**Assignment 1 – Conceptual Model**

**Discrete Event Driven Vehicle**

Ali Salaheddin – 100704976

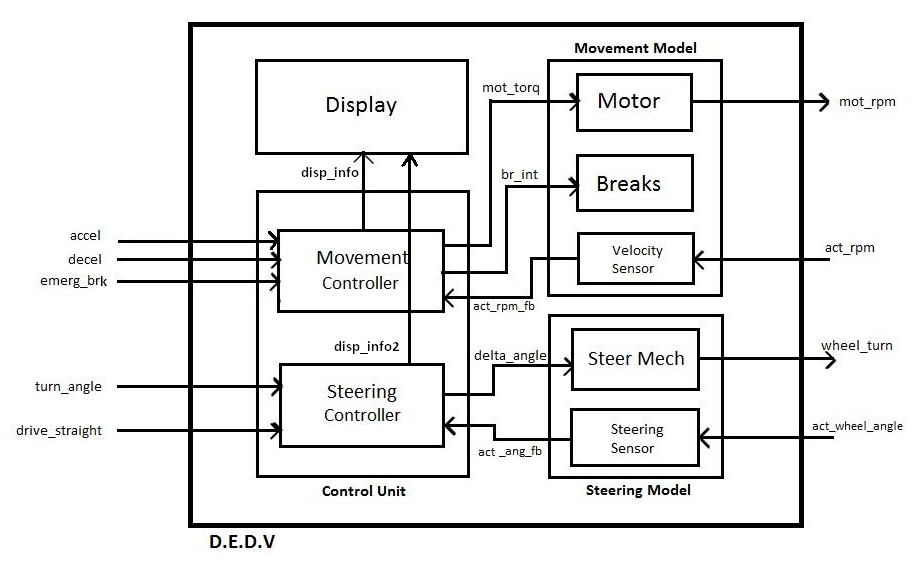
Mohamed Ebada – 100286033

**Part 1:**

The following model concept is inspired by the Google driverless car.

We intend to create a model of a self-driven automobile. This concept entails that the vehicle would have enough artificial intelligence to drive and break on its own while using periodic feedback from sensors mounted on-board to let it know of its surroundings.

To simplify the model, we consider a human to push buttons that would send events to the model to tell the automobile to accelerate/decelerate, to break or to turn.



**Components:**

**Control Unit** is made of 2 atomic models:

**Movement Controller:** receives input events to accelerate, decelerate and do an emergency stop. These inputs are processed and the controller sends corresponding signals to the Movement Model’s components to adequately speed up the motor or break and slow the vehicle. This controller also gets feedback on the actual speed of the vehicle and keeps track of that velocity. This velocity information is sent to a Display which displays it in a visual form to the human user.

**Steering Controller:** receives input events turn the vehicle with a certain angle or keep straight. These inputs are processed and the controller sends corresponding signals to the Movement Model’s components to adequately steer the vehicle with a desired angle. This controller also gets feedback on the actual turning angle of the vehicle and keeps track of that for later calculation of narrower turns. This turning information is sent to a Display which displays it in a visual form to the human user.

**Display:** receives information from the Movement Controller regarding the vehicle’s speed and turning angle and visually displays it as an output to the human user.

**Movement Model** is made of 3 atomic models:

**Motor:** receives input from the Control Unit telling it to accelerate, decelerate or stop due to an emergency. The output of the Motor is a RPM rate that causes the vehicle to move at that speed.

**Breaks:** receives input from the Control unit telling them to clamp or keep loose. They affect the vehicle’s wheels and have no particular output.

**Velocity Sensor:** as the vehicle moves this sensor calculates the speed of the rotating wheels. It sends the results to the Control Unit for feedback purpose.

**Steering Model** is made of 2 atomic models:

**Steering mechanism:** receives input from the Control Unit telling it to steer the wheels at a given angle from the axle of the vehicle (angle 0). The output is turned wheels.

**Steering Sensor:** as the wheels turn this sensor reads their angle with regards to the vehicle's axle (angle 0). It sends the results to the Control Unit for feedback purpose.

**Part 2:**

**Atomic\_Mod = <X, Y, S, d.int, d.ext, lambda, ta>**

**Note:** in the following pseudo-code describing the transition and output functions of the models, we’ve defined a custom function void output(int val, out port), that sends a value “val” on an output port “port”.

**Steering Controller:**

X = { (turn\_angle, N) , (drive\_straight, N) , (act\_ang\_fb, N) }

Y = { (delta\_angle, N) , (disp\_info2, N)}

S = { state={active, passive}, flag={proc\_turn, proc\_straight, proc\_fb} , out\_wheel\_ang =Z, act\_angle =Z}

//if Steering Controller doesn't get any input after a while then it turns idle

d.int() {

passivate(); //go idle

}

//transition function of Steering Controller as it receives a signal to read data

d.ext() {

int in\_wheel\_ang;

if(x.port == turn\_angle) {

in\_wheel\_ang = read(turn\_angle);

out\_wheel\_ang = in\_wheel\_ang - act\_ang\_fb;

flag = proc\_turn;

}

if (x.port == drive\_straight) {

in\_wheel\_ang = read(drive\_straight);

out\_wheel\_ang = ZERO; //ZERO is an internal define equal to the value zero.

flag = proc\_straight;

}

if (x.port == act\_ang\_fb) {

act\_angle = read(act\_ang\_fb);

flag = proc\_fb;

}

State = active;

}

//output function: outputs signals to Steering Model to turn the wheels of the vehicle to make a turn.

lambda {

if(flag == proc\_turn)

output(out\_wheel\_ang, delta\_angle);

if(flag == proc\_straight)

output(out\_wheel\_ang, delta\_angle);

if(flag == proc\_fb)

output(act\_angle, disp\_info2);

}

**Movement Controller:**

X = { (accel, N) , (decel, N) , (emerg\_brk, N) , (act\_rpm\_fb, N) }

Y = { (mot\_torq, N) , (brk\_int, N) , (disp\_info, N) }

S = { state={passive, active}, flag={proc\_accel, proc\_decel, proc\_brk, proc\_fb }, act\_rpm =Z}

//if Movement Controller doesn't get any input after a while then it turns idle

d.int() {

passivate();

}

//transition function of Movement Controller as it receives a signal to read data

d.ext() {

if(x.port == accel && act\_rpm < MAX\_RPM) {

flag = proc\_accel;

} else if (x.port == decel && act\_rpm > ZERO) {

flag = proc\_decel;

} else if (x.port == emerg\_brk) {

flag = proc\_brk;

} else if (x.port == act\_rpm\_fb) {

act\_rpm = read(act\_rpm\_fb);

flag = proc\_fb;

}

State = active;

}

//output function: outputs signals to Movement Model to accelerate and decelerate the vehicle incrementally + emergency stop

lambda {

if(flag == proc\_accel)

output(MOT\_UP1, mot\_torq); //MOT\_UP1 is an internal defined incremental value that increments the speed of the motor.

output(NO\_BRK, brk\_int); //NO\_BRK is an internal defined step value that disables the breaks.

else if(flag == proc\_decel) {

output(MOT\_DWN1, mot\_torq); //MOT\_DWN1 is an internal defined incremental value that decrements the speed of the motor.

output(BRK\_UP1, brk\_int); //BRK\_UP1 is an internal defined step value that increments the intensity of the breaks.

} else if(flag == proc\_brk)

output(MAX\_BRK, brk\_int); //MAX\_BRK is an internal defined value to clamp breaks to maximum intensity.

output(ZERO, mot\_torq);

output(act\_rpm, disp\_info);

}

**Motor:**

X = { (mot\_torq, N) }

Y = { (mot\_rpm, N) }

S = { state={active, passive} , out\_rpm=N }

//if Motor doesn't get any input after a while then it stays the same (turning or idle)

d.int() {

passivate();

state = passive;

}

//transition function of Motor as it receives a signal to turn

d.ext() {

if(x.port == mot\_torq) {

out\_rpm += read(mot\_torq);

if(out\_rpm > 0)

state = active;

else

state = passive;

}

}

//output function: outputs the rpm of the Motor

lambda {

output(out\_rpm, mot\_rpm);

}

**Breaks:**

X = { (brk\_int, N) }

Y = { Ø }

S = { state ={active, passive} }

//if Breaks don't get any input after a while then they stop clamping

d.int() {

passivate();

}

//transition function of Breaks as they receive a signal to clamp

d.ext() {

int brk\_intensity;

if(x.port == brk\_int) {

brk\_intensity += read(brk\_int);

if(brk\_intensity > 0)

//clamp(brk\_intensity); //clamps on the wheels

state = active;

else

state = passive;

}

}

//output function: no output. The effect of the breaks is on the Wheels.

lambda { }

**Velocity Sensor:**

X = { (act\_rpm, N) }

Y = { (act\_rpm\_fb, N) }

S = { state={active, passive} , wheel\_rpm =N}

//if Velocity Sensor doesn't get any input after a while then it turns idle

d.int() {

passivate();

}

//transition function of Sensor as it receives a signal to read data

d.ext() {

if(x.port == act\_rpm) {

wheel\_rpm = read(act\_rpm);

state = active;

}

}

//output function: outputs the actual read rpm of the Wheels to the Control Unit.

lambda {

output(wheel\_rpm, act\_rpm\_fb);

}

**Steering Mech:**

X = { (delta\_angle, N) }

Y = { (wheel\_turn, N) }

S = { state={active, passive}, angle=Z }

//if Steer Mech doesn't get any input after a while then it turns straight again

d.int() {

passivate();

}

//transition function of Sensor as it receives a signal to read data

d.ext() {

if(x.port == delta\_angle) {

angle = read(delta\_angle);

state = active;

}

}

//output function: outputs the actual read rpm of the Wheels to the Control Unit.

lambda {

output(angle, wheel\_turn);

}

**Steering Sensor:**

X = { (act\_wheel\_ang, N) }

Y = { (act\_ang\_fb, N) }

S = { state={active, passive}, wheel\_angle=Z }

//if Steering Sensor doesn't get any input after a while then it turns idle

d.int() {

passivate();

}

//transition function of Sensor as it receives a signal to read data

d.ext() {

if(x.port == act\_wheel\_angle) {

wheel\_angle = read(act\_wheel\_angle);

state = active;

}

}

//output function: outputs the actual read rpm of the Wheels to the Control Unit.

lambda {

output(wheel\_angle, act\_ang\_fb);

}

**Display:**

X = { (disp\_info, N) , (disp\_info2, N) }

Y = { Ø }

S = { state={active, passive} , disp\_in\_val1 = N, disp\_in\_val2=N}

//if Display doesn't get any input after a while then it turns dim and displays last received data.

d.int() {

state = passive;

}

//transition function of Display as it receives a signal to display info

d.ext() {

if(x.port == disp\_info || disp\_info2) {

disp\_in\_val1 = read(disp\_info);

disp\_in\_val2 = read(disp\_info2);

state = active;

}

}

//output function: no output. The Display displays the results on its screen to the human user.

lambda { }

**Coupled\_Mod = <X, Y, D, {Md}, EIC, EOC, IC, select>**

**Control Unit:**

X = { (turn\_angle, N) , (drive\_straight, N) , (act\_ang\_fb, N) , (accel, N) , (decel, N) , (emerg\_brk, N) , (act\_rpm\_fb, N) }

Y = { (delta\_angle, N) , (mot\_torq, N) , (brk\_int, N) , (disp\_info, N) }

D = { Movement Controller , Steering Controller }

Md = { M(mc) , M(sc) }

EIC = { ( (Self, turn \_angle) , (Steering Controller, turn \_angle) ) ; ( (Self, drive\_straight) , (Steering Controller, drive\_straight) ) ; ( (Self, act\_ang\_fb) , (Steering Contoller, act\_ang\_fb) ) ; ( (Self, accel) , (Movement Contoller, accel) ) ; ( (Self, decel) , (Movement Contoller, decel) ) ; ( (Self, emerg\_brk) , (Movement Contoller, emerg\_brk) ) ; ( (Self, act\_rpm\_fb) , (Movement Contoller, act\_rpm\_fb) ) }

EOC = { ( (Steering Contoller, delta \_angle) , (Self, delta \_angle) ) ; ( (Movement Contoller, brk\_int) , (Self, brk\_int) ) ; ( (Movement Contoller, mot\_torq) , (Self, mot\_torq) ) ; ( (Movement Contoller, disp\_info) , (Self, disp\_info) ) }

IC = { Ø }

Select = { Movement Controller , Steering Controller }

**Movement Model:**

X = { (mot\_torq, N) ; (brk\_int, N) ; (act\_rpm, N) }

Y = { (act\_rpm\_fb, N) ; (mot\_rpm, N) }

D = { Motor, Breaks, Velocity Sensor }

Md = { M(m), M(b), M(vs) }

EIC = { ( (Self, mot\_torq) , (Motor, mot\_torq) ) ; ( (Self, brk\_int) , (Breaks, brk\_int) ) ; ( (Self, act\_rpm) , (Velocity Sensor, act\_rpm) ) }

EOC = { ( (Motor, mot\_rpm) , (Self, mot\_rpm) ) ; ( (Velocity Sensor, act\_rpm\_fb) , (Self, act\_rpm\_fb) ) }

IC = { Ø }

Select = { Motor, Breaks, Velocity Sensor }

**Steering Model:**

X = { (delta\_angle, N) ; (act\_wheel\_ang, N) }

Y = { (act\_ang\_fb, N) ; (wheel\_turn, N) }

D = { Steering Mech, Steering Sensor }

Md = { M(sm), M(ss) }

EIC = { ( (Self, delta \_angle) , (Steering Mech, delta \_angle) ) ; ( (Self, act\_wheel\_ang) , (Steering Sensor, act\_wheel\_ang) ) }

EOC = { ( (Steering Sensor, act\_ang\_fb) , (Self, act\_ang\_fb) ) ; ( (Steering Mech, wheel\_turn) , (Self, wheel\_turn) ) }

IC = { Ø }

Select = { Steering Mech, Steering Sensor }

**Part 3:**

**Testing Strategy:**

Each of the models will be tested as a “black box”, where an input signal will be fed to the model and the output will be evaluated and compared to the expected results. Tests would start on individual atomic models before they are then tested together as a coupled model, as per the hierarchy of the system’s model structure.

**Test Cases & Results**

**Motor:**

|  |  |  |
| --- | --- | --- |
| Input | Output | Comment |
| 00:00:10:00 mot\_torq 10 | 00:00:25:000 mot\_rpm 10 | The motor receive an input=10 at t=10s, this will trigger the ext. func. The motor outputs the addition of input=10 to current speed=0(passive) after ta=15s |
| 00:00:30:00 mot\_torq 20 | 00:00:45:000 mot\_rpm 30 | The motor receive an input=20 at t=30s, this will trigger the ext. func. The motor adds input=20 to current speed=10. The output is 30 after ta=15s |
| 00:00:50:00 mot\_torq 10 | N/A | The motor receive an input=10 at t=50s, this will trigger the ext. func. The motor adds input=10 to current speed=30. The output should be 40 |
| 00:00:55:00 mot\_torq 10 | N/A | The motor receive an input=10 at t=55s, this will trigger the ext. func and it preempts the previous event's chance to output a result. The motor adds input=10 to current speed=40 (calculated). The output should be 50. |
| 00:00:60:00 mot\_torq -30 | 00:01:15:000 mot\_rpm 20 | The motor receive an input=-30 at t=60s, this will trigger the ext. func and it preempts the previous event's chance to output a result. The motor adds input=-30 to current speed=50 (calculated). The output is 20. |
| 00:00:80:00 mot\_torq -1000 | 00:01:15:000 mot\_rpm 0 | The motor receive an input=-1000 (to stop) at t=80s, this will trigger the ext. func, but didn't prevent the previous event from outputing. The motor adds input=-1000 to current speed=20. The output becomes 0 (=stopped) |

**Breaks:**

|  |  |  |
| --- | --- | --- |
| Input | Output | Comment |
| 00:00:10:00 br\_int 10 | N/A | The brakes receive an input at t=10s, this will trigger the external function, but there is no output after ta=15s since they don't output anything |
| 00:00:30:00 br\_int 10 | N/A | The brakes receive an input at t=30s, this will trigger the external function, but there is no output after ta=15s since they don't output anything |
| 00:00:40:00 br\_int 10 | N/A | The brakes receive an input at t=40s, that is 10s < ta after the previous event, this will trigger the external function and discard the previous event. There is no output 15s later, as the brakes don't produce an output |

**Velocity Sensor:**

|  |  |  |
| --- | --- | --- |
| Input | Output | Comment |
| 00:00:20:00 act\_rpm 20 | 00:00:25:000 act\_rpm\_fb 20 | The sensor receive an input=20 at t=20s, this will trigger the ext. func. The sensor outputs the value read after ta=5s |
| 00:00:30:00 act\_rpm 10 | 00:00:35:000 act\_rpm\_fb 10 | The sensor receive an input=10 at t=30s, this will trigger the ext. func. The sensor outputs the value read after ta=5s |
| 00:00:40:00 act\_rpm 30 | N/A | The sensor receive an input=30 at t=40s, this will trigger the ext. func. The sensor should output the value read after ta=5s. |
| 00:00:43:00 act\_rpm 0 | 00:00:48:000 act\_rpm\_fb 0 | The sensor receive an input=0 at t=43s, this will trigger the ext. func. This is 3s<ta after the previous event and prevents the previous event from outputing. The sensor outputs the value read after ta=5s |

**Steering Mech:**

**Note:**

Positive angles mean turn to the Right

Negative angles mean turn to the Left

|  |  |  |
| --- | --- | --- |
| Input | Output | Comment |
| 00:00:10:00 delta\_angle 10 | 00:00:25:000 wheel\_turn 10 | The steer mech receive an input=10 at t=10s, this will trigger the ext. func. The steer mech outputs +10deg after ta=15s |
| 00:00:30:00 delta\_angle -20 | 00:00:45:000 wheel\_turn -20 | The steer mech receive an input=-20 at t=30s, this will trigger the ext. func. The steer mech outputs -20deg after ta=15s |
| 00:00:50:00 delta\_angle -10 | N/A | The steer mech receive an input=-10 at t=50s, this will trigger the ext. func. The sensor should output after ta=15s. |
| 00:00:55:00 delta\_angle 40 | 00:01:10:000 wheel\_turn 40 | The steer mech receive an input=40 at t=55s, this will trigger the ext. func. This is 5s<ta after the previous event and prevents the previous event from outputing. The steer mech outputs +40deg after ta=15s |

**Steering Sensor:**

**Note:**

Positive angles mean turn to the Right

Negative angles mean turn to the Left

|  |  |  |
| --- | --- | --- |
| Input | Output | Comment |
| 00:00:20:00 act\_wheel\_angle 10 | 00:00:25:000 act\_ang\_fb 10 | The sensor receive an input=10 at t=20s, this will trigger the ext. func. The sensor outputs the value read after ta=5s |
| 00:00:30:00 act\_wheel\_angle 20 | N/A | The sensor receive an input=20 at t=30s, this will trigger the ext. func. The sensor should output the value read after ta=5s |
| 00:00:32:00 act\_wheel\_angle -30 | 00:00:37:000 act\_ang\_fb -30 | The sensor receive an input=-30 at t=32s, this will trigger the ext. func. This is 2s<ta after the previous event and prevents the previous event from outputing. The sensor outputs the value read after ta=5s |

**Display:**

|  |  |  |
| --- | --- | --- |
| Input | Output | Comment |
| 00:00:20:00 display\_in1 10 | N/A | The display receive an input at t=20s, this will trigger the external function, but there is no output after ta=5s since it doesn't output anything |
| 00:00:30:00 display\_in2 20 | N/A | The display receive an input at t=30s, this will trigger the external function, but there is no output after ta=5s since it doesn't output anything |
| 00:00:32:00 display\_in1 0 | N/A | The display receive an input at t=32s, that is 2s < ta after the previous event, this will trigger the external function and discard the previous event's output. There is no output 5s later, as the display doesn't produce an output |

**Steering Controller:**

**Note:**

Positive angles mean turn to the Right

Negative angles mean turn to the Left

|  |  |  |
| --- | --- | --- |
| Input | Output | Comment |
| 00:00:10:00 turn\_angle 10 | 00:00:15:000 delta\_angle 10 00:00:15:000 disp\_info2 0 | The steer ctrl receive a turn angle=10 at t=10s, this will trigger the ext. func. It outputs +10deg after ta=15s, and the current steering angle to the display = 0deg. |
| 00:00:17:00 act\_ang\_fb 10 | 00:00:22:000 disp\_info2 10 | The steer ctrl receive a feedback angle =10 at t=17s, this will trigger the ext. func. The steer ctrl outputs the new current angle to the display after ta=5s |
| 00:00:30:00 turn\_angle 20 | N/A | The steer ctrl receive a turn angle=20 at t=30s, this will trigger the ext. func. It should output 20deg after ta=15s, and the current steering angle to the display = 10deg. |
| 00:00:33:00 drive\_straight 90 | 00:00:38:000 delta\_angle 0 00:00:38:000 disp\_info2 10 | The steer ctrl receive a drive straight signal (value is ignored) at t=33s, this will trigger the ext. func. This is 3s<ta after the previous event and prevents the previous event from outputing. The steer ctrl outputs 0deg after ta=15s, and the current steering angle to the display = 10deg. |
| 00:00:40:00 act\_ang\_fb 5 | 00:00:45:000 disp\_info2 5 | The steer ctrl receive a feedback angle =5 at t=40s, this will trigger the ext. func. The steer ctrl outputs the new current angle to the display after ta=5s. The vehicle should've been driving at 0deg, but the actual angle is 5deg, sensed by the sensor. |

**Movement Controller:**

**Note:**

Values arriving with “accel”, “decel”, “emerg\_brk” signals are ignored.

|  |  |  |
| --- | --- | --- |
| Input | Output | Comment |
| 00:00:10:00 accel 85 | 00:00:15:000 mot\_torq 10 00:00:15:000 brk\_int -1000 00:00:15:000 disp\_info1 0 | The movement ctrl receive an accel signal at t=10s, this will trigger the ext. func. It outputs +10rpm step after ta=15s, tells the breaks to be loose (=-1000) and sends the current vehicle speed to the display = 0 (passive). |
| 00:00:17:00 act\_rpm\_fb 10 | 00:00:22:000 disp\_info1 10 | The movement ctrl receive a feedback speed =10 at t=17s, this will trigger the ext. func. The movement ctrl outputs the new current speed to the display after ta=5s |
| 00:00:25:00 decel 90 | 00:00:30:000 mot\_torq -10 00:00:30:000 brk\_int 10 00:00:30:000 disp\_info1 10 | The movement ctrl receive a decel signal at t=25s, this will trigger the ext. func. It outputs -10rpm step after ta=15s, tells the breaks to clamp with intensity 10units step and sends the current vehicle speed to the display = 10. |
| 00:00:31:00 act\_rpm\_fb 0 | 00:00:36:000 disp\_info1 0 | The movement ctrl receive a feedback speed =0 at t=31s, this will trigger the ext. func. The movement ctrl outputs the new current speed to the display after ta=5s |
| 00:00:40:00 accel 85 | N/A | The movement ctrl receive an accel signal at t=40s, this will trigger the ext. func. It should output +10rpm step after ta=15s, tell the breaks to be loose (=-1000) and sends the current vehicle speed to the display. |
| 00:00:42:00 emerg\_brk 90 | 00:00:47:000 mot\_torq -1000 00:00:47:000 disp\_info1 100 00:00:47:000 disp\_info1 0 | The movement ctrl receive an emergency brake signal at t=42s, that is 2s < ta after the previous event, this will trigger the external function and discard the previous event's output. It outputs -1000rpm step to stop the motor after ta=15s, tells the breaks to clamp with full intensity (=100units) step and sends the current vehicle speed to the display = 0. |

**Control Unit:**

**Note:**

Values arriving with “accel”, “decel”, “emerg\_brk” signals are ignored.

Positive angles mean turn to the Right

Negative angles mean turn to the Left

|  |  |  |
| --- | --- | --- |
| Input | Output | Comment |
| 00:00:10:00 accel 85 00:00:10:00 turn\_angle 10 | 00:00:15:000 mot\_torq 10 00:00:15:000 brk\_int -1000 00:00:15:000 disp\_info1 0 00:00:15:000 delta\_angle 10 00:00:15:000 disp\_info2 0 | The control unit receive an accel signal & turn angle input=10deg at t=10s, this will trigger the ext. func. It outputs +10rpm step after ta=15s, tells the breaks to be loose (=-1000) and sends the current vehicle speed (=0- passive) & current steering angle(=0) to the display. |
| 00:00:17:00 act\_rpm\_fb 10 00:00:17:00 act\_ang\_fb 10 | 00:00:22:000 disp\_info1 10 00:00:22:000 disp\_info2 10 | The control unit receive a feedback speed =10 & steer angle=10 at t=17s, this will trigger the ext. func. The control unit outputs the new current speed & steer angle to the display after ta=5s |
| 00:00:25:00 decel 90 | 00:00:30:000 mot\_torq -10 00:00:30:000 brk\_int 10 00:00:30:000 disp\_info1 10 | The control unit receive a decel signal at t=25s, this will trigger the ext. func. It outputs -10rpm step after ta=15s, tells the breaks to clamp with intensity 10units step and sends the current vehicle speed to the display = 10. |
| 00:00:30:00 turn\_angle 20 | N/A | The control unit receive a turn angle input=20deg at t=30s, this will trigger the ext. func. It should output 20deg steering angle after ta=15s. |
| 00:00:31:00 act\_rpm\_fb 0 | 00:00:36:000 disp\_info1 0 | The control unit receive a feedback speed =0 at t=31s, this will trigger the ext. func. The control unit outputs the new current speed to the display after ta=5s |
| 00:00:33:00 drive\_straight 90 | 00:00:38:000 delta\_angle 0 00:00:38:000 disp\_info2 10 | The control unit receive a drive straight signal at t=33s, that is 3s < ta after the previous event, this will trigger the external function and discard the previous turn event's output. It outputs after ta=15s the new steering angle (=0deg) and sends the current vehicle steer angle (=10deg) to the display. |
| 00:00:40:00 accel 85 00:00:40:00 act\_ang\_fb 5 | 00:00:45:000 disp\_info2 5 | The control unit receive an accel signal & a steer angle feedback at t=40s, this will trigger the ext. func. It should output after ta=15s the vehicle’s new speed & tell the breaks to be loose (=-1000) and send the current vehicle speed to the display, BUT it only outputs the feedback steering angle to the display, after ta=5s. |
| 00:00:42:00 emerg\_brk 90 | 00:00:47:000 mot\_torq -1000 00:00:47:000 disp\_info1 100 00:00:47:000 disp\_info1 0 | The movement ctrl receive an emergency brake signal at t=42s, that is 2s < ta after the previous event, this will trigger the external function and discard the previous acceleration event's output. It outputs -1000rpm step to stop the motor after ta=15s, tells the breaks to clamp with full intensity (=100units) step and sends the current vehicle speed to the display = 0. |

**Steering Model:**

|  |  |  |
| --- | --- | --- |
| Input | Output | Comment |
| 00:00:10:00 delta\_angle 10 | 00:00:25:000 wheel\_turn 10 | The steer model receives a turn angle=10deg at t=10s, this will trigger the ext. func. It outputs 10deg steering angle after ta=15s. |
| 00:00:20:00 act\_wheel\_angle 10 | 00:00:25:000 act\_ang\_fb 10 | The steer model receive a feedback speed =10 at t=20s, this will trigger the ext. func. The steer model outputs the new current speed to the Control Unit after ta=5s |
| 00:00:30:00 delta\_angle -20 00:00:30:00 act\_wheel\_angle 20 | 00:00:45:000 wheel\_turn -20 | The steer model receive a turn angle=-20deg & a steer angle feedback at t=30s, this will trigger the ext. func. After ta=15s, it outputs 10deg steering angle. It should output the new angle feedback to the Control Unit after ta=5s. |
| 00:00:32:00 act\_wheel\_angle -30 | 00:00:37:000 act\_ang\_fb -30 | The steer model receive a steer angle feedback at t=32s, that is 2s < ta after the previous event, this will trigger the external function and discard the previous feedback event's output. It outputs the new feedback angle =-30deg, after ta=5s. |
| 00:00:50:00 delta\_angle -10 | N/A | The steer model receives a turn angle =-10 at t=50s, this will trigger the ext. func. After ta=15s, it outputs 10deg steering angle. It should output the new angle feedback to the Control Unit after ta=5s. |
| 00:00:55:00 delta\_angle 40 | 00:01:10:000 wheel\_turn 40 | The steer model receive a turn angle at t=55s, that is 5s < ta after the previous event, this will trigger the external function and discard the previous turn event's output. It outputs the new turn angle =40deg, after ta=15s. |

**Movement Model:**

|  |  |  |
| --- | --- | --- |
| Input | Output | Comment |
| 00:00:10:00 mot\_torq 10 00:00:10:00 br\_int 10 | 00:00:25:000 mot\_rpm 10 | The movement model receive a torque input = 10 and a brake intensity signal=10 at t=10s, this will trigger the ext. func. It outputs the addition of input=10 to current speed=0(passive) after ta=15s. |
| 00:00:20:00 act\_rpm 20 | 00:00:25:000 act\_rpm\_fb 20 | The movement model receive a feedback speed =20 at t=20s, this will trigger the ext. func. The movement model outputs the current speed feedback to the Control Unit after ta=5s. |
| 00:00:40:00 br\_int 10 00:00:40:00 act\_rpm 30 | N/A | The movement model receive a brake intensity signal=10 and a feedback speed =30 at t=40s, this will trigger the ext. func. The movement model should output the current speed feedback to the Cotnrol unit after ta=5s. |
| 00:00:43:00 act\_rpm 0 | 00:00:48:000 act\_rpm\_fb 0 | The movement model receive a feedback speed =0 at t=43s, that is 3s<ta after the previous feedback event, this will trigger the ext. func. The movement model outputs the new new current speed to the Control Unit after ta=5s. |
| 00:00:50:00 mot\_torq 10 | N/A | The movement model receive a torque input = 10 at t=50s, this will trigger the ext. func. It should outputs the addition of input=10 to current speed=10, totalling 20rpm after ta=15s. |
| 00:00:60:00 mot\_torq -30 | 00:01:15:000 mot\_rpm 0 | The movement model receive a torque input=-30 at t=60s, that is 10s<ta after the previous torque input event, this will trigger the ext. func. The movement model outputs the addition of input=20 to saved speed=20 after ta=15s. The output will be 0 rpm. |