	class Robot *Implements the robot's functionality. Member variables allow easy sharing of data between different functions.*					**class Map** *State machine that represents the playing area.*
	Line Following / Recovery	**Position tracking**	**Egg processing**	**Propulsion**	**Framework & Communication**	
Member functions	**update_line_sensors**() *Updates line_sensors_history with latest data.* **calculate_demand**() *Carries out control loop operations; outputs desired wheel velocities.* ***recovery()*** *Carries out recovery strategy to attempt to reach a line.*	**update_tracking**() *Reads the motor speeds, and then uses a second-order Euler integrator to infer current position, orientation, speed and path curvature.*	**egg_pickup**() **egg_identify**() **egg_crack**() **egg_inside_identify**() **egg_inside_drop**() **egg_drop**()	**move**(MotorDemand) *Scales the requested motor demand and sends it to the motors via PWM.*	**Robot**(), **~Robot**() *Memory management.* **load_constants**() *Retrieves constants from robot.cfg configuration file, parses command-line options.* **initialise**() *Opens connection to robot; preliminary setup.*	**populate**() *Parses the human-readable playing area representation file and populates the data structure.* **advance**() Changes the state machine state by advancing in one direction.
Data types	union **LineSensors** *Represents a data frame from the I2C light sensor board.* typedef **LineSensorsHistory** *Represents a series of timestamped LineSensors data, stored in a std::vector.*	struct **Tracking** *Represents the robot's kinematic state. Has members to store respectively position, orientation, speed and path curvature.* typedef **TrackingHistory** *Represents a series of timestamped Tracking data, stored in a std::vector.*	enum **EggType** *Contains each of the possible egg types.* enum **EggInsideType**	class **MotorDemand** *Contains the wheel speeds for the two propulsion motors.*	class **LinkError** : Exception *Exception that triggers reconnection & reconfiguration.* class **Vector2d** *A simple 2-dimensional vector.* template<typename T> struct **Timestamp** *Contains a value and an integer number of milliseconds.*	class **Point** *Represents a point of interest in the playing area.* class **Line** *Represents a white line path between two points of interest.*
Key member variables	LineSensorsHistory **line_sensors_history** *Contains all the line sensor values since the beginning of time.* bool **on_line** *Whether the line following algorithm has lost track of the white line.* bool **at_crossroad** *Whether the robot is at a crossroad.*	TrackingHistory **tracking_history** *Contains all the Tracking values since the beginning of time.*	EggType **egg_type** EggInsideType **egg_inside_type**		robot_link **rlink** *Interface to microcontroller functions.* stopwatch **sw**	std::vector<Point> **points** *Points of interest in the playing area.* std::vector<Line> **lines** *White lines connecting the points of interest.* bool **at_point** int **cur_point** int **cur_line** *Current robot location.*

Fig.	**Software subsystem structure**
Drawn by: Antoine Dupuis (ard61)	Accepted by:
Date: 24/02/2016	Date: