Super Loop vs Multi-threaded Application

1. Super loop architecture also known as “bare-metal programming”. Which generally execute in a round-robin fashion? There is no operating system, and the structure is fairly simple.   
  
2. In your main () function, you set up any variables, drivers, libraries, etc. and then perform one or more periodic tasks in a while (true) loop.  
  
3. There is nothing wrong with the super loop application. In fact, it’s still one of the most popular ways to program a MCU, as it’s easy to implement and easy to debug.   
  
4. You can even add in interrupts by using interrupt service routines (ISRs) that cause the program to halt and execute some arbitrary code when an external event occurs (such as a timer is expiring or a button being pushed).  
  
5. If you have multiple tasks to accomplish during the main loop, you generally execute them in a round-robin fashion.   
  
6. The problem comes in when you start adding so many tasks that some start missing deadlines or preventing other features from working.

This is where an RTOS can help.   
  
1. RTOS provides functions like multithreading, inter-process communication and memory handling to embedded applications.  
  
2. Rather than execute everything in a round-robin fashion, you can essentially execute everything concurrently.  
  
3. If you need multi-threading support, an RTOS also has several other benefits as well. like it allows you to modularize your code more easily, as tasks can be written separately.   
  
4. If you are working on a team, individual tasks can be assigned to different team members to allow concurrent development. Note that there will still be debugging at the end to ensure everything is integrated correctly.

