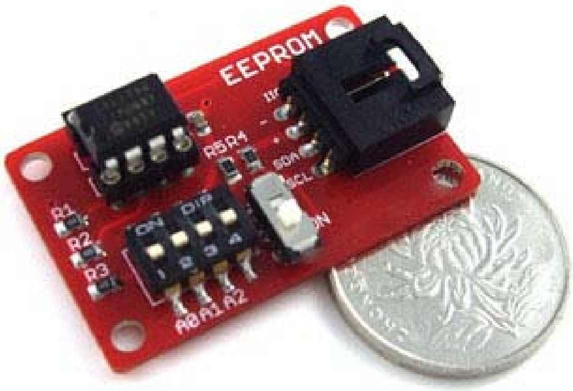
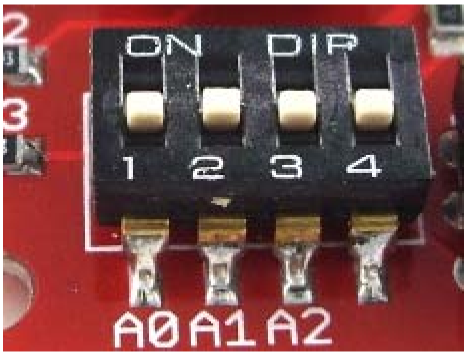
Arduino EEPROM module 512K for Sensor Shield

Experiment Steps

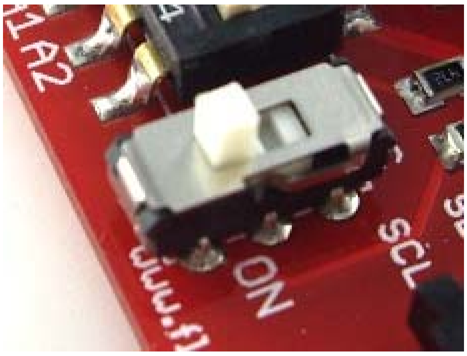
This is a new designed for small data size storage. It can help to extend the EEPROM storage of Arduino. This module uses I2C to connect to Arduino, and the EEPROM chip is pluggable for future expansion.



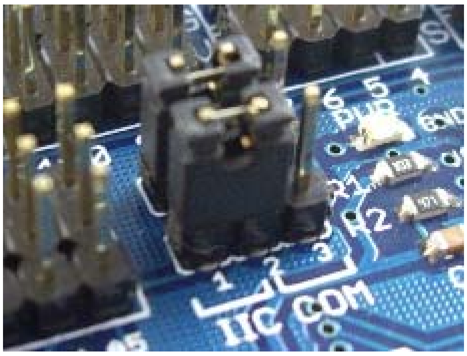
The storage module is based on AT24C series of EEPROM chips, which the I2C base address is 0×50, and the last three addresses can be set according to need. So we should set the last three address when we firstly use, it is through the A2, A1 and A0 of the four position DIP switch. Dial the DIP switch to up,the corresponding is 1; Dial the DIP switch to down,the corresponding is 0. Namely,if move the A2, A0, A1 to up, the corresponding address is 0×57, But if move the A2, A0, A1 to down, the corresponding address is 0×50.



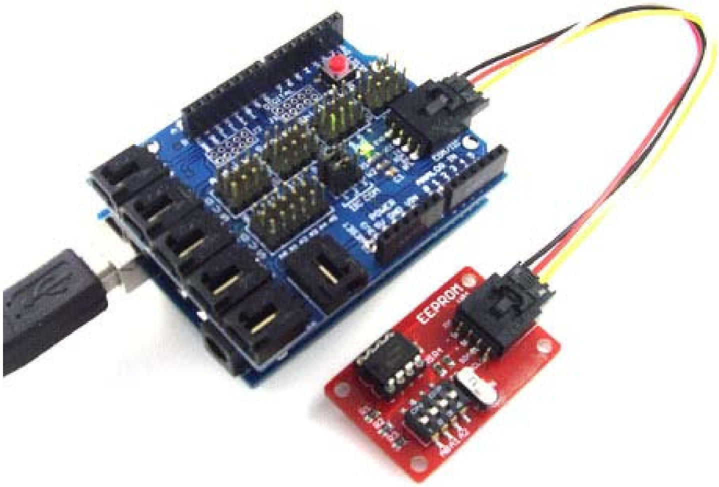
The other place which need to be set is switch RS of this storage module, it is used to set that whether connect the SDA and SCl of I2C bus line with pull-up resistors. We know that the I2C is called bus line, just because it can connect multiple the I2C equipment, according to the provisions of this agreement, the I2C allowed that we only be permitted to connect pull-up resistors with one I2C device that is the nearest from controllor. Here you may already know, if there are multiple this storage module Arduino connected to the I2C bus line, we must set the RS switch of storage module of the nearest from Arduino to the position "ON", and the other RS of other storage module can not be set to ON location:



When IIC connection is used, don't forget to set the jumper on Sensor Shield to IIC, as follows.

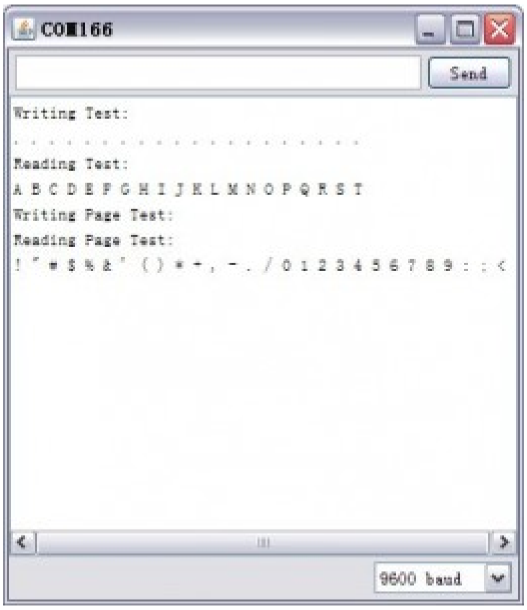


Then connect the module to Arduino Sensor Shield by the provided cable.



Demo Code

```
#include <Wire.h>
#define EEPROM_ADDR 0x50           // I2C Buss address of 24LC256 256K EEPROM

void setup()
{
  Wire.begin();                        // join I2C bus (address optional for master)
  Serial.begin(9600);

  // TESTS FOR EACH FUNCTION BEGIN HERE
  Serial.println("Writing Test:");
  for (int i=0; i<20; i++){            // loop for first 20 slots
    i2c_eeprom_write_byte(EEPROM_ADDR,i,i+65);   // write address + 65 A or 97 a
    Serial.print(". ");
    delay(10);                         // NEED THIS DELAY!
  }
  Serial.println("");
  delay(500);

  Serial.println("Reading Test:");
  for (int i=0; i<20; i++){            // loop for first 20 slots
    Serial.print(i2c_eeprom_read_byte(EEPROM_ADDR, i),BYTE);
    Serial.print(" ");
  }

  // setup for page tests . . .
  byte PageData[30];                   // array that will hold test data for a page
  byte PageRead[30];                   // array that will hold result of data for a page
  for (int i=0; i<30; i++){            // zero both arrays for next test
    PageData[i] = 0;
    PageRead[i] = 0;
  }
  Serial.println("");
  for (int i=0; i<30; i++) PageData[i] = i+33;  // fill up array for next test char 33 = !

  Serial.println("Writing Page Test:");
  i2c_eeprom_write_page(EEPROM_ADDR, 100, PageData, 28 ); // 28 bytes/page is max
  Serial.println("Reading Page Test:");
  i2c_eeprom_read_buffer( EEPROM_ADDR, 100, PageRead, 28);
  for (int i=0; i<28; i++){
    Serial.print(PageRead[i],BYTE);    // display the array read
    Serial.print(" ");
  }
}

void loop()
{
}

void i2c_eeprom_write_byte( int deviceaddress, unsigned int eeaddress, byte data )
{
  int rdata = data;
  Wire.beginTransmission(deviceaddress);
  Wire.send((int)(eeaddress >> 8));    // Address High Byte
  Wire.send((int)(eeaddress & 0xFF));  // Address Low Byte
  Wire.send(rdata);
  Wire.endTransmission();
}

// Address is a page address, 6-bit (63). More and end will wrap around
// But data can be maximum of 28 bytes, because the Wire library has a buffer of 32 bytes
void i2c_eeprom_write_page( int deviceaddress, unsigned int eeaddresspage, byte* data, byte length )
{
  Wire.beginTransmission(deviceaddress);
  Wire.send((int)(eeaddresspage >> 8)); // Address High Byte
  Wire.send((int)(eeaddresspage & 0xFF)); // Address Low Byte
  byte c;
  for ( c = 0; c < length; c++)
    Wire.send(data[c]);
  Wire.endTransmission();
  delay(10);                           // need some delay
}

byte i2c_eeprom_read_byte( int deviceaddress, unsigned int eeaddress )
{
  byte rdata = 0xFF;
  Wire.beginTransmission(deviceaddress);
  Wire.send((int)(eeaddress >> 8));    // Address High Byte
  Wire.send((int)(eeaddress & 0xFF));  // Address Low Byte
  Wire.endTransmission();
  Wire.requestFrom(deviceaddress,1);
  if (Wire.available()) rdata = Wire.receive();
  return rdata;
}

// should not read more than 28 bytes at a time!
void i2c_eeprom_read_buffer( int deviceaddress, unsigned int eeaddress, byte *buffer, int length )
{
  Wire.beginTransmission(deviceaddress);
  Wire.send((int)(eeaddress >> 8));    // Address High Byte
  Wire.send((int)(eeaddress & 0xFF));  // Address Low Byte
  Wire.endTransmission();
  Wire.requestFrom(deviceaddress,length);
  //int c = 0;
  for ( int c = 0; c < length; c++ )
    if (Wire.available()) buffer[c] = Wire.receive();
}
```



```
Writing Test:
. . . . . . . . . . . . . . . . . . . .
Reading Test:
A B C D E F G H I J K L M N O P Q R S T
Writing Page Test:
Reading Page Test:
! " # $ % & ' ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; <
```

Output as shown on serial window