Watts: A Mobile Laboratory Design

**Table of Contents**

[Overview 1](#_Toc69716582)

[Architecture of Watts 3](#_Toc69716583)

[Data Flow Design Diagram 3](#_Toc69716584)

[Services Diagram 5](#_Toc69716585)

[Persona / Parts Diagram 7](#_Toc69716586)

[Infrastructure Diagram 9](#_Toc69716587)

[Developer Flow Diagram 11](#_Toc69716588)

[Summary 13](#_Toc69716589)

# Overview

Watts is the name I've chosen for this new rolling robot.

Watts is an in-the-world laboratory for exploring subsumption architecture fused with machine learning to produce emergent behaviors that are interesting to interact with for the humans in their world.

A subsumption architecture (Rodney Brooks) is made up of layered behaviors that are able to stand on their own and that are combined in a priority scheme that allows only one to control the behavior of Watts at a time. Watts will evolve into presenting behaviors in multiple dimensions (movement, sound interaction, visual interaction) that will provide a rich playground for exploration of how a subsumption architecture can create interesting emergent behaviors.

Brooks demonstrated his ideas in a robot that included an Avoiding behavior, a Wandering behavior, and Exploring behavior and a Seeking behavior. These behaviors were directly sensing the world of the robot as it moved, and they were given a priority scheme that chose one of them to control the motion of the robot. There is no representation of the world built into the robot's software. The robot uses the world itself as the model it tests its sensor inputs against. It implemented the basic functions as an augmented finite state machine (AFSM).

Minsky, in *The Emotion Machine* (2007), presents a hierarchy of intelligent behavior.

|  |
| --- |
| Self-Conscious Thinking |
| Self-Reflective Thinking |
| Reflective Thinking |
| Deliberative Thinking |
| Learned Behavior |
| Instinctive Behavior |

In one review of his work, Brooks' subsumption architecture was designated as the learned and instinctive behaviors, while earlier AI ideas (SOAR and General-Purpose Problem Solving) were assigned to the Deliberative thinking layer. A new AI approach that uses language to guide perception and to describe events, eventually to tell and understand stories and to guide and understand culture (both in the macro society and micro personal realms) is an attempt to model the observed intelligence in Reflective, Self-Reflective and Self-Conscious thinking. This approach might be integrated into the Deliberative, Learned and Instinctive layers through a subsumption architecture, or there may be some other more useful way of fusing thinking into behavior.

Parts work in human psychology identifies narratives or behaviors in the unconscious that are communicating with each other (or not) and can control outwardly observable behavior. The naming of Parts indicates the function of the Part in a psyche. That practice is adopted in Watts as an interesting way to indicate intelligent behaviors that are analogous to behaviors observed in humans. In the end, Watts' Parts will interact with our Parts. This may be a kind of Turing test.

# Architecture of Watts

## Data Flow Design Diagram

The major component of the *architecture flow diagram* is the inclusion of all the moving parts. No details on how the pieces interact with each other are described, but the diagram does show the connections. It shows how data flows through the system.

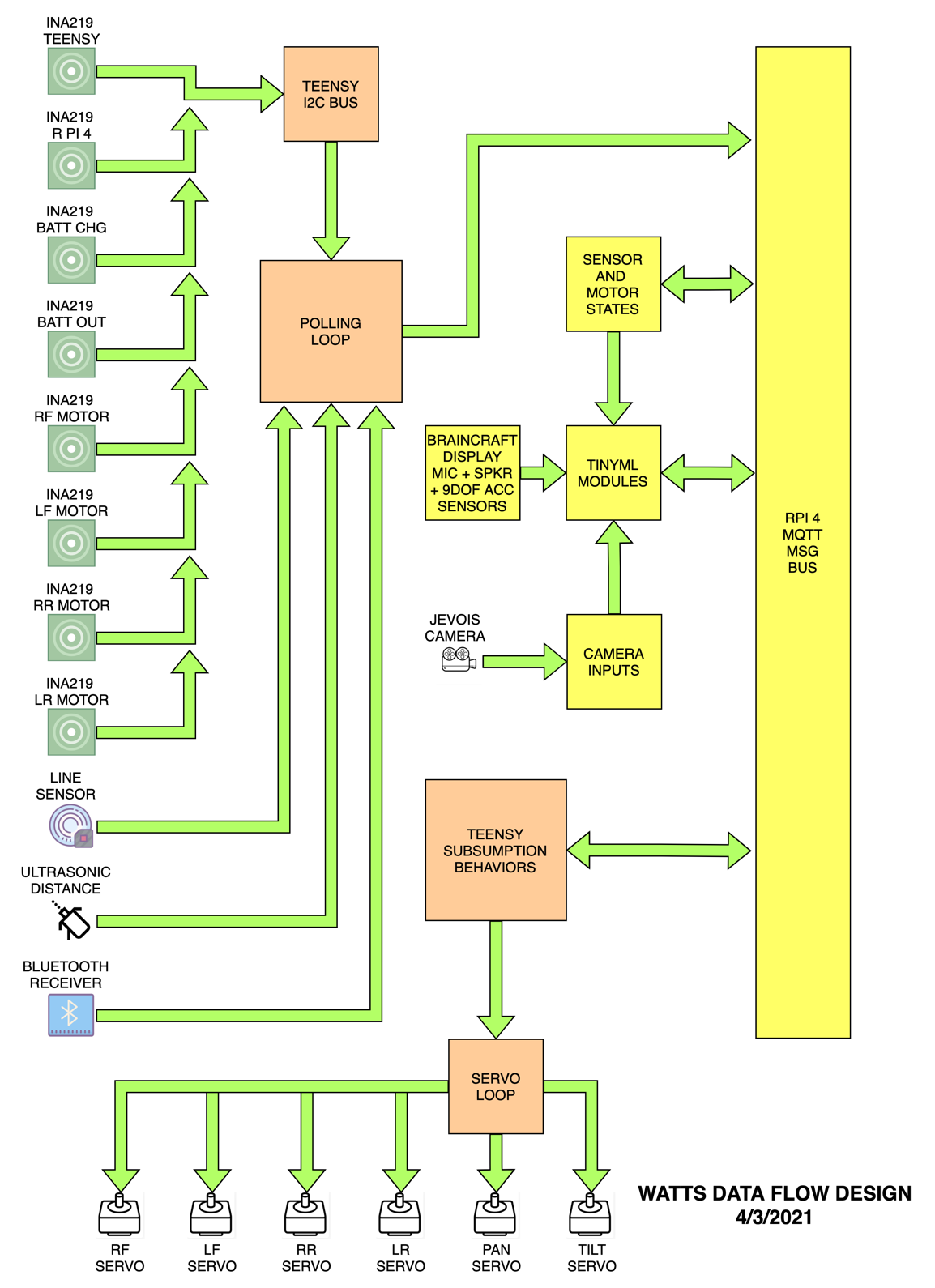


Figure 1. Data flow design. INA219 current-voltage sensors are accessed via I2C bus while other sensors have dedicated GPIO pins, as do the actuator data for servos. MQTT message bus serves data to and from controllers and sensors/actuators.

## Services Diagram

An *architecture service diagram* illustrates connectivity from a high level. It does not show any details on how the workflow or service works but instead shows the major pieces at play. This is a diagram intended to show the internal vs. external services used in an application. List all the microservices that make up the application or ecosystem. Label which services communicate with each other.

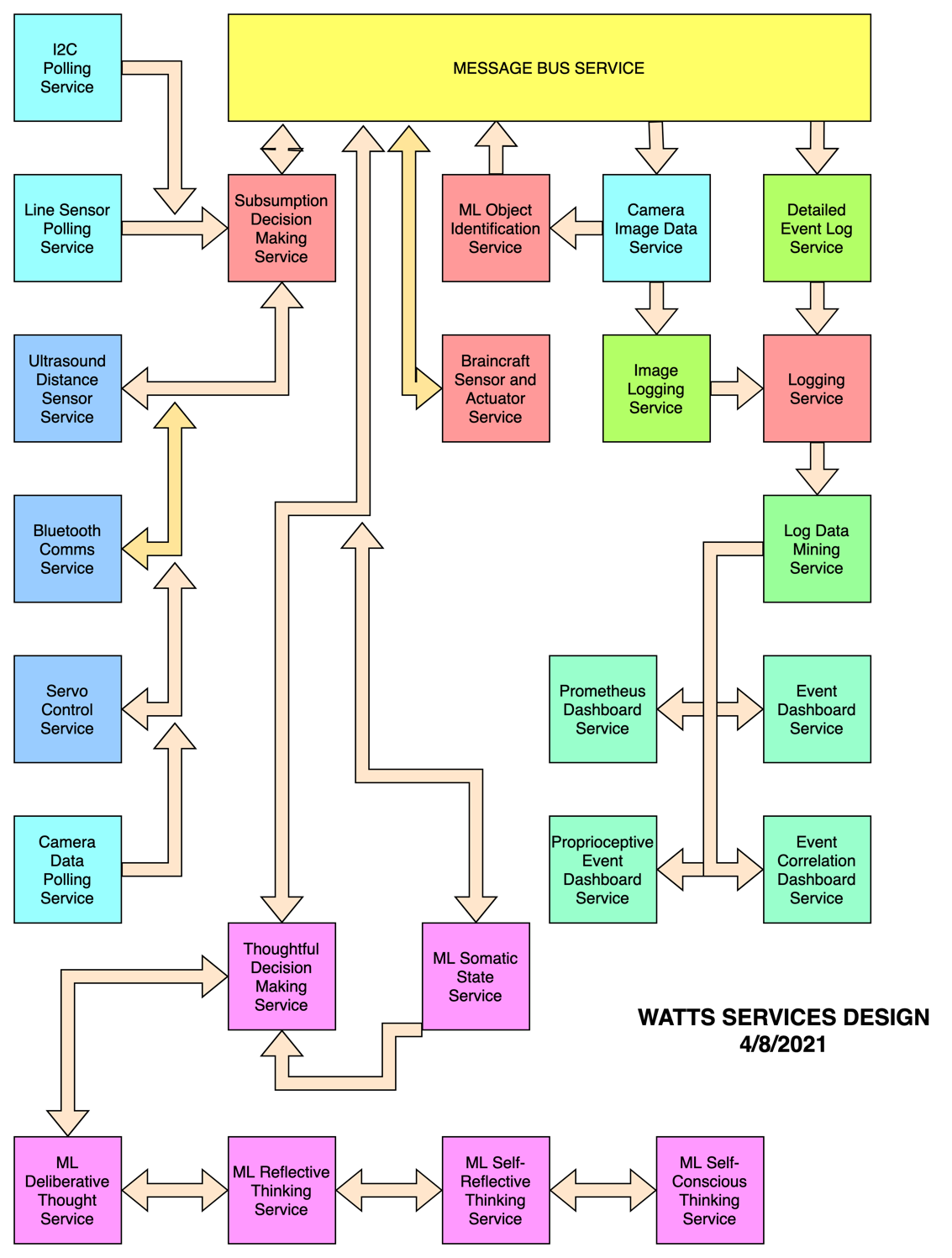


Figure 2. Service design diagram. Service colors indicate a general level of the service as described in the architecture narrative, not where they reside in the infrastructure.

*Data polling services* are colored turquoise in the diagram and are data flows from those services in one direction. The line sensor and camera data polling services are separate because they poll on dedicated GPIO pins of the direct controller (the embedded controller connected directly to the actuator and sensor hardware). The I2C polling service can harvest data from any I2C device connected to the direct controller.

*Actuator/sensor controller services* are colored blue in the diagram to indicate two-way data flow from the devices to the direct controller.

Both the data polling services, and the actuator/sensor controller services provide data to the *subsumption decision making service*. This service consumes sensor data and uses an augmented finite state machine (AFSM) to determine the instinct level behavior to select based on that sensor data. This service logs the sensor data received to the message bus and takes input for the AFSM from the higher-level thoughtful decision-making service, integrating that input into the AFSM for complete behavior control.

The *detailed event log service* in green on the diagram consumes data from the message bus as provided by the subsumption decision making service and uses the *logging service* to write logs to persistent storage for later use by other services.

The *camera image data service* consumes image data directly from the camera hardware and correlates that data with the camera sensor data from the message bus. That data is logged through the image logging service.

The *Machine Learning (ML) object identification service* consumes data from the camera image data service and produces object identification data that it provides as events to the message bus.

The *ML somatic state service* consumes device state data from the message bus provided by the subsumption decision making service and aggregates it into a state ready for consumption by the *thoughtful decision-making service*.

The *thoughtful decision-making service* is an AFSM that takes into account the somatic state and the inputs from higher level ML thinking services and provides behavior selection events to the message bus for both logging and consumption by the subsumption decision-making service that controls the hardware actuators and sensors.

The purple-colored thinking services modularize ML services for higher level functions according to Minsky’s framework. They are, in hierarchical order, the *ML deliberative thinking service*, the *ML reflective thinking service*, the *ML self-reflective thinking service*, and the *ML self-conscious thinking service*.

The *log data mining service* consumes data from the logging service that includes device event data, subsumption decision-making events, object identification events and thoughtful decision-making events and analyzes and aggregates them as needed for downstream monitoring and analytical services.

The consumers of the data from the log data mining service are all dashboards meant for human interaction with developers and humans interacting directly with the robot. The anticipated services for monitoring in this way are *Prometheus dashboard service*, *event dashboard service*, *proprioceptive event dashboard service*, and *event correlation dashboard.*

## Persona / Parts Diagram

A persona and parts diagram describes a chronological view and actors in a particular workflow. Personas are higher-level entities like the robot or a human and Parts are deeper entities in the system like Battery Level and Wheel Stall. A Part will typically be named for a trigger event, whether that is a sensor threshold or a system state change. A Persona will be named for the higher-level entity that is perceived as taking a higher-level action. Watts and Humans and the Environment all can have multiple Personas and Parts.

Make use of swim lanes to show the different actors in a workflow. This type of diagram tends to be lower level, as it includes more detail than the others. Be sure to label the personas and parts with one swim lane for each, the events that the Persona or Part triggers, the workflow of actions that the Persona or Part takes, and the change of system state as a result of the actions. The change of system state can become the trigger event for another Persona or Part.

{DIAGRAM}

Personas / Parts

Wheel Stall

Battery Charging

Battery Power

Pi Power

Pi CPU Temp

Teensy Power

Line

Ultrasound Distance

Pan Tilt Direction

Bluetooth Message

Message Bus

Voice Request

Noise Detected

Answer Request

Make Voice Request

Tone Alert

Display Image

Display Object ID

Display State

Accelerometer

Camera Object ID

Camera Image

Avoid Obstacle

Find Target

Find Charging Station

Ask for Help

Avoid Steps

Find Person

Voice ID

Request ID

Event Logger

Image Logger

Analytics Dashboard

Event Dashboard

Prioperceptive Dashboard

Event Correlation Dashboard

Deliberative Thinker

Reflective Thinker

Self-Reflective Thinker

Self-Conscious Thinker

## Infrastructure Diagram

It represents everything that has been implemented. A low-level diagram in nature, it is meant to be inclusive of everything that exists in a service/application/ecosystem.

The purpose of this diagram is to show what has been built and where it has been built. At the top level, the hardware that runs the application is depicted and the services that run on each piece of hardware are enumerated. In a robotic system, the number of sensors and actuators will be greater than the compute instances, in most cases. But the services running on the compute instances will far outnumber those associated with the sensor and actuator hardware. The flow of data through the hardware from sensor and actuator to service to service to application will create a roadmap for an operator of the system to monitor critical connection points to indicate correct operation and to be used to find the problem when a fault occurs.

When building an *architecture infrastructure diagram*, don’t leave out any pieces. The goal of this type of diagram is to show everything in your app and how it all connects. You don’t need to go into too much detail on *how* it works but rather focus on getting all the pieces of your app included in the diagram.

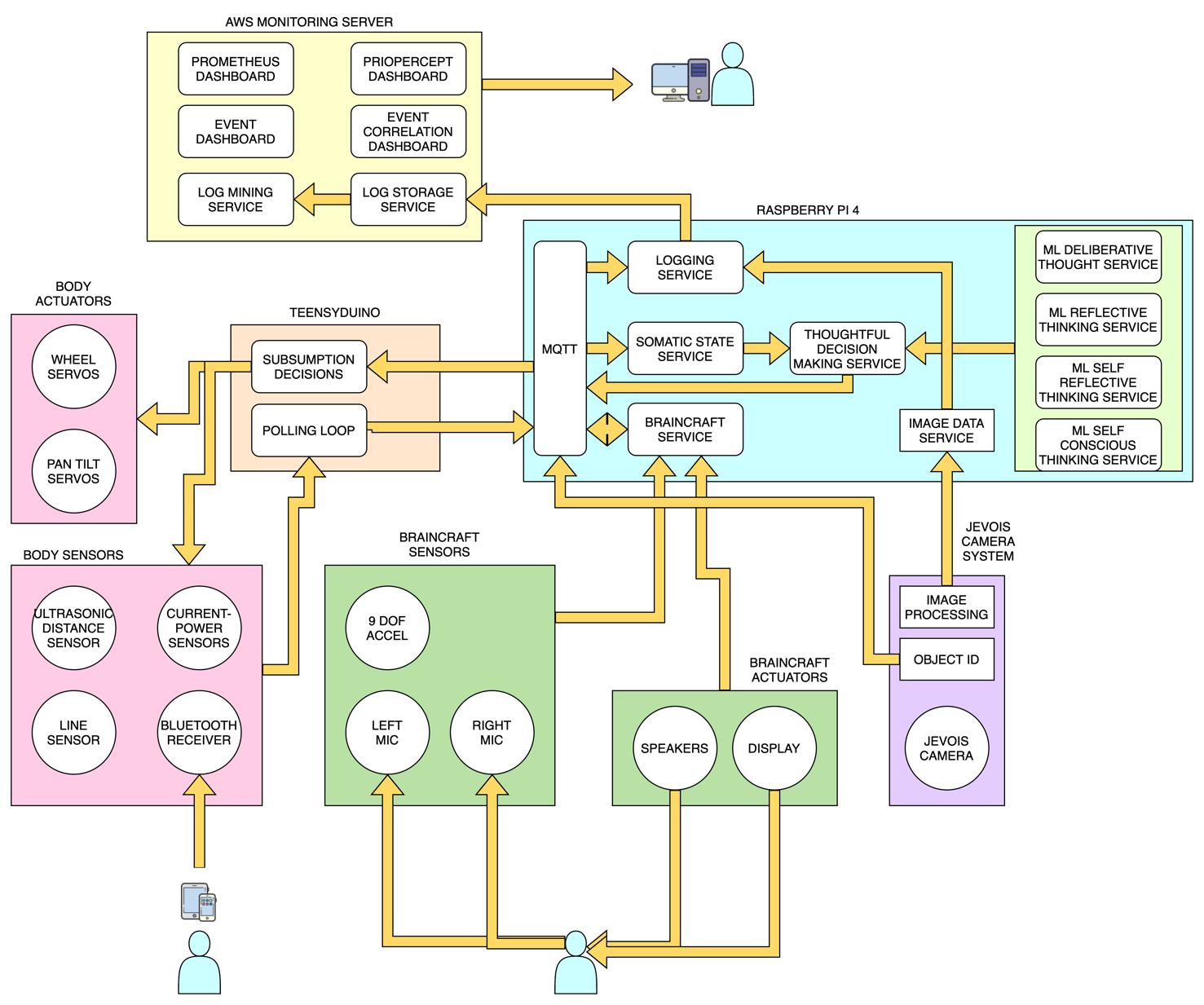


Figure 3. Infrastructure diagram. Colors generally indicate hardware components where functions are implemented.

Sensors

Actuators

Teensyduino

Raspberry Pi 4

Cloud Monitoring Server

## Developer Flow Diagram

The Developer Flow Diagram includes everything a developer would need in order to build the solution. The goal is to answer any questions that might come up by looking at a *flow diagram* and include them in the design. This is the lowest-level diagram and is intended to get the idea across without your presence. The *architecture developer diagram* is essentially the *flow diagram* with added detail. Label each piece with any specific implementation detail you can think of and be sure to label important transitions.

Polling Loop [Arduino] PowerMonitors, LineDetector, UltrasoundRange, Bluetooth, CameraData

Subsumption Behavior [Arduino] {behavior list}

Servo Loop [Arduino] WheelMotor(x4), PanTiltMotor(x2)

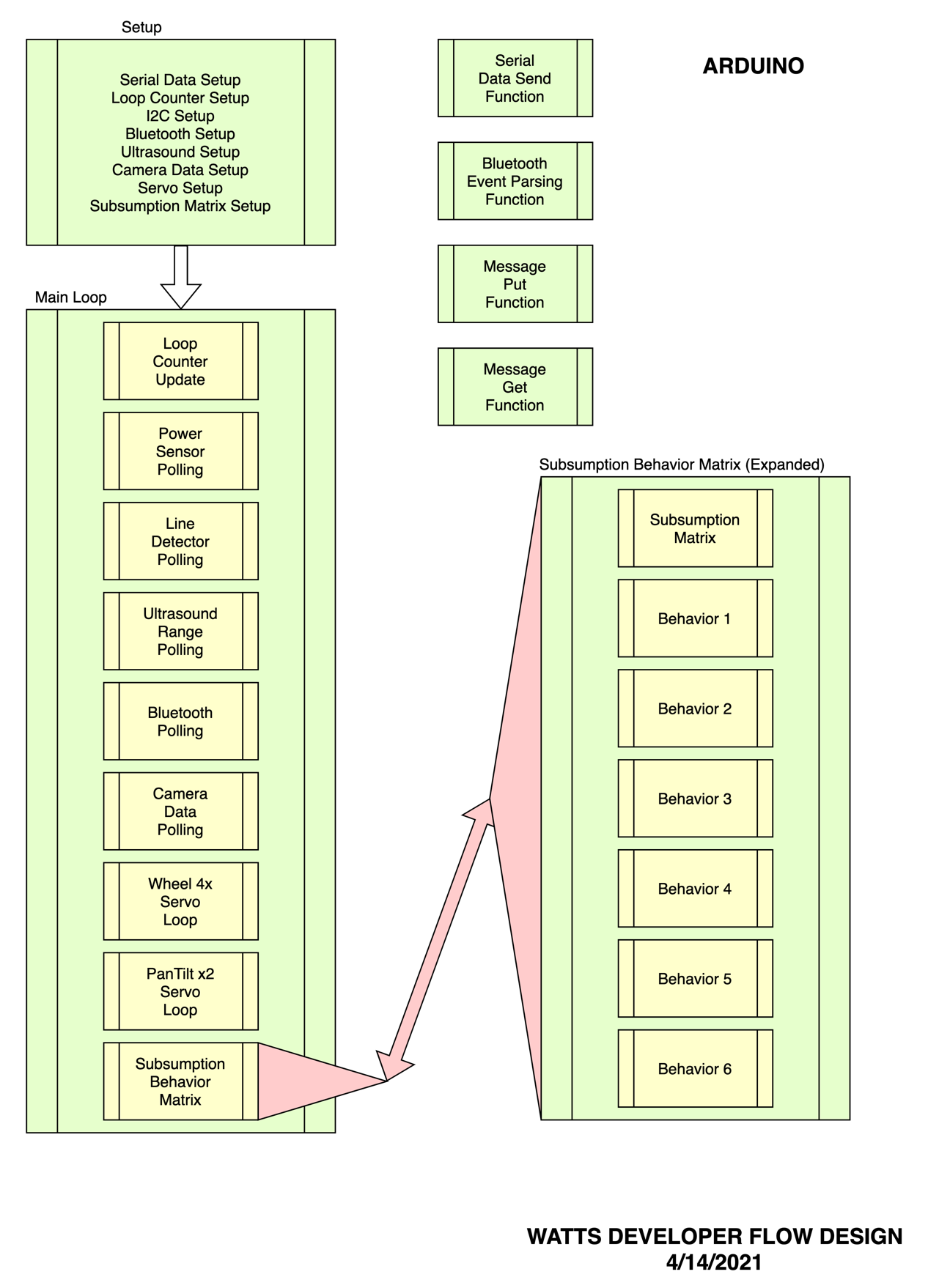


Figure 5.1 Developer Flow Diagram for Arduino Code. The Arduino code architecture consists of a Setup Section and a Main Loop which calls sub-loops and uses functions to handle all of the services in the flow of the device.

Message Bus [RPi] MQTT and configuration

Sensor and Actuator State Service [RPi] {tbd}

TinyML Modules [RPi] ImageCapture, ObjectId, NewObjectTraining, ThoughtfulDecisions, MLDeliberativeThought, MLReflectiveThought, MLSelfReflectiveThought, MLSelfConsciousThought

Logging [RPi] ImageLog, EventLog, BehaviorLog

Monitoring [Cloud] ImageBrowse, PrometheusDashboard, EventDashboard, PrioperceptiveDashboard, CorrelationDashboard, DataBrowse

## Summary