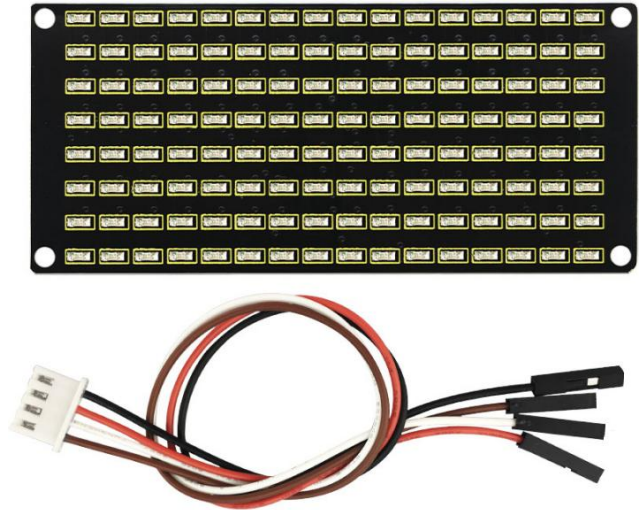


Project 9 Facial Expression LED Board

1.Description

How fun it is if a expression board is added to the robot. And the Keyestudio 8*16 LED board can do the trick. With the help of it, you could design facial expressions, images, patterns and other displays by yourselves.



The 8*16 LED board comes with 128 LEDs. The data of the microprocessor (Arduino) communicates with the AiP1640 through a two-wire bus interface. Therefore, it can control the on and off of 128 LEDs on the module, so as to make the dot matrix on the module to display the pattern you need. A HX-2.54 4Pin cable is provided for your convenience of wiring.

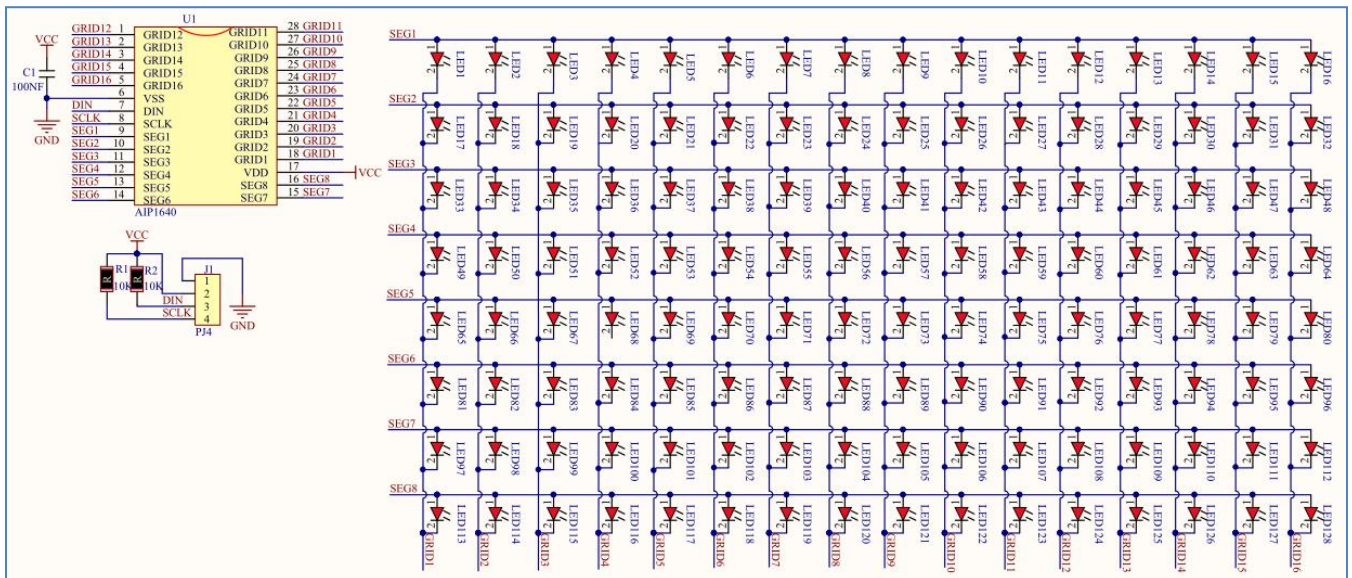
2.Specification

- Working voltage: DC 3.3-5V
- Power loss: 400mW
- Oscillation frequency: 450KHz
- Drive current: 200mA

keyestudio

- Working temperature: -40~80°C
- Communication mode: I2C

Circuit Diagram



Working Principle

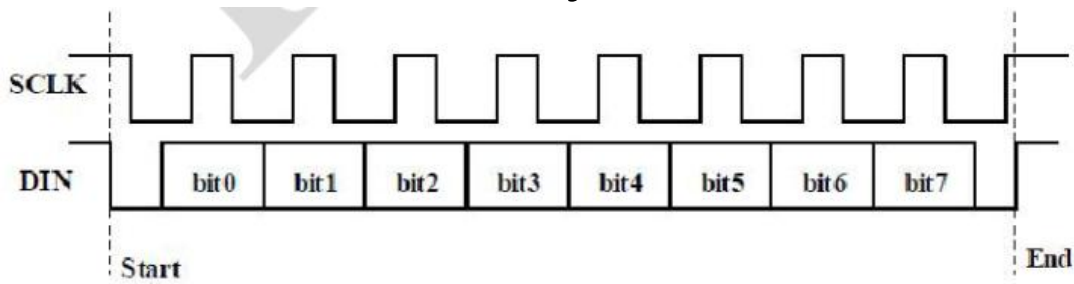
How to control each LED of the 8*16 dot matrix? It is known that each byte has 8 bits and each bit is 0 or 1. when it is 0, LED is off while when it is 1 LED is on. One byte can control one column of the LED, and naturally 16 bytes can control 16 columns of LEDs, that's the 8*16 dot matrix.

Pins description and communication protocol

The data of the microprocessor (Arduino) communicates with the AiP1640 through a two-wire bus cable.

The communication protocol diagram is as follows (SCLK) is SCL, (DIN) is SDA.

keyestudio



① The starting condition for data input: SCL is high level and SDA changes from high to low.

② For data command setting, there are methods as shown in the figure below

In our sample program, select the way to **add 1 to the address automatically**, the binary value is 0100 0000 and the corresponding hexadecimal value is 0x40

B7	B6	B5	B4	B3	B2	B1	B0	Description
0	1	Irrelevant choice, fill in 0			0	Irrelevant choice, fill in 0		add 1 to the address
0	1				1			automatically
0	1		0					Fixed address
0	1		1					Universal mode
								Test mode

③ For address command setting, the address can be selected as shown below.

The first 00H is selected in our sample program, and the binary number 1100 0000 corresponds to the hexadecimal 0xc0.

keyestudio

B7	B6	B5	B4	B3	B2	B1	B0	Display address
1	1	Irrelevant choice, fill in 0		0	0	0	0	00H
1	1			0	0	0	1	01H
1	1			0	0	1	0	02H
1	1			0	0	1	1	03H
1	1			0	1	0	0	04H
1	1			0	1	0	1	05H
1	1			0	1	1	0	06H
1	1			0	1	1	1	07H
1	1			1	0	0	0	08H
1	1			1	0	0	1	09H
1	1			1	0	1	0	0AH
1	1			1	0	1	1	0BH
1	1			1	1	0	0	0CH
1	1			1	1	0	1	0DH
1	1			1	1	1	0	0EH
1	1			1	1	1	1	0FH

④The requirement for data input is that when SCL is at high level when inputting data, the signal on SDA must remain unchanged. Only when the clock signal on SCL is at low level, can the signal on SDA be changed. The input of data is the low bit first, and the high bit later.

⑤The condition for the end of data transmission is that when SCL is at low level, SDA at low level and SCL at high level, the level of SDA becomes high.

⑥Display control, set different pulse width, pulse width can be selected as shown in the figure below

In the example, the pulse width is 4/16, and the hexadecimal corresponding to 1000 1010 is 0x8A

keyestudio

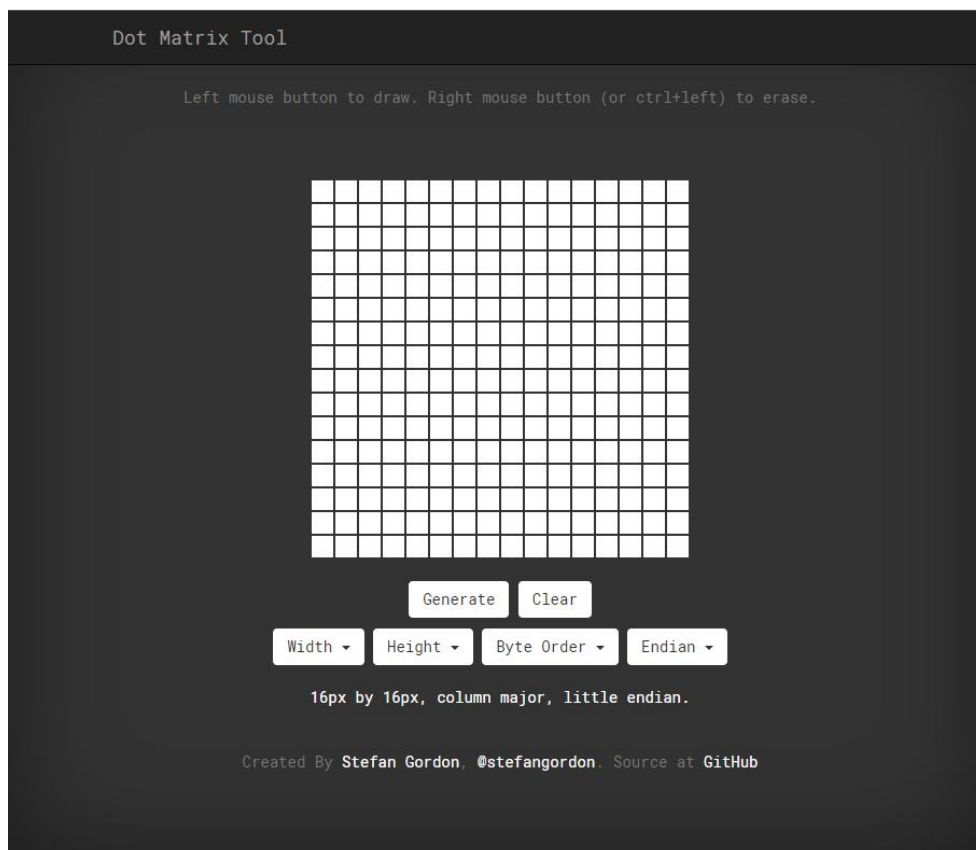
B7	B6	B5	B4	B3	B2	B1	B0	Function	Description
1	0	Irrelevant choice, fill in 0		1	0	0	0	Clear quantity setting (Brightness setting)	Set pulse width to 1/16
1	0			1	0	0	1		Set pulse width to 2/16
1	0			1	0	1	0		Set pulse width to 4/16
1	0			1	0	1	1		Set pulse width to 10/16
1	0			1	1	0	0		Set pulse width to 11/16
1	0			1	1	0	1		Set pulse width to 12/16
1	0			1	1	1	0		Set pulse width to 13/16
1	0			1	1	1	1		Set pulse width to 14/16
1	0			0	X	X	X	Display switch setting	On off
1	0			1	X	X	X		

Instructions for the use of modulus tool

The dot matrix tool uses the online version, and the link is :

<http://dotmatrixtool.com/#>

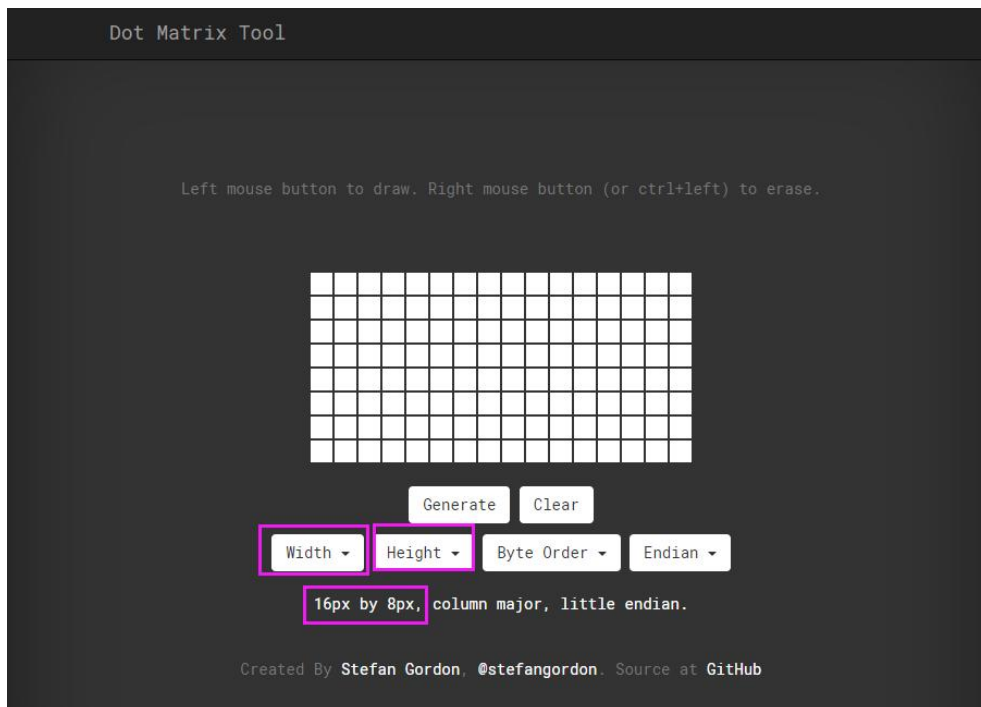
①Enter the link and the page appears as shown below



www.keyestudio.com

keyestudio

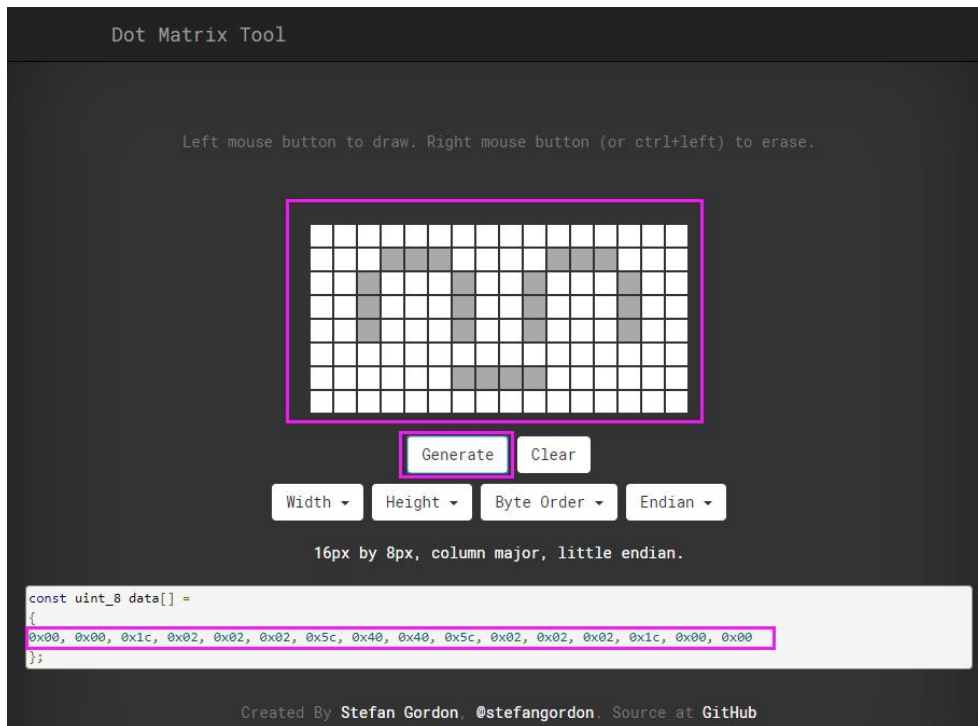
②The dot matrix is 8*16, so adjust the height to 8 and width to 16, as shown in the figure below



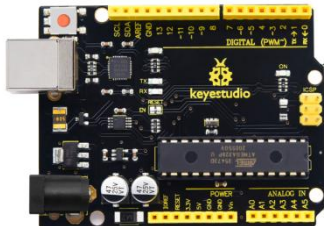
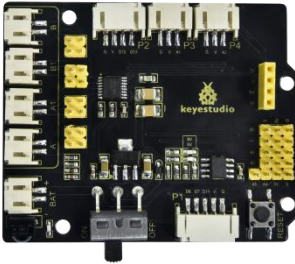



③Generate hexadecimal data from the pattern

As shown in the figure below, press the left mouse button to select, right click to cancel; draw the pattern you want, click Generate, and the hexadecimal data we need will be generated.

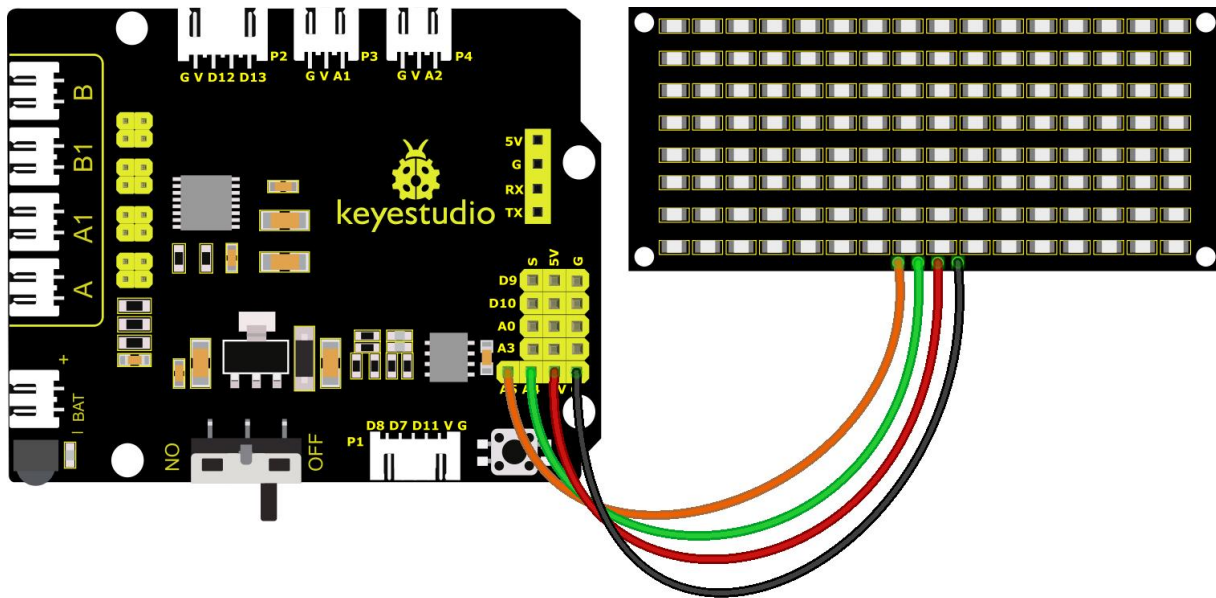
keystudio



3.Components

Keystudio 4.0 Development Board *1	Keystudio 8833 Motor Driver Expansion Board *1	Keystudio 8x16 LED Panel*1
		
USB Cable*1	HX-2.54 4P Dupont Wire 200mm *1	
		

4.Wiring Diagram



The GND, VCC, SDA, and SCL of the 8x16 LED light board are respectively connected to the keystudio sensor expansion board-(GND), + (VCC), A4, A5 for two-wire serial communication.

(Note: Though it is connected with the IIC pin of Arduino, this module is not for IIC communication. And the IO port here is to simulate I2C communication and can be connected with any two pins)

5.Test Code

The code will show the smile face

```
//*****  
/*  
  keystudio 4wd BT Car  
  lesson 9.1  
  Matrix face  
  http://www.keystudio.com  
*/  
  
//Data from the smile pattern obtained from the touch tool  
unsigned char smile[] = {0x00, 0x00, 0x1c, 0x02, 0x02, 0x02, 0x5c, 0x40, 0x40, 0x5c, 0x02, 0x02, 0x02, 0x1c,
```


keystudio

```
0x00, 0x00};
#define SCL_Pin  A5  //Set the clock pin to A5
#define SDA_Pin  A4  //Set data pin to A4
void setup() {
    //Set pin to output
    pinMode(SCL_Pin, OUTPUT);
    pinMode(SDA_Pin, OUTPUT);
    //clear
    //matrix_display(clear);
}
void loop() {
    matrix_display(smile); //display smiling expression pattern
}
//this function is used for dot matrix display
void matrix_display(unsigned char matrix_value[])
{
    IIC_start(); //the function that calls the data transfer start condition
    IIC_send(0xc0); //select address

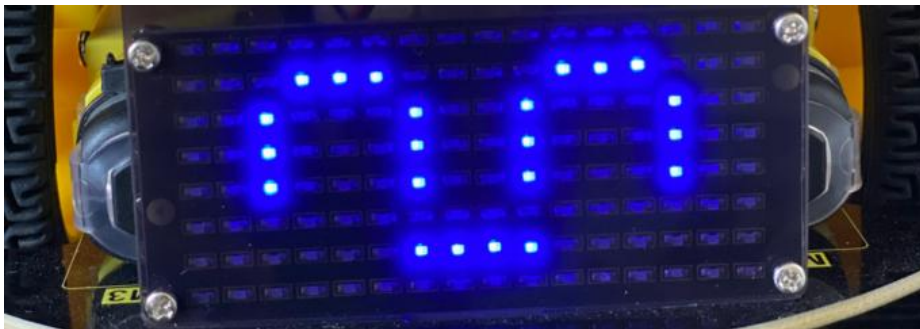
    for (int i = 0; i < 16; i++) //the pattern data is 16 bytes
    {
        IIC_send(matrix_value[i]); //Transmit the data of the pattern
    }
    IIC_end(); //End pattern data transmission
    IIC_start();
    IIC_send(0x8A); //Display control, select 4/16 pulse width
    IIC_end();
}
//Conditions under which data transmission begins
void IIC_start()
{
    digitalWrite(SDA_Pin, HIGH);
    digitalWrite(SCL_Pin, HIGH);
    delayMicroseconds(3);
    digitalWrite(SDA_Pin, LOW);
    delayMicroseconds(3);
    digitalWrite(SCL_Pin, LOW);
}
//Indicates the end of data transmission
void IIC_end()
{
    digitalWrite(SCL_Pin, LOW);
    digitalWrite(SDA_Pin, LOW);
    delayMicroseconds(3);
}
```

keyestudio

```
digitalWrite(SCL_Pin, HIGH);
delayMicroseconds(3);
digitalWrite(SDA_Pin, HIGH);
delayMicroseconds(3);
}
//transmit data
void IIC_send(unsigned char send_data)
{
    for (byte mask = 0x01; mask != 0; mask <<= 1) //Each byte has 8 bits and is checked bit by bit starting
    at the lowest level
    {
        if (send_data & mask) { //Sets the high and low levels of SDA_Pin depending on whether each bit of the
        byte is a 1 or a 0
            digitalWrite(SDA_Pin, HIGH);
        } else {
            digitalWrite(SDA_Pin, LOW);
        }
        delayMicroseconds(3);
        digitalWrite(SCL_Pin, HIGH); //Pull the clock pin SCL_Pin high to stop data transmission
        delayMicroseconds(3);
        digitalWrite(SCL_Pin, LOW); //pull the clock pin SCL_Pin low to change the SIGNAL of SDA
    }
}
//*****
```

6.Test Result

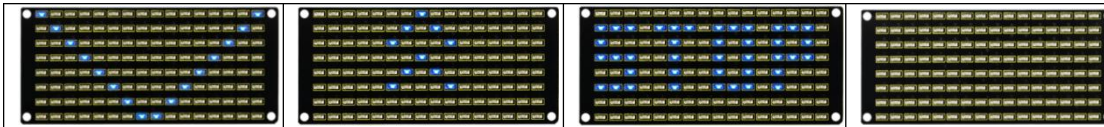
After successfully uploading the code to the V4.0 board, connect the wirings according to the wiring diagram, then turn the DIP switch to ON, a smile-shaped pattern will be displayed on the LED board.



keyestudio

7.Code Explanation

We use the modulus tool we just learned, <http://dotmatrixtool.com/#>, to make the dot matrix display the start pattern, going forward, and stop and then clear the pattern. The time interval is 2000 ms.



Code obtained from the module tool:

Code for the pattern start:

0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80,0x80,0x40,0x20,0x10,0x08,0x04,0x02,0x01

Code for the pattern going forward:

0x00,0x00,0x00,0x00,0x00,0x24,0x12,0x09,0x12,0x24,0x00,0x00,0x00,0x00,0x00,0x00

Code for the pattern stepping back:

0x00,0x00,0x00,0x00,0x00,0x24,0x48,0x90,0x48,0x24,0x00,0x00,0x00,0x00,0x00,0x00

Code for the pattern turning left:

0x00,0x00,0x00,0x00,0x00,0x00,0x44,0x28,0x10,0x44,0x28,0x10,0x44,0x28,0x10,0x00

Code for the pattern turning right:

0x00,0x10,0x28,0x44,0x10,0x28,0x44,0x10,0x28,0x44,0x00,0x00,0x00,0x00,0x00,0x00

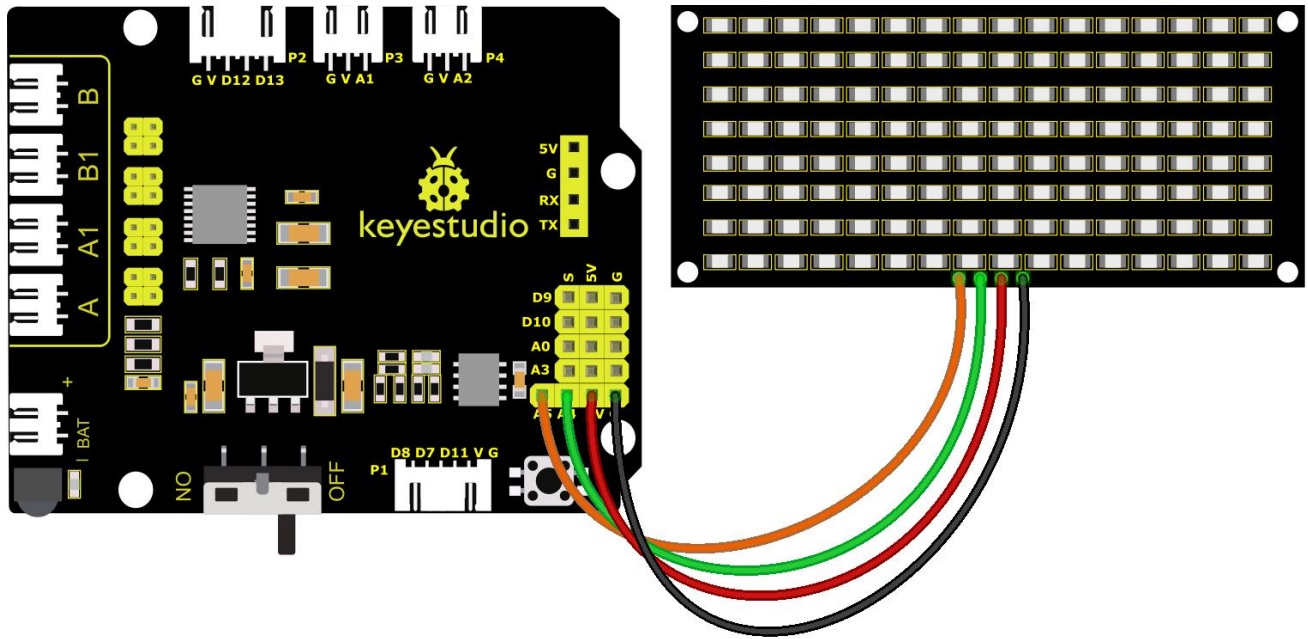
Code for the pattern stop:

0x2E,0x2A,0x3A,0x00,0x02,0x3E,0x02,0x00,0x3E,0x22,0x3E,0x00,0x3E,0x0A,0x0E,0x00

Code to clear screen:

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00

keystudio



```
//*****  
/*  
  keystudio 4wd BT Car  
  lesson 9.2  
  Matrix face  
  http://www.keystudio.com  
*/  
//Data from the smile pattern obtained from the touch tool  
unsigned char start01[] = {0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x80, 0x40, 0x20, 0x10, 0x08,  
0x04, 0x02, 0x01};  
unsigned char front[] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x24, 0x12, 0x09, 0x12, 0x24, 0x00, 0x00, 0x00, 0x00,  
0x00, 0x00};  
unsigned char back[] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x24, 0x48, 0x90, 0x48, 0x24, 0x00, 0x00, 0x00, 0x00,  
0x00, 0x00};  
unsigned char left[] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x44, 0x28, 0x10, 0x44, 0x28, 0x10, 0x44, 0x28,  
0x10, 0x00};  
unsigned char right[] = {0x00, 0x10, 0x28, 0x44, 0x10, 0x28, 0x44, 0x10, 0x28, 0x44, 0x00, 0x00, 0x00, 0x00,  
0x00, 0x00};  
unsigned char STOP01[] = {0x2E, 0x2A, 0x3A, 0x00, 0x02, 0x3E, 0x02, 0x00, 0x3E, 0x22, 0x3E, 0x00, 0x3E,  
0x0A, 0x0E, 0x00};  
unsigned char clear[] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  
0x00, 0x00};  
#define SCL_Pin  A5  //Set the clock pin to A5  
#define SDA_Pin  A4  //Set data pin to A4  
void setup() {  
  //Set pin to output  
  pinMode(SCL_Pin, OUTPUT);  
  pinMode(SDA_Pin, OUTPUT);  
}
```

keyestudio

```
//clear
//matrix_display(clear);
}

void loop() {
    matrix_display(start01); //Show start pattern
    delay(2000);
    matrix_display(front);    //Show forward pattern
    delay(2000);
    matrix_display(STOP01);   //Show stop pattern
    delay(2000);
    matrix_display(clear);    //Clear Screen
    delay(2000);
}

//this function is used for dot matrix display
void matrix_display(unsigned char matrix_value[])
{
    IIC_start(); //the function that calls the data transfer start condition
    IIC_send(0xc0); //select address

    for (int i = 0; i < 16; i++) //the pattern data is 16 bytes
    {
        IIC_send(matrix_value[i]); //Transmit the data of the pattern
    }
    IIC_end(); //End pattern data transmission
    IIC_start();
    IIC_send(0x8A); //Display control, select 4/16 pulse width
    IIC_end();
}

//Conditions under which data transmission begins
void IIC_start()
{
    digitalWrite(SDA_Pin, HIGH);
    digitalWrite(SCL_Pin, HIGH);
    delayMicroseconds(3);
    digitalWrite(SDA_Pin, LOW);
    delayMicroseconds(3);
    digitalWrite(SCL_Pin, LOW);
}

//Indicates the end of data transmission
void IIC_end()
{
    digitalWrite(SCL_Pin, LOW);
    digitalWrite(SDA_Pin, LOW);
    delayMicroseconds(3);
}
```

keystudio

```
digitalWrite(SCL_Pin, HIGH);
delayMicroseconds(3);
digitalWrite(SDA_Pin, HIGH);
delayMicroseconds(3);
}
//transmit data
void IIC_send(unsigned char send_data)
{
    for (byte mask = 0x01; mask != 0; mask <<= 1) //Each byte has 8 bits and is checked bit by bit starting
    at the lowest level
    {
        if (send_data & mask) { //Sets the high and low levels of SDA_Pin depending on whether each bit of the
        byte is a 1 or a 0
            digitalWrite(SDA_Pin, HIGH);
        } else {
            digitalWrite(SDA_Pin, LOW);
        }
        delayMicroseconds(3);
        digitalWrite(SCL_Pin, HIGH); //Pull the clock pin SCL_Pin high to stop data transmission
        delayMicroseconds(3);
        digitalWrite(SCL_Pin, LOW); //pull the clock pin SCL_Pin low to change the SIGNAL of SDA
    }
}
//*****
```

After uploading test code, the facial expression board shows these patterns orderly and repeats this sequence.

