

The output message from the Arduino IDE terminal provides information about the memory usage of your sketch (program) after it has been compiled and uploaded to the Arduino board. Here's a breakdown of what each part means:

1. Sketch uses 7096 bytes (21%) of program storage space. Maximum is 32256 bytes.

- Sketch uses 7096 bytes: This indicates that your compiled program (sketch) is occupying 7096 bytes of the flash memory (program storage space) on the Arduino board.

- (21%): This is the percentage of the total program storage space that your sketch is using.

- Maximum is 32256 bytes: This is the total amount **of flash memory** available for storing your program on the Arduino board. For an Arduino Uno, for example, this is typically 32,256 bytes.

2. Global variables use 534 bytes (26%) of dynamic memory, leaving 1514 bytes for local variables. Maximum is 2048 bytes.

- Global variables use 534 bytes: This indicates that your global variables are using 534 bytes of SRAM (dynamic memory).

- (26%): This is the percentage of the total dynamic memory that is being used by your global variables.

- Leaving 1514 bytes for local variables: This is the amount of **SRAM remaining** for use by local variables, stack, and heap during the execution of your program.

- Maximum is 2048 bytes: This is the total amount of SRAM available on the Arduino board. For an Arduino Uno, this is typically 2048 bytes (2 KB).

Flash memory and SRAM are two different types of memory used in microcontrollers, including Arduino boards. They serve different purposes and have distinct characteristics:

**Flash Memory**

1. **Purpose**:
   * Flash memory is used to store the program code (sketch) and constants. It is *non-volatile*, meaning it retains its data even when the power is turned off.
2. **Characteristics**:
   * **Non-volatile**: Data is retained without power.
   * **Write cycles**: Limited number of write/erase cycles (typically around 10,000 to 100,000 cycles).
   * **Speed**: Generally slower in terms of read and write operations compared to SRAM.
   * **Size**: Typically larger in capacity than SRAM. For example, an Arduino Uno has 32 KB of flash memory.
3. **Usage**:
   * Used to store the program code and any constant data that does not change during program execution.

**SRAM (Static Random Access Memory)**

1. **Purpose**:
   * SRAM is used for storing variables, the stack, and dynamically allocated data (heap) during program execution. It is volatile, meaning it loses its data when power is turned off.
2. **Characteristics**:
   * **Volatile**: Data is lost when power is turned off.
   * **Write cycles**: Unlimited number of write cycles.
   * **Speed**: Faster read and write operations compared to flash memory.
   * **Size**: Typically, smaller in capacity than flash memory. For example, an Arduino Uno has 2 KB of SRAM.
3. **Usage**:
   * Used for storing variables, the call stack, and temporary data that changes during program execution.

**Key Differences**

1. **Volatility**:
   * Flash memory is non-volatile; SRAM is volatile.
2. **Purpose**:
   * Flash memory is used for storing the program code and constants; SRAM is used for storing variables and dynamic data.
3. **Speed**:
   * Flash memory is slower; SRAM is faster.
4. **Capacity**:
   * Flash memory generally has a larger capacity; SRAM has a smaller capacity.
5. **Write Cycles**:
   * Flash memory has a limited number of write cycles; SRAM has unlimited write cycles.

**Example in Arduino Context**

* **Flash Memory**: When you upload a sketch to an Arduino, the code is stored in the flash memory.
* **SRAM**: When the sketch runs, any variables, the stack, and dynamic data are stored in the SRAM.

Understanding these differences helps in optimizing the memory usage of your programs and avoiding memory-related issues.