Part 2

lesson 25

EEPROM

Overview

EEPROM (Electrically Erasable Programmable Read-Only Memory)，a memory chip that can keep data from interrupting power supply. In short, if you want Arduino to save some parameters after power off, use EEPROM.

AVR chips on various types of Arduino controllers are equipped with EEPROM, as well as external EEPROM chips. EEPROM sizes of common Arduino controllers are as follows:

The EEPROM of Arduino UNO is 1K.
The EEPROM of Arduino 2560 is 4K

Component Required:

(1) x Elegoo Uno R3

Because of the use of on-chip EEPROM, there is no need to connect wires in this course.

Code

- Please open the program in the code folder- EEPROM and click UPLOAD to upload the program. See Lesson 5 of part 1 for details about program uploading if there are any errors.
- Arduino UNO's EEPROM can only be erased 100,000 times. So you' d better not put the code that implements the function of reading and writing data in loop().
- In Arduino EEPROM library, the address of EEPROM starts at 0, and each address can store 1 Byte data. So when the data is larger than 1 Byte, it needs to read and write byte by byte.
- The EEPROM of Arduino UNO and Arduino Leonardo has a storage space of 1 KB = 1024, and the corresponding address is 0 ~ 1023.
- The EEPROM of Arduino Mega2560 has 4 KB = 4096B storage space, and the corresponding address is 0 ~ 4095.
- EEPROM.write(address,value);
  Function: Write data to specified address;
- Parameters:
  Address: EEPROM address, starting address is 0
  in UNO, the range is 0~1023
  in MEGA2560, the range is 0~4096
- Value: data, 'byte ' or 'char' type, the range is 0~255
  which means that if the incoming data is larger than this range, it will be cut off.
- EEPROM. write takes up 3 ms at a time. If the program keeps erasing EEPROM, it won't take long to damage EEPROM. Take care not to erase it frequently. When it is really needed, please consider it carefully, add any delay and so on.

See Demo1 in the code for details:

```
char company[7] = {"elegoo"};
 char company2[7] = {"0"};

 for(int i=0 ; i<6 ;i++ )
 {
 EEPROM.write( i,company[i]);
 }

  for(int i=0 ; i<6 ;i++ )
 {
  company2[i] = EEPROM.read(i);
 }
```

- EEPROM.read(address);
  Function: Read data from a specified address. Read 1B data at a time. If the specified address has no data, the read-out value is 255.

- Grammar: EEPROM. read (address);
  Parameters:
  Address: EEPROM address, starting address is 0;
  Return value: byte type, returns the data stored at the specified address;

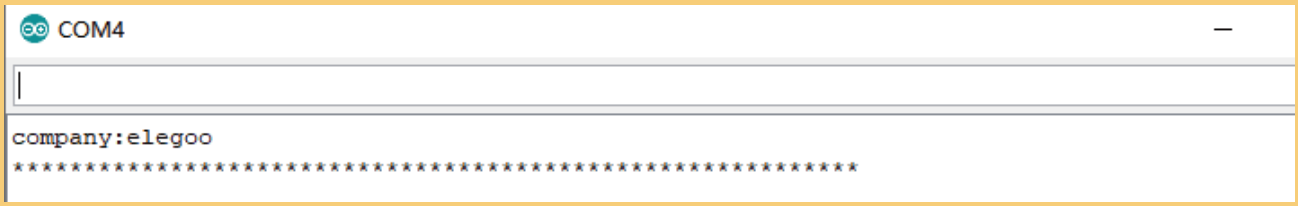
- After you download the program, the data will be saved to eeprom. Then you can try to comment out the program about EEPROM. Write. Download the program again and you can find that the data you read from eeprom is the same as the data you saved before, even you reset the Arduino board.
  This realizes saving the data when the board is turned off.

```
void test1 (void)
{
 char company[7] = {"elegoo"};
 char company2[7] = {"0"};

// for(int i=0 ; i<6 ;i++ )
// {
// EEPROM.write(i,company[i]);
// }

  for(int i=0 ; i<6 ;i++ )
 {
  company2[i] = EEPROM.read(i);
 } }
```

- Click the Serial Monitor button to turn on the serial monitor. The basics about the serial monitor are introduced in details in Lesson 4 of part 2.



- The address of EEPROM starts at 0, and each address can store 1B data.
- **EEPROM. write** and **EEPROM. read** can only operate on single-byte data (such as char, byte type data)

- If the incoming data (such as int, float, etc.) is larger than 1B, it will be truncated, so when the data is larger than 1B, it needs to be read and written byte by byte.
- So next we'll write EEPROM read and write functions for all types.
- In this example, we used the function template to apply to various data types.
- **See demo2 in code for details.**

```
template<typename T>
void EEPROM_write(T data1,int address)
{
union data{
T my_type;
char charbuf[];
 } data2;
data2.my_type=data1;
for(int i=address ; i<sizeof(data1) ;i++ )
 {
 EEPROM.write(i , data2.charbuf[i]);
  }
  }
```

template<typename T>

- The template header <> can be either an undetermined data type defined by typename or an explicit data type.
- 'T' is the generic name of the category. The data types you use are int, char, float, double and so on.
- The format above is the function template. union:Several different types of variables are stored in the same memory unit.
  That is to say, using coverage technology, several variables cover each other. This structure, in which several different variables occupy a section of memory together, is called union.

```
union data{
T my_type;
char charbuf[];
} data2;
```

implementation of EEPROM_write

- Based on the nature of the union, we store the data to be written to EEPROM in the same type of union member''my_type'. Since "my_type" and "charbuf[]" occupy the same memory, so we write an array of char type "charbuf" in bytes to eeprom, which is equivalent to writing'my_type'to eeprom, and EEPROM_read can be obtained equally.