SDP 18: EfficienSeat

Midway Design Review
5 December 2017

Team 26
Advised by Professor Tilman Wolf
Kristina Georgadarellis, Matthew Donnelly, Dennis Donoghue, Aarsh Jain
Problem Statement

- Dining halls suffer from inefficiencies due to their current method of seating patrons
- A potential solution addresses needs of the patrons as well as the dining hall
  - Patrons need an easy way to identify and locate open seats in dining halls
  - Dining halls need an easily implementable and maintainable system
System Specifications

- Allows app users to search for available seats by party size through app
- Users will receive response from the app within 2 seconds
- The app will generate a depiction of table locations accurate to $\frac{1}{2}$ a table length
- Can function in a dining hall of several hundred seats
- Table unit is compliant with IPX4
- System accommodates non-app using patrons
- Needs to be maintained once a month
Timeline for User Interaction

**App-Users**
At or approaching the dining hall, find seats and reserve.

**Hit button**
Seat is now claimed

**45 minute window**
Seats will stay claimed

**Leaving**
Patron will hit button to leave, seat will become available

**3 minute warning**
Hit button to renew session (unlimited renewals)

**After 45 minutes**
Seats will become available

**No action**
Seats become available after 45 minutes

**15 sec warning**
Prompted to extend reservation by 3 mins (one renewal only)

**After 3 minutes**
Seats will become available

**3 minute window**
Seats will be reserved

Key
- App Specific
- Both users

UMassAmherst Department of Electrical and Computer Engineering
Previous Solution: Block Diagram

Server
- Database
- Data Processing

Sky Unit
- Camera
- Thermal Sensor
- Transmitter/Receiver
- Power Supply
- Data Processing
- Microprocessor

User Interface
- GUI
- Seating Info Feedback
- Search Functionality
- Dining Hall Map

Table Unit
- Table Status Indicator
- Power Supply
- Transmitter/Receiver
- Button

Data Processing

Table Unit

Dining Hall Map
Redesigned Solution: Block Diagram

**Sky Unit**
- Raspberry Pi NoIR Camera Module V2
- Camera Port
- Raspberry Pi 3
- Wall power

**Atmel SAM R21G**

**Table Unit**
- Atmel SAM R21G
- Battery
- LEDs and Buttons
- IR LEDs

**Database Server**
- Amazon Authentication Tools
- Amazon DynamoDB

**Android App**
- AWS Mobile SDK
  - Database connection
  - User Interface

**AWS**
- Boto3 (AWS SDK for Python)
- Atmel Bitcloud (IEEE 802.15.4)
Redesigned Solution: Block Diagram

**Sky Unit**
- Raspberry Pi NoIR Camera Module V2
- Atmel SAM R21G
- Raspberry Pi 3
- Wall power

**Database Server**
- Amazon Authentication Tools
- Amazon DynamoDB

**Android App**
- AWS Mobile SDK
- Database connection
- User Interface

**Table Unit**
- Tasks
  - Physical interface for patrons
  - Relay table status to Sky Unit

**Tasks**
- Physical interface for patrons
- Relay table status to Sky Unit

**Boto3** (AWS SDK for Python)

**Atmel Bitcloud (IEEE 802.15.4)**
Redesigned Solution: Block Diagram

**Tasks**
- Table lateralization
- Communication with Table Unit
- Handles seat claiming requests

**Sky Unit**

**Table Unit**
- Atmel SAM R21G
- IR LEDs
- LEDs and Buttons
- Battery

**Database Server**
- Amazon Authentication Tools
- Amazon DynamoDB

**Android App**
- Database connection
- User Interface

**AWS Mobile SDK**

**Boto3** (AWS SDK for Python)

**Atmel Bitcloud (IEEE 802.15.4)**
Redesigned Solution: Block Diagram

**Task**
- Manages and stores seating information

**Database Server**

**Sky Unit**
- Raspberry Pi NoIR Camera Module V2
- Atmel SAM R21G
- Raspberry Pi 3
- Wall power
- Camera Port
- Serial USB

**Table Unit**
- Atmel SAM R21G
- Battery
- LEDs and Buttons
- IR LEDs

**Android App**
- Database connection
- User Interface

**Boto3 (AWS SDK for Python)**

**AWS Mobile SDK**
Redesigned Solution: Block Diagram

- **Tasks**
  - Implement a table search algorithm
  - Render tables and seats
  - Send seat reservation requests to the Sky Unit

**Sky Unit**
- Raspberry Pi NoIR Camera Module V2
- Wall power
- Camera Port

**Table Unit**
- Atmel SAM R21G
- Battery
- LEDs and Buttons
- IR LEDs

**Database Server**
- Amazon Authentication Tools
- Amazon DynamoDB

**Mobile SDK**
- AWS Mobile SDK
- Boto3 (AWS SDK for Python)

**Android App**
- Tasks
  - Implement a table search algorithm
  - Render tables and seats
  - Send seat reservation requests to the Sky Unit
Table Unit: Choosing the MCU

**AIM:** Low powered, wireless communication between many devices & cheap.

<table>
<thead>
<tr>
<th>Microcontroller</th>
<th>Cost</th>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atmel SAM R21G</td>
<td>$5.2</td>
<td>2.4 GHz on-chip transceiver using IEEE 802.15.4 protocol</td>
<td>Relatively complex functionality</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Low powered</td>
<td></td>
</tr>
<tr>
<td>STMicroelectronics SPWF04</td>
<td>$15.8</td>
<td>High data rate with SoC using IEEE 802.11/b/g/protocol</td>
<td>Relatively power hungry</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Powerful functionality</td>
<td>Expensive for application</td>
</tr>
<tr>
<td>Texas Instruments CC430 w/ MSP430</td>
<td>$15</td>
<td>Simplistic Communication</td>
<td>Expensive for application</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Low Powered</td>
<td>Low support with stacks and protocol</td>
</tr>
<tr>
<td>ON-Semiconductor AX8052F143-3-TB05</td>
<td>$4</td>
<td>Low Powered</td>
<td>Extremely low data-rate and range</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cheapest device available</td>
<td>Low support with stacks and protocol</td>
</tr>
</tbody>
</table>
Table Unit: Atmel SMART SAM R21

Functions of Table Unit

● Connect to Coordinator Node at Sky Unit in a network
● Change state of LED based on user input.
● Send the changed state to Coordinator Node.
● Receive data from Coordinator Node and process it.

Functions of Coordinator

● Network and data management of the Table Units.
**Table Unit: State Diagram**

- **Vacant**
  - LED = ON
  - Chair Button Pressed

- **Unconfirmed**
  - LED = SLOW PULSE
  - Chair Button Pressed
  - Coordinator sends request (if not claimed)

- **Occupied**
  - IF(BUTTON), send CMD
  - LED = OFF
  - Chair Button Pressed

- **Failure**
  - LED = SOS
  - In case of network disconnect

- **Chair Button Pressed**
  - Chair Button Pressed (to claim)
Table/Sky Unit Communication Protocol

Table unit and Sky unit must communicate in order for system to maintain current, working knowledge of seating status

- Table unit must send data concerning:
  - Table ID
    - table unit address
    - seat number
  - State of seat (occupied, vacant)

- Sky unit must send data concerning:
  - Table ID
    - table unit address
    - seat number
  - Command type
    - new seat state, if change is needed
Example

Sky unit communicating to table unit after a seat is reserved at Table 1

Payload: 01 122

<table>
<thead>
<tr>
<th>addr</th>
<th>pin</th>
<th>cmd</th>
<th>data</th>
</tr>
</thead>
<tbody>
<tr>
<td>addr</td>
<td>pin</td>
<td>cmd</td>
<td>data</td>
</tr>
</tbody>
</table>
Database Server

- Started with free mySQL database hosted on Amazon RDS
  - Faced Difficulties with...
    - Secure Pi-database connection
    - Table flexibility
    - App-database connection
- Moved on to DynamoDB
  - In-house, fully integrated Amazon system
  - Manages Users, Roles, and Rules for Security
  - Support for real-time, automatic scaling
  - Well documented and supported

<table>
<thead>
<tr>
<th>Server</th>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>mySQL on Amazon RDS</td>
<td>Simple, Free, and partly integrated with AWS</td>
<td>Harder to access, less scalable, no support</td>
</tr>
<tr>
<td>Amazon DynamoDB</td>
<td>Powerful, flexible, and fully integrated with AWS</td>
<td>Not Free</td>
</tr>
</tbody>
</table>
Phone App: Rendered Map

- “View” frames scale to size of device
- Loaded tables scale to available canvas without distortion
- App “draws” (fetches, filters, and renders) data in under two seconds
- App draws on startup, then refreshes...
  - On party size change
  - On user action (swipe down)
  - Every 30 seconds
- Connection to Database is authenticated by AWS
- Pan, Zoom, and other usability features not yet implemented
Table Localization: Concept

Table Unit position found by:

1. mapping the boundaries of the room
2. Generating a grid
3. Finding table unit on grid

1st Step
Configuration

IR LED in a known location

IR-sensing camera

Sky Unit

2nd Step
Generate grid

3rd Step
Find each table unit position

Table
Table Localization: Components

**Raspberry Pi NoIR Camera V2**
- 8 megapixel resolution
- Blue filter (filter out red and green light)
- Synonymous with a SLR 35mm lens (considered a wide angle lens)

**IR LEDs - 10mm**
- 30-degree beamwidth
- Power consumption: 45mW
- Infrared Ray: 850nm (Near-Infrared Range)
Demo 1: Table Unit Functionality

➔ Push a button on the table unit, LED turns OFF or ON, data gets transmitted and displayed on terminal

➔ Type in a command on the computer, see response on table unit

---

**Sky Unit**

- Raspberry Pi NoIR Camera Module V2
- Raspberry Pi 3
- Wall power

**Table Unit**

- Atmel SAM R21G
- IR LEDs
- LEDs and Buttons
- Camera Port
- Serial USB
- Atmel Bitcloud (IEEE 802.15.4)
- Battery
Table/Sky Unit Communication Protocol

Payload

<table>
<thead>
<tr>
<th>addr</th>
<th>cmd</th>
<th>data</th>
<th>pin</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 bytes</td>
<td>1 byte</td>
<td>1 byte</td>
<td>1 byte</td>
</tr>
</tbody>
</table>

Definitions

<table>
<thead>
<tr>
<th>ID</th>
<th>Command</th>
<th>ID</th>
<th>Data Entry</th>
<th>ID</th>
<th>Pin Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Invalid</td>
<td>0</td>
<td>Vacant</td>
<td>0</td>
<td>Invalid</td>
</tr>
<tr>
<td>1</td>
<td>Change table state</td>
<td>1</td>
<td>Occupied</td>
<td>1</td>
<td>Seat 1</td>
</tr>
<tr>
<td>2</td>
<td>Sleep</td>
<td>2</td>
<td>Reserved</td>
<td>2</td>
<td>Seat 2</td>
</tr>
<tr>
<td>3</td>
<td>Wake up</td>
<td>'3</td>
<td>About to expire</td>
<td>3</td>
<td>Seat 3</td>
</tr>
<tr>
<td>4</td>
<td>Turn IR LED on</td>
<td>4</td>
<td>Sleep</td>
<td>4</td>
<td>Seat 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td>Turn IR LED on</td>
<td>5</td>
<td>All Seats</td>
</tr>
</tbody>
</table>
Demo 2: App and Server Functionality

- Change seating information from the Pi terminal
- See change reflected on app
- Show rendered map with party size filtering

Sky Unit

- Raspberry Pi NoIR Camera Module V2
- Atmel SAM R21G
- Raspberry Pi 3
- Wall power
- Camera Port
- Serial USB

Database Server

- Amazon Authentication Tools
- Amazon DynamoDB

Android App

- AWS Mobile SDK
- Database connection
- User Interface

Boto3 (AWS SDK for Python)
CDR Deliverables

- Demonstrate full system communication
  - Hitting the button on the table unit is reflected on the app and vice versa
  - Raspberry Pi as hub for seat claiming requests
- Prototype of table unit
  - Actual hardware for buttons and LEDs, battery
- Demonstrate table localization ability
  - Create a grid for a rectangular room
  - Determine position of a table unit
Looking Forward

ACTIVITY
Design
Order/ship time
Re-design/order/test
Table unit <-> Sky
Sky <-> Database
Integration

Improve Table Search
Support multiple DBs
Improve & finish GUI
iOS Port
IR Camera Tests
Grid Creation
Identify table location
Identify table rotation
Report

UMass Amherst Department of Electrical and Computer Engineering