**Pseudo Code for Navigation Service**

**Navigation State Machine:**

**RunNavigationSM**

*Takes event as parameter; Returns an event*

Define the make transition variable as false

Set next navigation state to current navigation state

Set entry event kind to normal entry to new state

Set return event to current event (assume we are not consuming event)

Switch between current state

Case: current state is POSITIONING

Execute during function for this state

If an event is active

Switch between the event type of the current event

Case: ES\_NewPosition

Set next state to DRIVING

Set make transition variable as true

Option to consume event (decide later)

Break Case

End switch

End if

Break case

Case: current state is DRIVING

Execute during function for this state

If an event is active

Switch between the event type of the current event

Case: ES\_Reached\_City

Set next state to POSITIONING

Set make transition variable as true

Option to consume event (decide later)

Break Case

End switch

End if

Break case

End Switch

If we are making a state transition

Execute exit function for current state

Set current state to next state

Execute entry function for new state

End if

Return ReturnEvent

**StartNavigationSM**

*Takes event parameter of current event. Returns nothing. Does any required initialization for this state machine.*

Set volatile local variable to get debugger to display the value of CurrentEvent.

Set current state to POSITIONING (Entry State) – (assumes no history)

Call the entry function for the Entry\_State

**DuringPositioningState**

*Private function. During function for positioning state.*

Assume no re-mapping of consumption of event

If event is ES\_Entry or ES\_Entry\_History

Implement any entry actions

Start the positioning state machine

Else if event is ES\_Exit

Pass the exit event to positioning state machine

If we’re off road

Run full triangulation algorithm for coordinates and orientation

Else if we’re at a city

Run partial triangulation algorithm for orientation only

End if

Do other local exit functionality (return stepper to reference)

Else

Run positioning state machine

End if

Return NewEvent

**DuringDrivingState**

*Private function. During function for positioning state.*

Assume no re-mapping of consumption of event

If event is ES\_Entry or ES\_Entry\_History

Implement any entry actions

Start the driving state machine

Else if event is ES\_Exit

Pass the exit event to driving state machine

Do local exit functionality (stop motors)

Else

Run driving state machine

End if

Return NewEvent

**Positioning State Machine:**

**RunPositioningSM**

*Takes event as parameter; Returns an event*

Define the make transition variable as false

Set next positioning state to current positioning state

Set entry event kind to normal entry to new state

Set return event to current event (assume we are not consuming event)

Switch between current state

Case: current state is CALIBRATING

Execute during function for this state

If an event is active

Switch between the event type of the current event

Case: Step\_Timer\_Timeout

Set make transition variable as true

Option to consume event (decide later)

Break Case

Case: Step\_Calibration\_Done

Set next state to SCANNING

Set make transition variable as true

Option to consume event (decide later)

Break Case

End switch

End if

Break case

Case: current state is SCANNING

Execute during function for this state

If an event is active

Switch between the event type of the current event

Case: Step\_Timer\_Timeout

Set make transition variable as true

Option to consume event (decide later)

Break Case

End switch

End if

Break case

End Switch

If we are making a state transition

Execute exit function for current state

Set current state to next state

Execute entry function for new state

End if

Return ReturnEvent

**StartPositioningSM**

*Takes event parameter of current event. Returns nothing. Does any required initialization for this state machine.*

If the current event type is not equal to ES\_ENTRY\_HISTORY

Set current state to calibrating

End if

Call the Run positioning SM with the current event parameter

**DuringCalibratingState**

*Private function. During function for positioning state.*

Assume no re-mapping of consumption of event

If event is ES\_Entry or ES\_Entry\_History

Implement any entry actions: Read Magnet Sensor, Start step timer

Else if event is ES\_Exit

Do local exit functionality

Else

Do nothing

End if

Return NewEvent

**DuringScanningState**

*Private function. During function for positioning state.*

Assume no re-mapping of consumption of event

If event is ES\_Entry or ES\_Entry\_History

Implement any entry actions: Read IR Sensor, Start step timer

Else if event is ES\_Exit

Do local exit functionality

Else

Do nothing

End if

Return NewEvent

**Driving State Machine:**

**RunDrivingSM**

*Takes event as parameter; Returns an event*

Define the make transition variable as false

Set next driving state to current driving state

Set entry event kind to normal entry to new state

Set return event to current event (assume we are not consuming event)

Switch between current state

Case: current state is WAITING2DRIVE

Execute during function for this state

If an event is active

Switch between the event type of the current event

Case: ES\_OFFROAD

Set next state to OFFROAD

Set make transition variable as true

Option to consume event (decide later)

Break Case

Case: ES\_WIREFOLLOW

Set next state to WIREFOLLOW

Set make transition variable as true

Option to consume event (decide later)

Break Case

End switch

End if

Break case

Case: current state is OFFROAD

Execute during function for this state

If an event is active

Switch between the event type of the current event

Case: ES\_FoundCityFreq

Set next state to WAITING2DRIVE

Set make transition variable as true

Option to consume event (decide later)

Break Case

End switch

End if

Break case

Case: current state is WIREFOLLOW

Execute during function for this state

If an event is active

Switch between the event type of the current event

Case: ES\_FoundCityFreq

Set next state to WAITING2DRIVE

Set make transition variable as true

Option to consume event (decide later)

Break Case

End switch

End if

Break case

End Switch

If we are making a state transition

Execute exit function for current state

Set current state to next state

Execute entry function for new state

End if

Return ReturnEvent

**StartDrivingSM**

*Takes event parameter of current event. Returns nothing. Does any required initialization for this state machine.*

Set current state to Waiting2Drive

Call the Run Driving SM with the current event parameter

**DuringWaiting2DriveState**

*Private function. During function for positioning state.*

Assume no re-mapping of consumption of event

If event is ES\_Entry or ES\_Entry\_History

Implement any entry actions: Stop wheel motors

Else if event is ES\_Exit

Do local exit functionality

Else

Do nothing

End if

Return NewEvent

**DuringOffRoadState**

*Private function. During function for positioning state.*

Assume no re-mapping of consumption of event

If event is ES\_Entry or ES\_Entry\_History

Do entry functionality (we moved driving functionality to during)

Else if event is ES\_Exit

Do local exit functionality

Else

Implement any during actions: Drive to travel coords along shortest path

End if

Return NewEvent

**DuringWireFollow**

*Private function. During function for positioning state.*

Assume no re-mapping of consumption of event

If event is ES\_Entry or ES\_Entry\_History

Do entry functionality (we moved driving functionality to during)

Else if event is ES\_Exit

Do local exit functionality

Else

Implement any during actions: Drive along wire until city found

End if

Return NewEvent