Big Robot Platform

DESIGN DOC

User Stories & Use Cases

As a developer I can import an existing rig and mesh in order to provide the simulator with a new head shape and structure

AC: new meshes and rigs can be imported at any time, if the motor type and content are equivalent, content will work to new model

AC: able to save a configuration as a project and load it next time

need: UI

As a developer I can set up a motor in order to define a default position, min/max values, motor speed

AC: able to save a configuration as a project and load it next time

AC: assign the properties for each motor using 3d or list method

AC: name each motor

AC: preview how the motor will interact with mesh in 3d simulator

need: UI - functional as developed, brad will do design and art

USER may be robot developer/designer or, perhaps, end user.

1. SETUP
    1. The user is able to choose a number of motors for a particular robot.
    2. The user is able to configure a given number of motors within a robot skeleton, assigning them locations within the head or body of a robot.
    3. The user is able to adjust the type, positioning and range of motion for the motors within a simulator configuration.
    4. The user can import a mesh to use as the robot skin in order to accept a mesh created by a third party.
2. SENSORS
    1. The robot is able to receive input from varying hardware methods, in an extensible way, so that platform users can add additional sensors.
    2. The robot is able to receive visual input via a camera
    3. The robot is able to receive camera input and trigger motors based on image content.
    4. The robot is able to receive audio input via microphone
    5. The robot is able to receive audio input and trigger motors based on audio content, volume, or positioning
3. SPEECH/LIP SYNCH
    1. The platform is able to lip sync spoken text automatically to scripted or dynamic text using volume sampling in order to facilitate simpler mouth motors.
    2. The platform is able to lip synch spoken text to scripted or dynamic text via phonemes to provide a better lip synching experience on more complex robots.
    3. The user is able to enable dynamic speech using alternative user inputs
    4. The user is able to utilize a baked in chatbot service to enable ad-libbing with a robot, in conjunction with a text-to-speech service

5. The user is able to utilize a baked in text-to-speech capability in order to enable ad-libbing in conjunction with a chatbot service.
6. The user is able to plug in differing chatbot and text-to-speech solutions/APIs with a minimal amount of effort.
7. The user is able to use any REST API based text to speech solutions.

4. EDITOR
    1. The user will be able to enter specific motor parameters, in order to provide the simulator with accurate preview on timing and precision of motor commands.
    2. The user can group sets of motors together (via macros) in order to enable group motor control on larger robots.
    3. The user is able to preview scripted and unscripted behaviors within a simulator view.
    4. The user is able to construct scripted behaviors using speech packets (spoken text interspersed with motor control commands) within a timeline view
    5. The user is able to import pre-recorded audio in addition to text-to-speech dynamic speech in order to have the widest range of audio capability, sound effects, etc.
    6. The user is able to designate specific keywords, in order to assign specific behaviors to these words when they are picked up via the robots mic.
5. USABILITY
    1. The user is able to customize a working environment using multiple displays, and customized windows sizes.
    2. The system is able to generate logs for software, hardware and server functions to enable better understanding of system failures and aid in debugging.
6. PUPPETEERING
    1. The user is able to control the robot in real-time to facilitate live product demos or record scripts/macros that can be turned into scripted behavior
    2. The user needs a way to monitor wi-fi/server communications in order to debug connectivity problems
    3. The system will be able to accept motion capture data and will allow a user to edit and make different sequences of movements and expressions.
    4. The system will copy movements and facial expressions from a human actor and apply these to a configured robot in real-time.

Summarized Feature Set

Robot Authoring Platform (RAP)

The RAP is a toolset designed to create consistent, reliable interaction and behavior performances for practical robotic and digital personalities. Using the RAP, an interaction designer can create ‘blocks’ of interaction, each of which can be performed in one (or more) MODES.

Demonstrable Scope & Goals

The features of the RAP toolset, as outlined below, are all contained within the scope of the big Einstein robot and will be demonstrated within the content and context of this product, when the robot is completed. The intention is that further functions and features will be made to this platform as future product requirements are defined and added.

Any new or changed requirements based on emergent hardware requirements on the big robot will have to be considered as they emerge, with extra consideration paid to changes or impact to the RAP platform as well.

RAP will be developed to provide:

- A WSIWYG toolset (the simulator) that allows the creation and integration of robotic functions for prototype, research or consumer use.
- A toolset that is easily learned and usable by engineers and designers, leveraging commonly known scripting languages and user interface conventions.
- A framework that performs reliably provided consistent input/output to and from the target hardware.
- An extensible platform that can incorporate third party technology and libraries for use in advanced robotic functions.
- An organized, clearly documented and commented code base that is easily usable by multiple teams.

Summarized Feature Set

Graphical User Interface (GUI)

RAP allows simple to complex behavior design via the following GUI elements:

- Timelines
- Input triggers
- Macro (and motor level) icons
- Parameter entries
- Dialogue entry (TTS)
- Emotion icons (TTS)

REACTIONS

Designable reactions (output) include:

1. SPOKEN DIALOG – Using TTS (with emotion control), expansive dialog can be designed via interaction trees. Conversations can be navigated via scripted, autonomous, puppeteer or ‘game logic’ modes.
2. FACIAL EXPRESSION – Facial motor reactions can be triggered via scripted, autonomous, puppeteer or ‘game logic’ modes.

3. EXTREMITY CONTROL – Extremity motor reactions can be triggered via scripted, autonomous, puppeteer or ‘game logic’ modes.
4. REACTION TOGGLES – Basic autonomous movements (like blinks and eye-line following) - as well as any unique reactions that may be specific to robotic personality - can be toggled on/off for any ‘block’ of interaction, by the interaction designer.

INPUTS

The RAP designer can make use of the following input types when designing a ‘block’ of interaction:

- Spoken (pre-scripted) Keywords – STT Audio capture*
- Spoken (pre-scripted) Phrases – STT Audio capture*
- Spoken chatbot (unscripted) queries**
- Non-verbal sound recognition – *high frequency input? Recognizable sounds like sneeze, etc?*
- Head tracking
- Facial recognition
- Emotion tracking/recognition
- Other object tracking/recognition - *Hand and finger. Displayed shape/symbol?*
- Triggers from Local or Cloud Networks

Requires integration with third party text-to-speech libraries.
****Requires integration of third party chatbot technology.***
*****Requires hardware integration via Intel Realsense cameras.***

MODES

Based on the type of input that controls how ‘blocks’ of interaction are being navigated, a robotic personality can be said to be in an ‘Interaction MODE.’ It's convenient to classify at least four different MODES of interaction:

- Scripted – A robot is in scripted mode if it is running a linear sequence of interactions OR navigating interaction blocks via pre-scripted speech input.
- Autonomous – A robot is in autonomous mode if it is performing interactions based on non-proactive input (Non-verbal sound, head tracking, face capture,..) or unscripted speech queries.
- Puppeteer – A robot is in puppeteer mode when KEY interactions are prompted by explicit operator triggers (local/cloud network signal, high freq sound,…)
- Game Logic – A robot can be controlled by custom coded game logic. This mode can create a more complex interaction utilizing any input type.

Simulator

The RAP incorporates a robot *simulator* into the authoring process, so that interaction design can continue in parallel with the development of practical robotics. *At each significant iteration of*

the motor systems, the simulator is updated to match all motor output and (potentially) additional control elements added to the RAP.

Most significantly, the simulator provides the designer with the ability to:

1. View accurate output of any 'block' of interaction run in the RAP.
2. Create and perfect motor Macros (single facial expressions or an unbroken range of expressions) that can then be exported into the RAP as unique Macro icons.

Reverse Deliverables

Since the RAP platform is highly dependent on hardware development, certain hardware deliverables will be required during development. The timing and lockdown of such deliverables will be negotiated during the course of M1. A few known examples:

- Sample motors with reasonably consistent communications protocols (currently working off the motors used on the little Einstein robot platform).
- A USB board with a C/C++ library to actually interface with the board.

Design

What Does it Do?

In essence, the platform does three general types of things:

- Configure a Robot
- Create Content for a robot
- Operate a robot

Configuration

The platform supports various numbers of motors and sensors, but the properties and locations of these must be configured in the software, for any particular robot to function.

- Motor settings
- Sensor settings

Content Creation

This software provides a user interface for development of robot content as well as an integrated Simulator for playback. A motor configuration module defines capabilities and constraints for each robot.

The following is a list of some of the types of interactive content that can be created:

- Dialog Trees
- Sensor Logic
- Chatbot Logic
- Gameplay Logic
- Motor Animation

Operation

The robot will be able to run in one of several different modes to satisfy customer needs ranging from fully automated to fully scripted. In these modes, the robot will be able to use various different kinds of content, such as live motion-capture, or hand-made animations.

- Scripted Interaction mode
- Live Puppeteer Mode
- Autonomous Chat Mode

Terminology

Animation

An animation is a combination of speech and motor movement in the form of either recorded motion-capture or handcrafted text and motor commands.
Animations are meant to convey the robot talking, while also subtly moving in a believable way.

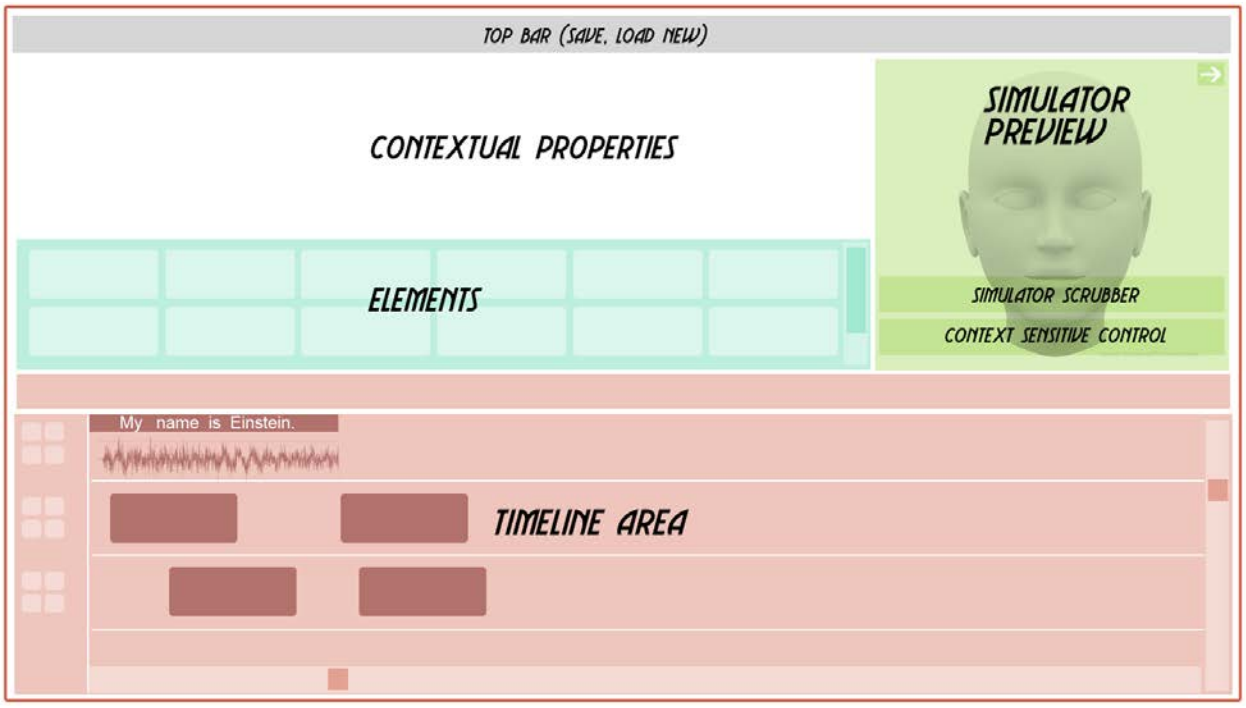
Intent

Intents are nodes animation data that forms branching paths of conversation.
Each intent has the animation, and optional sensor triggers that link the conversation to other intents.

Behavior

A behavior is custom user-written code that can perform various motor or speech commands. Behaviors can be written to take into account all kinds of data such as sensors or time, to influence how and when they perform an action.
The code is written in unity c# script, that is executed by a version of unity that runs on android powered robots.

Tool Interface



The robot authoring tool will be a desktop application, have an easily editable timeline, drag & drop icons for macros and other actions, along with a seamless connection to Dragon.AI

Panels

All of the content creation or robotic operation modules exist inside of different dock-able panels.

The panels can be used free-floating, or each expanded to fill a full screen if desired... but they also dock with one another to more comfortably fill the main window.

Robot Operations Panel

The above image displays the common controls for all 3 modes, but none of the unique ones. The further controls will extend the panel taller

The Robot operation panel is where the user can operate a robot, or monitor the operation of a robot.

The panel is built with the intention of being open alongside content creation panels. This allows users under some circumstances to append the content of the robot, based on live situations the robot is currently participating in.

There are multiple modes that the robot can operate in. depending on what mode the user selects, the panel will have different controls.

Common Controls

Robot Animation Viewer

Like the simulator panel, the robot animation viewer shows an animated version of the robot, but in the current state the robot is in. This distinction is important for operating and observing the robot while also creating new content in other panels.

Camera View

A portion of the panel is devoted to displaying a live camera feed of what the robot sees. This assists in not only live interaction, but also debugging.
If multiple cameras are available, the user will have the ability to select which one is being viewed. There is also an option for cycling between several cameras.

Listening Properties

Users can adjust which microphones are currently listening, and at what volume they are playing on the user's computer.

Global behaviors

This is a list that can be expanded or shrunk, of behaviors that are always active (such as blinking).
Only custom coded behaviors, or special default behaviors are allowed as choices here.
Regular text and animation behaviors are not accepted.
Global behaviors can be built to override other behaviors, puppeteering, or chatbot actions.

Operation Modes

The operation modes can be changed at will, but users will be warned that it may interrupt the robot.

Default Mode

The default mode is a placeholder that only tells a user to choose the mode they would like to operate in. It exists only to tell players about the mode choices. It has no controls (including the common controls listed above).
When a user chooses a different mode, they are not able to return to the default mode.

Scripted Interaction Mode

Scripted animations or recorded motion and voice capture data are played, triggered by sensors, cameras, audio-cues, or programming logic. In this way, many complex interactions can be linked together in a sophisticated way.

A user is not required to be present to operate this mode, but they can watch, debug, and even edit the robot's content and logic, while the robot is in operation.

Enable / Disable

This control allows the user to enable or disable the scripted interaction mode.

With it disabled, the robot will not listen for triggers, and will not act out any behaviors.
As soon as it is enabled though, the robot will begin executing whatever is listed as the starting behavior.

Starting Intent

The starting intent is a choice of where the scripted interactions should start.
When the interaction mode is enabled, the robot will begin executing this intent's animation, and then listening to it's triggers.

Current Intent

The current intent is shown, along with the triggers for the next intent, and the intents they are connect to.
For debug purposes, when a trigger is heard, it flashes, before the interface switches to show that new intent as the current.

Puppeteer Mode

Puppeteer mode allows the user to directly control the voice and movement of the robot, based on their own voice and movement.
This mode requires the user have precise information about what the robot is seeing and hearing, to foster a more intimate interaction with people engaging the robot.

Quick Animations

The puppeteer has access to a list of shortcut animations to use in place of animation or voice.

Puppeteer settings

The puppeteer has the option of turning off the puppeteer microphone, in the event that they wish to use the audio from a quick-animation.
The puppeteer also has the option of turning off the puppeteer motion-control, to allow the animations from a quick-animation, to override the puppeteer movement.

Autonomous Chat Mode

Similar to Puppeteer Mode, but the robot is controlled by an AI (Chatbot).
The user also has the ability to specify keywords that lead to particular animations, functioning as a more dynamic version of Scripted Interaction Mode.

Special Animations

A list of word-groups, and the animations that they trigger.
Use of special animations along with chatbot, allows the robot to fill in the gaps in content, while still having some pre-defined structure.

Chat State

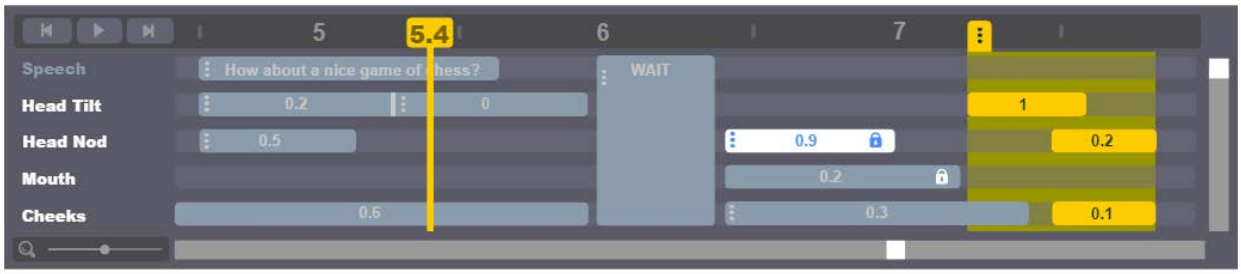
The robot shows the current state of chat in what was last heard by the user, and the response that the robot is preparing to use.

Animation Editor Panels

The animation editor is a collection of 3 different panels that are created together. They can be docked and moved separately, but they all close and open together.

Keeping these 3 related panels separate, is useful for better utilizing screen real-estate... but at the same time, none are especially useful without the others, so they must still all be present.

Animation Timeline



In my mind, moving to a timeline is the absolute most important step that could be made to increase speed and ease of use. Specifically, this vastly improves the ability to test and tune the look of facial animations, which is a vital step to bringing out that lifelike feel to any animation. This timeline does prevent conflicting commands to the same motor, but those produce strange results anyway, and should be bad practice.

Pre-generation

An important concept of this version of the simulator is that after any edit, the simulation is generated in advance, rather than in real time on preview. This allows for pausing on particular frames, and a representation of the length of time each command exists for (especially speech).

Saving

Saving happens by looking at the relative positions of each command, and converting those into the equivalent codes. Each command at the same time position is done in order from top to bottom… and wait commands are used to delay until the next listed command.

Motor Lists

The list of rows corresponds to the list of specified motors, and one speech channel. If the number of rows exceeds the vertical size of the timeline channel, then a vertical scroll bar will allow the user to scroll through the multiple rows.

Commands

Commands are created and moved around on the row for their same type. They can’t overlap with other commands of the same type. Dragging the front of a command moves it, and dragging the back of a command lengthens or shortens it.

By default, when an action is moved up, all the actions that start after, are also moved up. Users can avoid this if they wish, by “locking” a command at a certain position either permanently or temporarily. Eventually, a series of commands that works well can be turned into a macro and locked together more conveniently, without worry.

Each command will have additional properties (beyond duration and start time), that can be modified in a separate properties window (that is not mocked up here).

Motors

Most commands are motor commands, where the state a particular motor is asked to change over a specified period of time.
For more information on motors, see the Motor Details section.

Speech Commands

Speech commands are special in that the duration is fixed, and cannot be modified. The duration of a speech command is computed in advance, based on the text that has been entered.
Changes to a text string that affect its duration, only affect other text strings (either moving them back or forward). If a speech command grows larger, it always pushes later speech commands back. If a speech command shrinks smaller, it always pulls later speech commands forward.

Wait Commands

There are 3 wait commands that users need to worry about: wait for motors, wait for speech, and wait for all. These are represented by a vertical bar in the image above, but can have a width the length of any duration the player sets. Regular wait commands are added to the code automatically during saving, and don’t need to be specified by the user (the user can instead just drag the commands further apart to increase the time.

Adding & Removing Commands

New commands can be added by clicking one any row in the timeline. A new minimum length command will appear where the user clicked, and it will be selected automatically. In the properties panel, the user will be able to set up its initial details.
Existing commands can be removed in two different ways:

- Clicking the delete button in the properties panel
- Dragging the command vertically (off of the timeline panel) and releasing.

Moving Commands

Commands can be moved by dragging from the front of the bar. Commands can’t be moved further left than the back of the next previous command. Similarly, a command can’t move further right than the start of the next following command (without snapping to it).

Resizing Commands

Commands can be resized by dragging the rear end of the command.

If resizing a command larger makes it collide with another command, the commands snap together. Before the user releases the button, they can resize it smaller and the snap will not take place (the other command will stop at its original location).

Resizing a command with another snapped behind it, will push the other command backward to accommodate the size of the resized command.

Command Snapping

When two commands from the same motor (or speech) are right up against one another, they “snap” onto each other like lego pieces. When snapped, the back command cannot move further forward than the front command, and moving it back will un-snap it. When moving the front command, it can move both forward and back, and the back command will follow. Resizing the front command will move the other command back or forward.

More than 2 commands can be snapped together in this way. There is no limit to the number that can be joined, but only the first command can be moved.

Selecting Commands

Clicking on a command selects it. Properties for a selected command show up in the properties window. Any adjustments in the properties window will show up in the preview, and take focus away from the timeline panel. Even though the timeline panel loses focus, the selected command remains displayed and selected.

Macro commands can be selected individually, but all the fields in the properties panel are grayed out and not interactive.

Locking Commands

A selected command can be locked, to prevent it from being moved due to snapping. Locked commands cannot be snapped to. They will block other commands from being resized larger, or moving backwards further than where the locked command resides. Once a command has been locked, the lock icon will remain displayed even when it is no longer selected. Locked commands can still be manually moved or resized.

Locking and unlocking is not available for macro commands, as they are already always locked.

Macros

Macros are predefined lists of commands that can be imported into any intent. They might be a gesture such as head shaking or head nodding, or they might be something as simple as just a particular static facial expression. These are extremely useful for repeating a particular common animation, rather than re-animating it each time.

Macros are shown on the timeline with a yellow background. All the component commands of a macro are shown in their entirety. The yellow background extends from the front of the very first

command, and to the back of the very last command. The macros can be moved as a group, just as a command can… but they are rigidly locked to one another.

Other commands can extend into the yellow background area, but the individual macro commands collide with other commands as normal (preventing movement from the rest of the macro commands). Other commands can also snap to macro commands (from either side). There is a “draggable” tab that extends above the macro, that users can use to drag the macro forward and back.

Player Marker

The yellow line in the center of the mock, represents the current position of the preview relative to the list of commands. The preview can play or even be paused. Because the simulation preview is pre-generated, the yellow line can be dragged around the simulation to look at individual point in time in the animation.

At the top of the player marker is the current time in the intent, that is being displayed. If the timeline panel is not in control focus, then the timer portion is not displayed.

The player marker moves automatically with playback, but the user can also drag it manually, or click on a spot on the time ruler at the top.

Player Controls

In the upper left of the timeline panel are the player controls. The player can play/pause the playback, or also jump to beginning, or jump to the end.

Zoom controls

In the bottom left corner of the panel, is the zoom control. Sliding the bar left will give a wider perspective of the timeline (scaling the timeline view to be shorter), while sliding it further to the right will give a more granular perspective (scaling the timeline to be longer). The default view is halfway in-between both perspectives, but the last setting should be saved for each project (or at least for the editor as a whole if each project is not possible).

The scrollbar reflects the changing size of the timeline, after any changes to the zoom level. Users will be able to scroll left or right through the timeline, if it doesn’t all fit within the panel.

Command Properties

The properties panel is a panel that presents editable property controls for interacting with timeline commands.

Because the properties vary wildly between the various types of commands, the controls inside of the properties panel change depending on the item that is selected.

Any changes to the properties immediately affects the timeline panel, and should therefore be represented in the viewer as well.

Motor Command

Motor commands have the following controls:

- Motor end position
- Duration
- Position lock toggle

Speech Command

Speech commands have the following controls:

- Text to be spoken
- Position lock toggle

Wait Command

Wait commands have the following controls:

- Duration
- Position lock toggle

Animation Viewer

The animation viewer allows the user to preview the current animation. It is a window that displays a simulated version of the current point in the animation (either paused or animating).

Speech

Ideally the text-to-speech (TTS) module used by the simulator should be the same as that in the robot. When this is not possible, as for Little Einstein, speech packet motor commands are loosely coupled to speech to be suitable for use with any TTS. Flexibility in timing of expressions with speech also simplifies localization to different languages.

Lip Sync

On robots with limited mouth control, volume sampling is adequate for lip synching. On big robots with enough motors to show phonemes, improvement is possible. To accomplish this, we need:

- Very closely timed mouth motor control with audio stream
- Phoneme identification software

Viewer Controls

The viewer itself has a few controls:

- Toggle play/pause
- Viewer volume
- Restart

- Motor setup button

Dialog Tree Panels

The dialog tree panel allows the player to construct and link intents, with a visual interface.

Intents Editor

The intents editor lets users create, edit , and link intents. Each intent is a node in a branching conversation. The primary component of an intent, is the animation that typically includes dialog that the robot speaks aloud. But an intent typically has branching links to other intents as well.

Animation

The animation in any intent can either be a handcrafted animation, or motion-captured.

Linking

Each intent can link itself to a number of other intents, to make a web of conversational topics or animations that the robot can seamlessly travel through.
This requires a number of intents to link to, and each linked intent requires a trigger that gates progress to that intent.

Triggers

Triggers are an intelligent way of determining which branch of the conversation should be travelled down.
There are a variety of choices of what can be used as a trigger (based on the sensors available on the current robot):

- Vocabulary triggers (a menu for managing each animation and associating them with different trigger words)
- Sensor triggers
- Camera triggers
- Microphone triggers (different from vocabulary?)

See Sensor Setup Window for more information on what is available as a trigger.

Linked intents

Each link, requires an existing saved intent, to be used as the next point to visit in the conversation. This intent will be complete with its own animation, and optionally links to other future intents.

No Valid Response

For each intent, there is an option to link one or more intents as a special behavior, for use only when no proper response is heard. These special links are handed in a separate part of the UI from the other intent links.

Dialog Tree

The Dialog tree view is a scrolling window, showing a web of intents and how they link to each other.
The player can scroll the list, to see how all the different intents link together.
Intents that are not linked to anything are still displayed, floating by themselves.

Controls

- Toggle showing local links only vs global links
- horizontal and vertical scroll bars (can also be dragged)

Motion Capture Panels

Motion Capture Timeline

Soundwaves will be shown where speech was recorded, and horizontal strips below for motor commands.

- record button
- toggle audio recording
- Toggle play/pause
- Restart
- select all
- select all before marker
- select all after marker

Timeline Data

The motion capture timeline is broken into two parts, motors and vocal. They are represented as two separate rows of data, and can either be interacted with at the same time, or separately.

The user can click and drag to select and highlight areas on either or both rows of the timeline. When selected, various interactions are available to the user, on how to operate on that data.

- copy
- cut
- paste
- move
- delete
- select

Player Marker

The yellow line in the mock, represents the position of the preview, relative to the recorded voice and motion data.
The player marker is used to show what the replay viewer is currently showing. It moves when the replay plays, and it stops when the replay pauses.
The user can either drag the marker back and forth, or they can click on spot on the time ruler at the top.

Recording

when the record button is pressed, the marker moves forward, and all motion and speech data is overwritten from that point forward. The duration of the recording is extended if necessary.

Replay viewer

The replay viewer allows the user to see motion data as its recorded, or review it afterward. It shows a simulated view of the current point in the animation (either paused or animating).

Viewer Controls

The viewer itself has a few controls:

- Viewer volume
- Motor setup button

Panel Layout Modes

Whenever a user starts the tool, they have an option of choosing various tool modes, that have different starting panel layouts:

	Animation Editor **Create Animation**	**Dialog** **Create Dialog**	**Robot Operation** **Performance**
Content Creation Mode	yes	yes	yes

Operation Only Mode	no	no	yes

Users will still be able to add or remove panels from any of these starting modes.

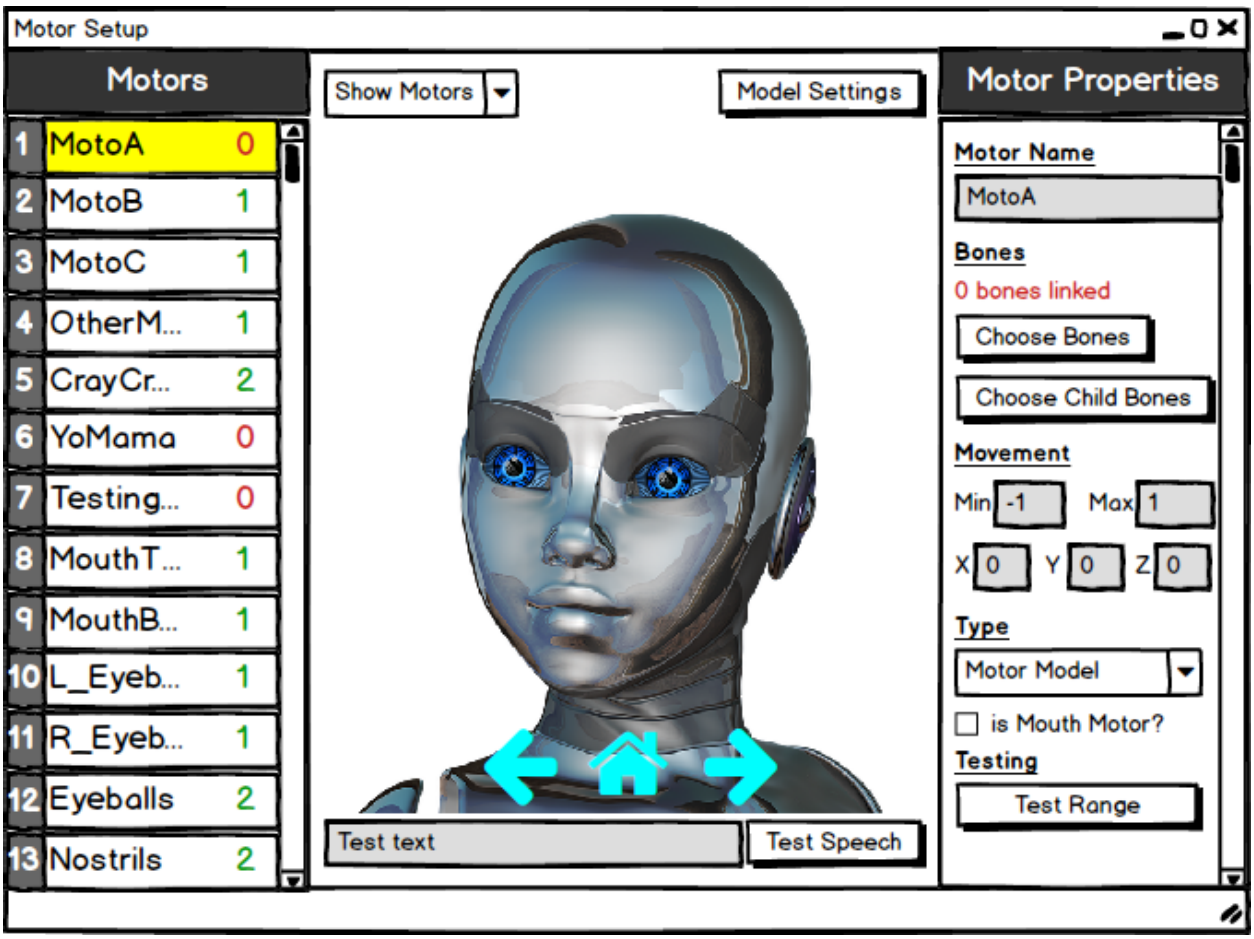
Users will also be able to save their current panel configuration, which will then become a new selectable choice on startup.

The user will be able to change to a different panel layout mode, while the tool is still running.

Windows

Although much of the functionality of the tool is in the modular panels, there is also some functionality that doesn't need to be ever present while operating or creating content for the robot. The remaining functionality is handled by pop-up full-screen windows, that can be closed to return to the panel view.

Motor Setup Window



UI Components

Motor List

The motor list on the left hand side, shows the chosen number of motors, in order. The user can click on any of these, to select that motor, and make changes to it.

Robot Viewer

The viewer portion of the window, shows a 3d model, simulating the appearance of the robot. The user will be able to view the robot at different angles, and make changes to the angle and position of the currently selected motor.

Viewer Controls

The viewer controls allow the user to scroll around the model, and return to facing from the front. THe user is also able to drag the mouse in this window, to change the angle and zoom level as well.

Display Mode Toggle

This window can display just the model, just the bones, just the motors, or some hybrid combination. These different modes let the user specify how they would most like to visualize all of this motor data.

Model Settings

The model settings button takes users to the model settings window, where they can change the number of motors the robot uses, as well as change the 3d model the simulator uses.

Test Speech

The test speech option, allows the user to type text, and see how the robot moth motors handle it. Because multiple motors can be assigned mouth responsibilities, this functionality is separate from the selected motor properties.

Motor Properties

The motor properties portion of the window displays related properties to the currently selected motor.

Motor Name

The motor name, changes the label this motor goes by, wherever else it is referenced in the tool.

Bones

The bones options, allow users to select where the motor is attached to on the model, and what geometry it affects (for purpose of simulating in the tool).

Movement

Movement allows the user to specify the range of motion, and axis that the motor operates on.

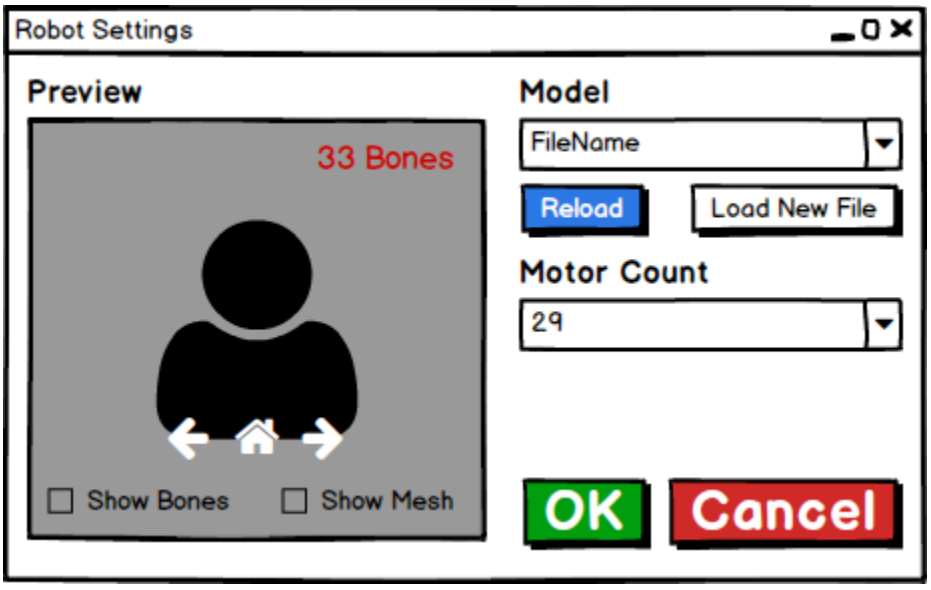
Type

Some motors may have different performance or operating parameters that are important to list for the robot to work properly. This field allows for that future flexibility.

Testing

The testing fields, allow the user to see the range of motion, of that motor, demoed with the current settings, in the simulated robot view.

Model Settings Window



Preview Panel

The preview panel shows an overview of the currently selected model. The default model is a pre-supplied generic version, that will be used until the user specifies an alternative.

Viewing Controls

Users can left-click and drag on the preview panel, to rotate the model about the y or x axes.

Bones & Mesh Toggles

At the bottom of the panel are toggles for Bones and Mesh. By default, both are on.

- Mesh displays the visual skin of the 3d model
- Bones displays red points that mark locations for motors to be assigned or linked to

When both are on, the bones show through the skin, so the user can get an idea of where the motors can connect, in relation to the face.

If both are toggled off, the window appears empty, except for the toggles and the bone readout.

Bone Count

The bone count appears above (and in front of) the model preview. It shows the number of valid bones that were detected when the model was loaded.

Options

Model

This lists the filename of the current model. It is also a pulldown that can let the user select generic models that are supplied with the software. The pulldown will also display the last 3 custom models that were selected.

The reload button simply reloads the currently selected model, so it accurately reflects any new changes in the file.

The load new file button brings up a load file window, that lets the player choose a new model to load. Error checking should be performed on this model before it replaces the model that is currently selected.

Loading a model or choosing a previous model does not affect the robot settings until the user presses OK. Selecting a new model is only a temporary preview of an option the user could choose to commit to.

Motor Count

There are several pre-defined numbers of motors that a robot can conform to. This pull-down lets users select from the valid choices of the number of motors the robot is expected to use.

Changing the number of motors does not delete the existing names of any motors:

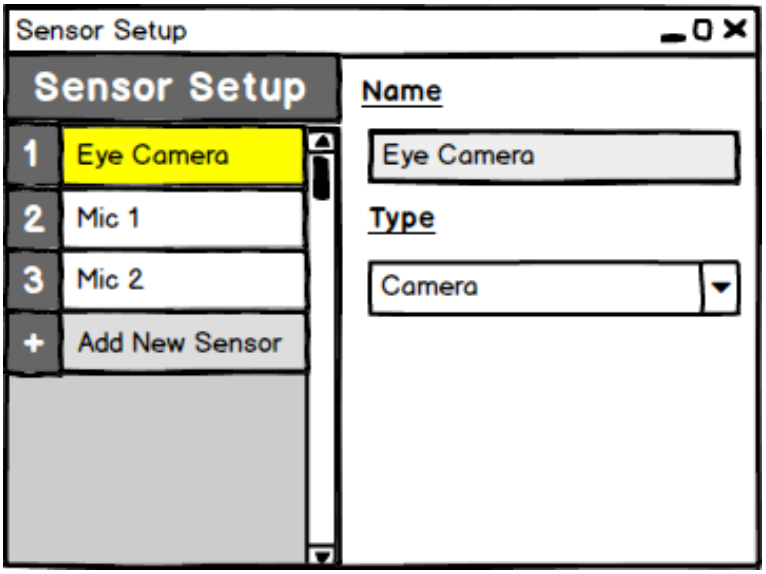
- If a user selects less motors than before, the still remaining motors will retain their names and settings

- The data for the old motors should still be saved, so if the number of motors is increased again, the previous settings return (even if this data is unused by the final robot)

OK & Cancel

The OK & Cancel buttons are standard behavior.
If cancel is pressed, the changes are ignored and the robot reverts to its previous settings.
If OK is pressed, the changes are committed.

Sensor Setup Window



The sensors setup window is an ordered list of slots, that can optionally hold a sensor (or none).
When any slot is selected, properties including sensor type are shown.

Toolbar

The toolbar at the top of the window, allows access to some regular actions:

Layouts	A list of the different default layout options, user created layout options, and the ability to save or delete the current layout.
Panels	A list of the different window panels that can be opened. Clicking on a panel name that is closed, will open it at minimum size. Clicking on a panel name that is open, will close that panel.
Configuration	A list of the different configuration windows that can be opened. Clicking on any will open it.

Save / Load Data

Each panel or window is responsible for its own data saving and loading. This is why a save and load option is not necessary on the toolbar.
Occasionally, the user can attempt to close a window or panel that has data

First Time Startup

The first time a user runs the tool, they are taken to the robot configuration window, to set up the robot. a minimum number of motors must be set up initially before the user can leave the setup window and use the rest of the tool. The user is not prevented from coming back to the settings window later, and altering the configuration.
In this first time through, the close window button is replaced by a "next" button, and is grayed out until a minimum number of motors or sensors is set up.

Technology

Technology Overview

Sensors

Intel® RealSense Camera SR300
Microphone(s)

Leveraged Technology

Authoring Tool - JavaScript, HTML5
Simulator - Unity, C#, WebGL
Motor Driver - C/C++
IOT Cloud Services - Dragon.ai
Text to Speech - Watson or other, pending evaluation

Expected Integrations

ROS

Robot Operating System. Two integration points
1. The Motor Driver will support receiving ROS messages for robot control
2. The Authoring Tool will support a configuration layer to output ROS messages to a third party robot.

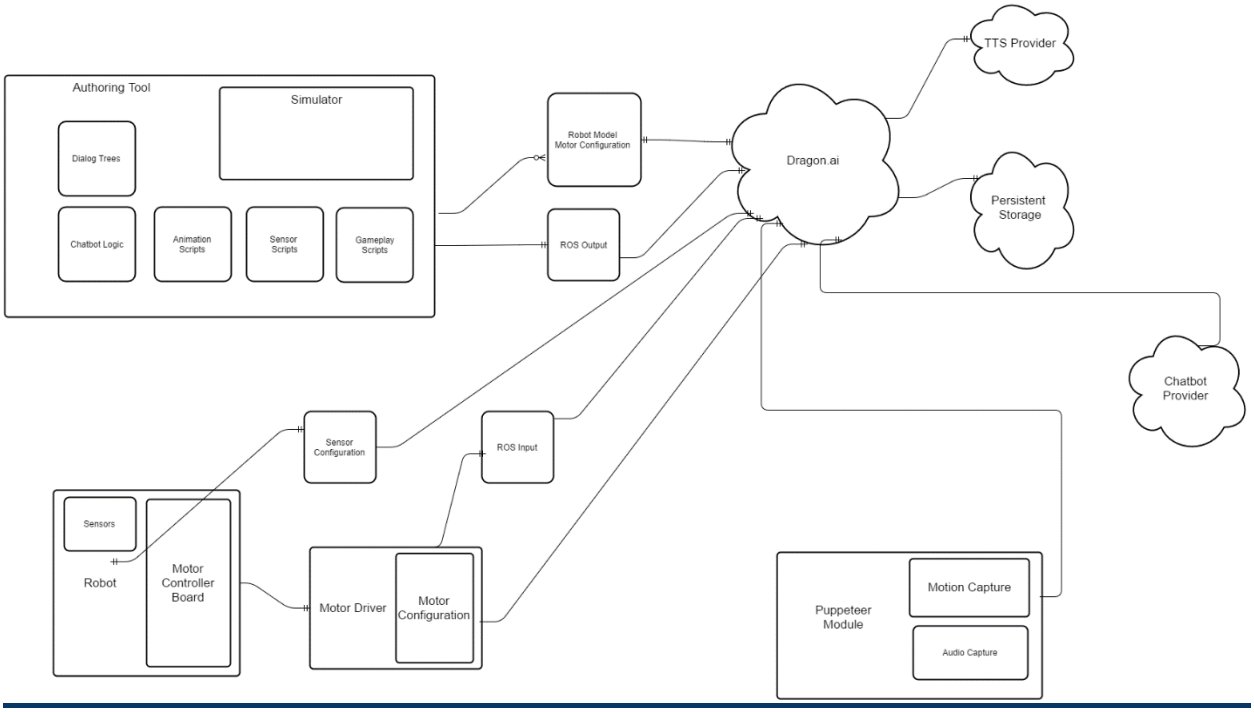
Chatbot

A chatbot service, paired with a text-to-speech integration allows unscripted interaction with a robot. A pres-designated service (such as Watson) will be built into the first implementation, with allowances for further integrations as the platform is revised. The goal here is to keep the platform as open as possible so developers may select the best service for their use cases.

Text-to-Speech

A baseline text-to-speech engine will be integrated as part of the Einstein robot package, see Appendix II for basic metrics on different text-to-speech options.
No matter what integration is selected as part of the first iteration of the RAP platform, drive the system will be designed in such a way to make alternative text-to-speech integrations possible, at the discretion of the platform user.

Deployment



Motion Capture Details

Motion capture makes puppeteering possible, and if sufficiently accurate could also be used to help compose scripted interactions.

Puppeteering Overview

Puppeteering will consist of two modes: Playback and Realtime

Playback state:

In Playback Mode, the system will be able to store the movements and face expressions from a human actor to then allow an operator to edit and make different sequences of movements and expressions.

Those sequences can be transmitted later to the robot or simulator as a scripted sequence.

Realtime Mode:

In Realtime Mode, the system will be able to copy the movements and face expressions from a human actor and translate-transmit this information to the robot or a simulator in

real time. The system also will work as an interface between the data from the motion capture systems and the robot.

Hardware and Software.
To capture, processing and broadcasting to the robot, a PC computer will be required with a suitable hardware configuration.

Requirements

The puppeteering software performs the following functions:

- Reads incoming data from the full body and face expression mocap system.
- Process the information: The software is an interface between the mocap data and how it is translated to control robot motors.
- Broadcasting: The software sends the information to the robot or simulator over a network.
- Filtering: The software analyzes the incoming data to validate potentially wrong or illegal movements from the mocap systems.
- Storage and manipulation: The software allows a user to store and edit the incoming input data. The user will be able to generate special custom sequences to be played to the robot at any time.

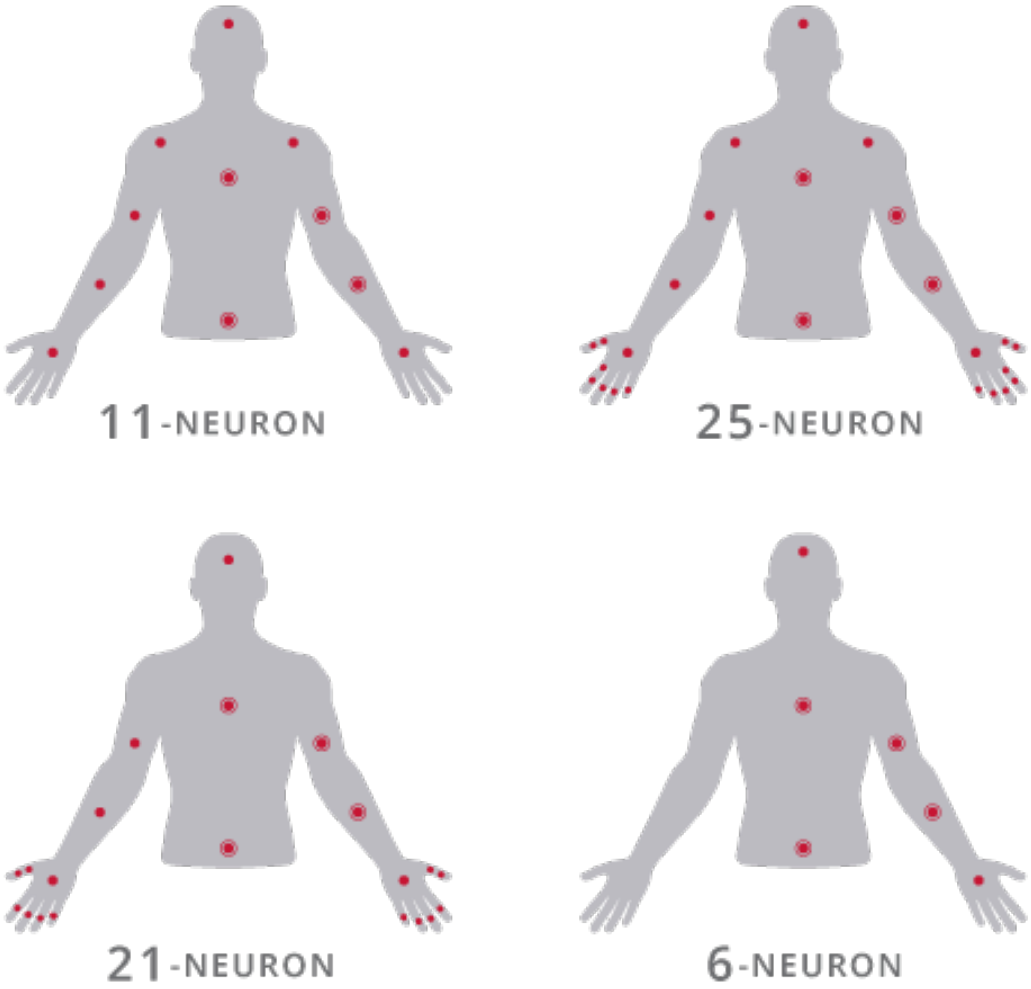
Motion Capture Systems

Two major components are required to cover data for a robot with articulated limbs and facial rig: a full body motion capture system as well as a specific facial expression capture system.
Note: The list will be updated at the time of each capture system be reviewed
Full body motion capture systems: Perception Neuron

PERCEPTION NEURON is the first tool of its kind to deliver SMALL, ADAPTIVE, VERSATILE and AFFORDABLE motion capture technology. The modular system is based on the NEURON, an IMU (Inertial Measurement Unit) composed of a 3-axis GYROSCOPE, 3-axis ACCELEROMETER and 3-axis MAGNETOMETER. The strength of the system lies in Perception Neuron's proprietary Embedded Data Fusion, Human Body Dynamics and Physical Engine algorithms which deliver smooth and true motion with minimal latency.
The PERCEPTION NEURON 9-Axis sensor units output data at 60fps or 120fps*. The data stream is channeled to the HUB where it can then be transferred to a computer in three different ways: (1) via WIFI, (2) via USB or (3) recorded onboard using the built-in micro-SD slot.
PERCEPTION NEURON then connects to the software AXIS Neuron or AXIS Neuron PRO for calibration and management of the system, as well as recording and exporting data files for manipulation in most professional 3D, previz and game development tools.
PERCEPTION NEURON was specially conceived as a professional tool for video game developers, film makers, visual effects professionals, biomechanics researchers, sports and medical analysts, and virtual reality enthusiasts to finally have a flexible and affordable platform to experiment with and push the limits of motion capture.
*60fps: 19 - 32 Neurons, 120fps: 18 Neurons or less

Advantages

- Full body and hands and fingers motion capture.
- Do not need a room setup environment.
- Setup time, between 10’ and 15’
- Cost $1499
- Adaptive: operates with up to 32 individual neuron sensors that can be placed on the body using body/finger straps. They can be applied in different configurations — from hand motion with as few as 3 sensors, to full body with 17, and all the way to complete detailed body and hands with up to 32 including one prop. This level of adaptability is a freedom no other motion capture system offers out-of-the-box.



- Small: The system is composed of interchangeable neuron sensors (inertial trackers) connected to a Hub. The Hub connects wirelessly to a computer through WIFI or can be wired directly through a USB connection. The system is powered by any external USB power pack. The whole system weighs less than a 300 grams (excluding battery).

- Unity sdk.

Cons:

- Between 98% of the time the information is reliable.
- Fragile, active sensors on the motion capture suit.

System Cost:

	32 Neuron Alum Edition	32 Neuron Alum Academic	18 Neuron Alum Academic
Restrictions	None	Must be academic verified*	Must be academic verified*
Neuron	32	32	18
Neuron Build	Aluminum	Aluminum	Aluminum
Hub	1	1	1
Anti-MAG Container	2	2	1
Body Strap	Full Body	Full Body	Full Body
Finger Strap	2 (Left + Right)	2 (Left + Right)	None
Base Gloves	6 (2xS, 2xM, 2xL)	6 (2xS, 2xM, 2xL)	None
Premium Studio Organizer Bag	Yes	No	No
	$1499	$1199	$799

Manufacture Information

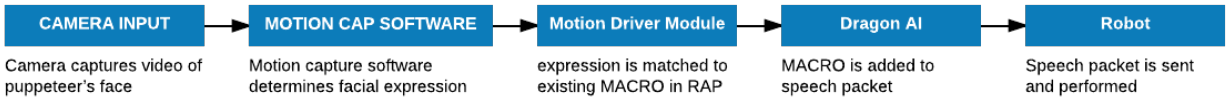
www.neuronmocap.com

5th Floor, Bldg A
28 Xinjiekouwai Blvd.
Beijing 100088, China

- +86-10-82055391
- contact@neuronmocap.com

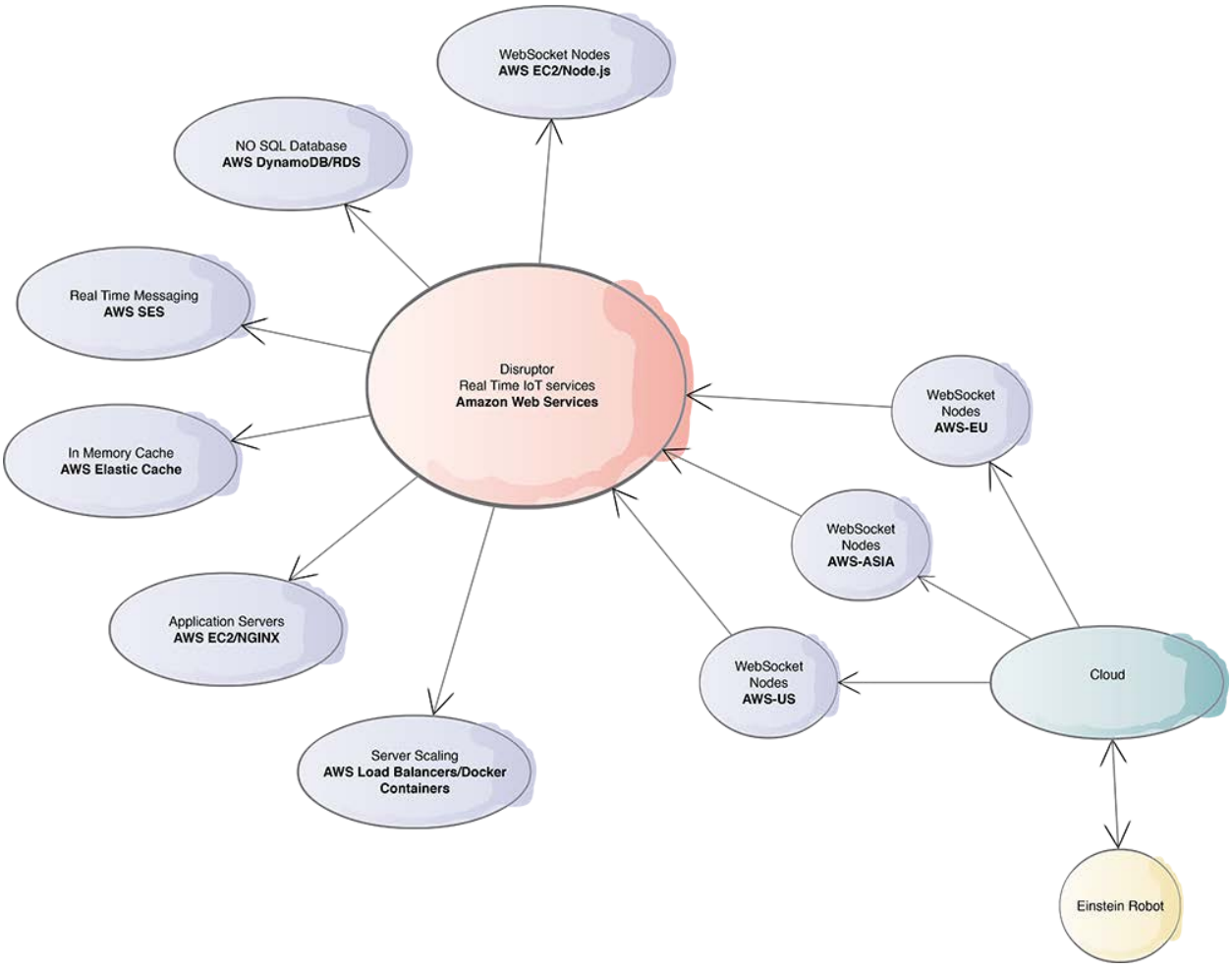
Puppeteering

The following steps translate puppeteer facial expressions onto the robot:

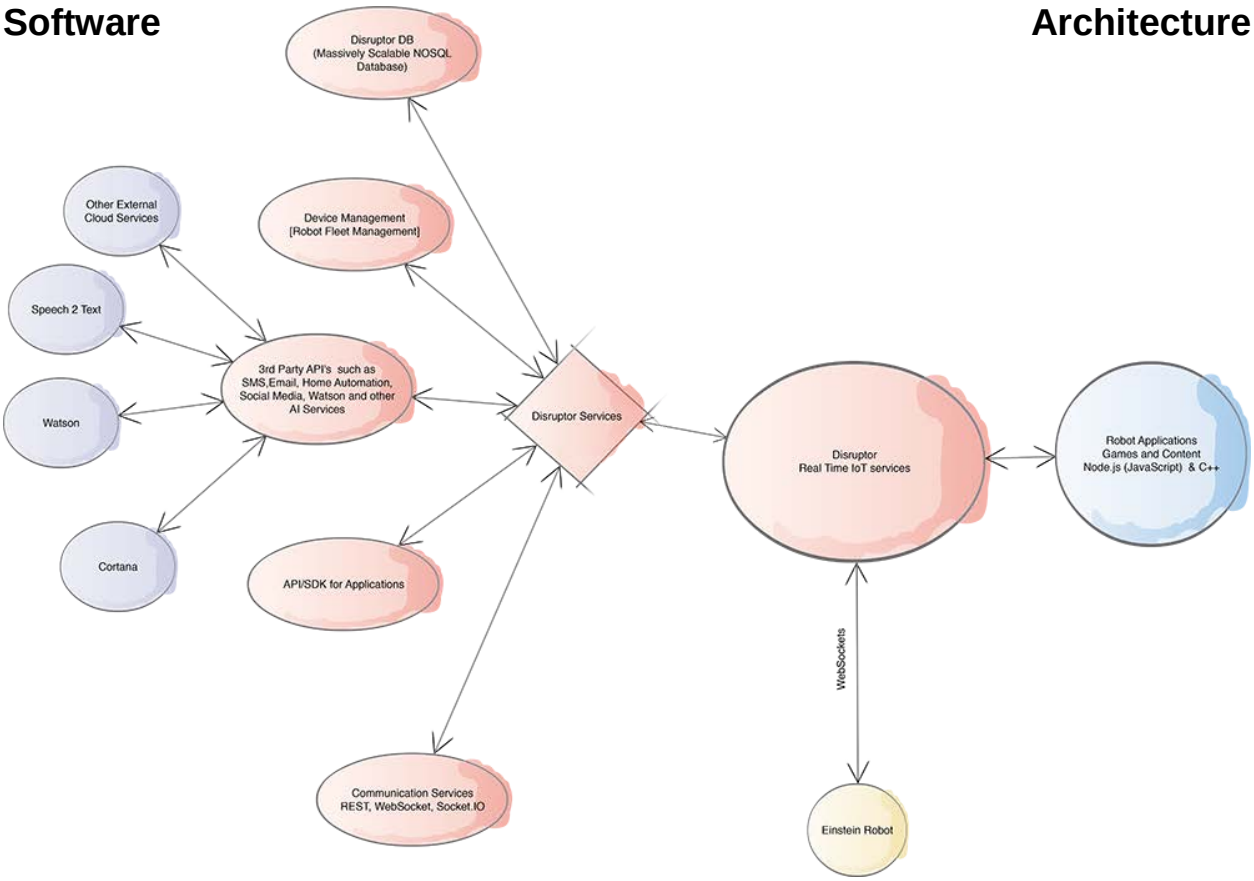


Dragon AI Implementation Reference

Network Architecture



Software Architecture



Uptime and Status of Dragon.AI Network.
http://status.dragon.ai/

Device Activation

During the setup process of the a device_id must be generated and stored on the robots memory.
HTTP POST: https://api.dragon.ai/device.register/

- client_id
- api_token
- device_name - used to ref uniqe_id in Robot DB (Robot:1001)

Results (JSON Format):

- device_name
- device_id
- device_secret

Status Codes

- 200 - (int) Successful Action
- 400 - (int) Not Found
- 500 - (int) Server Failed

TCP/TLS Endpoint

TCP Service Endpoint: socket.disruptor.io:9000

- device_id: Device ID
- token: md5 hash of device_id+device_secret+session
- session: unique / random alphanumeric string for server to calculate token

Send / Receive Format: JSON

Auth Device (Register Device as ONLINE)

Auth Device tells the network that the robot is online, should re-auth anytime the robot reconnects to the network.

TCP Send from from Robot or IPAD

```
{
  "cmd": "auth.register",
  "device": {
     "version": "1.0.0",
     "device_id": "device_id",
     "token": "md5_hash (device_id + device_secret + session_id)",
     "session_id": "unique_id(generated_by_robot)"
  }
}
```

TCP Response

```
{
  "cmd": "auth.response",
  "device": {
     "version": "1.0.0",
     "device_id": "device_id",
     "token": "md5_hash(device_id + device_secret + request_id)",
     "session_id": "unique_id(generated_by_robot)"
  },
  "validate": {
     "request_id": "id_generated_by_server",
     "request_token": "md5_hash (device_id + device_secret + session_id + request_id)"
  },
  "status": "200"
}
```

Response Validation

Every response from Dragon.AI will include a validate message. This helps confirm that the message is authentic.

Command Types Examples

TCP Intent Request (Sent from ROBOT > SERVER)

```
{
  "cmd": "activity.request",
  "device": {
     "version": "1.0.0",
     "device_id": "device_id",
     "token": "md5_hash (device_id + device_secret + session_id)",
     "session_id": unique_id(generated_by_robot)"
  },
  "relay": {
     "device_id": "device_id of tablet or phone",
  },
  "activity": "tfdemointro/demo_intro/research/",
 "intent": "gravity",
}
```

TCP Intent Response (Sent from SERVER > ROBOT)

```
{
   "cmd":                                                    "activity.recieved",

   "device":                                                                  {
      "version":                                                          "1.0.0",
      "device_id":                                                    "device_id",
      "token":    "md5_hash    (device_id    +    device_secret    +    session_id)",
      "session_id":                                  unique_id(generated_by_robot)"
   },

   "validate":                                                               {
      "request_id":                                        "id_generated_by_server",
      "request_token":   "md5_hash   (device_id   +   device_secret   +   session_id   +
```

```
request_id)"
    },

   "data":                                                                    {
      "activity":                                          "tfdemointro/demo_intro/animal/",
      "output":  "<MO=EB,0.55,1>  whats  your  favorite  animal  <MO=EB,0.01,1>",

      //  The  words  from  the  wordgroup  to  listen  for
      "intents":                                                              [
        "mouse",
        "duck",
        "dog",
        "cat",
        "pig"
      ],

      // Wordgroup File Name to search onboard memory for and download if not
present.
      "wordgroup":                                                "tfdemointro001",
      "wordgroup_url":                "https://static.dragon.ai/einstein/tfdemointro001.wg",
    },

    //  Tell  the  Server  who  to  relay  the  message  to
    "relay":                                                                  {
      "device_id":          "device_id  of  tablet  or  phone",
    },
    "status":                                                                "200"
}
```

Ping & Pong (keep alive)

TCP Intent Request (Sent from SERVER > ROBOT)

```
{"cmd": "ping"}
```

TCP pingResponse (Sent from ROBOT > SERVER)

```
{"status":"200","data": "pong"}
```

Testing URLS

Direct URL for a push
https://einstein.dragon.ai/intent/relay/ Intent_ID / Robot device ID
For example;
https://einstein.dragon.ai/intent/relay/10459347501463646/652df043941ec9fc81e2a7bb2a78c4ff2379822f
Auth and Push Page
https://api.dragon.ai/test.tcp

Text-to-Speech Comparisons

Name	URL	Cld	Emb		Licensing / Cost	Voices	Lang	Notes
Watson	http://www.ibm.com/smarterplanet/us/en/ibmwatson/developercloud/text-to-speech.html	YES	NO	JAVA, Pyrhon, Node, mobile	Free for 1 mil characters per month $0.2/1000 characters	13	9	
Vocalware	https://www.vocalware.com/#&panel1-1	YES	NO		scaling per stream 50k - $2.49 perK 1MIL - $1.29 per K	100	20	
Ivona	https://www.ivona.com/	YES	YES	JAVA	$1000/month (prepaid) + $0.003/unit for usage above 250k units/month.	51	23	best quality
ISpeech	www.ispeech.org/	YES	NO	Many languages and native mobile	pay per install or per word $0.1 - $0.2/word depending on plan	41	20	wide range of supported languages and devices
Bing	https://www.microsoft.com/cognitive-services/en-us/speech-api	YES	NO		5000 transactions/month free $4 per 1000 transactions	17	9	
Acapela	http://www.acapela-group.com/	YES	YES		unknown	100	34	custom available

Chatbots

A.L.I.C.E.

http://alice.pandorabots.com/

API.AI

https://api.ai/

WIT.AI

https://wit.ai/ Home page for Design Documents and Callouts

Motor Details

Robot Motor Controls

Robot motion occurs when a motor moves from one position to another. The speed of the motion may be defined by specifying a duration over which this transition is to happen.
To avoid great complexity, the initial position is assumed either to be known or unimportant; it is the final position that must be reached to show a desired expression or position.
Therefore, a command to control a motor includes the final motor position and a motion duration. The motor position is specified in a normalized range of [0, 1] and the duration is in seconds.
Precision is limited by the robot. For Little Einstein, the duration precision is approximately 100 milliseconds. Depending on the robot, scripting tools will automatically limit designers to the correct precision.

Linear Independence

Motors are assumed to be linearly independent. This means that specifying the position of one motor will not change the range of motion of another motor. If this condition is violated for closely spaced motors, range adjustments will have to be hand-tuned to obtain agreement between the simulator and robot.

Need for Macros

Individual motor control is reasonable when only a few motors are present. On larger robots with tens of motors, scripted group motor controls become increasingly necessary. Macros contain sets of motor commands to simplify scripting. For example, a macro might specify all the motor positions for a look of surprise.

Meeting Notes

2016-8-9 Big Robot Meeting

PROJECT GOALS (High Level)

Simulator to show robot actions for robot content development
Demonstrate ability to apply tool-set to any robot
Apply to robots with fewer or greater motors in various configurations
Demonstrate tools to easily creating content.
Toolset should have the ability to accept multiple third party integrations for text to speech, AI, etc.
Platform should enable live puppeteering for internal and external usage.

NOTES

Technical Direction for Facial Capture

- Realsense 300 camera -
    - Able to get it working
    - Unsure how they are doing cheek tracking
- Faceware vs. Realsense?
    - One can move ahead with other as back-up but leaning towards 300
    - Doesnt have to be completely settled now
- Simulator breakdown
    - How the motors move in the sim
    - How the input comes in
- Content creation may be a different UI/app than main toolset.

NEXT GOALS

1 Demo and 1 Design doc

DEMO

Sophia's face with a UI to move motors (29 motor control).
Adjust motor placement to adjust expressions.
Dummy interface with some initial wireframe blocked that is not yet functioning.

DESIGN DOC

Design doc to describe rest of the proposed design.

- Should have wireframes
- Initial mockups for a sample of how the app may look (Nick, Jonny, Andy to assist)

Next Steps:

1. Brad work out UI on motor movement and hand-off.
2. Then work on a big document showing all the things.

August 23rd to pull this together

Seth will review JIRAs with Brad and Fernando Thursday.