Aware Alexa

Sydney Hung
Michael Lo

Introduction

- Smart devices and personal assistants are become more common
    - Garage door sensor, thermometers, light dimmers, etc.
    - Alexa, Siri, Google Assistant, etc...
- Derived Empathetic and Aware Alexa project idea
    - Give personal assistant awareness of user’s state and context of an event
    - Our project allows Alexa to differentiate different users, to keep track of users’ states, and to greet them accordingly as they entering or leaving a house
- Aware “*Alexa?*”

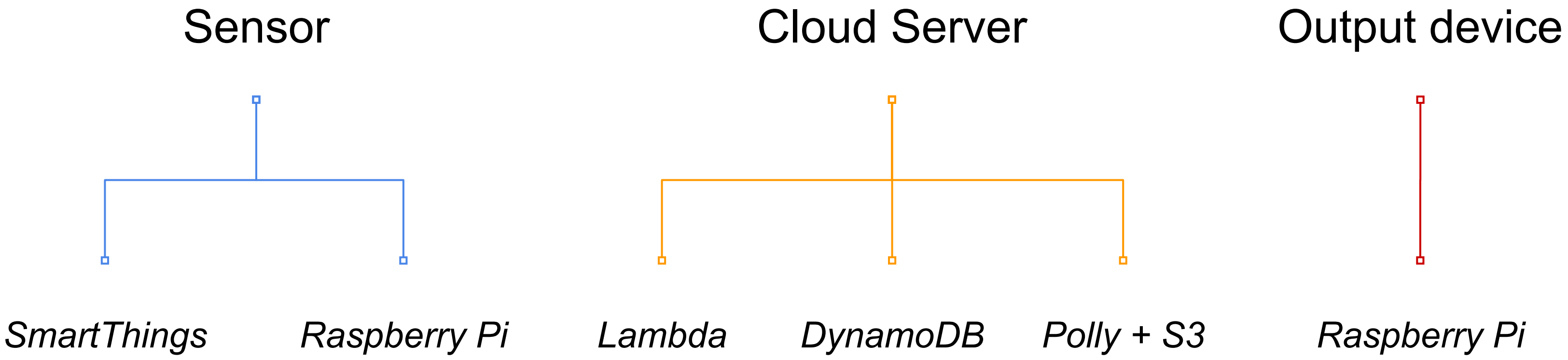
Related Work

- Integrated smart devices
    - Allow an event to trigger something else to happen
    - Ex: a voice command to turn on a coffee machine
    - Ex: when door is opened, turn on a light
    - Based solely on sensor readings, and have no concept of history or states
- Provide more detailed event context through tracking users’ states
- Introduction of Contact and Motion Sensor API
- SmartThings hub and sensor

Approach

- Integrate multiple sensor to determine unique user
- Usually, people have smartphones and carry it with them at all times
    - Each smartphone has an unique Bluetooth MAC address
    - We can sense if a person is near a sensor by pinging their device
- Perform checks depending on readings of sensors
    - If door is opened... check if user is near door… check if user changes state of distance sensor
    - When a specific state is reached, play a message tailored for that user
- Provide an Alexa App
    - Add & modify users and the inputs are stored in a AWS database

Implementation Overview



Sensor

SmartThings

Raspberry Pi

Cloud Server

Lambda

DynamoDB

Polly + S3

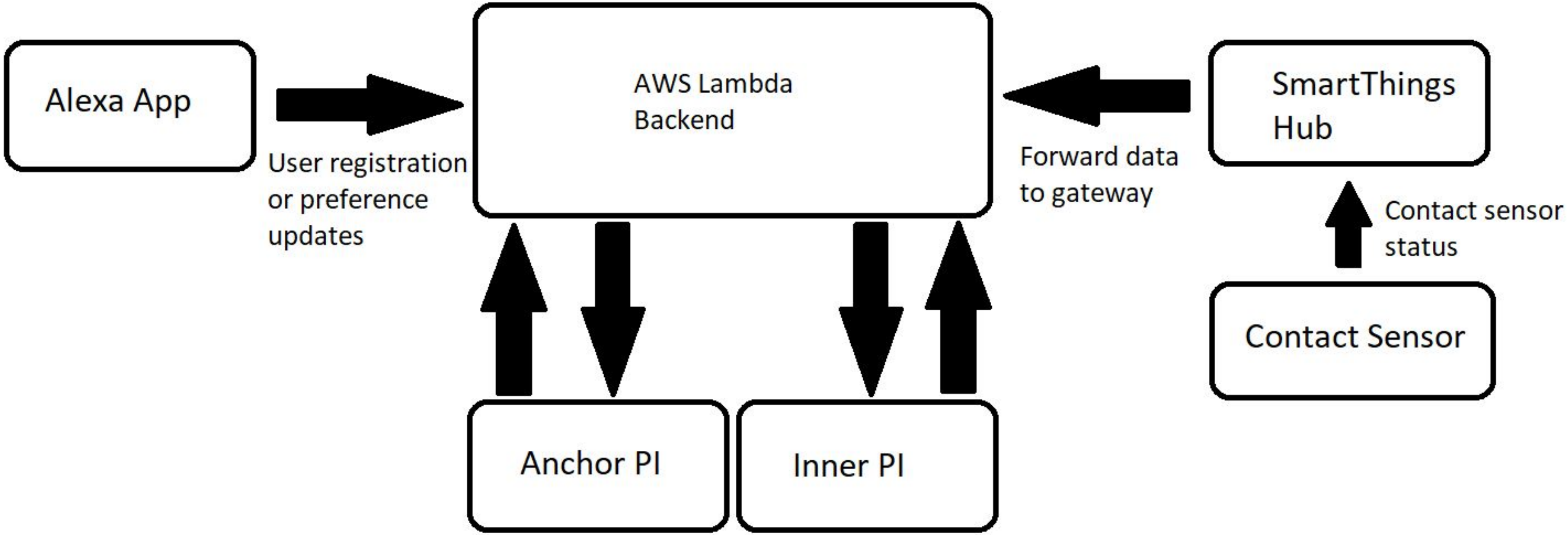
Output device

Raspberry Pi

Components

- Two Raspberry PIs
  - One is mounted on door to determine if user is nearby
  - The other is in the room for detecting if user goes through door and plays custom messages
  - Both ping registered users Bluetooth MAC address and sends to AWS for processing
- Contact Sensor + SmartThings Hub
  - Using a custom SmartThings app, the hub sends the data from sensor to AWS
- AWS Backend
  - Provides a HTTP endpoint for sensor data to be posted
  - Generates user specific message and tells the inner PI when to play a message

Architecture

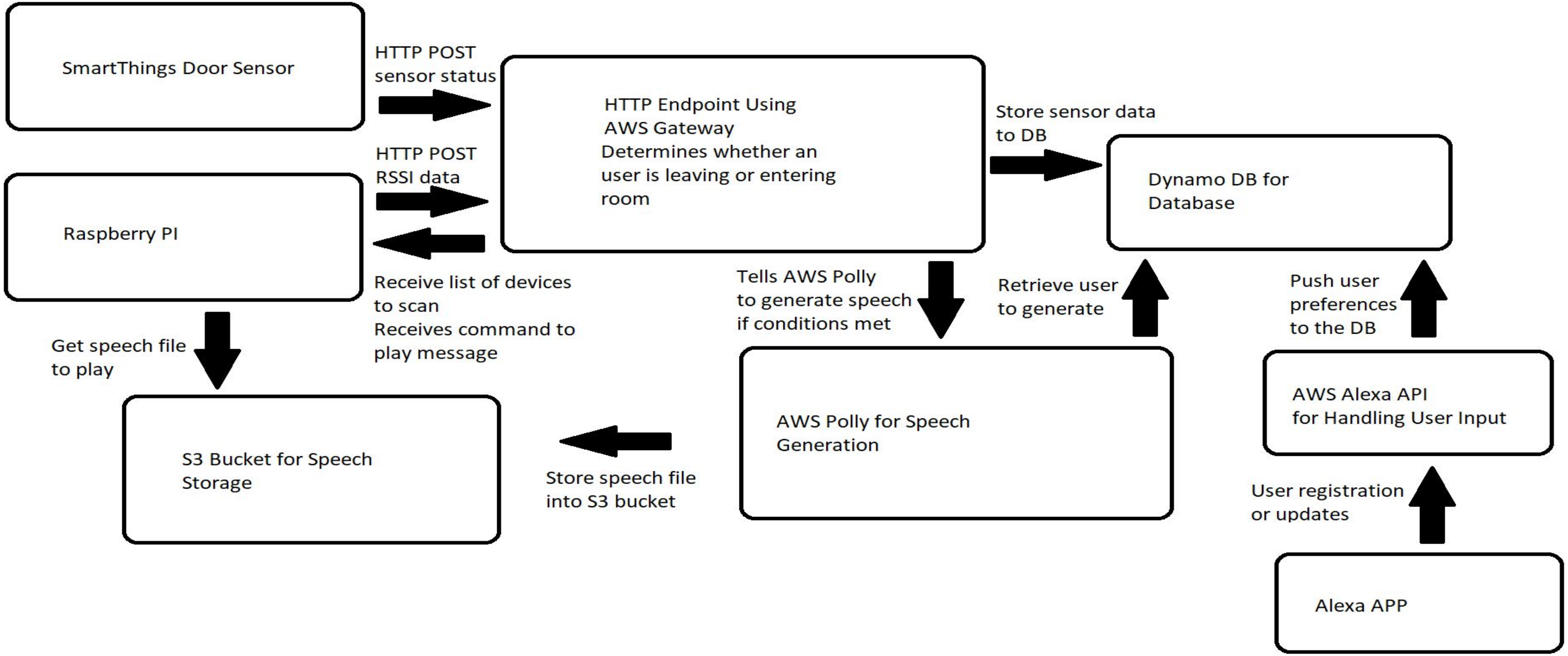


-The PIs retrieve Bluetooth MAC addresses to scan
-The RSSI values are forwarded to gateway
-Gateway tells the Inner PI whether or not to play customized message

AWS Backend Overview

- HTTP Endpoint
  - Data from the sensors are sent here using HTTP POST, and stored into a database
- AWS DynamoDB
  - Store sensor data and users’ state and preferences
- AWS Polly
  - Generates synthetic speech tailored to say “welcome” or “goodbye” to a specific user
  - Ex: “Welcome home Sydney”
- AWS S3 Bucket
  - Stores the speech file generated by AWS Polly and is the place where the PI retrieves it

AWS Lambda Backend



SmartThings Door Sensor

HTTP POST
sensor status

HTTP POST
RSSI data

HTTP Endpoint Using
AWS Gateway
Determines whether an
user is leaving or entering
room

Store sensor data
to DB

Dynamo DB for
Database

Raspberry PI

Receive list of devices
to scan
Receives command to
play message

Tells AWS Polly
to generate speech
if conditions met

Retrieve user
to generate

Push user
preferences
to the DB

Get speech file
to play

AWS Alexa API
for Handling User Input

AWS Polly for Speech
Generation

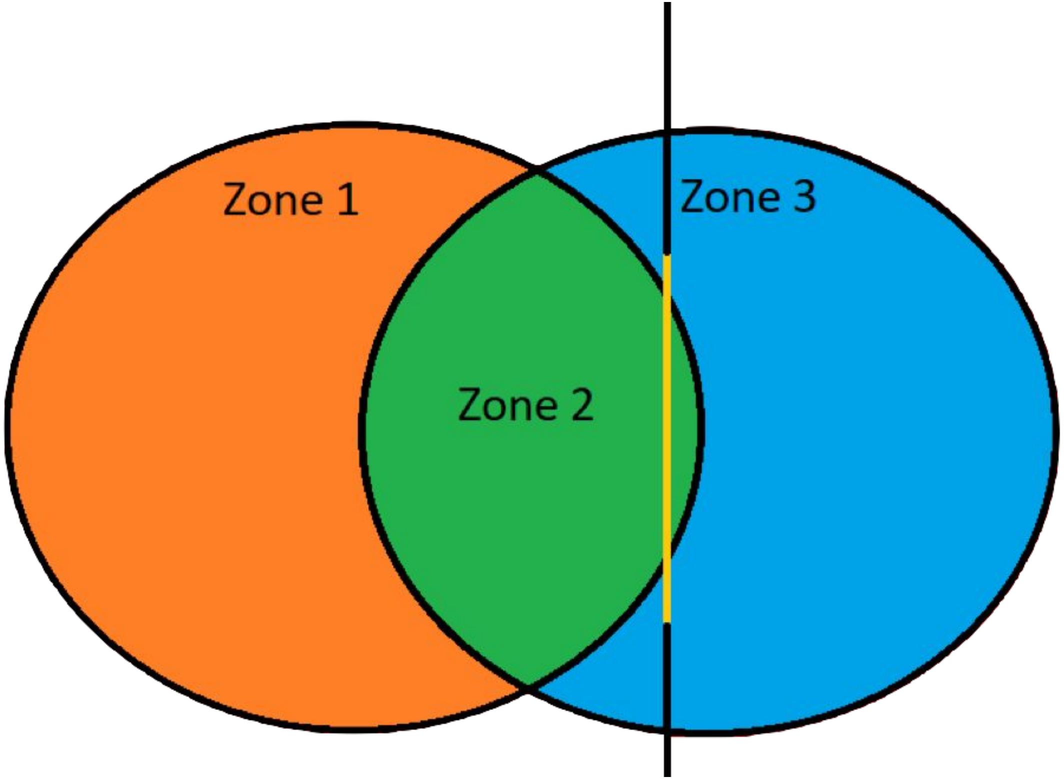
S3 Bucket for Speech
Storage

Store speech file
into S3 bucket

User registration
or updates

Alexa APP

Bluetooth Sensing: Operating Zones



Challenges

- Amazon did not expose Alexa to be a proactive system
    - Instead had to use AWS Polly to generate speech
    - Alexa only responds when spoken to first
    - Other APIs such as notification, alarms require confirmation before being set
- The Raspberry PI does not come with a Class 1 Bluetooth hardware
    - Cannot change TX power through command line so the inner PI has to be positioned strategically
        - If we can change TX power, it would require receiving device to be closer
- Not all edge cases covered
    - User can still have messages played even though not going through door
- RSSI can fluctuate wildly
    - Can use averages but at the expense of responsiveness

Result & Performance

- System works as intended
  - Users’ state are captured by our set of sensor
  - Can speak to users by their names
- Long response time
  - Takes 4-5 seconds for a message played
  - Overhead due to sequential operation of raspberry pi
  - Speech generation takes time
- Unstable RSSI reading
  - RSSI values fluctuating even if user standing in same spot

Improvements

- The time from satisfied condition to message playing is slow
  - Each user’s state is scanned and processed separately
    - Thus more devices requires longer period before it gets scanned again
  - Batch the operation: scan multiple devices and report to server
    - Maybe even parallelize, launch process for each device
  - Take noticeable time to play speech response
    - Simplify the process of getting speech onto the raspberry pi
- Image recognition can replace bluetooth

Relevance

- Alexa wasn’t really needed for the most part
    - Alexa can act as sensor for GPS location, username, etc
        - But not relevant to the project
        - Does not provide the required funicality
    - Create a server on raspberry pi instead of using AWS
        - Essentially combining sensors and server together
- Use AWS for a more generalizable, extendable system
    - Our backend implementation is task specific
    - With modification, it can be used for different sets of sensors

Conclusion

- Discover the limits and capability of Alexa and smart devices
- Started on a system that can keep track of a set of finite states
  - Though there are noticeable performance issues
  - Several steps for improving the performance of the task
- We could modify our implementation to build APIs for creating stateful smart home systems