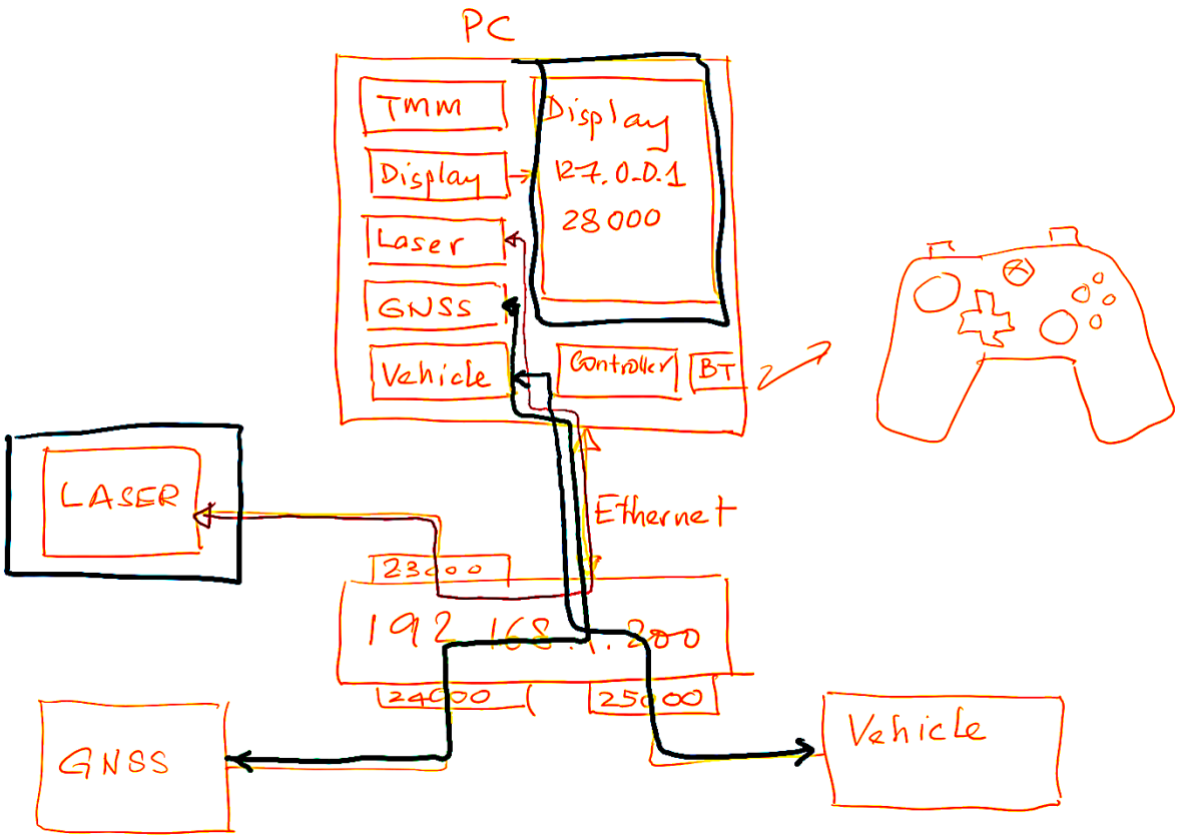
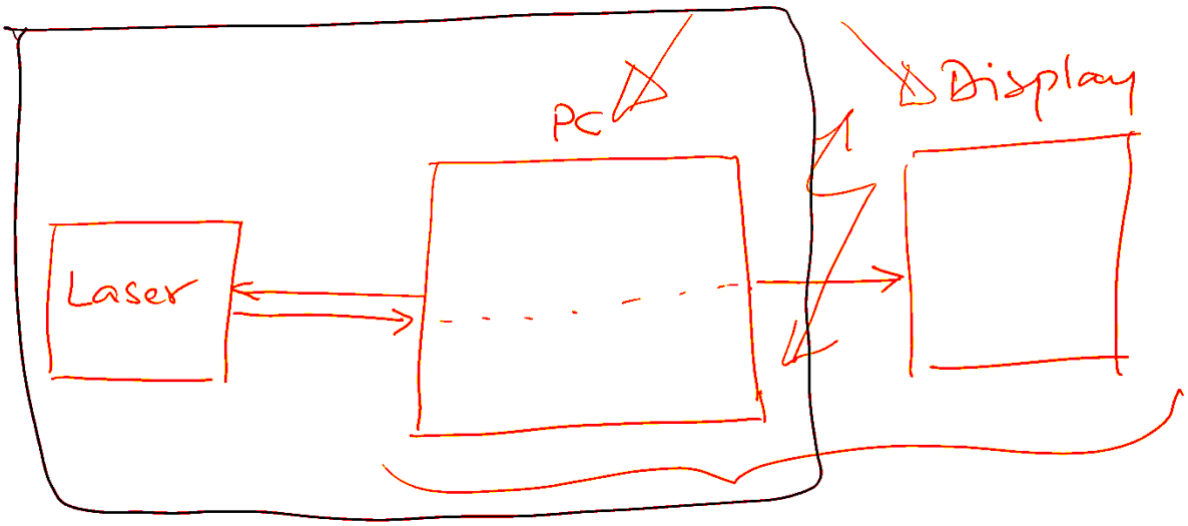
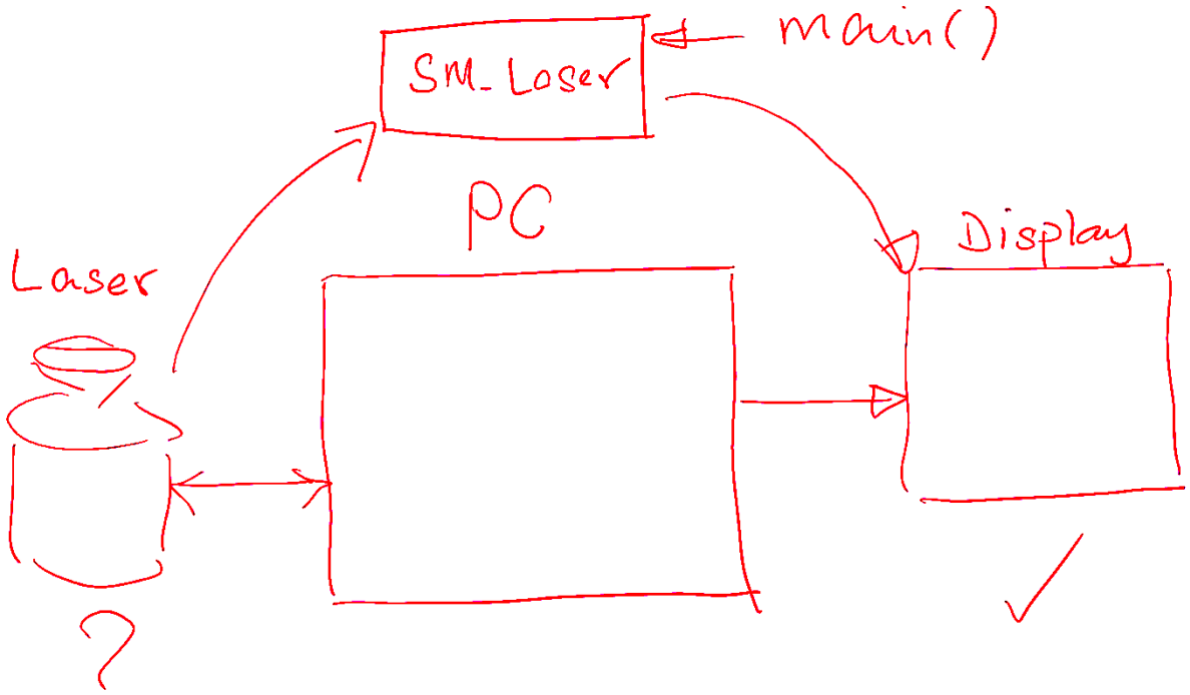

PC
TMM
Display
Display
127.0.0.1
28000
Laser
GNSS
Vehicle
Controller
BT
LASER
Ethernet
23000
192
24000
25000
GNSS
Vehicle

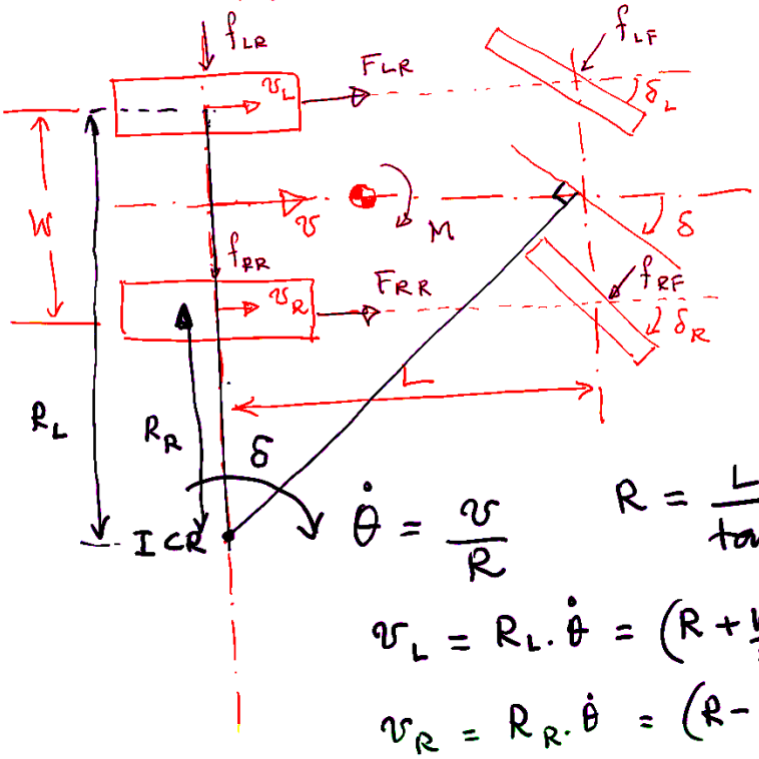


runner

[runDisplayEngine.]


SM_Laser
main()
PC
Display
Laser
?
✓

UGV - hardware.



F_{xx} = All drive forces we have control over.

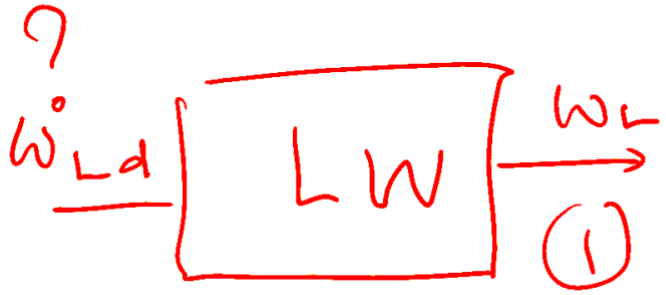
f_{xx} = Lateral frictional forces we do not have control of.

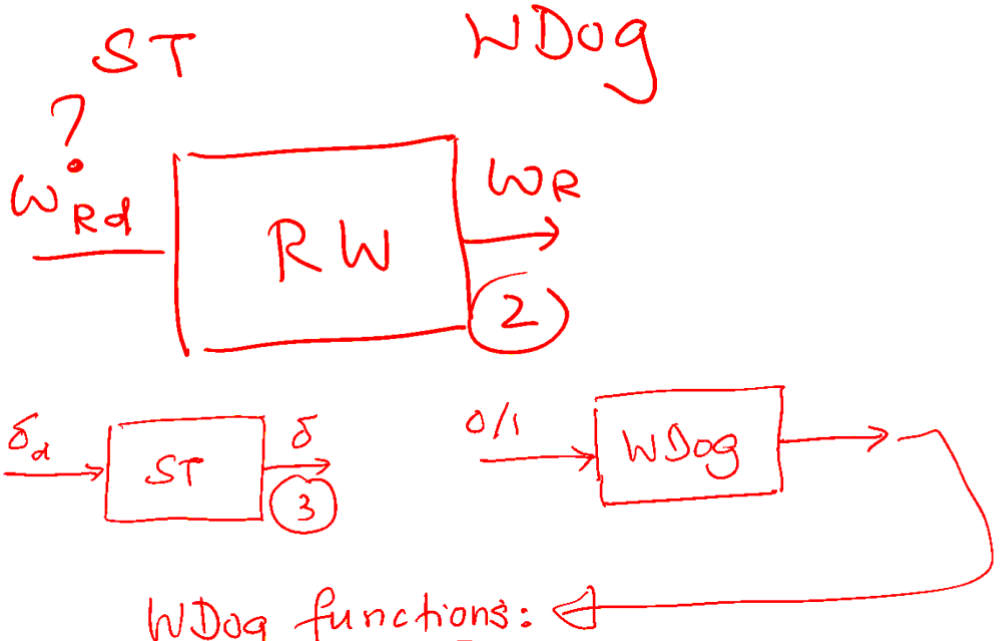
$$\dot{\theta} = \frac{v}{R} \qquad R = \frac{L}{\tan\delta}$$

$$v_L = R_L \cdot \dot{\theta} = \left(R + \frac{W}{2}\right) \cdot \frac{v}{R} = \left(1 + \frac{W}{2R}\right) \cdot v$$

$$v_R = R_R \cdot \dot{\theta} = \left(R - \frac{W}{2}\right) \cdot \frac{v}{R} = \left(1 - \frac{W}{2R}\right) \cdot v$$

$\angle$ LW $\qquad$ $\rightarrow$ RW





WDog functions:

(A) Stopping → $\omega_{Ld} = \omega_{Rd} = 0$; $\delta_d = 0$.

(B) Starting → allow external ω_{Ld}, ω_{Rd}, δ_d to be re-applied.

$$v_L = R_L \cdot \dot{\theta} = \left(R + \frac{W}{2}\right) \cdot \frac{v}{R} = \left(1 + \frac{W}{2R}\right) \cdot v$$

$$v_R = R_R \cdot \dot{\theta} = \left(R - \frac{W}{2}\right) \cdot \frac{v}{R} = \left(1 - \frac{W}{2R}\right) \cdot v$$

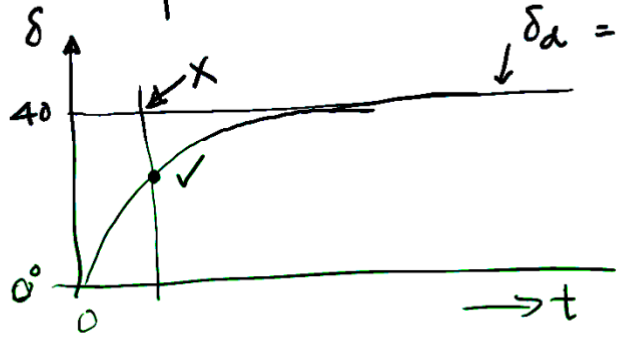
$$v_L = \left(1 + \frac{W}{2L} \tan\delta\right) \cdot v$$

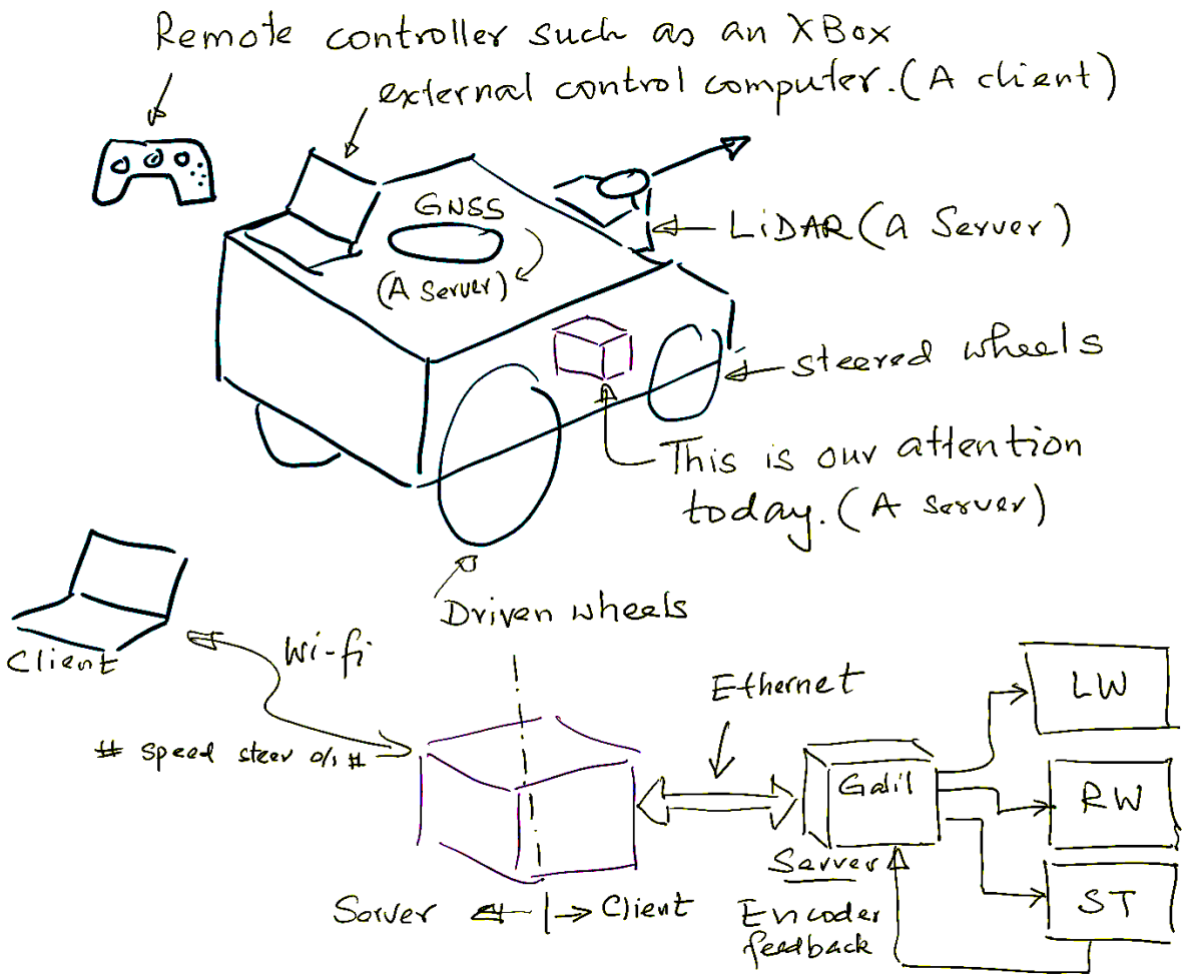
$$v_R = \left(1 - \frac{W}{2L} \tan\delta\right) \cdot v$$

$$\omega_{LD} = \left(1 + \frac{W}{2L} \cdot \tan\delta\right) \cdot \frac{v}{r} \checkmark$$

$$\omega_{RD} = \left(1 - \frac{W}{2L} \tan\delta\right) \cdot \frac{v}{r} \checkmark$$

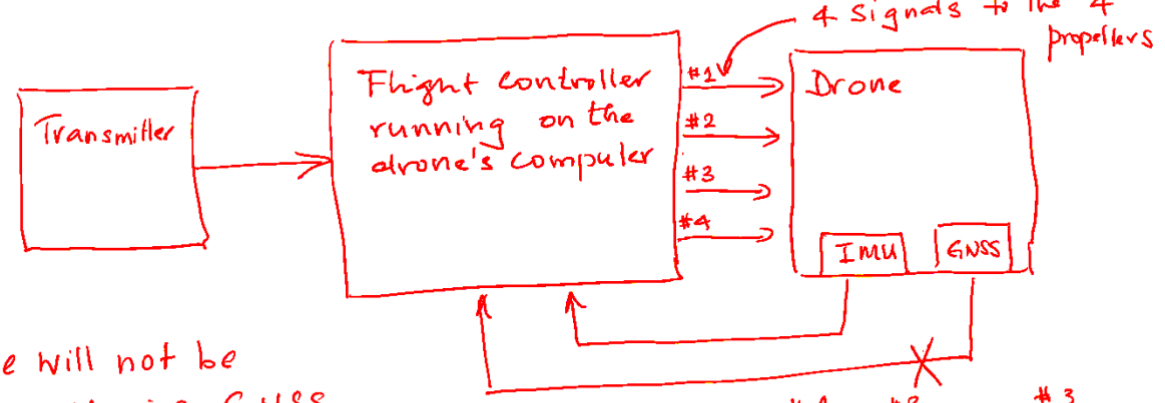
δ used must be measured steering angle obtained by reading the steering encoder.




Remote controller such as an XBox
external control computer.(A client)
GNSS
(A Server)
LiDAR(A Server)
steered wheels
This is our attention
today.(A server)
Driven wheels
Client
Wi-fi
speed steer o/s
Ethernet
Galil
Server
LW
RW
ST
Server
Client
Encoder
feedback

Quadcopter. (Drone)

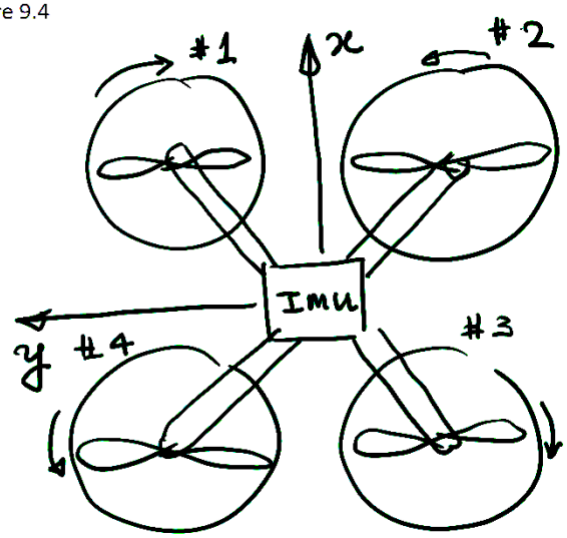
How can we develop software for a flight controller.

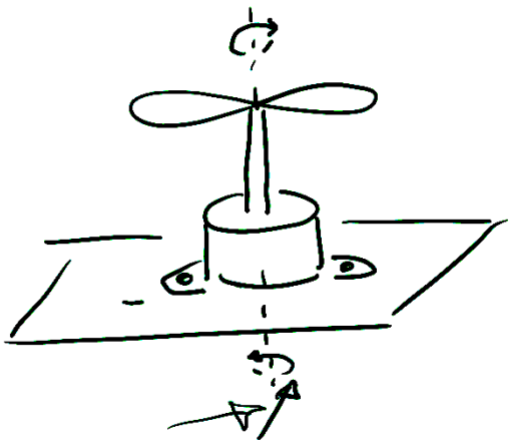


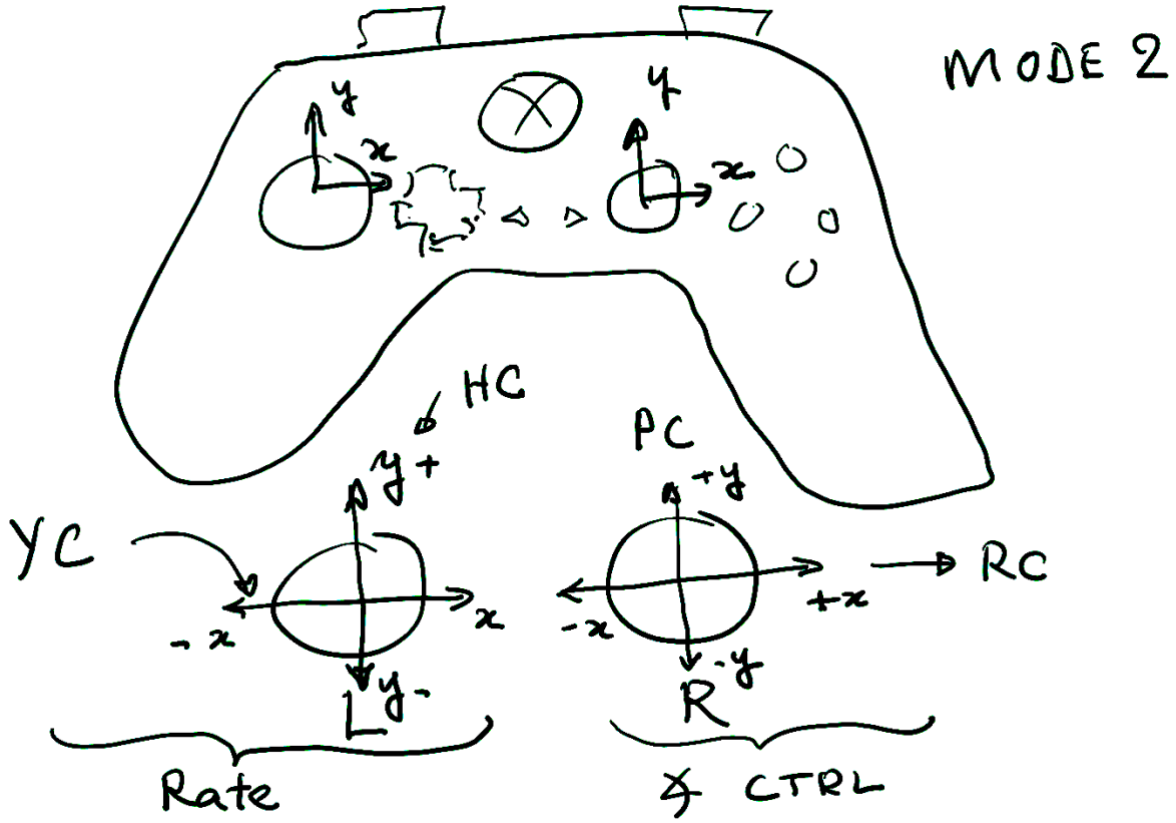
We will not be considering GNSS.

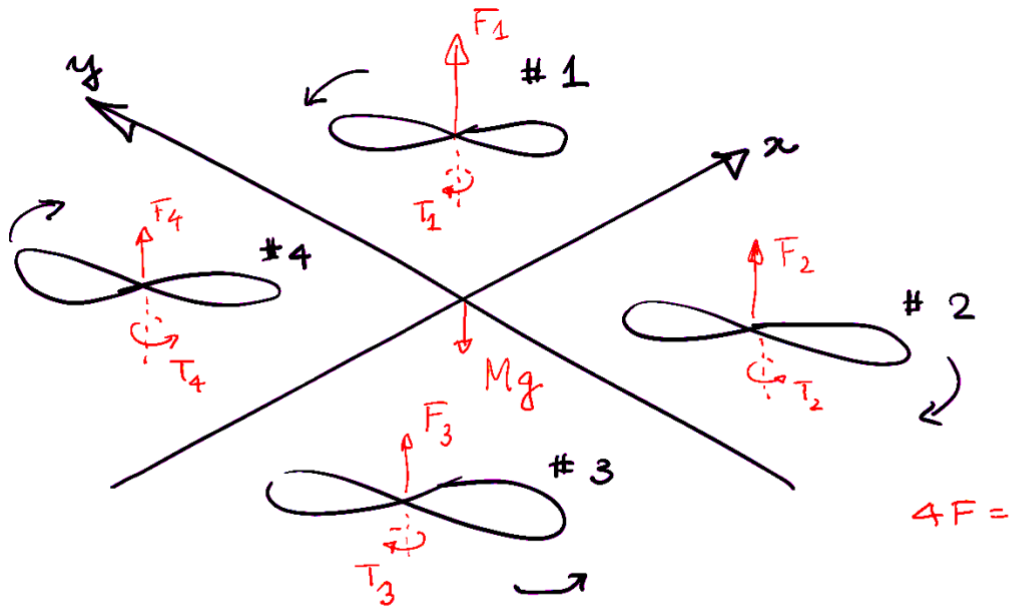
Our flight controller can only control #1 Roll, #2 Pitch and #3 Yaw Hight #4.

Lecture 9.4









$4F = Mg$

Height control: Add an equal amount of force to the 4 rotors by increasing the rotor speeds equally.

(A) Height Control

✓ Rotor #1 : $F \to F \oplus \Delta F$, $T_1 \to T + \Delta T$

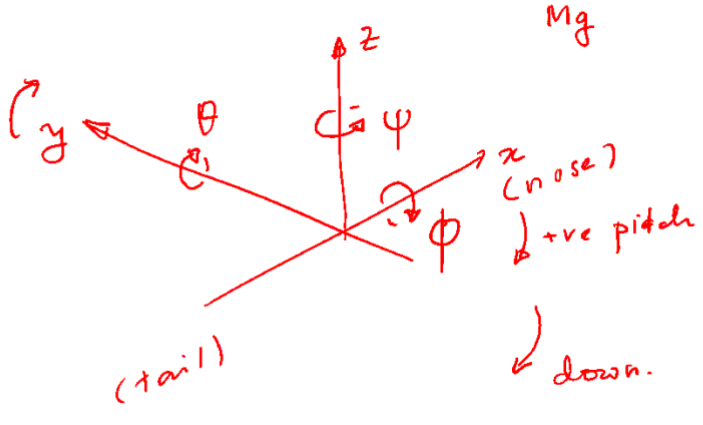
Rotor #2 : $F \to F \oplus \Delta F$, $T_2 \to -T - \Delta T$

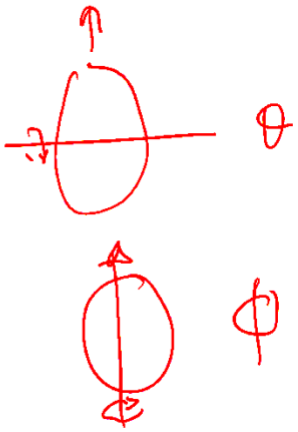
Rotor #3 : $F \to F \oplus \Delta F$, $T_3 \to T + \Delta T$

Rotor #4 : $F \to F \oplus \Delta F$, $T_4 \to -T - \Delta T$

$\sum_1^4 F_i = 4F + 4\Delta F$ ↓ Mg

$\sum_1^4 T_i = 0$





(B) Pitch Control

✓ Rotor #1: $F \rightarrow F \ominus \Delta F$, $T_1 \rightarrow T - \Delta T$

Rotor #2: $F \rightarrow F \ominus \Delta F$, $T_2 \rightarrow -(T - \Delta T)$

Rotor #3: $F \rightarrow F \oplus \Delta F$, $T_3 \rightarrow T + \Delta T$

Rotor #4: $F \rightarrow F \oplus \Delta F$, $T_4 \rightarrow -(T + \Delta T)$

$\sum_1^4 F_i = 4F/\cos\theta$; $\sum_1^4 T_i = 0$

$4F = Mg$

↳ 'Gravity' compensation.

(C) Roll control

✓ Rotor #1: $F_1 \rightarrow F \oplus \Delta F$, $T_1 \rightarrow T + \Delta T$

Rotor #2: $F_2 \rightarrow F \ominus \Delta F$, $T_2 \rightarrow -(T - \Delta T)$

Rotor #3: $F_3 \rightarrow F \ominus \Delta F$, $T_3 \rightarrow (T - \Delta T)$

Rotor #4: $F_4 \rightarrow F \oplus \Delta F$, $T_4 \rightarrow -(T + \Delta T)$

$\sum_1^4 F_i = 4F/\cos\phi = Mg$; $\sum_1^4 T_i = 0$

(D) Yaw Contl

✓ Rotor #1: $F_1 \rightarrow F \ominus \Delta F$, $T_1 \rightarrow T - \Delta T$

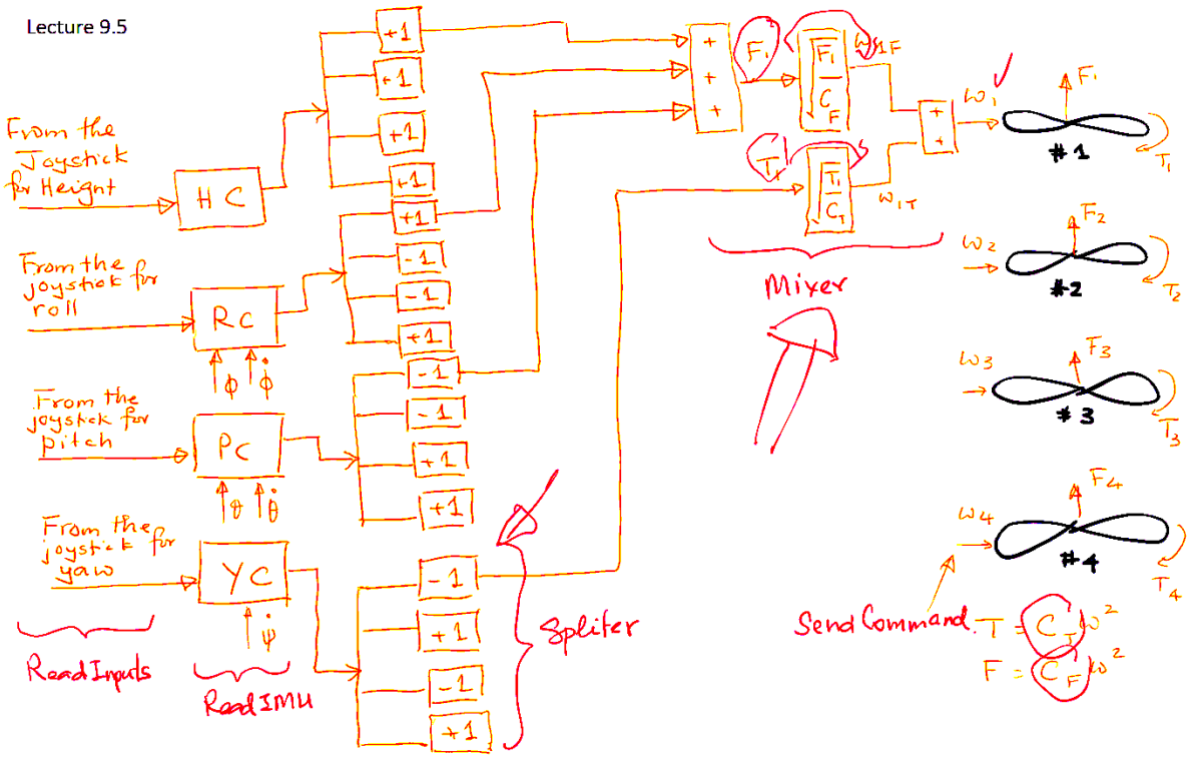
Rotor #2: $F_2 \rightarrow F \oplus \Delta F$, $T_2 \rightarrow -(T + \Delta T)$

Rotor #3: $F_3 \rightarrow F \ominus \Delta F$, $T_3 \rightarrow (T - \Delta T)$

Rotor #4: $F_4 \rightarrow F \oplus \Delta F$, $T_4 \rightarrow -(T + \Delta T)$

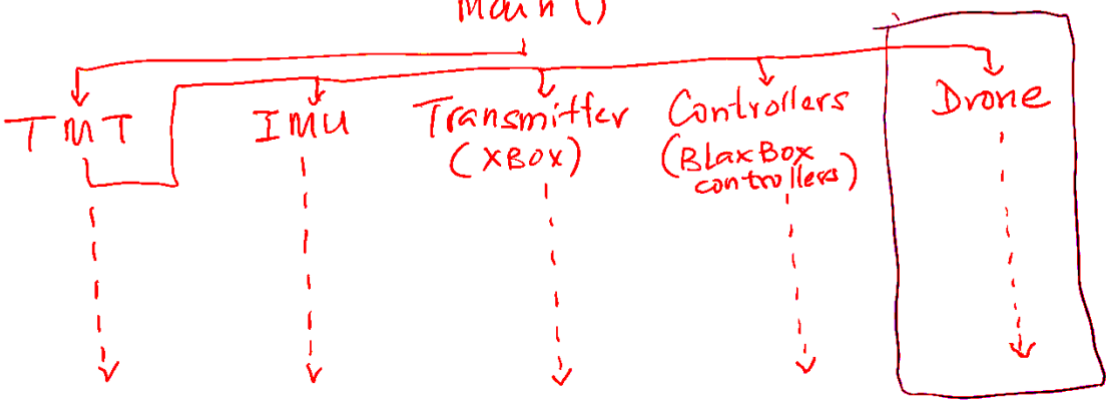
$\sum_1^4 F_i = 4F = Mg$; $\sum_1^4 T_i = -4\Delta T$ (+ve ψ)

Lecture 9.5


From the Joystick for Height
HC
From the joystick for roll
RC
From the joystick for pitch
PC
From the joystick for yaw
YC
Read Inputs
Read IMU
Spliter
Mixer
Send Command
#1
#2
#3
#4

Lecture 9.6

Multi-threaded software for a quadcopter.



Installing Visual Studio. ✓

(1). We need a language ⇒ C++

(2). We need to use an editor ⇒ Visual Studio IDE

(3). Compiling (preceded by pre processor)

← these lines are replaced by the pre processor.

→ (i). Error checking
(ii). Notes down all undefined references.
(iii). If no errors, the compiler generates a file called object file.

(4). Linker

→ (i). Resolves all undefined refevences by finding the proper code from libraries.

(ii). Generate the .exe file

(5). Loads the .exe file to memory and sets the first instruction of your program as the starting point of execution.

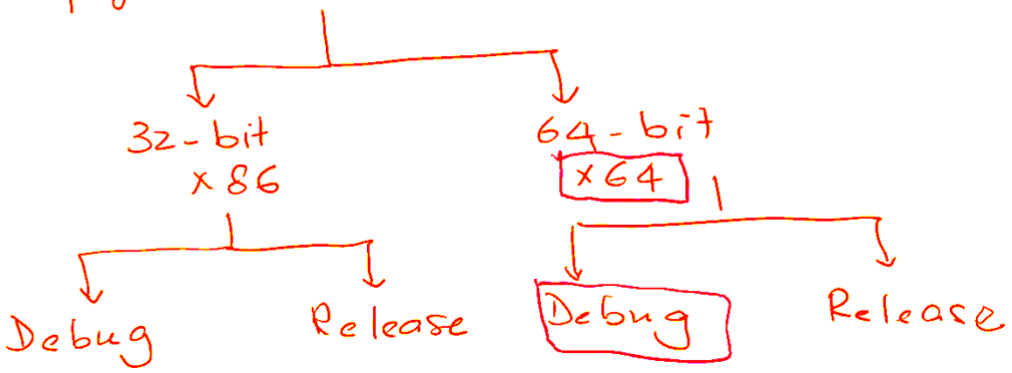
IDE - Integrated Development Environment

(A). A solution ⇒ A collection of projects

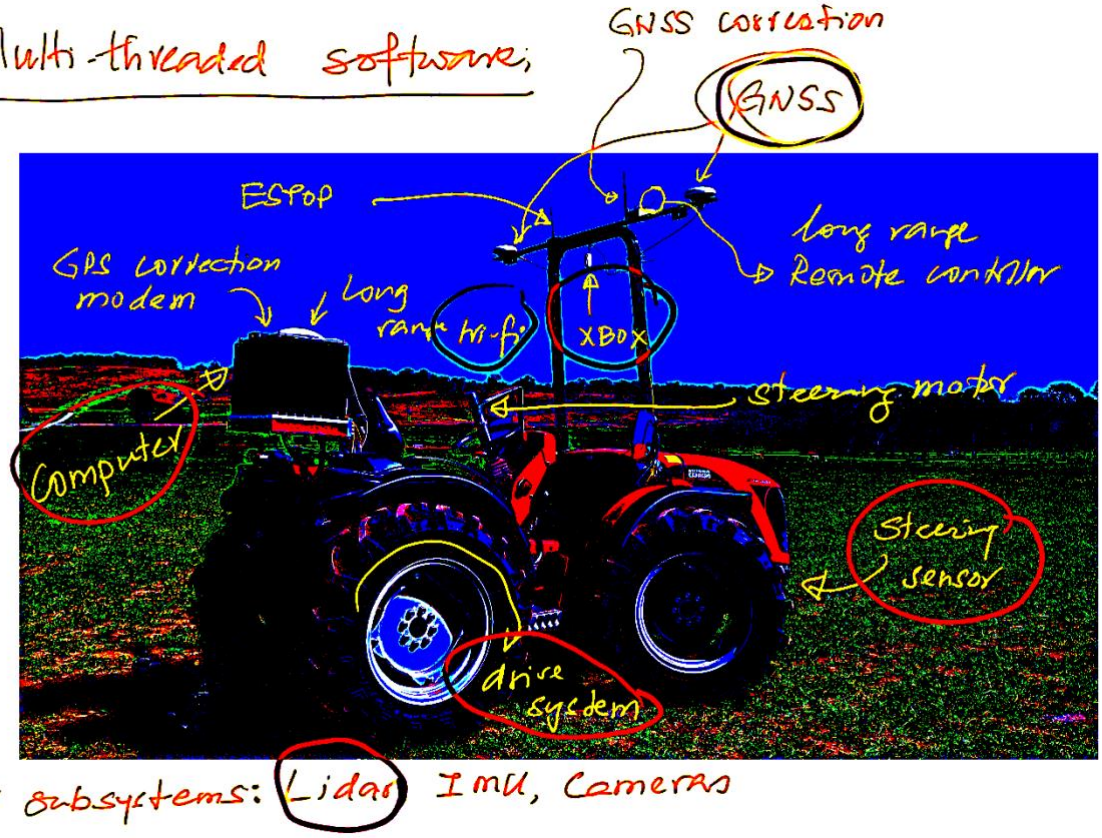
(B). A project

project can have one
↳ header files
↳ source files

① Configurations

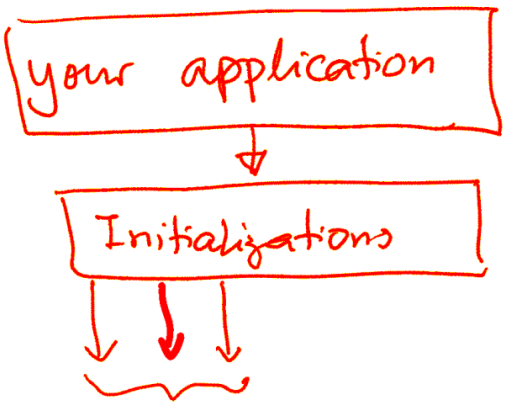


Multi-threaded software:



Other subsystems: Lidar, IMU, Cameras

1. We want Scalable. Our system to b

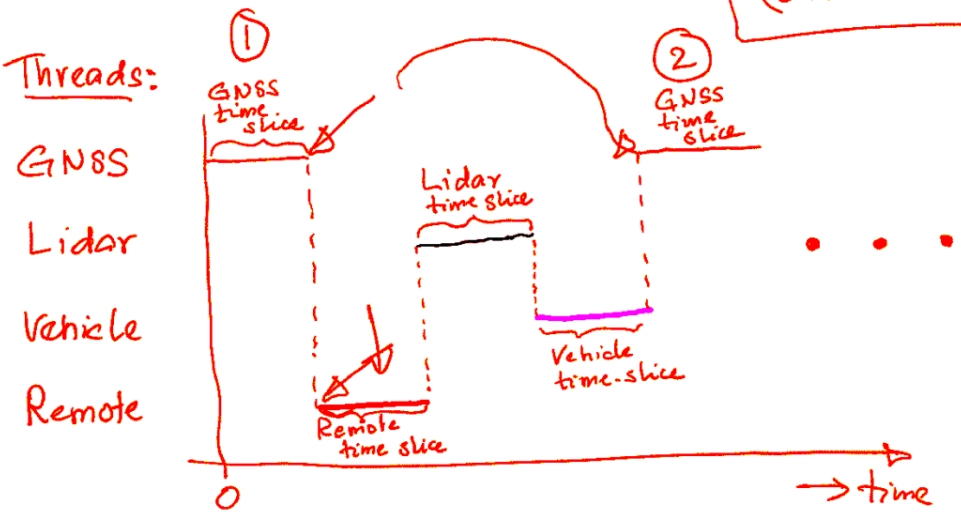


(2) We want our to be

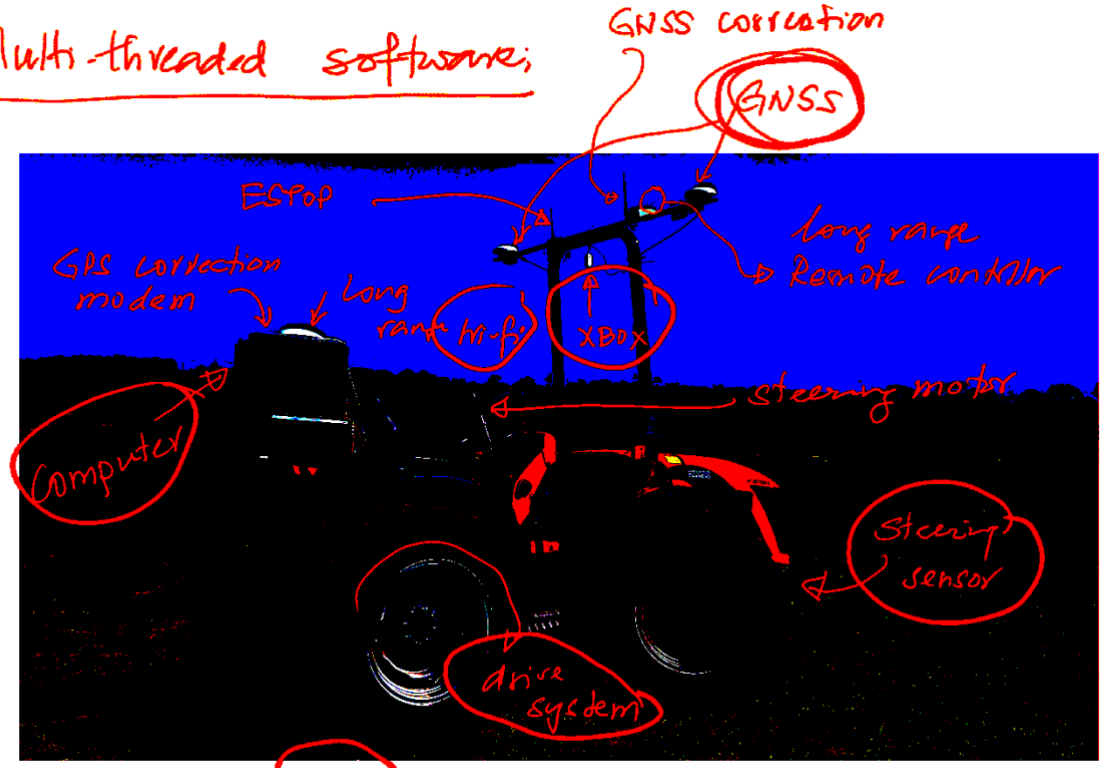
system

③ The system must be responsive.
- Equal attention is given to all subsystems.

(Context switch)

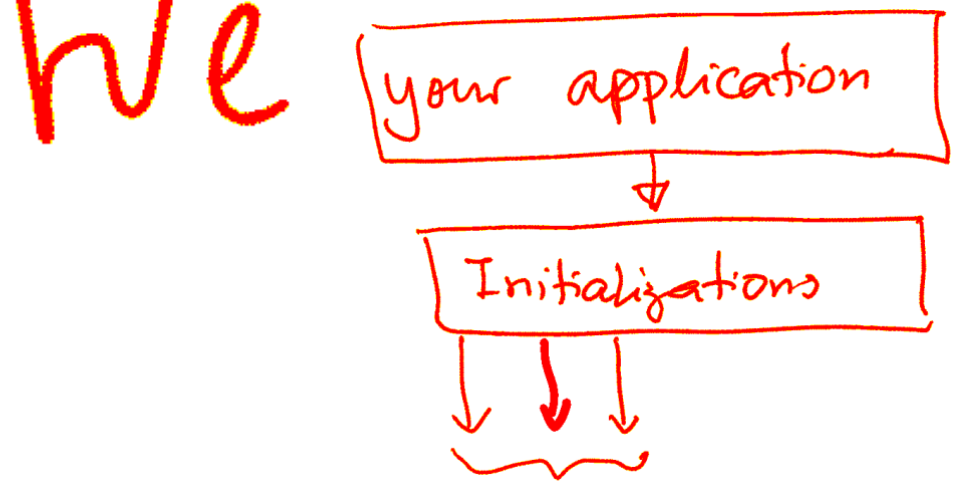


Multi-threaded software:



Other subsystems: Lidar, IMU, Cameras

① We want to be our system our

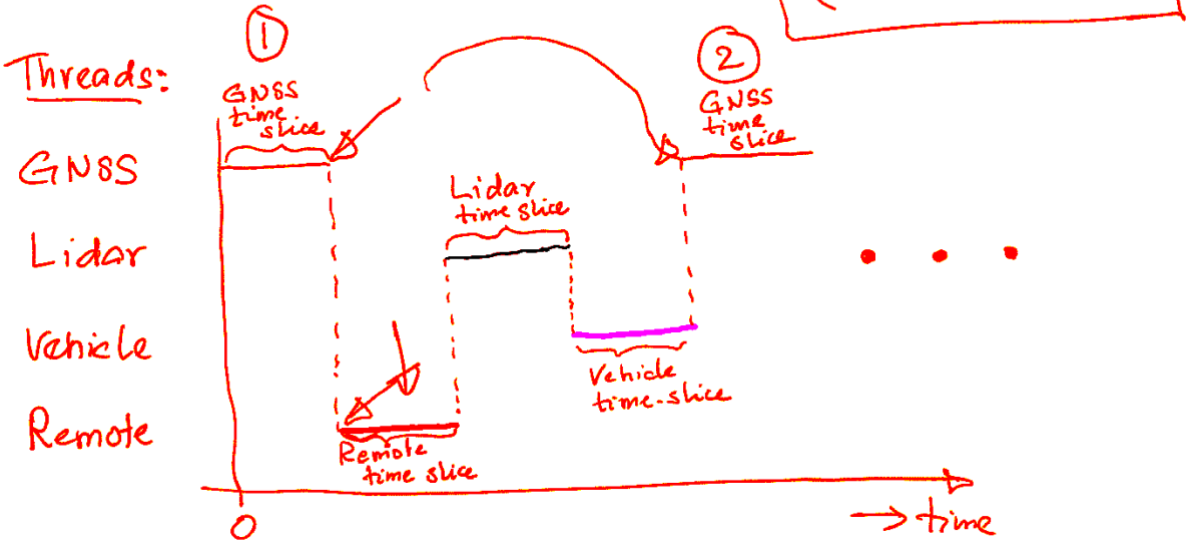


②

want to be system

(3) The system must be responsive.
- Equal attention is given to all subsystems.

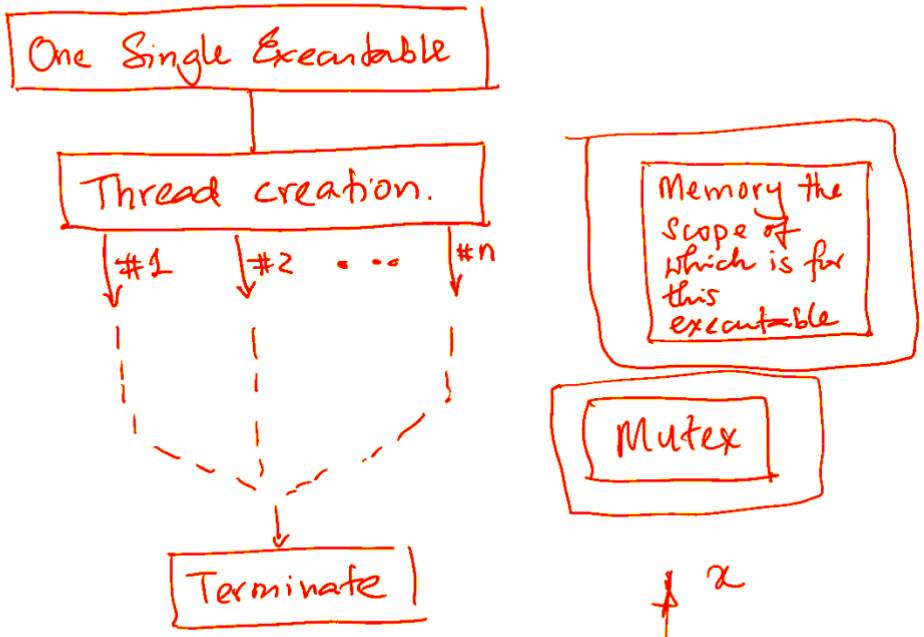
(Context switch)

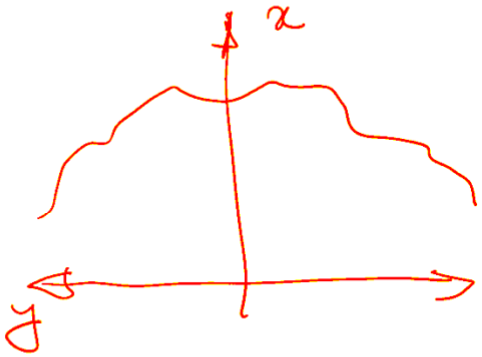


(20,0,20)

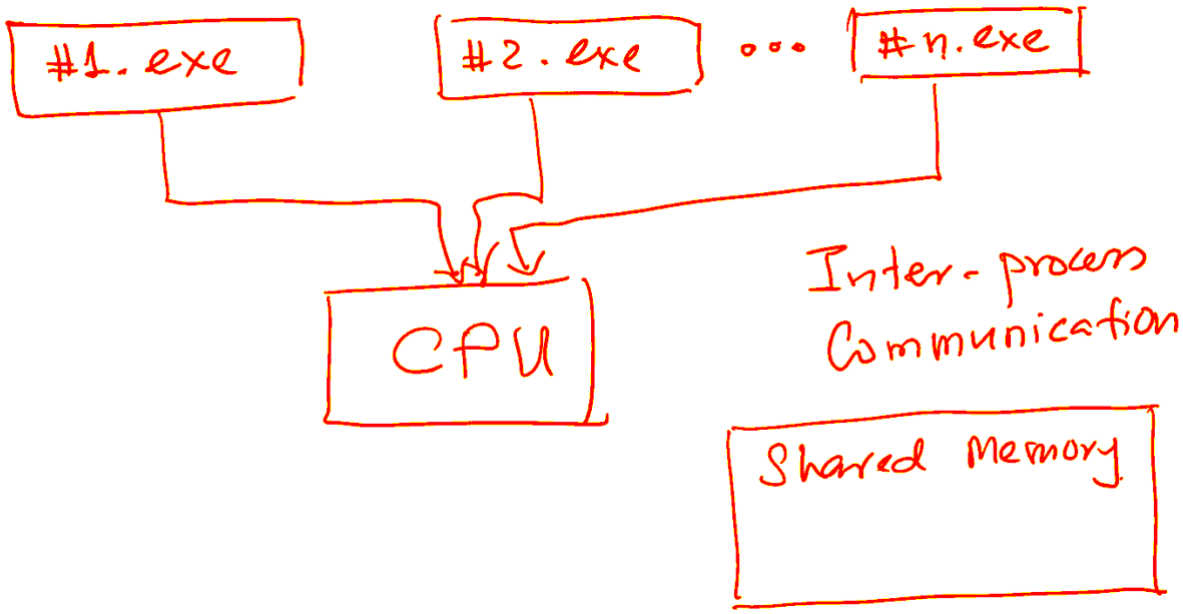
Terminology:

(i). Multi-threaded application.

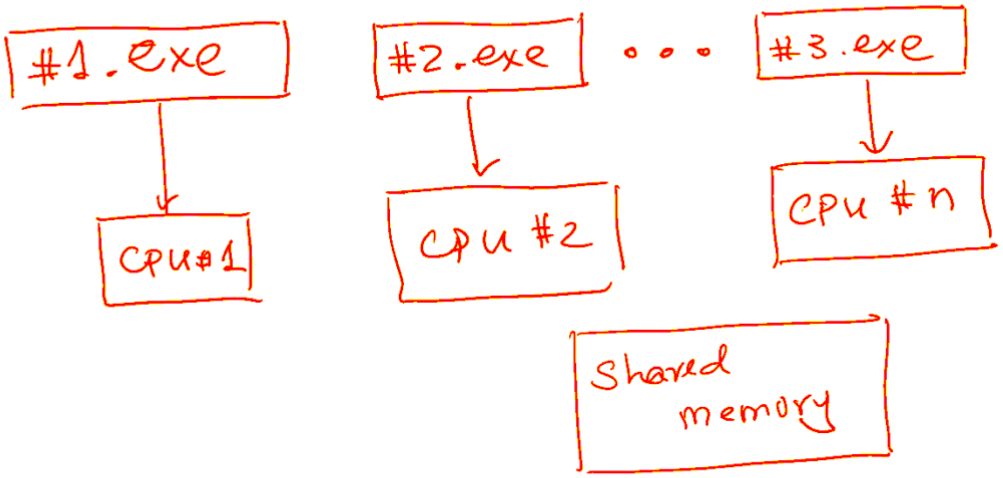




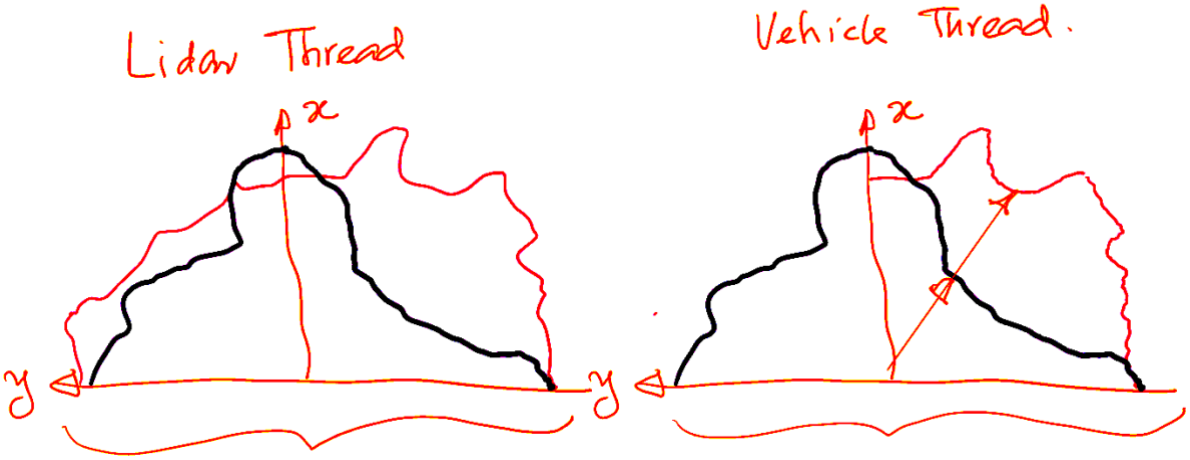
(ii) Multi-tasking systems:



(iii) Multi-processing



What is a race condition?



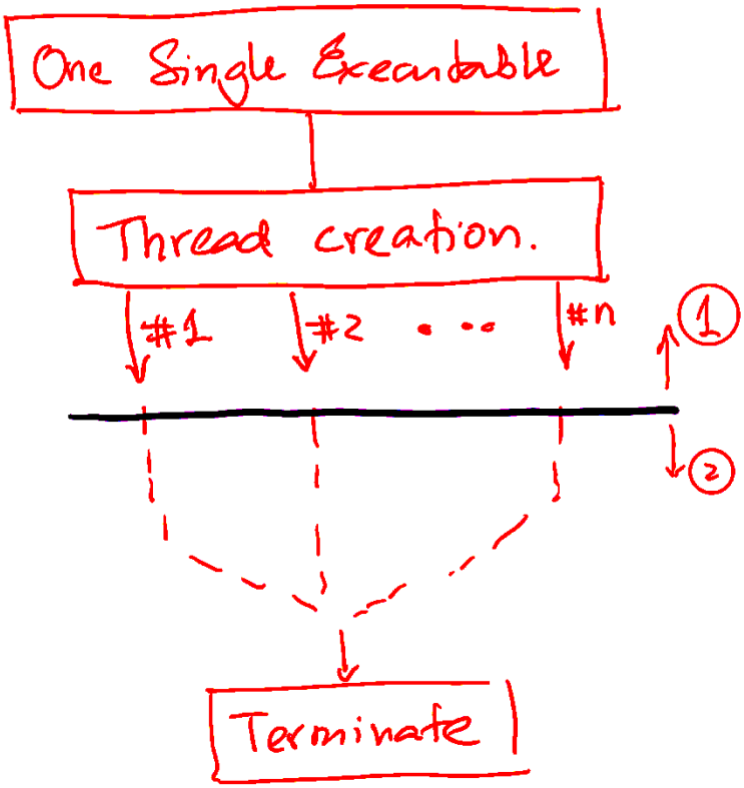
CS

Use of a mutex: You can create a mutex as a member of your class. All threads of this class can then see the mutex. When one thread acquires it, all other threads cannot have it until it is released by the thread that currently holds it. The thread executes the critical section in between acquisition and release.

Use of Semaphores:

A semaphore is mostly used as a book keeping method indicating how many of the multiple resources are currently available for threads. In semaphores WAIT function decrease the count indicating a new user. The signal function increases the count.

Thread Barriers:-



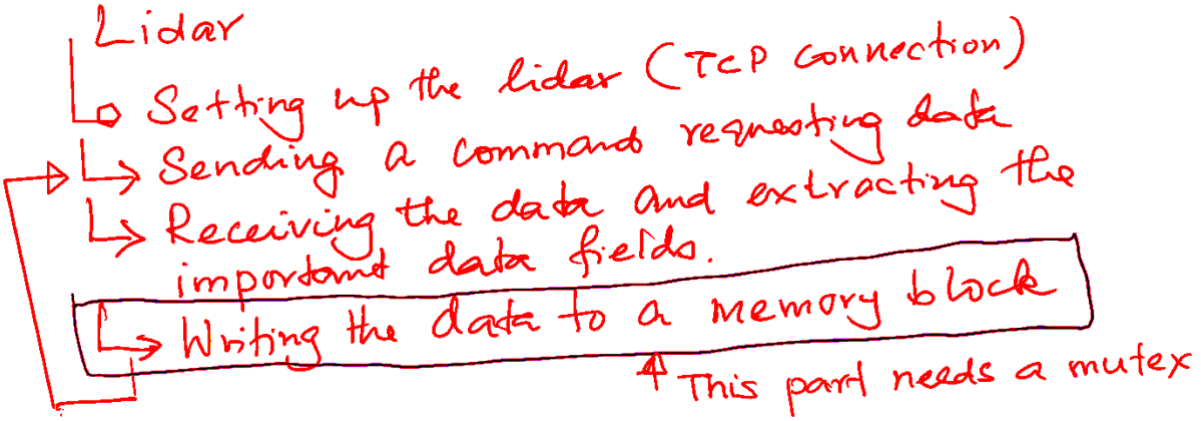
Barrier objects: The barrier can be informed of the number of threads that needs to be readied at the end of phase (1) so that all requirements are met before proceeding with full execution of all the threads.

Pre-emptive Scheduler:

The operating system or the multi-threading software platform put a scheduler into action that regularly interrupts a thread and passes the execution of a thread that was not running. In our software development we do not plan to interfere with this system driven proces.

Multi-threaded software development:

(1). How to create threads
(2). How to use barriers
(3). How to use mutexes.



① Mutex ⟶

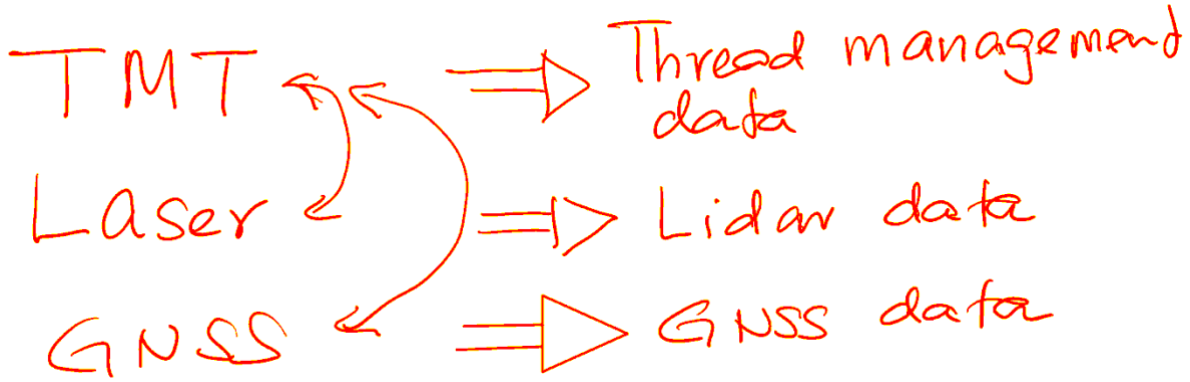
② Monitor ⟷

Thread Management

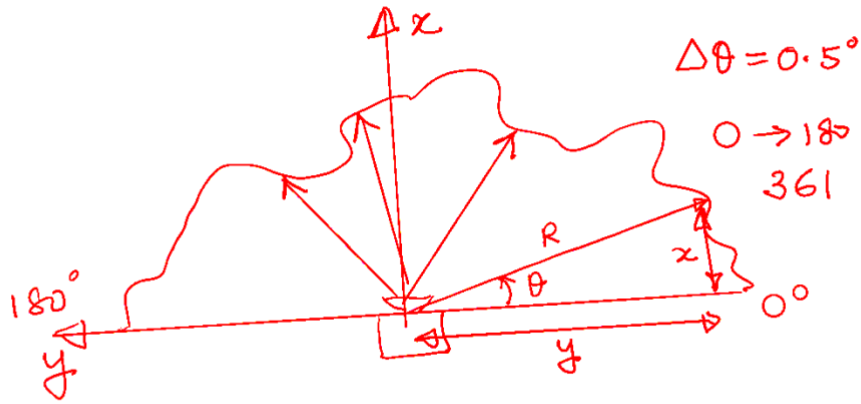
Is to make sure all threads operate according to the thread management policy!

Why do we need a policy?

1. Which thread starts first?
2. " " terminates first?
3. Should the threads follow a custom start up sequence?
4. How can we make sure of all the threads are operating?
5. What will you do when a thread fails? (Restart?, shutdown all?)

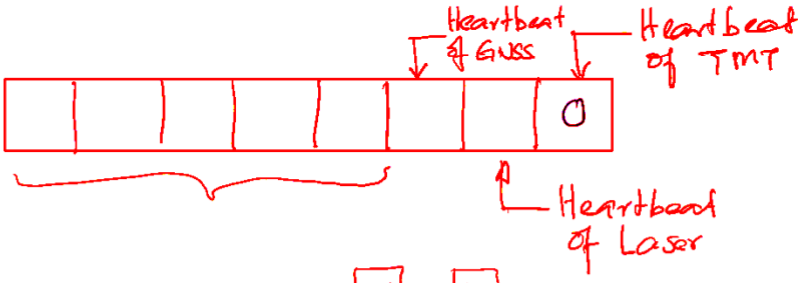


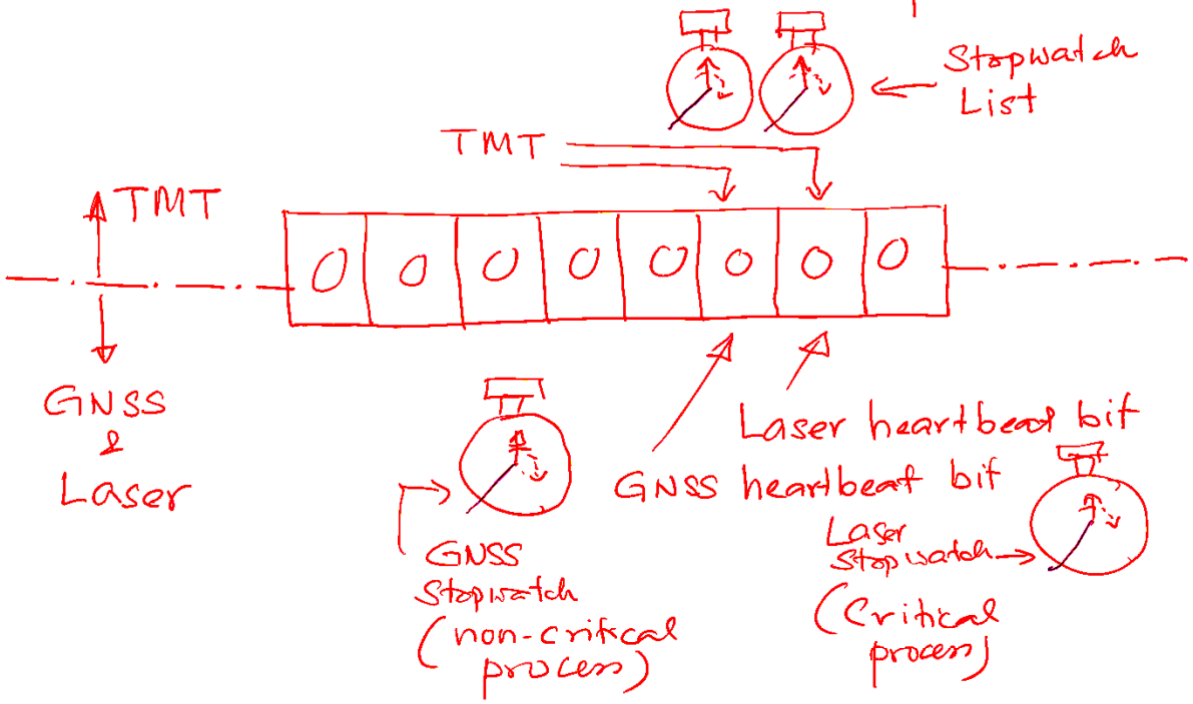
GNSS data ⇒ Latitude, Longitude, Height, ~~Data~~ Intergrity check.

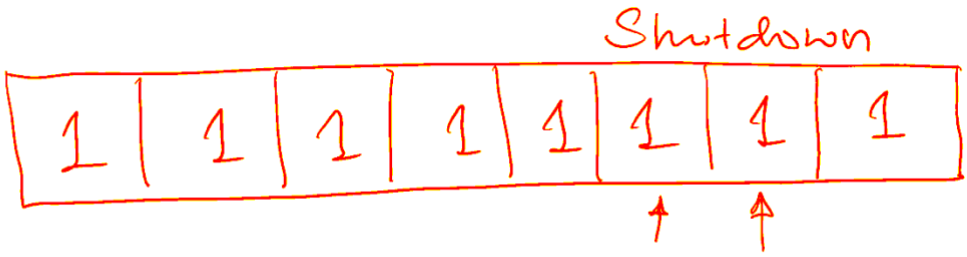


Lidar data ⇒ array <double>^ x;
array <double>^ y;

TMT data ⇒ Uint8_t shutdown; }
Uint8_t heartbeats; }







Shutdown = 0xFF ;
= 0b11111111;

1. ASCII incoming messages
2. Binary incomming messages

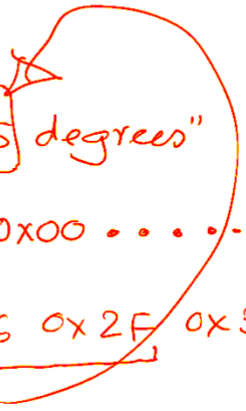
They both consist of a Sequence of bytes arriving at us.

ASCII ⇒ "Room temperature is 23.5 degrees"

Binary ⇒ 0x44 0x12 0xAA 0x1C 0x00 .....

0x44 0x12 0xAA, 0x1C 0x22 0x96 0x2F 0x3F ...

float



Binary data extraction.

double A = 626592.3987869; ⇐ 14

A needs 8-bytes for its storage in memory, because it is of type "double."

Binary data streams

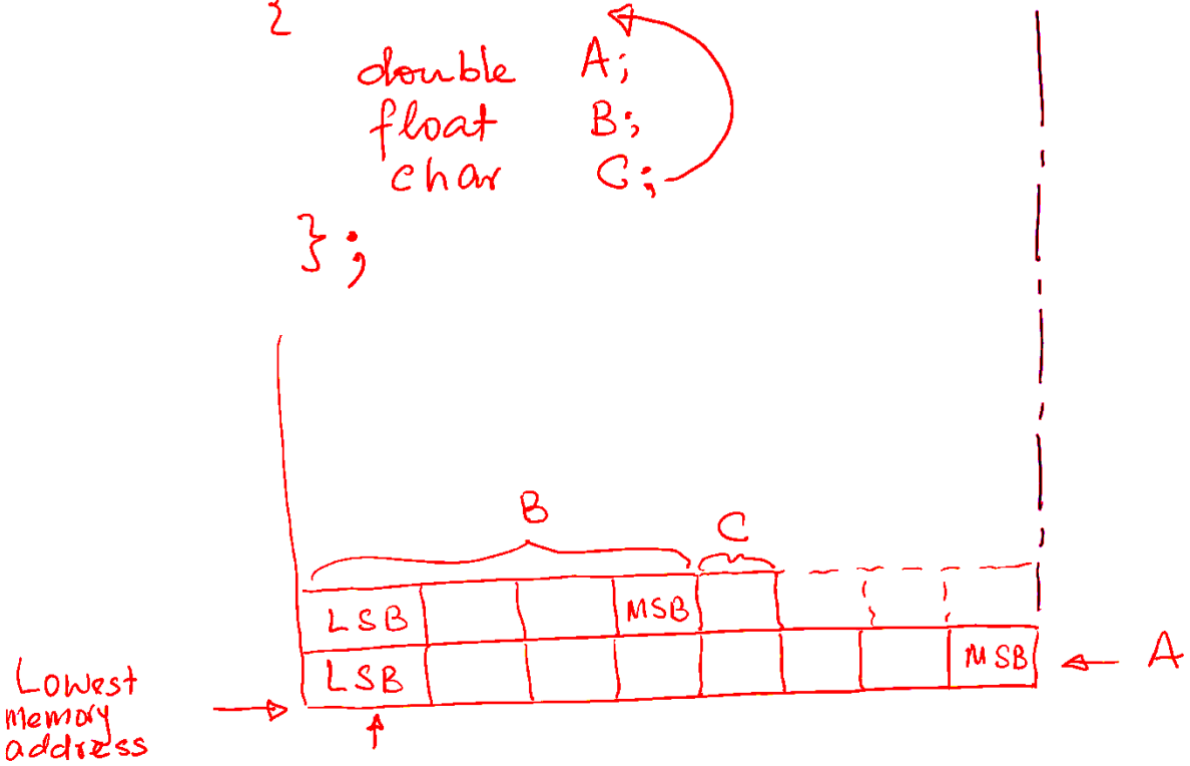
1. 4-bytes of header
2. 40 bytes to skip
3. 8-bytes of Northing
4. 8-bytes of Easting
5. 8-bytes of Height
6. 40-bytes to skip again
7. 4-bytes of CRC (quality check indicator)

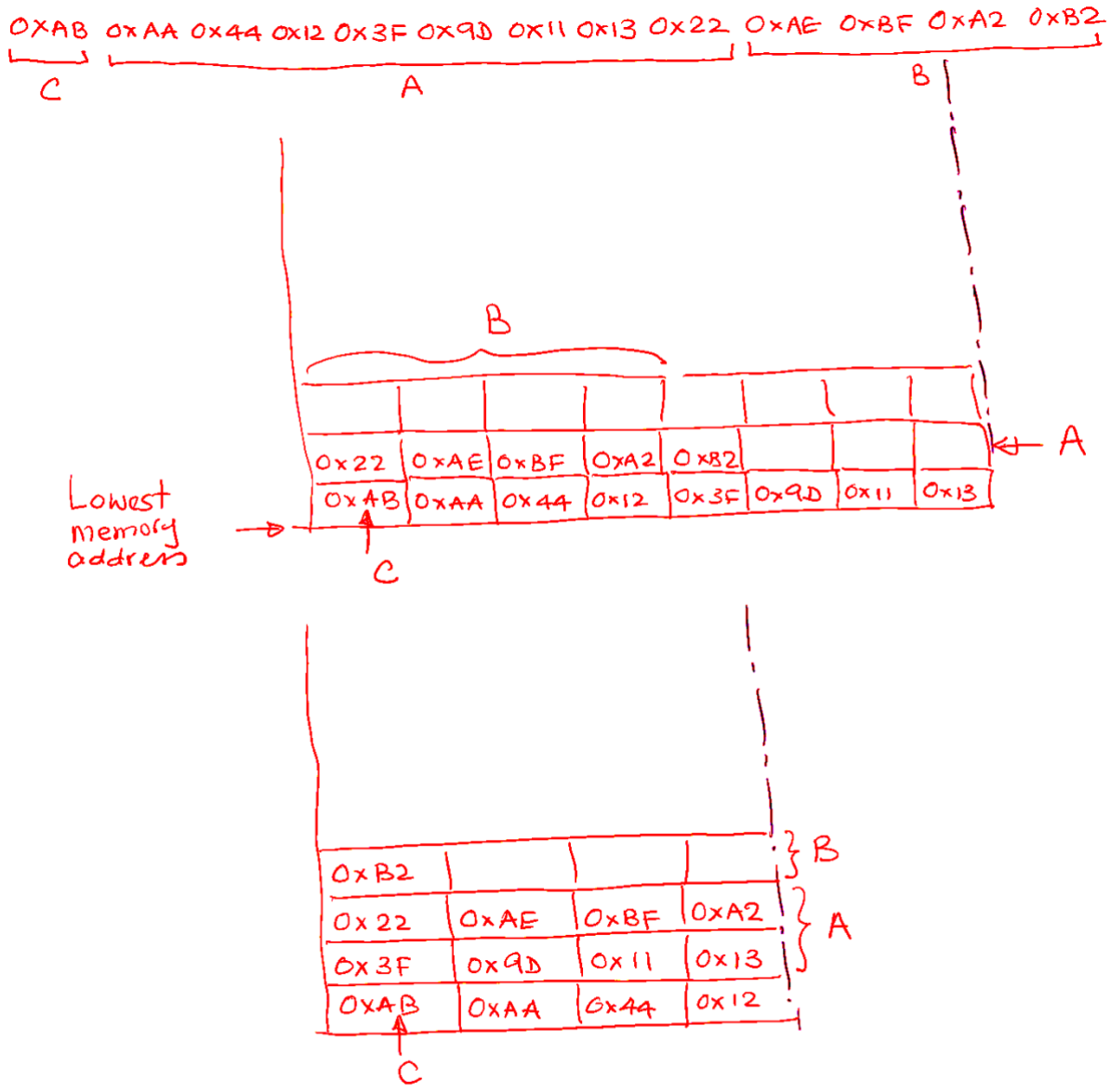
How data structures store data

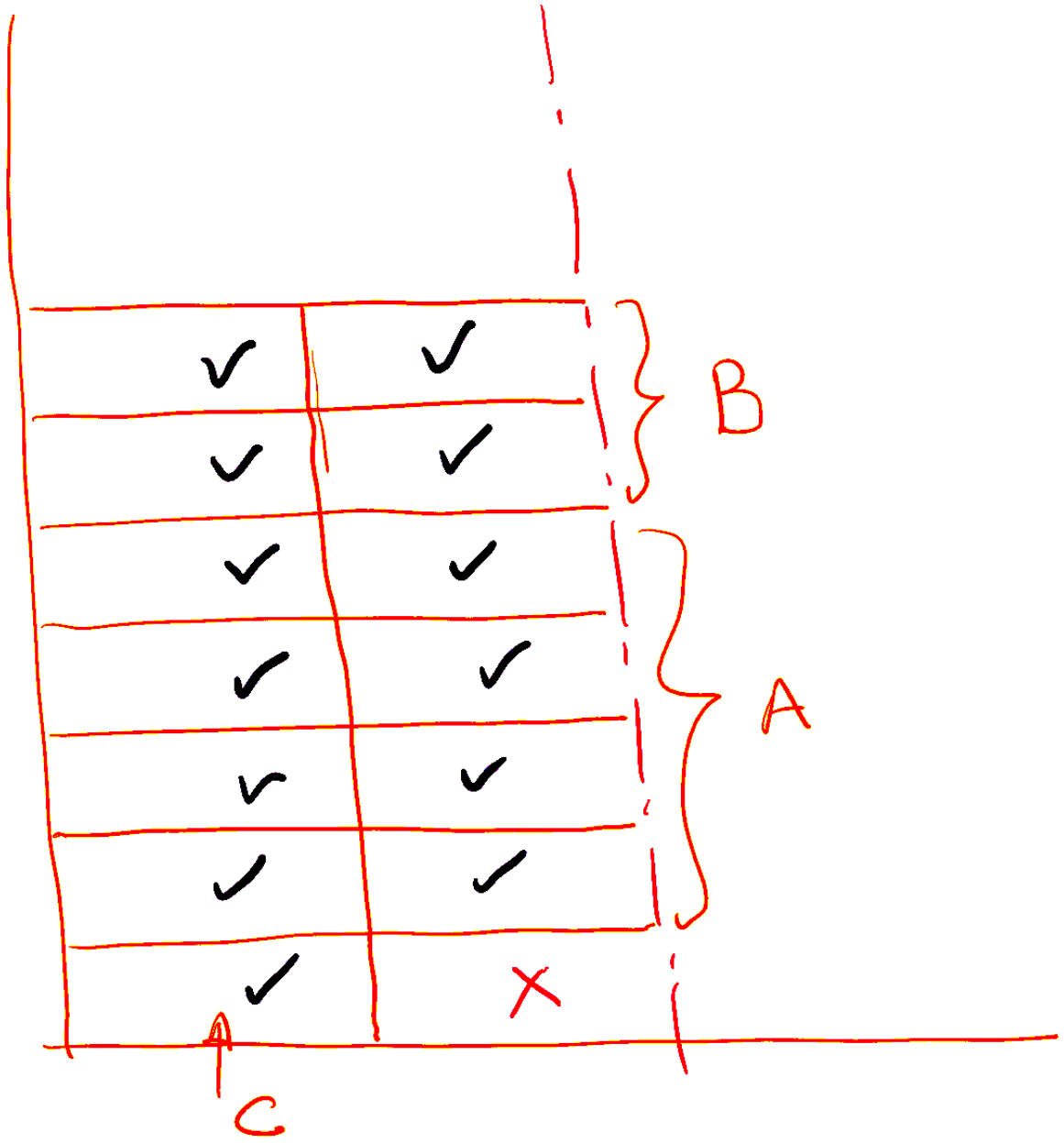
We consider three simple data types

- a double (8 bytes) A
- a float (4 bytes) B
- a char (1 byte) C

```
struct Data
{
    double A;
    float  B;
    char   C;
};
```




0XAB 0xAA 0x44 0x12 0x3F 0x9D 0x11 0x13 0x22 0xAE 0xBF 0xA2 0xB2
C
A
B
B
A
0x22 0xAE 0xBF 0xA2 0xB2
0xAB 0xAA 0x44 0x12 0x3F 0x9D 0x11 0x13
Lowest memory address
C
0xB2
0x22 0xAE 0xBF 0xA2
0x3F 0x9D 0x11 0x13
0xAB 0xAA 0x44 0x12
B
A
C


B
A
X
C

```
unsigned int Header = 0;
Byte Data;

do
{
    Data = GNSSStream->ReadByte(); // 0xe8
    Header = (Header << 8) | Data

} while (Header != 0xaa44121c);

// Now read 108 bytes of data to be stored into the
// GNSS type data structure (Note: No header bytes in the structure)
```