**Prepared By:**  
Kyle Bjornson (100936493)  
Ben Earle (100970237)

**Prepared For:**  
Dr. Gabriel Wainer

Automated Vehicle Control System

Assignment 1 – SYSC 5104 – Fall 2018

# Part I – Description:

## Problem Statement

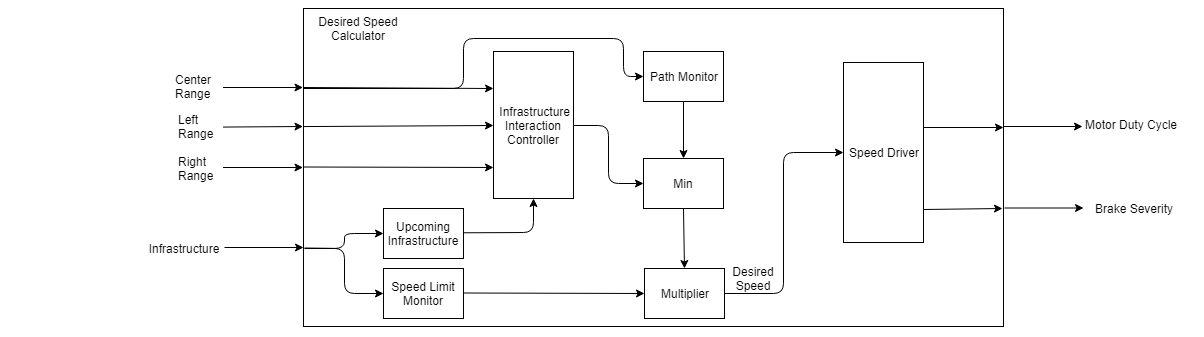
In our fourth-year project we will be creating a self-driving vehicle. In this course, we want to use DEVs to model sensors on a vehicle, create a control system, and simulate its behavior. The vehicle will use a LiDAR (Light Detection and Ranging) range finder, a camera, and vehicle to infrastructure communication to generate inputs. These inputs will be fed into a model that handles the outputs: an electric motor, brakes, and steering.

The model will make decisions based on discrete input events and current state to navigate its environment. The model will steer the vehicle, control its speed, and react to unpredictable stimuli. The sensors will detect obstacles in the vehicles path, oncoming intersections, stop signs, and yield signs.

The model be a simplified representation of normal vehicle operation. For this assignment, the sensor data processing will be done by an external handler that provides our system with a discrete set of inputs for the model. The sensor data manipulation will include components to handle lane following, path finding, and interacting with road infrastructure. The data manipulation is out of the scope of this assignment; however, we hope to implement these blocks using Cell-DEVs in Assignment 2.

## Model Sketch

**Figure 1: Top Level Model Sketch**

**Figure 2: Desired Speed Calculator submodel**

## Component Description

The model will receive inputs from a few external sources. These are a camera, a LiDAR, a GPS for turning information, and some infrastructure signals such as stop lights, stop signs, etc. The model will have outputs that adjust steering, braking intensity, signal lights, and electric motor output.

The LiDAR Data Processor will output a discrete position to the Input Storage block that will store the current state of objects around the vehicle. The Camera Data Processor will turn a stream of images in front of the car into a discrete left or right adjustment indication to the Steering Controller. Both the LiDAR and Camera systems are outside the scope of our model, as discussed in the Problem Statement.

In the Direction Controller, the Steer Controller submodel will adjust the current direction of the front wheels based on inputs from the camera. Also located in this block is the GPS Queue which will store GPS instructions for upcoming turns. When the turn comes, it will turn the signal lights on and request help to turn the vehicle from the sensors. The Turn Request signal will indicate to the Camera and LiDAR that a turn either left or right at an intersection is desired, and they will provide inputs to the model to do so using the existing Direction Controller model.

The Speed Controller model is responsible for accelerating, braking, and maintaining desired speed. It will use the LiDAR and infrastructure signals to determine the desired speed of the vehicle in any given situation. The Desired Speed Calculator will determine the best output to drive an electric motor, brakes, and brake lights.

The Desired Speed Calculator collects and compares data from multiple sources to determine the speed that the vehicle should be travelling. The Center/Left/Right Range signals come from the LiDAR range finder to check for unpredicted objects in the path of the vehicle. The Infrastructure signal will contain events such as an upcoming stop sign or intersection. It will also indicate updated speed limits and suggested speed for corners. These will be handled by the Infrastructure Interaction Controller.

The Min submodel will simply determine which of the preceding blocks has requested the lowest speed. For example, if an object is in the Center Range and the vehicle must come to a stop, this will take precedence over the normal speed limit. All of this will determine the outputs that control the motor and brakes.

## Adaptations to System after Implementation

Implementing the system uncovered important functionalities that were not included in our original model. This includes an odometer as well as a block to calculate our actual speed and feed it back into the system (this will operate like a speedometer input). To keep this model within the scope of assignment 1, the desired speed calculator was simplified into one atomic block. We also removed the input storage block; after modeling the system it was apparent it is not needed. Due to constraints on time we also removed the signal lights, since it is not a part of the systems core functionality.

The speed changes are not smooth as we did not have time to perfect the control system. Our goal was to implement more complex speed management software, but it was simplified due to time constraints. Ideally the messages passed between modules would be structures however we were not able to get this to work. To simplify we implemented the data structures as 16-bit unsigned integers, with the data bit-mapped. Originally, we used 32-bit integers but found the simulator was rounding our inputs. To be safe we are using just 16 bits. This lack of precision causes us to be limited to distances between 0 and 511 meters (truncated). At times this causes us to accelerate faster than required when adjusting speeds over long distances.

We were unable to implement realistic periods for motor and servo PWM voltage output. When testing using real period lengths the simulation was flooded with outputs. As a work around the period was increased to make the system more manageable. We also implemented a top testing model that outputs the duty cycles instead of voltage. This output is easier to understand and evaluate.

There is also a known issue with that causes the model to stop indefinitely at an intersection if it is also waiting for a LiDAR reading. There is also a bug where the yield operation leads to a wait forever. Unfortunatly we did not have the time to resolve these issues. However, the stop sign does work when the LiDAR ranges are clear.

# Part II – DEVS Models:

## ATOMIC: GPS Queue

This block takes its input from the GPS controller and the current speed to calculate the appropriate time to execute a turn. The turn request will be sent to the LiDAR and Camera, so they can adjust and decide the new direction for the car.

**State Variables:**  
**sigma** = , **phase** = Passive;

**NextTurnRequest** = none (TurnDirection  {Left, Right}); /\* Represents the next turn to be executed\*/

**Job-Queue** = Empty (GPSInstruction\*); /\* Contains the GPS instructions waiting for the CPU \*/

**LastSpeed** = 0 (double) /\*Last known speed\*/

**LastDistance** = 0 (double) /\*Last known distance\*/

**Formal specification:**

**X** = { GPSInstruction  <N, {Left, Right}>, CurrentSpeed } /\* input speed is in km/hr, however this block uses m/s, so it will be converted in delta external. \*/

**Y** = { TurnRequest  {Left, Right} }

**S** = { { phase, sigma, GPSInstruction-Queue, LastSpeed, LastDistance} }

**ext** (s={ phase, sigma, GPSInstruction-Queue, LastSpeed, LastDistance}, e, x={GPSInstruction or CurrentSpeed)) {

case x

GPSInstruction:

If (phase == busy) {

Enqueue(x, GPSInstruction-Queue);

Sigma = sigma – e;

} Else {

LastDistance = x.getDistance()

If(LastSpeed == 0) {

Sigma = ;

} else {

sigma = x.getDistance()/LastSpeed;

}

NextTurnRequest = x.getDirection();

phase = busy;

}

CurrentSpeed:

X=x/3.6 /\* This block uses m/s for it’s calculations \*/

if (phase == busy) {

LastDistance = LastDistance – (e\*lastSpeed) /\* calculate the distance remaining \*/

If(x == 0) {

Sigma = ;

} Else {

sigma = LastDistance /(x); /\* Calculate the amount of time required to travel the new distance using the new speed \*/

}

}

lastSpeed = x;

}

**int** (s={ GPSInstruction-Queue, NextTurnRequest, LastSpeed}) {

case phase

busy:

if empty(GPSInstruction-Queue) {

phase = passive;

sigma = ;

} else {

TempGPSInstruction = dequeue(GPSInstruction-Queue);

NextTurnRequest = TempGPSInstruction.getDirection();

LastDistance = TempGPSInstruction.getDistance();

If(LastSpeed == 0) {

Sigma = ;

} else {

sigma = LastDistance /LastSpeed;

}

}

passive: /\* Never happens \*/

}

****s) { send NextTurnRequest to the port out }

## ATOMIC: PWM

**State Variables:**  
**sigma** = , **phase** = Passive;

**CurrentDutyCycle** = 0 (double, from 0-100)

**CurrentPeriod** = 0 (double, positive)

**PWMState** = Off (Boolean)

**Formal specification:**

**X** = {PWMConfig  <double, double> } /\*Duty Cycle, Period \*/

**Y** = { PWMState  {On, Off} }

**S** = { { phase, sigma, CurrentDutyCycle, CurrentPeriod } }

**ext** (s={CurrentDutyCycle, CurrentPeriod}, e, x={PWMConfig}) {

CurrentDutyCycle = PWMConfig.DutyCycle;

CurrentPeriod = PWMConfig.Period;

Case phase:

Passive:

PWMState = Off;

Sigma = CurrentPeriod\*CurrentDutyCycle/100;

Busy:

Sigma = sigma – e;

}

**int** (s={ CurrentDutyCycle, CurrentPeriod}) {

If (CurrentDutyCycle == 0 && CurrentPeriod == 0) {

Passivate();

} else {

If (PWMState == On) {

Sigma = CurrentPeriod\*(1 - CurrentDutyCycle/100)

} else {

Sigma = CurrentPeriod\*CurrentDutyCycle/100;

}

PWMState = !PWMState;

}

}

****s) { send PWMState to the port out }

## ATOMIC: Steer Driver

The Steer Controller will take an input from the Camera block requesting the car to adjust its wheel position to an angle between -25 and +25 degrees. The car’s steering will be controlled by a rack and pinion using a servo to turn the pinion gear. The servo will be controlled using PWM with period of 20ms and its central position will be reached at a duty cycle of 7.5%. The gear ratio will be configured such that when the servo is at -90 degree position (5% duty cycle) the car is turning as sharply as it can to the left and when the servo is at +90 degree position (10% duty cycle) ) the car is turning as sharply as it can to the right. Each degree in steering adjustment is represented by 0.1ms in the outputted pulse or 0.5% in the duty cycle. For this model the speed the wheel is turned at is relative to the current velocity, at 15 km/hr the wheel will be turned from center to max in 1s, this time will scale linearly with speed.

**State Variables:**  
**sigma** = , **phase** = Passive;

**CurrentWheelDirection** = 0 (Integer from -25 to +25); /\* Represents the current wheel direction \*/

**DesiredWheelDirection** = 0 (Integer from -25 to +25); /\* Represents the desired wheel direction\*/

**Formal specification:**

**X** = { WheelDirection  I {-25 to +25}, CurrentSpeed }

**Y** = { ServoDutyCylce  R}

**S** = { { phase, sigma, CurrentWheelDirection, DesiredWheelDirection, speed } }

**ext** (s={phase, sigma,CurrentWheelDirection, DesiredWheelDirection}, e, x= {WheelDirection or CurrentSpeed}) {

case x

WheelDirection:

DesiredWheelDirection = WheelDirection;

If (phase == busy) {

Sigma = sigma – e;

} Else {

Sigma = speed/(15\*25);

phase = busy;

}

CurrentSpeed:

speed = x;

Sigma = sigma – e;

}

**int** (s={phase, sigma,CurrentWheelDirection, DesiredWheelDirection}) {

case phase

busy:

if(CurrentWheelDirection < DesiredWheelDirection) {

CurrentWheelDirection++;

Sigma = speed/(15\*25);

} else if (CurrentWheelDirection > DesiredWheelDirection ) {

CurrentWheelDirection--;

Sigma = speed/(15\*25);

} else {

Passivate();

}

passive: /\* Never happens \*/

}

****s) { send 7.5 + (0.1\*CurrentWheelDirection) to the PWM\_SteerController\_DutyCycle }

## ATOMIC: Speed Driver \*\*\*PENDING REVIEW

**State Variables:**  
**sigma** = , **phase** = Passive;

**CurrentSpeed** = 0 (Integer from 0 to 100); /\* Represents the current speed \*/

**MotorSpeed** = 0 (Integer from 0 to 100); /\* Represents the Speed the motor is being driven to achieve \*/

**BrakeIntensity** = 0 (Ratio from 0 to 1); /\* Represents how much the brakes are depressed \*/

**DesiredSpeed** = 0 (Integer from 0 to 100); /\* Represents the desired speed \*/

**Distance** = 0 (Real, positive); //Distance in meters that velocity should change in

**AccelerationTimeout** = 0 /\*Timeout Required to accelerate to the desired speed \*/

**AccelerationInterval**= 0 /\*Timeout required to accelerate 1km/h. Evenly spaced acc timeout \*/

**Formal specification:**

**X** = { NewDesiredSpeedAndDistance  I {0 to 100, > 0}, NewCurrentSpeed  I {0 to 100}}

**Y** = { PWMConfig  <double, double> , BrakeIntensity} /\*Duty Cycle, Period\*/

**S** = { { phase, sigma, MotorSpeed, BrakeIntensity,DesiredSpeed, CurrentSpeed, AccelerationInterval } }

**ext** (s={phase, sigma,MotorSpeed, BrakeIntensity, DesiredSpeed, CurrentSpeed, AccelerationInterval }, e, x= { NewDesiredSpeedAndDistance, NewCurrentSpeed, }) {

Case X

NewDesiredSpeedAndDistance:

DesiredSpeed = NewDesiredSpeedAndDistance.Speed;

Distance = NewDesiredSpeedAndDistance.Distance;

/\* t seconds = 3600\*(2\*distance in km) / (vf km/h + vi km/h)

\* t seconds in 1km time slices = t / abs(vf – vi) \*/

AccelerationTimeout = (3600\*2\*Distance)/

(NewDesiredSpeedAndDistance.Speed + CurrentSpeed);

AccelerationInterval = AccelerationTimeout/ abs(NewDesiredSpeedAndDistance.Speed - CurrentSpeed);

If (AccelerationInterval < 0.1) { //Max acceleration 0.1s per km/h

AccelerationInterval = 0.1;

}

If (DesiredSpeed < CurrentSpeed) {

//This assumes speed in km/h, and coefficient of friction 0.8.. Dry pavement

BrakeIntensity = (abs(CurrentSpeed – DesiredSpeed)^2)/(200\*Distance)

If (BrakeIntensity > 1) { BrakeIntensity = 1; }

} else {

BrakeIntensity = 0;

}

Sigma = 0; //Deal with all of this right away

NewCurrentSpeed: //An update to current speed, to be used at next timeout

CurrentSpeed = NewCurrentSpeed;

Sigma = sigma – e;

}

**int** (s={phase, sigma,MotorSpeed, BrakeIntensity, DesiredSpeed, CurrentSpeed}) {

If (DesiredSpeed > CurrentSpeed) {

Sigma = AccelerationInterval;

If (MotorSpeed == 0) {

MotorSpeed = CurrentSpeed + 1;

} else {

MotorSpeed++;

}

Else if (DesiredSpeed < CurrentSpeed) {

MotorSpeed = 0; // Coast

Sigma = AccelerationTimeout;

} else {

MotorSpeed = CurrentSpeed; //Deals with “coast” case complete

BrakeIntensity = 0;

Passivate(); /\*Nothing has changed \*/

}

}

****s) {

If (DesiredSpeed == CurrentSpeed) {

Output(DesiredSpeedReached)

}

Output(BrakeIntensity)

Output(MotorSpeed and Period=TBD to PWM\_MotorController \_DutyCycle)

}

## ATOMIC: Desired Speed Calculator \*\*\* PENDING REVIEW

This block was simplified from the original proposal to save time. This block will take inputs and update the desired speed right away, for this reason the assigned sigma is always zero. To simplify the car only deals with 4 way stop signs where it never has the right of way when more than one car is at the intersections.

Brake function = -1/10x^2

**State Variables:**  
**sigma** = , **phase** = Passive;

**CurrentLeftRange** = 0 (Integer from 0 to 50); /\* Represents the distance to an object on the left \*/

**CurrentCenterRange** = 0 (Integer from 0 to 50); /\* Represents the distance to an object in the center \*/

**CurrentRightRange** = 0 (Integer from 0 to 50); /\* Represents the distance to an object on the right\*/

**CurrentSpeedLimit** = 0  N

**CurrentOdometer** = 0  N

**EmergancySpeed** = { Speed  <speed, distance>}

**NextSign** ={ Sign <Type {Stop, Yield, Speed}, OdometerLocation N>}

**Formal specification:**

**X** = { LeftRange , CenterRange , RightRange , Infrastructure {SpeedLimit  N, Sign <Type {Stop, Yield}, Distance N>}, DesiredSpeedReached, CurrentSpeed }

**Y** = { DesiredSpeed  N {0 to 100}}

**S** = { { phase, sigma, CurrentLeftRange, CurrentCenterRange, CurrentRightRange, CurrentSpeedLimit, NextSign, EmergancySpeed} }

**ext** (s= { { phase, sigma, CurrentLeftRange, CurrentCenterRange, CurrentRightRange, CurrentSpeedLimit, NextSign, EmergancySpeed} }, e, x = { LeftRange , CenterRange , RightRange , Infrastructure {Sign <Type {Stop, Yield, <Speed, limit  N >} }, Distance N> }, DesiredSpeedReached, OdometerReading }) {

Sigma = 0;

Phase = busy;

case x

LeftRange:

CurrentLeftRange = x

CenterRange:

CurrentCenterRange = x

brakingDistance = max(CurrentSpeed, DesiredSpeed)^2 /200 - 2

if(CurrentCenterRange =< 2) {

EmergancySpeed = <0,1>

} else if(CurrentCenterRange > brakingDistance) {

EmergancySpeed = <(200 CurrentCenterRange-2 )^1/2, 2>

} else if (EmergencySpeed != none) {

EmergancySpeed = none;

If(NextSign == none) {

DesiredSpeed = <CurrentSpeedLimit, 10>;

} else If(NextSign.type == Speed ) {

DesiredSpeed = <CurrentSpeedLimit, NextSign.OdometerLocation - CurrentOdometer >;

} else If(NextSign.type == Stop ) {

DesiredSpeed = <0, NextSign.OdometerLocation - CurrentOdometer >;

} else If(NextSign.type == Yield ) {

DesiredSpeed = <5, NextSign.OdometerLocation - CurrentOdometer >;

}

}

RightRange:

CurrentRightRange = x;

OdometerReading:

CurrentOdometer = x;

Sign:

If(x.type == Speed) {

CurrentSpeedLimit = x.limit;

DesiredSpeed = <CurrentSpeedLimit, x.distance>;

} else If(x.type == Stop ) {

DesiredSpeed = <0, x.distance>;

} else If(x.type == Yield ) {

DesiredSpeed = <5, x.distance>;

}

NextSign = x;

NextSign.OdometerLocation = CurrentOdometer + x.distance;

DesiredSpeedReached:

If(NextSign != none && EmergancySpeed == none) {

If(NextSign.type == Speed) {

Passivate();

} else If(NextSign.type == stop) {

If( LeftRange > 10 && CenterRange > 10 && RightRange > 10) {

/\* The intersection is clear, continue. \*/

DesiredSpeed = <CurrentSpeedLimit, 10>;

NextSign = none;

Passivate();

} else {

Sigma = 3;

}

} else If(NextSign.type == yield) {

If( LeftRange > 20 && CenterRange > 20 && RightRange > 20) {

/\* The intersection is clear, continue. \*/

NextSign = none;

DesiredSpeed = <CurrentSpeedLimit, 10>;

Passivate();

} else {

DesiredSpeed = <0,1>;

Sigma = 3;

}

}

}

}

**int** (s={ { phase, sigma, CurrentLeftRange, CurrentCenterRange, CurrentRightRange, CurrentSpeedLimit, NextSign, EmergancySpeed}}) {

case phase

busy: /\*The only way

If (NextSign !=none && NextSign.type == stop) {

If ( LeftRange > 10 && CenterRange > 10 && RightRange > 10) {

/\* The intersection is clear, continue. \*/

DesiredSpeed = <CurrentSpeedLimit, 10>;

NextSign = none;

Passivate();

} else {

Sigma = 3;

}

} else If (NextSign != none && NextSign.type == yield) {

If( LeftRange > 20 && CenterRange > 20 && RightRange > 20) {

/\* The intersection is clear, continue. \*/

NextSign = none;

DesiredSpeed = <CurrentSpeedLimit, 10>;

Passivate();

} else {

DesiredSpeed = <0,1>;

Sigma = 3;

}

} else {/\*NextSign is not stop or yield, just passivate\*/

Passivate();

}

passive: /\* Never happens \*/

}

****s) {

If(EmergencySpeed != none ) {

Output(EmergencySpeed);

} else {

Output(DesiredSpeed);

}

}

## ATOMIC: Actual Speed Calculator

This block will be external to our system. It will be used to model the physical response to a given breaking intensity and motor driver signal. For simplicity it will take the motor duty cycle and break intensity as input. Additionally, we will be assuming the velocity of the motor will be represented by the following linear function: v = duty cycle \* 100. Acceleration in this model will be 10 m/s2, constantly. The number of meters traveled slowing 1 km/hr is calculated using the derivative of the stopping distance function used for previous modules: change in m = speed(km/hr)/100. Using this and the previous speed we will calculate the time to slow 1 km/hr and find it is a constant of 0.036 seconds. Calculation and explanation:

*[(100 / CurrentBrakingIntensity) \* (CurrentSpeed)/100]/(CurrentSpeed \* 1/3.6); [] gives us the numbers of meters traveled slowing 1 km, divide this by the speed in meters/second to get the amount of time to slow 1 km/hr*

**State Variables:**  
**sigma** = , **phase** = Passive;

**CurrentSpeed** = 0 (Integer from 0 to 100); /\* Represents the current speed in km/hr from 0 to 100\*/

**CurrentMotorDutyCylce** = 0 (Integer from 0 to 100); /\* Represents the current motor duty cycle in percentage \*/

**CurrentBrakingIntensity** = 0 (Integer from 0 to 100); /\* Represents the current braking intensity in percentage\*/

**Formal specification:**

**X** = { BrakingIntensity  I {0 to 100}, MotorDutyCylce  I {0 to 100}}

**Y** = {ActualSpeed  I}

**S** = { { phase, sigma, CurrentSpeed, CurrentMotorDutyCylce, CurrentBrakingIntensity} }

**ext** (s={ phase, sigma, CurrentSpeed, DesiredSpeed, CurrentMotorDutyCylce, CurrentBrakingIntensity}, e, x= { BrakingIntensity or MotorDutyCylce }) {

case x

BrakingIntensity:

CurrentBrakingIntensity = x;

If (phase == busy) {

Sigma = sigma – e;

} Else {

// Sigma = (100/CurrentBrakingIntensity)\*0.036; /\*time to slow 1km/h times intensity \*/

Sigma = (CurrentBrakingIntensity/100)\*0.036; /\*RIGHT?!?! \*/

phase = busy;

}

MotorDutyCylce:

CurrentMotorDutyCylce = x;

If (phase == busy) {

Sigma = sigma – e;

} Else {

Sigma =0.1; /\*time to increase 1km/h \*/

phase = busy;

}

}

**int** (s={ phase, sigma, CurrentSpeed, CurrentMotorDutyCylce, CurrentBrakingIntensity}) {

case phase

busy:

if(CurrentSpeed < CurrentMotorDutyCylce) {

CurrentSpeed++;

Sigma = 0.1;

} else if (CurrentSpeed > CurrentMotorDutyCylce) {

CurrentSpeed--;

Sigma = (100/CurrentBrakingIntensity)\*0.036;

} else {

Passivate();

}

passive: /\* Never happens \*/

}

****s) { send CurrentSpeed to out}

## ATOMIC: Odometer

This block will be external to our system. It will be used to model the actual speed to calculate meters traveled.

**State Variables:**  
**sigma** = , **phase** = Passive;

**CurrentSecondsPerMeter** = Infinity (Integer from 0 to 100); /\* Represents the current motor duty cycle in percentage \*/

**DistanceTraveled** = 0 (Integer from 0 to 100); /\* Represents the current braking intensity in percentage\*/

**partOfMeterLeft** = 0 (Integer from 0 to 100); /\* Represents the current braking intensity in percentage\*/

**Formal specification:**

**X** = { ActualSpeed  I }

**Y** = { DistanceTraveled  I}

**S** = { { phase, sigma, CurrentSecondsPerMeter, DistanceTraveled, partOfMeterLeft }}

**ext** (s = { phase, sigma, CurrentSecondsPerMeter, DistanceTraveled, partOfMeterLeft }}, e, x= { ActualSpeed }) {

case x

ActualSpeed:

If (phase == busy) {

partOfMeterLeft -= (e/CurrentSecondsPerMeter) /\* the amount of the meter left to be traveled \*/

if(x==0) passivate();

else sigma = partOfMeterLeft \* (1/x);

} Else {

If(partOfMeterLeft = 0) sigma = 1/x;

Else sigma = partOfMeterLeft \* (1/x);

phase = busy;

}

CurrentSecondsPerMeter = 1/x;

}

**int** (s={ phase, sigma, CurrentSpeed, CurrentMotorDutyCylce, CurrentBrakingIntensity}) {

case phase

busy:

DistanceTraveled ++;

Sigma = CurrentSecondsPerMeter;

passive: /\* Never happens \*/

}

****s) { send DistanceTraveled to out}

## COUPLED: Speed Controller

X = { LeftRange , CenterRange , RightRange , Infrastructure {SpeedLimit  N, Sign <Type {Stop, Yield}, Distance N>}, CurrentSpeed, DistanceTraveled};

Y = {motorDutyCycleOut, breakIntensityOut};

D = {desiredSpeedCalculator, speedDriver};

EIC = {

( SpeedController.DistanceTraveled, desiredSpeedCalculator .DistanceTraveled),

( SpeedController. LeftRange, desiredSpeedCalculator . LeftRange),

( SpeedController. CenterRange, desiredSpeedCalculator . CenterRange),

( SpeedController. RightRange, desiredSpeedCalculator . RightRange),

( SpeedController. Infrastructure, desiredSpeedCalculator . Infrastructure),

( SpeedController. CurrentSpeed, desiredSpeedCalculator . CurrentSpeed),

( SpeedController.DistanceTraveled, speedDriver. CurrentSpeed),

};

EOC = {

(speedDriver. breakIntensityOut, SpeedController. breakIntensityOut),

(speedDriver. motorDutyCycleOut, SpeedController. motorDutyCycleOut),

};

IC = {

(speedDriver.desiredSpeedReached desiredSpeedCalculator. desiredSpeedReachedIn),

(desiredSpeedCalculator.desiredSpeedOut, speedDriver.desiredSpeedIn)

};

SELECT = ({speedDriver, desiredSpeedCalculator}) = desiredSpeedCalculator;

## COUPLED: Direction Controller

X = {laneDetection I {-25 to +25}, GPSInstruction, ActualSpeed };

Y = {turnRequest{Left, Right}, servoDutyCycle};

D = {gpsQueue, steerDriver};

EIC = {

(DirectionController. laneDetection, steerDriver.WheelDirection),

( DirectionController. GPSInstruction, gpsQueue. GPSInstruction),

( DirectionController. ActualSpeed, gpsQueue. CurrentSpeed),

( DirectionController. ActualSpeed, steerDriver. CurrentSpeed),

};

EOC = {

(gpsQueue. turnRequest, DirectionController. turnRequest),

(steerDriver. servoDutyCycleOut, DirectionController. servoDutyCycle),

};

IC = {};

SELECT = ({gpsQueue, steerDriver }) = steerDriver;

## COUPLED: Car Controller

X = { LeftRange , CenterRange , RightRange , Infrastructure {SpeedLimit  N, Sign <Type {Stop, Yield}, Distance N>}, CurrentSpeed, DistanceTraveled, laneDetection I {-25 to +25}, GPSInstruction};

Y = {motorDutyCycleOut, breakIntensityOut, turnRequest{Left, Right}, servoDutyCycle };

D = {directionController, speedController};

EIC = {

( CarController.DistanceTraveled, speedController.DistanceTraveled),

( CarController. LeftRange, speedController.LeftRange),

( CarController. CenterRange, speedController.CenterRange),

( CarController. RightRange, speedController.RightRange),

( CarController. Infrastructure, speedController.Infrastructure),

( CarController. CurrentSpeed, speedController.CurrentSpeed),

( CarController. CurrentSpeed, directionController.CurrentSpeed),

(CarController. laneDetection, directionController. laneDetection),

( CarController. GPSInstruction, directionController. GPSInstruction),

};

EOC = {

(speedController. breakIntensityOut, CarController. breakIntensityOut),

(speedController. motorDutyCycleOut, CarController.motorDutyCycleOut),

(directionController. turnRequest, CarController. turnRequest),

(directionController. servoDutyCycleOut, CarController. servoDutyCycle),

};

IC = {};

SELECT = ({directionController, speedController }) = speedController;

## COUPLED: Top Model

X = { LeftRange , CenterRange , RightRange , Infrastructure {SpeedLimit  N, Sign <Type {Stop, Yield}, Distance N>}, laneDetection I {-25 to +25}, GPSInstruction};

Y = {motorVoltage, servoVoltage, turnRequest{Left, Right}, brakeIntensity};

D = {carController, motorPwm, servoPwm, odometer, actualSpeedCalculator};

EIC = {

( Top. LeftRange, carController.LeftRange),

( Top. CenterRange, carController.CenterRange),

( Top. RightRange, carController.RightRange),

( Top. Infrastructure, carController.Infrastructure),

(Top. laneDetection, carController. laneDetection),

( Top. GPSInstruction, carController. GPSInstruction),

};

EOC = {

(servoPwm. PWMState, Top.servoVoltage),

(carController. turnRequest, Top. turnRequest),

(carController.breakIntensityOut, Top. breakIntensity),

(motorPwm. PWMState, Top.motorVoltage),

};

IC = {

( carController.DistanceTraveled, odometer.DistanceTraveled),

(actualSpeedCalculator.actualSpeed, carController.CurrentSpeed),

(actualSpeedCalculator.actualSpeed, odometer.CurrentSpeed),

(carController.breakIntensityOut, actualSpeedCalculator. CurrentBreakIntensity),

(carController.breakIntensityOut, Top. breakIntensity),

(carController.MotorDutyCylce, actualSpeedCalculator. CurrentDutyCycle),

(carController.MotorDutyCylce, motorPwm.CurrentDutyCycle),

(carController. servoDutyCycle, servoPwm.CurrentDutyCycle),

};

SELECT = ({carController, motorPwm, servoPwm, odometer, actualSpeedCalculator}) = carController;

({motorPwm, servoPwm, odometer, actualSpeedCalculator}) = servoPwm;

({motorPwm, odometer, actualSpeedCalculator}) = motorPwm;

({odometer, actualSpeedCalculator}) = actualSpeedCalculator;

# Part 3 – Implementation and Validation

This section will include an overview of the testing done for each model and the results.

## PWM Atomic Model

|  |  |
| --- | --- |
| *Model Description* | This model is given a duty cycle, and a period specified before simulation and produces a digital signal output |
| *Test Case* | 1. At 5 seconds, set duty cycle to 25% Output will be high for .5 seconds, and low for 1.5 seconds with the period below. 2. At 30 seconds, change the duty cycle to 50%. |
| *Input Events* | 00:00:05:00 dutyCycleIn 25  00:00:30:00 dutyCycleIn 50 |
| *Coupled Model Layout* | [top]  components : pwm@Pwm  out : pwmStateOut  in : dutyCycleIn  Link : dutyCycleIn dutyCycleIn@pwm  Link : pwmStateOut@pwm pwmStateOut  [pwm]  period : 2 |
| *Simulation Time* | 1 Minute |
| *Output* | 00:00:05:000 pwmstateout 1  00:00:05:500 pwmstateout 0  00:00:07:000 pwmstateout 1  00:00:07:500 pwmstateout 0  00:00:09:000 pwmstateout 1  00:00:09:500 pwmstateout 0  00:00:11:000 pwmstateout 1  00:00:11:500 pwmstateout 0  … … …  00:00:29:000 pwmstateout 1  00:00:29:500 pwmstateout 0  00:00:30:500 pwmstateout 1 //Duty cycle changes here  00:00:31:500 pwmstateout 0  00:00:32:500 pwmstateout 1  00:00:33:500 pwmstateout 0  00:00:34:500 pwmstateout 1  00:00:35:500 pwmstateout 0  00:00:36:500 pwmstateout 1  00:00:37:500 pwmstateout 0  00:00:38:500 pwmstateout 1  00:00:39:500 pwmstateout 0  … … … |

## Steer Driver Atomic Model

|  |  |
| --- | --- |
| *Model Description* | This model calculates an output duty cycle for a steering servo based on a desired direction and current speed (Steering less intense at higher speeds)  Servo ‘center’ = 7.5% duty cycle. |
| *Test Case* | 1. Set speed to 100, then change wheel direction to +10 at 10 seconds. 2. Reset wheel direction at 30. And change speed to 10.. 3. Change direction again at 1:10 but now with the lower speed, observe the scaling. |
| *Input Events* | 00:00:5:00 speedIn 100  00:00:10:00 wheelDirectionIn 10  00:00:25:00 speedIn 10  00:00:30:00 wheelDirectionIn 0  00:01:10:00 wheelDirectionIn 10 |
| *Coupled Model Layout* | [top]  components : steerDriver@SteerDriver  out : servoDutyCycle  in : wheelDirectionIn  in : speedIn  Link : wheelDirectionIn wheelDirectionIn@SteerDriver  Link : speedIn speedIn@SteerDriver  Link : servoDutyCycle@SteerDriver servoDutyCycle |
| *Simulation Time* | 1:30 minutes |
| *Output* | 00:00:10:074 servodutycycle 7.5 //New direction  00:00:10:148 servodutycycle 7.6  00:00:10:222 servodutycycle 7.7  00:00:10:296 servodutycycle 7.8  00:00:10:370 servodutycycle 7.9  00:00:10:444 servodutycycle 8  00:00:10:518 servodutycycle 8.1  00:00:10:592 servodutycycle 8.2  00:00:10:666 servodutycycle 8.3  00:00:10:740 servodutycycle 8.4  00:00:10:814 servodutycycle 8.5  //Bring direction back to straight  00:00:30:007 servodutycycle 8.5  00:00:30:014 servodutycycle 8.4  00:00:30:021 servodutycycle 8.3  00:00:30:028 servodutycycle 8.2  00:00:30:035 servodutycycle 8.1  00:00:30:042 servodutycycle 8  00:00:30:049 servodutycycle 7.9  00:00:30:056 servodutycycle 7.8  00:00:30:063 servodutycycle 7.7  00:00:30:070 servodutycycle 7.6  00:00:30:077 servodutycycle 7.5  //Change direction, same as the first but at a lower speed (Time to change is faster)  00:01:10:007 servodutycycle 7.5  00:01:10:014 servodutycycle 7.6  00:01:10:021 servodutycycle 7.7  00:01:10:028 servodutycycle 7.8  00:01:10:035 servodutycycle 7.9  00:01:10:042 servodutycycle 8  00:01:10:049 servodutycycle 8.1  00:01:10:056 servodutycycle 8.2  00:01:10:063 servodutycycle 8.3  00:01:10:070 servodutycycle 8.4  00:01:10:077 servodutycycle 8.5 |

## Odometer Atomic Model

|  |  |
| --- | --- |
| *Model Description* | This model is given the current speed and outputs the distance travelled in meters |
| *Test Case* | 1. Set speed to 50 km/h at t=0 and observe 1 meter travelled every 0.071 seconds (t = distance/speed = 1m /13.888 m/s = 0.071 seconds) 2. At 10 seconds, set speed to 100 and observe timeout half of what it was at 50km/h 3. At 20 seconds reset speed to 50 and check consistency 4. At 30 seconds set speed to 0 and observe no distance travelled anymore. |
| *Input Events* | 00:00:00:00 speedIn 50  00:00:10:00 speedIn 100  00:00:20:00 speedIn 50  00:00:30:00 speedIn 0 |
| *Coupled Model Layout* | [top]  components : odometer@Odometer  out : distanceTraveled  in : speedIn  Link : speedIn speedIn@odometer  Link : distanceTraveled@odometer distanceTraveled |
| *Simulation Time* | 31 seconds |
| *Output* | //Speed = 50, 0.071 seconds per meter travelled  00:00:00:071 distancetraveled 1  00:00:00:142 distancetraveled 2  00:00:00:213 distancetraveled 3  00:00:00:284 distancetraveled 4  … … …  00:00:09:727 distancetraveled 137  00:00:09:798 distancetraveled 138  00:00:09:869 distancetraveled 139  00:00:09:940 distancetraveled 140  //New Speed = 100, 0.035 seconds per meter travelled now  00:00:10:005 distancetraveled 141  00:00:10:040 distancetraveled 142  00:00:10:075 distancetraveled 143  00:00:10:110 distancetraveled 144  … … …  00:00:19:840 distancetraveled 422  00:00:19:875 distancetraveled 423  00:00:19:910 distancetraveled 424  00:00:19:945 distancetraveled 425  00:00:19:980 distancetraveled 426  //New Speed = 50, 0.071 seconds per meter again  00:00:20:031 distancetraveled 427  00:00:20:102 distancetraveled 428  00:00:20:173 distancetraveled 429  00:00:20:244 distancetraveled 430  00:00:20:315 distancetraveled 431  00:00:20:386 distancetraveled 432  … … …  00:00:29:261 distancetraveled 557  00:00:29:332 distancetraveled 558  00:00:29:403 distancetraveled 559  00:00:29:474 distancetraveled 560  00:00:29:545 distancetraveled 561  00:00:29:616 distancetraveled 562  00:00:29:687 distancetraveled 563  00:00:29:758 distancetraveled 564  00:00:29:829 distancetraveled 565  00:00:29:900 distancetraveled 566  00:00:29:971 distancetraveled 567  // New speed = 0, no outputs as not travelling any distance |

## Actual Speed Calculator Atomic Model

|  |  |
| --- | --- |
| *Model Description* | This model calculates the current speed of the vehicle given information about how the motor is being driven and if/how much the brakes are applied. It outputs an actual speed to be used by other models. It is meant to be external to the system. |
| *Test Case* | 1. Replicate an acceleration from 0 to 20 km/h 2. Replicate a brake from 20 km/h to 0 |
| *Input Events* | //Replicate speed control inputs coming in at regular intervals. (Intervals in reality are smaller than 1 second)  00:00:05:00 motorDutyCycleIn 1  00:00:06:00 motorDutyCycleIn 2  00:00:07:00 motorDutyCycleIn 3  00:00:08:00 motorDutyCycleIn 4  00:00:09:00 motorDutyCycleIn 5  00:00:10:00 motorDutyCycleIn 6  00:00:11:00 motorDutyCycleIn 7  00:00:12:00 motorDutyCycleIn 8  00:00:13:00 motorDutyCycleIn 9  00:00:14:00 motorDutyCycleIn 10  00:00:15:00 motorDutyCycleIn 11  00:00:16:00 motorDutyCycleIn 12  00:00:17:00 motorDutyCycleIn 13  00:00:18:00 motorDutyCycleIn 14  00:00:19:00 motorDutyCycleIn 15  00:00:20:00 motorDutyCycleIn 16  00:00:21:00 motorDutyCycleIn 17  00:00:22:00 motorDutyCycleIn 18  00:00:23:00 motorDutyCycleIn 19  00:00:24:00 motorDutyCycleIn 20  //Replicate a brake, where motor stops being driven, and brakes are applied 10%  00:00:25:00 motorDutyCycleIn 0  00:00:26:00 brakeIntensityIn 0.10 |
| *Coupled Model Layout* | [top]  components : actualSpeedCalc@ActualSpeedCalc  out : actualSpeed  in : brakeIntensityIn motorDutyCycleIn  Link : brakeIntensityIn brakeIntensityIn@actualSpeedCalc  Link : motorDutyCycleIn motorDutyCycleIn@actualSpeedCalc  Link : actualSpeed@actualSpeedCalc actualSpeed |
| *Simulation Time* | 30 seconds |
| *Output* | // Start accelerating. Speed increases 1 km/h every 0.1s (constant max acceleration always for simplification of the model)  00:00:05:000 actualspeed 0  00:00:05:100 actualspeed 1  00:00:06:100 actualspeed 2  00:00:07:100 actualspeed 3  00:00:08:100 actualspeed 4  00:00:09:100 actualspeed 5  00:00:10:100 actualspeed 6  00:00:11:100 actualspeed 7  00:00:12:100 actualspeed 8  00:00:13:100 actualspeed 9  00:00:14:100 actualspeed 10  00:00:15:100 actualspeed 11  00:00:16:100 actualspeed 12  00:00:17:100 actualspeed 13  00:00:18:100 actualspeed 14  00:00:19:100 actualspeed 15  00:00:20:100 actualspeed 16  00:00:21:100 actualspeed 17  00:00:22:100 actualspeed 18  00:00:23:100 actualspeed 19  00:00:24:100 actualspeed 20 //Speed constant for a second…  //Then braking begins. Braking time is calculated with the provided braking intensity  00:00:26:359 actualspeed 19  00:00:26:718 actualspeed 18  00:00:27:077 actualspeed 17  00:00:27:436 actualspeed 16  00:00:27:795 actualspeed 15  00:00:28:154 actualspeed 14  00:00:28:513 actualspeed 13  00:00:28:872 actualspeed 12  00:00:29:231 actualspeed 11  00:00:29:590 actualspeed 10  00:00:29:949 actualspeed 9  00:00:30:308 actualspeed 8  00:00:30:667 actualspeed 7  00:00:31:026 actualspeed 6  00:00:31:385 actualspeed 5  00:00:31:744 actualspeed 4  00:00:32:103 actualspeed 3  00:00:32:462 actualspeed 2  00:00:32:821 actualspeed 1  00:00:33:180 actualspeed 0 //Braking complete. |

## Speed Driver/Actual Speed Calculator/Odometer Test Coupled Model

|  |  |
| --- | --- |
| *Model Description* | This coupled model is exclusively for testing the functionality of the speed driver. Values from the desired speed calculator are stubbed, but the rest of the test relies on the actual speed calculator and the odometer. |
| *Test Case* | 1. Accelerate to 20 km/h in ~500 meters 2. Decelerate to 0 km/h in 500 meters |
| *Input Events* | //Bit mapped 16 bit values: 7 MSB = speed, 9 LSB = distance  00:00:10:000 desiredSpeedIn 10751  00:05:00:000 desiredSpeedIn 500 |
| *Coupled Model Layout* | [top]  components : speedDriver@SpeedDriver  components : actualSpeedCalc@ActualSpeedCalc  components : odometer@Odometer  out : motorSpeedOut brakeIntensityOut actualSpeed distanceTraveledOut  in : desiredSpeedIn  Link : desiredSpeedIn desiredSpeedIn@speedDriver  Link : actualSpeed@ActualSpeedCalc currentSpeedIn@speedDriver  Link : motorSpeedOut@speedDriver motorDutyCycleIn@ActualSpeedCalc  Link : brakeIntensityOut@speedDriver brakeIntensityIn@ActualSpeedCalc  Link : actualSpeed@ActualSpeedCalc actualSpeed  Link : motorSpeedOut@speedDriver motorSpeedOut  Link : brakeIntensityOut@speedDriver brakeIntensityOut  Link : actualSpeed@ActualSpeedCalc speedIn@Odometer  Link : distanceTraveled@Odometer distanceTraveledOut  [speedDriver]  [actualSpeedCalc] |
| *Simulation Time* | 8 minutes |
| *Output* | // Accleration to 20 km/h  00:00:10:354 motorspeedout 1  00:00:10:354 actualspeed 0  00:00:10:454 actualspeed 1  00:00:10:708 motorspeedout 2  00:00:10:808 actualspeed 2  00:00:11:062 motorspeedout 3  00:00:11:162 actualspeed 3  00:00:11:416 motorspeedout 4  00:00:11:516 actualspeed 4  00:00:11:770 motorspeedout 5  00:00:11:870 actualspeed 5  00:00:11:881 distancetraveledout 1  00:00:12:124 motorspeedout 6  00:00:12:224 actualspeed 6  00:00:12:478 motorspeedout 7  00:00:12:538 distancetraveledout 2  00:00:12:578 actualspeed 7  00:00:12:832 motorspeedout 8  00:00:12:932 actualspeed 8  00:00:13:042 distancetraveledout 3  00:00:13:186 motorspeedout 9  00:00:13:286 actualspeed 9  00:00:13:469 distancetraveledout 4  00:00:13:540 motorspeedout 10  00:00:13:640 actualspeed 10  00:00:13:846 distancetraveledout 5  00:00:13:894 motorspeedout 11  00:00:13:994 actualspeed 11  00:00:14:186 distancetraveledout 6  00:00:14:248 motorspeedout 12  00:00:14:348 actualspeed 12  00:00:14:499 distancetraveledout 7  00:00:14:602 motorspeedout 13  00:00:14:702 actualspeed 13  00:00:14:791 distancetraveledout 8  00:00:14:956 motorspeedout 14  00:00:15:056 actualspeed 14  00:00:15:067 distancetraveledout 9  00:00:15:310 motorspeedout 15  00:00:15:324 distancetraveledout 10  00:00:15:410 actualspeed 15  00:00:15:569 distancetraveledout 11  00:00:15:664 motorspeedout 16  00:00:15:764 actualspeed 16  00:00:15:806 distancetraveledout 12  00:00:16:018 motorspeedout 17  00:00:16:030 distancetraveledout 13  00:00:16:118 actualspeed 17  00:00:16:246 distancetraveledout 14  00:00:16:372 motorspeedout 18  00:00:16:457 distancetraveledout 15  00:00:16:472 actualspeed 18  00:00:16:657 distancetraveledout 16  00:00:16:726 motorspeedout 19  00:00:16:826 actualspeed 19  00:00:16:855 distancetraveledout 17  00:00:17:044 distancetraveledout 18  00:00:17:080 motorspeedout 20  00:00:17:180 actualspeed 20  00:00:17:230 distancetraveledout 19  00:00:17:410 distancetraveledout 20  00:00:17:590 distancetraveledout 21  00:00:17:770 distancetraveledout 22  00:00:17:950 distancetraveledout 23  … … …  00:04:59:650 distancetraveledout 1588  00:04:59:830 distancetraveledout 1589  //Apply very little brakes..  00:05:00:000 brakeintensityout 0.004  00:05:00:010 distancetraveledout 1590  … … …  00:05:08:830 distancetraveledout 1639  00:05:09:000 actualspeed 19  00:05:09:010 distancetraveledout 1640  … … …  00:07:24:000 actualspeed 4  00:07:24:350 distancetraveledout 2091  … … …  00:07:32:441 distancetraveledout 2100  00:07:33:000 actualspeed 3  00:07:33:454 distancetraveledout 2101  … … …  00:07:41:854 distancetraveledout 2108  00:07:42:000 actualspeed 2  00:07:43:580 distancetraveledout 2109  00:07:45:379 distancetraveledout 2110  00:07:47:178 distancetraveledout 2111  00:07:48:977 distancetraveledout 2112  00:07:50:776 distancetraveledout 2113  00:07:51:000 actualspeed 1  00:07:54:151 distancetraveledout 2114  00:07:57:750 distancetraveledout 2115  00:08:00:000 actualspeed 0  00:08:00:000 motorspeedout 0 |

## GPS Queue Atomic Model

|  |  |
| --- | --- |
| *Model Description* | This model stores “GPS” instructions and outputs them when the rest of the system needs to start acting on them. |
| *Test Case* | GPS instructions MSB = turn direction (0=LEFT, 1=RIGHT)  15 LSBs = distance   1. Set speed to 60 km/h 2. New instruction, turn right in 500 meters 3. New instruction, turn left in 200 meters 4. New speed = 100 km/h 5. New instruction, turn right in 200 meters 6. New instruction, turn left in 50 meters 7. Set speed to 0 8. Queue up an instruction but can’t act on it as speed is 0 9. Set speed to 50.. then input an additional turn.. |
| *Input Events* | 00:00:05:00 speedIn 60  00:00:07:00 GpsInstructionIn 33268  00:00:30:00 GpsInstructionIn 200  00:01:00:00 speedIn 100  00:05:00:00 GpsInstructionIn 32969  00:06:00:00 GpsInstructionIn 50  00:11:00:00 speedIn 0  00:12:00:00 GpsInstructionIn 32969  00:13:00:00 speedIn 50  00:15:00:00 GpsInstructionIn 500 |
| *Coupled Model Layout* | [top]  components : gpsQueue@GpsQueue  out : turnRequest  in : gpsInstructionIn speedIn  Link : gpsInstructionIn gpsInstructionIn@gpsQueue  Link : speedIn speedIn@gpsQueue  Link : turnRequest@gpsQueue turnRequest |
| *Simulation Time* | 20 minutes |
| *Output* | 00:00:38:000 turnrequest 1  00:00:50:000 turnrequest 0  00:05:07:000 turnrequest 1  00:06:01:000 turnrequest 0  //Wait until speed is back up then turn  00:13:14:471 turnrequest 1  00:15:38:000 turnrequest 0 |

## Direction Controller Coupled Model

|  |  |
| --- | --- |
| *Model Description* | This coupled model contains the GPS queue and steer driver |
| *Test Case* | Receive combined input events for GPS and steer driver and handle them in coupled together (same as individually) |
| *Input Events* | 00:00:05:00 speedIn 60  00:00:07:00 gps 33268  00:00:10:00 laneDetection 10  00:00:25:00 speedIn 10  00:00:30:00 gps 200  00:00:30:00 laneDetection 0  00:01:00:00 speedIn 100  00:05:00:00 gps 32969  00:06:00:00 gps 50  00:11:00:00 speedIn 0  00:12:00:00 gps 32969  00:13:00:00 speedIn 50  00:15:00:00 gps 500  00:01:10:00 laneDetection 10  00:01:50:00 laneDetection -10 |
| *Coupled Model Layout* | [top]  components : DirectionController  out : turnRequest servoDutyCycle  in : laneDetection gps speedIn  Link : gps gps@DirectionController  Link : speedIn speedIn@DirectionController  Link : turnRequest@DirectionController turnRequest  Link : laneDetection laneDetection@DirectionController  Link : servoDutyCycle@DirectionController servoDutyCycle  [DirectionController]  components : gpsQueue@GpsQueue  components : steerDriver@SteerDriver  out : turnRequest servoDutyCycle  in : laneDetection gps speedIn  Link : gps@DirectionController gpsInstructionIn@gpsQueue  Link : speedIn@DirectionController speedIn@gpsQueue  Link : turnRequest@gpsQueue turnRequest@DirectionController  Link : laneDetection@DirectionController wheelDirectionIn@steerDriver  Link : speedIn@DirectionController speedIn@steerDriver  Link : servoDutyCycle@SteerDriver servoDutyCycle@DirectionController |
| *Simulation Time* | 20 minutes |
| *Output* | //Deal with input, adjust steering  00:00:10:044 servodutycycle 7.5  00:00:10:088 servodutycycle 7.6  00:00:10:132 servodutycycle 7.7  00:00:10:176 servodutycycle 7.8  00:00:10:220 servodutycycle 7.9  00:00:10:264 servodutycycle 8  00:00:10:308 servodutycycle 8.1  00:00:10:352 servodutycycle 8.2  00:00:10:396 servodutycycle 8.3  00:00:10:440 servodutycycle 8.4  00:00:10:484 servodutycycle 8.5  00:00:30:007 servodutycycle 8.5  00:00:30:014 servodutycycle 8.4  00:00:30:021 servodutycycle 8.3  00:00:30:028 servodutycycle 8.2  00:00:30:035 servodutycycle 8.1  00:00:30:042 servodutycycle 8  00:00:30:049 servodutycycle 7.9  00:00:30:056 servodutycycle 7.8  00:00:30:063 servodutycycle 7.7  00:00:30:070 servodutycycle 7.6  00:00:30:077 servodutycycle 7.5  00:01:10:074 servodutycycle 7.5  00:01:10:148 servodutycycle 7.6  00:01:10:222 servodutycycle 7.7  00:01:10:296 servodutycycle 7.8  00:01:10:370 servodutycycle 7.9  00:01:10:444 servodutycycle 8  00:01:10:518 servodutycycle 8.1  00:01:10:592 servodutycycle 8.2  00:01:10:666 servodutycycle 8.3  00:01:10:740 servodutycycle 8.4  00:01:10:814 servodutycycle 8.5  //output turn request from GPS instruction  00:01:18:000 turnrequest 1  00:01:25:000 turnrequest 0  00:01:50:074 servodutycycle 8.5  00:01:50:148 servodutycycle 8.4  00:01:50:222 servodutycycle 8.3  00:01:50:296 servodutycycle 8.2  00:01:50:370 servodutycycle 8.1  00:01:50:444 servodutycycle 8  00:01:50:518 servodutycycle 7.9  00:01:50:592 servodutycycle 7.8  00:01:50:666 servodutycycle 7.7  00:01:50:740 servodutycycle 7.6  00:01:50:814 servodutycycle 7.5  00:01:50:888 servodutycycle 7.4  00:01:50:962 servodutycycle 7.3  00:01:51:036 servodutycycle 7.2  00:01:51:110 servodutycycle 7.1  00:01:51:184 servodutycycle 7  00:01:51:258 servodutycycle 6.9  00:01:51:332 servodutycycle 6.8  00:01:51:406 servodutycycle 6.7  00:01:51:480 servodutycycle 6.6  00:01:51:554 servodutycycle 6.5 |

## Speed Controller Coupled Model

|  |  |
| --- | --- |
| *Model Description* | This coupled model contains the Desired speed calculator and the speed driver. It uses external models actual speed calculator and odometer to operate. |
| *Test Case* | Receive input events for desired speed calculator and test response in terms of the Speed Driver/Actual Speed Calculator/Odometer coupled test model. |
| *Input Events* | //Output converges to correct speed given different obstacle distances, and speed limit.  00:00:05:00 centerRange 60  00:00:05:01 leftRange 50  00:00:05:02 rightRange 50  00:01:00:00 infrastructure 50226 //Speed limit Change  00:05:00:00 infrastructure 59692//Speed limit change  00:07:00:00 centerRange 40  00:09:00:00 centerRange 45  00:12:00:00 centerRange 60  00:15:00:00 centerRange 10 |
| *Coupled Model Layout* | [top]  components : SpeedController  components : actualSpeedCalc@ActualSpeedCalc  components : odometer@Odometer  out : motorDutyCycleOut brakeIntensity actualSpeed  in : centerRange leftRange rightRange infrastructure  Link : actualSpeed@ActualSpeedCalc speedIn@SpeedController  Link : actualSpeed@ActualSpeedCalc actualSpeed  Link : brakeIntensity@SpeedController brakeIntensityIn@ActualSpeedCalc  Link : motorDutyCycleOut@SpeedController motorDutyCycleIn@ActualSpeedCalc  Link : actualSpeed@ActualSpeedCalc speedIn@Odometer  Link : distanceTraveled@Odometer distanceTraveledOut@SpeedController  Link : motorDutyCycleOut@SpeedController motorDutyCycleOut  Link : brakeIntensity@SpeedController brakeIntensity  Link : centerRange centerRange@SpeedController  Link : leftRange leftRange@SpeedController  Link : rightRange rightRange@SpeedController  Link : infrastructure infrastructure@SpeedController  [SpeedController]  components : desiredSpeedCalculator@DesiredSpeedCalculator  components : speedDriver@SpeedDriver  out : motorDutyCycleOut brakeIntensity  in : speedIn centerRange leftRange rightRange infrastructure distanceTraveledOut  Link : leftRange@SpeedController leftRangeIn@desiredSpeedCalculator  Link : centerRange@SpeedController centerRangeIn@desiredSpeedCalculator  Link : rightRange@SpeedController rightRangeIn@desiredSpeedCalculator  Link : speedIn@SpeedController speedIn@desiredSpeedCalculator  Link : speedIn@SpeedController currentSpeedIn@speedDriver  Link : distanceTraveledOut@SpeedController odometerIn@desiredSpeedCalculator  Link : infrastructure@SpeedController infrastructureIn@desiredSpeedCalculator  Link : desiredSpeedOut@desiredSpeedCalculator desiredSpeedIn@speedDriver  Link : desiredSpeedReached@speedDriver desiredSpeedReachedIn@desiredSpeedCalculator  Link : motorSpeedOut@speedDriver motorDutyCycleOut@SpeedController  Link : brakeIntensityOut@speedDriver brakeIntensity@SpeedController |
| *Simulation Time* | 30 minutes |
| *Output* | //Accelerate  00:01:00:138 motordutycycleout 1  00:01:00:138 actualspeed 0  00:01:00:238 actualspeed 1  00:01:00:276 motordutycycleout 2  00:01:00:376 actualspeed 2  … … …  00:01:01:342 actualspeed 9  00:01:01:380 motordutycycleout 10  00:01:01:480 actualspeed 10  //Accelerate more  00:05:00:100 motordutycycleout 11  00:05:00:200 motordutycycleout 12  00:05:00:200 actualspeed 11  00:05:00:300 motordutycycleout 13  00:05:00:300 actualspeed 12  00:05:00:400 motordutycycleout 14  00:05:00:400 actualspeed 13  00:05:00:500 motordutycycleout 15  00:05:00:500 actualspeed 14  … … …  00:05:08:800 motordutycycleout 98  00:05:08:800 actualspeed 97  00:05:08:900 motordutycycleout 99  00:05:08:900 actualspeed 98  00:05:09:000 motordutycycleout 100  00:05:09:000 actualspeed 99  00:05:09:100 actualspeed 100  //something in center range, slow down  00:07:00:000 brakeintensity 1  00:07:00:035 actualspeed 99  00:07:00:035 motordutycycleout 0  00:07:00:070 actualspeed 98  00:07:00:105 actualspeed 97  00:07:00:140 actualspeed 96  00:07:00:175 actualspeed 95  00:07:00:210 actualspeed 94  00:07:00:245 actualspeed 93  00:07:00:280 actualspeed 92  00:07:00:315 actualspeed 91  00:07:00:350 actualspeed 90  00:07:00:385 actualspeed 89  00:07:00:385 brakeintensity 0.990125  00:07:00:385 actualspeed 88  00:07:00:385 brakeintensity 0.0005  00:07:00:385 actualspeed 87  00:07:00:385 motordutycycleout 1  00:07:00:385 actualspeed 86  00:07:00:385 brakeintensity 0  00:07:00:485 motordutycycleout 87  00:07:00:485 actualspeed 85  00:07:00:485 motordutycycleout 85  00:07:00:485 actualspeed 86  00:07:00:485 motordutycycleout 86  00:07:00:485 actualspeed 85  00:07:00:485 motordutycycleout 85  00:07:00:585 motordutycycleout 86  00:07:00:685 motordutycycleout 87  00:07:00:685 actualspeed 86  00:07:00:685 motordutycycleout 86  00:07:00:785 motordutycycleout 87  //Speed back up, range changed a bit  00:09:00:100 motordutycycleout 88  00:09:00:100 actualspeed 87  00:09:00:100 motordutycycleout 87  00:09:00:200 motordutycycleout 88  00:09:00:300 motordutycycleout 89  00:09:00:300 actualspeed 88  00:09:00:300 motordutycycleout 88  00:09:00:400 motordutycycleout 89  00:09:00:500 motordutycycleout 90  00:09:00:500 actualspeed 89  00:09:00:500 motordutycycleout 89  00:09:00:600 motordutycycleout 90  00:09:00:700 motordutycycleout 91  00:09:00:700 actualspeed 90  00:09:00:700 motordutycycleout 90  00:09:00:800 motordutycycleout 91  00:09:00:900 motordutycycleout 92  00:09:00:900 actualspeed 91  00:09:00:900 motordutycycleout 91  00:09:01:000 motordutycycleout 92  //Front is clear, speed up to speed limit  00:12:00:100 motordutycycleout 93  00:12:00:100 actualspeed 92  00:12:00:200 motordutycycleout 94  00:12:00:200 actualspeed 93  00:12:00:300 motordutycycleout 95  00:12:00:300 actualspeed 94  00:12:00:400 motordutycycleout 96  00:12:00:400 actualspeed 95  00:12:00:500 motordutycycleout 97  00:12:00:500 actualspeed 96  00:12:00:600 motordutycycleout 98  00:12:00:600 actualspeed 97  00:12:00:700 motordutycycleout 99  00:12:00:700 actualspeed 98  00:12:00:800 motordutycycleout 100  00:12:00:800 actualspeed 99  00:12:00:900 actualspeed 100  //Something directly in the path, slow down a lot!  00:15:00:000 brakeintensity 1  00:15:00:035 actualspeed 99  00:15:00:035 motordutycycleout 0  00:15:00:070 actualspeed 98  00:15:00:105 actualspeed 97  00:15:00:140 actualspeed 96  … … …  00:15:01:645 actualspeed 53  00:15:01:680 actualspeed 52  00:15:01:715 actualspeed 51  00:15:01:750 actualspeed 50  00:15:01:785 actualspeed 49  00:15:01:820 actualspeed 48  00:15:01:855 actualspeed 47  00:15:01:890 actualspeed 46  00:15:01:925 actualspeed 45  00:15:01:960 actualspeed 44  00:15:01:960 brakeintensity 0.968  00:15:01:960 actualspeed 43  00:15:01:960 brakeintensity 0.9245  00:15:01:960 actualspeed 42  00:15:01:960 brakeintensity 0.002  00:15:01:960 actualspeed 41  00:15:01:960 brakeintensity 0.0005  00:15:01:960 actualspeed 40  00:15:01:960 motordutycycleout 1  00:15:01:960 actualspeed 39  00:15:01:960 brakeintensity 0  00:15:02:060 motordutycycleout 40  00:15:02:060 actualspeed 38  00:15:02:060 motordutycycleout 38  00:15:02:060 actualspeed 39  00:15:02:060 motordutycycleout 39  00:15:02:060 actualspeed 38  00:15:02:060 motordutycycleout 38  00:15:02:160 motordutycycleout 39  00:15:02:260 motordutycycleout 40  00:15:02:260 actualspeed 39  00:15:02:260 motordutycycleout 39  00:15:02:360 motordutycycleout 40 //Converge ~40 km/h for safe following distance |

## Top Test Coupled Model

|  |  |
| --- | --- |
| *Model Description* | Top model with meaningful outputs for validation. |
| *Test Case* | React to a variety of inputs, based on previous tests. |
| *Input Events* | 00:00:05:00 centerRange 60  00:00:05:01 leftRange 50  00:00:05:02 rightRange 50  00:00:05:03 infrastructure 50226  00:01:00:00 gps 33268  00:01:30:00 laneDetection 10  00:02:00:00 laneDetection 0  00:03:00:00 infrastructure 59692  00:06:00:00 gps 50 |
| *Coupled Model Layout* | [top]  components : CarController  components : actualSpeedCalc@ActualSpeedCalc  components : motorPwm@Pwm  components : servoPwm@Pwm  components : odometer@Odometer  out : motorDutyCycle turnRequest brakeIntensity servoDutyCycle  in : laneDetection centerRange leftRange rightRange infrastructure gps  Link : actualSpeed@ActualSpeedCalc speedIn@CarController  Link : brakeIntensity@CarController brakeIntensityIn@ActualSpeedCalc  Link : motorDutyCycleOut@CarController motorDutyCycleIn@ActualSpeedCalc  Link : actualSpeed@ActualSpeedCalc speedIn@odometer  Link : distanceTraveled@odometer distanceTraveledOut@CarController  Link : motorDutyCycleOut@CarController motorDutyCycle  Link : motorDutyCycleOut@CarController dutyCycleIn@motorPwm  Link : servoDutyCycle@CarController dutyCycleIn@servoPwm  Link : servoDutyCycle@CarController servoDutyCycle  Link : turnRequest@CarController turnRequest  Link : brakeIntensity@CarController brakeIntensity  Link : laneDetection laneDetection@CarController  Link : centerRange centerRange@CarController  Link : leftRange leftRange@CarController  Link : rightRange rightRange@CarController  Link : infrastructure infrastructure@CarController  Link : gps gps@CarController  [motorPwm]  period : 100  [servoPwm]  period : 10  [CarController]  components : SpeedController  components : DirectionController  out : motorDutyCycleOut turnRequest brakeIntensity servoDutyCycle  in : speedIn laneDetection centerRange leftRange rightRange infrastructure gps distanceTraveledOut  Link : leftRange@CarController leftRange@SpeedController  Link : centerRange@CarController centerRange@SpeedController  Link : rightRange@CarController rightRange@SpeedController  Link : speedIn@CarController speedIn@SpeedController  Link : distanceTraveledOut@CarController distanceTraveledOut@SpeedController  Link : infrastructure@CarController infrastructure@SpeedController  Link : brakeIntensity@SpeedController brakeIntensity@CarController  Link : motorDutyCycleOut@SpeedController motorDutyCycleOut@CarController  Link : laneDetection@CarController laneDetection@DirectionController  Link : gps@CarController gps@DirectionController  Link : speedIn@CarController speedIn@DirectionController  Link : turnRequest@DirectionController turnRequest@CarController  Link : servoDutyCycle@DirectionController servoDutyCycle@CarController  [SpeedController]  components : desiredSpeedCalculator@DesiredSpeedCalculator  components : speedDriver@SpeedDriver  out : motorDutyCycleOut brakeIntensity  in : speedIn centerRange leftRange rightRange infrastructure distanceTraveledOut  Link : leftRange@SpeedController leftRangeIn@desiredSpeedCalculator  Link : centerRange@SpeedController centerRangeIn@desiredSpeedCalculator  Link : rightRange@SpeedController rightRangeIn@desiredSpeedCalculator  Link : speedIn@SpeedController speedIn@desiredSpeedCalculator  Link : speedIn@SpeedController currentSpeedIn@speedDriver  Link : distanceTraveledOut@SpeedController odometerIn@desiredSpeedCalculator  Link : infrastructure@SpeedController infrastructureIn@desiredSpeedCalculator  Link : desiredSpeedOut@desiredSpeedCalculator desiredSpeedIn@speedDriver  Link : desiredSpeedReached@speedDriver desiredSpeedReachedIn@desiredSpeedCalculator  Link : motorSpeedOut@speedDriver motorDutyCycleOut@SpeedController  Link : brakeIntensityOut@speedDriver brakeIntensity@SpeedController  [DirectionController]  components : gpsQueue@GpsQueue  components : steerDriver@SteerDriver  out : turnRequest servoDutyCycle  in : laneDetection gps speedIn  Link : gps@DirectionController gpsInstructionIn@gpsQueue  Link : speedIn@DirectionController speedIn@gpsQueue  Link : turnRequest@gpsQueue turnRequest@DirectionController  Link : laneDetection@DirectionController wheelDirectionIn@steerDriver  Link : speedIn@DirectionController speedIn@steerDriver  Link : servoDutyCycle@SteerDriver servoDutyCycle@DirectionController |
| *Simulation Time* | 10 minutes |
| *Output* | 00:00:05:141 motordutycycle 1  00:00:05:279 motordutycycle 2  00:00:05:417 motordutycycle 3  00:00:05:555 motordutycycle 4  00:00:05:693 motordutycycle 5  00:00:05:831 motordutycycle 6  00:00:05:969 motordutycycle 7  00:00:06:107 motordutycycle 8  00:00:06:245 motordutycycle 9  00:00:06:383 motordutycycle 10  00:01:30:007 servodutycycle 7.5  00:01:30:014 servodutycycle 7.6  00:01:30:021 servodutycycle 7.7  00:01:30:028 servodutycycle 7.8  00:01:30:035 servodutycycle 7.9  00:01:30:042 servodutycycle 8  00:01:30:049 servodutycycle 8.1  00:01:30:056 servodutycycle 8.2  00:01:30:063 servodutycycle 8.3  00:01:30:070 servodutycycle 8.4  00:01:30:077 servodutycycle 8.5  00:02:00:007 servodutycycle 8.5  00:02:00:014 servodutycycle 8.4  00:02:00:021 servodutycycle 8.3  00:02:00:028 servodutycycle 8.2  00:02:00:035 servodutycycle 8.1  00:02:00:042 servodutycycle 8  00:02:00:049 servodutycycle 7.9  00:02:00:056 servodutycycle 7.8  00:02:00:063 servodutycycle 7.7  00:02:00:070 servodutycycle 7.6  00:02:00:077 servodutycycle 7.5  00:03:00:100 motordutycycle 11  00:03:00:200 motordutycycle 12  00:03:00:300 motordutycycle 13  00:03:00:400 motordutycycle 14  00:03:00:500 motordutycycle 15  00:03:00:600 motordutycycle 16  00:03:00:700 motordutycycle 17  00:03:00:800 motordutycycle 18  00:03:00:900 motordutycycle 19  00:03:01:000 motordutycycle 20  00:03:01:100 motordutycycle 21  00:03:01:200 motordutycycle 22  00:03:01:300 motordutycycle 23  00:03:01:400 motordutycycle 24  00:03:01:500 motordutycycle 25  00:03:01:600 motordutycycle 26  00:03:01:700 motordutycycle 27  00:03:01:800 motordutycycle 28  00:03:01:900 motordutycycle 29  00:03:02:000 motordutycycle 30  00:03:02:100 motordutycycle 31  00:03:02:200 motordutycycle 32  00:03:02:300 motordutycycle 33  00:03:02:400 motordutycycle 34  00:03:02:500 motordutycycle 35  00:03:02:600 motordutycycle 36  00:03:02:700 motordutycycle 37  00:03:02:800 motordutycycle 38  00:03:02:900 motordutycycle 39  00:03:03:000 motordutycycle 40  00:03:03:100 motordutycycle 41  00:03:03:200 motordutycycle 42  00:03:03:300 motordutycycle 43  00:03:03:400 motordutycycle 44  00:03:03:500 motordutycycle 45  00:03:03:600 motordutycycle 46  00:03:03:700 motordutycycle 47  00:03:03:800 motordutycycle 48  00:03:03:900 motordutycycle 49  00:03:04:000 motordutycycle 50  00:03:04:100 motordutycycle 51  00:03:04:200 motordutycycle 52  00:03:04:300 motordutycycle 53  00:03:04:400 motordutycycle 54  00:03:04:500 motordutycycle 55  00:03:04:600 motordutycycle 56  00:03:04:700 motordutycycle 57  00:03:04:800 motordutycycle 58  00:03:04:900 motordutycycle 59  00:03:05:000 motordutycycle 60  00:03:05:100 motordutycycle 61  00:03:05:200 motordutycycle 62  00:03:05:300 motordutycycle 63  00:03:05:400 motordutycycle 64  00:03:05:500 motordutycycle 65  00:03:05:600 motordutycycle 66  00:03:05:700 motordutycycle 67  00:03:05:800 motordutycycle 68  00:03:05:900 motordutycycle 69  00:03:06:000 motordutycycle 70  00:03:06:100 motordutycycle 71  00:03:06:200 motordutycycle 72  00:03:06:300 motordutycycle 73  00:03:06:400 motordutycycle 74  00:03:06:500 motordutycycle 75  00:03:06:600 motordutycycle 76  00:03:06:700 motordutycycle 77  00:03:06:800 motordutycycle 78  00:03:06:900 motordutycycle 79  00:03:07:000 motordutycycle 80  00:03:07:100 motordutycycle 81  00:03:07:200 motordutycycle 82  00:03:07:300 motordutycycle 83  00:03:07:400 motordutycycle 84  00:03:07:500 motordutycycle 85  00:03:07:600 motordutycycle 86  00:03:07:700 motordutycycle 87  00:03:07:800 motordutycycle 88  00:03:07:900 motordutycycle 89  00:03:08:000 motordutycycle 90  00:03:08:100 motordutycycle 91  00:03:08:200 motordutycycle 92  00:03:08:300 motordutycycle 93  00:03:08:400 motordutycycle 94  00:03:08:500 motordutycycle 95  00:03:08:600 motordutycycle 96  00:03:08:700 motordutycycle 97  00:03:08:800 motordutycycle 98  00:03:08:900 motordutycycle 99  00:03:09:000 motordutycycle 100  00:03:27:100 turnrequest 1  00:06:01:000 turnrequest 0 |

## Top Test Coupled Model Extended testing – Stopping at a stop sign

|  |  |
| --- | --- |
| *Model Description* | Top model with meaningful outputs for validation. |
| *Test Case* | Simulate stopping and starting again. |
| *Input Events* | 00:00:05:00 centerRange 60  00:00:05:01 leftRange 50  00:00:05:02 rightRange 50  //Setup speed for 10 km/h  00:00:5:03 infrastructure 50226  //Stop in 500 meters  00:05:00:00 infrastructure 33268 |
| *Coupled Model Layout* | <Same as above> |
| *Simulation Time* | 20 minutes |
| *Output* | 00:00:05:141 motordutycycle 1  00:00:05:279 motordutycycle 2  00:00:05:417 motordutycycle 3  00:00:05:555 motordutycycle 4  00:00:05:693 motordutycycle 5  00:00:05:831 motordutycycle 6  00:00:05:969 motordutycycle 7  00:00:06:107 motordutycycle 8  00:00:06:245 motordutycycle 9  00:00:06:383 motordutycycle 10  00:05:00:000 brakeintensity 0.001  00:11:00:000 motordutycycle 0  00:11:00:000 brakeintensity 0  00:11:00:100 motordutycycle 1  00:11:00:200 motordutycycle 2  00:11:00:300 motordutycycle 3  00:11:00:400 motordutycycle 4  00:11:00:500 motordutycycle 5  00:11:00:600 motordutycycle 6  00:11:00:700 motordutycycle 7  00:11:00:800 motordutycycle 8  00:11:00:900 motordutycycle 9  00:11:01:000 motordutycycle 10 |

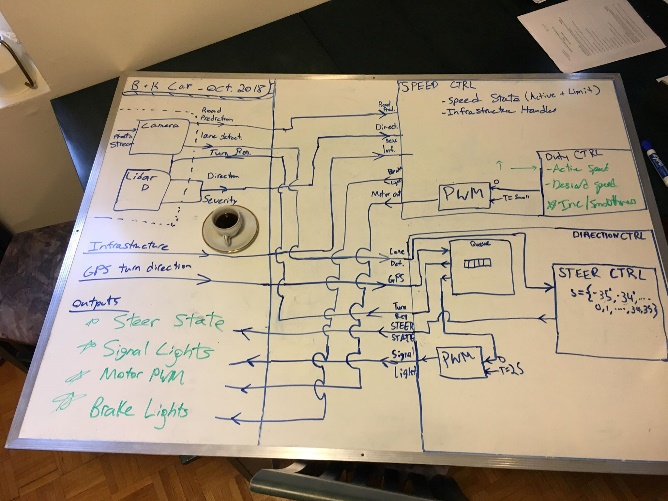
## Top FULL Coupled Model

|  |  |
| --- | --- |
| *Model Description* | Top model with real outputs! |
| *Test Case* | React to same inputs above, real motor and servo outputs! |
| *Input Events* | 00:00:05:00 centerRange 60  00:00:05:01 leftRange 50  00:00:05:02 rightRange 50  00:00:05:03 infrastructure 50226  00:01:00:00 gps 33268  00:01:30:00 laneDetection 10  00:02:00:00 laneDetection 0  00:03:00:00 infrastructure 59692  00:06:00:00 gps 50 |
| *Coupled Model Layout* | [top]  components : CarController  components : actualSpeedCalc@ActualSpeedCalc  components : motorPwm@Pwm  components : servoPwm@Pwm  components : odometer@Odometer  out : motorVoltage turnRequest brakeIntensity servoVoltage  in : laneDetection centerRange leftRange rightRange infrastructure gps  Link : actualSpeed@ActualSpeedCalc speedIn@CarController  Link : brakeIntensity@CarController brakeIntensityIn@ActualSpeedCalc  Link : motorDutyCycleOut@CarController motorDutyCycleIn@ActualSpeedCalc  Link : actualSpeed@ActualSpeedCalc speedIn@odometer  Link : distanceTraveled@odometer distanceTraveledOut@CarController  Link : motorDutyCycleOut@CarController dutyCycleIn@motorPwm  Link : pwmStateOut@motorPwm motorVoltage  Link : servoDutyCycle@CarController dutyCycleIn@servoPwm  Link : pwmStateOut@servoPwm servoVoltage  Link : turnRequest@CarController turnRequest  Link : brakeIntensity@CarController brakeIntensity  Link : laneDetection laneDetection@CarController  Link : centerRange centerRange@CarController  Link : leftRange leftRange@CarController  Link : rightRange rightRange@CarController  Link : infrastructure infrastructure@CarController  Link : gps gps@CarController  [motorPwm]  period : 1  [servoPwm]  period : 10  [CarController]  components : SpeedController  components : DirectionController  out : motorDutyCycleOut turnRequest brakeIntensity servoDutyCycle  in : speedIn laneDetection centerRange leftRange rightRange infrastructure gps distanceTraveledOut  Link : leftRange@CarController leftRange@SpeedController  Link : centerRange@CarController centerRange@SpeedController  Link : rightRange@CarController rightRange@SpeedController  Link : speedIn@CarController speedIn@SpeedController  Link : distanceTraveledOut@CarController distanceTraveledOut@SpeedController  Link : infrastructure@CarController infrastructure@SpeedController  Link : brakeIntensity@SpeedController brakeIntensity@CarController  Link : motorDutyCycleOut@SpeedController motorDutyCycleOut@CarController  Link : laneDetection@CarController laneDetection@DirectionController  Link : gps@CarController gps@DirectionController  Link : speedIn@CarController speedIn@DirectionController  Link : turnRequest@DirectionController turnRequest@CarController  Link : servoDutyCycle@DirectionController servoDutyCycle@CarController  [SpeedController]  components : desiredSpeedCalculator@DesiredSpeedCalculator  components : speedDriver@SpeedDriver  out : motorDutyCycleOut brakeIntensity  in : speedIn centerRange leftRange rightRange infrastructure distanceTraveledOut  Link : leftRange@SpeedController leftRangeIn@desiredSpeedCalculator  Link : centerRange@SpeedController centerRangeIn@desiredSpeedCalculator  Link : rightRange@SpeedController rightRangeIn@desiredSpeedCalculator  Link : speedIn@SpeedController speedIn@desiredSpeedCalculator  Link : speedIn@SpeedController currentSpeedIn@speedDriver  Link : distanceTraveledOut@SpeedController odometerIn@desiredSpeedCalculator  Link : infrastructure@SpeedController infrastructureIn@desiredSpeedCalculator  Link : desiredSpeedOut@desiredSpeedCalculator desiredSpeedIn@speedDriver  Link : desiredSpeedReached@speedDriver desiredSpeedReachedIn@desiredSpeedCalculator  Link : motorSpeedOut@speedDriver motorDutyCycleOut@SpeedController  Link : brakeIntensityOut@speedDriver brakeIntensity@SpeedController  [DirectionController]  components : gpsQueue@GpsQueue  components : steerDriver@SteerDriver  out : turnRequest servoDutyCycle  in : laneDetection gps speedIn  Link : gps@DirectionController gpsInstructionIn@gpsQueue  Link : speedIn@DirectionController speedIn@gpsQueue  Link : turnRequest@gpsQueue turnRequest@DirectionController  Link : laneDetection@DirectionController wheelDirectionIn@steerDriver  Link : speedIn@DirectionController speedIn@steerDriver  Link : servoDutyCycle@SteerDriver servoDutyCycle@DirectionController |
| *Simulation Time* | 10 minutes |
| *Output* | 00:00:05:141 motorvoltage 1  00:00:05:150 motorvoltage 0  00:00:05:408 motorvoltage 1  00:00:05:426 motorvoltage 0  00:00:05:684 motorvoltage 1  00:00:05:702 motorvoltage 0  00:00:05:960 motorvoltage 1  00:00:05:978 motorvoltage 0  00:00:06:236 motorvoltage 1  … … …  00:03:27:100 turnrequest 1  … … …  00:03:39:334 servovoltage 1  00:03:40:084 servovoltage 0  … … …  00:06:01:000 turnrequest 0  … … …  00:09:55:100 motorvoltage 1  … … …  00:09:59:334 servovoltage 1  //Real outputs not very meaningful for validation, but just for demonstrating that outputs are working fine. |

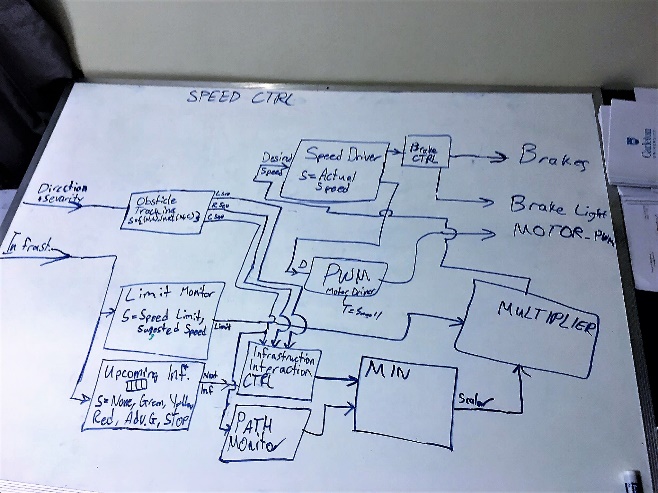
# Appendix – Sketches and Preliminary Model Structure



**Figure 3: Rough Sketch of Vehicle to be Modelled**

****

**Figure 4: Rough Sketch of Top Level Model**

****

**Figure 5: Rough Sketch of Speed Controller Submodel**