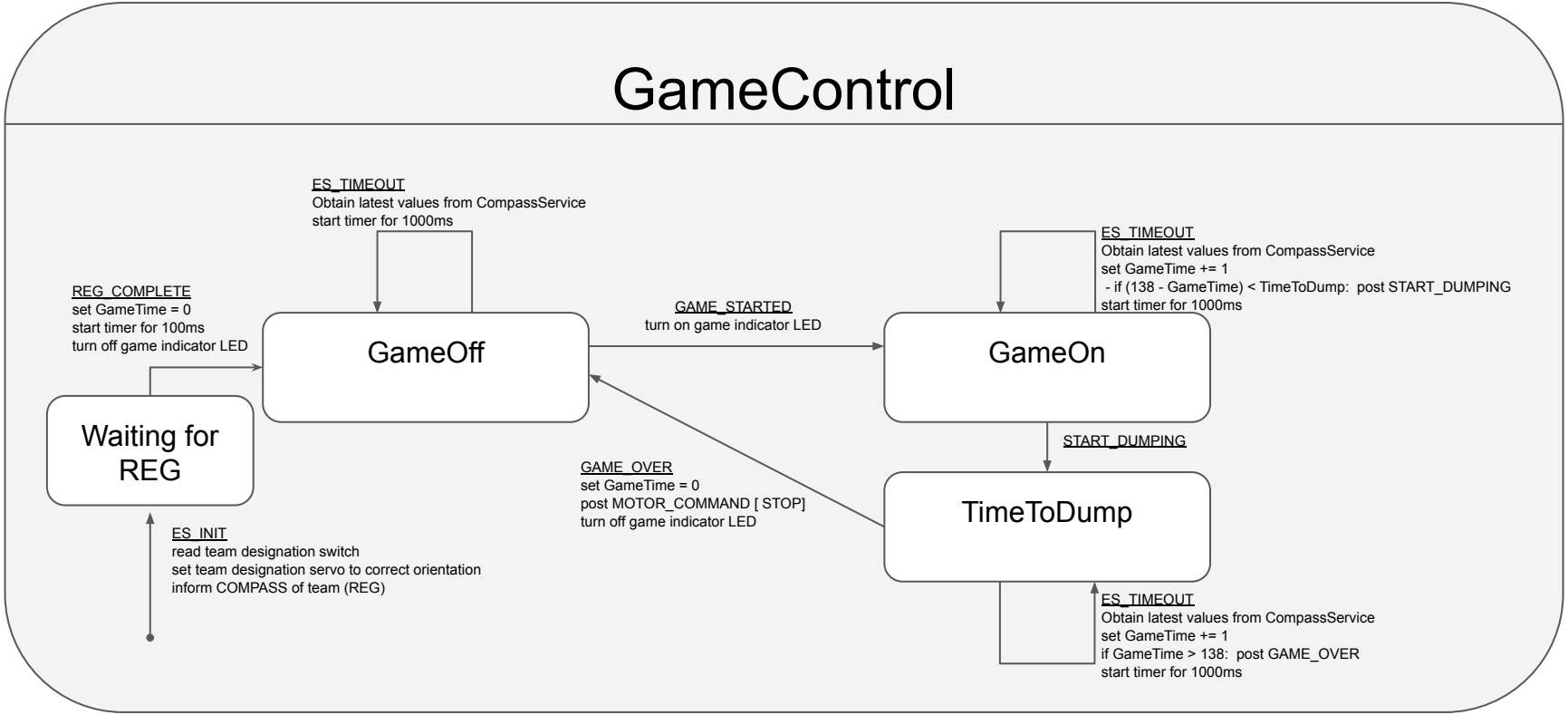
State Diagrams

ME 218B Winter 2019
Team 3
#spacetacocorgis

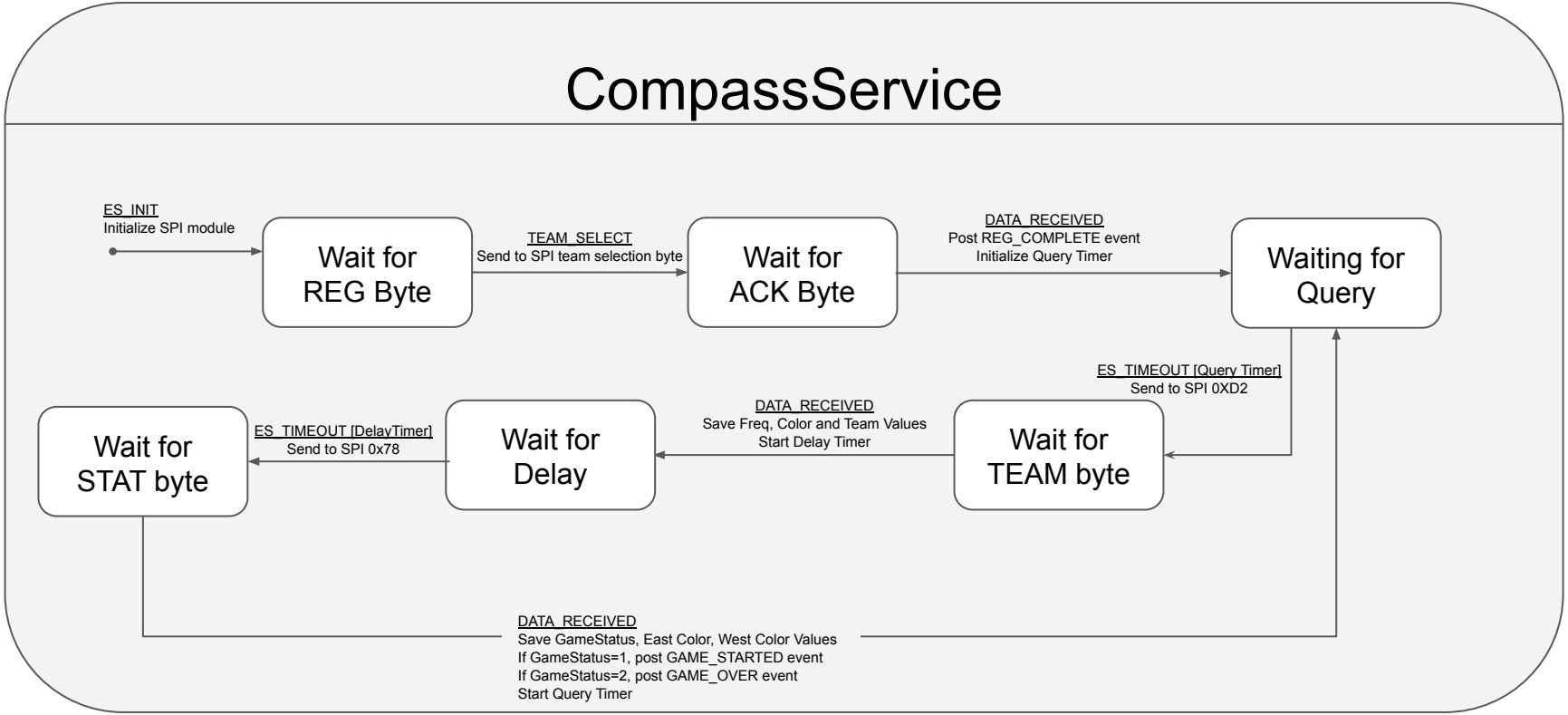
All Services

- [7] GameControlService
  - interprets information obtained from the Compass
  - determines when a game has started and ended
  - stores updated values of recycling frequency and which center is available for us
  - reads the team designation switch
  - keeps a “GameTime”
- [6] CompassService
  - communicates with Compass via SPI
- [5] GameplayService
  - implements our overall strategy
  - turns on and off based on events posted by CompassService
- [4] BallSortingService
  - sorts balls and keeps track of how many we have
  - turns on and off based on events posted by CompassService
- [3] MotorService
  - takes care of all the motor commands
- [2] I2CService
  - updates color sensor readings using the I2C lines with the color sensor
- [1] TapeService
  - keeps track of tape sightings when we are trying to line up with recycling center

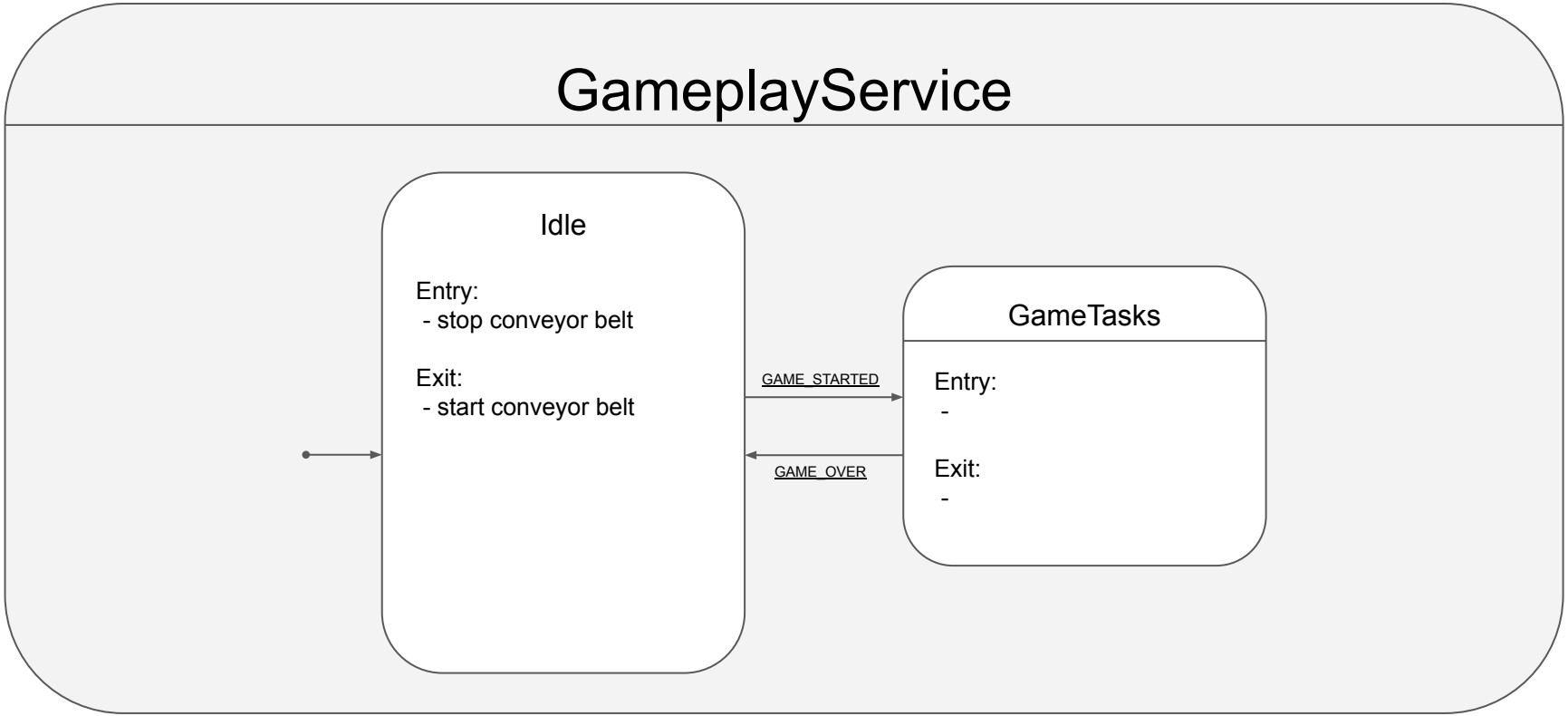
GameControlService



CompassService

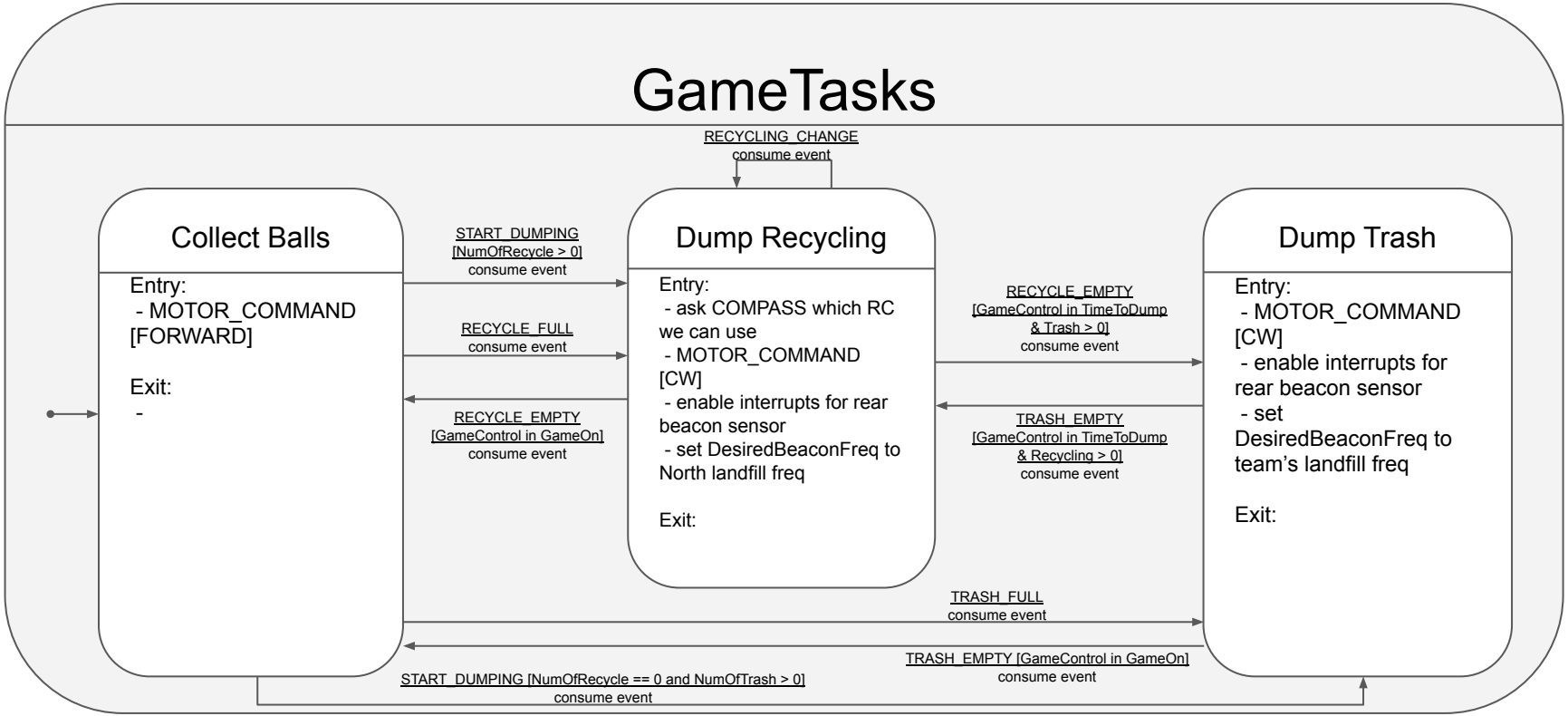


GameplayService



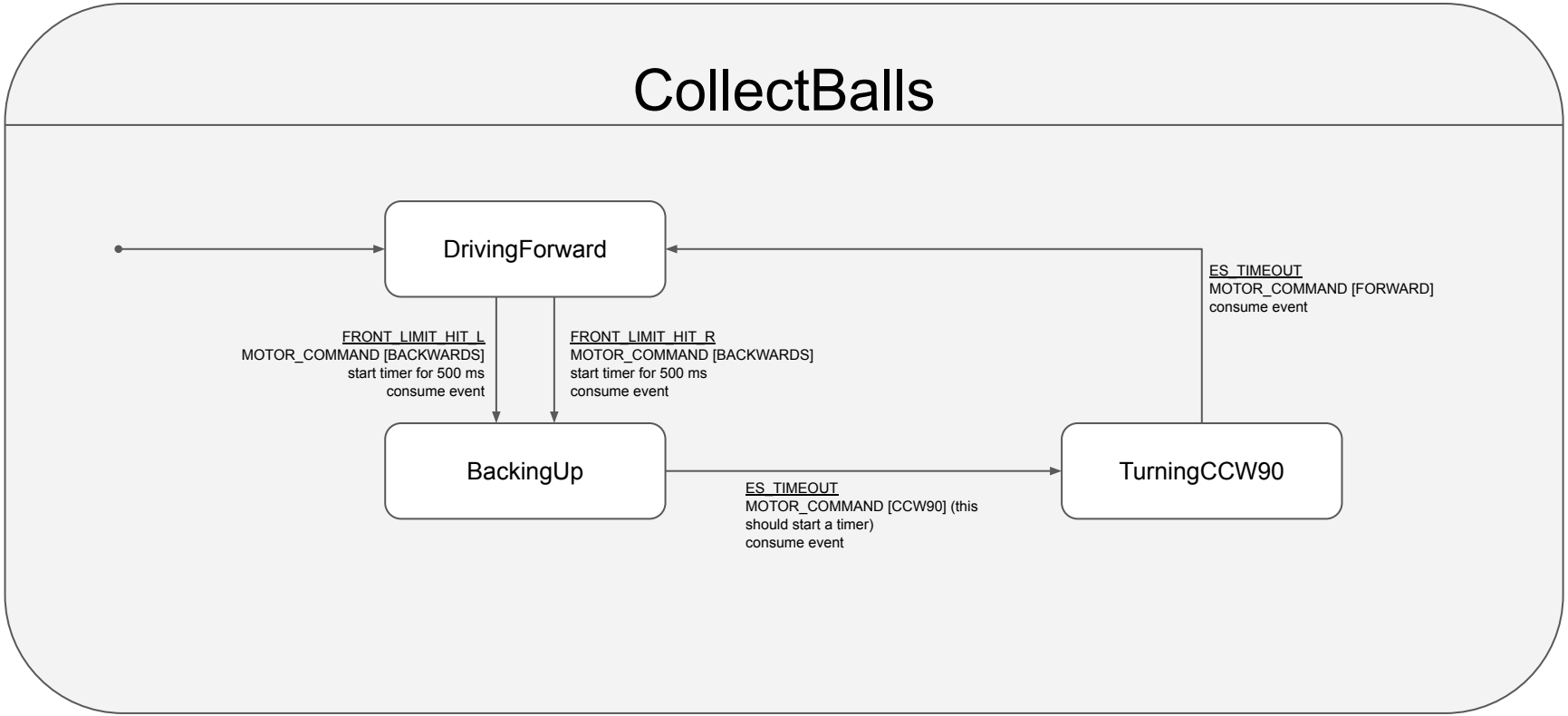
Service 5, Level 0

GameplayService → GameTasks

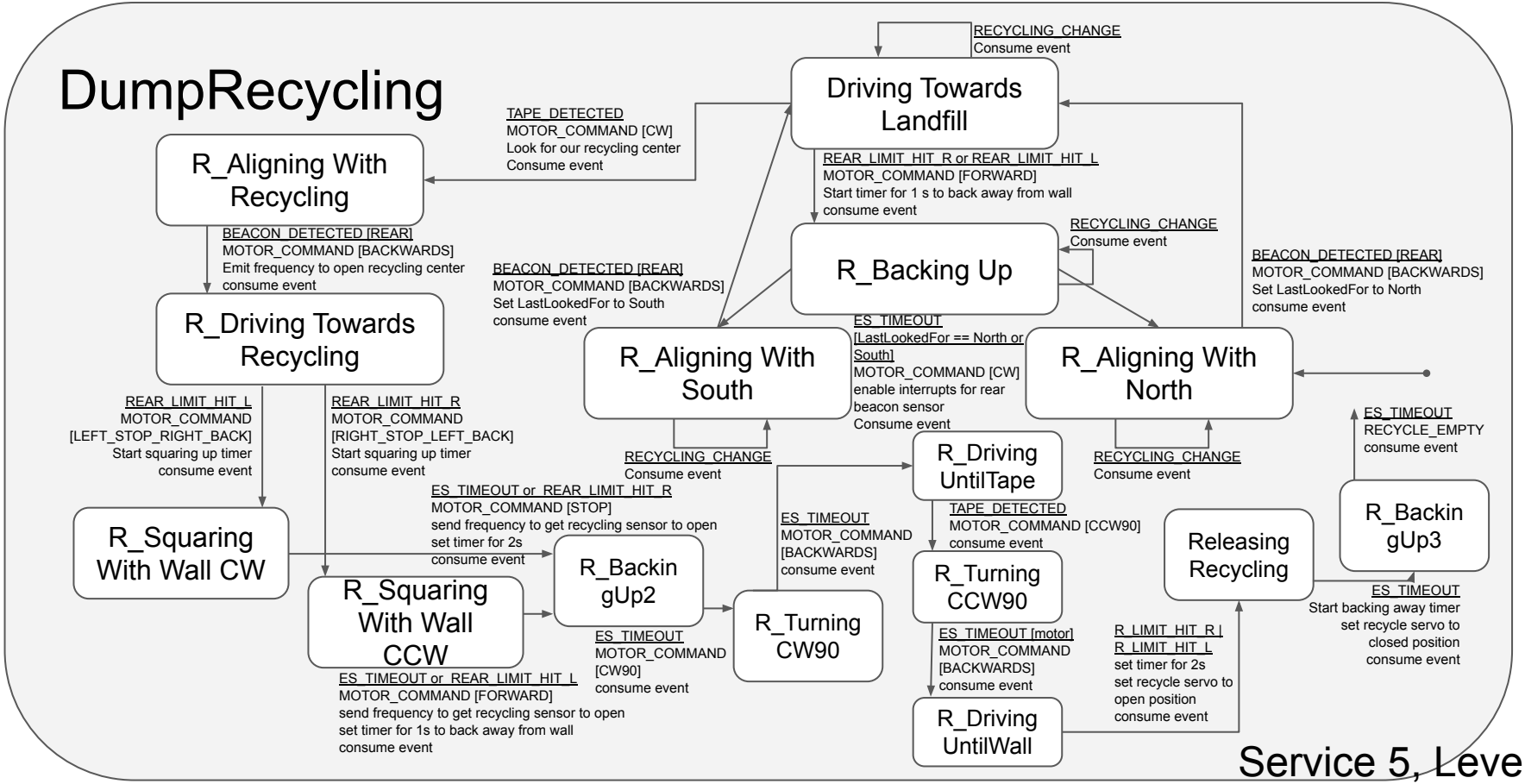


Service 5, Level 1

GameplayService → GameTasks → CollectBalls

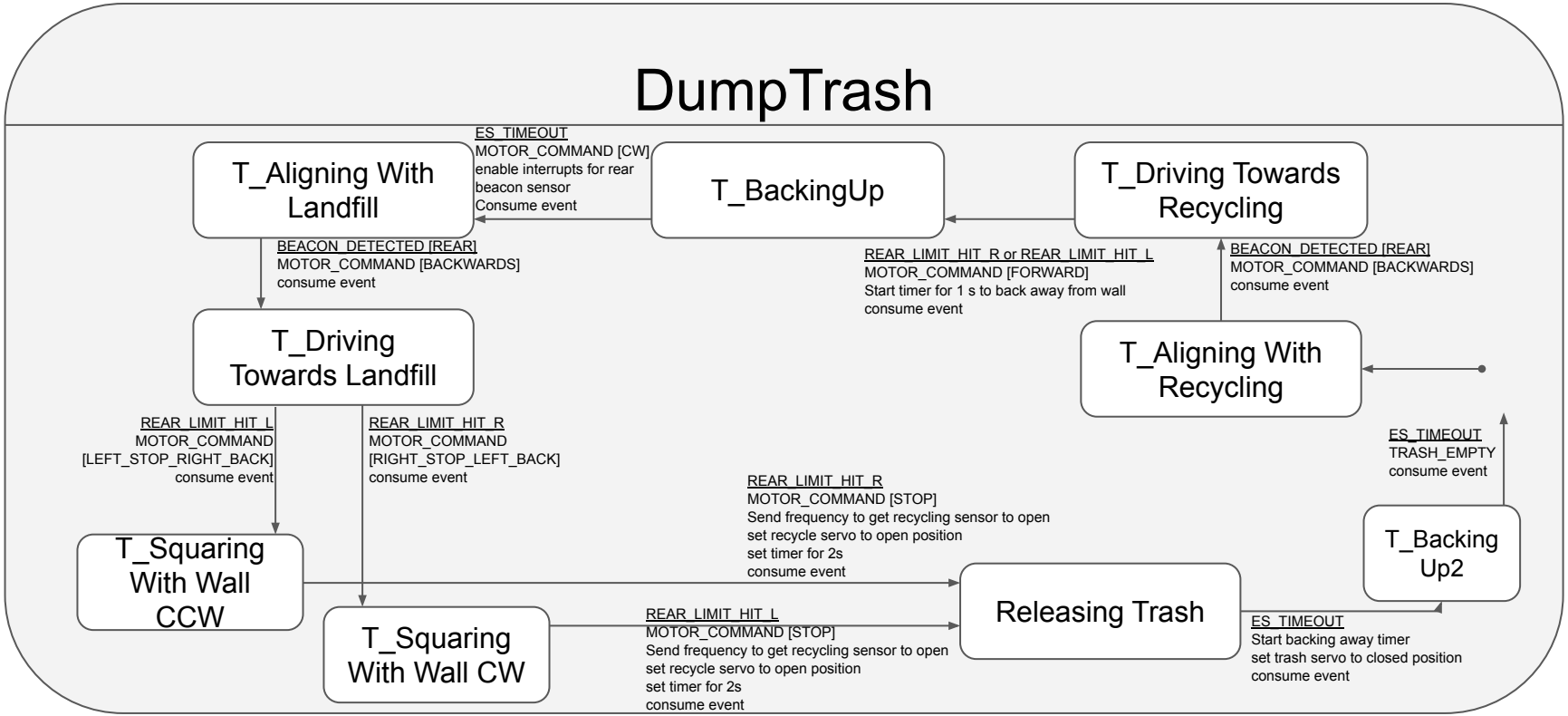


GameplayService → GameTasks → DumpRecycling



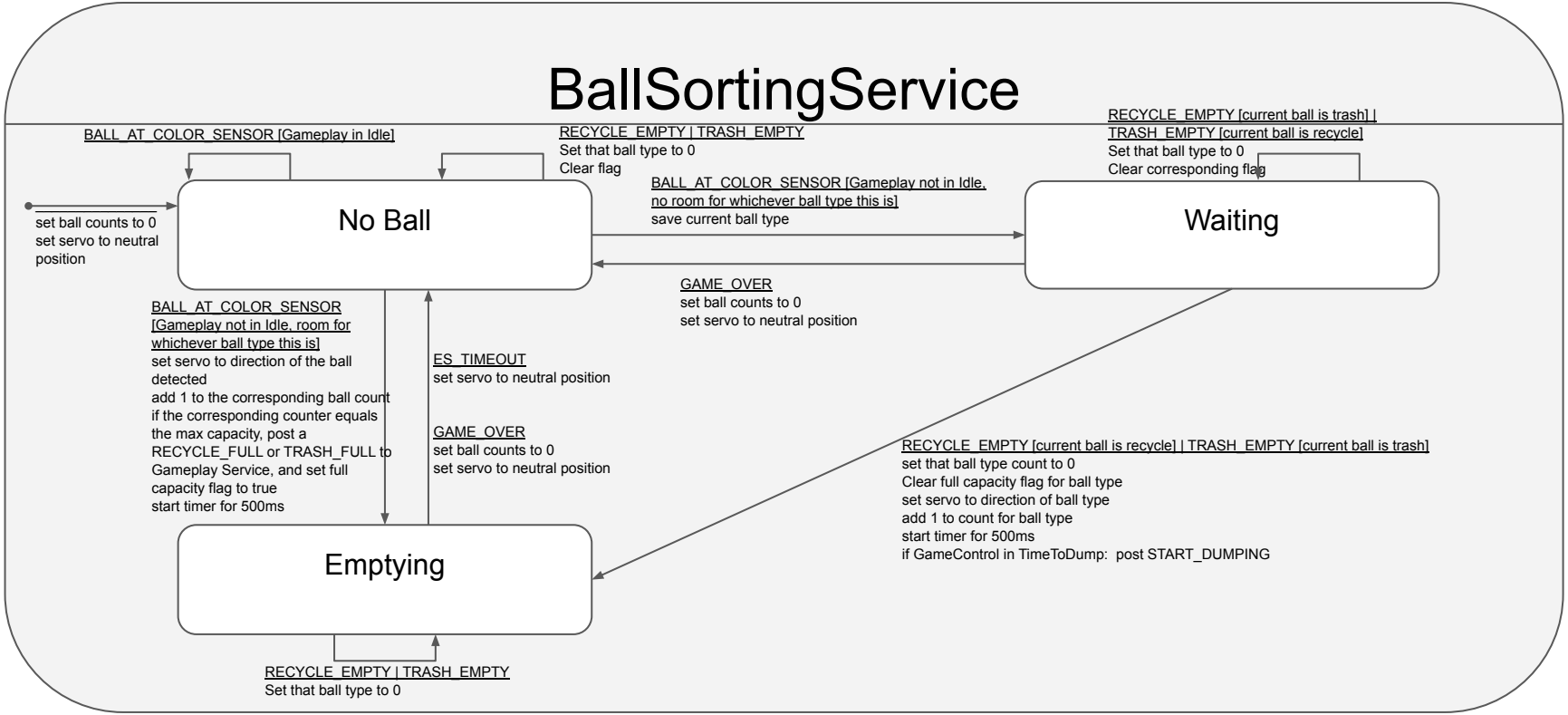
Service 5, Level 2

GameplayService → GameTasks → DumpTrash



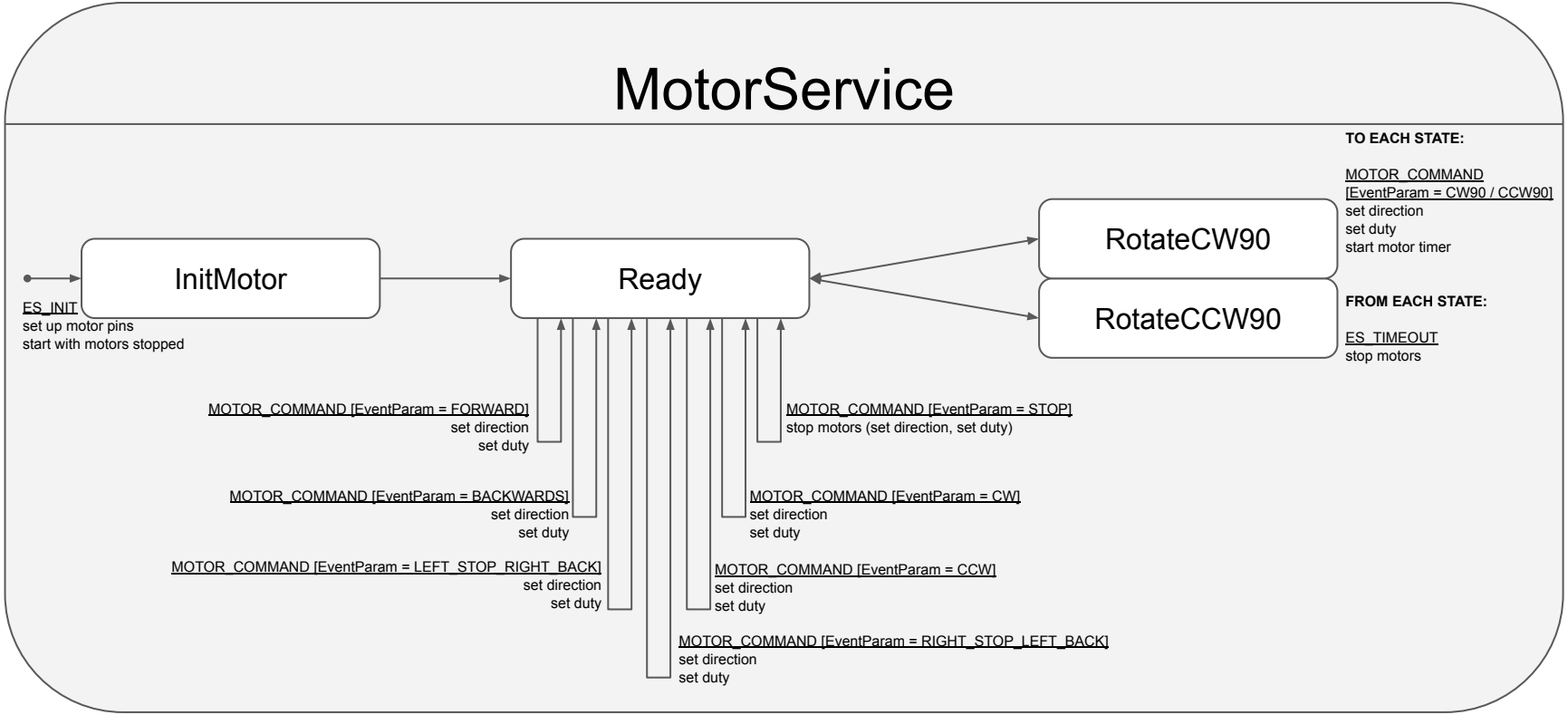
Service 5, Level 2

BallSortingService



This state machine relies on getting an event BALL_AT_COLOR_SENSOR from an event checker If the color sensor detects a clear value above a certain threshold, after a moment in time, it will check if the ball is trash or recycle and post the event BALL_AT_COLOR_SENSOR, and will wait to detect a low color (clear value below 2000) before starting to count to the next ball

MotorService



I2CService

- Has functions for reading current values of the color sensor
  - ClearValue = I2C_GetClearValue();
  - RedValue   = I2C_GetRedValue();
  - GreenValue = I2C_GetGreenValue();
  - BlueValue = I2C_GetBlueValue();

TapeService

- InitializeTapeService
  - Initializes the 3 tape sensor pins
- Check4Tape
  - Returns a TAPE_DETECTED event whenever it sees a rising edge from the tape sensor
  - Uses EventParams of RIGHT, CENTER, and LEFT

BeaconSense

- InitializeBeaconSense
  - initializes 2 pins for input capture
- BeaconRearISR
  - calculates the period
  - gets the desired beacon signal from GameplayService
  - if the period is within that of the desired beacon signal:
    - posts BEACON_DETECTED event with EventParam REAR
    - disables its own interrupts
- BeaconRightISR
  - calculates the period
  - gets the desired beacon signal from GameplayService
  - if the period is within that of the desired beacon signal:
    - posts BEACON_DETECTED event with EventParam RIGHT
    - disables its own interrupts

Module

CollisionModule

- InitializeCollisionDetection
  - initializes 4 pins for GPIO input (limit switches)
  - initializes 1 pin for PWM output (IR emitter for robot detection)
  - initializes 4 pins for input capture (IR sensors)
- RobotSenseRearRightISR, RobotSenseRearLeftISR, RobotSenseFrontRightISR, RobotSenseFrontLeftISR
  - calculates the period
  - if the period is within that of the outputted signal:
    - saves (ES_Timing is fine enough) TimeOfLastRobotSightingRearRight, TimeOfLastRobotSightingRearLeft, TimeOfLastRobotSightingFrontRight, TimeOfLastRobotSightingFrontLeft
- LimitSwitchRearRightChecker, LimitSwitchRearLeftChecker, LimitSwitchFrontRightChecker, LimitSwitchFrontLeftChecker
  - event checker to detect if the limit switches have changed + are pressed down
    - if (CurrentTime - TimeOfLastRobotSightingXX) < 0.5 seconds: post ROBOT_COLLISION with EventParam = (REAR, FRONT)
    - else: post REAR_LIMIT_HIT_L / REAR_LIMIT_HIT_R / FRONT_LIMIT_HIT_L / FRONT_LIMIT_HIT_R

Module