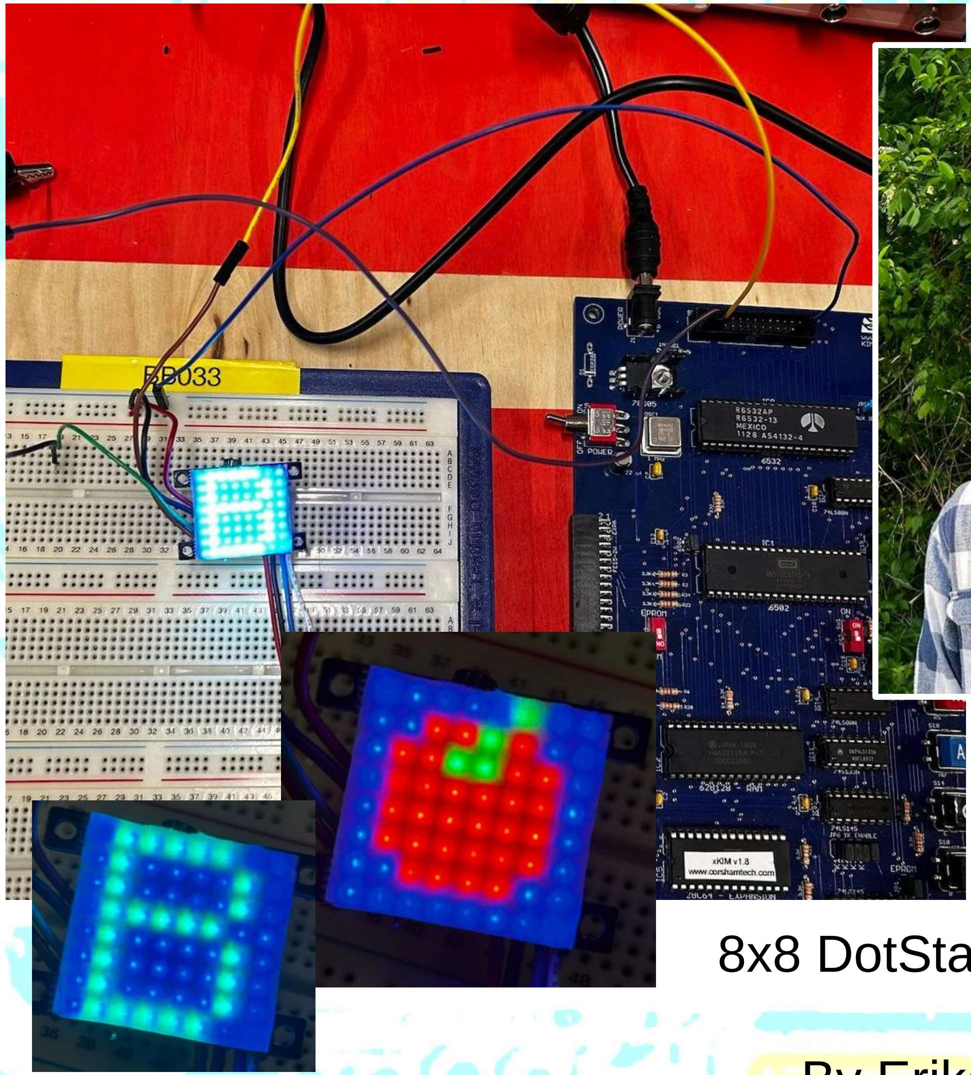
Circuits and Code

Saturday 13 May 2023



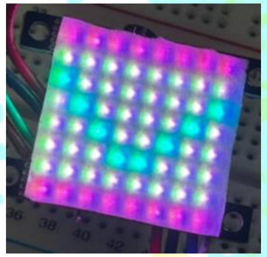
8x8 DotStar Pixel Art Display with KIM-1 Input

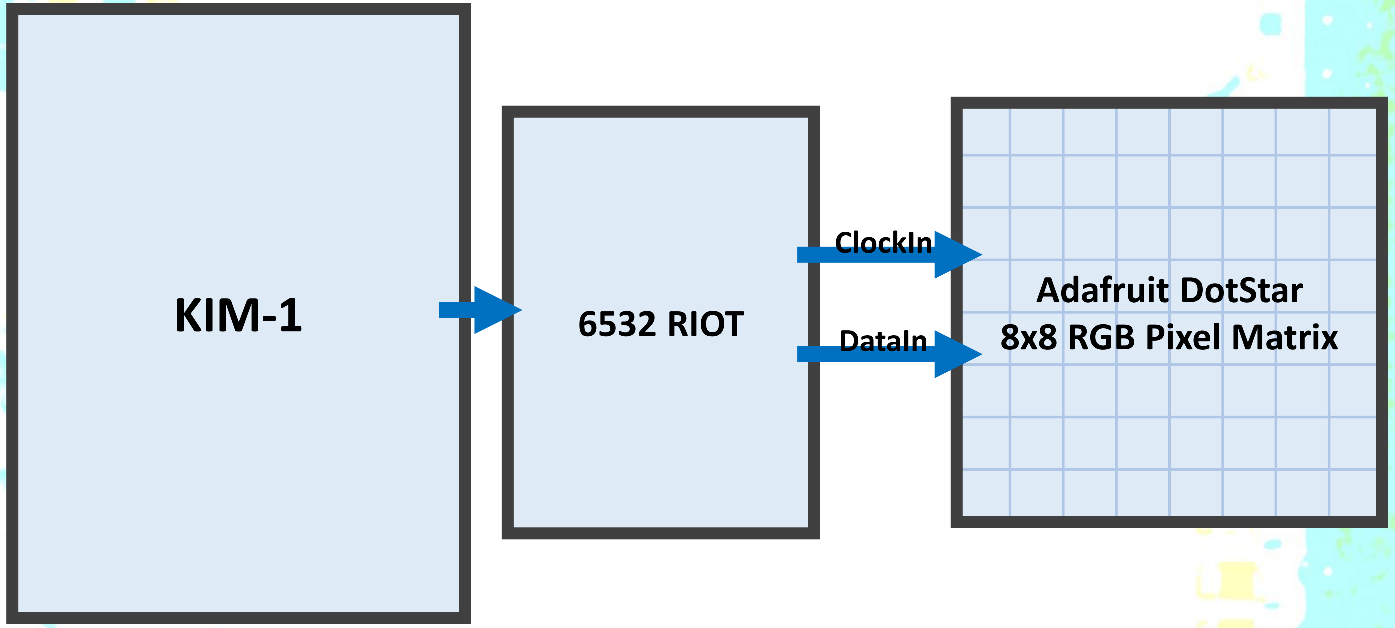
By Erika Elston, EE 2026

Circuits and Code at M5, the ECE Makerspace, Marcus Hall
Department of Electrical and Computer Engineering
College of Engineering
University of Massachusetts Amherst

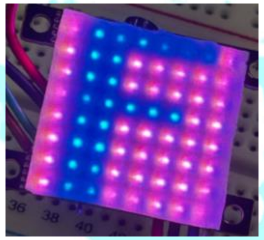
Circuits and Code

Saturday 13 May 2023





My KIM-1 program utilizes the GETKEY subroutine and continuously checks for KIM-1 keyboard input (specifically keys A through F). Each LED in the DotStar takes 32 bits of data, controlling brightness and RGB values. When a key is pressed, the program jumps to a subroutine which sends 256 bytes of data for the corresponding pixel art pattern.

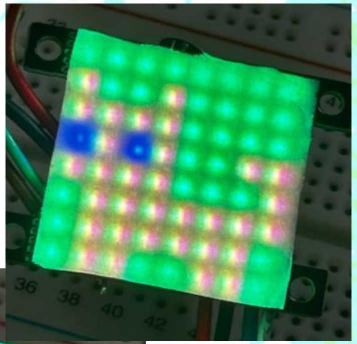
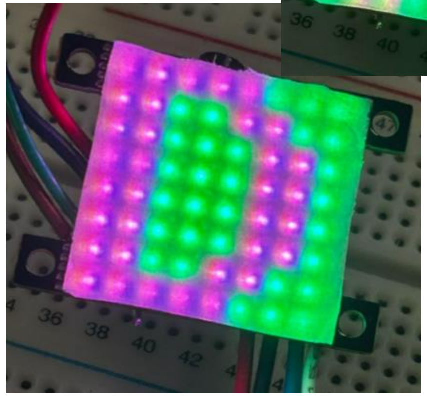


8x8 DotStar Pixel Art Display with KIM-1 Input
By Erika Elston, EE 2026

Parts List

- Adafruit DotStar 8x8 RGB LED Grid

- KIM-1 Clone
  - 6532 RIOT

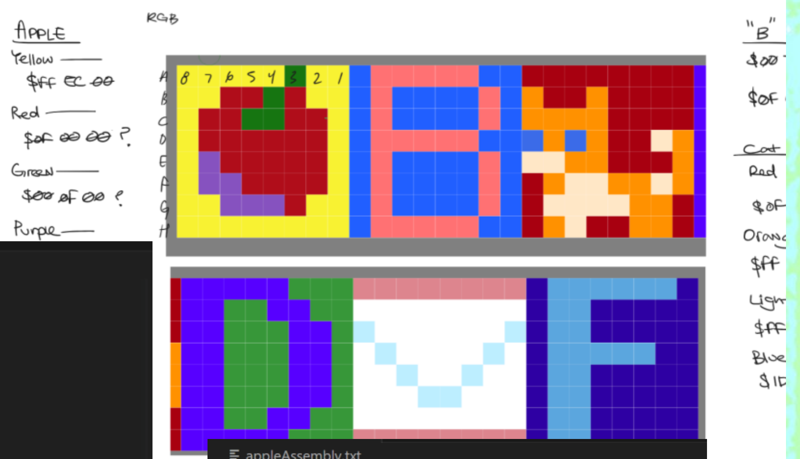
Conclusion

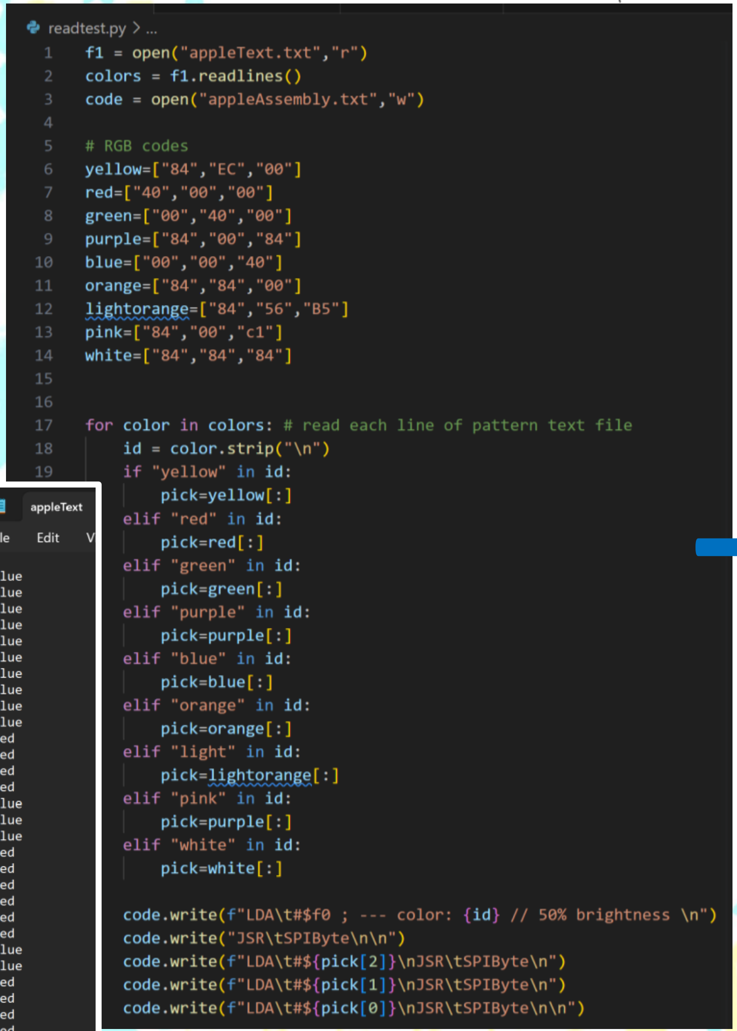
My system worked as I intended, but a few things changed throughout the process. For example, the majority of my pixel art patterns changed from their initial drafts to improve appearance in brightness, color, or both. Due to the significant size of my program, I eventually needed to reference the KIM-1 clone memory map to find a place to load my program where it could actually fit.

I enjoyed the challenge in problem solving and ability to change plans as obstacles occurred. Additionally, I appreciated applying my interests in both art and Python, something which definitely added to my engagement and fulfillment in this project. In the future, I'd like to work on a project that aims to solve a problem or make a tedious task a little bit easier.

Circuits and Code

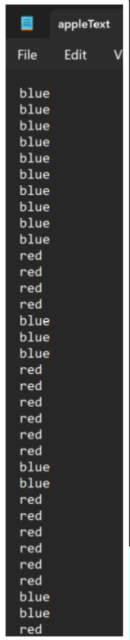
Saturday 13 May 2023





readtest.py > ...

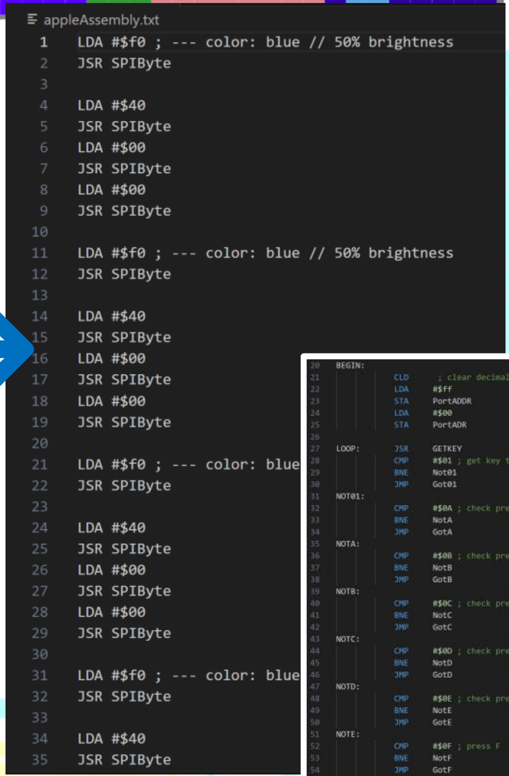
```python
f1 = open("appleText.txt","r")
colors = f1.readlines()
code = open("appleAssembly.txt","w")

# RGB codes
yellow=["84","EC","00"]
red=["40","00","00"]
green=["00","40","00"]
purple=["84","00","84"]
blue=["00","00","40"]
orange=["84","84","00"]
lightorange=["84","56","B5"]
pink=["84","00","c1"]
white=["84","84","84"]


for color in colors: # read each line of pattern text file
    id = color.strip("\n")
    if "yellow" in id:
        pick=yellow[:]
    elif "red" in id:
        pick=red[:]
    elif "green" in id:
        pick=green[:]
    elif "purple" in id:
        pick=purple[:]
    elif "blue" in id:
        pick=blue[:]
    elif "orange" in id:
        pick=orange[:]
    elif "light" in id:
        pick=lightorange[:]
    elif "pink" in id:
        pick=purple[:]
    elif "white" in id:
        pick=white[:]

    code.write(f"LDA\t#$f0 ; --- color: {id} // 50% brightness \n")
    code.write("JSR\tSPIByte\n\n")
    code.write(f"LDA\t#${pick[2]}\nJSR\tSPIByte\n")
    code.write(f"LDA\t#${pick[1]}\nJSR\tSPIByte\n")
    code.write(f"LDA\t#${pick[0]}\nJSR\tSPIByte\n\n")
```



appleText

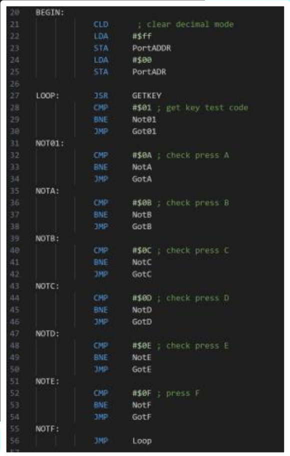
File Edit V

```
blue
blue
blue
blue
blue
blue
blue
blue
blue
blue
red
red
red
red
blue
blue
blue
red
red
red
red
red
red
blue
blue
red
red
red
red
red
red
blue
blue
red
```



appleAssembly.txt

```
LDA #$f0 ; --- color: blue // 50% brightness
JSR SPIByte

LDA #$40
JSR SPIByte
LDA #$00
JSR SPIByte
LDA #$00
JSR SPIByte

LDA #$f0 ; --- color: blue // 50% brightness
JSR SPIByte

LDA #$40
JSR SPIByte
LDA #$00
JSR SPIByte
LDA #$00
JSR SPIByte

LDA #$f0 ; --- color: blue
JSR SPIByte

LDA #$40
JSR SPIByte
LDA #$00
JSR SPIByte
LDA #$00
JSR SPIByte

LDA #$f0 ; --- color: blue
JSR SPIByte

LDA #$40
JSR SPIByte
```



```
BEGIN:
                CLD       ; clear decimal mode
                LDA     #$ff
                STA     PortADDR
                LDA     #$00
                STA     PortADR

LOOP:           JSR     GETKEY
                CMP     #$01 ; get key test code
                BNE     Not01
                JMP     Got01
NOT01:
                CMP     #$0A ; check press A
                BNE     NotA
                JMP     GotA
NOTA:
                CMP     #$0B ; check press B
                BNE     NotB
                JMP     GotB
NOTB:
                CMP     #$0C ; check press C
                BNE     NotC
                JMP     GotC
NOTC:
                CMP     #$0D ; check press D
                BNE     NotD
                JMP     GotD
NOTD:
                CMP     #$0E ; check press E
                BNE     NotE
                JMP     GotE
NOTE:
                CMP     #$0F ; press F
                BNE     NotF
                JMP     GotF
NOTF:
                JMP     Loop
```

8x8 DotStar Pixel Art Display with KIM-1 Input
By Erika Elston, EE 2026