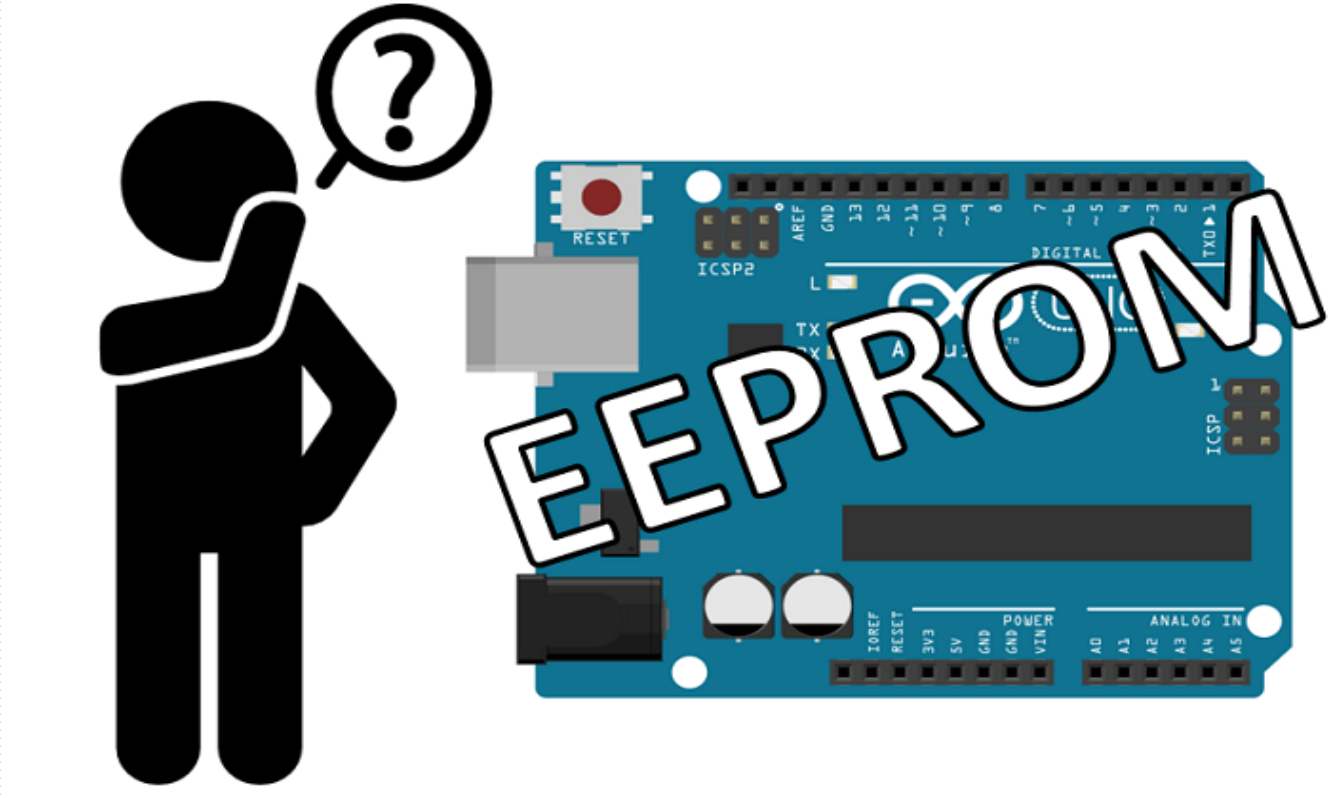
CSE423: Embedded System Summer-2020

Storing data in EEPROM



Todays Lecture

- *Understanding EEPROM*
- *When EEPROM is applicable?*
- *Data size*
- *How to write on EEPROM*
- *How to read from EEPROM*

Storing data in EEPROM

An EEPROM is an Electrically Erasable Programmable Read-Only Memory.

It is a form of non-volatile memory that can remember things with the power being turned off, or after resetting the Arduino. The beauty of this kind of memory is that we can store data generated within a sketch on a more permanent basis.

Why EEPROM to store data?

For situations where data that is unique to a situation needs a more permanent home.

- Storing unique serial number /date for commercial purpose (a function of the sketch could display the serial number on an LCD).
- You may need to count certain events and not allow the user to reset them – such as an odometer or operation cycle-counter.

Storing data in EEPROM

What sort of data can be stored?

Anything that can be represented as *bytes* of data.

Now each digit in that binary number uses one ‘bit’ of memory, and eight bits make a byte. Due to internal limitations of the microcontrollers in our Arduino boards, we can only store 8-bit numbers (one byte) in the EEPROM.

This limits the decimal value of the number to fall between zero and 255.

How to Store data in EEPROM

Step-1: To use the EEPROM, a library is required, so use the following library in your sketches:

```
#include "EEPROM.h“
```

The rest is very simple. To store a piece of data, we use the following function:

```
EEPROM.write(a,b);
```

The parameter a is the position in the EEPROM to store the integer (0~255) of data b. In this example, we have 1024 bytes of memory storage, so the value of a is between 0 and 1023.

How to Store data in EEPROM

Step-2:To retrieve a piece of data is equally as simple, use:

```
z = EEPROM.read(a);
```

Where *z* is an integer to store the data from the EEPROM position *a*.

Sample **Example** to store data in EEPROM

This sketch will create random numbers between 0 and 255, store them in the EEPROM, then retrieve and display them on the serial monitor.

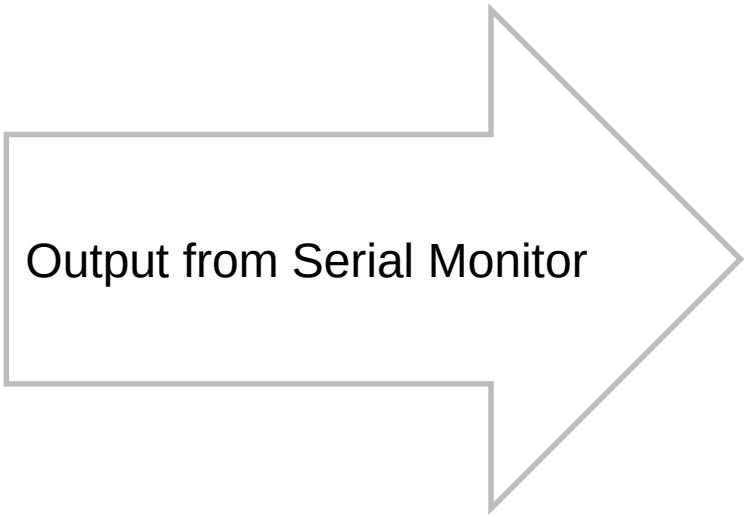
Sample Example to store data in EEPROM

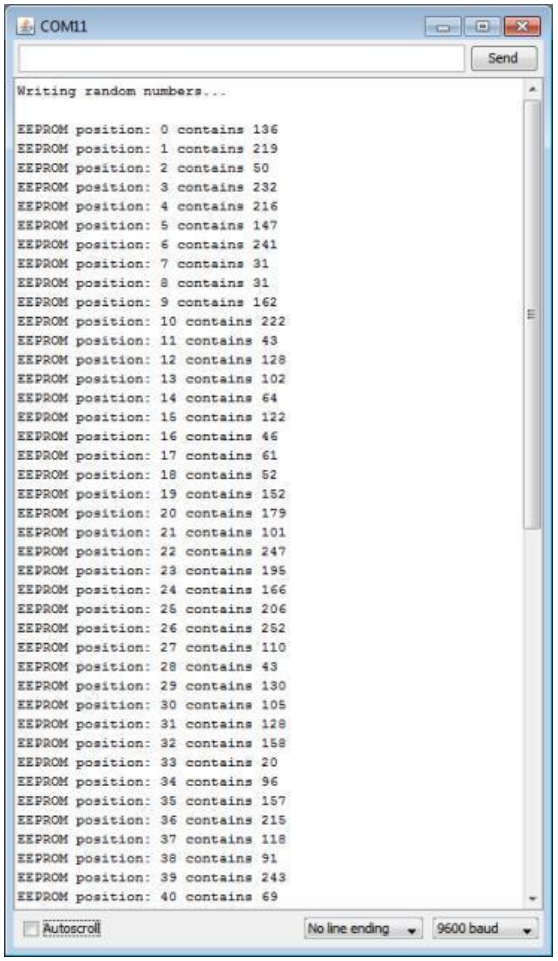

```
// Arduino internal EEPROM demonstration

#include <EEPROM.h>
int zz;
int EEsize = 1024; // size in bytes of your board's EEPROM

void setup()
{
  Serial.begin(9600);
  randomSeed(analogRead(0));
}
void loop()
{
  Serial.println("Writing random numbers...");
  for (int i = 0; i < EEsize; i++)
  {
    zz=random(255);
    EEPROM.write(i, zz);
  }
  Serial.println();
  for (int a=0; a<EEsize; a++)
  {
    zz = EEPROM.read(a);
    Serial.print("EEPROM position: ");
    Serial.print(a);
    Serial.print(" contains ");
    Serial.println(zz);
    delay(25);
  }
}
```

Sample Example to store data in EEPROM





Storing data in EEPROM

- What about replacing an existing data with a new one?

The EEPROM.update( ) function is particularly useful. It only writes on the EEPROM if the value is written is different from the one already saved.

As the EEPROM has limited life expectancy due to limited write/erase cycles, using the EEPROM.update() function instead of the EEPROM.write() saves cycles.

Syntax: EEPROM.update(address, value);

Task

- Try the described example mentioned earlier and see the output!