Robots 101

Introduction to operation and maintenance of civilian units

Samantha Wright, Nanite Systems Consumer Products

January 31, 2016
Companion 8.3 version

Objectives of this lecture

- Introduce the fundamentals of robot operation
- Explain common maintenance procedures
- Provide a complete understanding of the items in the menu system
- Introduce and demystify the command line

Prerequisites

- An interest in robotics
- An NS robot or robot control system (recommended)

Accessing the system

- As easy as clicking it
- Provided you have sufficient access control, this should immediately grant a menu
- Some controllers (SXD, DAX/2, Aide, Aegis) have touch screens as well; beginners should be careful to not click them directly

The local console HUD

- Provides the unit with an at-a-glance view of many major menu items
- Can be acquired by typing `@setup console` (more on commands and chat later)

Main way of controlling what the robot can and cannot do
Every subsystem requires power to run
Moving, flying, teleporting, and talking cost additional power
    Varies depending on speed, distance, and volume
At full power, a unit uses **780 W** when idle
This drops to **40 W** in sleep mode (5.12%)

POWER AND SUBSYSTEM CONTROL

Subsystems: what they do and why

- Main way of controlling what the robot can and cannot do
- Every subsystem requires power to run
- Moving, flying, teleporting, and talking cost additional power
  - Varies depending on speed, distance, and volume
- At full power, a unit uses **780 W** when idle
- This drops to **40 W** in sleep mode (5.12%)

Subsystems menu

- **video**: See (windlight + pixelated view)
- **audio**: Hear (not emotes)
- **network › SMS send** and **receive**: Send and receive IMs
- **motors**: run/fly; walk/touch
- **FTL**: Touch at a distance, sit at a distance, and teleport
- **volume**: speak at all; speak above a whisper; shout
- **mind**: ability to form sentences
- **network › GPS**: see location and map
- **network › identify**: see names

Actual subsystems

- The **SMS** subsystems are really called *transmitter* and *receiver*
- The **volume** subsystems are really called *voice* (ability to chat at all), *preamplifier* (ability to speak normally), and *power-amplifier* (ability to shout)
- The **motor** subsystems are really called *move* (all movement) and *rapid* (flying and running)
- You will see these names if you choose **status** from the main or subsystems menu

Power status

Available power: 65841/69413 kJ

Power source: NS Nanite-Assisted Sonofusion Power Cell 13-7026-T

Power draw rate: 780.00 W

Estimated remaining charge time: 23:26:51

Enabled subsystems: video, audio, receiver, move, teleport, rapid, voice, mind, preamplifier, transmitter, GPS, identify, power-amplifier

Disabled subsystems: none

Batteries and charging

- Battery duration is calibrated around the 780 W rate
- The system cannot run without a battery (except the BaCdrive)
- When attaching the controller or logging in, there is a ~30 sec delay before devices are recognized, including the battery (this is to give SL time to attach everything properly)

Charging

- Wireless chargers (such as those here at NS) are automatic; use `/5start` and `/5stop` to control them
- Display booths and stands are also automatic chargers
- Other chargers (ACS-style) require confirmation to start after sitting
  - NS Induction Chargers use a holographic screen
- Compatibility exists for ACS, Qetesh, and UMD chargers

Running out of power

- The system will automatically shut down at 5% to protect itself
- Speech becomes slower at 20% and 10%
- Some subsystems will automatically shut off below 10% and (if the unit is powered back on) 5% power
  - These will automatically turn back on once you recharge
  - You may need to force them back on to e.g. teleport to a charger

Units talk in object (green) text!
RLV is used to control this
It allows us to do many things with text
    Censor it, filter it, translate it
    Process commands (these start with . ! @)
Chat redirection can be disabled with `!release` and restarted with `/1capture`

THE CORTEX

The cortex: Chat redirection

- Units talk in object (green) text!
- RLV is used to control this
- It allows us to do many things with text
  - Censor it, filter it, translate it
  - Process commands (these start with . ! @)
- Chat redirection can be disabled with `!release` and restarted with `/1capture`

The cortex: Using personas

- Personas overlay different behavior on your unit in 4 ways:
  1. Load folders from #RLV/~NS/<name>
  2. Change tone markers to fit personality
  3. Alter the responses available to the unit when the mind subsystem is disabled
  4. Activate Arabesque scripts on start (more on this later)

The cortex: Mind-less mode

- The unit can type `.info` to get a list of built-in speech phrases the active persona supports:
  - E.g.: .yes, .no, .hi, .bye, .ok, .lol, .cannot, .error, .use me, .use me?, .dance?, .help, .thanks, .explain, .need help?, .mind
  - .yes → “Sure, okay!”
- These can be used even if the mind subsystem is on
- Easy way to sound convincingly in-character and yet also robotically formulaic

The cortex: The vox pipeline

- System of filters that permit alteration of the unit's speech
- Processed in four layers:
  - Semantic: Changes to the underlying meaning of the sentence
  - Linguistic: Changes to the ideal text representation
  - Phonetic: Changes to the simulated 'mouth' for text generation
  - Typographic: Changes to the outputted text

The cortex: The vox pipeline

- Gag mode: keep this off for now (it blocks `/#channel` chat, and may make the unit incapable of talking if the gags are not written correctly)
- Release: can `!release` be used?
- Clear: removes all active filters
- Reset: clear + search for filters again

The cortex: Setting up filters

- Add: turn the filter on
- Add+config: choose settings and turn it on
  - Type ‘help’ in the box for instructions
- Configure: configures an active filter
- Remove: remove the filter

The cortex: Included filters

- **Vocabulary**: word replacement (unprefixed `d_` filename)
- **Polyglot**: translation (`en|fr`)
- **Barnyard**: nonsense (`m o o`)
- **Glitch**: traditional noise filter (`0-40`)
- **Serpentine**: sssssssnakesssssss (`0-100`)
- **Lisp**: thubthituthion (`0-100`)
- **Dropout**: severe noise filter (`0-100`)
- **Microvox**: tiny unicode talk (no parameters)

Controlled through **manage › users** menu
    Only managers and owners may use the **manage** menu
Add nearby users with **manage › users › add**
Add absent users with: `@keychain add <UUID>`
Three levels of access:
    User (0): can access system when public access is removed
    Manager (1): can access **manage** menu
    Owner (2): can always access all system functions
Units may have multiple owners
The **submit** button sets ownership and removes all other owners

ACCESS AND USER MANAGEMENT

Managing users

- Controlled through **manage › users** menu
  - Only managers and owners may use the **manage** menu
- Add nearby users with **manage › users › add**
- Add absent users with: `@keychain add <UUID>`
- Three levels of access:
  - User (0): can access system when public access is removed
  - Manager (1): can access **manage** menu
  - Owner (2): can always access all system functions
- Units may have multiple owners
- The **submit** button sets ownership and removes all other owners

Managing access settings

- Two categories of access
  - Local: when touching the unit directly
  - Remote: when using a remote control
- Access levels: Public, users + group, users only, owner only
  - Group = anyone in the group of the tag you're wearing
- Self access overrides **all** other permissions

PIN locking

- Locking the unit presents a PIN pad to all physical access attempts (except devices like handles)
- The PIN must be entered to unlock the unit
- Auto-lock will re-lock the system if the controller is idle for more than 30 seconds
- No PIN is set by default (just choose ‘unlock’!)

Getting out of sticky situations

- Regret giving up self access? Type `@safeword`
  - Grants guaranteed self-access
- Regret submitting to an owner? Type `@keychain reset`
  - Removes all users
- Both commands tell your owner(s) before they are performed
- Ran out of power? Another unit can jump-start you with `@zap`
  - Or anyone can use a **pulse charger**

Three main categories:

- **dot** (.) commands: preset messages (see previous)
- **cortex** (!) commands: simple chat actions and lighting effects
- **system** (@) commands: generic commands for management and use of the system

Difference between cortex and system commands is mostly historical (bang commands are older)

COMMAND LINE FEATURES

Commands

- Three main categories:
  - **dot** (.) commands: preset messages (see previous)
  - **cortex** (!) commands: simple chat actions and lighting effects
  - **system** (@) commands: generic commands for management and use of the system
- Difference between cortex and system commands is mostly historical (bang commands are older)

Cortex commands

`!light`	Send a message directly to devices (debugging)
`!spark`	Simulates a spark
`!fault`	Simulates a *lot* of sparks
`!broken`	Causes lights to flicker
`!fixed`	Fixes !broken
`!working`	Indicates the unit is busy processing
`!done`	Indicates the unit is done processing

Some other useful commands

`@help`	Built-in manual
`@power`	Current power status and subsystem toggles
`@about`	Users, devices, and identity info at a glance
`@persona`	Change personas
`@keychain`	Manage users
`@scheme`	Change sound settings
`@volume`	Change more sound settings

Remote access

- Anyone with a Remote Management Console can **scan** for and **connect** to nearby units
- Permission is required to actually interface with the unit
- Direct input of system commands is used (no @ sign)
- Use the `relay` command to simulate unit speech

Arabesque scripts

- Make your unit perform a series of actions at a button press
- Notecards containing a list of commands as if entered over remote console
- Not a full programming language (not Turing-complete)
- Full documentation at: support.nanite-systems.com/id=231

- Gestures can also use commands, as if entered locally!

1. Unlock the safety bolts with `@bolts unlock`
2. Edit your controller and go to the **Contents** tab
3. If editing files from **user memory** (personas, arabesque scripts, installed apps, filter dictionaries…) enable ‘edit linked’ and find the **memory card** (description “program”)
4. If installing/editing **sounds** (including vox files) enable ‘edit linked’ and find the **audio processor** (description “speaker”)
5. In Firestorm, use Ctrl+comma and Ctrl+period to cycle linked parts

“Link number: ##” will appear on the edit tool window to tell you where you are

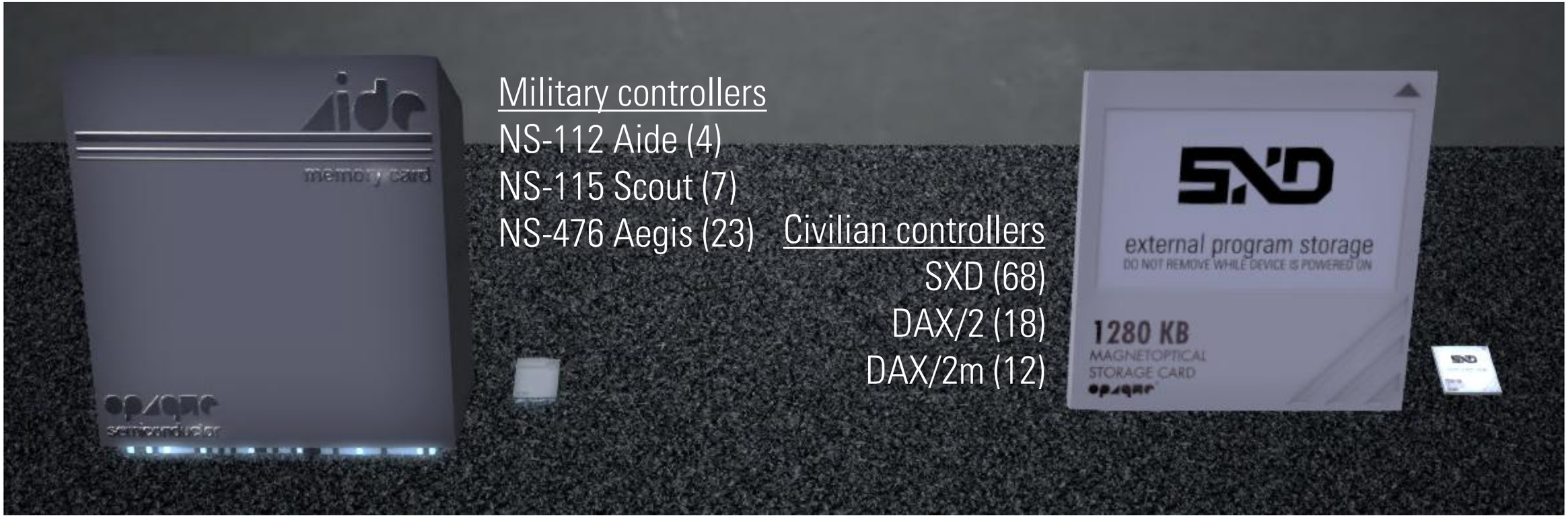
FILE AND PACKAGE MANAGEMENT

Installing data files by hand (self access)

1. Unlock the safety bolts with `@bolts unlock`
2. Edit your controller and go to the **Contents** tab
3. If editing files from **user memory** (personas, arabesque scripts, installed apps, filter dictionaries…) enable ‘edit linked’ and find the **memory card** (description “program”)
4. If installing/editing **sounds** (including vox files) enable ‘edit linked’ and find the **audio processor** (description “speaker”)
5. In Firestorm, use Ctrl+comma and Ctrl+period to cycle linked parts

   “Link number: ##” will appear on the edit tool window to tell you where you are

Locating user memory



BaCdrive: stored in optical disc prim

NS-304 Daybreak: sloped prim above screen

Locating the audio processor

- No fixed shape
- Textured to look like a speaker in some controllers

SXD	**11**	NS-115	**8**
DAX/2	**11**	NS-304	**4**
DAX/2m	**3**	BaCdrive	**3**
NS-112	**6**		
NS-476	**19**		

Applications

- Sometimes you need *more* functionality
- Applications are LSL scripts that anyone can write to add on to the system
- Popular uses include extra controller-side integration for complicated devices, sim policy application—or anything you might conceivably put into a mobile app

Installing applications and updates

- Controlled through the **manage › software** menu
  - Can be done by any manager or owner—not just the unit!
- Requires a proper package on a package server
- You must **connect** to a server and then **install** or **update** from it
- Deleting unwanted software is easily done through **manage › software › remove**

Software tips

- Servers can only be used if you're in the right sim
  - As of this writing, that's Eisa and Dolly Dreams
  - The unit must have rez rights
- Press the **info** button to get a package readme
  - Available from install, update, and remove menus

System updates

- These can be found on the **xcentral:0** server, named `System_x.x.x`
  - Back up your users with `@keychain save` first!
  - Type `@setup console` after the update to get your HUD back!
- The **xpatch:1** server contains old updates for reinstalling
  - Named `Regress_x.x.x`
  - Use these in case of a bad update, or if you want to determine when a bug first appeared for reporting it
  - Going back before 8.3.0 may do *odd* things
    - old installer scripts don't know how to remove newer system modules

Identity settings

`@color`	Set system lighting
`@name`	Set name
`@authority`	Set the name of your company/organization
`@gender`	Set gender (mental, physical, voice)

Settings that can’t be changed so easily: serial number, chat prefix, vendor, and system model

Backing up identity settings

- Edit the **_oem** file in system memory
  - the root prim of your controller; no need to select ‘edit linked’
- Has several built-in commands: serial, name, vendor, prefix, color, vendor, authority, model
  - Do not mess with model! The folly of mankind awaits ye!
- More commands can be added (like a script)
- Manually reload the file with `@reset foundation`

Getting more help

- **Nanite Systems User Group**
  - Ask here first!
- **support.nanite-systems.com**
  - Still mostly empty, but these things take time…
- Pestering rhetOrica with questions
  - Or **support@nanite-systems.com**
- DAX/2 8.0.5 Manual PDF
  - Not *entirely* out-of-date, just increasingly incomplete

Getting more help

- **`@help`**
  - Contains command reference for the system; not yet complete
- **`@commands`**
  - List of all supported commands
- **nanite-systems.com/progress**
  - Current and upcoming system changelog
- Come to the NS main campus (here!)
  - Lots of other people are often around and can answer questions

This has been…

Robots 101

Introduction to operation and maintenance of civilian units

Samantha Wright, Nanite Systems Consumer Products

Thank you!