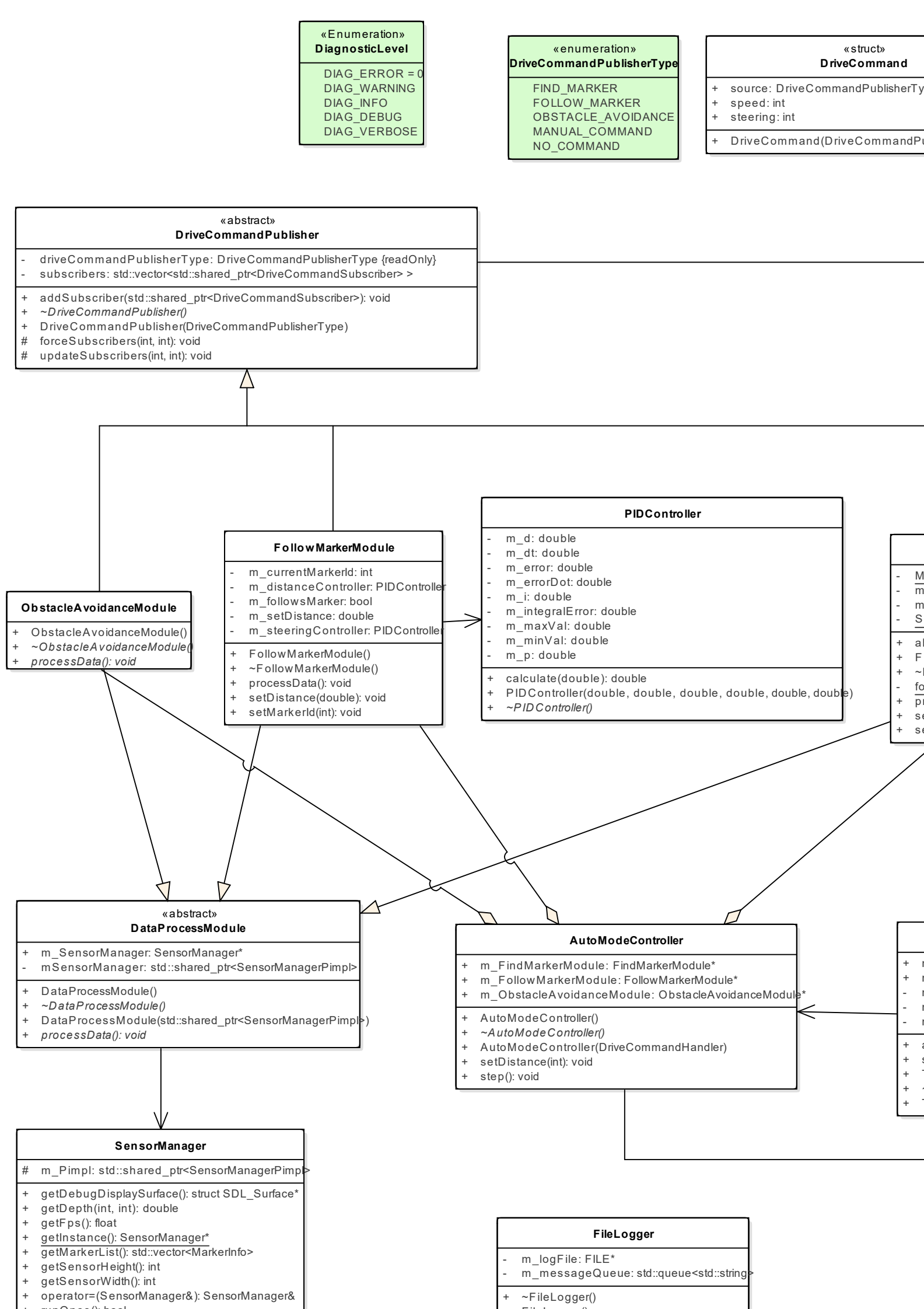
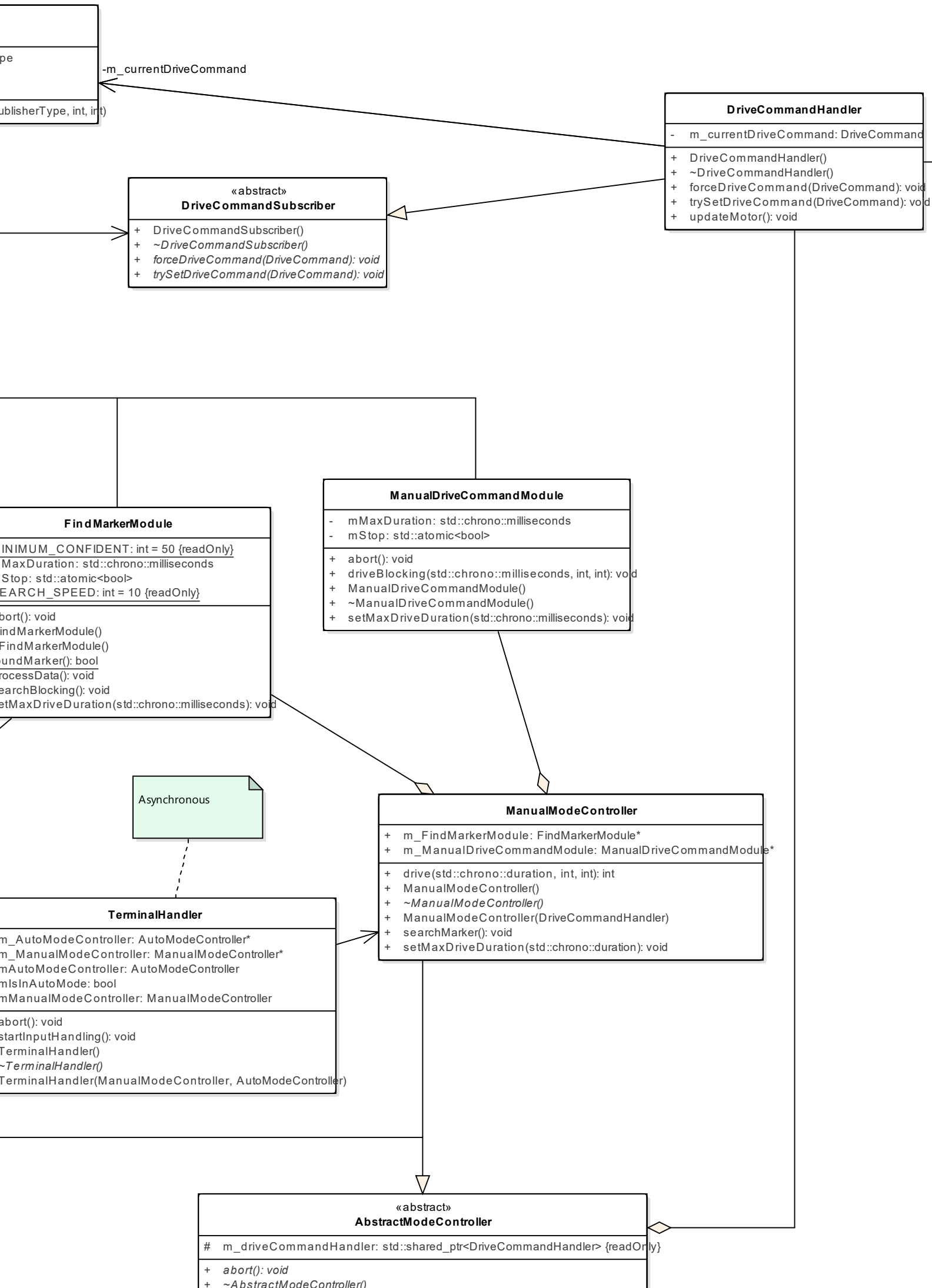
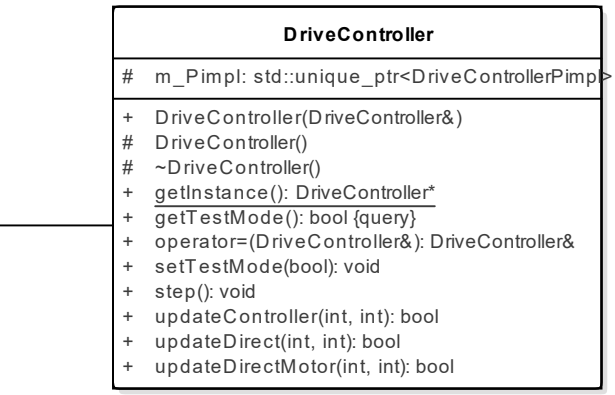
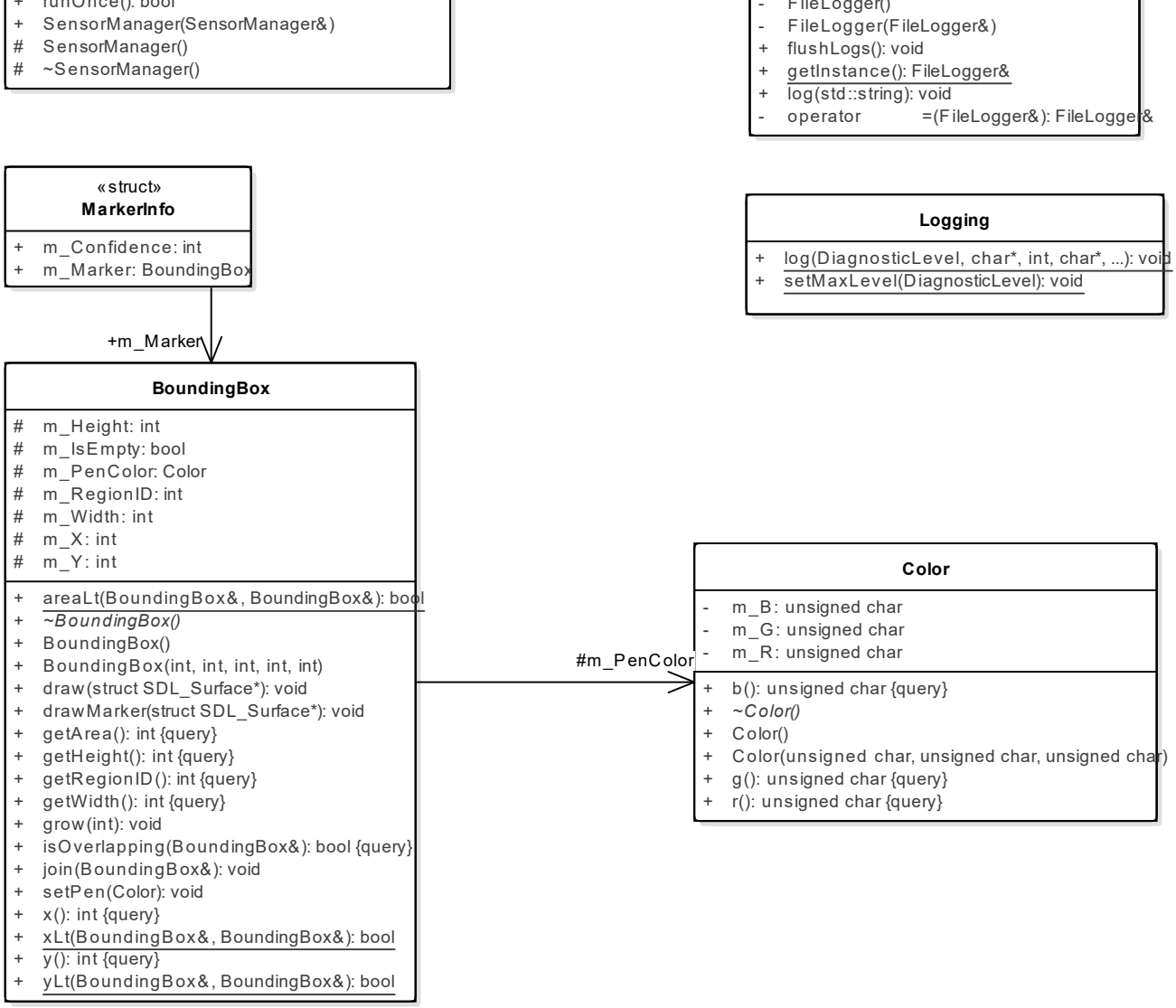
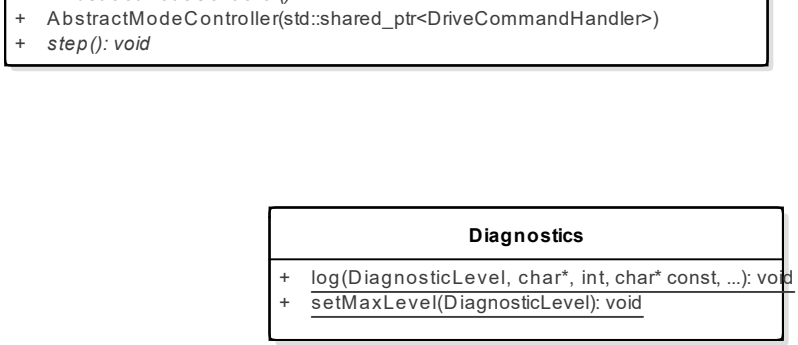

«Enumeration»
DiagnosticLevel
DIAG_ERROR = 0
DIAG_WARNING
DIAG_INFO
DIAG_DEBUG
DIAG_VERBOSE
«enumeration»
DriveCommandPublisherType
FIND_MARKER
FOLLOW_MARKER
OBSTACLE_AVOIDANCE
MANUAL_COMMAND
NO_COMMAND
«struct»
DriveCommand
+ source: DriveCommandPublisherTy
+ speed: int
+ steering: int
+ DriveCommand(DriveCommandPu
«abstract»
DriveCommandPublisher
- driveCommandPublisherType: DriveCommandPublisherType {readOnly}
- subscribers: std::vector<std::shared_ptr<DriveCommandSubscriber> >
+ addSubscriber(std::shared_ptr<DriveCommandSubscriber>): void
+ ~DriveCommandPublisher()
+ DriveCommandPublisher(DriveCommandPublisherType)
# forceSubscribers(int, int): void
# updateSubscribers(int, int): void
PIDController
- m_d: double
- m_dt: double
- m_error: double
- m_errorDot: double
- m_i: double
- m_integralError: double
- m_maxVal: double
- m_minVal: double
- m_p: double
+ calculate(double): double
+ PIDController(double, double, double, double, double, double)
+ ~PIDController()
FollowMarkerModule
- m_currentMarkerId: int
- m_distanceController: PIDController
- m_followsMarker: bool
- m_setDistance: double
- m_steeringController: PIDController
+ FollowMarkerModule()
+ ~FollowMarkerModule()
+ processData(): void
+ setDistance(double): void
+ setMarkerId(int): void
ObstacleAvoidanceModule
+ ObstacleAvoidanceModule()
+ ~ObstacleAvoidanceModule()
+ processData(): void
«abstract»
DataProcessModule
+ m_SensorManager: SensorManager*
- mSensorManager: std::shared_ptr<SensorManagerPimpl>
+ DataProcessModule()
+ ~DataProcessModule()
+ DataProcessModule(std::shared_ptr<SensorManagerPimpl>)
+ processData(): void
AutoModeController
+ m_FindMarkerModule: FindMarkerModule*
+ m_FollowMarkerModule: FollowMarkerModule*
+ m_ObstacleAvoidanceModule: ObstacleAvoidanceModule*
+ AutoModeController()
+ ~AutoModeController()
+ AutoModeController(DriveCommandHandler)
+ setDistance(int): void
+ step(): void
SensorManager
m_Pimpl: std::shared_ptr<SensorManagerPimpl>
+ getDebugDisplaySurface(): struct SDL_Surface*
+ getDepth(int, int): double
+ getFps(): float
+ getInstance(): SensorManager*
+ getMarkerList(): std::vector<MarkerInfo>
+ getSensorHeight(): int
+ getSensorWidth(): int
+ operator=(SensorManager&): SensorManager&
FileLogger
- m_logFile: FILE*
- m_messageQueue: std::queue<std::string>
+ ~FileLogger()


pe
ublisherType, int, int)
-m_currentDriveCommand
DriveCommandHandler
- m_currentDriveCommand: DriveCommand
+ DriveCommandHandler()
+ ~DriveCommandHandler()
+ forceDriveCommand(DriveCommand): void
+ trySetDriveCommand(DriveCommand): void
+ updateMotor(): void
«abstract»
DriveCommandSubscriber
+ DriveCommandSubscriber()
+ ~DriveCommandSubscriber()
+ forceDriveCommand(DriveCommand): void
+ trySetDriveCommand(DriveCommand): void
ManualDriveCommandModule
- mMaxDuration: std::chrono::milliseconds
- mStop: std::atomic<bool>
+ abort(): void
+ driveBlocking(std::chrono::milliseconds, int, int): void
+ ManualDriveCommandModule()
+ ~ManualDriveCommandModule()
+ setMaxDriveDuration(std::chrono::milliseconds): void
FindMarkerModule
INIMUM_CONFIDENT: int = 50 {readOnly}
MaxDuration: std::chrono::milliseconds
Stop: std::atomic<bool>
EARCH_SPEED: int = 10 {readOnly}
bort(): void
indMarkerModule()
FindMarkerModule()
undMarker(): bool
rocessData(): void
earchBlocking(): void
etMaxDriveDuration(std::chrono::milliseconds): void
Asynchronous
ManualModeController
+ m_FindMarkerModule: FindMarkerModule*
+ m_ManualDriveCommandModule: ManualDriveCommandModule*
+ drive(std::chrono::duration, int, int): int
+ ManualModeController()
+ ~ManualModeController()
+ ManualModeController(DriveCommandHandler)
+ searchMarker(): void
+ setMaxDriveDuration(std::chrono::duration): void
TerminalHandler
m_AutoModeController: AutoModeController*
m_ManualModeController: ManualModeController*
mAutoModeController: AutoModeController
mIsInAutoMode: bool
mManualModeController: ManualModeController
abort(): void
startInputHandling(): void
TerminalHandler()
~TerminalHandler()
TerminalHandler(ManualModeController, AutoModeController)
«abstract»
AbstractModeController
m_driveCommandHandler: std::shared_ptr<DriveCommandHandler> {readOnly}
+ abort(): void
+ ~AbstractModeController()


DriveController
m_Pimpl: std::unique_ptr<DriveControllerPimpl>
+ DriveController(DriveController&)
# DriveController()
# ~DriveController()
+ getInstance(): DriveController*
+ getTestMode(): bool {query}
+ operator=(DriveController&): DriveController&
+ setTestMode(bool): void
+ step(): void
+ updateController(int, int): bool
+ updateDirect(int, int): bool
+ updateDirectMotor(int, int): bool


+ runOnce(): bool
+ SensorManager(SensorManager&)
# SensorManager()
# ~SensorManager()
- FileLogger()
- FileLogger(FileLogger&)
+ flushLogs(): void
+ getInstance(): FileLogger&
+ log(std::string): void
- operator =(FileLogger&): FileLogger&
«struct»
MarkerInfo
+ m_Confidence: int
+ m_Marker: BoundingBox
Logging
+ log(DiagnosticLevel, char*, int, char*, ...): void
+ setMaxLevel(DiagnosticLevel): void
+m_Marker
BoundingBox
m_Height: int
m_IsEmpty: bool
m_PenColor: Color
m_RegionID: int
m_Width: int
m_X: int
m_Y: int
+ areaLt(BoundingBox&, BoundingBox&): bool
+ ~BoundingBox()
+ BoundingBox()
+ BoundingBox(int, int, int, int)
+ draw(struct SDL_Surface*): void
+ drawMarker(struct SDL_Surface*): void
+ getArea(): int {query}
+ getHeight(): int {query}
+ getRegionID(): int {query}
+ getWidth(): int {query}
+ grow(int): void
+ isOverlapping(BoundingBox&): bool {query}
+ join(BoundingBox&): void
+ setPen(Color): void
+ x(): int {query}
+ xLt(BoundingBox&, BoundingBox&): bool
+ y(): int {query}
+ yLt(BoundingBox&, BoundingBox&): bool
#m_PenColor
Color
- m_B: unsigned char
- m_G: unsigned char
- m_R: unsigned char
+ b(): unsigned char {query}
+ ~Color()
+ Color()
+ Color(unsigned char, unsigned char, unsigned char)
+ g(): unsigned char {query}
+ r(): unsigned char {query}


+ AbstractModeController(std::shared_ptr<DriveCommandHandler>)
+ step(): void
Diagnostics
+ log(DiagnosticLevel, char*, int, char* const, ...): void
+ setMaxLevel(DiagnosticLevel): void