Robot Authoring Platform Design & Technical Design Document

Table of Contents

Contents

Revision History

0.8	May 31st, 2016	Document creation for Milestone 1

Overview

Our general approach is to expand upon previous methods applied to Little Einstein while adding new modules to suit the greater scope of big robots. For example, motor controls will be very similar to Little Einstein, but third party software needed to capture facial expressions of puppeteers will be a new feature.

Modes of Operation

Several modes of operation will be available to satisfy customer needs ranging from fully automated to fully scripted. It is possible to switch between modes via a super-user command, master application control, or one mode may request switching to a different mode.

Diagram

Scripted Interaction Mode

A *speech packet* holds text to be spoken interspersed with motor control commands. This animates the robot. A group of keywords associated with the speech packet are used to branch to other speech packets when the robot “hears” a user response. Other user inputs such as facial expressions or entering/leaving robot proximity may trigger a speech packet script.

Application Logic Mode

Very similar to Scripted mode, but allows for greater variety of user inputs and potentially dynamically generated speech packets.

Puppeteer Mode

The robot is an avatar for an actor. The actor may be remote but needs various hardware to capture input and receive feedback.

Actor says "Hello, what is your name?"
Microphone Audio → Speech to Text (STT) → Speech Packet Builder → Robot → TTS

Actor Smiles
Camera → Facial Motion Capture → Facial Expression → Speech Packet Builder → Robot → Motor Driver

User responds: "I'm Lilly."
Robot Camera → Image Capture → screen viewed by Puppeteer
Robot Microphone → Puppeteer headphone

Puppeteer responds "What a pretty name!" and so forth.

Autonomous Mode

Similar to Puppeteer Mode, but the robot is controlled by AI (Chatbot). If the interaction goes down a scripted path (many specific questions and answers may be anticipated) it can temporarily switch to Scripted Mode.

Motor Driver and Motor Configuration

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.

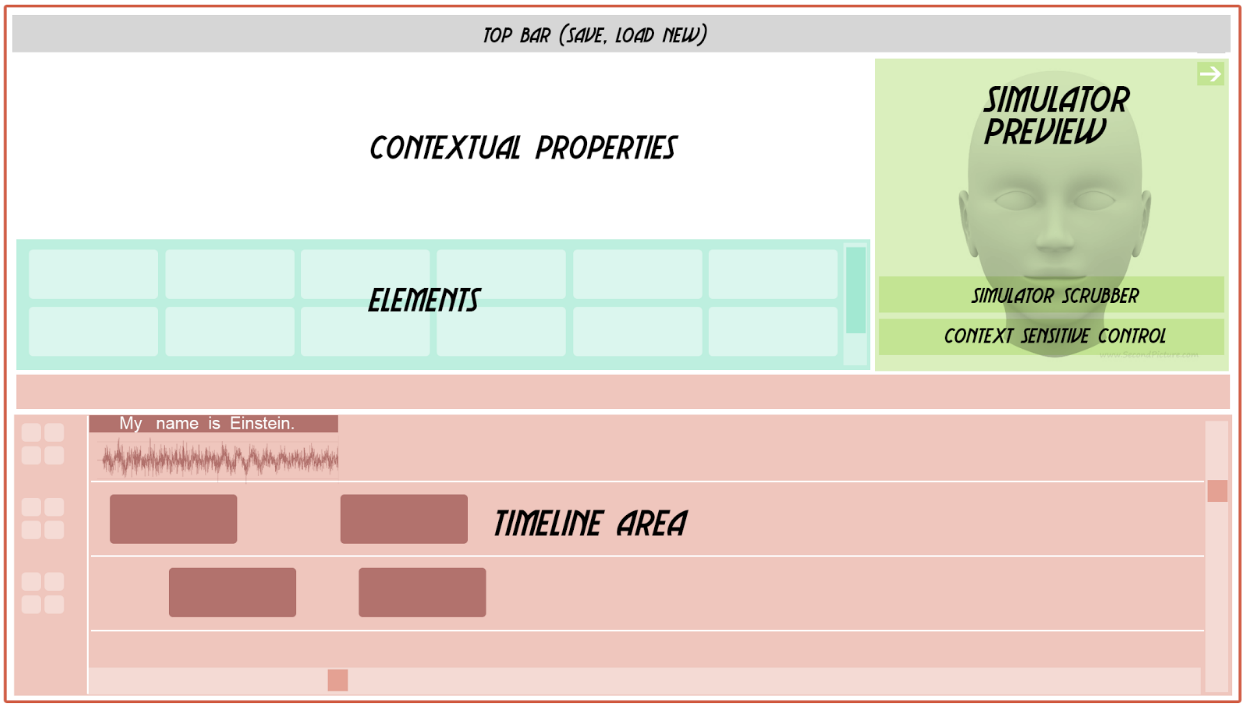
Need for Macros

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.

Authoring Tool

This software built as a web application 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.

Modules:
Dialog Trees
Sensor Logic
Chatbot Logic
Gameplay Logic
Motor Animation



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

Simulator

A simulator allows for testing and integration of content in advance of available hardware. The simulator is integrated into the Authoring Tool and is built with Unity and exported to WebGL.

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

Text can be pre-processed to record a list of phonemes to be included in the speech packet.

What software is used for this? What about other languages?

Motion Capture

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 puppetering 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.

<u>Note:</u> 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 Embeded 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.

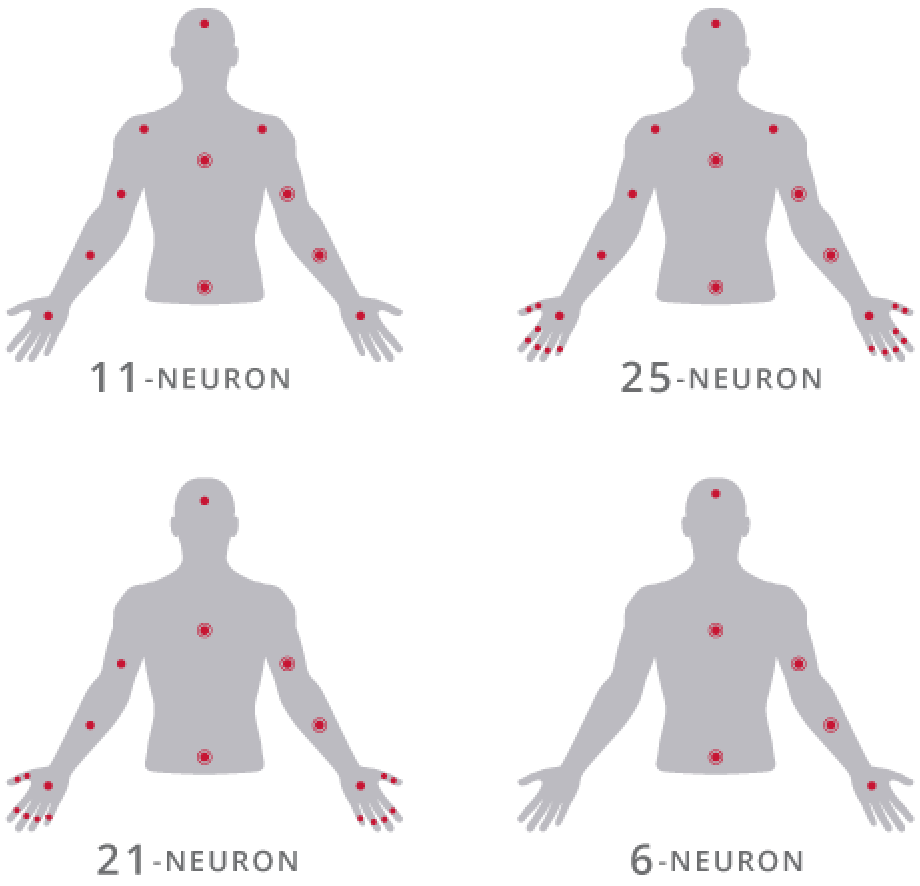
PERCEPTION NEURON was specially conceived as a professional tool for video game developers, film makers, visual effects professionals, biomechanics researchers, sports and medical analysts, and virtual reality enthusiasts to finally have a flexible and affordable platform to experiment with and push 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

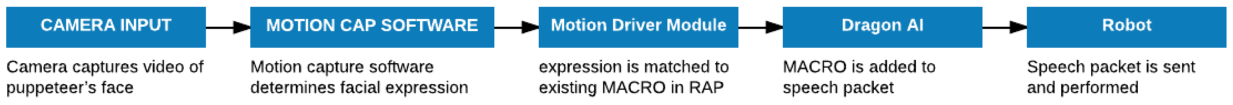
www.neuronmocap.com

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:


CAMERA INPUT
Camera captures video of puppeteer's face
MOTION CAP SOFTWARE
Motion capture software determines facial expression
Motion Driver Module
expression is matched to existing MACRO in RAP
Dragon AI
MACRO is added to speech packet
Robot
Speech packet is sent and performed

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.

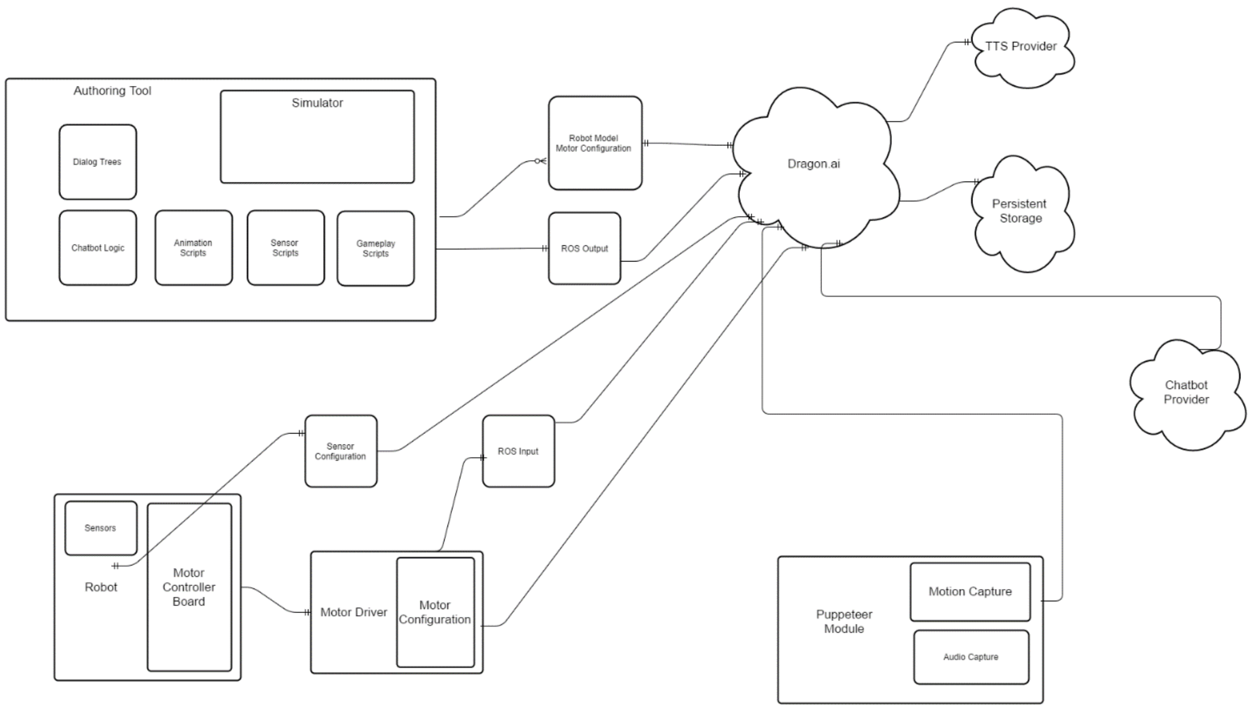
TEXT-TO-SPEECH

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 dsicreation of the platform user.

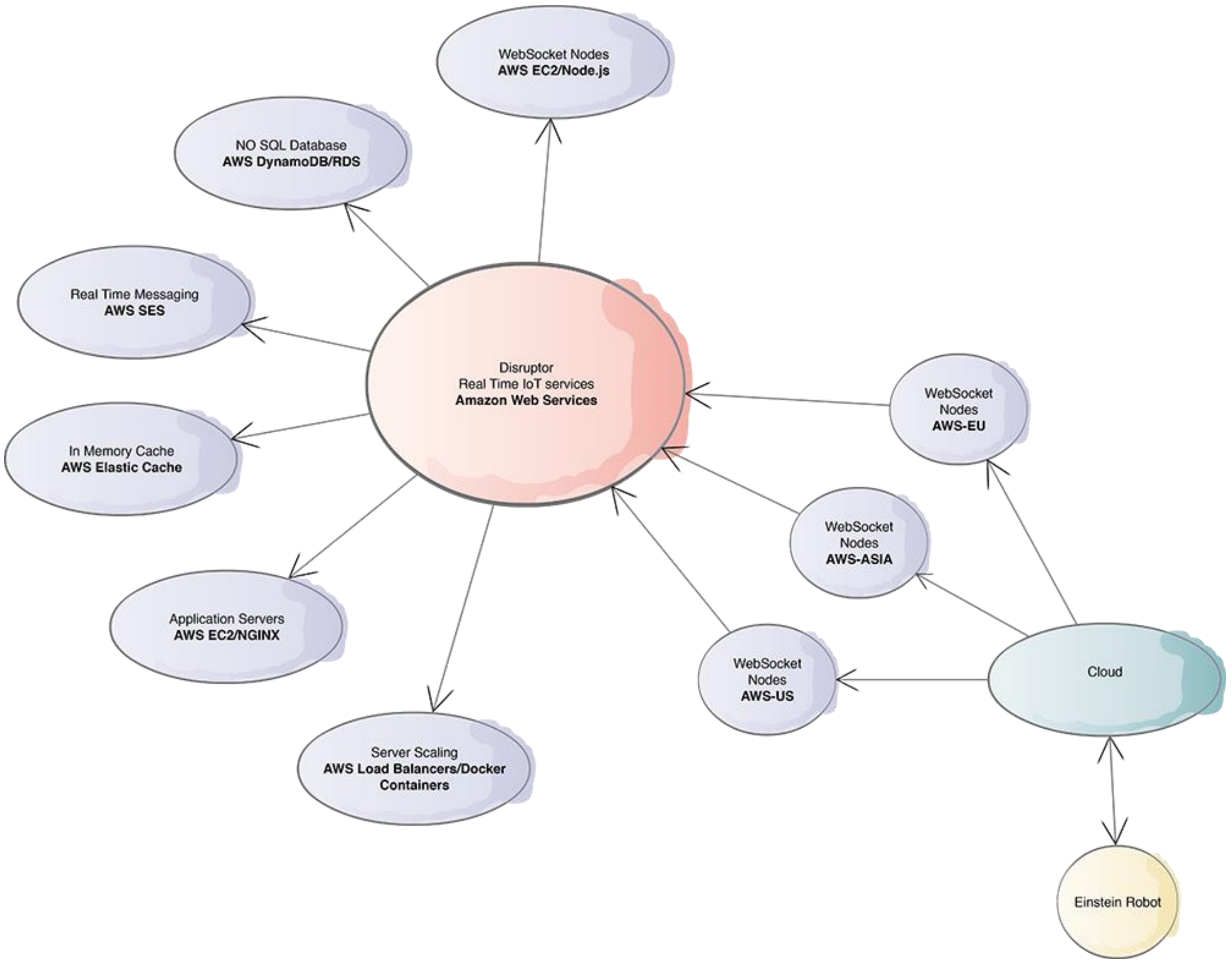
SecurityTBD

Deployment

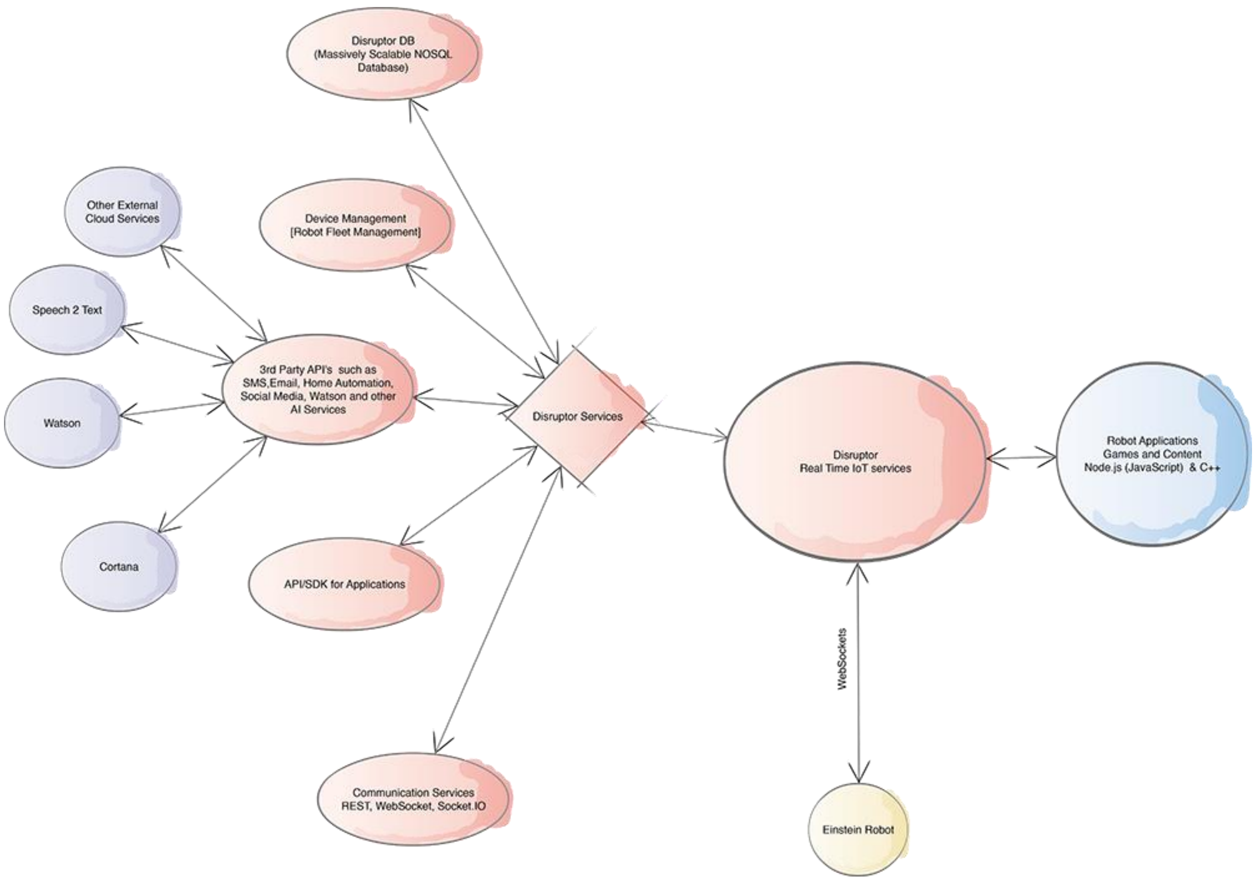


Appendix I: Dragon AI Cloud Backend

NETWORK ARCHITECTURE



Software Architecture


Disruptor DB (Massively Scalable NOSQL Database)
Other External Cloud Services
Device Management [Robot Fleet Management]
Speech 2 Text
3rd Party API's such as SMS,Email, Home Automation, Social Media, Watson and other AI Services
Watson
Disruptor Services
Disruptor Real Time IoT services
Robot Applications Games and Content Node.js (JavaScript) & C++
Cortana
API/SDK for Applications
WebSockets
Communication Services REST, WebSocket, Socket.IO
Einstein Robot

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/10459347501463646/652df043941ec9fc81e2a7bb2a78c4f f2379822f

Auth and Push Page

https://api.dragon.ai/test.tcp

Appendix II: Text-to-Speech Comparisons

Name	URL	Cloud	Embedded		Licensing / Cost	Voices	Lang	Notes
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
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	
Acapela	http://www.acapela-group.com/	YES	YES		unknown	100	34	custom available
Bing	https://www.microsoft.com/cognitive-services/en-us/speech-api	YES	NO		5000 transactions/month free $4 per 1000 transactions	17	9	
Ivona	https://www.ivona.com/	YES	YES	JAVA	$1000/month (prepaid) + $0.003/unit for usage above 250k units/month.	51	23	best quality
Vocalware	https://www.vocalware.com/#&panel1-1	YES	NO		scaling per stream 50k - $2.49 perK 1MIL - $1.29 per K	100	20	