ARIA

Autonomous Real-Time Interactive Architecture

Introduction - Matt

- Rename from “Machine Learning Smart Home”
- ARIA - Autonomous Real-Time Interactive Architecture
- Autonomous - The system can operate without user input
- Real-Time - Event driven system that responds to environmental stimulus
- Interactive - Allows the user to monitor and control the system through a simple interface
- Architecture - A system that supports multiple vendors and allows for extension to new devices

Overview - Matt

- Introduction - A brief introduction of the system, what it is, how it works and where we are at
- Demo - A demonstration of the current product
- Technical Details - Some more technical information about the implementation of the system
- Project Roadmap - Where we want project to go
- Meeting Objectives - How we plan to meet objectives and our development workflow

Project Context - Peter

- We want to give some background of the smart home industry, as they have been around for awhile now.
- We want to look at some existing systems and how our stands out
- But first, we would like to define the terms in our project

“Smart Home” - Definition

- Difference in our definition is that we want to be available for any internet of things device

“Machine Learning” - Definition

Existing Home Automation - Jeremy

- Existing systems are closed to proprietary hardware
- Requires manual scheduling and configuration
- Basic learning for single function
- Can’t directly interact with hubs (need to go through servers)

Motivation - Cameron

- Our system wants to join multiple technologies in a simple, automated manner
- We want to bring this novel, technical setup to the average consumer
- We want reduce or even eliminate the need for manual configuration of devices

Project Objectives - Matt

- So technically speaking, what do we want this system to offer the user

Project Scope - Peter

- Started with building our own devices, realized quickly that this was not a feasible solution, distracts from the purpose of the project
- This presented a safety hazard, as we would need to work with high voltages
- Decided to move to 3rd party devices, mainly zwave
- Simple setup, configuration and monitoring of devices
- Provide control of devices connected to the system through a simple, user friendly interface
- Provide a learning mechanism for auto-configuring the system to the user’s needs

Demo

Technical Details - Matt

- We would like to discuss the technical implementation of the system

- This will dive into some specifics of the design choices and patterns used to build the system

Scenarios - Matt

- We wanted to model how a user would interact with the system. We modeled the interaction of users and the devices as well as the interface of the system
- From these interactions, we need to make be able to capture the interactions for data processing. Sometimes, the data that is captured does not tell the whole story of the interaction. Some of the data requires filling in the blanks through the extraction of properties from the data. These scenarios gave us some insight into what types of data can be extracted and how it could be determined
- Finally, the extracted data and the captured data needs to be used to determine the behaviours that is learned. The scenarios allowed us to envision what behaviours should be learned by the system given specific inputs

Use Cases - Jeremy

- Use case diagram - Why do we have it, how do we use it for development
- 1 user - with both technical and non-technical operations
- 1 interface to allow this behaviour
- 3 modes:
    - Standby - Operational but no learning or behaviours
    - Normal / Playback - Operation using the learned behaviours
    - Learning / Training - Teaching the system a behaviour based on given inputs

Non-Functional Requirements - Cameron

- Responsiveness - want to be able to see a change in the system immediately
- Security - don’t want anyone to be able to control devices
- Availability - needs to be available 24/7
- Reliability - When the user schedules something, they expect it to work

Communication Protocols - Peter

- ZigBee
- Z-Wave
- Bluetooth
- WiFi
- INSTEON

Protocol Comparison - Peter

WiFi is overkill for smart devices, draining the battery.
ZigBee is the parent of Z-Wave, but does not have a standard API, leading to compatibility issues between devices. Bluetooth is on the same frequency as WiFi, which can lead to lag in response time (one of our non-functional requirements). It also follows a star topology, which has a single point of failure (more non functional requirements). Z-Wave has an implemented standard API, making it easy to use.

System Design - Cameron

Basic overview of components
- 3-tier architecture
- patterns
    - MVC
    - Adapter
    - Client-server
- Say we have integration points between each layer

Data Storage - Jeremy

- Compared various data storage techniques
- Data needs to be structured for processing and displaying so it was best to store it in that way
- Decided to use structured, relational storage
- No need to have heavyweight storage, we have limited storage and limited resources
- Chose SQLite for storage of local events

Project Roadmap - Matt

- Where do we want to go with the project?
- We want to build upon the existing platform to expose more information to the user
- We need to add a learning component to the system

System Statistics - Peter

- The system will have to many logs to view individually
- Graphs give a quick overview of the general behaviour and performance of the system

- Gives the user good insight into the operation of the system

Scheduling - Jeremy

- Want to provide the user with the ability to perform manual scheduling of events
- Events can be one time, repeating many times or just a few times

Machine Learning - Matt

- Considered a number of algorithms for machine learning including k-nearest-neighbours, k-means, and state vector machines
- State-vector machines are the ideal option and would allow us to preserve the currently learned model when a new device is added
- The key to our machine learning will be the feature extraction from the captured data in the system

Meeting Objectives - Blanch

- To meet objectives up until this point and to continue to meet objectives we have a development workflow

Code Lifecycle - Cameron

- One a weekly basis we have a backlog grooming meeting to select tasks for our week long sprint

Sprint Tasks

- Sprint tasks are ordered by priority and size and moved through the sprint workflow

Code Lifecycle (2)

- For each feature we first write the specification in the form unit tests
- Next, the feature is implemented. This is typically done in parallel with testing. Both the test and the code are modified until the test passes
- All of the development is tracked using git and once the feature is complete it is pushed to github
- Once the branch is pushed it kicks off continuous integration build which tests the entire system. The feature is also tested for basic functionality with manual functional tests. Finally, all code is reviewed by a peer before it is ready to be added to the system

- Once the feature branch has been tested and verified, a pull request is created and it is merged into the system

Summary

Peter

- Where are we?
    - We can discover devices
    - Have a real-time updating event feed
    - We have a working interface that can control devices

Jeremy

- Where we want to go?
    - We want to expand the functionality of the system to include scheduling, statistics and machine learning

Matt

- We want to strive for an autonomous system that requires minimal user input to receive events from sensors in real-time. The system must provide a completely interactive interface for a user to perform technical and non-technical tasks on an architecture that supports many different devices