# Animatronics
# Brain Document

By: Jim Schrempp and Bob Glicksman; v1.0, 12/3/2025

**NOTICE:** Use of this document is subject to the terms of use described in the document "Terms_of_Use_License_and_Disclaimer" that is included in this release package. This document can be found at:

https://github.com/TeamPracticalProjects/Animatronics/blob/main/Terms_of_Use_License_and_Disclaimer.pdf

# TABLE OF CONTENTS.

# DOCUMENT OVERVIEW.

- Theory of Operation
- 8x8 TOF Sensor
- Room Calibration
- Noise rejection
    - Spatial processing
    - Temporal processing
- Events
- Scenes and Scenarios
- Eye tracking based upon sensor data

# THEORY OF OPERATION

## Software Layers

Starting at the lowest layer, each servo is controlled by an instance of the animateServo class. You call this class with a command to move to a certain position at a certain speed and the class object will make it happen in a non-blocking way.

The next layer up is one instance of the class animatePuppet. This class owns all the animateServo objects. It can handle coordinated movements. For example, we have a method to “blink”. This requires the coordinated movement of four servos, one for each eyelid. In this method we might also have the focus of the eyes move just a bit to make the overall operation more life-like. If we had a servo to move the entire head left-right, then this would be the layer to implement moving the eyes at the same time the head moves so that the eyes remain focused on one spot in space.

On top of animatePuppet we have animationList. A scene is a group of animatePuppet method calls. An animationList is a set of Scenes that happen over time. For example, one scene might be: Look Up and Left and blink. A second scene might be: Look right, close the eyes slowly, then quickly open the right eye only. The animationLIst object can be used to make these two scenes play out over time:

1. Starting immediately, move from the current positions to scene 1.
2. Starting in 10 seconds, move from the now current positions to scene 2.

By stringing together a series of well thought out scenes, complex animation sequences can be run to make the head appear life-like.

animationLists provide the ability to have higher level concepts such as “go to sleep” or “wake up and look at the closest object” or (if we had neck movement) “wake up, quickly look at the nearest object and then slowly rotate the head so that it is facing the object head on”.

| | |
|---|---|
| main() | Calls animationList.process() |
| Class: animationList | A singleton. Define a series of “scenes” (mechanism positions) and then run through them as the clock ticks.<br>● addScene(scene, speed, delay) - predefined, or mechanism by mechanism<br>● startAnimation()<br>● stopAnimation()<br>● clearSceneList() |
| Class: animatePuppet | One for each puppet (we have only one); owns animateServo objects<br>● eyeball.look(x, y, speed)<br>● eyelid.open(position, speed)<br>● mouth.open(position, speed)<br>● Puppet<br>○ eyesOpen()<br>○ blink()<br>○ wink() |
| Class: animateServo | An instance for each servo. Moves servo slowly, or quickly.<br>● moveTo(x, speed) |

## The software stack

How it works.

1. Mainloop calls animationList.process() every time
2. animationList.process()
    a. notes elapsed time and decides if there is a new scene to set.
    b. Calls animatePuppet to set new scene positions.
    c. Calls animatePuppet.process()
3. animatePuppet.process()
    a. Decides if the movement of one servo affects another (e.g. moving eyes all the way left might cause the head to rotate left)
    b. Calls animateServo.process() for each servo it owns
4. animateServo.process()
    a. Notes current servo position vs target servo position
    b. Notes time and moves servo towards target position, if needed