**Project 10：8×8 Dot-matrix Display**

1. **Introduction**

The dot-matrix display is an electronic digital display device that can show information on machines, clocks and many other devices. In this project, we will use the Raspberry Pi Pico to control the 8x8 LED dot matrix to make a“❤”pattern.

1. **Components Required**

|  |  |  |
| --- | --- | --- |
| 17a6d7f241a04d4e932cb06b758197c4 | _DSC2552 | |
| Raspberry Pi Pico\*1 | Raspberry Pi Pico Expansion Board\*1 | |
| 8 | 杜邦线-3 |  |
| 8\*8 Dot-matrix Display \*1 | M-F Dupont Wires | USB Cable\*1 |

1. **Component Knowledge**

**8\*8 Dot-matrix display module:**

The 8\*8 dot matrix is composed of 64 LEDs, and each LED is placed at the intersection of a row and a column. When using a single-chip microcomputer to drive an 8\*8 dot matrix, we need to use a total of 16 digital ports, which greatly wastes the data of the single-chip microcomputer. For this reason, we specially designed this module, using the HT16K33 chip to drive an 8\*8 dot matrix, and only need to use the I2C communication port of the single-chip microcomputer to control the dot matrix, which greatly saves the microcontroller resources.

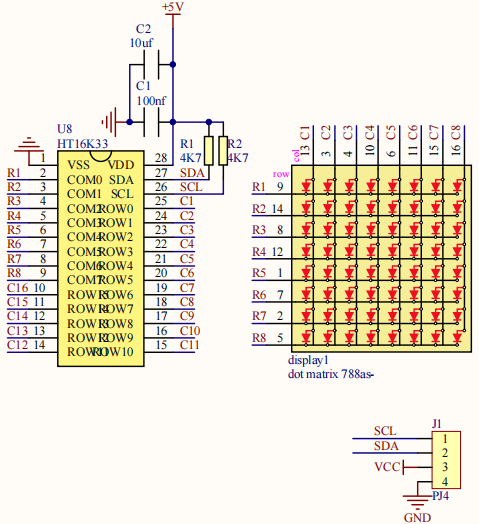
**Specifications:**

Working voltage: DC 5V

Current: 200MA

Maximum power: 1W

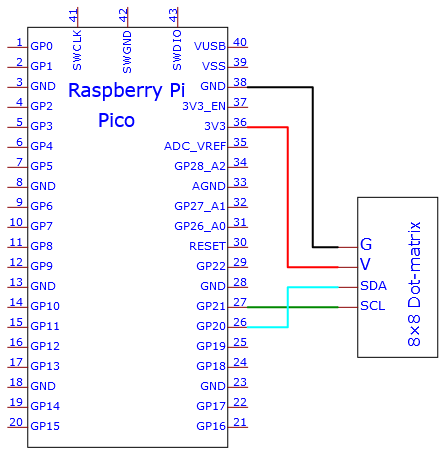
**Schematic diagram:**

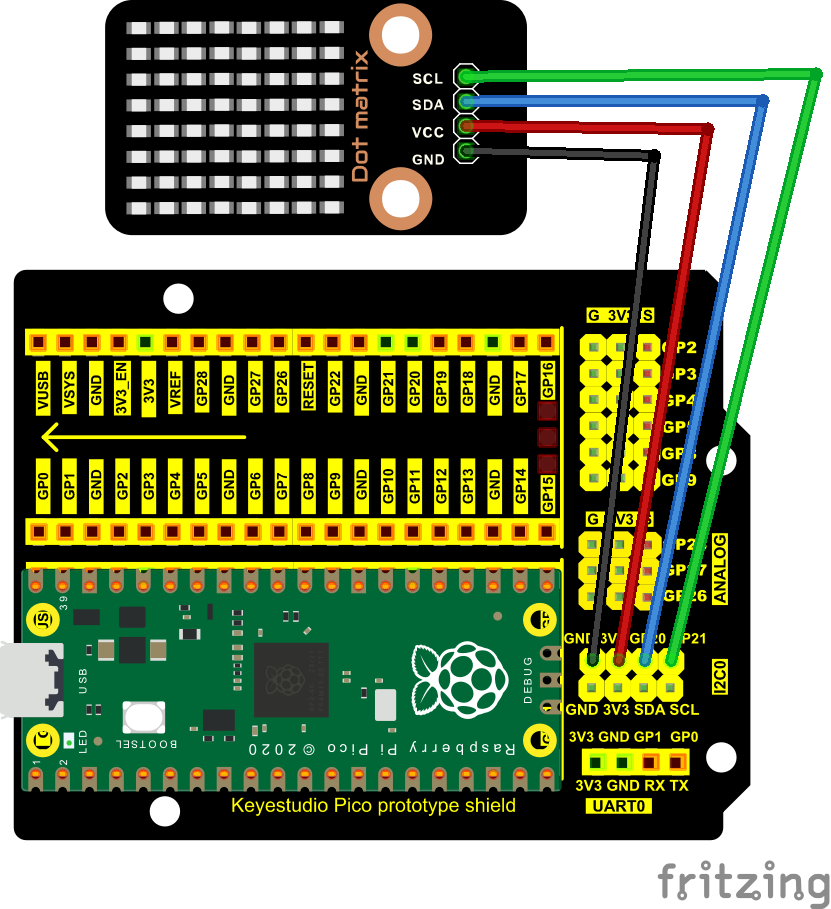


Some modules have three DIP switches that you can flip at will. These switches are used to set the I2C communication address. The setting method is as follows. The module has fixed the communication address. A0, A1 and A2 are connected to GND, and the address is 0x70.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A0（1） | A1（2） | A2（3） | A0（1） | A1（2） | A2（3） | A0（1） | A1（2） | A2（3） | |
| 0（OFF） | 0（OFF） | 0（OFF） | 1（ON） | 0（OFF） | 0（OFF） | 0（OFF） | 1（ON） | 0（OFF） | |
| 0X70 | | | 0X71 | | | 0X72 | | |
| A0（1） | A1（2） | A2（3） | A0（1） | A1（2） | A2（3） | A0（1） | A1（2） | A2（3） | |
| 1（ON） | 1（ON） | 0（OFF） | 0（OFF） | 0（OFF） | 1（ON） | 1（ON） | 0（OFF） | 1（ON） | |
| 0X73 | | | 0X74 | | | 0X75 | | |
| A0（1） | A1（2） | A2（3） | A0（1） | A1（2） | A2（3） |  | | | |
| 0（OFF） | 1（ON） | 1（ON） | 1（ON） | 1（ON） | 1（ON） |
| 0X76 | | | 0X77 | | |

1. **Circuit Diagram and Wiring Diagram**



****

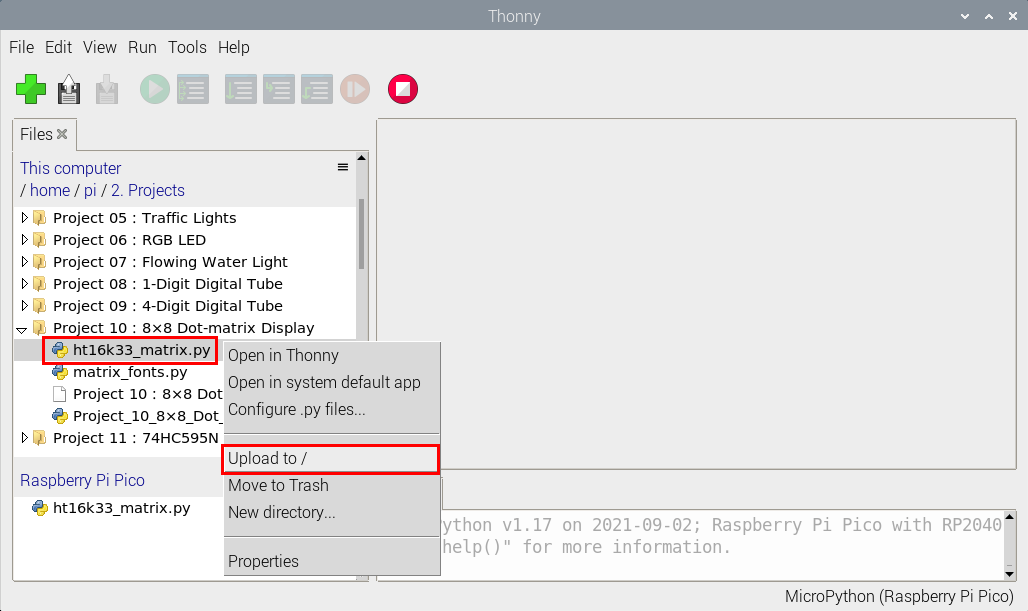
**Test Code**

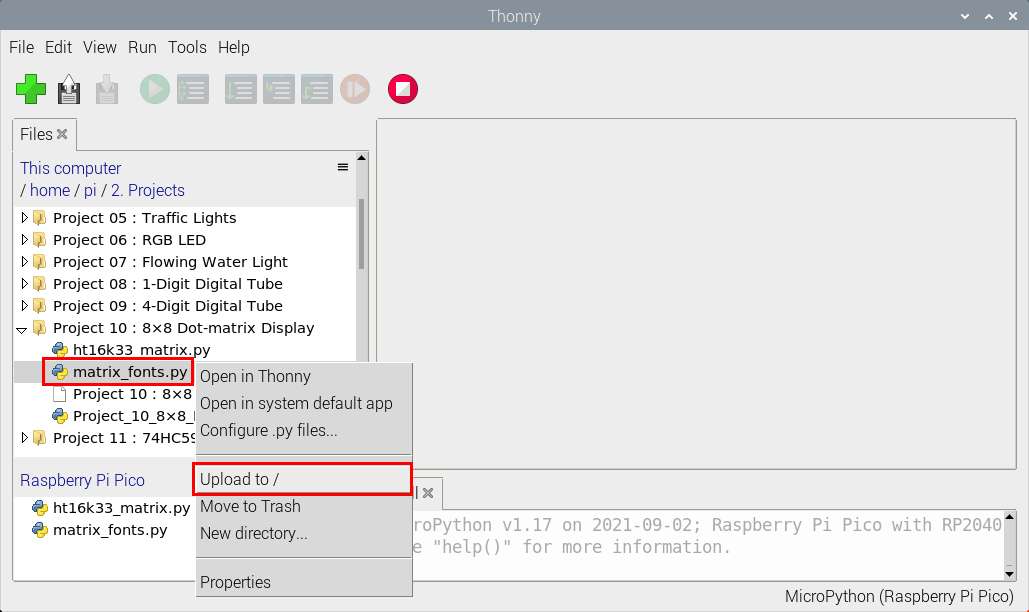
The code used in this project is saved in the file KS3020 Keyestudio Raspberry Pi Pico Learning Kit Ultimate Edition\3. Raspberry Pi System\Python\_Tutorial\2. Projects\Project 10：8×8 Dot-matrix Display.

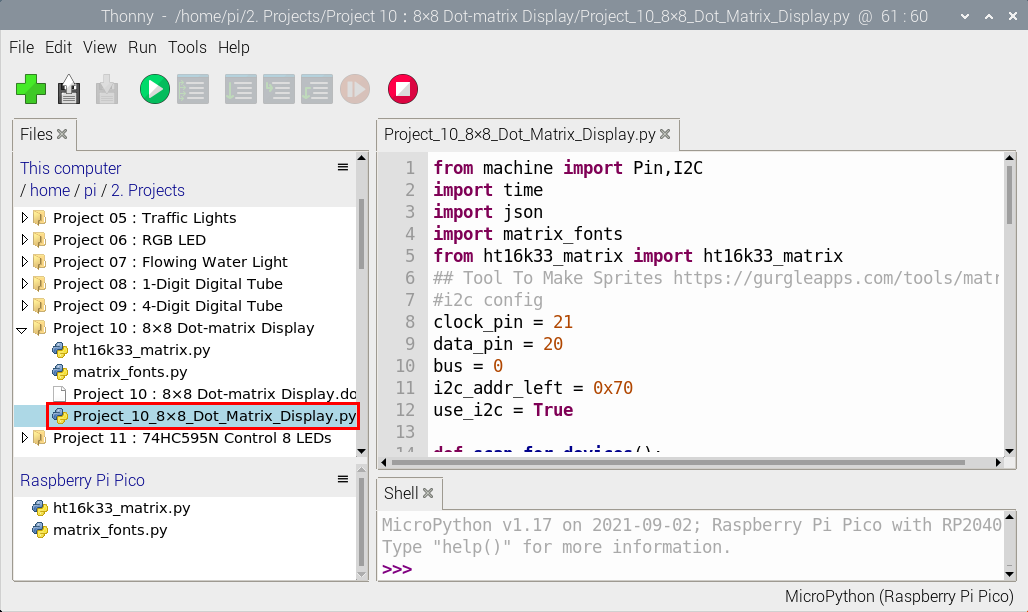
You can move the code to anywhere.For example, we save it in the pi folder of the Raspberry Pi system, the route is home/pi/2. Projects.



Open“Thonny, click “This computer”→“home”→“pi”→“2. Projects”→“Project 10：8×8 Dot-matrix Display”. Select“ht16k33\_matrix.py”and“matrix\_fonts.py”，right-click and select“Upload to /”and wait for“ht16k33\_matrix.py”and“matrix\_fonts.py”to be uploaded to the Raspberry Pi Pico. And double-click the“Project\_10\_8×8\_Dot\_Matrix\_Display.py”.



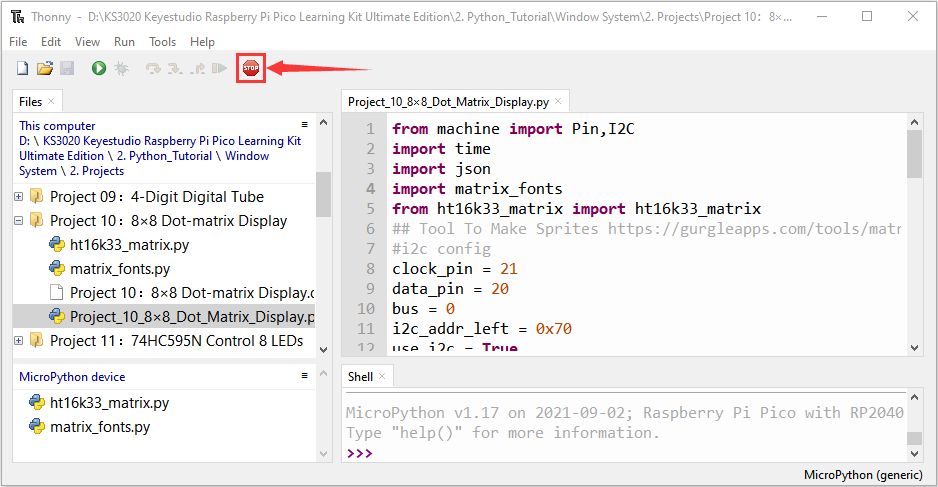




|  |
| --- |
| from machine import Pin,I2C  import time  import json  import matrix\_fonts  from ht16k33\_matrix import ht16k33\_matrix  ## Tool To Make Sprites https://gurgleapps.com/tools/matrix  #i2c config  clock\_pin = 21  data\_pin = 20  bus = 0  i2c\_addr\_left = 0x70  use\_i2c = True  def scan\_for\_devices():  i2c = machine.I2C(bus,sda=machine.Pin(data\_pin),scl=machine.Pin(clock\_pin))  devices = i2c.scan()  if devices:  for d in devices:  print(hex(d))  else:  print('no i2c devices')  if use\_i2c:  scan\_for\_devices()  left\_eye = ht16k33\_matrix(data\_pin, clock\_pin, bus, i2c\_addr\_left)  def show\_char(left):  if use\_i2c:  left\_eye.show\_char(left)    def scroll\_message(font,message='hello',delay=0.05): #Scrolling display  left\_message = ' ' + message  right\_message = message + ' '  length=len(right\_message)  char\_range=range(length-1)  for char\_pos in char\_range:  right\_left\_char=font[right\_message[char\_pos]]  right\_right\_char=font[right\_message[char\_pos+1]]  left\_left\_char=font[left\_message[char\_pos]]  left\_right\_char=font[left\_message[char\_pos+1]]  for shift in range(8):  left\_bytes=[0,0,0,0,0,0,0,0]  right\_bytes=[0,0,0,0,0,0,0,0]  for col in range(8):  left\_bytes[col]=left\_bytes[col]|left\_left\_char[col]<<shift  left\_bytes[col]=left\_bytes[col]|left\_right\_char[col]>>8-shift;  right\_bytes[col]=right\_bytes[col]|right\_left\_char[col]<<shift  right\_bytes[col]=right\_bytes[col]|right\_right\_char[col]>>8-shift;  if use\_i2c:  left\_eye.show\_char(left\_bytes)  time.sleep(delay)  while True:  show\_char(matrix\_fonts.textFont1['A']) #Show the letter A  time.sleep(1)  show\_char(matrix\_fonts.textFont1['B'])  time.sleep(1)  show\_char(matrix\_fonts.textFont1['C'])  time.sleep(1)  scroll\_message(matrix\_fonts.textFont1, ' Hello World ') |

1. **Test Result**

Ensure that the Raspberry Pi Pico is connected to the computer, click “Stop/Restart backend”.



Click “Run current script”, the code starts executing, we will see that the 8 x 8 dot matrix displays the character "A" 1S, "B" 1S, and "C" 1S. Then scroll to display the string "Hello World”repeatedly. Click“Stop/Restart backend”to exit the program.

