Software Overview

Year: 2023 Semester: FallAssignment Evaluation: See Rubric on Brightspace Assignment

1.0 Software Overview Team: 8 Project: Smart Seating System
Creation Date: 09/07/2023 Last Modified: 09/09/2023
Author: Roshan Sundar Email: rmsundar@purdue.edu

Control Unit:

This section describes the general software functionality of the Control Unit that will be mounted on the chair, where the software will be running on the micro.

The micro will poll sensors for data at certain intervals predefined in code (i.e. every 20 seconds), the sensors will be sampled for a predefined duration (i.e. 5 seconds) to get a range of values.

- Different sensors will have different polling intervals and durations. For example, a PIR sensor would have to be polled much more frequently than a force sensor.
- This is done to reduce data bandwidth and save power
- The micro will be able to toggle the power to each sensor, turning them off when not in use to save power

The data will then be processed to consolidate the data into some average. For example, if an ultrasonic distance sensor returned 5 values when it was polled, it would take the average of the values. The result is an average value for each sensor after it is polled.

- These averages are compared to the previous polled averages. If each sensor average has not changed much since the last time it was polled (some threshold), we continue the process above.
- However, if even one sensor has an average significantly different from its previous, then it could be that some change occurred. Either someone recently left the seat or has recently sat down; though, it could also be some kind of anomalous reading.
    - So, all sensors are then polled and averaged at once. The resulting data is sent through a Bluetooth or WIFI connection to the Web Server. The web server will then be responsible for using this aggregated series of sensor data to determine the occupancy of the seat.
- The reason for this is to avoid sending a constant stream of data the server, which saves power on the Control Unit and reduces usage of network bandwidth

A capability the team was thinking of adding (not sure yet) is the ability for the micro to go into some 'config' mode, where it can have parameters such as its unit ID, thresholds, polling freq, etc. set by a push from the Web Server. The purpose would be to easily customize/adjust these parameters without having to plug the unit in and change the code after deployment. This could easily be triggered by a button press-and-hold on the Control Unit, which sets it into a 'waiting' mode for a WIFI or Bluetooth packet from the server containing new parameters. Once received, these parameters would be written to a static EEPROM for permanent storage and reference.

Web Server:

The Web Server will likely exist on a standard computer/laptop. Its minimum requirement is to take in the series of sensor data sent by each Control Unit and determine occupancy for that unit.

The Server will maintain an internal table associating each Control Unit (likely via ID) with a Seat and its occupancy status. It will constantly be awaiting updates from the Units it is paired to. As soon as a data packet arrives for a given Unit, it will perform an analysis on that data to determine the occupancy status. The method has yet to be determined, but it could be as simple as a majoritarian analysis: if the majority of sensors indicate occupancy, then the Unit/Seat is occupied. Once the occupancy status is determined, it updates its internal table with the new occupancy status of that Unit/Seat.

The functionality of the web server could be extended given time, to perform historical record keeping and statistical analysis.

Phone App:

This software is an extra thing the team will do given time, to extend the project. The Phone App would be the way that users/visitors of the space would interact with our system to check seat occupancy. It would likely involve a connection to the Web Server in order to grab the internal table and display the seat occupancies visually on the App.

2.0 Description of Algorithms

Several algorithms, protocols, and processes will be utilized. We highlight a few here:

1) Polling:
    - Process by which a client can sample the status of an external device [1].
    - This method, as described in the preceding section, will be the method by which the sensors (external devices) will be monitored by the micro (client)
    - Polling in general can take many forms. The form we are using is a modified form of what is called roll call polling, in which multiple external devices (sensors) are queried in a sequence [1].
2) HTTP:
    - Assuming WIFI is used for wireless communication from Control Units to Web Server. Otherwise, Bluetooth protocols would be used.
    - HTTP is a high-level protocol that enables standard client-server communication between two entities that have a web connection. It does not handle the actual transportation of information which is done with lower-level protocols such as TCP/IP [2].
    - HTTP utilizes standard methods that have custom implementations in order to facilitate communication between clients and servers. For our purposes, POST or

PUT may be used to transfer the sensor data from the Control Unit to the Web Server [2].

3.0 Description of Data Structures

Several data structures will be utilized. We highlight a few here:

1) JSON:
    - JSON is likely the format that would be used to package the data that would be sent from the Control Unit to the Web Server.
    - JSON is a lightweight and standard format that is common and language-independent. It is both human-intuitive and easy to computationally create/parse [3].
    - JSON has objects that support name-value pairs. The values can be a variety of data types; such as strings, ints, longs, and many other data types [3].
        - Our implementation would include attributes such as Unit ID, and an attribute for every sensor value, bundled into one object.
2) SQL Databases:
    - The SQL database is the tool we will most likely use to create the occupancy table mentioned in *Section 1.0* on the Web Server, that will store the Control Unit IDs, corresponding seat name/identifier, and the occupancy status.
    - SQL itself is a language in a 'query' format to interact with database data structures, such as our table. The databases are files that are stored on the disk of a computer and are typically organized in a row-column structure [4].
    - We will use SQL on the Web Server to access the occupancy table for read/write purposes. A write operation would be carried out when a Control Unit has sent data. A read operation could be used to deliver data to the Phone App, if we decide to make it.

4.0 Sources Cited:

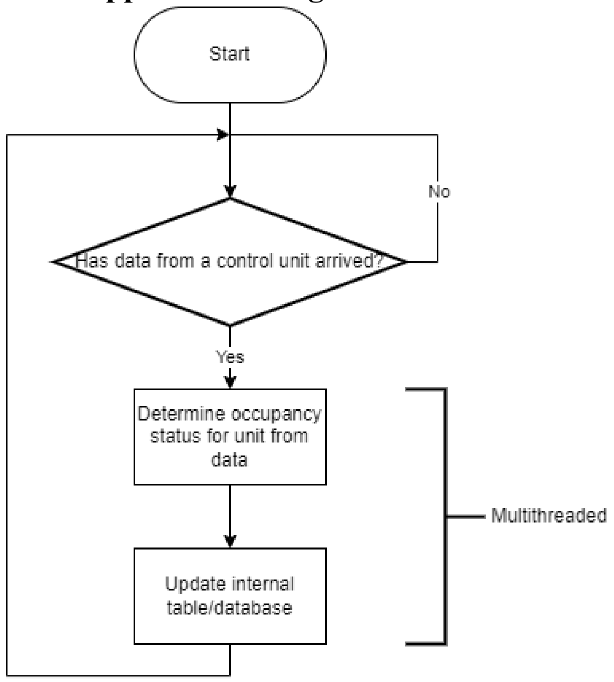
[1] "Polling (computer science)," *Wikipedia*, May 31, 2020. https://en.wikipedia.org/wiki/Polling_(computer_science) (accessed Sep. 9, 2023)

[2]"HTTP," *Wikipedia*, Sep. 01, 2023. https://en.wikipedia.org/wiki/HTTP#HTTP/1.1_response_messages (accessed Sep. 9, 2023)

[3] "Introducing json," JSON, https://www.json.org/json-en.html (accessed Sep. 9, 2023).

[4] "What is SQL? - SQL - AWS," *Amazon Web Services, Inc.* https://aws.amazon.com/what-is/sql/#:~:text=Structured%20query%20language%20(SQL)%20is

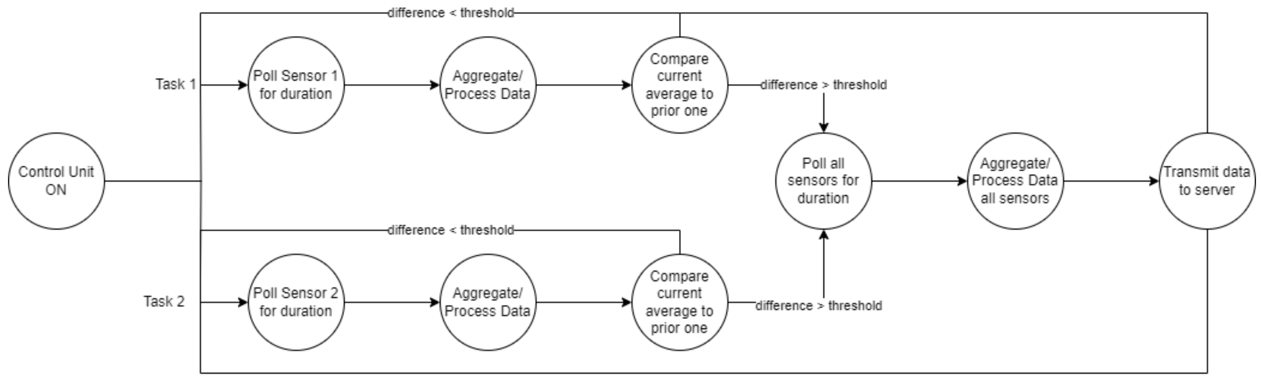
Appendix 1: Program Flowcharts


Start
No
Has data from a control unit arrived?
Yes
Determine occupancy status for unit from data
Multithreaded
Update internal table/database

The above is a flowchart to broadly describe the software outlined in *Section 1.0* on the Web Server.

- A flowchart of the Control Unit software is shown in the form of a state machine shown in the next section.
- The process once data has arrived from a Control Unit has to be multithreaded so that it can receive data sent while another set is currently being processed, or if two Control units send occupancy data at the same time.

Appendix 2: State Machine Diagrams


Control Unit ON
Task 1
Task 2
Poll Sensor 1 for duration
Poll Sensor 2 for duration
Aggregate/ Process Data
Aggregate/ Process Data
Compare current average to prior one
Compare current average to prior one
difference < threshold
difference < threshold
difference > threshold
difference > threshold
Poll all sensors for duration
Aggregate/ Process Data all sensors
Transmit data to server

This diagram is a state machine to broadly describe the software outlined in *Section 1.0* on the Control Unit.

- Since the ESP32 supports RTOS multitasking, each sensor polling and processing operation runs as a task. Two are shown above, but more tasks can be created for additional sensors.
- Also, understand that the durations, differences, and thresholds will likely vary from task to task.
- When a task is exited because (difference > threshold), then all other sensor-monitoring tasks should be suspended. When the transmission is done, all those tasks are resumed again. The ESP32 RTOS should be able to support this.
- An alternate way to get a similar outcome to this multitasking behavior (without
- multitasking) would be through the use of a timer.
    - The code would run a timer and loop through methods (one for each sensor, same three steps as task in diagram) linearly.
    - When a polling duration time is reached, the method would be invoked.