Word Document Version Available Here:
https://docs.google.com/document/d/1i-d_tOIEsSdwRATjyTX6yuYu4XKn49gh_DdGzUVQZek/edit?usp=sharing

CAMS OUTSAFE Group

Team: Niranjan Varma, Tyler Greenwood, Gregory Newman

Partners: LaTonya Reeves and Marvin Reid

CAMS Outsafe is a project that is going to improve the safety of members of college campuses across the country. Specifically, we plan on creating a method of notifying students and faculty by pushing notifications from a controller device through a server onto wearable bracelets regarding the safety status of the campus.

Table of Contents

Software Development Process (SDP) **7**
- **Principles** **7**
- **Process** **7**
- **Roles** **7**
- **Tooling** **8**
- **Definition of Done (DoD)** **9**
- **Release Cycle** **9**
- **Environments** **9**

Software Design and Architecture **10**
- **Introduction** **10**
- **Architectural Goals and Principles** **10**
- **System Overview** **11**
- **Architectural Patterns** **12**
- **Component Descriptions** **12**
- **Data Management** **13**
- **Interface Design** **13**
- **Considerations** **14**
  - Security 14
  - Performance 14
  - Maintenance and Support 14
- **Deployment Strategy** **14**
- **Testing Strategy** **15**
- **Glossary** **15**

Product Requirements Document (PRD) **16**
- **Problem Description** **16**
- Scope 17
- Use Cases 17
- **Purpose and Vision (Background)** **17**
- **Stakeholders** **18**
- **Preliminary Context** **18**
  - Assumptions 18
  - Constraints 18
  - Dependencies 18
- **Market Assessment and Competition Analysis** **19**
- **Target Demographics (User Persona)** **19**
- **Requirements** **20**
  - User Stories and Features (Functional Requirements) 20
  - Non-Functional Requirements 21

Data Requirements....21
Integration Requirements....21
User Interaction and Design....22
Milestones and Timeline....23
Milestones:....23
Full Timeline:....24
Goals and Success Metrics....24
Open Questions....25
Out of Scope....25
Progress Reports....26
CAMS Fall Week 3 Report....26
CAMS Fall Week 4 Report....27
Summary....27
Issues Completed....27
New Issues....27
Backlog Progress....28
Major Changes in Assets....28
CAMS Fall Week 5 Report....29
Summary....29
Issues Completed....29
New Issues....29
Backlog Progress....30
Major Changes in Assets....30
CAMS Fall Week 6 Report....31
Summary....31
Issues Completed....31
New Issues....31
Backlog Progress....32
Major Changes in Assets....32
CAMS Fall Week 7 Report....33
Summary....33
Issues Completed....33
New Issues....33
Backlog Progress....34
Major Changes in Assets....34
CAMS Fall Week 8 Report....35
Summary....35
Issues Completed....35
New Issues....35
Backlog Progress....36
Major Changes in Assets....36

CAMS Fall Week 9 Report....37
Summary....37
Issues Completed....37
New Issues....37
Backlog Progress....37
Major Changes in Assets....38
CAMS Fall Week 10 Report....39
Summary....39
Issues Completed....39
New Issues....39
Backlog Progress....39
Major Changes in Assets....40
CAMS Winter Week 1 Report....41
Summary....41
Issues Completed....41
New Issues....41
Backlog Progress....41
Major Changes in Assets....41
CAMS Winter Week 2 Report....42
Summary....42
Issues Completed....42
New Issues....42
Backlog Progress....42
CAMS Winter Week 3 Report....43
Summary....43
Issues Completed....43
New Issues....43
Backlog Progress....43
Major Changes in Assets....43
CAMS Winter Week 4 Report....44
Summary....44
Issues Completed....44
New Issues....44
Backlog Progress....44
Major Changes in Assets....44
CAMS Winter Week 5 Report....45
Summary....45
Issues Completed....45
New Issues....45
Backlog Progress....45
Major Changes in Assets....46

CAMS Winter Week 6 Report..... 47
Summary.....47
Issues Completed..... 47
New Issues..... 47
Backlog Progress..... 47
Major Changes in Assets..... 48
CAMS Winter Week 7 Report..... 49
Summary.....49
Issues Completed..... 49
New Issues..... 49
Backlog Progress..... 49
Major Changes in Assets..... 50
CAMS Winter Week 8 Report..... 51
Summary.....51
Issues Completed..... 51
New Issues..... 51
Backlog Progress..... 52
Major Changes in Assets..... 52
CAMS Spring Week 9 Report..... 53
Summary.....53
Issues Completed..... 53
New Issues..... 53
Backlog Progress..... 54
Major Changes in Assets..... 54
CAMS Spring Week 2 Report..... 55
Summary.....55
Issues Completed..... 55
New Issues..... 55
Backlog Progress..... 55
Major Changes in Assets..... 55
CAMS Spring Week 3 Report..... 56
Summary.....56
Issues Completed..... 56
New Issues..... 56
Backlog Progress..... 56
Major Changes in Assets..... 57
CAMS Spring Week 4 Report..... 58
Summary.....58
Issues Completed..... 58
New Issues..... 58
Backlog Progress..... 59

Major Changes in Assets **59**
CAMS Spring Week 5 Report 60
Summary **60**
Issues Completed **60**
New Issues **61**
Backlog Progress **61**
Major Changes in Assets **61**
CAMS Spring Week 6 Report 62
Summary **62**
Issues Completed **62**
New Issues **62**
Backlog Progress **62**
Major Changes in Assets **62**
CAMS Spring Week 7 Report 63
Summary **63**
Issues Completed **63**
New Issues **63**
Backlog Progress **63**
Major Changes in Assets **64**

Software Development Process (SDP)

[Principles](#)
[Process](#)
[Roles](#)
[Tooling](#)
[Definition of Done (DoD)](#)
[Release Cycle](#)
[Environments](#)

Principles

- Since work will be done asynchronously, group members must respond to all communications within a day.
- Work will be organized into a backlog, keeping track of in-progress and completed tasks.
- Sprints will be used as needed once the development process begins.
- Regular meetings and check-ins on particular tasks will be used to make sure we are staying on top of the backlog.
- When making a major update to the main repository, all team members must review (we are only 3 people)
- The Definition of Done will be followed for completing major tasks
- One team member will lead the progress report every week.
- Show up to the major weekly meetings

Process

We are all on our schedules so we plan on working asynchronously. We plan on meeting twice a week, once with our partner, and once with the TA. After each meeting, the three of us will stay on the call to discuss in-progress items or tasks to be completed. We all have each other's contact information, so we always have the option to meet very urgently/ inform the team about important updates. We will all contribute to the weekly progress report as well, with a different member taking the lead on them each week so the responsibility is distributed. The approach to development we are taking is a mixture of Kanban and Agile. Our objectives are very clearly defined and we will make decisions on the implementation specifics well before development begins. However, our team has little experience developing this kind of technology, so it might not be possible to accurately predict the amount of time a task will take. Therefore, an Agile approach to development will be suited to our project with diligent tracking of the backlog and in-progress items.

Roles

Niranjan will take charge of managing the backlog and making sure members are responsive and present at meetings. Tyler will take charge of the engineering and programming side of the project. Gregory will take charge of the design and research portion of the project.

All group members contribute to the weekly progress report, but we will have a rotating schedule for who will take the lead and submit the report each week.

When it comes to being present at meetings, it should be the main priority in terms of academic activities for the designated time. Exceptions can be made if there are personal or logistical reasons. Updates have to be given to other members of the group if they are a no-show.

Tooling

Version Control	GitHub
Project Management	GitHub Issues and Projects
Documentation	Google Docs and GitHub
Test Framework	Jest
Linting and Formatting	Prettier
CI/CD	GitHub Actions
IDE	Visual Studio Code, IntelliJ IDEA, Arduino Studio
Graphic Design	Pen and paper, draw.io
Others	Hardware(wristbands and controller)

When writing the document we were still not sure certain software regarding how we were going to program the hardware devices and connect between the controllers and the receivers using certain protocols. IntelliJ has been helpful for the controller software and sending messages to our central server. Arduino studio has helped program the receiver UI and connecting to the server to receive messages.

Definition of Done (DoD)

- The wants and needs of the partner are met
- Updates are integrated into the main Git repository
- No bugs or deterioration of the code
- Documentation is updated with a new task
- Inform partner and TA about the completion
- If possible demo/present what the new update helps with

Release Cycle

- Automatically deploy to staging every new update to the main repository
- Deploy to production every release, pushing updates to the hardware.
- Only make a version of the program once it is stable
- Categorize versions as minor touch-ups (minor formatting/bug fixes), major (new features), and breakthroughs (API changes)

Environments

Environment	Infrastructure	Deployment	What is it for?	Monitoring
Production	Render through Github Actions	Release	Make an actual functioning product that is usable for customers.	Sentry
Staging (Test)	Local(On the wristband itself)	PR	Testing the integration of new features into present technology.	N/A(since testing is being done on hardware)
Dev	Local	Commit	Developing and testing	N/A

Software Design and Architecture

CAMS Development Team

Tyler Greenwood, Gregory Newman, Niranjan Varma

Introduction
Architectural Goals and Principles
System Overview
Architectural Patterns
Component Descriptions
Data Management
Interface Design
Considerations
    Security
    Performance
    Maintenance and Support
Deployment Strategy
Testing Strategy
Glossary

Introduction

This document details the architecture for the OutSafe Campus Alert Management System. The main function of the system is to create a set of controllers that can send alerts and messages to a network of Wi-Fi bracelets over a network. A central server will be used to organize communication between the controllers and bracelets. A well-designed architecture will provide a system that is streamlined and efficient, not putting excessive loads on the network that they are connected to. It will also be reasonably secure so that messages cannot be intercepted or falsely transmitted between the devices.

When we initially had the idea for the project we assumed that it would be possible to have a central controller that would be able to directly connect with multiple bracelets through a wi-fi network. After more research, it became clear that a server would make the process of communication a lot easier for our project. Thus why we decided to include a server into our main components.

Architectural Goals and Principles

- Security
    - We need to only allow the controller to talk to the receivers
    - Accompanying apps should talk to the controller, not to the receivers directly

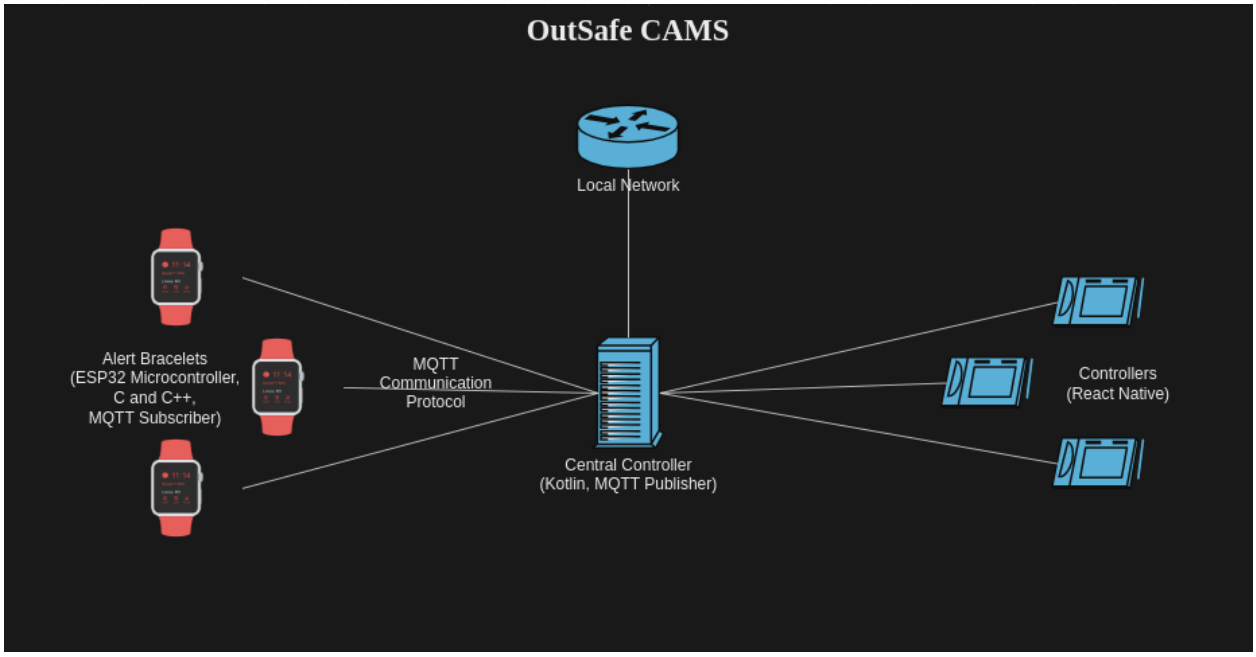
    - Ensure safe alert cannot be sent by bad actor
    - Auth keys should not be reverse engineerable
- Reliability
    - We need the receivers to receive messages from the controller 100% of the time when in range/on the same network (depends on the communication method)
    - Battery needs to be able to last all day (charge at night), ideally more than that. Software cannot drain the battery more than this.
    - Receivers should display a message within 0.5 seconds of the controller sending it.
    - Controller should turn its display on within 200ms of being turned on
    - Controller should be almost instantly responsive to interaction from user
    - Controller should send a message within 200ms of the execute button being hit

System Overview

To maintain reliability, the system will be very simple. It will only be a main controller that can send messages to receivers. The receivers are able to receive messages and display them, but not send out any information. The controllers will use a central controller/server to facilitate communication to the bracelets. This is using a local network which both the controller and receivers will be connected to.

All components of the system will exist on the same LAN using an Ethernet or Wi-Fi connection. Communication between the controllers and the central controller will be done through HTTP. Communication between the central controller and the bracelets will be done through the MQTT protocol, a lightweight publisher/subscriber protocol often used in Internet of Things (IoT) devices.

When coming across the programming of the software we were trying to find a viable solution for communication between the controllers and the receiver we thought about HTTP protocol. Specifically, we found that MQTT would be the most viable solution for a library to use for both the controller and receiver software for communication to the server.



This model is a great representation of the project architecture and goes into good detail about each component of the project and all the general software, programming languages, and protocols used.

Architectural Patterns

Our implementation uses a type of Publisher/Subscriber pattern. The central controller(publisher) has the job of relaying data to all the bracelets (subscribers), and the bracelets must simply respond in a predetermined way to the data they are being sent. The controllers will connect to the central controller and the controller and will facilitate the pushing of messages in a 1-way communication, performing any translation needed between the various languages and protocols being used.
This update to our documentation was necessary to clarify how the program was going to work with the addition of a Central controller(server).

Component Descriptions

Our implementation will consist of these components:

- **Alert Bracelets** - These will be lightweight, portable platforms equipped with Wi-Fi capabilities and a vibration motor. They will be built on the Arduino ESP-32 platform.
- **Controllers** - This will be a touchscreen-equipped device with a simple user interface. Connected to the same network as the bracelets, they will be able to send messages to the bracelets over Wi-Fi.
- **Central Controller** - This will be a lightweight REST API that can receive information from controllers and send them to the watches. Written in Kotlin.
  The formal terminology we have kept for the server is to name it the Central Controller since it controls the communication protocols for various controllers and recievers.

- **Database** - This will be a small database with the job of storing the MAC addresses and information associated with each bracelet, such as its owner, as well as keeping logs of device usage.

Data Management

Database Entities:

Bracelet (Each bracelet)
- bracelet_ID: int, auto-increment, unique, not NULL
- MAC_address: varchar, unique, not NULL
- first_name: varchar, not NULL
- last_name: varchar, not NULL
- Room_number: varchar, NULL

Log_Day (Entity to track all log entries from a given day)
- log_day_ID: int, auto-increment, unique, not NULL
- date: date, not NULL

Log_Entry (Each log entry)
- log_entry_ID: int, auto-increment, unique, not NULL
- log_day_ID: int, not NULL, FK from Log_Day
- entry_type: enum, varchar, not NULL
- bracelet_ID: int, NULL, FK from Bracelet
- time: time, not NULL
- Log_text: varchar, not NULL

Interface Design

Central Controller:
- Main screen:
    - Green button (all clear)
    - Blue button (drill)
    - Yellow button (shelter in place)
    - Red button (exit)
    - Gray button (send alert to all bracelets)
- Second screen:
    - Keyboard to type a message
    - Gray button to send message to all bracelets

Bracelets
- LED Screen with the ability to display alert levels
- Vibration chip
-

We removed a functionality we had previously which was to allow the wristbands to communicate with one another. However, after consulting with our project partners we realized that it would complicate the safety process of having other wristbands change the safety status. We want the controllers to be able to send messages and the receiver bracelets that are subscribed to display the status.

Considerations

Security

Having receivers constantly waiting for communication means they are open to receiving messages from a potential bad actor. Due to the nature of the device, someone could cause panic by sending a signal to the receivers when there is no emergency. To mitigate this risk, we will need to use some form of encryption to make sure that bad actors cannot send signals to cause a false alarm. The other security concern would be in an emergency situation, if a bad actor got ahold of the controller, they could send a false “Safe” message. We haven’t discussed this yet, but a potential solution would be to require a pin to send a “Safe” message when an emergency state has been declared.

Performance

The performance requirements of the product were declared prior in this document, but the response time of the controller must be fast, and the communication between the controller and receiver needs to be fast. Keeping the code simple will help with performance, but making sure the communication method between the controller and the receiver is fast enough will be an important aspect of that decision. The other will be to make sure the framework we use for the UIs will be able to paint the screens quickly.

Maintenance and Support

The ability to update hardware using an OTA system would be great, but is not our primary objective. In the event of a needed update, we will need to at least ensure that the hardware can be is updated fairly easily by a user, even if they need to plug it into a computer to update it. After the completed prototype is delivered to the project partner, they will be responsible for the next development steps and continued product support. They will be given full access to the codebase and will become its custodians from that point.

Deployment Strategy

A working prototype of both the central controller and the bracelet will be maintained which are programmed with the highest stable build of the software, acting as the production environment. These prototypes will only be updated with new software versions once a new stable release is prepared.

Deployment for our software will primarily be manual, and we will update our prototypes accordingly. If we are able to develop an OTA update process, we can do scheduled releases for updates, or hotfixes when needed, depending on what work needs to be done. A companion app could follow a monthly release schedule if it is developed.

Testing Strategy

The only method to test our technology is directly implementing the software into the wristband. No amount of simulations will get close to the analysis of communication to the wristbands. So all tests such as unit, integration, and other larger tests will only be done using the wristbands. Thus we will have to program our potentially new software onto one wristband to see how it responds.

In terms of environments, we are concerned about whether users will be able to receive notifications to their wristbands in particular locations. This is so that we could identify locations on campus that could become blindspots in terms of not being able to connect to a particular Wi-Fi network.

While manual testing will be a big part of this project, as we are working with hardware. Outside of this, we can test our functions automatically to ensure that given a specific input, we receive our expected output. We could consider an automated testing strategy for the hardware while it is plugged into a computer that could monitor the messages sent and received by the controller and receivers to make sure that each feature is working as expected.

Glossary

- OTA: Over the air, in reference to being able to update hardware automatically using an internet connection.
- Bad Actor: Someone intending to use our system for malicious purposes
- REST API: Representational state transfer application programming interface. A URL that can be accessed to get, update, create, or delete data.

Just an extra glossary definition for the updated SDA

Product Requirements Document (PRD)

Campus Alert Management System (CAMS)
Development Team: Tyler Greenwood, Gregory Newman, Niranjan Varma

Contents

[Problem Description](#)
- [Scope](#)
- [Use Cases](#)

[Purpose and Vision (Background)](#)
[Stakeholders](#)
[Preliminary Context](#)
- [Assumptions](#)
- [Constraints](#)
- [Dependencies](#)

[Market Assessment and Competition Analysis](#)
[Target Demographics (User Persona)](#)
[Requirements](#)
- [User Stories and Features (Functional Requirements)](#)
- [Non-Functional Requirements](#)
- [Data Requirements](#)
- [Integration Requirements](#)
- [User Interaction and Design](#)

[Milestones and Timeline](#)
[Goals and Success Metrics](#)
[Open Questions](#)
[Out of Scope](#)

Problem Description

Mass shootings and mass casualty events continue to be a problem in the United States, with schools being unfortunate and frequent targets. When one of these events occurs, evacuations and lockdowns are frequently disorganized as it is difficult to spread information during the chaos of these tragedies.

Students at these institutions value their safety and want to know that they will be accurately informed with clear directions on whether to flee, lockdown, or perform some other action in the event of such a tragedy.

Administrators and school workers prioritize the safety of their students and need a way to send out accurate information to everyone on their campuses even if they are unable to reach the building PA system or send out a mass message.

Police officers and security professionals also need to know accurate information during a mass shooting/MCI. Knowing that an entire campus was able to be warned and that specific instructions on what to do were given using the OutSafe system gives first responders critical information and can help them determine the predicted locations of students and staff as well as potential locations for a suspect.

Scope

To develop the full scope of this project, more discussion is needed. Based on the first documents sent by the stakeholders, the first goal of this project is to develop a primary controller that has two possible displays. The first has five buttons, Safe, Drill, SIP (Shelter in Place), Exit, and an execute button to confirm the selected message should be sent. The second screen would have a keyboard to send messages from, and be accessible by swiping from the first screen. The bracelets should receive any message sent from the controller and display them.

Use Cases

- Each button should send the corresponding message to all receivers that are connected to the controller.

- A typed message should send the typed message to all receivers that are connected to the controller.

- Any receivers not connected to a controller should not receive the message.

Purpose and Vision (Background)

There have been over 300 events defined as school shootings in 2023, resulting in the injury and death of many students. While these were all very different events, the chaos, panic, and fear experienced by those present is a shared characteristic of everyone. Forming a unified response across an entire campus is nearly impossible in these situations. In the best possible scenario, a staff member may be able to make an announcement over the PA system, directing all campus occupants to lock down or evacuate. However, there currently exists no reliable method of communication between staff and other building occupants during these tragedies, leading to more loss of life than might otherwise occur.

The Campus Alert Management System (“CAMS”) was first theorized after the Marjorie Stoneman Douglas HS shooting in 2018. Its purpose is to significantly reduce the amount of

damage that can be caused by those who wish to commit atrocities against students in schools. This will be accomplished through a network of wearable alert bracelets, capable of receiving both alerts and messages from one or several controllers located throughout a campus. This system will allow school staff and law enforcement to quickly and effectively distribute information to everyone on campus during an emergency. By diminishing the difficulty of providing reliable information to building occupants during an emergency, the plans of malicious individuals will be more effectively thwarted and many lives will be saved.

Stakeholders

- Marvin Reid is the founder of the product, and the person responsible for making decisions. We have standup meetings with Marvin and the Outsafe team every Monday evening to discuss our progress in the project and also be able to ask them any questions/suggestions regarding certain requirements.
- Gregory, Tyler, and Niranjan, the OSU team assigned to this project, who are the development team for this project and will provide updates for the management staff
- Other employees of CAMS, who will occasionally join our meetings for updates on the status of the project

Preliminary Context

Assumptions

- The hardware has already been acquired and is able to be programmed. Specifically, they are Squarofumi Watchy esp32 watches.

- The hardware has some method of interfacing with each other.

- We have until Spring 2024 to create a functional prototype, not including the companion app, if it takes us that long.

Constraints

- Development is limited to using existing hardware, until the time that proprietary devices are designed specifically for this product.

- The prototype system must be complete by Spring 2024.

- The program size likely cannot be too large due to constraints on small hardware devices.

Dependencies

- The language we develop the software will depend on the capabilities of the hardware.

- We need to have the hardware in order to begin development and testing. Since there is no simulation software that is able to replicate the mechanisms of

- Are there existing frameworks and libraries that can handle some of the implementations for us? Such as for communication between devices or reading touch screen inputs.

Market Assessment and Competition Analysis

There are some companies who have also thought about emergency communication solutions, with an emphasis on being for schools. One example of this is Alertus, which provides some alert devices, as well as software that interfaces with mobile phones, school landlines, and other technology. Another company is Apptegy, which focuses primarily on mobile alerts. The big difference between these other services and CAMS is that they don't put as much emphasis on controlling the spread of the information, meaning it could be harder to ensure critical information gets to the right people, and making sure that important information doesn't end up with the wrong people. Also, these solutions have a large emphasis on mobile devices, which can take a lot of the potential reliability and expected use cases out of the company's control.

Looking at the current situation of alert systems that are there for controlling the spread of misinformation, there isn't really an application out there. Places such as Oregon State University send alerts to students and faculty by notifying them of their University email. The problem is these email alerts have the same priority as academic notifications(canvas). Another issue is people don't immediately check their emails or have their notifications on for when they immediately get a notification.

Target Demographics (User Persona)

The primary users of this technology are staff at a school or other venue. The highest-ranking supervisor would likely be the one with the primary controller, while other staff would have the bracelets.

Requirements

User Stories and Features (Functional Requirements)

User Story	Feature	Priority	GitHub Issue	Dependencies
As a Principal, I want to be able to send out any alert, (Drill, SIP, Exit, custom message).	Controller Screen UI	Must Have	7 and 12	N/A
As a staff member, I want to receive any alerts that are sent out.	Wristband UI	Must Have	21 and 22	N/A
As campus security staff, I want alerts to be sent out securely so that only authorized users can see the transmissions.	Secure data transmission	Must Have	8	Functional transmissions between the controller and bracelets
As a staff member, I only want to receive alerts from the proper controller or app.	Wristband UI	Must Have	21 and 22	N/A
As a controller user, if a receiver doesn't receive my alert, I want the controller to resend the alert.	Controller Screen UI	Should Have	28	Finish creating Controller UI before looking into this
As a staff member, I want to have the receiver alert me in some way so that I don't miss an emergency message.	Wristband UI	Should Have	18	N/A
As a user who owns a mobile device, I want there to be an accompanying phone app for the bracelets so I can see alerts and updates on my phone as well.	Phone app	Might Have	TBD	Fully functional controller and bracelets

We labeled all of the user stories and features in the table with their relative Github issue numbers in our repository.

Non-Functional Requirements

- The receiver should receive any message from the controller within 5 seconds.
- The controller should respond to any interaction by the user within 100 milliseconds.
- Communications between the management system and bracelets should be sent securely, not using plain text.
- Clear documentation should be maintained in a shared file among all team members.

Data Requirements

- The server will keep a record of each bracelet's MAC address and the name of the user that MAC address is associated with. The controller will communicate with the server.
- The controller will need to accept inputs from the user to send out, and potentially store them to remember what state the controller currently is in (Ex: Safe, Drill, etc.).
- The receiver will need to be able to read data from the controller and store it temporarily so it can display continuously.

Integration Requirements

The software we write will need to integrate with the hardware, and the hardware might have certain functions built into it, or certain requirements to meet.

User Story	Feature	Priority	GitHub Issue	Dependencies
As a Principal, I want to be able to send out any alert, (Drill, SIP, Exit, custom message).	Controller Screen UI	Must Have	7 and 12	N/A
As a staff member, I want to receive any alerts that are sent out.	Wristband UI	Must Have	21 and 22	N/A
As campus security staff, I want alerts to be sent out securely so that only authorized users can see the transmissions.	Secure data transmission	Must Have	8	Functional transmissions between the controller and bracelets
As a staff member, I only want to receive alerts from the proper controller or app.	Wristband UI	Must Have	21 and 22	N/A
As a controller user, if a receiver doesn't receive my alert, I want the controller to resend the alert.	Controller Screen UI	Should Have	28	Finish creating Controller UI before looking into this

As a staff member, I want to have the receiver alert me in some way so that I don't miss an emergency message.	Wristband UI	Should Have	18	N/A
As a user who owns a mobile device, I want there to be an accompanying phone app for the bracelets so I can see alerts and updates on my phone as well.	Phone app	Might Have	TBD	Fully functional controller and bracelets

User Interaction and Design

Design Principles:

- User interface should be simple and as easy-to-use as possible.
- The interface should be intuitive, and not require any training for a user to use successfully.
- Simplicity and intuitiveness are key since this device is intended to be used in a high-stress, emergency situation.
- Buttons and keys should be large, obvious, and easy to press

Mockups provided by Marvin Reid.

Milestones and Timeline

Milestones:

- A functioning controller interface, with both screens and buttons the user can press.
- The controller can successfully send out messages, depending on a functioning controller interface.
- The receiver can successfully display messages.

- The receiver can successfully receive messages from a controller, depending on a functioning controller and a functioning receiver display.
- An app that can send alert messages to the control panel and receive alert messages, depending on functional receivers and controllers.

Full Timeline:

Phase 1: Source and acquire the hardware platform for the project
- *Completed when:* Hardware has been ordered and acquired
- *Complete by:* November 26th, 2023

Phase 2: Development of bracelets
- *Completed when:* Bracelets can receive and display simple alerts over the network
- *Complete by:* March 4th, 2024

  A little bit of fixing of the timeline for our project

Phase 3: Development of controller
- *Completed when:* Controller user interface is fully implemented
- *Complete by:* February 1st, 2024

Phase 4: Communication between controller and bracelets
- *Completed when:* Controller can securely send proper messages to all of the bracelets
- *Complete by:* March 31st, 2024

Phase 5: Bug fixes and final release
- *Completed when:* Prototype is fully-functional and polished.
- *Complete by:* June 1st, 2024

Goals and Success Metrics

Goal	Metric	Baseline	Target	Tracking Method
Successful drill using CAMS	All bracelets successfully receive the message	0%	90%	Verifying the communication was sent to all the bracelets.
User satisfaction	Do you feel that this is a helpful tool for safety?	Yes = 0%	Yes > 50%	Interview

We wanted to fix up our goal metrics a little bit for the project to be more in-line with what it looks like now rather than 3 months ago

Open Questions

- There are no open questions at the time of writing.

Out of Scope

- Full distribution of the product.
- Companion app (possible, but not first priority).
- Other devices to interface with the controller.

Progress Reports

CAMS Fall Week 3 Report

Author: Gregory

Last week

- We had our first meeting with our TA and discussed expectations for the project

This week

- Had our first meeting with the project partner on Monday
- Had our second meeting with our TA on Wednesday
- Did preliminary research into possible hardware and software solutions for the project
- Began work on our group's PRD

Next week

- Meet with the partner again to discuss our proposed solutions and gain additional clarity
- Decide on some implementation specifics such as the networking method for the devices (radio, Wi-Fi, or cellular)

Completed this week:

- Found a project with source code for a wireless bracelet and server system that can be integrated into our project: https://openhardware.metajnl.com/articles/10.5334/joh.17
- Found possible hardware platforms: https://diyusthad.com/2020/03/top-4-hackable-smart-watch.html
- Created GitHub repository, backlog, and board for our project
- Created Project Requirements Document
- From our project partner meeting, we learned the following:
- Alerting parents is not part of the design (i.e. system should be totally self-contained to the school and its occupants)
- Partner is open to a variety of implementation specifics, and is more concerned about the big-picture functionality of the system.

Tyler Greenwood
Gregory Newman
Niranjan Varma

CAMS Fall Week 4 Report

Summary
Issues Completed
New Issues
Backlog Progress
Major Changes in Assets

Summary

This week is primarily focused on more research and getting on the same page as our project partner. We are waiting on acquiring hardware for us to be able to start implementing prototype software to turn into a product. We have begun to build out a backlog in GitHub, and are working on non-development related tasks.

Issues Completed

4) Research hardware options
- Whole group
- With the addition of the ECE team and the change in direction of controller implementation, we have closed this as not planned.

10) Determine hardware/software implementation of controller
- Whole group
- We have opted to develop a React Native application that is portable to many types of hardware as a current solution, further decisions on controller hardware can be made down the line.

New Issues

11) Create React Native project
- Instantiate the underlying architecture needed for a React Native project
- This is important because creating the basic files needed for a project, as well as documentation as to how to interact with these files will allow us to move along in our project.

12) Create Controller Screen 2 UI
- Create the keyboard, input box, and other functionality needed to implement the screen 2 UI that can send custom alert messages.
- We created a ticket for Screen 1, so creating one for Screen 2 was needed.

Backlog Progress

11) Create React Native project
- Instantiate the underlying architecture needed for a React Native project
- Next up
- Whole group

7) Create Controller Screen 1 UI
- Tyler Greenwood
- Next up

12) Create Controller Screen 2 UI
- Next up
- Gregory Newman, Tyler Greenwood, Niranjan Varma

Major Changes in Assets

There were not many changes this week. However, we have altered our plan to begin developing controller software without having hardware yet, altering our solution to be more versatile across different pieces of hardware.

Tyler Greenwood
Gregory Newman
Niranjan Varma

CAMS Fall Week 5 Report

[Summary](#)
[Issues Completed](#)
[New Issues](#)
[Backlog Progress](#)
[Major Changes in Assets](#)

Summary

This week was about continuing our research into the software and understanding the hardware that we are going to receive to work on for the project. We have added to the backlog in terms of doing certain tasks regarding user stories and reviewing one another's Software Development Architecture documents. Another task is communicating with our project partner about receiving our hardware. Finally, in our meeting with our partner we had further clarifications on particular user stories to make sure we were all on the same page.

Issues Completed

3) Compare implementation options for communicating between controller and receiver
- Gregory Newman, Tyler Greenwood, Niranjan Varma

5) Determine hardware platform for bracelets
- Gregory Newman, Tyler Greenwood, Niranjan Varma

New Issues

5) Determine hardware platform for bracelets
- We need to explore options on what type of hardware to use for implementation. Mainly looking for hardware that resembles something of a fitbit watch.

6) Email project partner to purchase hardware
- This issue is necessary when it comes to being able to test our code and software. We need to get our hands on a device to program it to work how we desire it.

7) Create Controller Screen 1 UI
- This is one of the user stories we are using to work off of for the project. Specifically in regards to developing the controller and how it communicates with the wristbands. We are only in the beginning stages of these.

Backlog Progress

4) Research hardware options
- Gregory Newman, Tyler Greenwood, Niranjan Varma
- In Review

2) Complete Software Design and Architecture document
- Gregory Newman, Tyler Greenwood, Niranjan Varma
- In Progress

6) Email project partner to purchase hardware
- Gregory Newman, Tyler Greenwood, Niranjan Varma
- In Progress

7) Create Controller Screen 1 UI
- Tyler Greenwood
- Next Up

Major Changes in Assets

There have been additions to the repository with the completion of user stories from this week. As these user stories have been completed, there will be more additions of them to the repository in the future. There have not yet been any changes to other documents.

Tyler Greenwood
Gregory Newman
Niranjan Varma

CAMS Fall Week 6 Report

Summary

This week, an order of 3 Squarofumi Watchy esp32 watches was placed by the project partner, which will serve as the testbed for the bracelets. We also met with the project partner to gain additional insight into the hardware requirements which will allow us to research possible platforms for the controller. We were also informed that there is an ECE team working on this project in parallel, who may be providing us with hardware to develop on.

This coming week, we will meet with the project partner again to discuss our proposed solution for the controller (an Android application), we will plan a meeting with the ECE team to make sure that our software will be compatible with whatever hardware they are developing, and we will await the delivery of the bracelets to begin development.

Issues Completed

6) Email project partner to purchase hardware
- The project partner placed an order for 3 bracelets this week, which will be delivered in the coming week.
- Whole group

5) Determine hardware platform for bracelets
- Bracelets will be developed on the Squarofumi Watchy.
- Whole group

New Issues

10) Determine hardware/software implementation of controller
- We must decide whether the controller will be developed directly on hardware or through an app-based deployment. This question will be presented to the partner this week.
- Whole Group

Backlog Progress

4) Research hardware options
- Although we determined the platform for the bracelet, research continues for the controller.
- Gregory Newman, Tyler Greenwood, Niranjan Varma
- In Progress

2) Complete Software Design and Architecture document
- Gregory Newman, Tyler Greenwood, Niranjan Varma
- In Progress

6) Email project partner to purchase hardware
- Gregory Newman, Tyler Greenwood, Niranjan Varma
- In Progress

7) Create Controller Screen 1 UI
- Tyler Greenwood
- Next Up

Major Changes in Assets

There were no major changes to the documents this week, but we continue to make minor refinements as our conceptions of the project change with every partner meeting.

Tyler Greenwood
Gregory Newman
Niranjan Varma

CAMS Fall Week 7 Report

[Summary](#)
[Issues Completed](#)
[New Issues](#)
[Backlog Progress](#)
[Major Changes in Assets](#)

Summary

This week, an order of 3 Squarofumi Watchy esp32 watches was placed by the project partner, which will serve as the testbed for the bracelets. We also met with the project partner to gain additional insight into the hardware requirements which will allow us to research possible platforms for the controller. We were also informed that there is an ECE team working on this project in parallel, who may be providing us with hardware to develop on.

This coming week, we will meet with the project partner again to discuss our proposed solution for the controller (an Android application), we will plan a meeting with the ECE team to make sure that our software will be compatible with whatever hardware they are developing, and we will await the delivery of the bracelets to begin development.

Issues Completed

- #2) Complete Software Design and Architecture document
    - Gregory Newman, Tyler Greenwood, Niranjan Varma

New Issues

- #2) Complete Software Design and Architecture document
    - We each need to complete this document to further our understanding of the project.
- #3) Compare implementation options for communicating between controller and receiver
    - We need to explore some options before settling on a final implementation, factoring in things like cost, performance, reliability, and security.
- #4) Research hardware options
    - After figuring out a few more implementation details, we will need to settle on a hardware option for the project.

Backlog Progress

- #3) Compare implementation options for communicating between controller and receiver
    - Tyler Greenwood
    - In-Progress
- #4) Research hardware options
    - Developer TBD
    - Next-up

Major Changes in Assets

The Software Design and Architecture document was completed this week, there have not yet been any changes to the other documents. Any changes will be recorded for future weeks.

Tyler Greenwood
Gregory Newman
Niranjan Varma

CAMS Fall Week 8 Report

Summary

This week we mainly laid out the groundwork for setting up our project in React Native. We have added some code as well that has been added in the form of a pull request, mainly regarding the basic functionalities of the controller software displaying the 4 different options: Drill, SIP, Exit, and Safe.

This coming week we plan on sharing the progress in terms of the coding so far. We plan on merging the code into our repository once we discuss it with our project partners as well. We also anticipate receiving our wearable wristbands to be delivered so we are able to have an opportunity to test our receiving software for the security system.

Issues Completed

11) Create React Native project
- Whole group
- We now have a project that we can alter the code for in React Native thus laying the foundation for the beginning of coding.

New Issues

No new issues were created this week.

Backlog Progress

7) Create Controller Screen 1 UI
- Tyler Greenwood
- In Progress

12) Create Controller Screen 2 UI
- Whole group
- Next Up

Major Changes in Assets

There were no major changes to the documents this week. However, we are coming to a collective consensus on our Product Requirements Document. The major change came in terms of our code. We finally have some code that we added to our repository. As said before in the summary, we have a pull request with the functionalities regarding the software of the 4 buttons for the controller. We will approve the merge once we consult with our project partners as well.

Tyler Greenwood
Gregory Newman (Author)
Niranjan Varma

CAMS Fall Week 9 Report

Summary

This week, significant progress was made on the front end of the controller by Tyler. He created both main screens for the device with the colored buttons as requested by the partner. We also submitted our revised Software Design and Architecture document, and are prepared to send all our documents to the partner for final approval and revision.

Next week, we will be working on our fall retrospective and will focus also on implementing some basic communication over the network using the bracelets. Our version 0 will consist of the controller's front end and a proof-of-concept of bracelet communication over the network.

Issues Completed

4) Research hardware options
- Because the controller software will be developed as an application, it will be able to run on a wide variety of hardware.
- Gregory Newman, Tyler Greenwood, Niranjan Varma
- Completed

2) Complete Software Design and Architecture document
- Gregory Newman, Tyler Greenwood, Niranjan Varma
- Completed

7) Create Controller Screen 1 UI
- Tyler Greenwood
- Completed

New Issues

15) Create simple messaging between controller and bracelets
- Implement some kind of simple messaging over the network, where a bracelet can receive either an alert or data.
- Backlog, to be worked on this coming week

Backlog Progress

This week, almost all the items in our backlog that we worked on were completed, and are listed in the Issues Completed section.

12) Create Controller Screen 2 UI

- Moved from Todo to in-progress, and over to in-review
- Tyler Greenwood
- In Review

Major Changes in Assets

The Software Design and Architecture document was completed this week and the documents we have created this term were all sent to the project partner.

Tyler Greenwood
Gregory Newman
Niranjan Varma

CAMS Fall Week 10 Report

Summary

This week, we made more progress regarding our UI for the controller. We pushed some more code onto the repository regarding our second screen for the controller. If we use an Android emulator we can push notifications using the first screen or type out a particular message using the second screen. We also finished our fall retrospectives as well by getting together in person and making a recording of it.

An update regarding our receiver hardware was that the delivery date was pushed back by 2 months. This was unfortunate, however, we were able to discuss with our project partners an alternative product that would be delivered earlier. This is so we can begin testing and don't have to be dependent on a delayed delivery.

This upcoming week we will demo the second controller screen with our project partners. We will also have a brief presentation regarding the work we have done over the past term and reflections. Our version 0 will consist of the controller's front end and a proof-of-concept of bracelet communication over the network.

Issues Completed

12) Create Controller Screen 2 UI

- Gregory Newman, Tyler Greenwood, Niranjan Varma
- Completed

New Issues

No new issues were created

Backlog Progress

15) Create simple messaging between controller and bracelets

- In progress
- As of now, we are researching different options
- Right now we are looking to pair the devices over a Wi-fi network for communication
- Gregory Newman, Tyler Greenwood, Niranjan Varma

Major Changes in Assets

We also emailed our project partner our final group revisions of the SDA, PRD, and SDP documents.

CAMS Outsafe Group
Tyler Greenwood
Gregory Newman
Niranjan Varma
CS462 Senior Software Engineering Project II
Jan 12, 2024

CAMS Winter Week 1 Report

Summary

This week, our team finally accepted the delivery of the receiver hardware (watches). Now that we have the hardware in hand, we can begin programming them, starting with printing Hello World to the display. Accomplishing this task will give us an introduction to programming on the ESP32 microcontroller, and we will continue development from there.

Issues Completed

No issues were completed this week.

New Issues

16) Print Hello World on Watch

- Gregory Newman, Tyler Greenwood, Niranjan Varma
- In Progress

Backlog Progress

15) Create simple messaging between controller and bracelets

- Printing Hello World on the watch is a prerequisite for this task, and is being worked on now.

Major Changes in Assets

- We accepted delivery of the watches and are ready to start programming them.

CAMS Outsafe Group
Tyler Greenwood
Gregory Newman
Niranjan Varma
CS462 Senior Software Engineering Project II
Jan 19, 2024

CAMS Winter Week 2 Report

Summary

This week, we met with our project partners for the first time since winter break, and met our TA for this term. We gave a presentation to them to summarize our project and our work so far, as well as sharing our projected upcoming work. Due to the weather conditions this week, including transportation issues, as well as various power and WiFi outages, we weren't able to accomplish as much. However, since the weather has gotten better we can now disperse watches among the group and make more progress.

Issues Completed

No issues were completed this week.

New Issues

17) Send test alerts from controller to local server
- Gregory Newman, Tyler Greenwood, Niranjan Varma
- Todo

Backlog Progress

15) Create simple messaging between controller and bracelets
- Printing Hello World on the watch is a prerequisite for this task, and is being worked on now.

16) Print Hello World on Watch
- This was a new task last week and is now in progress

CAMS Outsafe Group
Tyler Greenwood
Gregory Newman
Niranjan Varma
CS462 Senior Software Engineering Project II
Jan 26, 2024

CAMS Winter Week 3 Report

Summary

This week we were in full swing in trying to program the ESP32 microcontroller. Earlier in the week we ran into some problems with how we were going to do so. The reason was that there were some doubts about the firmware for the microcontroller. However, after doing some more research and asking some questions on forums related to the device we were able to confirm that we could program the microcontroller with Arduino. We still are in the process of printing output(“hello world” onto the screen). We plan on getting something printed onto the hardware by the end of the week to begin the process of pairing the controller to the watches and try to begin pushing messages.

Issues Completed

No issues were completed this week.

New Issues

No new issues were created this week

Backlog Progress

15) Create simple messaging between controller and bracelets

- Printing Hello World on the watch is a prerequisite for this task, and is being worked on now.

16) Print Hello World on Watch

- Encountered some issues with how to program the watch but I have been making major progress in understanding the hardware

17) Send test alerts from controller to local server

- The prerequisite for this task is being able to pair the controller and the bracelets with one another

Major Changes in Assets

- Although there are no changes to the assets we are still making progress and plan on pushing code into the repository to print “hello world” eventually

CAMS Outsafe Group
Tyler Greenwood
Gregory Newman
Niranjan Varma
CS462 Senior Software Engineering Project II
Feb 2, 2024

CAMS Winter Week 4 Report

Summary

This week, significant progress was made on programming the ESP32 microcontroller-based watches. It took a moderate amount of time to get the Arduino IDE set up in a way that code could be flashed directly to the watches, but the development environment is now fully ready. Additionally, we were confused on how exactly to modify the original firmware of the ESP32 and change it to our specifications, but we now finally have a conception of how it all works and how to modify the original source code. To demonstrate this, changes were made to the code to print customized output reading both “Hello” and “OutSafe v0.1” to the screen of the watch.

The watch has several built-in functions that can be selected from a scrolling menu, so the next task will be to add a new function to the menu (or modify an existing function) which will execute our communication code with the controller. This coming week, we will also begin work on sending messages over the local network.

Issues Completed

16) Print Hello World on Watch

- “OutSafe v0.1” now prints to one of the menu options on the watch.
- Completed by: Gregory

New Issues

18) Fix Light/Dark Mode Toggling on Watch

- Gregory Newman, Tyler Greenwood, Niranjan Varma

Backlog Progress

17) Send test alerts from controller to local server

- This task will be next up, and is the first step towards communication over the network.

Major Changes in Assets

- Modified Watchy firmware that prints “OutSafe v0.1” was pushed to the GitHub.

CAMS Outsafe Group
Tyler Greenwood
Gregory Newman
Niranjan Varma
CS462 Senior Software Engineering Project II
Feb 9, 2024

CAMS Winter Week 5 Report

Summary

This week, we have been looking into the communication side of the project, specifically, between the controller and the receiver watch. As of now we have an idea of or plan for what we want to do with the communication side for the receiver. We plan on using a path that doesn't require a router for now(we have that plan as a backup). We have some code we plan on utilizing which would allow us to create an access point through the Wi-Fi network. This code will set up an IP address, SSID, and password to allow another device to pair with the receiver. We are still looking into the options for how we plan on getting the controller UI connected to the internet. Potentially have some sort of computer/server that connects to the app but also directly connects to the watches as well. We have discussed this with our project partners as well and they are happy with our plan for configuring the communication.

Besides this, we also have been making some progress for the watches' UI by adding more features. We are honing in on a solution for the mode toggling for the watch. However, the priority has been to establish some sort of communication between the controller and receiver. We have set up a consistent time to meet and work together in person every week as well. We plan on starting up the communication code this upcoming week and analyzing the difficulties and errors if we have any with our TA.

Issues Completed

No issues were completed this week

New Issues

No issues were created

Backlog Progress

17) Send test alerts from controller to local server

- This task will be next up, and is the first step towards communication over the network. Found some code to establish a socket for the watch.

18) Fix Light/Dark Mode Toggling on Watch

- Have been looking into some solutions and found some code to potentially solve the task

Major Changes in Assets

- No changes to assets were made this week

CAMS Outsafe Group
Tyler Greenwood
Gregory Newman
Niranjan Varma
CS462 Senior Software Engineering Project II
Feb 16, 2024

CAMS Winter Week 6 Report

Summary

The main goal of this week was to start working on the server architecture, as well as the connectivity between the watches and the server. We've opted for an MQTT communication protocol, and have begun altering source code we have for this communication. We then started working on the server code. We opted to use Kotlin with Spring Boot to create our server API, and successfully created the basic beginnings of a server application for this project.

Issues Completed

No issues were completed this week

New Issues

20) Send MQTT messages between two computers over the network
- This is the first test of communications using this protocol, since computers are the easiest to configure to properly communicate over a network.

21) Send MQTT messages between the controller and a bracelet
- After establishing a server, we should be able to complete this task to test the full communication line between a controller and a bracelet.

22) Send MQTT messages between controller and multiple bracelets
- Being able to send an alert to multiple bracelets is the end goal of the server.

23) Add a "Drill" option into the OutSafe Server API
- We've established a basic API endpoint, but need to factor in that drill can be an option set with any alert option.

24) Configure the OutSafe Server to store addresses of watches it needs to alert
- This will be necessary to push alerts to the watches

Backlog Progress

17) Send test alerts from controller to local server

- This task will be next up, and is the first step towards communication over the network. We have established the server now, and are working on sending test alerts.

Major Changes in Assets

- Source code for MQTT communication was pushed to our GitHub, as well as a Kotlin and Spring Boot foundation for the server API.

CAMS Outsafe Group
Tyler Greenwood
Gregory Newman
Niranjan Varma
CS462 Senior Software Engineering Project II
Feb 23, 2024

CAMS Winter Week 7 Report

Summary

This week, we completed MQTT communication between two different devices over the Wi-Fi network, with the Mosquitto MQTT broker running on the client (publisher) device and sending messages to the server (subscriber) device. This is a significant step forward, and it means we can now transfer the C++ code to the bracelets and attempt a communication between a computer and a bracelet.

Progress was also made on the central controller software, which is now successfully receiving API calls from a mock controller. This means that the development of all components in the project is firmly in progress.

The most important thing to do this week is to finally achieve communication between a computer and a bracelet, and be able to display a sample message sent over the network. Progress will also be made on the central controller software, with the goal of implementing over-the-network communication between the controller software and the central controller.

Issues Completed

20) Send MQTT messages between two computers over the network

- This is the first test of communications using this protocol, since computers are the easiest to configure to properly communicate over a network.

New Issues

28) Implement Paho MQTT on Central Controller

- Use the Java Paho MQTT library to communicate with the Mosquitto MQTT broker, running on the central controller.

Backlog Progress

21) Full E2E communication test, Controller - Bracelet

- Now that communication between two computers over the network has been achieved, we will transfer the code to the bracelet and attempt communication between a computer and a bracelet.

Major Changes in Assets

- Significant updates to the MQTT documentation (README.md)

CAMS Outsafe Group
Tyler Greenwood
Gregory Newman
Niranjan Varma
CS462 Senior Software Engineering Project II
Mar 1, 2024

CAMS Winter Week 8 Report

Summary

This week we went through the process of updating our project documents to appear more in line with the progress we have made over this term. Once we finished the updating of the Software Development Process, Software Development Architecture, and Product Requirements Documents we sent them over to our project partners for verification.

We did run into some problems with the controller side of the project. Specifically, the software that was being run had some versioning issues. Upgrading to a newer version of Java prevented certain parts of the controller UI program from running. We are now in the process of trying to adjust the Controller UI to the specific version of Java so we can run the controller again.

For the receiver UI, we did make some progress. We were able to figure out certain parts of the code that needed to be changed for the random toggling of the display from dark to light mode. This can be seen in detail in the section below. We also were able to get a Wi-Fi network connection on the receiver which is big. However, we are yet to be able to get the device to ping properly to the bracelet. Basic MQTT communication will be attempted by our project partner meeting on Monday, and if we cannot get it working we will make sure we know exactly what our blockers are and how we plan to tackle them.

For now, the two main priorities are to establish MQTT communication with the bracelet receiver. Then try to restore the older version of Java or adjust certain functionalities to the newer version to get the UI working again.

Issues Completed

18) Fix Light/Dark Mode toggling on watch

- The reason why the toggling of the modes happened was because of the fillscreeen and the textcolor in our display.cpp file was switching colors. Hence to fix this for dark mode it is display.fillScreen(GxEPD_BLACK); and GxEPD_WHITE for light mode.

New Issues

- No new issues were created for this week

Backlog Progress

21) Full E2E communication test, Controller - Bracelet

- As of now, we have been able to establish a Wi-fi connection to the bracelet, however, we have only been able to do this on a home network. Right now the bracelet has not been responsive to pings from the Wi-Fi, so it is yet to be seen if the MQTT communication has worked on the bracelet.

Major Changes in Assets

- We have updated our SDA, SDP, and PRD documentation and sent it to the project partners.

CAMS Outsafe Group
Tyler Greenwood
Gregory Newman
Niranjan Varma
CS462 Senior Software Engineering Project II
Mar 8, 2024

CAMS Winter Week 9 Report

Summary

A major milestone was hit this week: the bracelets are now able to receive MQTT messages. Additionally, communication between the controller and the central controller was also completed, and the central controller can now receive all data from the controller including custom-written text messages by a user.

The plan for this week is to fill in the remaining gap in communication between the central controller and the bracelets. This will involve implementing the Mosquitto MQTT Java library in the Kotlin code, enabling the central controller's program to communicate with a locally-hosted MQTT broker, which will automatically distribute the messages to all devices listening on that topic. Additionally, we will make further improvements to the bracelet UI and polish the communication program so that it can run continuously, constantly awaiting and displaying messages as they are sent by the central controller.

Issues Completed

15) Send MQTT messages between a computer and a bracelet

- The bracelet receiver can now connect to Wi-Fi, listen to the MQTT broker running on a computer on the network, receive a message, and print it out to the screen.

17) Send test alerts from controller to local server

- Communication between controller and local server is now complete, and the server can receive all needed information including custom text messages.
- The bracelet receiver can now connect to Wi-Fi, listen to the MQTT broker running on a computer on the network, receive a message, and print it out to the screen.

New Issues

30) Create main user interface for the bracelet

- This will be the main screen of the bracelet, and will present the user with the most recently received message, the current time, and “OutSafe” indicating the program running.

31) Write a function to correctly wrap message text

- Text from a received message is not wrapping correctly. Fix this to make sure words wrap correctly to the next line and are not broken in two.

Backlog Progress

21) Full E2E communication test, Controller - Bracelet

- All communication has been completed between the controller and the central controller. The only item left to complete is implementing MQTT messaging on the central controller, enabling it to connect to the broker which will run on the same machine.

Major Changes in Assets

- This week we completed our scope and vision document, which is a composite document of all our separate items that we have created so far.

Tyler Greenwood
Gregory Newman
Niranjan Varma

CAMS Spring Week 2 Report

Summary

This week, we wrote our executive summary for our project and made a game plan for what we hope to complete this term. Progress was also made on the bracelets, and the core loop of listening for MQTT messages indefinitely is almost completed minus some minor remaining polishing.

This coming week, we will begin end-to-end testing the system, attempting to send messages from the controller, through the server, and to the bracelets over the network.

Issues Completed

34) Write Executive Summary
- Gregory Newman, Tyler Greenwood, Niranjan Varma
- Completed

New Issues

33) Indefinite Listening for Bracelets
- The bracelet software now loops indefinitely, listening for MQTT messages and reconnecting if the connection to the broker is ever dropped. Additional functionality should be added to automatically reconnect to WiFi if that connection is dropped.

Backlog Progress

No progress on backlog issues was made this week outside of the improvements to the bracelets, as we are beginning our end-to-end testing next week.

Major Changes in Assets

None this week.

Tyler Greenwood
Gregory Newman
Niranjan Varma

CAMS Spring Week 3 Report

Summary

This week we completed the first draft for the poster we are going to have at the expo. We have also established a time for the team to meet in person to work on the project. This will be 10am on Tuesdays. This is the time we plan on making progress for our end to end testing and bring different parts of the project together.

As far as progress made for the features, we have implemented indefinitely listening to Wi-Fi networks onto the bracelets and added the extra functionality of reconnecting if the network configuration is lost by the bracelet.

We have begun testing separate parts of the project but will conduct our first end to end tests this upcoming Tuesday for our team meeting/work time.

Issues Completed

35) Make first draft for poster
- Gregory Newman, Tyler Greenwood, Niranjan Varma
- Completed

New Issues

35) Make first draft for poster
- Placeholder issue for the group assignment that was due for the week

Backlog Progress

33) Indefinite Listening for Bracelets
- We have pushed the code for the indefinite listening for the bracelets onto the github with the polished up code. Specifically, the addition we had was to to automatically reconnect to WiFi if that connection is dropped. We will close the issue out with a little bit more testing of the watches.

Major Changes in Assets

The polished code with the extra reconnecting to network functionality has been added to our reciever_bracelet folder in our repository.

Tyler Greenwood
Gregory Newman
Niranjan Varma

CAMS Spring Week 4 Report

Summary

This week we had our first in-person team meeting of the term. It was very productive and we made a very large but useful change to our project. We ended up rewriting our entire server code in Node.js (originally it was in Kotlin/Spring Boot). This didn't take a lot of time and ended up being a good choice as it became super easy to implement MQTT communication inside of Node.js. This also marks the first time our server was able to communicate with the MQTT broker, meaning we are very close to getting a full end-to-end connection. The only caveat we can imagine with this change is that if we need to do any more complex data management, we will have to set up our own database system, where Spring Boot would have handled some of that for us.

Issues Completed

36) Create new server code in Node.js from the Kotlin/Spring Boot implementation
- Gregory Newman, Tyler Greenwood, Niranjan Varma
- Completed

37) Add MQTT broker publishing to Node server
- Gregory Newman, Tyler Greenwood, Niranjan Varma
- Completed

38) Write small MQTT subscriber in Node for testing purposes
- Tyler Greenwood, Gregory Newman
- Completed

New Issues

36) Create new server code in Node.js from the Kotlin/Spring Boot implementation
- Due to issues with Kotlin, we have opted to move to a Node.js framework

37) Add MQTT broker publishing to Node server
- This is the last piece needed to theoretically create an end-to-end application

38) Write small MQTT subscriber in Node for testing purposes

- For debugging purposes with the MQTT broker and the watch code, it was needed to write a simple MQTT subscriber in Node

Backlog Progress

Backlog progress is the completed issues mentioned above.

Major Changes in Assets

Our server code was rewritten entirely in a new language. It is now faster and more simple, and should be easier to ship as a product. We still have the old server code in case.

Tyler Greenwood
Gregory Newman
Niranjan Varma

CAMS Spring Week 5 Report

Summary

This week, we accomplished our **first full end-to-end test of our project.** In our Tuesday meeting, we set the controller software up on Tyler's computer, the Node.js server software and MQTT broker on Gregory's computer, and connected a bracelet to the same local network both these devices were on. The message was initiated on the controller, arrived at the server through an HTTP request over the network, was translated into an MQTT message by the server, and sent to the bracelet over the MQTT protocol. We also troubleshooted an issue where new incoming messages were not replacing the old message on the screen, just printing over it. This issue was resolved through wiping the screen with one of the built-in display methods.

The plan this week is to catch up on our documentation. We have some documentation already, but are going to work on combining it into one large document that can address everything about the project, from how to run it to how to set up a development environment to continue building on our foundation.

Issues Completed

21) Full E2E communication test, Controller - Bracelet
- Gregory Newman, Tyler Greenwood, Niranjan Varma
- Completed

33) Indefinite Listening for Bracelets
- Gregory Newman, Tyler Greenwood, Niranjan Varma
- Completed

New Issues

39) Make OutSafe Program Always Run

- Up to this point, the user has had to manually select the OutSafe program for it to run. Make it so the program runs indefinitely.

40) Create Master Technical Documentation Document

- Create a document with all our technical documentation, including how to set up the development environment.

Backlog Progress

The completion of our end to end test is the most significant achievement we have made on this project. Now, our focus will move to polishing the project and documenting the functionality of each piece of software.

Major Changes in Assets

We now have a fully-functioning prototype of our system.

Tyler Greenwood
Gregory Newman
Niranjan Varma

CAMS Spring Week 6 Report

Summary

This week there were some teammates under the weather. Hence we did not have an in person weekly meeting. Fortunately, since the main goals for this week was to make progress on the documentation for our project, we worked asynchronously. We also were able to show our end-to-end product to our project partners which they were happy with the progress. Our project poster also got approved to be sent for printing.

This upcoming week, we will be continuing working on documentation and polishing up the display for the messages shown on the watch. We also need to send our poster in for printing.

Issues Completed

No issues were completed this week.

New Issues

No new issues were made this week.

Backlog Progress

40) Create Master Technical Documentation Document

- Niranjan Varma, Tyler Greenwood, Gregory Newman
- Made a master document and have made some additions and progress to our documentation

Major Changes in Assets

Created the Master Technical Documentation document but are still in the working stages of completion

Tyler Greenwood
Gregory Newman
Niranjan Varma

CAMS Spring Week 7 Report

Summary

This week we continued working on documentation for the project in order to have the project partners set up the project independenty once we hand it over. We also have been working on the public landing page which gives a rundown of the project on a website. We met with our project partners this week to present another end-to-end test of the execution as well.

We also did send over the information to get our poster printed and have officially received a confirmation of the print job for the poster from OSU Media Hub. Another item we worked on this week was creating a docker for the MQTT broker and node server to make the execution process for the project easier and more seamless. We have pushed the code for the docker and also pushed the code for the landing page into a separate folder.

This upcoming week will be more of the same regarding the finishing touches for the Realease Candidate regarding demo videos, documentation and some final check through for the code.

Issues Completed

42) Create a Docker container for the node server and MQTT broker

- Gregory Newman, Tyler Greenwood, Niranjan Varma
- Completed

New Issues

42) Create a Docker container for the node server and MQTT broker

- Create a Docker container for the node server and broker in order to make a package that can be easily executable

Backlog Progress

40) Create Master Technical Documentation Document

- Niranjan Varma, Tyler Greenwood, Gregory Newman
- Still in the process of writing the documentation and testing to make sure we are providing the correct information

Major Changes in Assets

We pushed the code for the docker and also have created the landing page as well