|  |
| --- |
|  |
| B-ACS Node Specification |

# Contents

[Contents 2](#_Toc496790773)

[1 Introduction 3](#_Toc496790774)

[2 Node Diagrams 3](#_Toc496790775)

[2.1 Pod Demand 3](#_Toc496790776)

[3 Control Nodes 3](#_Toc496790777)

[3.1 Auto Demand 3](#_Toc496790778)

[3.2 Arduino 5](#_Toc496790779)

[3.3 TurnigyJoy 6](#_Toc496790780)

[3.4 Control Switch 7](#_Toc496790781)

[3.5 Dead Man’s Handle 8](#_Toc496790782)

[3.6 Demand Limiter 9](#_Toc496790783)

[3.7 Age Checker 10](#_Toc496790784)

[3.8 Auxiliary Switch 10](#_Toc496790785)

[4 Localisation Nodes 11](#_Toc496790786)

[4.1 GPS to Local 11](#_Toc496790787)

[4.2 GPS Path Replay 11](#_Toc496790788)

[5 Lutz Nodes 12](#_Toc496790789)

[5.1 General Messages 12](#_Toc496790790)

[5.2 Socket CAN Bridge 12](#_Toc496790791)

[5.3 Pod RX 13](#_Toc496790792)

[5.4 Pod TX 14](#_Toc496790793)

[5.5 Session Control 15](#_Toc496790794)

[5.6 Handshake 16](#_Toc496790795)

[5.7 Message Builder 16](#_Toc496790796)

[5.8 Pod State 18](#_Toc496790797)

[6 Sensor Nodes 18](#_Toc496790798)

[6.1 UDP 18](#_Toc496790799)

[6.2 xNav550 19](#_Toc496790800)

[6.3 Ibeo Message 19](#_Toc496790801)

[6.4 Ibeo 20](#_Toc496790802)

[6.5 Bumblebee Driver 20](#_Toc496790803)

[6.6 Camera 1394 20](#_Toc496790804)

[6.7 Bumblebee XB3 20](#_Toc496790805)

[6.8 Mobileye 21](#_Toc496790806)

[7 User Interface Nodes 21](#_Toc496790807)

[7.1 Web Server 21](#_Toc496790808)

[7.2 Console UI 21](#_Toc496790809)

[7.3 Text to Speech 21](#_Toc496790810)

[7.4 Key Entry Node 22](#_Toc496790811)

[7.5 Version 22](#_Toc496790812)

[8 Dead Code 23](#_Toc496790813)

# Introduction

# Node Diagrams

## Pod Demand



# Control Nodes

## Auto Demand

#### acs/auto\_demand

This is the central control node of the ACS.

It publishes steer and speed demand based on current location and current goal.

where is the lateral error to the goal, r is the distance to the goal and is the difference between goal heading and current heading.

### Subscribed Topics

#### acs/location : geometry\_msgs/Pose2D

Current location of the pod.

#### acs/goal : geometry\_msgs/Pose2D

Current goal pose of the pod.

### Published Topics

#### acs/demand : pod/PodDemand

Steer (desired curvature in rad/m) and speed (m/s) demand.

##### Frequency

As acs/location

#### aux\_demand : pod/AuxiliaryDemand

### Parameters

|  |  |  |
| --- | --- | --- |
|  | default |  |
| double maxOffset | 2 | Maximum distance between current location and goal. (m)  Beyond this distance the speed is set to zero. |
| double lateralGain | 0.5 | Curvature gain for lateral error. (rad/m). |
| double headingGain | 0.05 | Curvature gain for heading error (/m). |
| double speed | 1.2 |  |

### TODO

* Implement Aux demand.
* Refine control based on experiment.

## Arduino

An Arduino is used to communicate with a radio control receiver and the dead man’s handle hardware.

The radio control receiver outputs a PPM[[1]](#footnote-2) signal that encodes the eight analogue channels received from the controller. These values are read and published.

The dead man’s handle hardware provides a single analogue voltage into pin A0. This value is read and published.

### Subscribed Topics

None

### Published Topics

#### arduino/rc\_in

Radio PPM channels

##### Message

arduino/rcReader

|  |  |
| --- | --- |
| uint32 age | age in µs of these data |
| int16 ch1,2,3,4,5,6,7,8 | channel values |

##### Frequency

As fast as Arduino allows.

#### arduino/deadMansHandle

##### Message

std\_msgs/Int16

Single short reporting the analogue voltage as a number from 0 to 1023.

##### Frequency

As fast as Arduino allows.

#### arduino/diagnostics

Diagnostics of rc\_in

##### Message

arduino/arduinoDiagnostics

|  |  |
| --- | --- |
| uint32 blocks | number of times a new block has been started |
| uint32 count | number of times a new block has been completed with all signals |
| uint32 repeats | number of times a new block was completed while being read requiring a reread. |

##### Frequency

As fast as Arduino allows.

### Parameters

None

### TODO

* Consider changing message to be an array of channels with a configurable count.
* Investigate frequency at which messages are published. This currently appears to be as fast as the Arduino can go. This should be reduced to a fixed rate.

## TurnigyJoy

#### pod/turnigy\_joy

A driver for a Turnigy RC controller. The analogue channels of the controller are connected to an Arduino. The Arduino publishes those values, which this node subscribes to. The channel to stick/button mapping is hardcoded in this node.

A pod demand is published with each arriving update.

The central dial defines the speed limit from slow\_speed up to max\_speed for forward direction. Reverse speed limit is fixed at max\_reverse.

The right stick up-down defines throttle demand. This ratio provides a multiplier for the speed limit to produce speed demand, with a dead zone around the centre stick position.

The left stick left-right defines steer, with a dead zone around the centre stick position where zero steer will be published.

Control is abdicated if both left and right switches are in the centre position. Left and right switches otherwise turn on the indicators in the down position.

### Subscribed Topics

#### arduino/rc\_in : arduino/rcReader

### Published Topics

#### pod/turnigy\_pod\_demand : pod/PodDemandSource

#### pod/turnigy\_auxiliary\_demand : pod/AuxiliaryDemand

### Parameters

|  |  |  |
| --- | --- | --- |
|  | default |  |
| float slow\_speed | 5 | Maximum speed when central dial is in minimum position (km/h) |
| float max\_speed | 24 | Maximum speed when central dial is in maximum position (km/h) |
| float max\_reverse | 5 | Maximum reverse speed (km/h) |
| uint16 dead\_zone\_throttle | 100 | If absolute throttle input is less than this value, it is taken as zero. |
| uint16 dead\_zone\_steer | 20 | If absolute steer input is less than this value, it is taken as zero. |
| uint16 max\_steer | 100 |  |
| uint16 min\_stick | 0 | Stick position for minimum value |
| uint16 max\_stick | 1000 | Stick position for maximum value |
| uint16 centre\_stick | 500 | Stick position for zero value |

### TODO

-

## Control Switch

#### pod/control\_switch

The control switch has one or more prioritised pod demand sources in a prioritised list.

Each source will publish the pod\_demand\_source message. This message contains a pod\_demand and an abdicate Boolean.

At any time the active source is the highest priority source that is not abdicating; its pod\_demand is published by the node. If any topic has not published for 1s (configurable) then no lower priority source can be active. If there is no active source then speed and steer of zero will be published.

### Subscribed Topics

#### As sources parameter : pod/PodDemandSource

### Published Topics

#### pod/pod\_demand : pod/PodDemand

### Parameters

|  |  |  |
| --- | --- | --- |
|  | default |  |
| string sources |  | A semicolon separated list of topics that provide pod\_demand\_source messages. They are in priority order, highest priority first. |

### TODO

-

## Dead Man’s Handle

#### dead\_mans\_handle/deadMansHandle

The dead man’s handle node receives handle position from the Arduino node and publishes a speed limit based on this value.

### Subscribed Topics

#### arduino/deadMansHandle : std\_msgs/Int16

### Published Topics

#### pod/speed\_limit

##### Message

deadMansHandle/SpeedLimit

|  |  |
| --- | --- |
| Header header |  |
| float32 speedLimit | Fraction (0-1) of the requested speed that the vehicle should drive. |

##### Frequency

As arduino/deadMansHandle topic.

### Parameters

|  |  |  |
| --- | --- | --- |
|  | default |  |
| float rangeMin | 450 |  |
| float rangeMax | 950 |  |

### TODO

* Consider package
* Change name to dead\_mans\_handle.

## Demand Limiter

#### demand\_limiter/demandLimiter

Subscribes to a pod\_demand topic and publishes a pod\_demand message for each one received. The published demand has the same steer as the incoming demand.

If the ratio parameter is true, the incoming speed is multiplied by the most recent speedLimit ratio to produce the published speed.

If the ratio parameter is false, the published speed is the input speed or maximum speed times the speedLimit ration, whichever is smaller.

The timestamp of the published demand is the older of the two information sources.

### Subscribed Topics

#### pod/pod\_demand : pod/PodDemand

Requested pod demand.

#### pod/speed\_limit : deadMansHandle/SpeedLimit

Speed limit.

### Published Topics

#### pod/pod\_demand\_limited

##### Message

pod/PodDemand

##### Frequency

As pod/pod\_demand topic.

### Parameters

|  |  |  |
| --- | --- | --- |
|  | default |  |
| bool ratio | true |  |
| float maxSpeed | 24 |  |

### TODO

* Change the name to demand\_limiter.
* Consider package

## Age Checker

#### age\_checker/ageChecker

The age checker republishes all incoming messages provided that they are no older than age\_to\_start\_ramp. Once the most recent demand is older that age\_to\_start\_ramp the speed in the demand is reduced linearly down to zero after age\_to\_stop.

### Subscribed Topics

#### pod/pod\_demand\_limited : pod/PodDemand

### Published Topics

#### pod/pod\_demand\_age\_checked : pod/PodDemand

### Parameters

|  |  |  |
| --- | --- | --- |
|  | default |  |
| int rate | 25 | Publish rate when demand has not been received for longer than age\_to\_start\_ramp (Hz) |
| float age\_to\_start\_ramp | 0.2 | Start time for considering demand ‘old’ (s) |
| float age\_to\_stop | 2 | Time from demand to reach zero speed (s) |

### TODO

* Subscribe to generic PodDemand message and set correct value in launch file.

## Auxiliary Switch

#### pod/auxiliary\_switch

This node multiplexes multiple auxiliary demand inputs by passing on indicator or horn signals if any of the subscribed topics require it.

### Subscribed Topics

#### pod/xxx\_auxiliary\_demand : pod/AuxiliaryDemand

### Published Topics

#### pod/auxiliary\_demand : pod/AuxiliaryDemand

### Parameters

None

### TODO

* Wheel light functionality.
* Generalise by naming the topics as parameters.
* Pass on demand when demand changes rather than at fixed frequency.

# Localisation Nodes

## GPS to Local

#### acs/gps\_to\_local

Node to translate incoming GPS co-ordinates into a location in a local frame. Currently, the x-axis is assumed to be eastward.

where is the radius of the earth (6371000m), is the origin in local co-ordinates, are the GPS co-ordinates of the origin.

Heading of GPS co-ordinates is clockwise from north. is the heading in the local frame, anti-clockwise from east.

### Subscribed Topics

#### oxts/batchB : oxts::BacthB

Custom GPS fix is used as this contains heading as

### Published Topics

#### acs/location : geometry\_msgs::Pose2D

Location in the local frame.

### Parameters

The node waits for the global /acs/origin parameter to be set before starting. This vector of six doubles containing . Currently and are ignored.

### TODO

Allow frames that are not parallel to east.

## GPS Path Replay

#### acs/gps\_path\_replay

Reads a CSV file containing a sequence of GPS co-ordinates (lat, lng, heading).

Starting at the start of the list the points are published in turn as a goal as soon as they come into range. Published goals are in the local frame.

The first co-ordinate in the file is taken as the origin of the local frame.

### Subscribed Topics

#### acs/location : geometry\_msgs:Pose2D

Current location of the pod.

### Published Topics

#### acs/goal : geometry\_msgs:Pose2D

Next point on the path to act as goal for control.

### Parameters

|  |  |  |
| --- | --- | --- |
|  | default |  |
| string pathfile | - | Path to the CSV file containing the path. |
| double look\_ahead | 2 | Distance from current location to publish the next point on the path. (m) |

This node sets the global /acs/origin parameter to be the first point in the file. This vector of six doubles containing , all but the first two elements are set to zero.

### TODO

# Lutz Nodes

## General Messages

#### PodHeader

Most messages communicated from the ACS to the pod via CAN have a common headerof two bytes, in addition to the CAN id for the message.

|  |  |
| --- | --- |
| uint32 id | CAN id of the message. |
| uint8 frame\_count | Common frame that this message forms part of. |
| uint8 session\_id | Current session identifier |

## Socket CAN Bridge

This is a third party node that publishes messages received from a CAN bus onto a single topic and sends messages received from a subscribed topics onto the CAN bus.

Details can be found on the [ROS Wiki](http://wiki.ros.org/socketcan_bridge).

### Subscribed Topics

#### sent\_messages : can\_msgs/Frame

### Published Topics

#### received\_messages : can\_msgs/Frame

### Parameters

|  |  |  |
| --- | --- | --- |
|  | default |  |
| string can\_device | can0 | Name of SocketCAN device. |

### TODO

-

## Pod RX

This node translates the raw byes from the CAN bus (via the Socket CAN Bridge) into meaningful ROS topics. There is a one-to-one relationship of CAN IDs to topics and a message published for each known packet received.

### Subscribed Topics

#### pod\_can\_rx : can\_msgs/Frame

### Published Topics

#### aux\_power : lutz::AuxPower

#### battery : lutz::Battery

#### debug : lutz::Debug

#### debug2 : lutz::Debug

#### indicator : lutz::Indicator

#### epb : lutz::ParkBrake

#### epb\_ver : lutz::Version

#### epi\_ver : lutz::Version

#### epo\_ver : lutz::Version

#### eps\_inboard : lutz::Steer

#### eps\_outboard : lutz::Steer

#### eps : lutz::EPS

#### fault\_codes : lutz::FaultCodes

#### front\_ultrasonic : lutz::Ultrasonics

#### gwy\_ver : lutz::Version

#### pod\_handshake : lutz::Handshake

#### imu1 : lutz::Imu1

#### imu2 : lutz::Imu2

#### powertrain : lutz::Powertrain

#### rear\_ultrasonic : lutz::Ultrasonics

#### status : lutz::Status

#### vms\_ver : lutz::Version

#### wdg\_ver : lutz::Version

#### wheel\_count : lutz::WheelCount

#### whl\_ver : lutz::Version

### Parameters

None

### TODO

-

## Pod TX

This node translates meaningful ROS messages on subscribed topics to the raw byes forthe CAN bus (via the Socket CAN Bridge). There is a one-to-one relationship of CAN IDs to topics and a CAN frame is published for every subscribed message.

### Subscribed Topics

#### aux\_power\_request : lutz/AuxPowerRequest

#### control\_command1 : lutz/ControlCommand1

#### control\_command2 : lutz/ControlCommand2

#### debug\_emu\_control\_request : lutz/DebugEmuControlRequest

#### acs\_handshake : lutz/Handshake

#### session\_control : lutz/SessionControl

### Published Topics

#### pod\_can\_tx : can\_msgs/Frame

### Parameters

None

### TODO

-

## Session Control

To get the Lutz pod ready for drive-by-wire the system must be taken through various states. This requires actions to be performed by the safety driver.

This node subscribes to a pod\_demand which determines whether drive-by-wire mode is required. If it is then this node takes the system (through publishing on session control) and the user (through publishing to ui\_code) through the necessary steps to get the pod ready for drive-by-wire.

### Subscribed Topics

#### pod\_handshake

#### battery

#### eps

#### eps\_inboard

#### eps\_outboard

#### powertrain

#### epb

#### control\_command1

#### status

#### pod/pod\_demand\_limited

### Published Topics

#### session\_control : lutz/SessionControl

#### ui\_code : lutz/UICode

### Parameters

None

### TODO

* By default this node should subscribe to a generic pod\_demand. Launch files should switch this to the appropriate topic name.

## Handshake

In order for the ACS to control the Pod over the CAN bus, a successful challenge and response must be carried out in a timely fashion. Each side of this handshake maintains a current quality of service (QOS) assessing the response time of this interaction. A QOS of at least 15 is required for drive-by-wire operation.

The details of this interaction are given in the pod documentation.

### Subscribed Topics

#### pod\_handshake : Handshake

### Published Topics

#### acs\_handshake

##### Message

Handshake

|  |  |
| --- | --- |
| Header header |  |
| PodHeader pod\_header |  |
| uint8 seed |  |
| uint8 key\_status |  |
| uint32 key\_response |  |
| uint8 qos |  |

##### Frequency

20ms

### Parameters

None

### TODO

* Make this node monitor many other nodes to ensure that they are still active. Kill handshake if that fails.

## Message Builder

The message builder listens to a number of information sources and publishes those data as a series of messages that correspond to CAN messages on a one-to-one basis.

### Subscribed Topics

#### session\_control : lutz/SessionControl

#### pod/pod\_demand\_limited : pod/PodDemand

#### pod/auxiliary\_demand : pod/AuxiliaryDemand

#### status : lutz/Status

### Published Topics

#### aux\_power\_request

##### Message

lutz/AuxPowerRequest

|  |  |
| --- | --- |
| Header header |  |
| uint8 aux0,1,2,3,4,5,6,7 |  |
| uint8 mimic\_light |  |
| bool left\_indicator |  |
| bool right\_indicator |  |
| bool horn |  |

##### Frequency

20ms

#### control\_command1

##### Message

lutz/ControlCommand1

|  |  |
| --- | --- |
| Header header |  |
| uint8 throttle\_pot |  |
| uint8 direction |  |
| bool horn |  |
| bool park\_brake\_on |  |
| bool park\_brake\_off |  |
| bool speed\_mode |  |
| bool torque\_mode |  |
| uint8 max\_speed |  |
| uint8 torque |  |
| uint8 pod\_state\_request |  |
| uint8 torque\_limit |  |

##### Frequency

20ms

#### control\_command2

##### Message

lutz/ControlCommand2

|  |  |
| --- | --- |
| Header header |  |
| bool emergency\_stop |  |
| uint8 front\_steeer\_angle |  |
| uint8 rear\_steer\_angle |  |

##### Frequency

20ms

### Parameters

None

### TODO

* By default this node should subscribe to a generic pod\_demand. Launch files should switch this to the appropriate topic name.
* Investigate the ‘dead’ or unimplemented or repeated (horn) elements of the CAN messages. Update DBC and these messages accordingly.

## Pod State

### Subscribed Topics

### Published Topics

### Parameters

### TODO

# Sensor Nodes

## UDP

#### udp/client

Various sensors and external services use UDP as a transport layer. Often the interfaces are extensive and complex but only a small portion of the functionality is initially needed.

This node abstracts the connection to the UDP port. All incoming packets (below max\_data\_size bytes) is read and published. Sensor nodes can then subscribe to this node’s topic and interpret those bytes into domain specific information. If these data are recorded in a rosbag then they can be reprocessed with changed code in a subscribing node with additional or corrected functionality.

### Subscribed Topics

None

### Published Topics

#### udp/rx

##### Message

#### udp/Packet

|  |  |
| --- | --- |
| Header header |  |
| uint8[] data | Data in UDP packet. |

##### Frequency

20ms

### Parameters

|  |  |  |
| --- | --- | --- |
|  | default |  |
| string interface |  | Local interface to bind to. |
| string ip\_address |  | Remote IP address to connect to. |
| int port | 3000 | Remote port to connect to. |
| int timeout | 10 | Timeout on each read |
| int max\_data\_size | 16\*1024 | Maximum size of UDP packets to report. |

### TODO

-

## xNav550

#### oxts/xnav550

The xNav550 sends a stream of data packets via UDP. These packets are subscribed to via a UDP node.

Each packet of normal contains a GPS fix, velocity estimate and IMU data. These data are published at high frequency: GPS (with recorded covariance) (gps/fix), velocity (gps/vel), IMU (imu/data) and GPS, velocity and orientation (oxts/batchB).

Each packet also contains one other piece of information. The currently useful subset of these are published as they arrive, with various frequencies in the other topics.

### Subscribed Topics

#### udp/rx : udp/Packet

### Published Topics

#### gps/fix : sensor\_msgs::NavSatFix

#### gps/vel : geometry\_msgs::TwistWithCovarianceStamped

#### imu/data : sensor\_msgs::Imu

#### gps/odom : nav\_msgs::Odometry

#### oxts/gnss\_info : oxts::GnssInfo

#### oxts/kalman\_innov : oxts::KalmanInnovation

#### oxts/primary\_rx\_info : oxts::ReceiverInfo

#### oxts/secondary\_rx\_info : oxts::ReceiverInfo

#### oxts/antenna\_position : oxts::AntennaPosition

#### oxts/batchB : oxts::BatchB

### Parameters

|  |  |  |
| --- | --- | --- |
|  | default |  |
| string frame\_id | “gps” |  |
| string frame\_id\_vel | “utm” |  |

### TODO

-

## Ibeo Message

### Subscribed Topics

### Published Topics

### Parameters

### TODO

## Ibeo

### Subscribed Topics

### Published Topics

### Parameters

### TODO

## Bumblebee Driver

### Subscribed Topics

### Published Topics

### Parameters

### TODO

## Camera 1394

### Subscribed Topics

### Published Topics

### Parameters

### TODO

## Bumblebee XB3

### Subscribed Topics

### Published Topics

### Parameters

### TODO

## Mobileye

### Subscribed Topics

### Published Topics

### Parameters

### TODO

# User Interface Nodes

## Web Server

### Subscribed Topics

### Published Topics

### Parameters

### TODO

## Console UI

#### lutz/console\_ui

Displays instructions and changes in pod state received on the subscribed topic on the user console.

### Subscribed Topics

#### ui\_code : lutz/UICode

### Published Topics

None

### Parameters

None

### TODO

* Subscribe to multiple topics
* Remove pod specific elements and move from lutz package. Add a package that turns state change into ‘instruction’.

## Text to Speech

Speaks instructions received on the subscribed topic.

### Subscribed Topics

#### ui\_code : lutz/UICode

### Published Topics

None

### Parameters

None

### TODO

* Subscribe to multiple topics
* Move from Lutz
* Change 2 to ‘to’

## Key Entry Node

#### pod/key\_entry

Receives console entry from a user and publishes that string. This node is used for adding notes into a recorded bag.

### Subscribed Topics

None

### Published Topics

#### keyboard\_input : std\_msgs/String

### Parameters

None

### TODO

* Create timestamped message instead of simple string.
* Move from the Lutz package to a general TSC ROS package.

## Version

Each controller on the Lutz pod has its own version number.

Subscribes to the seven version messages published by the pod. After 1s, it prints the versions received.

### Subscribed Topics

#### xxx\_ver : lutz/Version

### Published Topics

None

### Parameters

None

### TODO

-

# Dead Code

The following nodes should be removed from the codebase or moved to a ‘test’ area.

* DummyConsoleUI
* DummyAutoRequestNode
* DummyNode

|  |
| --- |
| Tel: 01908 359 999  [www.ts.catapult.org.uk](http://www.ts.catapult.org.uk) [linkedin.com/company/transport-systems-catapult](http://linkedin.com/company/transport-systems-catapult)  Twitter: [@TSCatapult](http://twitter.com/TSCatapult) |

1. Pulse Poisition Modulation – ([Wikipedia](https://en.wikipedia.org/wiki/Pulse-position_modulation)) [↑](#footnote-ref-2)