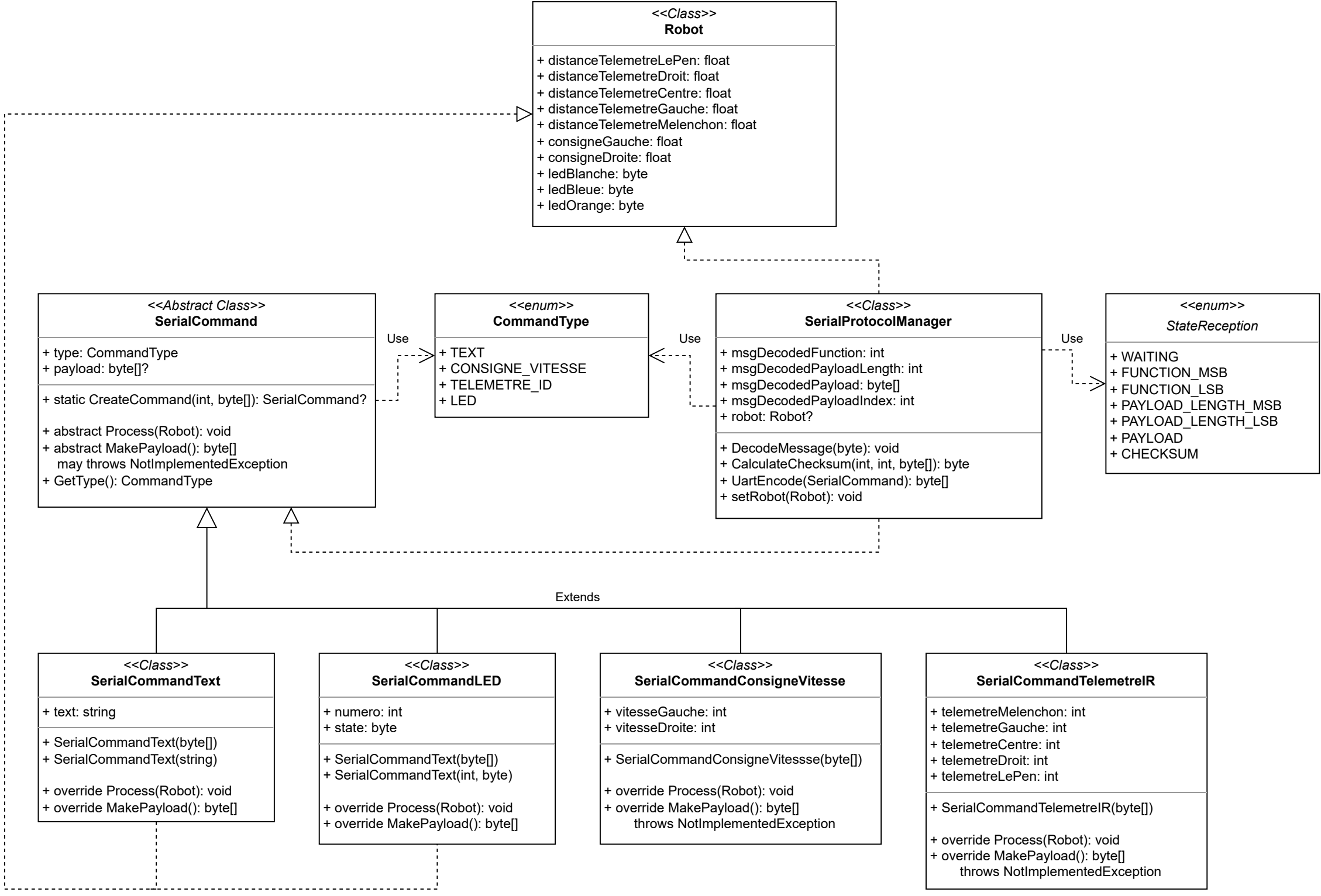
<<Class>>
Robot
+ distanceTelemetreLePen: float
+ distanceTelemetreDroit: float
+ distanceTelemetreCentre: float
+ distanceTelemetreGauche: float
+ distanceTelemetreMelenchon: float
+ consigneGauche: float
+ consigneDroite: float
+ ledBlanche: byte
+ ledBleue: byte
+ ledOrange: byte
<<Abstract Class>>
SerialCommand
+ type: CommandType
+ payload: byte[]?
+ static CreateCommand(int, byte[]): SerialCommand?
+ abstract Process(Robot): void
+ abstract MakePayload(): byte[]
may throws NotImplementedException
+ GetType(): CommandType
Use
<<enum>>
CommandType
+ TEXT
+ CONSIGNE_VITESSE
+ TELEMETRE_ID
+ LED
Use
<<Class>>
SerialProtocolManager
+ msgDecodedFunction: int
+ msgDecodedPayloadLength: int
+ msgDecodedPayload: byte[]
+ msgDecodedPayloadIndex: int
+ robot: Robot?
+ DecodeMessage(byte): void
+ CalculateChecksum(int, int, byte[]): byte
+ UartEncode(SerialCommand): byte[]
+ setRobot(Robot): void
Use
<<enum>>
StateReception
+ WAITING
+ FUNCTION_MSB
+ FUNCTION_LSB
+ PAYLOAD_LENGTH_MSB
+ PAYLOAD_LENGTH_LSB
+ PAYLOAD
+ CHECKSUM
Extends
<<Class>>
SerialCommandText
+ text: string
+ SerialCommandText(byte[])
+ SerialCommandText(string)
+ override Process(Robot): void
+ override MakePayload(): byte[]
<<Class>>
SerialCommandLED
+ numero: int
+ state: byte
+ SerialCommandText(byte[])
+ SerialCommandText(int, byte)
+ override Process(Robot): void
+ override MakePayload(): byte[]
<<Class>>
SerialCommandConsigneVitesse
+ vitesseGauche: int
+ vitesseDroite: int
+ SerialCommandConsigneVitessse(byte[])
+ override Process(Robot): void
+ override MakePayload(): byte[]
throws NotImplementedException
<<Class>>
SerialCommandTelemetreIR
+ telemetreMelenchon: int
+ telemetreGauche: int
+ telemetreCentre: int
+ telemetreDroit: int
+ telemetreLePen: int
+ SerialCommandTelemetreIR(byte[])
+ override Process(Robot): void
+ override MakePayload(): byte[]
throws NotImplementedException