Multithreading Management

A/Prof Shiyang Tang

MTRN3500

Computing Applications in Mechatronics Systems

Why Thread Management?

1. **Which thread starts first?** – Some subsystems must initialise before others.
Example: GNSS thread may need to lock satellites before Drive Control can use position data.

2. **Which thread terminates first?** – Controlled shutdown prevents corruption or unsafe states.
Example: Stop Drive Control first (so the vehicle halts), then shut down GNSS and LiDAR threads.

3. **Custom startup sequence** – Certain threads depend on others being ready.
Example: LiDAR and GNSS should be running before Drive Control starts moving the UGV.

4. **Ensuring all threads are operating** – Use thread monitoring or periodic “heartbeat” signals.
Example: If the GNSS thread stops updating for 5 seconds, Drive Control should be alerted.

5. **Handling thread failures** – Decide whether to restart only the failed thread or shut down the whole system.
Example: If the GNSS thread fails, the UGV might restart GNSS automatically, but if LiDAR fails, it may be safer to stop all threads and shut down.

Thread Management Thread (TMT)

1. **Centralised Control of Startup and Shutdown** – TMT enforces the correct order of starting and stopping threads.

2. **Health Monitoring (“Heartbeat” Checks)** – TMT regularly checks if each subsystem thread is alive and responsive.

3. **Failure Handling and Recovery** – TMT decides what happens when a thread crashes.

4. **Resource Coordination** – Prevents multiple threads from overloading the processor or memory.

5. **Scalability and Maintainability** – Easier to add or remove threads without redesigning the whole system.

6. **System Safety and Reliability** – Provides a single authority to keep the system stable under abnormal conditions.

Thread Data (GNSS)

- **Time Stamp**: 2025-09-17 12:30:45.235 UTC
- **Latitude:** –33.9173°
- **Longitude:** 151.2311°
- **Northing (m):** 6254321.45

  The distance (in meters) northwards from the equator (for the northern hemisphere) or from a *false origin* (for the southern hemisphere).
- **Easting (m):** 334512.78

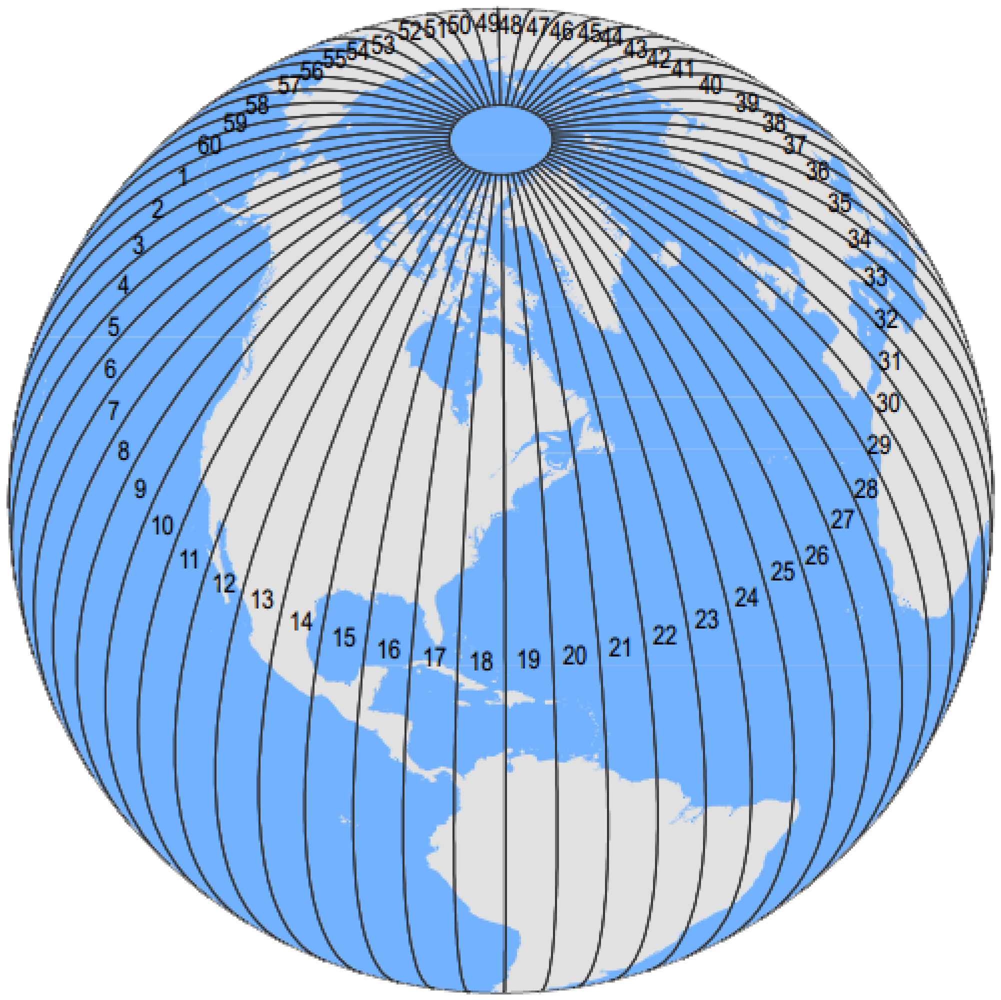
  The distance eastwards from the central meridian (false origin) of the Universal Transverse Mercator (UTM) zone.
- **Height/Altitude (m):** 45.23
- **Speed (m/s):** 1.52
- **Heading (degrees from North):** 87.4°
- **HDOP (Horizontal Dilution of Precision):** 0.7
- **Number of Satellites:** 14

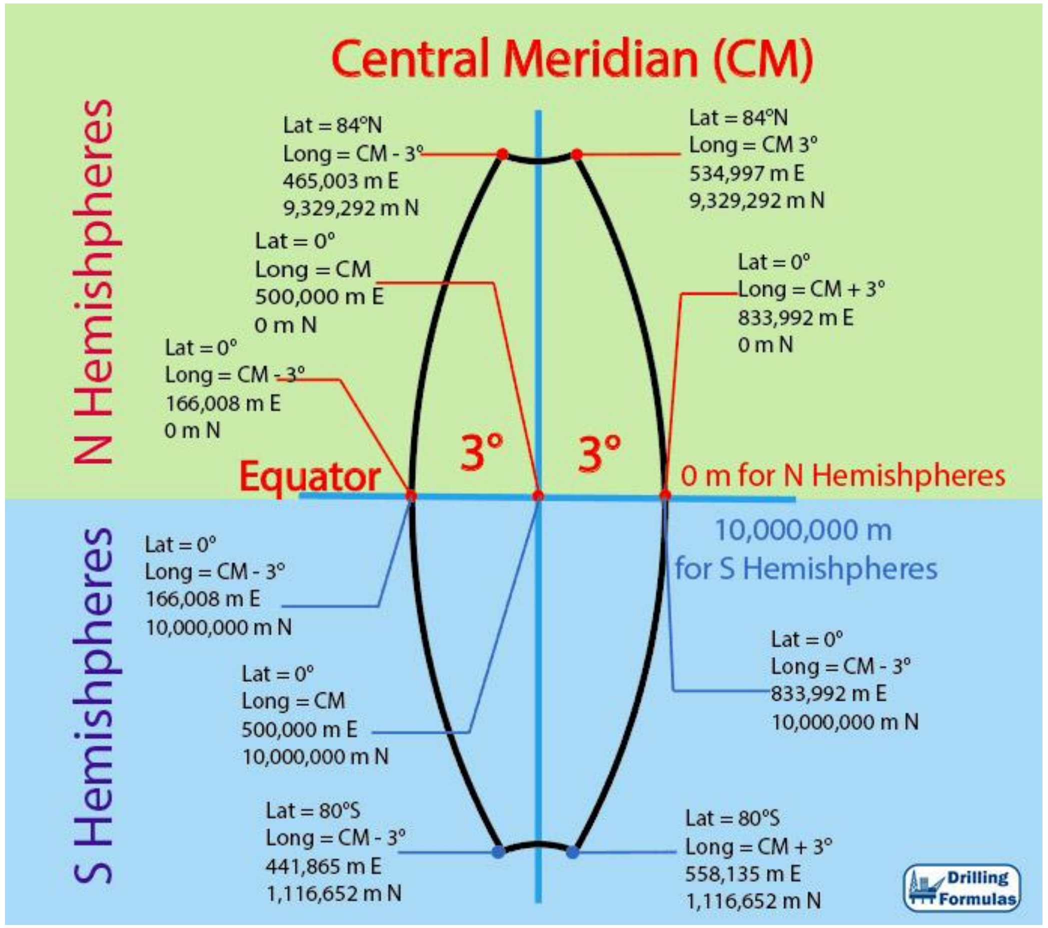
The **GNSS thread** generates this data continuously (e.g., 10 Hz).

Other threads (e.g., Drive Control) read **Northing, Easting, Heading, and Speed** to plan motion.

Thread Data (GNSS)

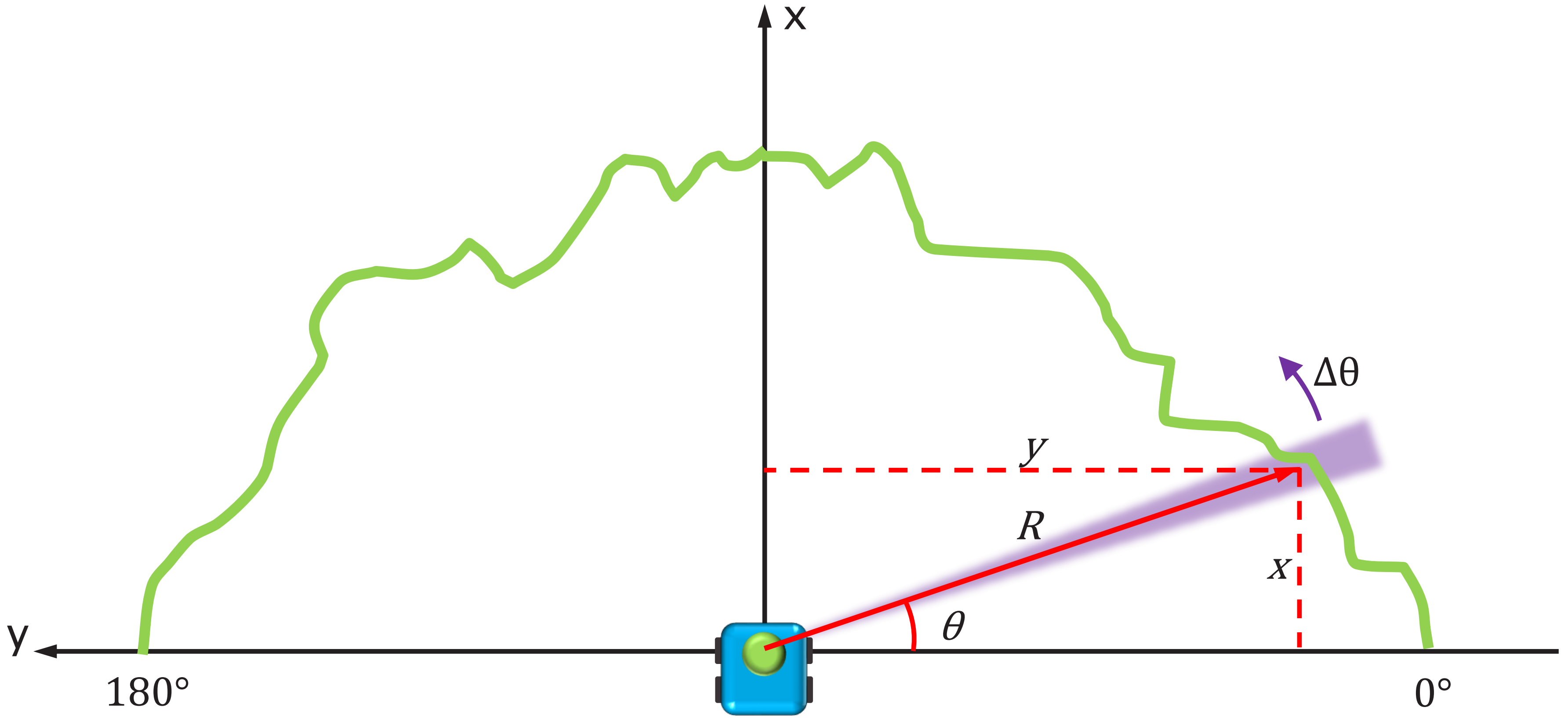
- **Northing:** The distance (in meters) northwards from the equator (for the northern hemisphere) or from a false origin (for the southern hemisphere).
- **Easting:** The distance eastwards from the central meridian of the UTM zone.





https://www.drillingformulas.com/universal-transverse-mercator-application-in-directional-drilling/

Thread Data (LiDAR)



Δθ (angular resolution) = 0.5°
End angle = 180° ➡ 361 data points/scan

$x = R \sin\theta$
$y = R \cos\theta$ ➡

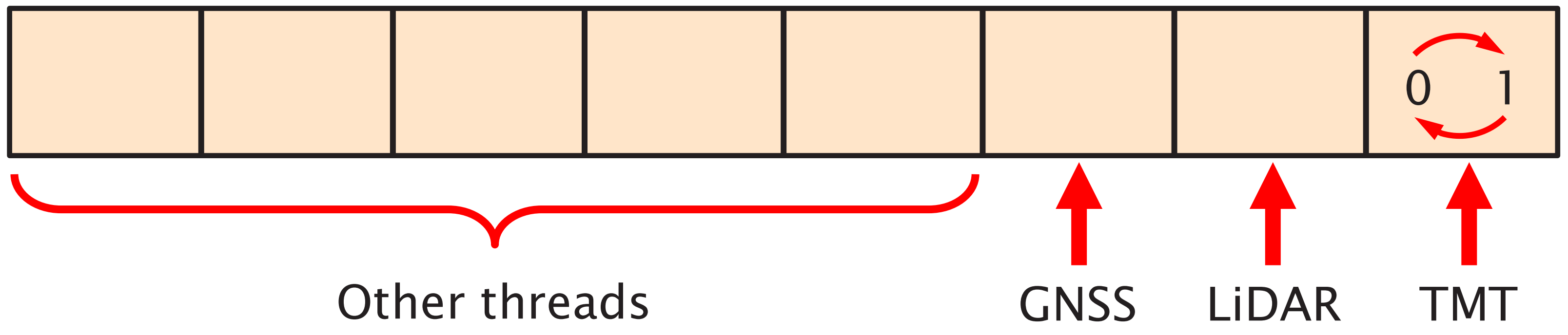
```
array<double>^ x;
array<double>^ y;
```

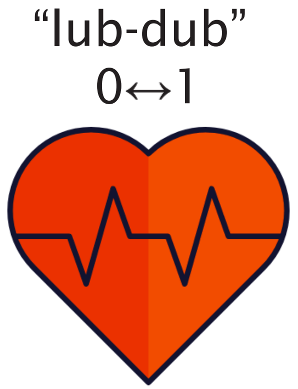
➡ LiDAR data

Thread Data (TMT)

- A **heartbeat** is a small, periodic message sent by each thread to the TMT to signal that it is still alive and running correctly.
- Think of it like a patient's heart monitor: if the beat stops, something is wrong.
- The message can be very small – in our example, just 1 byte per thread.

```
unit8_t heartbeat;
```

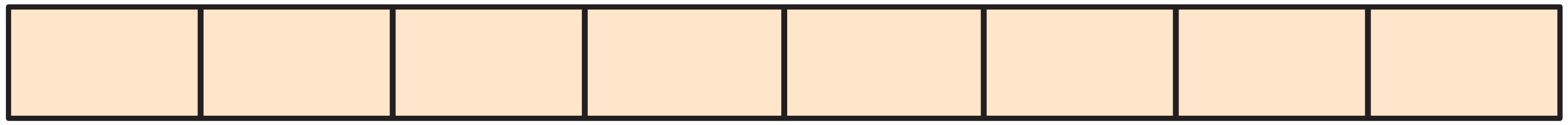




Threads may **crash silently** or become **stuck** in an infinite loop. The TMT cannot just assume all threads are alive. By monitoring regular heartbeats, the TMT can:

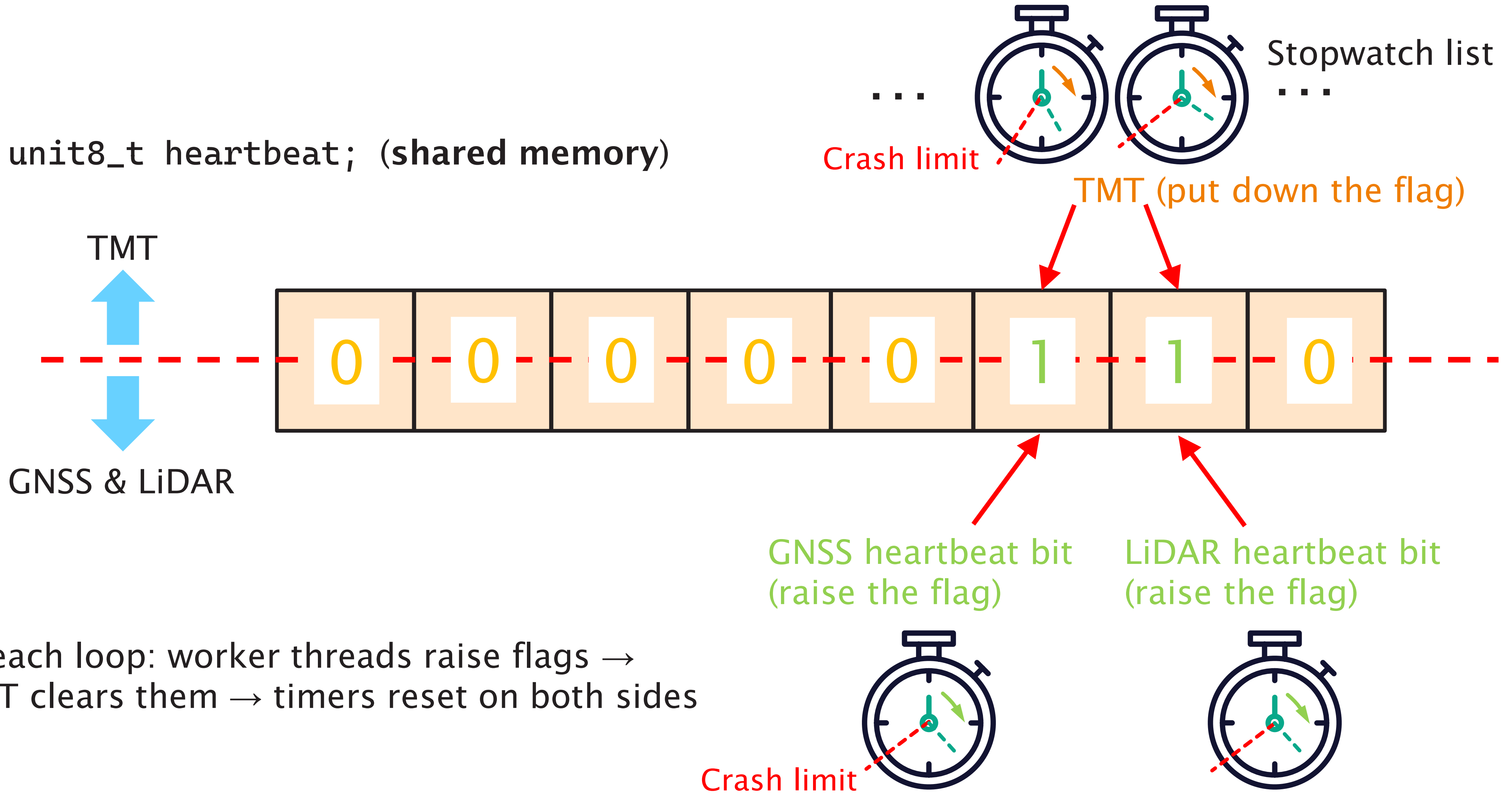
- Detect thread failure early.
- Decide whether to restart the failed thread or **shut down** the whole system.

```
unit8_t shutdown;
```



How Does Heartbeat Work?

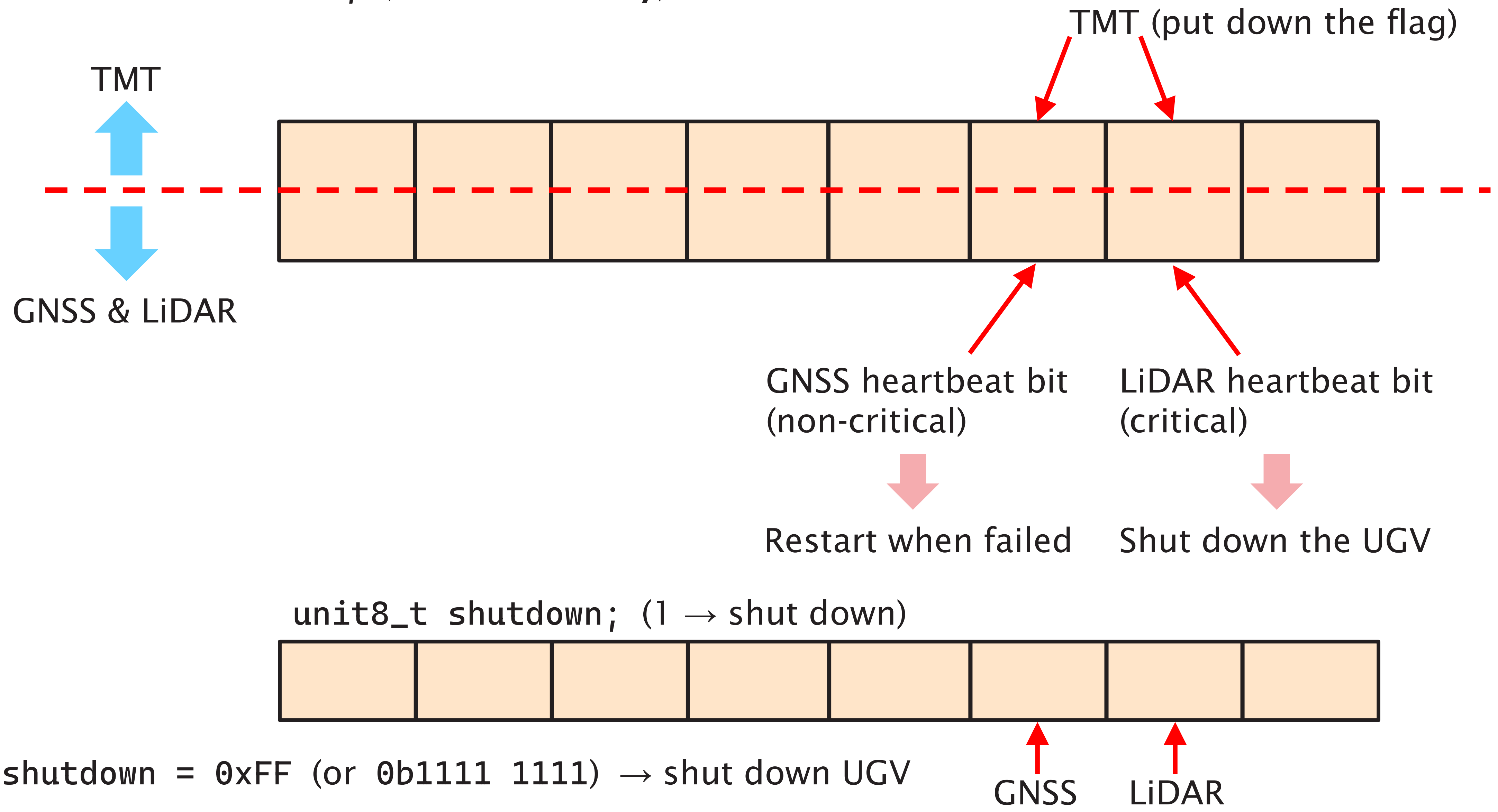
`unit8_t heartbeat;` (**shared memory**)



In each loop: worker threads raise flags → TMT clears them → timers reset on both sides

Critical & Non-Critical Threads

`unit8_t heartbeat;` (**shared memory**)



`unit8_t shutdown;` (1 → shut down)

`shutdown = 0xFF` (or `0b1111 1111`) → shut down UGV

Appendix 1: enum (Enumeration)

An **enumeration** (enum) is just a way to give names to integer values.

```
enum Color { Red, Green, Blue }; // auto: Red=0, Green=1, Blue=2

int main()
{
    Color c = Green;            // this is just the integer 1 with a nicer name
    Console::WriteLine(Green);    // prints: 1
}
```

Using an **enum** (`error_state`) function so we can report which outcome happened, not just "ok/fail". This lets callers make the right decision (keep running, restart a thread, or shut everything down) without throwing exceptions.

```
enum error_state
{
    SUCCESS,
    ERR_SM,
    ERR_CRITICAL_PROCESS_FAILURE,
    ERR_NONCRITICAL_PROCESS_FAILURE,
    ERR_TMT_FAILURE,
};
```

`error_state` is a user-defined type (like `int`, but restricted to the listed values).

SUCCESS, ERR_SM, … are named integer constants (by default backed by `int`).