Software Overview

Year: 2021 Semester: Spring Team: 4 Project: Virtual Queue
Creation Date: 2/3/2021 Last Modified: 2/5/2021
Author: Chok Yip Lau Email: lau55@purdue.edu

Assignment Evaluation:

Item	Score (0-5)	Weight	Points	Notes
Assignment-Specific Items				
Software Overview		x2		
Description of Algorithms		x2		
Description of Data Structures		x2		
Program Flowcharts		x3		
State Machine Diagrams		x3		
Writing-Specific Items				
Spelling and Grammar		x2		
Formatting and Citations		x1		
Figures and Graphs		x2		
Technical Writing Style		x3		
Total Score				

5: Excellent 4: Good 3: Acceptable 2: Poor 1: Very Poor 0: Not attempted

General Comments:
Relevant overall comments about the paper will be included here

1.0 Software Overview

The software for Virtual Queue will consist of two major components which will provide customers with hassle-free check-in to retail stores while complying with Covid-19 restrictions and ensuring the safety of both customers and employees. The first component is the software on the microcontroller which will retrieve data from on-board peripherals, such as the thermometer, microphone, speaker, PIR sensor, WiFi module, and display. The second component will be a web page that allows customers to check in via a QR code and notifies customers once it is their turn to enter the store.

The on-board software will first boot up all the aforementioned peripherals to check if it can communicate with them. If all the peripherals are working properly, the on-board software will then start to communicate with the web server to check the Virtual Queue via JSON[3] API. If there are no customers waiting on the queue, the microcontroller will go into standby mode [1]. Every time the QR code scanner is triggered, the on-board software will send a message to the web server with the QR code information it receives after encoding the QR code into JSON[3] format. The web server will determine if the customer should be put in the queue by checking if they are already on the queue. If the customer is not on the queue, then the web server will put the customer on the queue. A success or fail message will be sent back to the microcontroller accordingly in JSON[3] format. When it is time for a customer to enter the store, the thermometer (a thermopile sensor) will be activated. If the customer successfully measures their temperature and the motion sensor detects them entering the store , the display will be updated to show the status of the queue and the amount of customers in the store. If the motion sensor is triggered before the thermopile sensor records the customer's temperature, employees will be notified and microphone and speaker will be activated for the employees to communicate with the customers. On the other hand, if the motion sensor detects a customer is leaving the store, it will notify the web server and the web server will notify the next customer on the queue and update the device display. In any state where the motion sensor detects an unauthorized entry (no valid temperature reading was taken), employees will be notified and microphone and speaker will be activated.

2.0 Description of Algorithms

Queue management will be the core of our system and will be implemented in the web server. The Queue management will include add, remove, and notify functions. Everytime it receives a query from the on-board software, it will perform a lookup to see if the customer is already on the queue. If the customer is not on the queue, the add function will be called to add the customer on the queue. After that, we will check if the customer can enter the store or not. If the customer can enter the store, the notify function will immediately be called to notify the customer. The web server will send a JSON[3] message to every Observer (the microcontroller and customers) so the Observer pattern[2] will be implemented. If the customer is not allowed to enter the store, they will be put on the queue. When the next customer exits the store, then the notify function

will be called to notify the next customer on the queue. Once it is the customer's turn to enter the store and the customer has scanned their QR code to verify their presence, an API query will be sent to the web server. After the web server verifies if the customer can enter the store by checking if the customer is on the queue and the customer is next on the queue, the remove function will be called to remove the customer from the queue. We will use JSON[3] as our API and parsing the API received by the server can be accomplished using if-conditionals as shown in Figure 3.1.

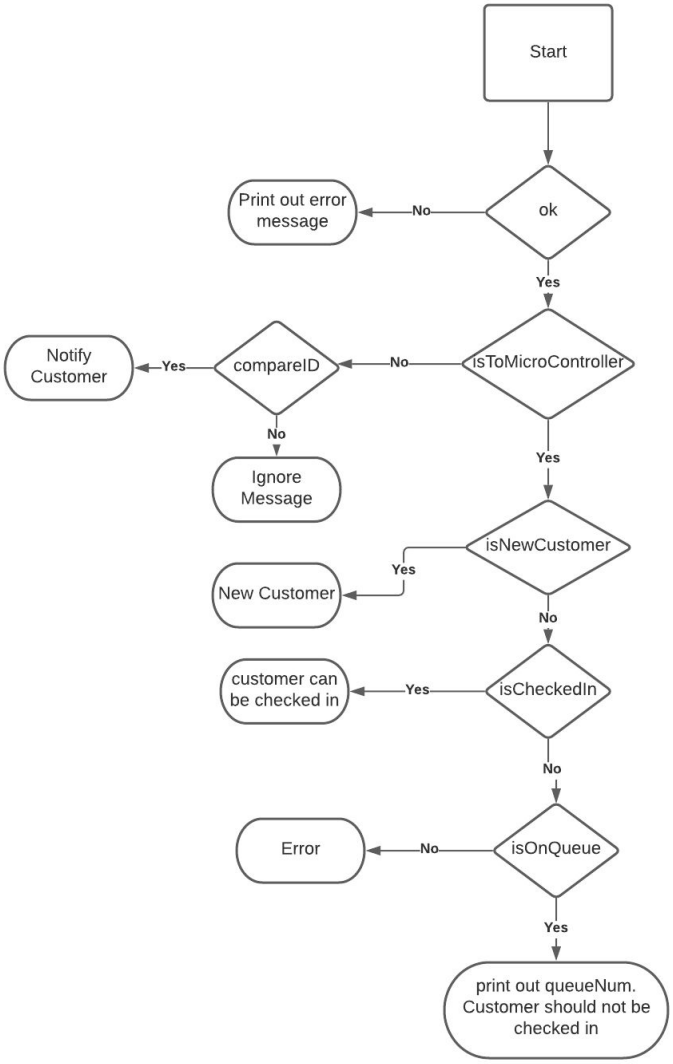

Figure 3.1: Algorithm to pass JSON API

Besides the Queue management, another vital algorithm that needs to be implemented is to detect if a customer is allowed to enter the store or not. This algorithm will need the coordination between the motion sensor and the temperature sensor and the microcontroller will perform computations using if-conditional statements. When a customer is ready to enter, the temperature

sensor will be triggered in order to measure the customer's temperature. If the motion sensor is triggered prior to the temperature sensor or the temperature measured is more than 100.6°F, employees will be notified and the employees will use microphone and speaker to communicate with the customer at a safe distance. The algorithm that will be used to determine if a customer is coming into the store or going out of the store will involve taking the threshold of the motion sensor reading. As Figure 3.2 shows, the motion sensor will have 4 IR sensors equipped. Taking the differences from different IR sensors will show the movement of the object when there is a sudden burst as shown in Figure 3.3.

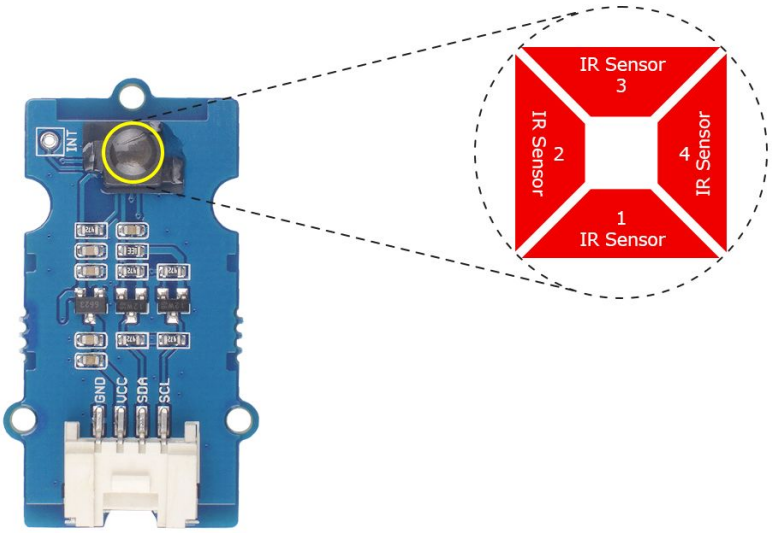

Figure 3.2 - AK9753 sensor [4]

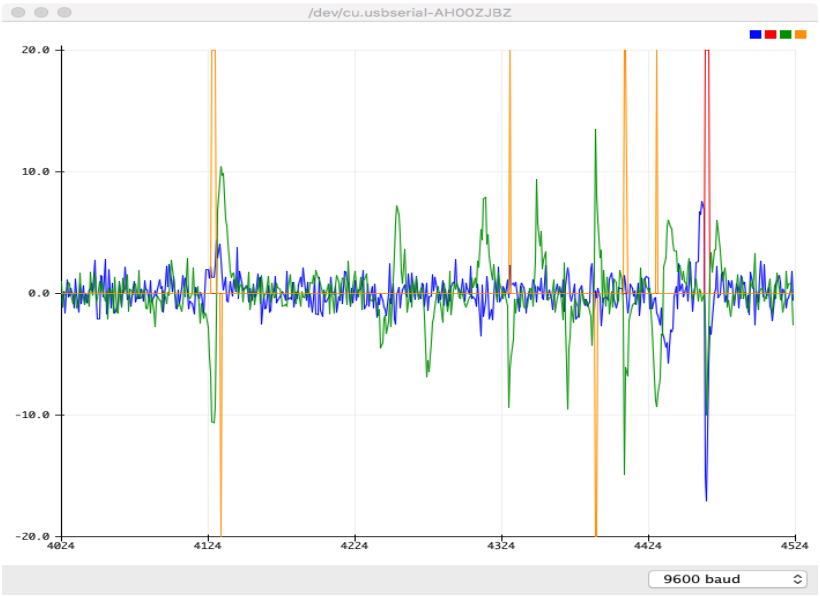

Figure 3.3- Derivative of the different value of IR1 IR3 (orange) and IR2 IR4 (red) [4]

3.0 Description of Data Structures

Based on the two characteristics for our Queue system - quick lookup and growing memory, the best data structure to implement our queue system will be a LinkedList with a HashMap. Using a LinkedList will allow the Queue system to quickly add and remove nodes by manipulating the pointer. For the HashMap, the key of the HashMap will be the customer id and the value will be the node that links to the customer ID. This will enable us to quickly locate the customer without having to iterate through the LinkedList.

The message being sent between the web server and microcontroller will be in JSON[3] format.

The JSON[3] that will be sent by the microcontroller will be in the format shown below.

```
{
  “isCustomerWentOut” : true/false
  “customer”: {
        “id” : xx
   }
}
```

The JSON[3] that will be sent by the server will be in the format shown below.

```
{
“status”: “ok/err”
“isToMicroController”: true/false
“customer”: {
        “id” : xx
        "numofCustomer": xx
        “isTimetoCheckIn”: true/false
        “isNewCustomer”: true/false
        “isCheckedIn”: true/false
        “isOnQueue”: true/false
        “queueNum” : xx
   }
“err_msg”: “xxxx”
}
```

The JSON[3] message will be received by either the user or microcontroller and will perform the appropriate action based on the algorithms mentioned above.

The data structure that we are going to use communicate with the display will be using SPI interfaces and will exchange 16 bits data from the display driver to the microcontroller when the SCS# bus goes low. The figure below shows an example of what will happen if we want to perform a read operation from the display driver. When the SCS# bus goes low, the MCU will start to exchange data with the display driver. 16 bits of data will be exchanged which the first 2 bits indicate which operation will be selected. If the read operation is selected, the 9th bits to 16th bits will be the read data. Likewise, if the write operation is selected, the 9th bits to 16th bits will be the data that the master wants to write to the slave.

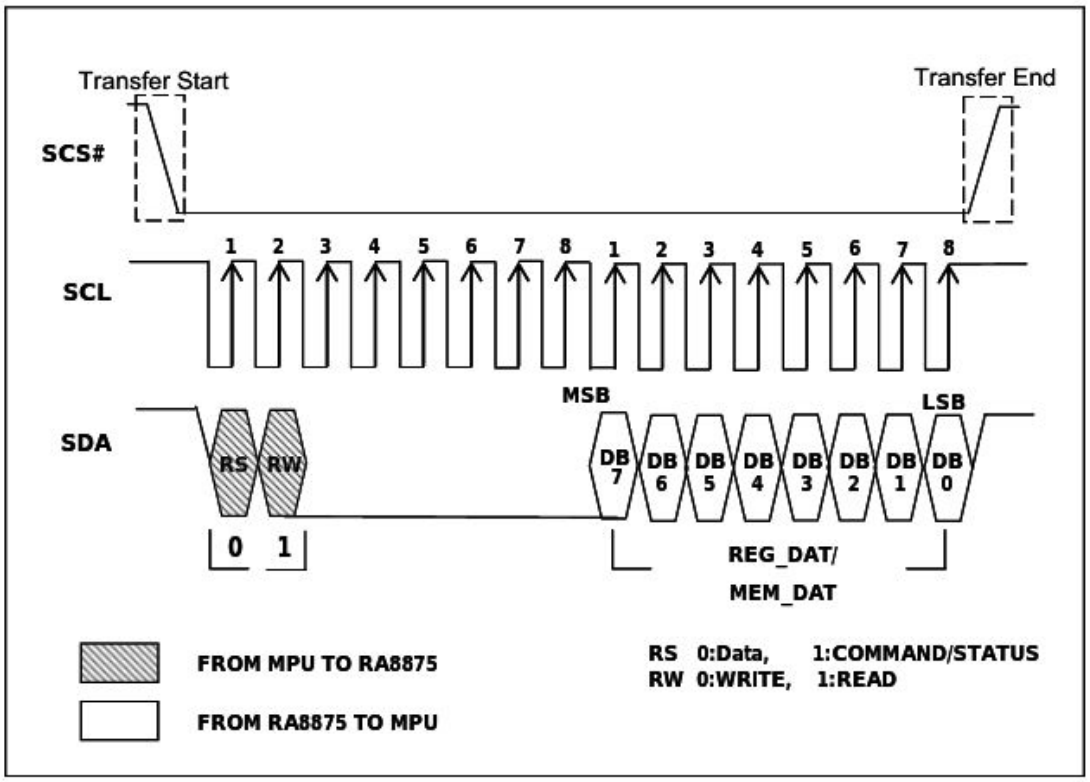

Figure 3.4 - Data Read on SPI-bus [5]

The motion sensor will use I2C interfaces to communicate with the microcontroller. According to the data sheet[6], the data format that will be used will be 1 byte data packet as shown in Figure 3.5

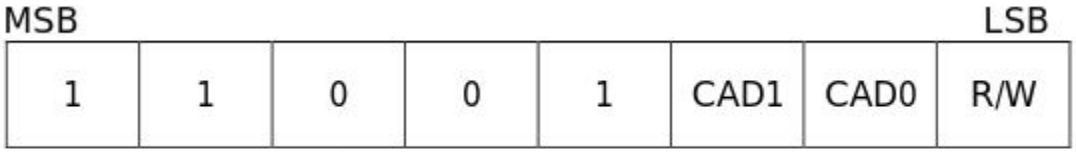

Figure 3.5 - Slave Address [6]

the microcontroller will send the one byte data that contains the slave address on the I2C bus, once the motion sensor receives the data packets, it will be selected and reply to the Master with an ACK packet. The LSB of the packet will indicate whether the master wants to perform read or write operation. Once the handshake has been successfully executed, 1 byte data packet will be exchanged between the master and slave on the I2C bus.

The QR code reader will utilize the UART interface to output scanned QR code data. According to the data sheet [7], the data format is as the following when we perform read:
{Head1} {Types} {Lens} {Address} {Data} {CRC}
Upon successful reading, the QR code scanners will return:

{Head2} {Types} {Lens} {Data} {CRC}
For example, if we want to read one byte at 0x00 and 0x0A, we will send the command:
{0x7E 0x00} {0x07} {0x01} {0x00 0x0A} {0x01} {0xEE 0x8A}
And upon successful reading, it should return (assume the data at 0x00 and 0x0A is 0x3E):
{0x02 0x00} {0x00} {0x01} {0x3E} {0xE4 0xAC}

The manufacturer of the WIFI module we are going to use, the ESP8266, has provided libraries that can be used to send and receive data from the WIFI following the IEEE 802.11b protocol [8]. There are two types of packet formats: long and short. The short one is used for the maximum data throughput but the long one is mandatory and that is what is described below. The packet consists of three parts:

1. Preamble: This consists of two sections
    a. Synch (128 bit) : It is composed of 128 bits with scrambles 1s used for synchronization of the receiver.
    b. SFB (16 bit): A 16 bit sequence that indicates the beginning of the frame
2. Header: This consists of four sections
    a. Signal (8 bit): It indicates the payload data rate
    b. Service (8 bit): It is reserved for future use, set to 0s
    c. Length (16 bit): It indicates the length of the payload
    d. Control error field (16 bit): It is used for error detection in the header.
3. PSDU: The layer that contains the data to be transferred.

4.0 Sources Cited:

[1] STMicroelectronics. *STM32L4: PWR* [Online]. Available: https://www.st.com/content/ccc/resource/training/technical/product_training/ce/57/a3/86/7a/3d/4d/87/STM32L4_System_Power.pdf/files/STM32L4_System_Power.pdf/jcr:content/translations/en.STM32L4_System_Power.pdf

[2] Wikipedia. *Observer pattern* [Online]. Available: https://en.wikipedia.org/wiki/Observer_pattern

[3] JSON[3]. *Introducing JSON* [Online]. Available: https://www.json.org/json-en.html

[4] SeedStudio. *Grove- Human Presence Sensor (AK9753)*[Online]. Available: https://wiki.seeedstudio.com/Grove-Human_Presence_Sensor-AK9753/

[5] RAio. *Character/Graphic TFT LCD Controller Specification* [Online]. Available: https://cdn-shop.adafruit.com/datasheets/RA8875_DS_V19_Eng.pdf

[6] AsahiKASEI. *AK9753 IR Sensor IC with I2C I/F* [Online]. Available: https://files.seeedstudio.com/wiki/Grove-Human_Presence_Sensor-AK9753/res/AK9753.pdf

[7] Waveshare. *Barcode Scanner Module User Manual* [Online]. Available: https://www.waveshare.com/w/upload/d/dd/Barcode_Scanner_Module_Setting_Manual_EN.pdf

[8] Tektronix. *Wi-Fi: Overview of the 802.11 Physical Layer and Transmitter Measurements* [Online]. Available: https://public.cnrood.com/public/docs/WiFi_Physical_Layer_and_Transm_Meas.pdf

Appendix 1: Program Flowcharts

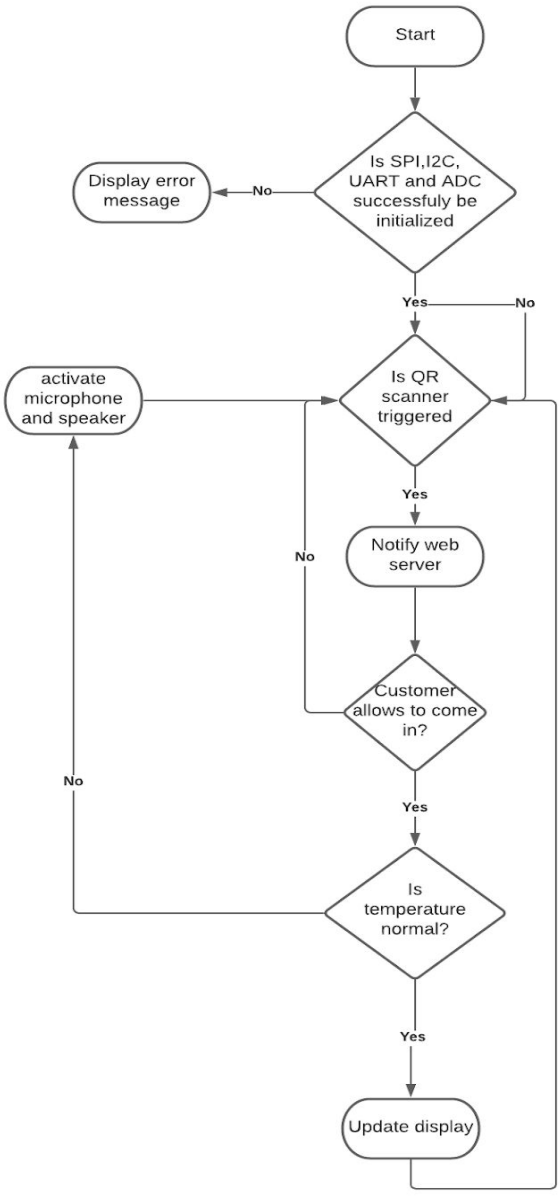

Figure 3.6 - Customer checking in

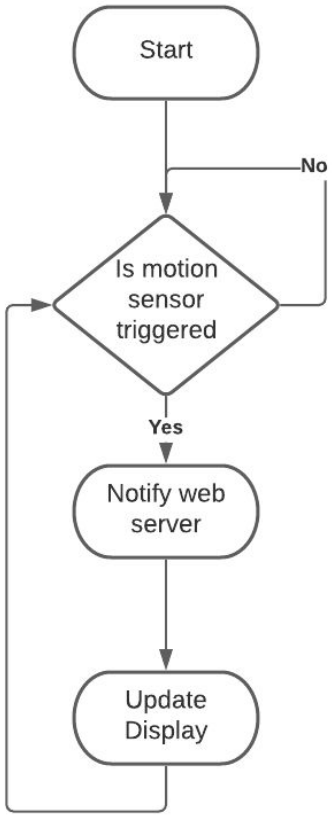

Figure 3.6 - Customer checking out

Appendix 2: State Machine Diagrams

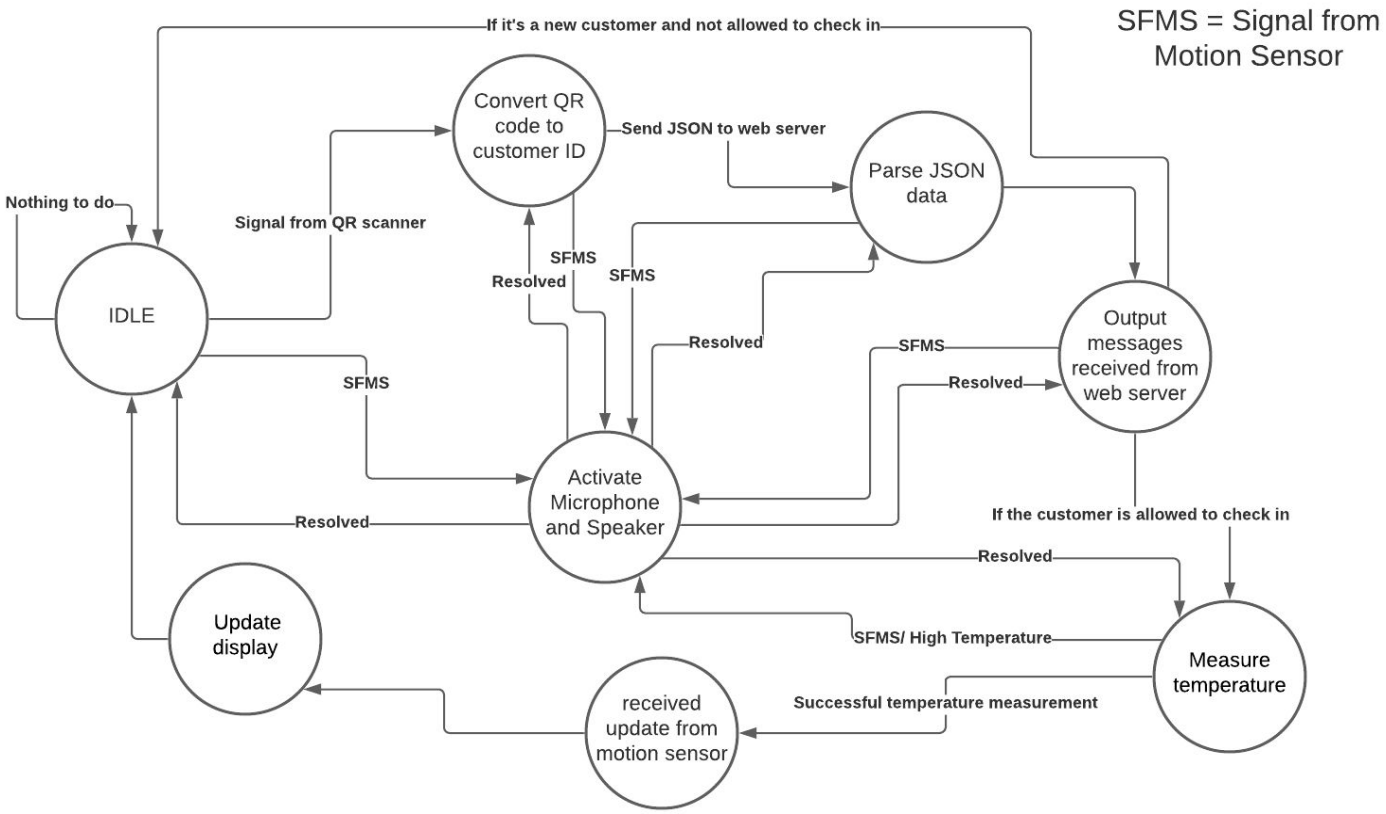

Figure 3.7- State Machine for Checking In

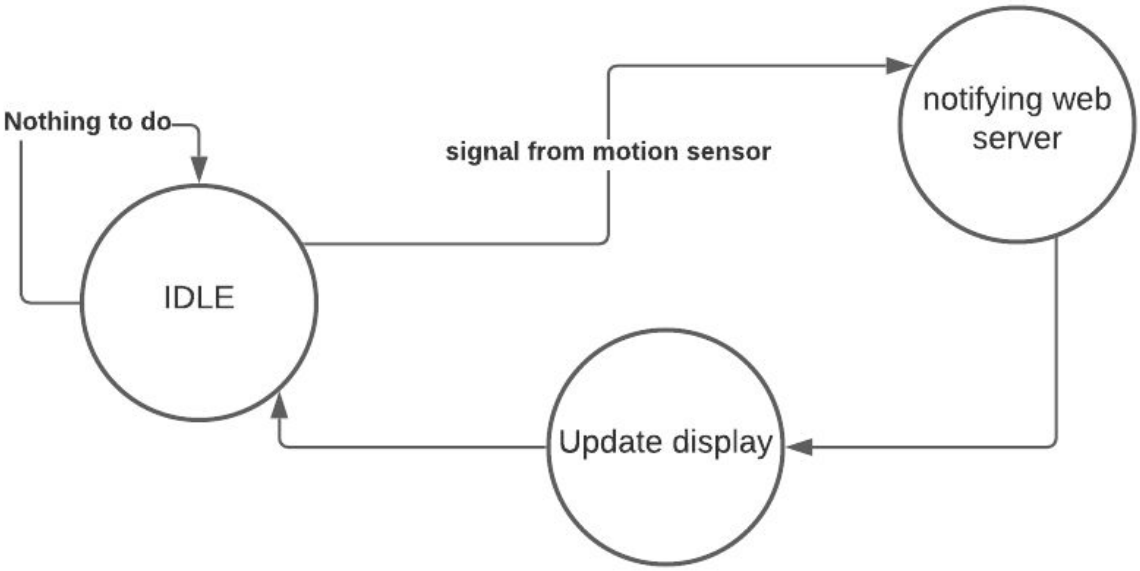

Figure 3.8 - State Machine for Checking Out