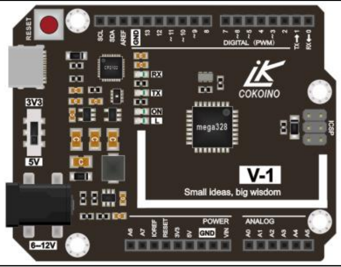
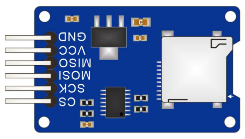
23.SD card and SD card module

ABOUT THIS PROJECT:

You will learn:

- Use the uno r3 to control the text of the LCD display

1、Things used in this project

Hardware components	Picture	Quantity
V-1 board		1 PCS
30CM USB Cable		1 PCS
SD card reading and writing module		1 PCS
Male to female DuPont line		6 PCS
128M Micro SD card		1 PCS

2、Micro SD Card Module Introduction

The module (Micro SD Card Adapter) is a micro SD card reading and writing module. The files in the Micro SD card can be read and written by the SPI interface driver of the single chip microcomputer. Arduino users can use the SD card library that comes with the Arduino IDE to initialize and read and write the card.

Features:
1. Support Micro SD (<=2G) card, Micro SDHC (<=32G) card (high speed card)
2. Onboard level shifting circuit, ie the interface level can be 5V or 3.3V
3. Power supply is 4.5V~5.5V, onboard 3.3V regulator circuit
4. Communication interface is standard SPI interface
5.4 M2 screw positioning holes for easy installation
Note: Make sure the MicroSD card is formatted in FAT16 or FAT32 format and plug it into the Micro SD card module.

2.1、SD Card Introduction

The Micro SD card is a very small flash memory card. The format is derived from SanDisk. The original memory card is called T-Flash and later renamed Trans Flash. The reason for renaming it to Micro SD is because it was The SD Association (SDA) is established. Its size is 15mm x 11mm x1mm, which is almost equal to the size of the fingernails. It is the smallest memory card available. It can also be used in the SD card slot via an SD riser adapter. At present, MicroSD cards provide 128MB, 256MB, 512MB, 1G, 2G, 4G, 8G, 16G, 32G, 64G, 128G, 256G, 512G and other capacities.

2.2、Serial Peripheral Interface (SPI)

SPI, short for Serial Peripheral Interface, is a high-speed, full-duplex, synchronous communication bus that occupies only four wires on the pins of the chip.

The communication principle of SPI is simple. It works in master-slave mode, which usually has one master device and one or more slave devices, requiring at least four wires, which are MISO (master device data input), MOSI (master device data output), SCLK (clock), and CS (chip).

SCLK: Serial Clock (output from master)
MOSI: Master Output Slave Input, or Master Out Slave In (data output from master)
MISO: Master Input Slave Output, or Master In Slave Out (data output from slave)
SS: Slave Select (often active low, output from master)

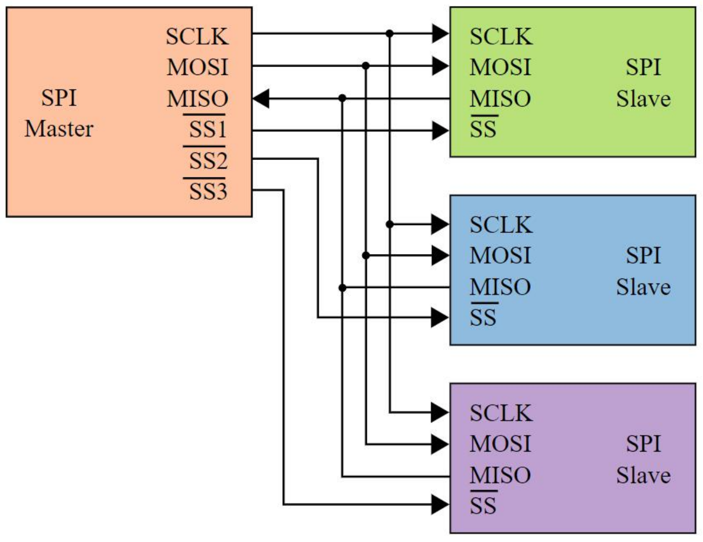
CS is the control signal of whether the chip is selected by the master chip. That is to say, the master chip can operate on the slave chip only when the chip selection signal is valid (high or low potential).

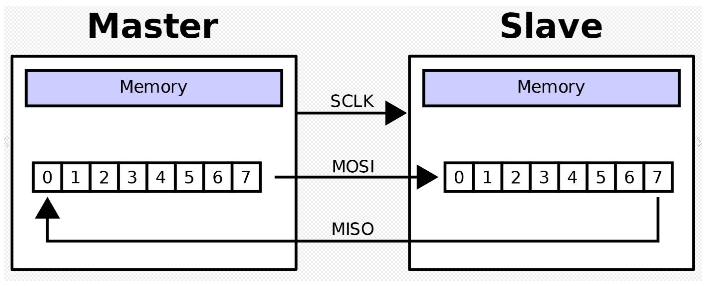
SPI is a serial communication protocol in which data is transmitted bit by bit.That is why the SCLK clock line exists, and the SCLK provides the clock pulse on which SDI and SDO transmit data.With SDO output data, the data changes on the rising or falling edge of the clock and is read on the following falling or rising edge.Input works the same way, so it takes at least eight clock changes (one rise and one fall) to complete the 8-bit data transfer.

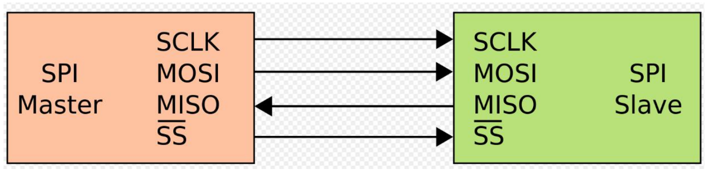
SCLK signal line is only controlled by the main device, the slave device cannot control the signal line.Also, in an SPI based device, there is at least one master device.One advantage of this transmission is that the SPI allows data to be transferred bit by bit, even to pause, because the SCLK clock line is controlled by the master device, and no data is collected or transmitted from the slave device when there is no clock jump.In other words, the master device can control the communication by controlling the SCLK clock line.

SPI is also a data exchange protocol, because SPI's data input and output lines are independent, allowing both data input and output to be completed at the same time.Different SPI devices are implemented in different ways, mainly due to different data change and acquisition time. Please refer to the relevant chip documentation for details.

Finally, one disadvantage of the SPI interface is that there is no specified flow control and no reply mechanism to confirm whether data is received or not.







For more information on SPI communication protocols, please refer to:
http://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus#Mode_Numbers

UNO R3 SPI pin:

Board	MOSI	MISO	SCK	SS (slave)	SS (master)	Level
UNO R3	11	12	13	10	-	5V

More arduino SPI library information reference:
https://www.arduino.cc/en/Reference/SPI

3、Read and write micro SD card

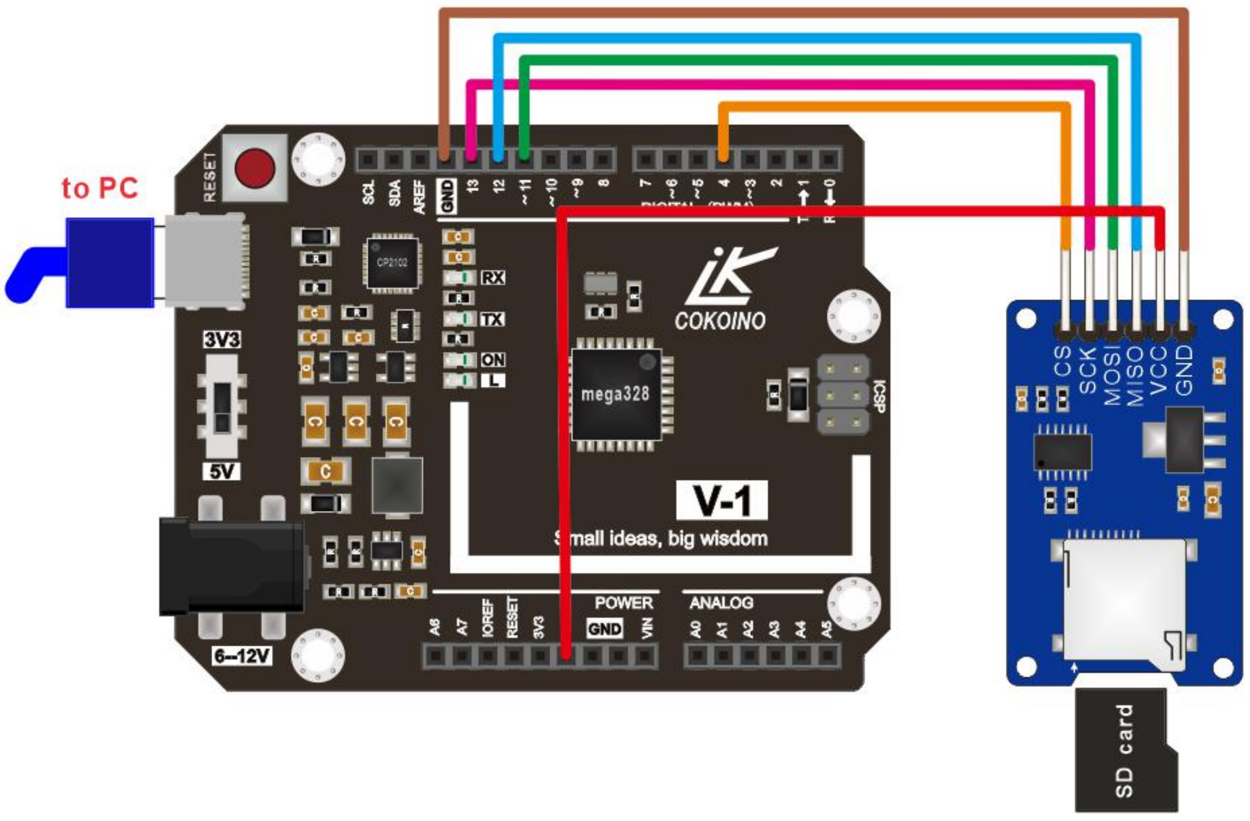
Arduino IDE comes with the relevant SD card library file, in File--->Examples--->SD. The official website has detailed instructions for use, please refer to the following web link:
https://www.arduino.cc/en/Reference/SD

3.1、Code:

```
/** SD card attached to SPI bus as follows:
  ** MOSI - pin 11
  ** MISO - pin 12
  ** CLK - pin 13
  ** CS - pin 4
  */
#include <SPI.h>
#include <SD.h>
File myFile;
/////////////////////////////////////////////////////
void setup() {
   Serial.begin(9600);    // Open serial communications and wait for port to open:
   while (!Serial);    // wait for serial port to connect. Needed for native USB port only
   Serial.print("Initializing SD card...");
   if (!SD.begin(4)) {
       Serial.println("initialization failed!");
       return;
   }
```

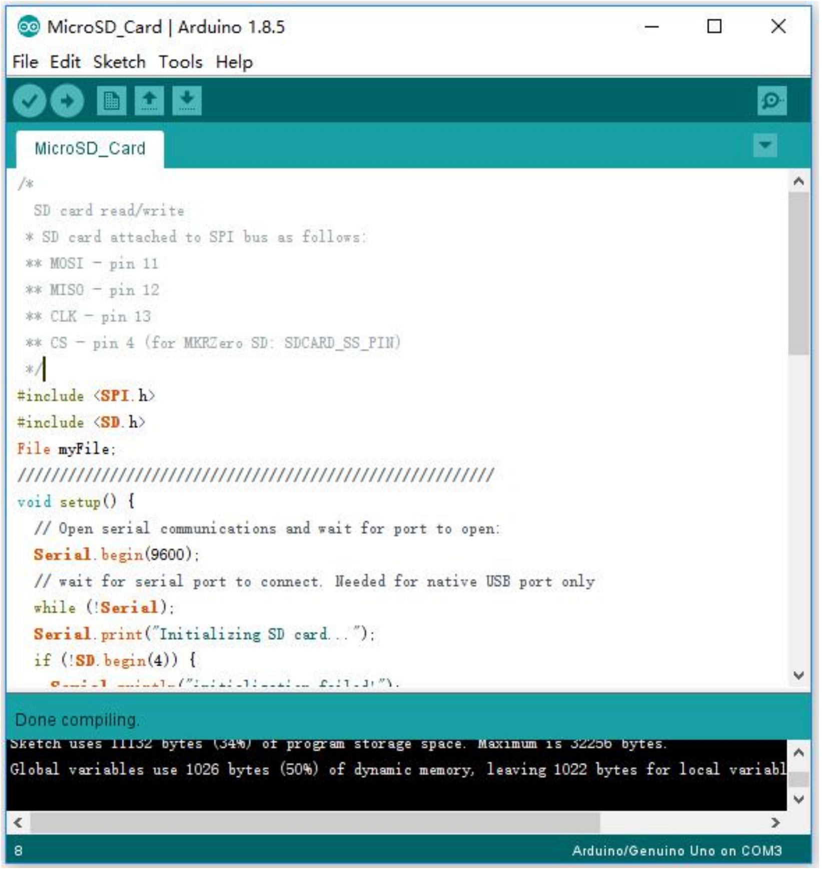
```
    Serial.println("initialization done.");
    // open the file. note that only one file can be open at a time,
    // so you have to close this one before opening another.
    myFile = SD.open("test.txt", FILE_WRITE);
    if (myFile) {           // if the file opened okay, write to it:
        Serial.print("Writing to test.txt...");
        myFile.println("my memory!");    //Records written to SD card
        myFile.close();      // close the file:
        Serial.println("done.");
    } else {
        Serial.println("error opening test.txt"); // if the file didn't open, print an error:
    }
    // re-open the file for reading:
    myFile = SD.open("test.txt");
    if (myFile) {
        Serial.println("test.txt:");
        // read from the file until there's nothing else in it:
        while (myFile.available()) {
            Serial.write(myFile.read());
        }
        myFile.close();    // close the file:
    } else {
        Serial.println("error opening test.txt");    // if the file didn't open, print an error:
    }
}
////////////////////////////////////////////////////////
void loop() {
    // nothing happens after setup
}
```

3.2、Connections diagram



3.3、Compile and upload

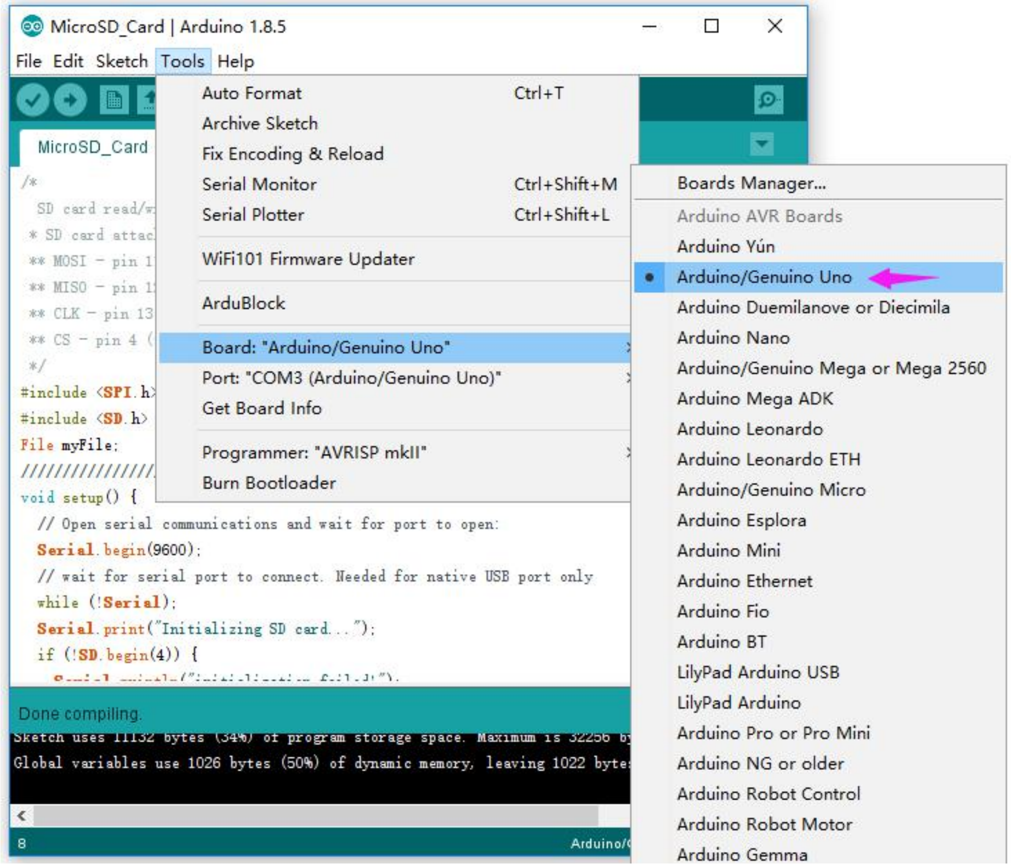
3.3.1、Using USB cable to connect computer to UNO R3 board，Open the Arduino IDE，copy the above code into the IDE：



```
/*
  SD card read/write
 * SD card attached to SPI bus as follows:
 ** MOSI - pin 11
 ** MISO - pin 12
 ** CLK - pin 13
 ** CS - pin 4 (for MKRZero SD: SDCARD_SS_PIN)
 */
#include <SPI.h>
#include <SD.h>
File myFile;
/////////////////////////////////////////////////////////
void setup() {
  // Open serial communications and wait for port to open:
  Serial.begin(9600);
  // wait for serial port to connect. Needed for native USB port only
  while (!Serial);
  Serial.print("Initializing SD card...");
  if (!SD.begin(4)) {
```

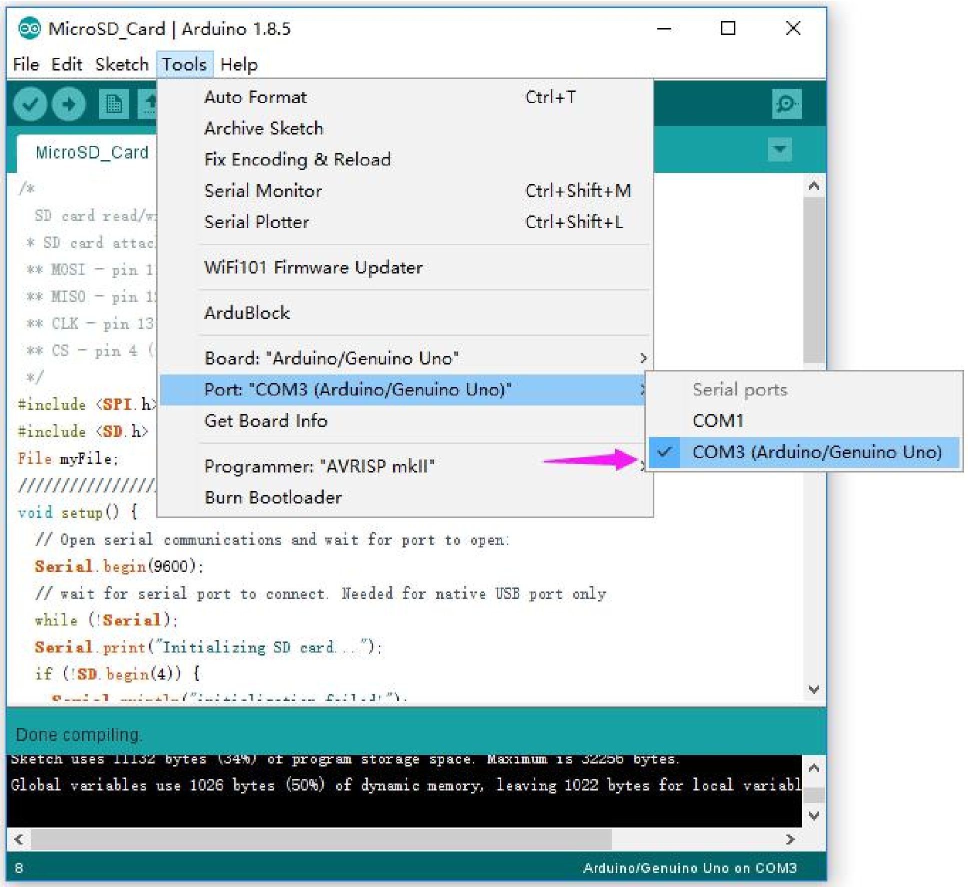
```
Done compiling.
Sketch uses 11132 bytes (34%) of program storage space. Maximum is 32256 bytes.
Global variables use 1026 bytes (50%) of dynamic memory, leaving 1022 bytes for local variabl
```

3.3.2、select corresponding board type



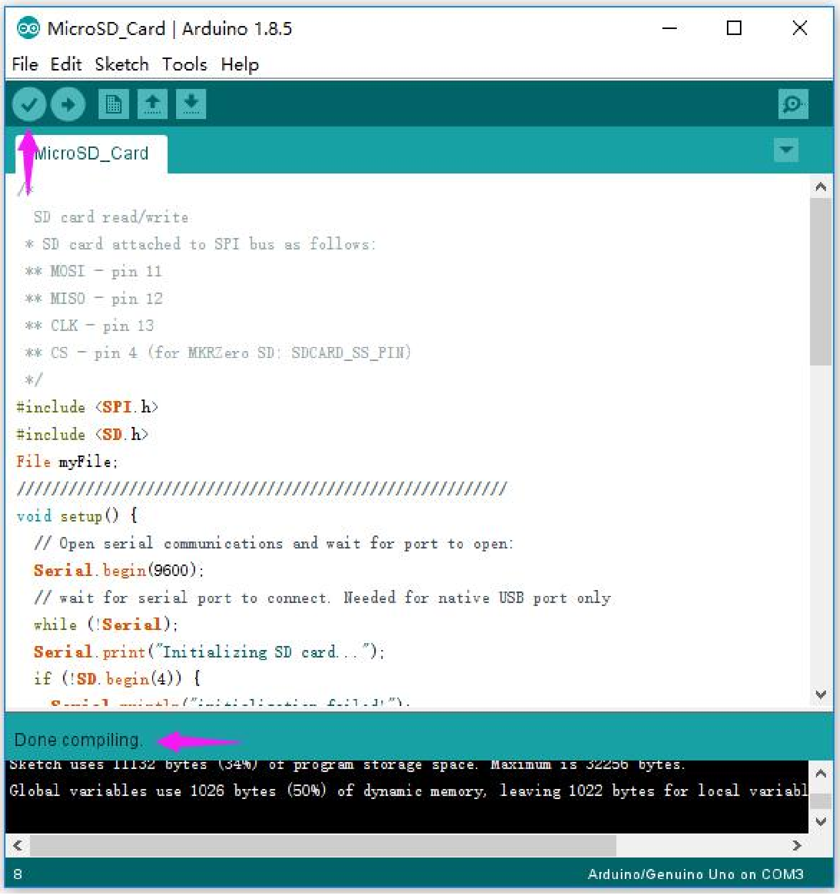
Board: "Arduino/Genuino Uno"
Port: "COM3 (Arduino/Genuino Uno)"
Programmer: "AVRISP mkII"

3.3.3、select corresponding port


MicroSD_Card | Arduino 1.8.5
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 Firmware Updater
ArduBlock
Board: "Arduino/Genuino Uno"
Port: "COM3 (Arduino/Genuino Uno)"
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
Serial ports
COM1
COM3 (Arduino/Genuino Uno)
MicroSD_Card
Arduino/Genuino Uno on COM3

```
Done compiling.
Sketch uses 11132 bytes (34%) of program storage space. Maximum is 32256 bytes.
Global variables use 1026 bytes (50%) of dynamic memory, leaving 1022 bytes for local variabl
```

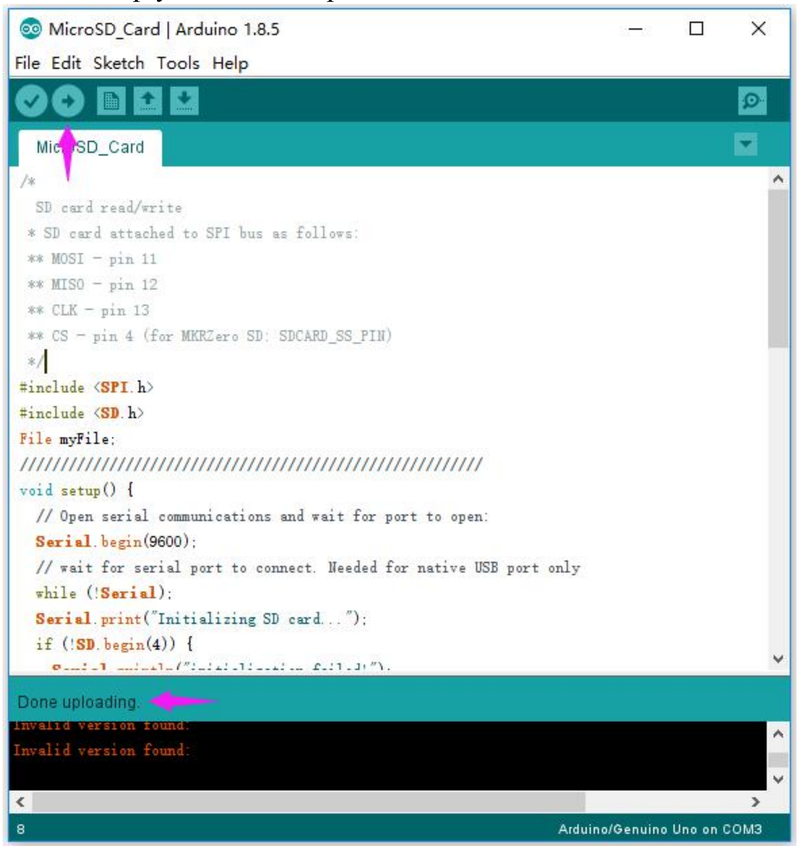
3.3.4、compile this sketch


MicroSD_Card | Arduino 1.8.5
File Edit Sketch Tools Help
MicroSD_Card
Arduino/Genuino Uno on COM3

```
/*
  SD card read/write
 * SD card attached to SPI bus as follows:
 ** MOSI - pin 11
 ** MISO - pin 12
 ** CLK - pin 13
 ** CS - pin 4 (for MKRZero SD: SDCARD_SS_PIN)
 */
#include <SPI.h>
#include <SD.h>
File myFile;
////////////////////////////////////////////////////////
void setup() {
  // Open serial communications and wait for port to open:
  Serial.begin(9600);
  // wait for serial port to connect. Needed for native USB port only
  while (!Serial);
  Serial.print("Initializing SD card...");
  if (!SD.begin(4)) {
```

```
Done compiling.
Sketch uses 11132 bytes (34%) of program storage space. Maximum is 32256 bytes.
Global variables use 1026 bytes (50%) of dynamic memory, leaving 1022 bytes for local variabl
```

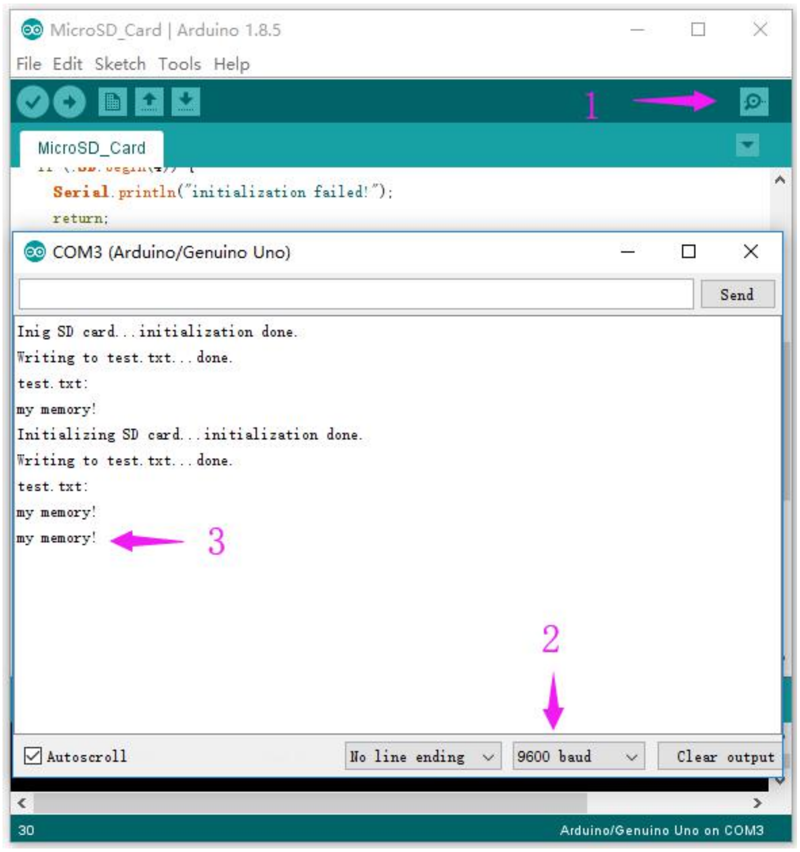
3.3.5、simply click the “Upload” button in the environment



```
/*
  SD card read/write
 * SD card attached to SPI bus as follows:
 ** MOSI - pin 11
 ** MISO - pin 12
 ** CLK - pin 13
 ** CS - pin 4 (for MKRZero SD: SDCARD_SS_PIN)
 */
#include <SPI.h>
#include <SD.h>
File myFile;
////////////////////////////////////////////////////////
void setup() {
  // Open serial communications and wait for port to open:
  Serial.begin(9600);
  // wait for serial port to connect. Needed for native USB port only
  while (!Serial);
  Serial.print("Initializing SD card...");
  if (!SD.begin(4)) {
```

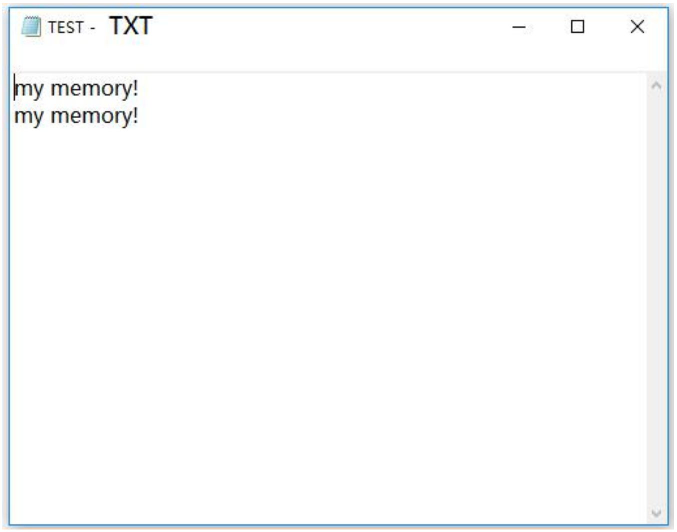
```
Done uploading.
invalid version found:
Invalid version found:
```

3.3.6、Open the IDE string monitor, the contents of the SD card will be printed; use the card reader to open the txt file named "TEST" in the micro SD card, which will display the record written by the IDE to the SD card, as shown below:


MicroSD_Card | Arduino 1.8.5
File Edit Sketch Tools Help
1
MicroSD_Card
COM3 (Arduino/Genuino Uno)
Send
3
2
Autoscroll
No line ending
9600 baud
Clear output
30
Arduino/Genuino Uno on COM3

```
    Serial.println("initialization failed!");
    return;
```

```
Inig SD card...initialization done.
Writing to test.txt...done.
test.txt:
my memory!
Initializing SD card...initialization done.
Writing to test.txt...done.
test.txt:
my memory!
my memory!
```



```
my memory!
my memory!
```