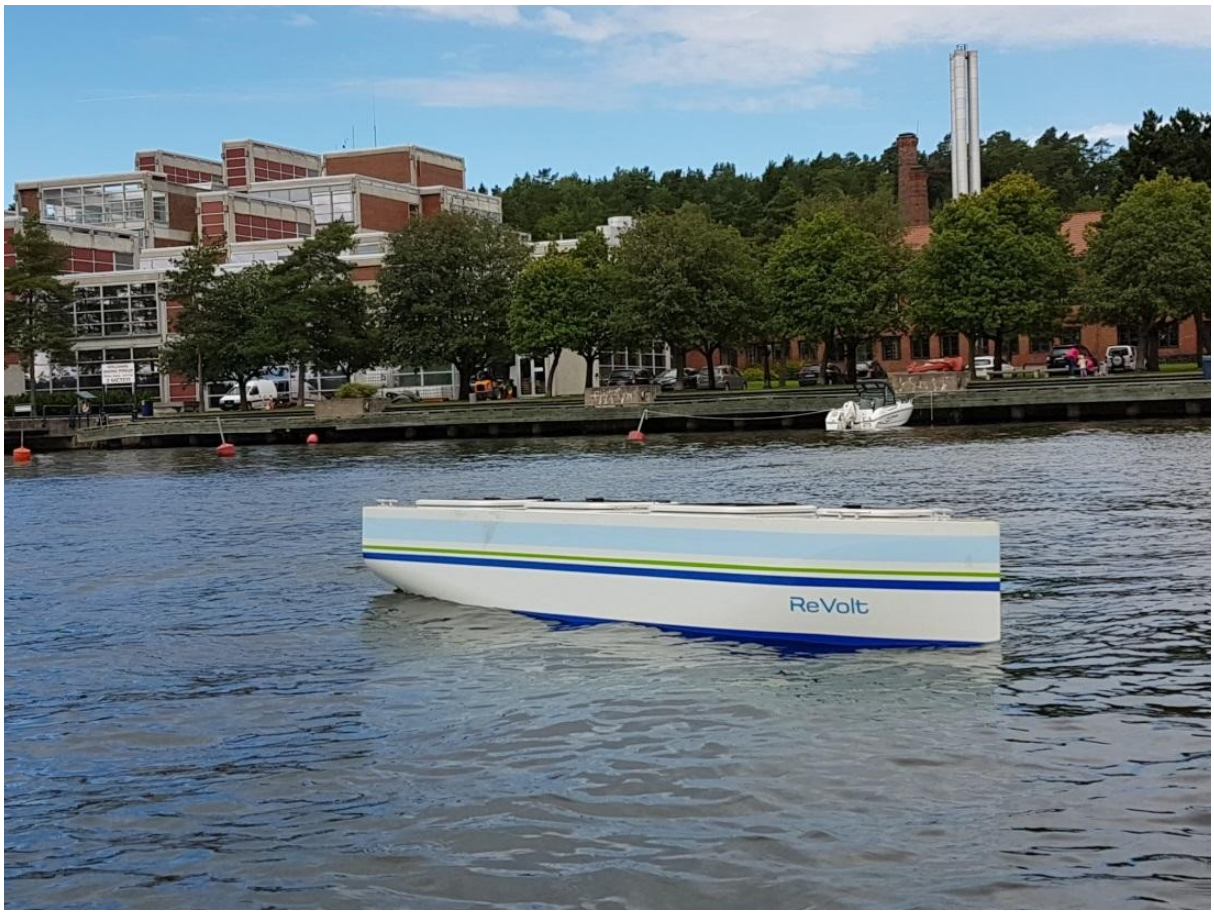
DNV·GL

REVOLT

# ReVolt User Manual

## GTR

**Rev**. 1.2
**Date:** 2018-05-07



| | |
|---|---|
| Project name: | ReVolt |
| Report title: | ReVolt User Manual |
| Date of issue: | 2018-05-07 |
| Project No.: | 10073922 |
| Organisation unit: | DNV GL Group Technology & Research |
| Rev: | 1.2 |

Applicable contract(s) governing the provision of this Report:

None

| Prepared by: | Verified by: | Approved by: |
|---|---|---|
| Tom Arne Pedersen<br>Principal Researcher | Jon Arne Glomsrud<br>Principal Researcher | Øyvind Smogeli<br>Program Director |
| [Name]<br>[title] | [Name]<br>[title] | |
| [Name]<br>[title] | [Name]<br>[title] | |



DNV GL Distribution:

☒ OPEN. Unrestricted distribution, internal and external.

☐ INTERNAL use only. Internal DNV GL document.

☐ CONFIDENTIAL. Distribution within DNV GL according to applicable contract.*

☐ SECRET. Authorized access only.

*Specify distribution:

Keywords:

[Keywords]

| Rev. No. | Date | Reason for Issue | Prepared by | Verified by | Approved by |
|---|---|---|---|---|---|
| 0 | 2016-08-12 | First issue | Kjetil Muggerud and Henrik L. Alfheim | | |
| 1 | 2017-08-17 | Second issue, updates after rewriting control software | Vegard Kamsvåg and Albert Havnegjerde | | |
| 1.1 | 2018-05-15 | Restructuring the manual, added safety considerations | Tom Arne Pedersen | Jon Arne Glomsrud<br>Trude C Helgesen | |

# TABLE OF CONTENTS

# 1 INTRODUCTION

## 1.1 Safety

Safety SHALL always come first when executing sea trials with ReVolt. The personnel attending the sea trials will need to familiarise themselves with this document before the sea trials.

## 1.2 Background

ReVolt is a concept developed by DNV GL for an unmanned and zero emission transport vessel suitable for short sea shipping, e.g. along the coast of Norway. The ship is designed and optimized for low energy demand at low speed and the cruise speed is 6 knots.

In 2014, Stadt Towing Tank delivered a 1:20 scale model of the ReVolt vessel with the same thruster configuration as the concept vessel. The model of the ReVolt has been created to allow testing and experiments when it comes to autonomy and situational awareness of autonomous vessels.

This document works as the user manual for the model ReVolt (hence forth called ReVolt). The document is divided in different parts. The first part of the document describes the sea trial, safety precautions that needs to be taken, how to launch ReVolt into the water and how to operate ReVolt. The second part describes the ReVolt, and the last part is written to give the reader an introduction to ROS and how this has been used on Revolt.

# 2 SEA TRIALS

## 2.1 Checklist for sea trials

The below checklist shall be followed before, during and after sea trial with ReVolt:

- Check forecast for weather and sea state at launch area, transfer area and test site. Always check actual sea state if there are any challenging areas where the sea trial is conducted.
- Check fuel level on following boat before staring the sea trial.
- Inspect the hull of ReVolt for any damage, inside and outside. Check around the azimuth thrusters for any damage.
- All persons onboard following boat shall use a life vest that will allow freedom to move.
- If launching ReVolt into the river Nidelven, the person standing in the river holding ReVolt needs to be aware of the strong current. Use life vest always when approaching the river.
- Remove waders when onboard the following boat
- Be aware of vessel to vessel contact, as this may introduce movement to both vessels that is unpredictable.
- Be aware of other boats during sea trial
- Assess the situation continuously during the sea trial, abort testing if necessary.

## 2.2 Test and launch areas

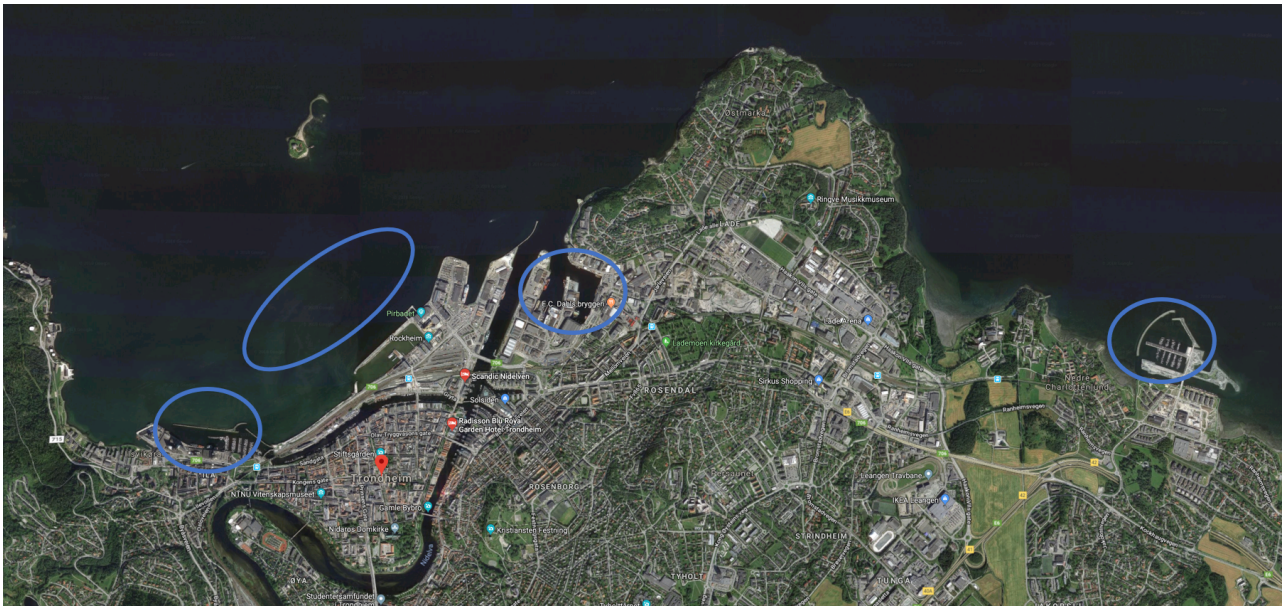



**Figure 2-1 Test area for ReVolt**


**Figure 2-2 Launch place for ReVolt**


**Figure 2-3 Tricky area with potentially rough sea**

There are several areas outside the city of Trondheim that may be used for sea trials for ReVolt, see blue circles in Figure 2 -1. The most used area has been the Dora basin, but also Grillstad and Skansen may be used. For sea trials involving other and bigger vessels than ReVolt, the area between Pirsenteret and Munkholmen may be used. If other areas are to be used for testing, contact e.g. Arve Knudsen on phone 91897842, Captain onboard Gunnerus, to check if there are any special safety precaution that needs to be taken in these new areas.

The most used launch place for ReVolt is shown in Figure 2 -2. The address is Transittgata 10, 7042 Trondheim. It is also possible to launch ReVolt using a ramp at Grillstad and Skansen.

Before launching ReVolt, the weather and sea state forecast for the actual area shall always be carefully evaluated. If strong wind or high waves are evaluated to potentially cause problem or imply uncertainty related to safety for ReVolt or the people in the following boat, sea trial with ReVolt shall not be carried out until weather conditions are improved.

Depending on the current speed of the river Nidelven and the wind direction, the waves in the mouth of Nidelven, see red circle in Figure  2 -3, may be much rougher than one should normally expect, especially if the wind is coming from the north. If this is the case, another test site should be considered, or the testing should be aborted.

## 2.3 Transport of ReVolt

The transport of ReVolt is fixed through Stefano B. Bertelli at the department of Engineering Cybernetics. He may be contacted either on phone: 91897021 or on e-mail: Stefano.bertelli@ntnu.no. The transport needs to be fixed some days before the day of sea trial.

## 2.4 Following boat


**Figure 2-4 Following boat**

When performing sea trial with ReVolt, a following boat is a necessity. A Gunnerus Workboat has been used most of the times. The boat may be rented by contacting Sten Terje Falnes, either on phone: 92297035 or by email: sten.terje.falnes@ntnu.no. NTNU requires that for renting this boat, a boating license is needed for the person operating the boat.

The price for renting the Gunnerus Workboat is 700,- NOK per day, in addition to fuel cost.

Refill fuel when the sea trial is finished.

## 2.5 Launching Revolt into water

Before launching ReVolt it is assumed that the batteries are charged and that all components in the boat are securely fastened and connected. There should be enough battery in the RC remote.

1. Inspect the hull for any damage, inside and outside. Check around the azimuth thrusters for any damage.
2. Turn on all the circuit breakers.
3. Using a laptop, connect to Revolt-Onboard and use “ssh” to access the onboard computer
4. type “launchrevolt” in the ssh-terminal, this starts roscore. Alternatively navigate to ~/revolt/src/ and type “roslaunch revolt.launch”.
5. Set RC Remote throttle in **neutral** position and turn it on.
6. Check that the system and actuators operate as expected.
7. Check that the front azimuth is in the up position.

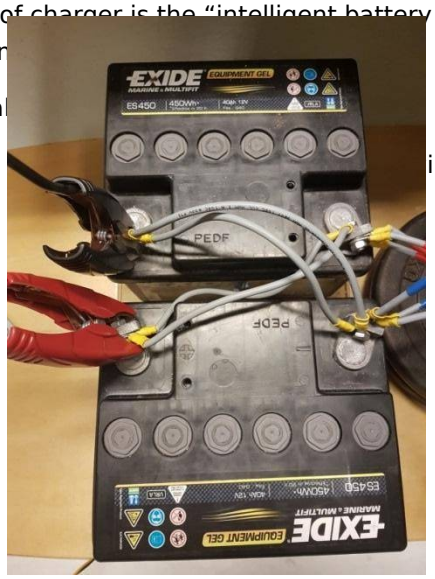
8. Close all hatches.
9. Attach a guide rope.
10. Roll the trailer into the water until the boat starts to float (To prevent damage to the ReVolt, it is important that the boat is floating when released from the trailer)
11. Unhook the boat and pull the trailer back on land while still keeping the ReVolt safe.
12. Test all actuators again.
13. Place the weights into the boat. (Recommended for stability) Watch out for any leaks.
14. Remove the guide rope and drive into the sunset.

## 2.6 Cleaning

After the ReVolt has been out to sea, the ReVolt should always be properly cleaned with fresh water on all surfaces that has been in contact with saltwater. This is to prevent rust and to ensure that the ReVolt looks good. Make sure to properly clean inside the front thruster housing and around the two stern pods.

## 2.7 Charging and power consumption

To charge the ReVolt a 12V DC charger must be used. A good choice of charger is the “intelligent battery charger” from Biltema. Remember to connect “pluss on pluss” and “r




It is recommended, but not necessary to turn off all three circuit brea

Note: When all the circuit breakers are on and the emergency switch ill draw a small current and in the end, drain the battery.

# 3 REVOLT IN A NUTSHELL

## 3.1 Components

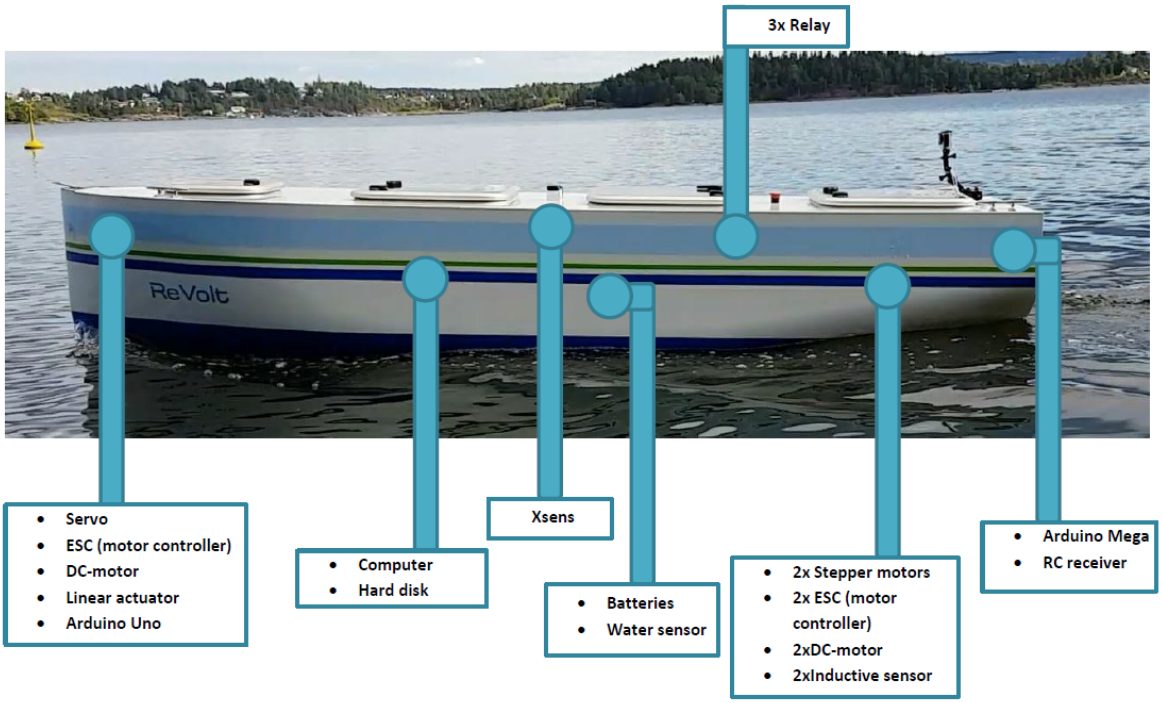



**Figure 3-5 ReVolt components**

The components are described below.

### 3.1.1 Nanotec PD2-N41

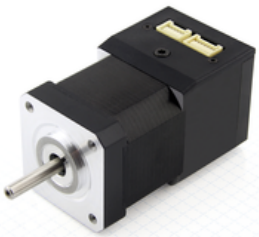


**Figure 3-6 Nanotec PD2-N41**

(http://en.nanotec.com/products/227-pd2-n41-plug-drive-high-pole-dc-servo-motor-for-rs485canopen-nema-17/)

This is the stepper motors which controls the angles of the pods in the rear end of the ReVolt.

The current motor driver for the Nanotec PD2-N41 only supports one way communication. It will not handle any error messages.

To get the initiation sequence, NanoPro ( http://en.nanotec.com/download/software/ ) has been used to set parameters and autotune the steppers. Then a RS232 communication sniffer (serial port monitor) was used to get all the commands sent to the stepper, when uploading and downloading the configuration for the each individual motor. This is then saved as a .txt file. The txt file is then read at initialization and

written to the stepper motors, whereas only the commands for position movement is used during runtime.

### 3.1.2 Xsens MTI-G-710

The Xsens is the Revolts IMU/GPS/GNSS/INS/magnetometer.

#### 3.1.2.1 Frame

The Xsens is referenced in the launch script as NorthEastDown (NED), but it's mounted upside down. This needs to be corrected which results in a 180deg rotation around the X axis which results in the following rotation matrix: $\begin{bmatrix}1&0&0\\0&-1&0\\0&0&-1\end{bmatrix}$ and is done in Xsens MT Manager / MT Settings for MT .. /

Filter Settings (https://www.xsens.com/mt-software-suite/ )

#### 3.1.2.2 Attitude Heading Reference System (AHRS)


**Figure 2-7 Attitude Heading Reference System**

The AHRS have different kinds of filter settings. It's recommended to use the GeneralMag setting which takes into account the magnetic field.

NB! If new equipment is installed or the Xsens is in an area with magnetic disturbance, a magnetic field mapper may be required.

### 3.1.3 Linear actuator

The linear actuator is driven by applying ±12V on the power input pins. The Arduino does not have a 12V output; therefore a simple circuit was created using a h-bridge. This h-bridge is placed inside the same box as the Arduino Uno. To see how this is connected look at the wiring diagram and the datasheet for the L293NE h-bridge.

### 3.1.4 Component summary

**Table 2-1 Component summary**

| Name | Placement | Model |
|---|---|---|
| Motor controller | Bow | Robbe NavyControl535R |
| DC-motor | Bow | Robbe Roxxy Starmax 48 |
| Linear actuator | Bow | Firgelli L16 |
| Servo | Bow | HiTEC HS-5485HB |
| H-bridge | Bow | L293NE |
| Motor controller | Stern | Robbe Roxxy Control 900 |
| AC-motor | Stern | Robbe Roxxy BL-outrunner 5055-45 |
| Stepper motor | Stern | Nanotec PD2-N41 |
| Current meas. sensor | 2x stern, 1x bow | Phidgets 1122_0 |
| Inductive sensor | Stern | XS618B1PAL2 |
| Xsens | Middle, top | Xsens MTi-G 710 |
| Water sensor | Under batteries | Homemade |
| Embedded computer | Middle, port side | Tank 720 |
| Hard drive | Middle, port side | Verbatim 500GB |
| 4G Router | Stern, port side | TP-Link MR200 |
| Arduino Uno | Bow | Arduino Uno R3 |
| Arduino Mega | Stern | Arduino Mega |
| Video Cameras | 2x not yet mounted | Muvi K2 Sport |
| Battery | Middle | Exide 12V 40Ah |
| Relay | Middle, starboard side | - |
| RC remote | - | Spektrum DX6i |
| RC receiver | Stern | Spektrum AR610 |

## 3.2 Operating the ReVolt

The ReVolt has four operating modes: manual control mode, heading controller mode, manual thrust allocation, and dynamic positioning. When the ReVolt is booted up the default mode is heading control mode.

To change between modes, run “rosrun rqt_reconfigure rqt_reconfigure” on the laptop. The modes can be selected in the pull-down menu in the reconfigure window. This can also be done using the laptop’s terminal and publishing to the topic /control_mode using the command: “rostopic pub /control_mode std_msgs::UInt8 ‘data: #’, where # is replaced with the integer corresponding to the desired control mode. The mapping between integers and modes is

**Table 7-2 Operating modes**

| Numerical value | Mode |
|---|---|
| 0 | Manual mode |
| 1 | Heading control mode |
| 2 | Manual thrust allocation |
| 3 | Dynamic positioning mode |
| 4 | Emergency stop mode |
| 5 | ZigZag test mode |

If the Revolt’s RC receiver loses connection with the remote the value of the throttle will be set to neutral. This is programmed into the RC receiver following the method as stated in the datasheet. Note that only the throttle value will be set to neutral, not the rudder value. So if the ReVolt loses connection when operating the Revolt using differential steering, the rudder input will still be the same as the last value it received.

### 3.2.1 Manual mode

In manual mode the user has control of all the actuators from the RC remote. In this mode it is possible to steer the ReVolt either by rotating the pods or by using differential thrust steering.

This is done with the left stick by applying some rudder input as well.

To raise and lower the front thruster the right stick (elevation) is pushed up or down. It is not necessary to hold the stick in the up or down position, one movement is enough.


### 3.2.2 Heading control

When operating the ReVolt with the heading controller, the differential throttle is disabled and it is not possible to operate the front actuators.

The reference to the heading controller is set using the right stick (aileron) on the remote. It is necessary to hold the stick down until the wanted reference is set. When the stick is let go, this will be the new reference for the ReVolt.

### 3.2.3 Manual Thrust Allocation

When ReVolt is operated in manual thrust allocation mode, the user gives direct input to the thrust allocation algorithm. In other words, the thrust vector generated by the actuators can be controlled with the left stick of the remote control, while the torque vector can be controlled using the right stick of the remote control. This mode is recommended if the user has to make fine adjustments to the position or heading of the ReVolt manually.

### 3.2.4 Dynamic Positioning Mode

In dynamic positioning mode, the position and heading of ReVolt is controlled by the dynamic positioning controller. The setpoints can be manipulated in the rqt_reconfigure window. The PID gains can also be changed in this window, if tuning is needed.

### 3.2.5 Emergency Stop Mode

This mode is automatically entered if the emergency stop push button is pressed. The main purpose of this mode is to set all outputs to neutral values, as well as letting the user know that the emergency stop button has been pressed. When the emergency stop button is released, ReVolt will default to Heading Control Mode.

### 3.2.6 ZigZag Test Mode

This mode automatically performs a zigzag-test, controlling the pod angles and thrust input accordingly. This mode will give a constant thrust of 30 % and rudder angle input of 10 degrees until the heading angle has changed by 10 degrees, upon which the rudder angle is reversed. This is repeated 5 times. The purpose of this mode is for system identification. NOTE: This mode is not fully tested, and should not be considered operational.

## 3.3 ROS on ReVolt

| | Operating system | ROS version | Password |
|---|---|---|---|
| Embedded computer | Ubuntu 16.04 LTS | Kinetic | revolt |
| Laptop | Ubuntu 16.04 LTS | Kinetic | revolt |

For more information about ROS, see Appendix A.

### 3.3.1 General information

- The workspace is named revolt and is located in the user directory: /home/ros/**revolt**
- The workspace setup.bash file is automatically sourced in .bashrc (/home/ros/.bashrc)
- The launch file is located in /src/revolt.launch and will launch everything needed to run the ReVolt
    - It is automatically launched from roscore_startup at bootup (comment to stop this )
    - The bootup script can be found at /etc/init.d/roscore_startup
- For debugging and finding errors, it's recommended to not start the launch, but run each node individually with rosrun as this will show errors and log information.

### 3.3.2 Roscore_startup

Roscore_startup is a script that is run at bootup, it is located at /etc/init.d/roscore_startup
Source /opt/ros/indigo/setup.bash
Source /home/ros/revolt/devel/setup.bash

export ROS_WORKSPACE=/home/ros/revolt
export ROS_IP=10.0.0.1
export ROS_MASTER_URI=http://10.0.0.1:11311

(cd /home/ros/revolt && source devel/setup.sh) #Might not be needed

(cd /home/ros/revolt && catkin_make) #Builds the workspace
(cd /home/ros/revolt && catkin_make actuators_firmware_stern-upload) #upload Arduino code
(cd /home/ros/revolt && catkin_make actuators_firmware_bow-upload) #upload Arduino code

(cd /home/ros/revolt/src && roslaunch revolt.launch) #launces the launch file

- This file is run at bootup
    - It will source the installed directories where ROS is installed
    - Setting the right IP addresses (see section 5.4 )
    - Builds the workspace
    - Upload code to arduinos
    - Execute launch file
- Where is roscore_startup launched from
    - /etc/init.d kjøres ved start
    - /etc/rc2d/s99.local → kjører→/etc/rc.local?
    - Pstree – startup tree. Linux command
    - Startup er lagt i rc.local filer
    - Skru av /på ros oppstart
        - Update –rc.d ro.local «enable/disable»

## 3.4 Transferring your code to Revolt

Connect your laptop to Revolt-Onboard. To transfer your ROS package folders e.g actuators/ from your laptop to Revolt, you use "scp" in the laptop-terminal.

scp -r path-to-folder-you-want-to-transfer username@ip-of-the-onboard-pc:path-to-place-folder
e.g

1. scp -r ~/revolt_ws/src/actuators/ revolt@192.168.1.100:~/revolt_ws/src/
2. enter password when prompted

You need to do this for every package folder which code you have changed.

**NOTE**: Do not transfer the entire src/ folder. It contains hidden git-files which messes things up.

### 3.4.1 Adding custom messages

In the package custom_msgs you can easily add new type of custom messages.
Look at /home/revolt/src/custom_msgs/msg/threeFloats.msg

```
float64 setpoint
float64 state
float64 error
```

It consists of three float64 types named setpoint, state and error.

This message needs to be built so we have to tell ROS to add this message. This is done in the CMakeLists.txt for this package.
located at /home/revolt/src/custom_msgs/CMakeLists.txt

```
## Generate messages in the 'msg' folder
Add_message_files(
FILES
...
###### ADDED CUSTOM msgs ####
podAngle.msg
threeFloats.msg
)
```

When adding new messages build the workspace with the command catkin_make from workspace folder ( in this case /revolt ).

New packages needs to depend on the custom_msgs package in order to use these messages.
This is done by adding dependencies in the /workspace/src/package/ (CMakelists and package.xml)
/revolt/src/actuators/CMakeLists.txt

```
Find_package( catkin RQUIRED COMPONENTS
        roscpp
        rospy
        ...
        custom_msgs                    # added dependency on custom_msgs package
)
```

/revolt/src/actuators/package.xml

```
<build_depend>.....</build_depend>
<build_depend>custom_msgs</build_depend>          #added build dependency on custom_msgs
<run_depend>.....</run_depend>
<run_depend>custom_msgs</run_depend>              # added run dependency on custom_msgs
```

Now the message can be imported in any python or cpp script for example like this in python:

Python: from custom_msgs import threeFloats

#### 3.4.1.1 Missing .h-files for custom messages in Arduino

When creating a custom message "myCustomMessage.msg" for use with an arduino in C++, you need to generate corresponding "myCustomMessage.h". Usually this is done when running "catkin_make" in the

revolt_ws (catkin workspace) directory. But in the event of this not happening, you need to build them manually, like this:

1. cd ~/revolt_ws/build/actuators/
2. rm -rf ros_lib (removes the folder containing all .h files)
3. rosrun rosserial_arduino make_libraries.py (this generates a new ros_lib folder which should include your myCustomMessage.h file)
4. cd ~/revolt_ws/
5. catkin_make

If you still have problems try deleting the build/ folder, like this:

6. cd ~/revolt_ws/
7. rm -rf build/
8. catkin_make

if it builds, you are good. Else repeat stage 1-5

**NOTE**: This needs to be done on the onboard computer as well.

### 3.4.1.2 Communication flow – advanced

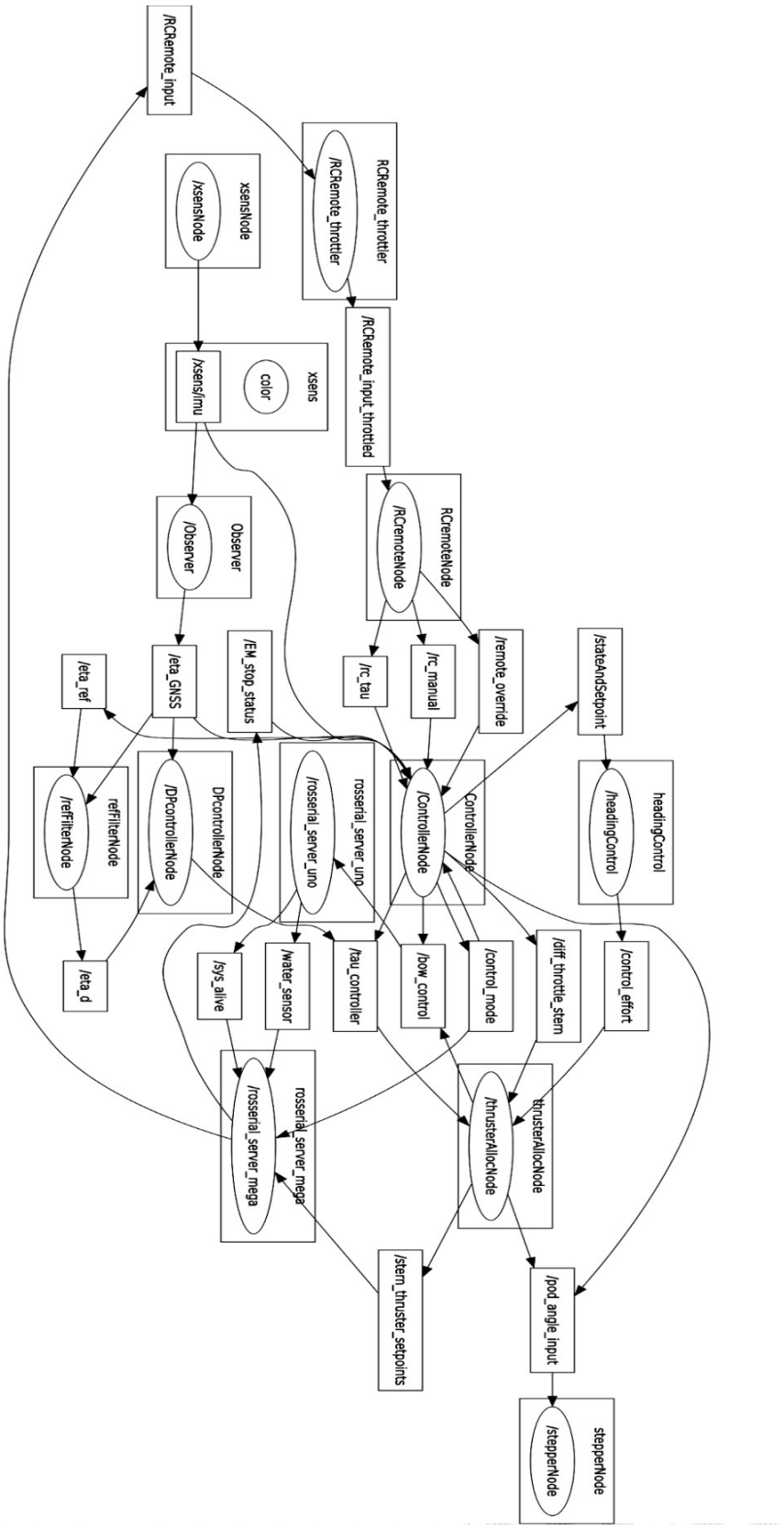



**Figure 4-8 Communication flow**

### 3.4.2 Overview of topics

| Topics | Package | Type | Publisher | Subscriber |
|---|---|---|---|---|
| /RCRemote_input | rcremote | Adc | rosserial_server_mega | RCRemote_throttler |
| /RCRemote_input_throttled | topic_tools | Adc | RCRemote_throttler | RCRemoteNode |
| /stateAndSetpoint | custom_msgs | threeFloats | ControllerNode | headingControlNode |
| /remote_override | std_msgs | Uint8 | RCRemoteNode | ControllerNode |
| /rc_manual | custom_msgs | RCManual | RCRemoteNode | ControllerNode |
| /rc_tau | custom_msgs | RCTau | RCRemoteNode | ControllerNode |
| /EM_stop_status | std_msgs | Uint8 | rosserial_server_mega | ControllerNode |
| /eta_GNSS | custom_msgs | NorthEastHeading | ObserverNode | ControllerNode DPcontrollerNode refFilterNode |
| /eta_ref | custom_msgs | NorthEastHeading | ControllerNode | refFilterNode |
| /control_effort | std_msgs | Float64 | headingControlNode | thrusterAllocNode |
| /diff_throttle_stern | custom_msgs | diffThrottleStern | ControllerNode | thrusterAllocNode |
| /control_mode | std_msgs | Uint8 | ControllerNode DynamicReconfig | DynamicReconfig ControllerNode |
| /bow_control | Custom_msgs | bowControl | ControllerNode | rosserial_server_uno |
| /tau_controller | custom_msgs | NorthEastHeading | ControllerNode DPControllerNode | thrusterAllocNode |
| /water_sensor | std_msgs | Int16 | rosserial_server_uno | rosserial_server_mega |
| /sys_alive | std_msgs | Uint64 | rosserial_server_uno | rosserial_server_mega |
| /eta_d | custom_msgs | NorthEastHeading | refFilterNode | DPcontrollerNode |
| /pod_angle_input | custom_msgs | podAngle | ControllerNode | stepperNode |
| /stern_thruster_setpoints | custom_msgs | SternThrusterSetpoints | thrusterAllocNode | rosserial_server_mega |
| | | | | |
| /motor_currents | custom_msgs | MotorCurrents | rosserial_server_uno | ---------- |
| /battery_votalge | std_msgs | Float64 | rosserial_server_mega | ---------- |
| | | | | |
| /headingControl/parameter_updates | std_msgs | Float64 | Controller node | ---------- |
| /heading_delta | std_msgs | Int16 | RC Remote node | Reference node |
| /mti/filter/orientation | custom_msgs | orientationEstimate | Xsens node | Reference node |
| /mti/filter/position | gps_common | GPSFix | Xsens node | ---------- |
| /mti/filter/velocity | custom_msgs | velocityEstimate | Xsens node | ---------- |
| /mti/sensor/gnssPvt | gps_common | GPSFix | Xsens node | ---------- |
| /mti/sensor/imu | sensor_msgs | Imu | Xsens node | ---------- |
| /mti/sensor/magnetic | | | Xsens node | ---------- |
| /mti/sensor/pressure | custom_msgs | baroSample | Xsens node | ---------- |
| /mti/sensor/sample | custom_msgs | sensorSample | Xsens node | ---------- |
| /pid_enable | std_msgs | Bool | ROS | ---------- |
| | | | | |

| | | | | |
|---|---|---|---|---|
| /pod_angles | custom_msgs | podAngle | stepperNode | ---------- |
| /rosout | Rosgraph_msgs | Log | ROS | ---------- |
| /rosout_agg | Rosgraph_msgs | Log | ROS | ---------- |
| /temperature | std_msgs | Float64 | Xsens node | ---------- |
| /diagnostics | diagnostic_msgs | DiagnosticStatus | Xsens node (heading?control) | ---------- |
| / headingControl/parameter _descriptions | std_msgs | Float64 | Controller node | ---------- |

### 3.4.3 Nodes

| Node name | Package | Programming language | Downloaded from/git source |
|---|---|---|---|
| ControllerNode (is to be renamed) | controller | C++ | |
| rosserial_server_uno | actuators | C++/Arduino | - |
| rosserial_server_mega | actuators | C++/Arduino | - |
| RC Remote node | rcremote | Python | - |
| refFilterNode | dp_controller | Python | - |
| Stepper node | actuators | Python | - |
| Reference node | headingcontrol | Python | - |
| Translate node | headingcontrol | Python | - |
| headingControl | headingcontrol | C++ | https://bitbucket.org/AndyZe/pid.git |
| Xsens node | xsens_driver | Python | https://github.com/xsens/xsens_mti_ros_node |
| vectorVS330 | nmea_navsat_driver | Python | https://github.com/ros-drivers/nmea_navsat_driver |
| DPcontrollerNode | dp_controller | Python | |
| thrusterAllocNode | dp_controller | Python | |
| Observer | observer | Python | |

## 3.5 Networking and ssh

Here is a short guide on how to connect to the ReVolts Onboard network

### 3.5.1 ROS / network setup Ubuntu 16.04 LTS

The ReVolt is connected to the onboard router TP-Link MR200 by ethernet cable. The router broadcasts Wi-fi signals with SSID "*Revolt-Onboard"* and has the IP address *192.168.1.1* (standard gateway). The WPA2-PSK network encryption key for the wireless network is "*revoltrevolt"*. The Revolt uses static IP *192.168.1.100*. Simply connect to the network and ping this address from the terminal to check connection, e.g "*ping 192.168.1.100"* .

### 3.5.2 SSH embedded computer

Using ssh-command from the terminal of your laptop, you can access the onboard computer via the Revolt-Onboard network. This is necessary when launching the Revolt or doing minor code changes at sea. The Revolt-Latitude laptop has named the IP address of Revolt, meaning that you enter "ssh revolt" and when prompted, enter the password for the onboard computer (password: revolt). You now have access to the shell of the onboard computer. If you haven't named IPs for revolt yet you need to write ssh revolt@ip-address-of-computer, e.g "ssh revolt@192.168.1.100" and enter the password when prompted. For the first time ssh'ing you need to do some additional configuration, just follow instructions displayed in the terminal.

### 3.5.3 Running ROS environment on remote computer:

(http://wiki.ros.org/ROS/NetworkSetup)

On the embedded computer the ROS_IP and ROS_MASTER_URI are on startup and terminal run set to 192.168.1.100 and http://192.168.1.100:11311. (found at /etc/init.d/roscore_startup and /home/ros/.bashrc)

On a new computer with ROS installed, ROS_IP should by default be set to localhost, but the ROS_MASTER_URI must be set to: http://192.168.1.100:11311

Helpful commands:

export ROS_MASTER_URI=http://192.168.1.100:11311

To display the current values for the ROS environment variables, use the following command:

echo $ROS_ENVIRONMENT_VARIABLE

for example

echo $ROS_MASTER_URI

### 3.5.4 Sourcing the workspace environment

In order to use commands from embedded computer on your external ROS system, the setup.bash file in workspace needs to be sourced.
Quickfix: Copy entire workspace folder from embedded computer over to external computer and run:

```
user@computer:~/workspace$ catkin_make
user@computer:~/workspace$ source devel/setup.bash
```

(work space = "revolt" on embedded computer)

### 3.5.5 Naming the IP addresses:

To make ssh simpler, add 192.168.1.100 revolt to /etc/hosts

....

```
127.0.0.1       localhost
127.0.1.1       revolt-Latitude-E6510
192.168.1.100   revolt
```

....

You can now use ssh onto the revolt without explicitly stating the IP address.

# 4 TIPS AND TRICKS

## 4.1 Errors and possible solutions

- If the Xsens is not operation properly, not sending out the correct parameters or will not start; you might have to send a setup command to the Xsens. The command is:

```
rosrun xsens_driver mtdevice.py  -m 1 –f 10
```

- The motor controller starts beeping with a steady rate and will not respond to throttle input. This may happen if the ReVolt is started with the RC remote throttle stick not in the neutral position.
  - Try to turn off all the circuit breakers, set the throttle stick to neutral and turn the circuit breakers on again. If this does not help, read the section in the motor controller datasheet on programming stick positions into the motor controller.
- Bad Wi-Fi connection between ReVolt and the laptop. Possible causes are:
  - A GoPro's Wi-Fi interfering with ReVolt's Wi-Fi.
  - The ReVolt might be too far away from the laptop.

## 4.2 More or less useful information

- Xsens
  - The Xsens is setup to publish at 10 Hz using a setup command.
  - Xsens is set to NED from the launch file.
- Default heading control PID parameters:
  - Kp: 1.5
  - Ki: 0.0
  - Kd: 5.0

## 4.3 Suggestions to improvements

- Source the setup.bash file via SSH on the laptop. This will make it possible to use ROS commands on the laptop without having to have the source code on the laptop.
- Improve the stepper drive to handle errors.
- Acquire a 6-channel RC receiver. This will make it possible to utilize the switches on the RC remote. These can be used to set the ReVolt in different modes for example.
- Create a waypoint controller with missions planner

## 4.4 Digital files

A set of files associated with the ReVolt have been gathered and created. Included in these files are, amongst others:

- Wiring diagram (.pdf and .dwg)
- Datasheets/user manuals for all components

## 4.5 Further reading/help/links

- www.ros.org
- www.wiki.ros.org
- http://wiki.ros.org/ROS/Tutorials
- http://wiki.ros.org/rosserial_arduino/Tutorials
- The respective components manuals

# APPENDIX A INTRODUCTION TO ROS

It's recommended to go through the tutorial at http://wiki.ros.org/ROS/Tutorials for a basic insight to ROS. Here is a simple introduction to ROS.

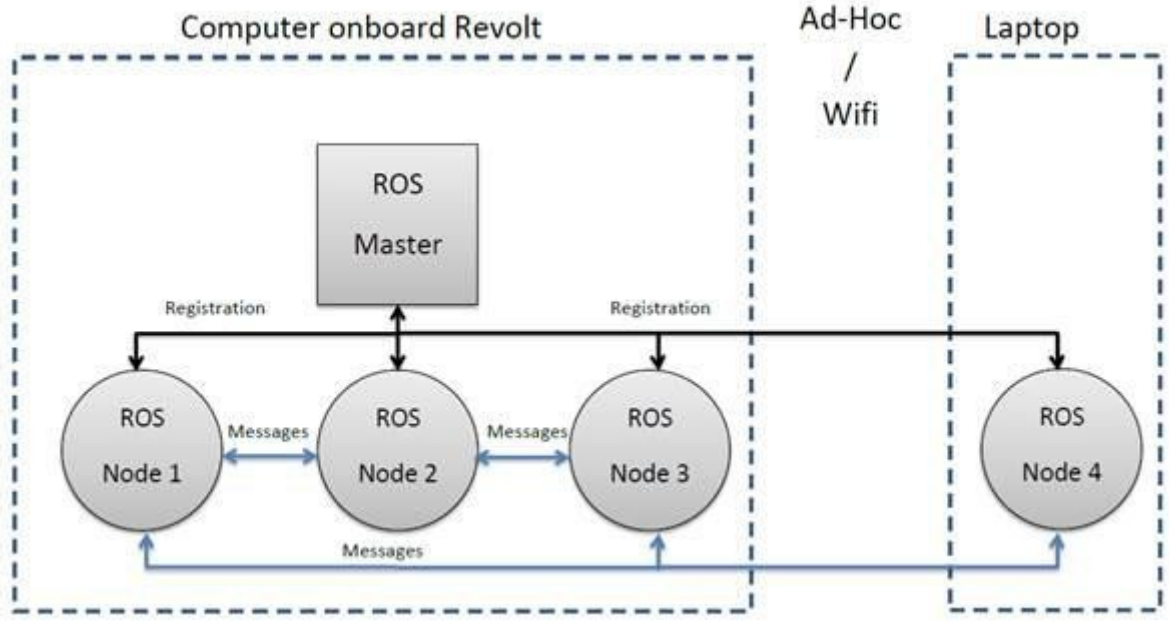



**Figure 3-9 Simple ROS network**

## Basics 101: (assuming ROS is correctly installed)

1. Create a "workspace"

   user@computer:~/$ mkdir –p ~/workspace/src

   user@computer:~/$ cd workspace/src

   user@computer:~/workspace/src/catkin_init_workspace

   user@computer:~/workspace/src/$ cd ..

   user@computer:~/workspace/catkin_make

   The above commands do the following:
   - Make a directory
   - Move into workspace/src
   - Initiate the workspace
   - Step back to workspace/
   - Build

   You will now have a workspace with essentially nothing in it. It is now necessary to source this workspace with:

   user@computer:~/workspace/ source devel/setup.bash

2. Create package

   An example package is now created that also depends on other packages:
   std_msgs: is a package containing many different structs for sending and receiving "topics"
   rospy: makes the package able to understand python language
   roscc: makes the package able to understand C++ language

   These dependencies can be edited in the CMakelists.txt and package.xml file.

3. Now let’s create a node example in python.
   First we create a publisher node that publishes the topic chatter

   File location: workspace/src/example/talker.py

```
#!/usr/bin/env python
# license removed for brevity
import rospy
from std_msgs.msg import String

def talker():
    pub = rospy.Publisher('chatter', String, queue_size=10)
    rospy.init_node('talker', anonymous=True)
    rate = rospy.Rate(10) # 10hz
    while not rospy.is_shutdown():
        hello_str = "hello world %s" % rospy.get_time()
        rospy.loginfo(hello_str)
        pub.publish(hello_str)
        rate.sleep()

if __name__ == '__main__':
    try:
        talker()
    except rospy.ROSInterruptException:
        pass
```

   Then we create a listener node that runs a function every time it picks up data on the topic chatter

   File location: workspace/src/example/listener.py

```
#!/usr/bin/env python
import rospy
from std_msgs.msg import String

def callback(data):
    rospy.loginfo(rospy.get_caller_id() + "I heard %s", data.data)

def listener():

    # In ROS, nodes are uniquely named. If two nodes with the same
    # node are launched, the previous one is kicked off. The
    # anonymous=True flag means that rospy will choose a unique
    # name for our 'listener' node so that multiple listeners can
    # run simultaneously.
    rospy.init_node('listener', anonymous=True)

    rospy.Subscriber("chatter", String, callback)

    # spin() simply keeps python from exiting until this node is stopped
    rospy.spin()


if __name__ == '__main__':
    listener()
```

4. Then we build this workspace by running catkin_make from the bottom of the workspace directory and starting roscore:

user@computer:~/workspace/$catkin_make
user@computer:~/workspace/$ roscore

Open two new terminals, source devel/setup.bash inside your workspace and run both nodes like this:

user@computer:~/workspace/$ rosrun example publisher.py

and
user@computer:~/workspace/$ rosrun example listener.py

5. Open a new terminal and try out these handy commands:

**Table 3-3 Command examples**

| Command | Info |
|---|---|
| rosnode list | Lists all active nodes |
| rostopic list | Lists all initiated topics |
| rostopic echo “topic” | Outputs data on the specified “topic” |
| rostopic pub topic msg_type args example rostopic pub chatter std_msgs/String -- Hello | Manually writes to the specified topic |

6. Launching both nodes from a launch file. First press Ctrl-C in all three windows to shutdown roscore, and both example nodes.

   **Writing a launch file**

   File location: workspace/src/ tutorial.launch

```
<launch>
  <node name=”listener” pkg=”example” type=”listener.py”>
  <node name=”publisher” pkg=”example” type=”publisher.py”>
</launch>
```

   user@computer:~/workspace/src/$ roslaunch tutorial.launch

   This will start roscore and the two nodes in one command.

7. Recording or “bagging”

   To bag, make a new directory anywhere suited to store the bag files, and run the command “rosbag record –a” which will record everything.

   user@computer:~/workspace/bags/$ rosbag record -a

   To play them use:
   Rosbag play “name.bag”

8. Some different graphical user interfaces that are handy for plotting and viewing ROS messages

   rqt_plot
   rqt_console

rqt_bag

## About DNV GL

Driven by our purpose of safeguarding life, property and the environment, DNV GL enables organizations to advance the safety and sustainability of their business. We provide classification and technical assurance along with software and independent expert advisory services to the maritime, oil & gas and energy industries. We also provide certification services to customers across a wide range of industries. Operating in more than 100 countries, our professionals are dedicated to helping our customers make the world safer, smarter and greener.