bdd [package] Architecture [Architecture]

CommsModule

+ client_IDs: std::vector<int>
- client_IDs_vec_mutex: pthread_mutex_t
- clientSocket: SOCKET = {}
- ID: int
- rx_messages: std::vector<Message>
- rx_vec_mutex: pthread_mutex_t
+ self_truck: TruckMetadata*
- serverAddr: sockaddr_in = {}
- timeout: timeval = {}
- tx_messages: std::vector<Message>
- tx_vec_mutex: pthread_mutex_t

+ add_tx_message_to_buffer(Message): int
+ CommsModule(int, long, TruckMetadata*)
+ connect_to_Server(): int
+ get_connected_client_IDs(): std::vector<int>
+ get_last_rx_message_from_buffer(bool): std::optional<Message>
+ get_length_of_rx_buffer(): int
+ get_rx_message_by_index_from_buffer(int, bool): std::optional<Message>
+ initialize(std::string&, u_short): int
+ print_rx_messages_from_buffer(): int
+ receive_rxBuffer(): int
- receive_string(std::string&): int
+ run(void*): void*
+ run_thread(): void*
- send_message(Message): int
- send_string(std::string&): int
+ send_txBuffer(): int

«struct» TruckMetadata

+ client_IDs_vec_mutex: pthread_mutex_t*
+ event_handler: event = ev_any
+ movement_leader: std::vector<movement>
+ movement_leader_vec_mutex: pthread_mutex_t
+ pending_send_message: std::vector<Message>
+ received_message: std::vector<Message>
+ received_message_vector_guard: std::mutex
+ received_message_vector_mutex: pthread_mutex_t
+ role: truckRole = NOT_SET
+ send_message_vector_mutex: pthread_mutex_t
+ send_messsage_vector_guard: std::mutex
+ surrounding_truck: std::vector<SurroundingTruck>
+ surrounding_truck_IDs: std::vector<int>*
+ truck_id: int = -1
+ truck_leader_id: int = -1
+ truck_logical_clock: logicalClock
+ watchdog: time_t

+self_truck

+self

Truck

+ dummy_client_IDs_vec_mutex: pthread_mutex_t
+ self: TruckMetadata
+ t_communication: pthread_t
+ t_controller: pthread_t
+ t_decryptor: pthread_t
+ t_interface: pthread_t
+ truck_communication: CommsModule = CommsModule(0,1...
+ truck_controller: controller = controller(0, &self)
+ truck_decryptor: Decryptor = Decryptor(&self)

+ run(): void

+truck_communication

-self_truck

+self_truck

Decryptor

- self_truck: TruckMetadata*

+ Decryptor(TruckMetadata*)
+ run(): void
+ run(void*): void*
+ run_thread(): void*

+truck_decryptor

controller

+ cruiseDriverLimit: int = 210 {readOnly}
+ cruiseDriverStep: int = 10 {readOnly}
+ current_movement: movement = {}
+ current_state: controllerState = initial
+ current_state_in_follower_state: controllerState = movin
+ current_state_in_leader_state: controllerState = moving
+ id: int = -1
+ initialized: bool = false
+ input_given: bool = false
+ input_mutex: pthread_mutex_t
+ next_state: controllerState = initial
+ next_state_in_follower_state: controllerState = moving
+ next_state_in_leader_state: controllerState = moving
+ self_truck: TruckMetadata* = {}

+ controller(int, TruckMetadata*)
+ find_leader(): bool
+ get_current_direction(): movementDirection
+ get_current_movement(): movement
+ get_current_role(): truckRole
+ get_current_speed(): int
+ get_current_state(): controllerState
+ key_board_run(void*): void*
+ key_board_run_thread(): void*
+ move(movement): event
+ move_emergency_stop(): event
+ move_follower(): event
+ move_leader(): event
+ move_stop(): event
+ next_state_computer(event): void
+ run(void*): void*
+ run_thread(): void*
+ send_message_to_follower(Message): void
+ set_current_direction(movementDirection): void
+ set_current_movement(movementDirection): void

+truck_controller

«struct» SocketClientID

+ clientSocket: SOCKET
+ ID: int

«struct» SurroundingTruck

+ id: int = -1
+ x_location: int = -1

«struct» SocketClientID

+ clientSocket: SOCKET
+ ID: int

«struct» Message

+ destID: int

logicalClock

-   logicalClock_u64: int
+   get_logicalClock(): int
+   logicalClock()
+   logicalClockTick(): void
+   logicalClockTickCompare(int): bool
+   logicalClockUpdate(int): void

uck_logical_clock

«Enumeration»
event

ev_any
ev_stop
ev_reset
ev_ready
ev_be_leader
ev_be_follower
ev_leader_found
ev_no_leader_found

+event_handler

«struct»
Movement

+   direction: movementDirection
+   speed: int

+role

-event_

«Enumeration»
truckRole

NOT_SET
LEADER
FOLLOWER

-role_e

+direction

«Enumeration»
movementDirection

MOVE_FORWARD
MOVE_BACK
MOVE_LEFT
MOVE_RIGHT
MOVE_EMERGENCY_STOP
MOVE_STOP

Message

-   controllerSerialNumber_u8: uint8_t
-   direction: movementDirection
-   event_: event
-   logicalClock_u64: uint64_t
-   receiver_id: int
-   role_e: truckRole
-   sender_id: int
-   speed: int
+   getControllerSerialNumber(): uint8_t {query}
+   getDirection(): movementDirection {query}
+   getEvent(): event {query}
+   getLogicalClock(): uint64_t {query}
+   getReceiverId(): int {query}
+   getRole(): truckRole {query}
+   getSenderId(): int {query}
+   getSpeed(): int {query}
+   Message()
+   setControllerSerialNumber(uint8_t): void
+   setDirection(movementDirection): void
+   setEvent(event): void
+   setLogicalClock(uint64_t): void
+   setReceiverId(int): void
+   setRole(truckRole): void
+   setSenderId(int): void
+   setSpeed(int): void

-direction

+next_state

+next_state_in_follower_state

+current_state

+next_state_in_leader_state

+current_state_in_follower_state

+current_state_in_leader_state

«Enumeration»
controllerState

initial
waiting
leader
follower
moving
aligning
stop
system_stop

+ messagestr: std::string
+ sourceID: int

MessageParser

+ <u>directionToString(movementDirection): std::string</u>
+ <u>fromJSON(std::string&): Message</u>
+ <u>stringToDirection(std::string&): movementDirection</u>
+ <u>stringToTruckRole(std::string&): truckRole</u>
+ <u>toJSON(Message&): std::string</u>
+ <u>truckRoleToString(truckRole): std::string</u>

+ set_current_role(truckRole): void
+ set_current_speed(int): void
+ state_align(): event
+ state_follower(): event
+ state_initial(): event
+ state_leader(): event
+ state_moving(): event
+ state_stop(): event
+ state_system_stop(): event
+ state_waiting(): event