# Summer Project 2015 – Decentralized Control of Truck Vehicles

This document contains instructions for how to use the programs developed in LabVIEW and information about the VI:s included in the project.

## How to use MainComputer.vi

The front panel shows the controls and the indicators for the VI. Before starting the program, the following settings in the front panel need to be chosen:

1. *address* is the IP address of the computer that sends the parameters from the supervisory layer to the computer on which the VI is run.
2. *remote port or service name* is the number of the port that the supervisory layer uses to write the parameters for each truck. These three values should be 16000 plus the Qualisys ID for the corresponding truck.
3. When starting the program, make sure that the boolean controllers *Stop Truck* are set to TRUE (this is the default value), since that keeps the trucks from moving before everything has started in a good way and the position of the trucks is received.

In the background, a program with the controllers needs to be run from the myRIO board. Each truck demands one myRIO device and one program. These should be started before starting the MainComputer.vi, and each truck also needs to be connected to the myRIO board. The myRIO VI is called MainRIO.vi and is located in the same project. It is important that the correct truck is connected to the correct myRIO since it will use the coordinates that is received from the supervisory layer, which will correspond to the truck with the ID specified in the *remote port or service name*.

In the MainRIO.vi, the parameters for the controllers can be changed as well as the proportional parameter *Proportional constant DistanceError t*hat is used in the cruise control when the truck is in a platoon. In the VI PathController.vi, the weight between the different types of errors can be changed (see information about PathController.vi).

When running the program, all information about the trucks can be found on the front panel.

* *Received values* is a cluster that contains all the information the truck receives from the supervisory layer. The number indicates which truck the cluster applies to.   
  - *Time* is the time at which the information was sent.  
  - *Vehicle Id* is the Qualisys ID of the truck. It is important that the vehicle ID matches the one specified in the *remote port or service name*.  
  - *Desired velocity* is the velocity that the truck wants to have. If the truck is in a platoon, this velocity will be the same as for the truck in front. If the truck is behind the wanted position, it will increase the velocity until it reaches the desired distance. From then the desired velocity will be kept.  
  - *Current Distance* is the current distance to the truck it follows. If the truck is not in a platoon, the current distance is set to 1.  
  - *Desired Distance* is the distance that the truck wants to have to the truck it follows. If the truck is not in a platoon, the desired distance is set to 1.   
  - *Pose* is the current position of the truck (at the time *Time*).  
  - *Lane Offset* is used when the truck should give space for an emergency vehicle. The *Lane Offset* is a distance that the truck should displace the trajectory. A positive value means that the truck should move to the left, and a negative to the right.
* *RIOparameters* is a cluster which contains all the information that is sent to the controllers on the myRIO board. To the path controller, the *PathController inputs* are sent.   
  - The *CurrentLateralError* is the lateral distance from the truck to the closest point on the trajectory.   
  - The *AheadLateralErro*r is the lateral distance between a point *d* (by default 0.25) meters in front of the truck in its current direction and the point on the desired trajectory closest to this point.  
  - The *ThetaError* is the error between the current angle of the truck and the desired angle set by the trajectory.   
  The *VelocityController inputs* are sent to the cruise controller.  
  - *VelocityReferenc*e is the same as the *Desired Velocity* received from the supervisory layer.  
  - *EstimatedVelocity* is the estimated current velocity of the truck. It is based on the current and previously positions.  
  - *DistanceError* isthe difference between the *Desired Distance* and the *Current Distance* (negative if the truck is too far away and should increase the velocity).  
  One boolean parameter is also sent to the myRIO.  
  - *Stop Truck* overwrites the voltage that should be sent to the truck and sets it to values that keeps the truck from moving as long as the *Stop Truck* is TRUE.
* *error out* shows if there is errors with the connection.
* *data out* shows the string that is being sent from the supervisory layer and processed into the *Received values*. When this is empty, the program is not receiving values and will use the latest ones until receiving new values.
* The tabs include information about the position of each truck, the path travelled by each truck and about the errors.   
  - In the main tab, *Main*, the current position of each truck as well as the three lanes can be seen.  
  - The tab Truck includes the path travelled by the truck in white, which lane the truck is currently following in red and the trajectory for each lane. The current reference point on the trajectory can also be seen, as well as the point *d* meters in front of the truck. To the right of the graph, parameters for the truck can be read. These includes the largest distance to the trajectory (*MaximalDistance*), the RMS distance to the trajectory (*RMS error*), the distance between the ahead point and the trajectory (*DistanceAheadPointToTrajectory*), the current reference (*CurrentRefTheta*) and the current error in ( *CurrentThetaError*) and the current distance between the truck and the trajectory (*DistanceCurrentPoseToTrajectory*).

## Information about VI:s in ... project

### NewMainComputer.vi

Information about MainComputer can be find above, under heading “How to use the MainComputer.vi”.

### ReadInfo.vi

*ReadInfo.vi* processes the string that is sent from the supervisory layer and returns the parameters that the *NewMainComputer.vi* needs.

### InitializeVariables.vi

The purpose of this VI is to set the size of arrays that are being used in the program, and it is needed to define the type for variables created in the program.

### Readpath.vi

ReadPath.vi is a VI that reads the three lanes from the data files left\_lane.traj, center\_lane.traj and right\_lane.traj.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Name | Type | Description |
| Output |  |  |  |
|  | Lanes | Cluster of 3 elements: Right\_lane – 2D array of doubles.  Center\_lane – 2D array of doubles.  Left\_lane –2D array of doubles. | Each array contains the x, y and θ coordinates for each point on the trajectory for the specified lane. |

#### Details

In the Block Diagram the paths for the data files with the different lanes are set in the constant connected to the openfile.vi. The data files give the x and y coordinates for the lanes, from this coordinates the angle between the x-axis and the tangent of the trajectory (Theta) is calculated.

### Truck.vi

Truck.vi is a subVi in MainComputer.vi that calculates the parameters that are used to control the trucks. For each truck that are used one truck.vi is needed. When a new truck is added one more Truck.vi has to be added in the MainComputer.vi.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Name | Type | Description |
| Inputs |  |  |  |
|  | Truck inputs | Cluster | Contains of parameters from the Supervisory layer. |
|  | Circle inputs | Cluster | Contains of parameters from the last iteration. |
|  | Lanes | Cluster of 3 elements: Right\_lane – 2D array of doubles.  Center\_lane – 2D array of doubles.  Left\_lane –2D array of doubles. | Each array contains the x, y and θ coordinates for each point on the trajectory for the specified lane. |
| Output |  |  |  |
|  | Controller inputs | Cluster | Contains of parameters that are used in the controls. |
|  | Circle output | Cluster | Contains of parameters that are used in the next iteration. |
|  | Path of truck | XY Graph | Plot of the lanes, the path of the truck, the current trajectory, the reference point on the trajectory and the current look ahead point. |

### LaneSwitch.vi

Compares the current desired lane with the previous desired lane and returns if the truck should switch lane to the right or to the left.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Name | Type | Description |
| Inputs |  |  |  |
|  | Desired Lane | String | Contains a string that tells the desired lane. |
|  | Lane0 | Integer | An integer that represents the desired lane from the previous iteration. 1 is the left lane, 2 the center lane and 3 the right lane. |
| Output |  |  |  |
|  | Switch | Cluster of 2 elements:  SwitchRight (boolean)  SwitchLeft (boolean) | TRUE if the truck changes lane to the right, otherwise FALSE.  TRUE if the truck changes lane to the left, otherwise FALSE. |
|  | lane | Integer | Lane is the input *Desired Lane* but converted to corresponding integer.To be sent around and used as input lane0 the next iteration. |

### ChangeLane.vi

|  |  |  |  |
| --- | --- | --- | --- |
|  | Name | Type | Description |
| Inputs |  |  |  |
|  | Trajectory In | Cluster of 3 elements:  X – 1D array of doubles  Y – 1D array of doubles  Θ – 1D array of doubles | The coordinates of the points on the current trajectory. |
|  | Lanes | Cluster of 3 elements: Right\_lane – 2D array of doubles.  Center\_lane – 2D array of doubles.  Left\_lane –2D array of doubles. | Each array contains the x, y and θ coordinates for each point on the trajectory for the specified lane. |
|  | Pose (X, Y, Theta) | 1D array of doubles | The current position of the truck, in x, y and θ coordinates. |
|  | Switch | Cluster of 2 elements:  SwitchRight (boolean)  SwitchLeft (boolean) | TRUE if the truck changes lane to the right, otherwise FALSE.  TRUE if the truck changes lane to the left, otherwise FALSE. |
|  | offset | Double | Can be used to displace the trajectory, in case of for example an emergency vehicle. |
|  | Lane | Integer | The current lane, 1 corresponds to left lane, 2 corresponds to center lane and 3 to right lane. |
| Output |  |  |  |
|  | Trajectory out | Cluster of 3 elements:  X – 1D array of doubles  Y – 1D array of doubles  Θ – 1D array of doubles | The coordinates of the points on the current trajectory, that the truck should follow. |

The ChangeLane VI is used to give the truck a trajectory between the current and the new desired lane if the truck is told to switch. By knowing which lane the car currently is driving in, the position of the truck and the trajectory of each lane, the VI uses *CalculateTrajectory.vi* and creates a trajectory between the current lane and the new lane. The new trajectory is sent to the VI *EstimateErrors*, where the parameters needed for following the trajectory is being calculated. This trajectory is being followed until the truck finds a point on the new lane that is closer to the current position than any point on the trajectory between the lines. The length of the change is set by the parameter *ChangingDistance* (in *CalculateTrajectory.vi*) making the car change lane more or less quickly.

### CalculateTrajectory.vi

When the truck is switching lane, CalculateTrajectory.vi calculates a trajectory between the lanes.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Name | Type | Description |
| Inputs |  |  |  |
|  | Lane trajectories | Cluster of 3 elements:  Right\_lane – 2D array of doubles.  Center\_lane – 2D array of doubles.  Left\_lane –2D array of doubles. | Each array contains the x, y and θ coordinates for each point on the trajectory for the specified lane. |
|  | Inputs | Cluster of 4 elements:  Current Position – 1D array of doubles  Current lane – double  Switch Right – boolean  Switch Left – boolean | The current x, y and θ coordinates of the truck.  The number of the lane that the truck is currently following.  True when the truck should change lane to the right.  True when the truck should change lane to the left. |
|  | dm | Double | Sets the distance during which the truck should have reached the new lane. |
| Output |  |  |  |
|  | Trajectory | Cluster of 3 elements:  X – 1D array of doubles  Y – 1D array of doubles  Θ – 1D array of doubles | The x, y and θ coordinates of the chosen trajectory that the truck should follow. |
|  | lane | Double | The lane that the truck is following when the loop finishes. This should be passed around and sent as the input *Current lane* for the next iteration. |

#### Details

*CalculateTrajectory* is the sub that actually calculates the coordinates of the points of the new trajectory when the truck should change lane. The way between the lanes is calculated as a weighted average of the points of the old lane and the new lane. The weight function that is used is atan.

### EstimateErrors.vi

The *EstimateErrors.vi* calculates the theta angle error, the lateral error for the look ahead point and the current point.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Name | Type | Description |
| Input |  |  |  |
|  | Trajectory | Cluster of 3 elements:  X – 1D array of doubles  Y – 1D array of doubles  Θ – 1D array of doubles | The x, y and θ coordinates of the trajectory that the truck is currently following. |
|  | Circle variables | Cluster of 5 elements:  punktDistSquare – 1D array of doubles  Xtravelled – 1D array of doubles  Ytravelled – 1D array of doubles | Variables from the last iteration  Square of the distance to the closest point, from earlier iterations.  X positions of the truck from earlier iteration, which is used to plot the travelled path.  Y positions of the truck from earlier iteration, which is used to plot the travelled path. |
|  | Pose (X, Y, Theta) | 1D array of doubles | The current position of the truck, in x, y and θ coordinates. |
|  | Pose0 (X, Y, Theta) | 1D array doubles | The position of the truck the previous iteration. (The previous position is used when the current position is NaN.) |
|  | Clear graph | Boolean | Clears the graph from *PlotPath.vi* if true. |
| Output |  |  |  |
|  | Circle variables | Cluster of 5 elements:  punktDistSquare – 1D array of doubles  TrackX – 1D array of doubles  TrackY– 1D array of doubles | Variables used in the next iteration  Square of the distance to the closest point, from earlier iterations.  X positions of the truck from earlier iterations, which is used to plot the travelled path.  Y positions of the truck from earlier iterations, which is used to plot the travelled path. |
|  | PathController inputs | Cluster of 3 elements:  CurrentLateralError – Double  AheadLateralError – Double  ThetaError – Double | The lateral error in the truck's current position.  The lateral error at the truck's look ahead point.  The truck's angular error. |
|  | Plot variables | Cluster of 5 elements:  TrackX – 1D array of doubles  TrackY – 1D array of doubles  currentRefX – Double  currentRefY – Double  pointAhead – 1D array of doubles | The previous x positions of the truck.  The previous y positions of the truck.  The current x coordinate of the truck’s reference point.  The current y coordinate of the truck’s reference point.  x and y coordinates for the look ahead point. |
|  | Truck data | Cluster of 6 elements:  RMS error – Double  DistanceAheadPointToTrajectory – Double  CurrentRefTheta – Double  CurrentThetaError – Double  DistanceCurrentPoseToTrajectory – Double  MaximalDistance – Double | RMS value of the distance to the closest point on the trajectory.  Current distance between the look ahead point and the trajectory.  The current theta angle (Yaw) reference.  The current theta error.  Current distance between the Truck and the trajectory.  The maximum distance between the Truck and the trajectory since the truck started. |
|  | Errors | Cluster of 2 elements:  error\_x – Double  error\_y – Double | The current error in x position.  The current error in y position. |

#### Details

The look ahead distance is set in the Mathscript code.

##### Equations used for estimating the errors



Figure 1. Truck and the trajectory.

The distance to the closest point on trajectory *d* is calculated with equation



where *xr* and *yr* are the closest point on the trajectory, *xtruck* and *ytruck* is the current position of the truck. The angular error is given by



The lateral error is calculated with equation



To calculate the lateral error for the look ahead point equation 3 is used, but *xr*and *yr* are now a new reference point on the trajectory, *xtruck* and *ytruck* are instead *xahaedpoint* and *yahaedpoint*.

### EstimateVelocityAndAngle.vi

Estimates the current velocity and steering angle of the truck based on the current and previous positions.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Name | Type | Description |
| Inputs |  |  |  |
|  | Pose (X, Y, Theta) | 1D array of doubles | The latest position of the truck, in x, y and θ coordinates. |
|  | Circle variables in | Cluster of 4 elements:  Pose0 (X, Y, Theta) (1D array of doubles)  Omega\_0 (1D array of doubles)  v\_estimated\_0 (1D array of doubles)  t0 (double) | The position of the truck the previous iteration.  The estimated steering angle based on the previous iterations.  The estimated velocity based on the previous iterations.  The time when the previous iteration was carried out. |
|  | Time | Double | Current time. |
| Output |  |  |  |
|  | Circle variables out | Cluster of 4 elements:  v\_estimated (1D array of doubles)  Pose (X, Y, Theta) (1D array of doubles)  OmegaUt (1D array of doubles)  time (double) | Each iteration, the estimated velocity is added to this array and from these values the median value is calculated and used as estimated velocity.  Current position of the truck (x, y and θ).  Each iteration, the estimated steering angle is added to this array and from these values the median value is calculated and used as estimated steering angle.  Time that should be sent around to enter the VI as time0 the next iteration. |
|  | Estimated values | Cluster of 2 elements:  EstimatedVelocity  EstimatedSteeringAngle | The estimated velocity.  The estimated steering angle. |

#### Details

*EstimateVelocityandAngle* uses the current and the previously position to estimate the velocity. This is done every time a new position is given (20 times per second). The estimated velocity is stored in v\_estimated, and a median filter is used. How many values that should be used is set in the sub *InitializeVariables*, default is 5 values. The current steering angle is estimated by using the measured theta position. The estimated velocity is fed to the velocity controller on the myRIO board.

### PlotPath.vi

The *PlotPath.vi* plots the current position of the truck, the trucks previous path, the look ahead point, the three different lanes, the current lane, the position of the truck to follow and the position where the following truck should be.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Name | Type | Description |
| Input |  |  |  |
|  | Plot variables | Cluster of 5 elements:  TrackX – 1D array of doubles  TrackY – 1D array of doubles  currentRefX – Double  currentRefY – Double  pointAhead – 1D array of doubles | The previous x positions of the truck.  The previous y positions of the truck.  The current x coordinate of the truck’s reference point.  The current y coordinate of the truck’s reference point.  x and y coordinates for the look ahead point. |
|  | Pose (X, Y, Theta) | 1D array of doubles | The current position of the truck, in x, y and θ coordinates. |
|  | Trajectory | Cluster of 3 elements:  X – 1D array of doubles  Y – 1D array of doubles  Θ – 1D array of doubles | The x, y and θ coordinates of the trajectory that the truck should follow. |
|  | Lanes | Cluster of 3 elements: Right\_lane – 2D array of doubles.  Center\_lane – 2D array of doubles.  Left\_lane –2D array of doubles. | Each array contains the x, y and θ coordinates for each point on the trajectory for the specified lane. |
| Output |  |  |  |
|  | Path of truck | XY Graph | Plot of the lanes, the path of the truck, the current trajectory, the reference point on the trajectory and the current look ahead point. |

#### Details

### PlotTrucks.vi

Plots all trucks in the same plot, the trucks are marked as cross and the lanes are plotted as lines.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Name | Type | Description |
| Input |  |  |  |
|  | Lanes | Cluster of 3 elements: Right\_lane – 2D array of doubles.  Center\_lane – 2D array of doubles.  Left\_lane –2D array of doubles. | Each array contains the x, y and θ coordinates for each point on the trajectory for the specified lane |
|  | Trucks | Cluster of 10  Truck1 – 2D arrays doubles  Truck2 – 2D arrays doubles  Truck3 – 2D arrays doubles  And so on….. | Each array contains of the x, y and θ position of each truck. |
| Output |  |  |  |
|  | Truck positions | XY Graph | Plot of the lanes and the current positions of the trucks |

#### Details

The VI is prepared to plot up to 10 different trucks.

### MainRIO.vi

The main program on the myRIO board uses parameters calculated by the computer to control the truck. Each myRIO has one main program and can control one truck. The program receives seven parameters; *CurrentLateralError*, *AheadLateralError*, *ThetaError*, *VelocityReference*, *EstimatedVelocity* and *DistanceError.* The output is a velocity voltage and a steering voltage that is sent to myRIO and the truck.

*CurrentLateralError* - the lateral distance between the current position of the truck and the closest point on the desired trajectory.

*AheadLateralError* - the lateral distance between a point *d* meters in front of the truck in its current direction and the point on the desired trajectory closest to this point. The distance *d* is set in the VI *EstimateErrors*.

*ThetaError* - the error between the current angle of the truck and the desired angle set by the trajectory.

*VelocityReference* - the desired velocity of the truck.

*EstimatedVelocity* - the current velocity of the truck, estimated by the using the current and the previous position, see VI *EstimateVelocityAndAngle.vi*.

*DistanceError* - when the truck is set to follow another truck in a platoon, the difference between the desired distance between the trucks and the current distance is given in the parameter *DistanceError*.

### PathController.vi

A PID controller that controls the truck along a given trajectory.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Name | Type | Description |
| Inputs |  |  |  |
|  | CurrentLateralError | Double | The lateral distance between the current position of the truck and the closest point on the desired trajectory |
|  | AheadLateralError | Double | The lateral distance between a point *d* meters in front of the truck in its current direction and the point on the desired trajectory closest to this point. The distance *d* is set in the VI *EstimateErrors* |
|  | ThetaError | Double | the error between the current angle of the truck and the desired angle set by the trajectory |
|  | Reinitialize | Boolean | Reinitializes the internal parameters of the controller, such as the integrated error. By default FALSE. |
|  | PID Gains | Cluster of 3 elements | Proportional gain (Kc), integral time (Ti, min) and derivative time (Td, min). More info, see PID.vi. |
| Output |  |  |  |
|  | Steering Voltage | Double | The steering voltage for the truck is returned from the controller. |

|  |  |
| --- | --- |
|  |  |

The VI *PathController* is located on the myRIO board, and uses three different errors to make the truck move along a decided trajectory; *CurrentLateralError*, *AheadLateralError* and the *ThetaError* . The controller can be adjusted to consider one, two or all three of the errors. The default settings use the *CurrentLateralError* and the *AheadLateralError*, weighted 0.4 and 0.6 respectively. The weighted parameters are added together and sent to the PID controller. The output from the controller varies from -100 to 100, but is transformed to a value between 0.8 and 3.3 V that is sent as a steering voltage to the truck, corresponding to approximately -30 and 30 degrees, depending on which truck is being controlled.

### CruiseControl.vi

A PID controller that controls the velocity of the truck and, if in a platoon, the distance to the leader.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Name | Type | Description |
| Inputs |  |  |  |
|  | VelocityReference | Double | Setpoint for the velocity (without compensating for distance error) |
|  | EstimatedVelocity | Double | The estimated current velocity |
|  | DistanceError | Double | Desired distance to the truck to follow minus the current distance |
|  | PID Gains Velocity Control | Cluster of 3 elements | Parameters for the cruise control. Proportional gain (Kc), integral time (Ti, min) and derivative time (Td, min). More info, see PID.vi. |
|  | Reinitialize | Boolean | Reinitializes the internal parameters of the controller, such as the integrated error. By default FALSE. |
|  | K | Double | Constant that is being used to set the reference velocity based on the current distance error. |
| Output |  |  |  |
|  | Velocity Voltage | Double | The voltage that the controller returns. |
|  | VelocityReference | Double | The velocity that is sent to the controller, including compensating for the distance error. Only used as output to display what is happening. |
|  | Velocity Voltage without limit | Double | The voltage that is sent to the truck is limited. Without limitation, this would be the voltage. Used to display what is happening. |

The cruise control is a PID controller based on a reference velocity that can be controlled from supervisory layer. If the truck is following another truck, the velocity reference will be the same as for the truck it follows. The input to the controller, “ReferenceVelocity”, will be based on this but also include a term compensating for the distance error according to:

The ReferenceVelocity is limited to maximum 30 % higher than the VelocityReference, and the VelocityVoltage is limited to 3.1 V because around this value the path controller becomes unstable.