Kennesaw State University

IoT Lanyard ID

Group 5: Caleb Goodman, Nolan Gilmore, Anthony Lopez, Johnny Lozano, Etta Archere, Alem Sahic

Task 1 video link:

https://kennesawedu.sharepoint.com/sites/Team-CPE-4020Team5/_layouts/15/stream.aspx?id=%2Fsites%2FTeam%2DCPE%2D4020Team5%2FShared%20Documents%2FGeneral%2FGroup%205%20Final%20Project%2FGroup5%2DVideoDemo%2Emp4&referrer=Teams%2ETEAMS%2DWEB&referrerScenario=teamsSdk%2DopenFilePreview

CPE 4020: Device Networks

Dr. Billy Kihei

14 November 2022

Summary:

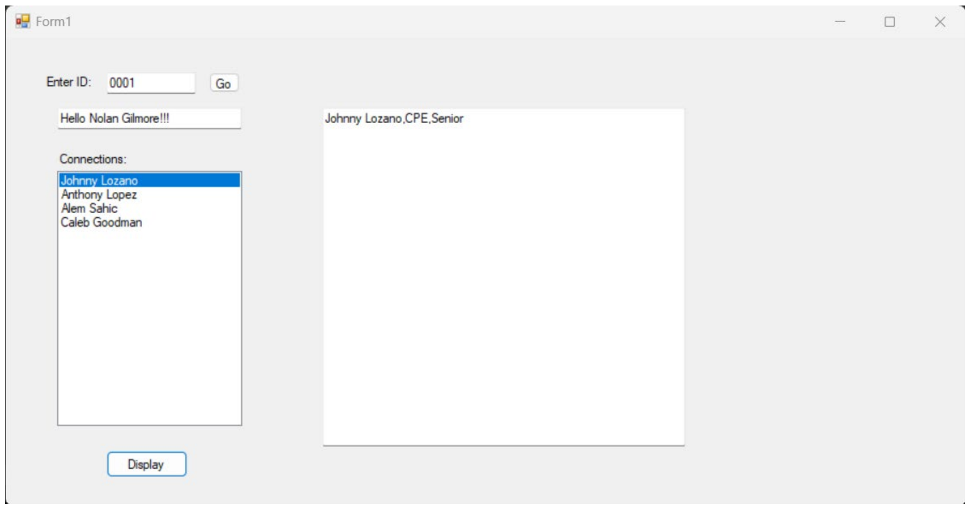
Task 1:

We have created an RFID lanyard that when you link up with another person and tap their lanyard, you can later pull up a GUI that shows all the people you have tapped and their information.

The publisher is the ESP32 embedded device which is connected to an RFID reader. The embedded device has a four-digit code which identifies the “user”, and the four-digit code that is read from the RFID identifies the “target”. The ESP32 puts both the user and target codes into a http request to the broker to publish the scanned information.

The broker is implemented on the Beaglebone Black (BBB) which uses JavaScript to host a server. A database with all the registered users and their respective data is stored directly on the server. The server receives the http request from the ESP, and it stores the target information to the user’s connections list. The server listens for an API endpoint for the client to access, and it parses the requested information and sends it to the client in a string format. The server also has an API endpoint for the client to receive all the registered users, so that the client can make the appropriate http requests for each user.

Part b: Our subscriber is a laptop that is running a C# Windows Form application. We designed an interactive GUI that allows the user to input their ID and press a button to go to their “page” and display their connection list, which is retrieved from the broker. The user can select a name from the connections list and press a “Display” button to display the selected user's information on the right side of the screen. The connection list is constantly updating allowing for a user to see their connections increase in real time. Here is an example screen shot screenshot of our app running with Nolan’s ID



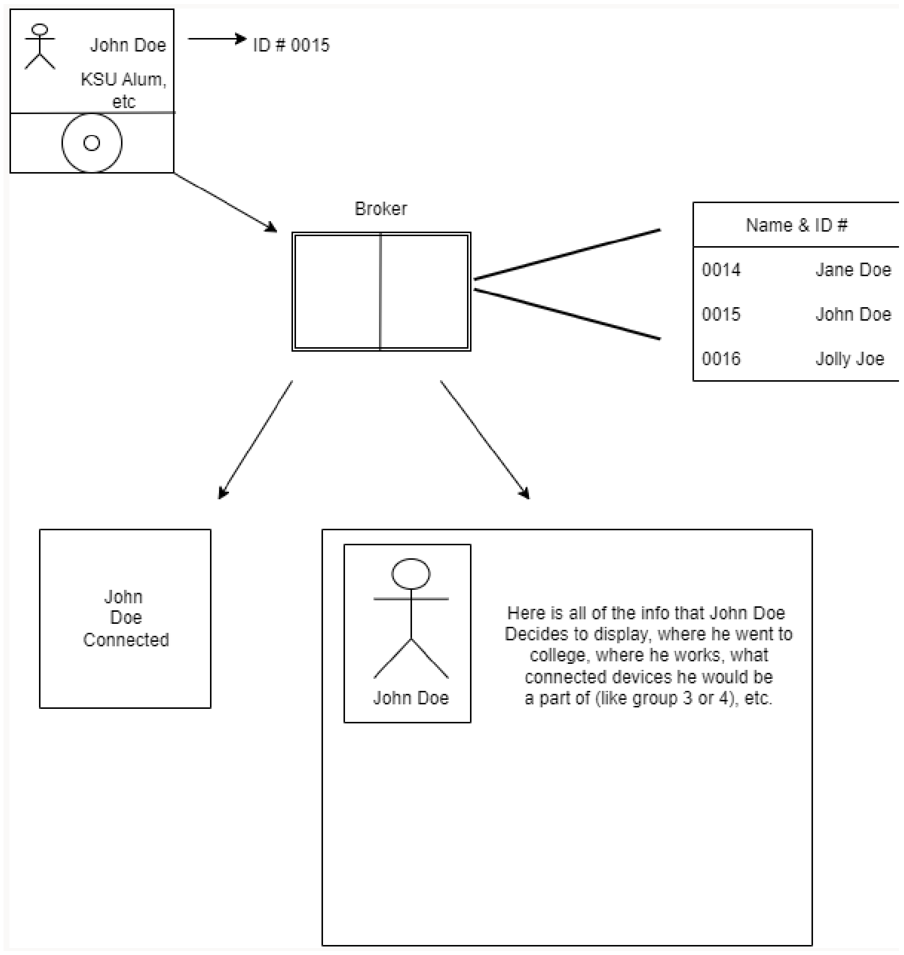
Enter ID: 0001

Hello Nolan Gilmore!!!

Johnny Lozano
Anthony Lopez
Alem Sahic
Caleb Goodman

Johnny Lozano,CPE,Senior

System Diagram:



Name & ID #	
0014	Jane Doe
0015	John Doe
0016	Jolly Joe

Here is all of the info that John Doe Decides to display, where he went to college, where he works, what connected devices he would be a part of (like group 3 or 4), etc.

Task 2 Theoretical Implementation:

(Since we did not have enough time to properly integrate with another team's project, we have created a few proposals based on their initial ideas on how we could improve and enhance our product)

Task 2 – Proposal 1: Antisocial Deterrent

Our idea for integrating with another group's project is to create a way to encourage someone to make connections with others at conventions, school conventions, the workplace, etc. using our smart lanyard design. We would be collaborating with Group 4's ring taser. A timer would be added to our lanyard that counts how long it has been since the last connection the wearer had made. When it reaches a max value (30 minutes or so) The lanyard will call the ring taser to give the wearer a shock to remind and encourage them to continue making connections. The only way to reset the timer is to tap the wearer's smart lanyard with that of another, which will make a new connection and therefore accomplish the goal of the proposal.

Task 2 – Proposal 2: Study buddy integration

Another idea for integration with another group's project would be to enhance group 2's study buddy concept with the smart lanyard. It will work in a similar way to the first proposal mentioned above, in that it serves to encourage someone to network with others at a convention, but integrating with group 2's tech. The study buddy will operate as normal, but it will also be monitoring the smart lanyard to see if the wearer is making connections with others. If they have not networked after a certain amount of time the study buddy watch is alerted, and the watch wearer will be obligated to encourage the lanyard wearer to make new connections. This could serve in tandem with the ring taser as the first, more lenient warning to continue networking before the more assertive ring taser warning.

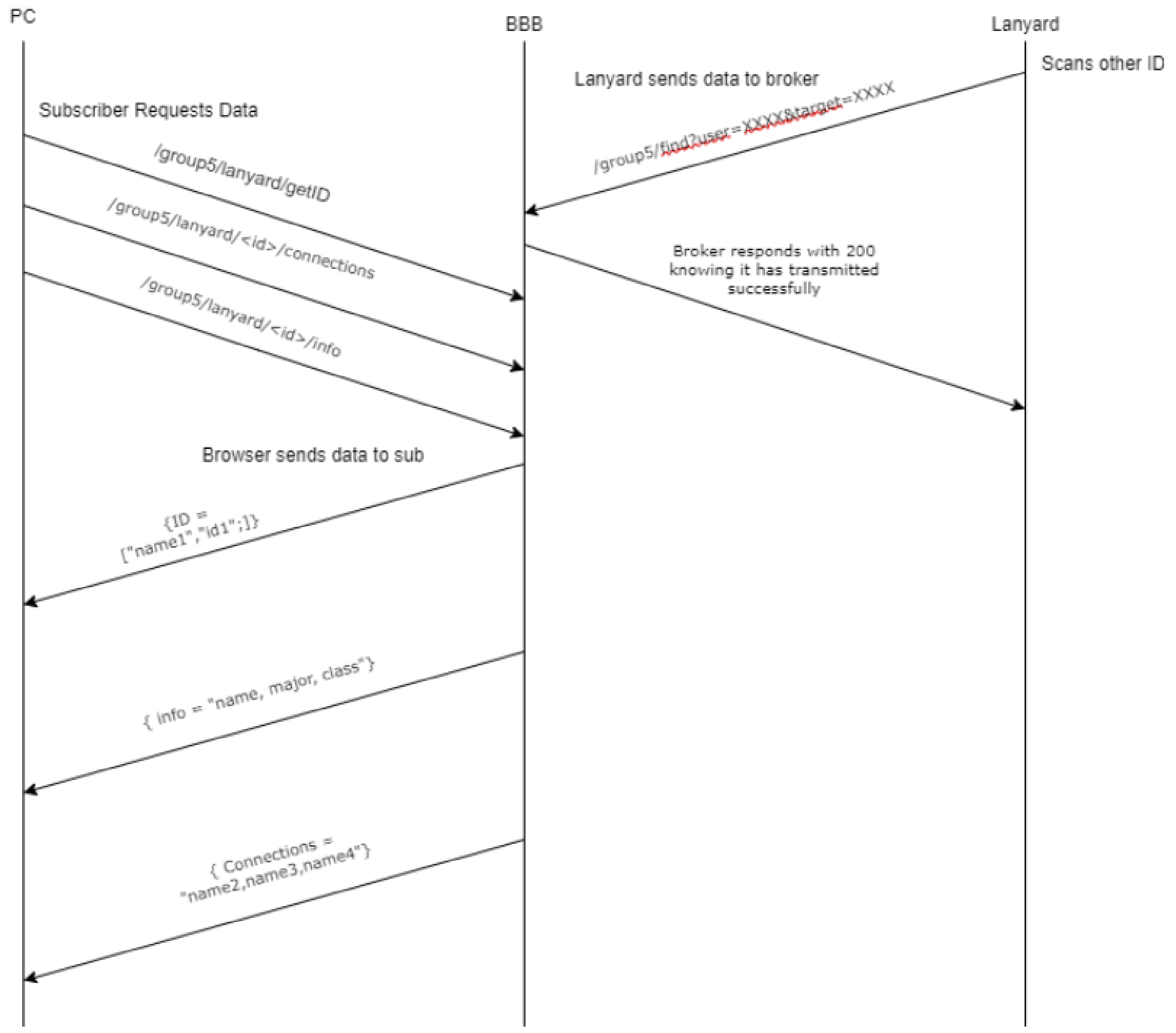
Task 2 – Proposal 3: Fitness Integration

For the third proposal we developed, we could integrate our platform with that of the fitness tracker that group 3 is developing. Alongside the biographic & contact information our platform stores for its users, we could also store the data from the fitness tracker and allow the user to share that data with others when they make a connection with our lanyard. This could be useful if these lanyards are used at gyms or athletic events such as marathons if the users would wish to share their fitness information with others for competitive or comparative purposes. Storing this additional information could allow us to expand the intended market of conventions to other events where this information would be useful to possess and share with others.

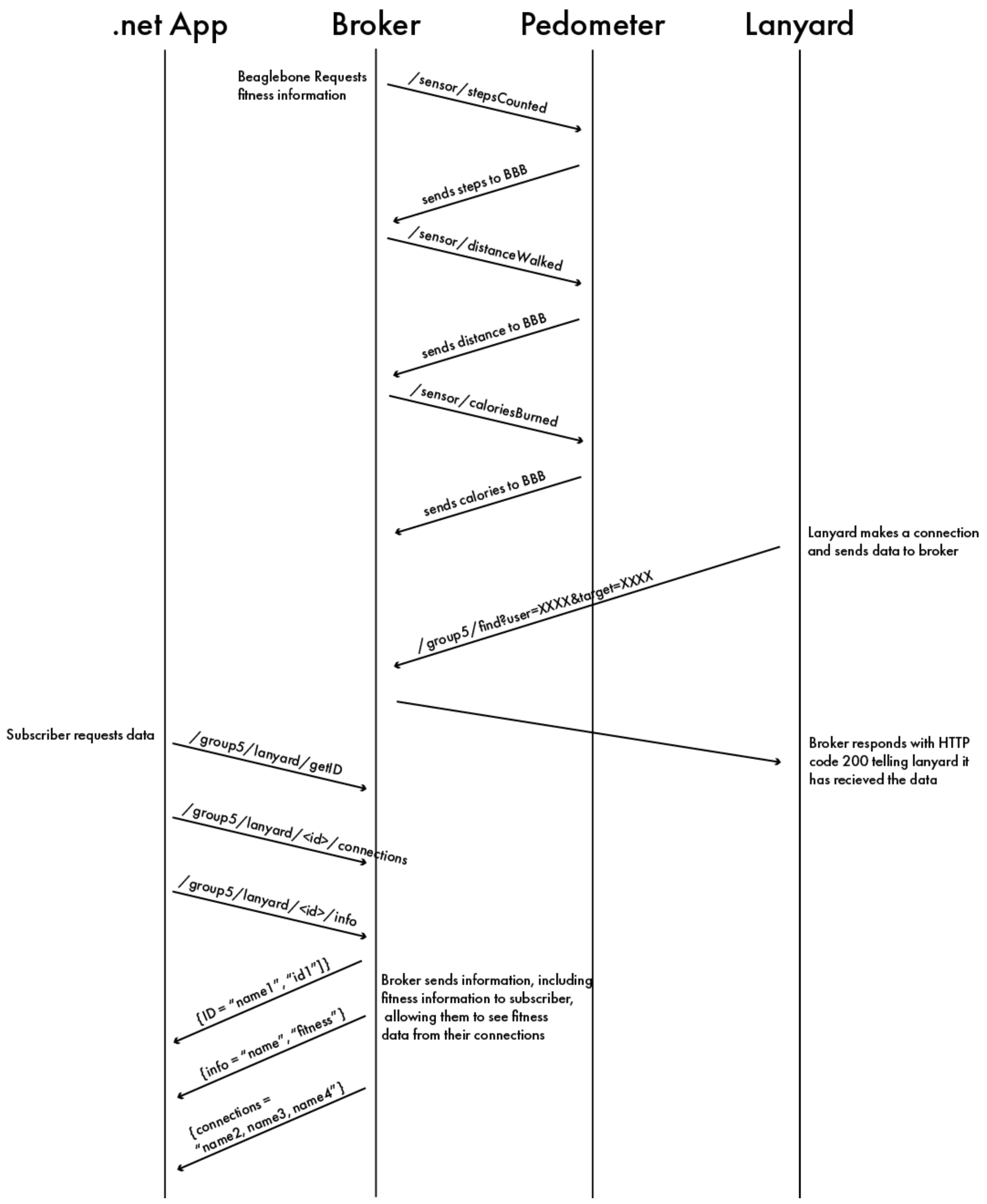
Pub/Sub Protocol

Our task 1 pub/sub protocol is simple and implements a message broker to handle message requests. The broker is a data server that is constantly listening for requests. It establishes a TCP socket for publisher and subscriber data transfer. It uses API requests to allow subscribers to request data from the broker and publishers to send data to the broker. The publisher in our case is the smart lanyard device, sending information to the broker. The subscriber is our .net application, it is responsible for making most of the API calls to retrieve the information

Task 1 Waterfall plot:


PC
BBB
Lanyard
Scans other ID
Lanyard sends data to broker
/group5/find?user=XXXX&target=XXXX
Subscriber Requests Data
/group5/lanyard/getID
/group5/lanyard/<id>/connections
/group5/lanyard/<id>/info
Broker responds with 200 knowing it has transmitted successfully
Browser sends data to sub
{ID = ["name1","id1";]}
{ info = "name, major, class"}
{ Connections = "name2,name3,name4"}

Task 2 Waterfall plot: (Theoretical for proposal 3)


.net App
Broker
Pedometer
Lanyard
Beaglebone Requests fitness information
/sensor/stepsCounted
sends steps to BBB
/sensor/distanceWalked
sends distance to BBB
/sensor/caloriesBurned
sends calories to BBB
Lanyard makes a connection and sends data to broker
/group5/find?user=XXXX&target=XXXX
Broker responds with HTTP code 200 telling lanyard it has recieved the data
Subscriber requests data
/group5/lanyard/getID
/group5/lanyard/<id>/connections
/group5/lanyard/<id>/info
{ID = "name1", "id1"]}
{info = "name", "fitness"}
{connections = "name2, name3, name4"}
Broker sends information, including fitness information to subscriber, allowing them to see fitness data from their connections

API Documentation

Our BBB Broker is organized around RESTful APIs. The broker server is constantly listening and receives requests from our ESP32 Lanyard when a connection is made. The APIs can also return JSON-encoded responses upon client requests.

API's Request Reference:

To Request Data:

/group5/lanyard/getIDs - returns an array of names and IDs in JSON format

```
{
ID = [
"name1", "id1",
"name2","id2" …
          ]
     }
```

/group5/lanyard/<id>/info - returns a string of the ID's info

```
     {
info = "name,major,class"
     }
```

/group5/lanyard/<id>/connections - returns a string of the ID's connections

```
     {
     Connections = "name2,name3,name4"
     }
```

Example: Nolan's ID is 0001

Example: /group5/lanyard/0001/connections - shows connections that the user has encountered, in this case all the people that Nolan has connected with. - "Anthony,Caleb,Johnny"

Example: /group5/lanyard/0001/info - returns Nolan's info - "Nolan Gilmore,CPE,Senior"

To Send Request:

/group5/find?user=XXXX&target=XXXX

- Where user "XXXX" is the ID associated with the lanyard, and target "XXXX" is the ID associated with the person they have scanned/connected with.

Security Measures:

Our project does not include security measures. We are vulnerable to all sorts of attacks by the CAMA standards. For example:

- We are assuming that no one is going to want to do anything with our data such as steal/ spoof/ DDOS/ etc. Confidentiality is not upheld because there is nothing stopping anyone from calling our API, seeing that info, and taking it for themselves/send it to someone else.
- There is no encryption method when data is being transferred, thus any request can be sniffed by any intruder.
- There is also nothing that prevents them from attempting a DDOS attack on our project, so there is potential for accessibility problems.
- Message integrity can also be broken because the information is there in the actual API call itself. We do not implement any hashing, therefore data transmitted can be altered and the receiver would have no clue about it.
- There is also no authentication in our project, because any device can claim to be our ESP32 device and say that they are something/someone they are not.

These are the flaws that are in our project and can be fixed by creating a more complex way of doing things. We were told to keep asking the question of “can we make it simpler” and the answer to this was “yes” by creating a less secure and less complex code. For example, authentication can be implemented in order to ensure that only the registered ESP32 devices can communicate with the server.

Summary

This project was a great way to apply the lessons we learned throughout the year. In the end, we were able to design a publisher-subscriber IoT wearable product. Our product is meant to connect people at events where meeting and greeting people can be a great quantity, such as at conventions. The problem we aim to solve was the overwhelmingness of trying to remember everyone you spoke to. Instead with our product, a person can scan another person's lanyard to get their information and have it accessible later. Unfortunately, due to the time crunch of the project we couldn't fully integrate our product with another group's pub/sub IoT network. Though, we did theorize some ideas based on the other group's project declarations.

Challenges:

One of the main challenges we faced was we were not able to write to our RFID card. We were able to read what was already stored on the card and the key tag, but we could not write to the card to change what was on it. We fixed this problem by just using the predetermined values that were on the card and saying in our code that "user 1 = card 1 number, user 2 = card 2, etc." then do "if this users data matches the one from this card, then match them up and call it a day". On the broker side, the main challenge was coordinating the API endpoint names and formatting how the data would be sent. For instance, the broker code originally was going to send the list of connections in JSON format, but the client required the connection list to be in string format. Also, the client required more API calls than what the broker originally implemented, so we had to add to and reformat the existing endpoints. On the subscriber side, one of the main challenges was to implement the use of a list box. This would allow the user to select a name from the list box and display the selected name's information to a textbox. Thus, we needed a way to correspond each name to an ID to make sure we are calling the right API for whatever name we have selected. In our subscriber code we used a dictionary to correspond a name (key) to an id (value) and concatenated the id to our API call.

<u>Zip File Contents:</u>

Group5_Final_Project.zip contains three directories:

client/

Form1.cs

server/

broker.js

sensors/

Group5_ESP32_Source.ino