

IOT SMART HOME SYSTEM

Progress Journal

Phebe Le

Robotics and Mechatronics

31/7/2023 - 18/8/23

Overview:

This project will be broken into two sections. These being:

1. The outside of the smart home will be a security system that will require a high reading from a force sensor and motion sensor to activate the servo-powered door. If motion is detected a LED will turn on. This door can also be controlled from a web server for accessibility.
2. The inside of the smart home will contain three unique sections. These will all be combined together
 - a. A LED system: LEDs will turn on sequentially depending on the location of object in the smart home. Lights can also be controlled by webserver where the modes: all_on (all LEDs on), all_off (all LEDs off), all_wave (wave mode for LEDs). This mode aspect will only occur if time permits.
 - b. Monitoring system: An LCD screen displays all the action occurring and is the physical display for all the data sent to the SD card
 - c. Voice system: A mic will control music from a piezo speaker. When the microphone records a sound, an LED will light up and the piezo speaker will play a song. The music can be turned on and off from a web server

These 2 sections will be segregated into sections which focus on new technical elements. This will include

- Self-defined library: Voice System
- Multi-dimensional array: LED system
- Stack and queues: Security system
- Pointers: Voice system, Security system
- Ethernet webserver: All systems
- Read and writing to a file: inside of smart home

Through this documentation, videos will be used for main source of viewing data.

Section 1: Outside of Smart Home

Basic structure of the code with two sensors:

The code itself is structured using two sensors to control the opening of the servo-powered door. Whilst the code structure aligns perfectly with the goal, the lack of complexity and alignment to web servers may cause an issue. As a result, a new iteration of this code architecture will be required.

The new iteration of this goal will be to replace the force sensor with a web server button. This will all be recorded through the web server and placed into a micro-SD card.

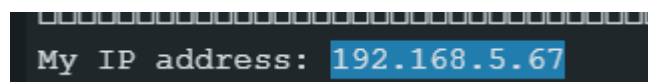
First Iteration:

The first step to create this is to test the modules in a working environment that aligns with the goals of this section. For this, the testing of the motion sensor, servo motor and light will be conducted. The logic will be if the motion sensor detects movement, the servo will spin 90 degrees and the LED light will turn on.

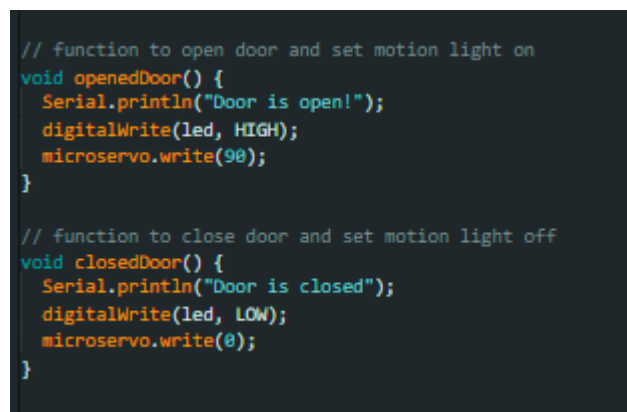
This showed to give a positive result as it did this through numerous counts of testing. An anomaly that occurred was the detection ('opening') often lasted for 0.5 seconds before resetting itself to the 'closed' state. As this result does not affect the integrity of the design or bug the process, this will be a considered a successful iteration.

Second iteration:

The second iteration is to add a working webserver. The first step is to find the required mac and Ip address. To get access to the mac and Ip address of the ethernet shield, an ethernet plug has to be plugged from the Arduino Ethernet Board to the router. This is because it then provides a direct access route that would allow an internet connection to occur. To find the individual mac and Ip address logically the Arduino example named 'DHCP AddressPrinter'. Below the Ip address that was found can be seen. This is then typed into the URL bar. The webserver that was used was based upon an output display system created in my previous assignment, yet this was catered to my current project needs (see image of the server before the serial monitor outputs below).

A screenshot of a serial monitor window. The text "My IP address: 192.168.5.67" is displayed. The IP address "192.168.5.67" is highlighted in blue.

The next step was to create the open and close door function for if the motion sensor detected motion open the door, else close it. This was created through functions.

A screenshot of an Arduino IDE showing two functions: openDoor() and closeDoor(). The openDoor() function sets the LED to HIGH and the servo to 90 degrees. The closeDoor() function sets the LED to LOW and the servo to 0 degrees.

```
// function to open door and set motion light on
void openDoor() {
  Serial.println("Door is open!");
  digitalWrite(led, HIGH);
  microservo.write(90);
}

// function to close door and set motion light off
void closeDoor() {
  Serial.println("Door is closed");
  digitalWrite(led, LOW);
  microservo.write(0);
}
```

This led to the initial testing on the main code which showed to be unsuccessful.

- The reasoning as to why the initial start up of the web server failed was due to network connectivity issues that delayed the web server and caused it to timeout. Below is the image of the initial web server and the two different Serial Monitor outputs. The first serial monitor output is the initialisation between the client and the network. This part was mainly used for debugging and seeing what errors were occurring that shouldn't. For this the main error was that nothing was

occurring after the 'connected' and 'available' output. This was later fixed by moving it and restarting the run time.

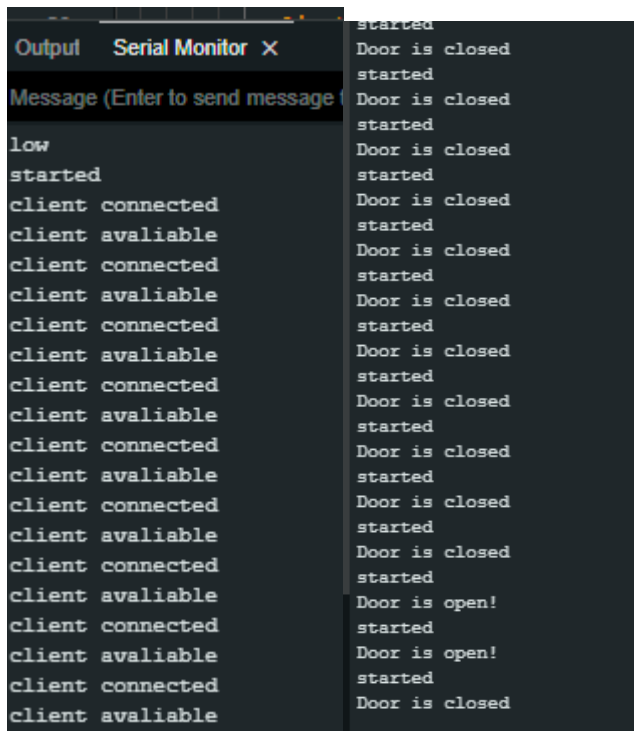
- The second serial monitor output is the opening and closing of the door based on the detection of the motion sensor. This showed to be both successful and unsuccessful in terms of the opening and closing of the door. The specifications of this test were that if motion was detected it would open. Alongside this if a button on the webserver (meant to open the door, labelled as turn off led) was pressed, then the door would also open. The test for the first specification showed to be successful with the door opening when motion was detected. But the second test criteria failed to work. When the webserver button was pressed, a new page would open, and nothing would register within the Arduino. This showed to be detrimental and impossible to change. Given the goal of this assignment is to create a Smart home, a new goal had to be established. One that removed the motion sensor and allowed the web server to work its intended use.

Security Door

Door Status:

Turn Off LED

Created by Phebe Le in Session 2 2023



(See first implementation of code in Arduino Testing folder labelled as 'Code 1')

New goal:

Due to the error with the motion sensor and the effect that it had when trying to implement the required webserver, the motion sensor was removed. As a result, the new goal is controlling the security of a door logically through webserver buttons and physically through a physical button. There will be an LED to show that a door is open or closed and a piezo will be used as a doorbell like aspect for playing sound (this will only be activated through web server). Other functions such as a quick door close may be used as well.

Testing of new goal

The first step was to replace the motion sensor with the input button in order to activate/test the physical security feature. This was created by changing the previous code. Here issues were faced due to the not calling the functions. Within the code, the physical door is opened through a function (is closed by another statement), that states that if the button input is read as high then open the door and print a 'door opened' statement for indication. Yet by not calling this function, this cause errors in nothing working. This showed to be successful when called. This then meant that the webserver could be added.

```
void physicalOpen() {
  buttonState = digitalRead(doorButton);
  if (buttonState == HIGH) {
    openedDoor();
    Serial.println("Door has been physically opened");
  }
}
```

Adding different webserver buttons:

A new webserver was used, due to the lack of input/output availability (unable to use buttons only able to show data). Using the code from 'RandomNerdTutorials', the page loaded with no issues and the CSS file worked. As a result, buttons were added to control the opening and closing of the door. Showing to be successful, the implementation of more buttons was added. The buttons which I wanted to include were a 'turn on/off LED' as the function could be useful if implemented in a real-life situation. Alongside this I wanted to include a 'quick close' button for when somebody at the entrance is not welcome yet meeting them is required, and a music button for comedic sake (in a real-life situation, a webserver-controlled loudspeaker). The documentation for each button is shown below.

LED button:

The LED buttons were created through the normal setup and initialisation of defining the pin variable, the pinMode of the pin, and the digitalWrite of the pin was written within the ethernet button structure. When tested these were successful in turning the LED on and off from the web server. The code snippet can be seen below.

```
122 //methods to control buttons
123 if (readString.indexOf("?button1on") > 0) { //turn on LED
124     digitalWrite(led, HIGH);
125 }
126 if (readString.indexOf("?button1off") > 0) { //turn off LED
127     digitalWrite(led, LOW);
128 }
```

Quick-close button:

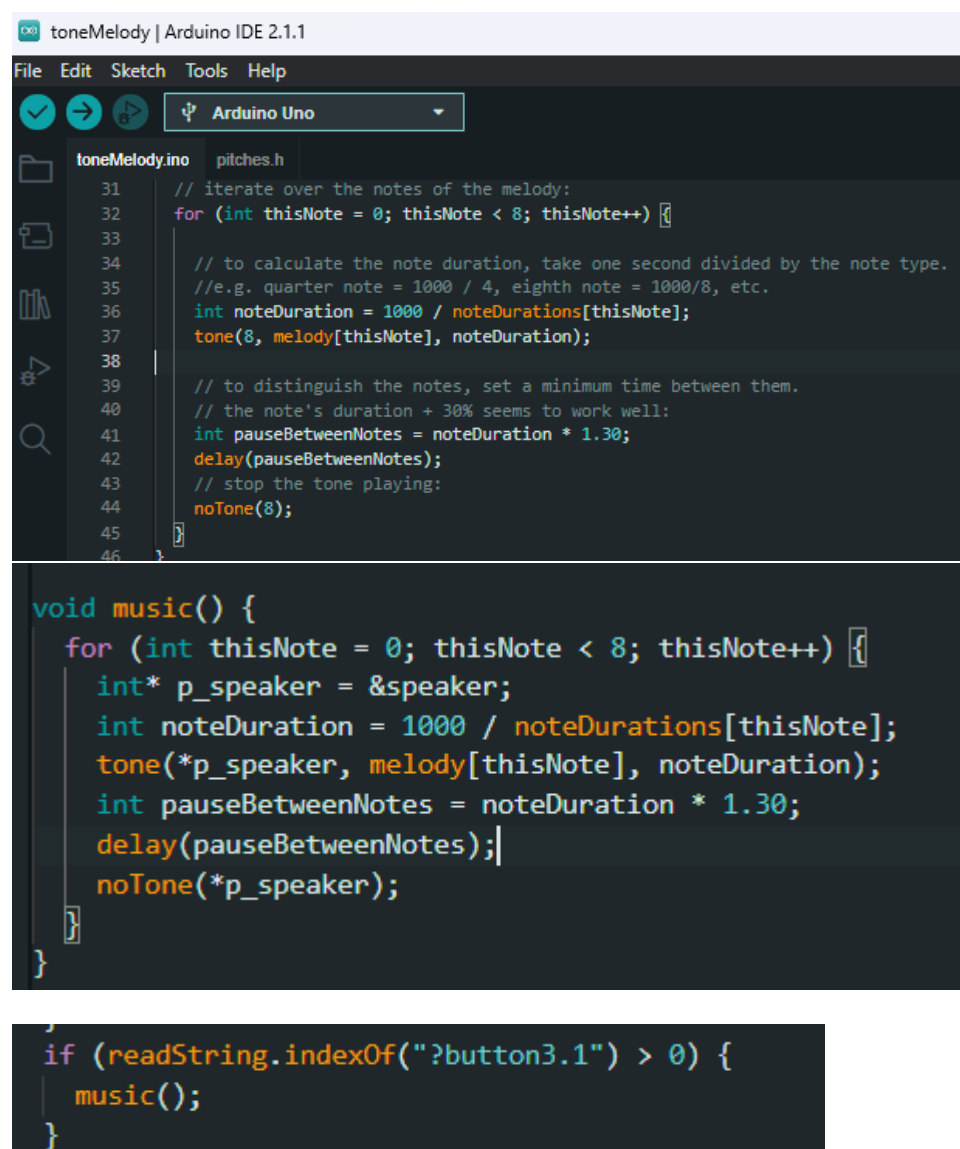
To create the quick close button, the changing the servo direction is required. This was established by using for statements. These worked in the intended way of opening it then closing it. These were written inside of a function to reduce the code in the main code.

```
if (readString.indexOf("?button2off") > 0) {
    for(pos = 180; pos >= 1; pos -= 3) //
    {
        microservo.write(pos);
        delay(15);}
    for(pos = 0; pos < 180; pos += 3) //
    {
        microservo.write(pos);
        delay(15);
    }
}
```

```
if (readString.indexOf("?button3.2") > 0) {
    quickOpen();
}
```

Music button:

The music button will be created by using an Arduino example that integrates the piezo library. By using this I aim to show the use of pointer and dereferencing. The first step for this was implementing the Arduino example into the current code to make sure it worked before changing the variables to accustom the pointers. Through this step, it showed to be successful. The next step was to implement the pointers. I structured the code into a function for simplicity's sake which can be seen below. The inclusion of the piezo library was done by creating a new tab within the Arduino sketch and implementing the music notes from the Arduino official documentation. The images listed below can be judged based on the order of: Arduino example with library, self-created function that integrates pointers based upon the example, the web server button that calls the function



```
toneMelody | Arduino IDE 2.1.1
File Edit Sketch Tools Help
Arduino Uno
toneMelody.ino pitches.h
31 // iterate over the notes of the melody:
32 for (int thisNote = 0; thisNote < 8; thisNote++) {
33
34 // to calculate the note duration, take one second divided by the note type.
35 //e.g. quarter note = 1000 / 4, eighth note = 1000/8, etc.
36 int noteDuration = 1000 / noteDurations[thisNote];
37 tone(8, melody[thisNote], noteDuration);
38
39 // to distinguish the notes, set a minimum time between them.
40 // the note's duration + 30% seems to work well:
41 int pauseBetweenNotes = noteDuration * 1.30;
42 delay(pauseBetweenNotes);
43 // stop the tone playing:
44 noTone(8);
45 }
46 }

void music() {
  for (int thisNote = 0; thisNote < 8; thisNote++) {
    int* p_speaker = &speaker;
    int noteDuration = 1000 / noteDurations[thisNote];
    tone(*p_speaker, melody[thisNote], noteDuration);
    int pauseBetweenNotes = noteDuration * 1.30;
    delay(pauseBetweenNotes);
    noTone(*p_speaker);
  }
}

if (readString.indexOf("?button3.1") > 0) {
  music();
}
```

Testing all buttons, logical and physical:

This test was conducted through a video and showed to be successful.

As a result, the next (unnecessary) setup is the implementation of further data structures. These being stack and queue. I initially came up with this idea after I viewed a YouTube video that went into depth about different things that c programmers should learn. As a result, I wanted to do this. Although this may not work, I will do my best to try and attempt it. Alongside this, I also wanted to implement this to increase the complexity of the code and to view the different incrementation for the servo position for each iteration. Essentially storing past data and then being able to view it after a new iteration (when a button is pressed once, then twice). Whilst this is the main reason for implementing the two data structures, there are many other possibilities of implementing stacks and queues such as if there was more, more complex logic could be applied such as detecting and showing data/activity of the servo through an output source.

Implementation of queue and stack:

Research:

- The process of finding documentation for this concept is quite difficult (for Arduino). Most resources provided are from Stack Overflow (website), this showed that this is possible. Within this process, many individuals listed libraries and files to look to be able to do this. This led me to the official documentation. This official documentation showed that the enqueue() and dequeue() were included, the main elements. After finding this I went to the c documentation of queues (Geeks for geeks) and went read about enqueue(). Here it stated that enqueue should be portrayed through an if statement to check if the queue is full or not, if not add the variable to it. As this was understandable to me, I moved onto the next part, which is displaying the actual data, there was no official Arduino code online for this particular reference, as a result, I planned to go against documentation and serial. Print the variable that I reference in the enqueue.
- The process of finding documentation of stack similarly difficult, so I relied on the geeks for geeks site. This seemed to show the same process as enqueue(), so I intend to replace the variables and function name with somethings else but copy the same structure as the enqueue() code.

Implementation: Here the implementation was completed in a separate file so that testing wouldn't ruin the file. Since I'm testing for data, I will use a servo so it's easier to check if I can't do it. This was successful in showing the incrementing stacking and queuing as seen in the screen shot below. The code used is shown below as well.

Reflecting: This entire process of implementing queues and stacks revolved around the ideology of research, presume a particular concept, implement it, find the errors and try again from the beginning. This process could have been improved by asking people for help.


```

Stack: 124
Printing Queue:
Queue: 124
Printing stack:
Stack: 124 121
Printing Queue:
Queue: 124 121
Printing stack:
Stack: 124 121 123
Printing Queue:
Queue: 124 121 123
Printing stack:
Stack: 124 121 123 152
Printing Queue:
Queue: 124 121 123 152
Printing stack:
Stack: 124 121 123 152 125

```

```

queue_array.ino
1  #include <Servo.h>
2
3  const int stackSize = 10;
4  int stack[stackSize];
5  int stackIndex = 0;
6
7  const int queueSize = 10;
8  int queue[queueSize];
9  int queueFront = 0;
10 int queueRear = 0;
11
12 Servo microservo;
13 bool motionDetection = false;
14 int pos = 0;
15
16 void setup() {
17   Serial.begin(9600);
18   microservo.attach(13);
19 }
20
21 void loop() {
22   for (int x = 0; x < 4; x++){
23     quickOpen();
24   }
25
26   void pushStack(int value) {
27     if (stackIndex < stackSize) {
28       stack[stackIndex] = value;
29       stackIndex++;
30     } else {
31     }
32   }
33
34   void enqueueQueue(int value) {
35     if ((queueRear + 1) % queueSize != queueFront) {
36       queue[queueRear] = value;
37       queueRear = (queueRear + 1) % queueSize;
38     } else {
39       // Handle queue overflow
40     }
41   }
42   void printStack() {
43     Serial.print("Stack: ");
44     for (int i = 0; i < stackIndex; i++) {

```

```

queue_array.ino
~
~
39     // Handle queue overflow
40 }
41 }
42 void printStack() {
43     Serial.print("Stack: ");
44     for (int i = 0; i < stackIndex; i++) {
45         Serial.print(stack[i]);
46         Serial.print(" ");
47     }
48     Serial.println();
49 }
50
51 void printQueue() {
52     Serial.print("Queue: ");
53     int currentIndex = queueFront;
54     while (currentIndex != queueRear) {
55         Serial.print(queue[currentIndex]);
56         Serial.print(" ");
57         currentIndex = (currentIndex + 1) % queueSize;
58     }
59     Serial.println();
60 }
61
62 void quickOpen() {
63     int cat = analogRead(pos);
64
65     enqueueQueue(cat);
66     pushStack(cat);
67     Serial.println("Printing stack:");
68     printStack();
69     Serial.println("Printing Queue:");
70     printQueue();
71
72     // make the quick open happen (servo go from 0-180 in 15ms)
73     for (cat = 0; cat < 90; cat += 3) {
74         microservo.write(cat);
75         delay(20);
76     }
77     for (cat = 0; cat < 90; cat += 3) {
78         microservo.write(cat);
79         delay(15);
80     }
81 }
82

```

Through this section, there were over 20 errors as I had never implemented this before. Due to the scale of mistakes made, I will list the main categories that I had trouble with through a dot point structure and explain the problem and how it was solved in brief terms

- Variables: The variables were difficult to understand because of the different programming languages. Yet, this will be understood through research
- Structure of functions: Documentation on these was limited to Arduino in the sources which were looked at. This meant most of this was guess and tested. Functions contained elements or different variables that were not understood properly and hence were constructed incorrectly. These were solved through a guess, check, research loop until there were no errors
- Structure of functions against each other: This error was that different functions called before they were initialised, this meant many functions had to be moved around to accommodate this.
- Servo and linkage: All the documentation that was looked at did not contain sensors, or hardware inputs and as a result I tried to push variables directly from the servo (e.g. if (pos < queue + queueSize) {}) this was not correct and failed to work. There were other similar mistakes to this as well. To solve it I read a GitHub repository named "SimpleQueue.ino" and presumed that when a variable was placed inside a data structure like 'enqueue()', it would work in the same way as if I did a 'digitalRead' or a similar function.

Further implementation: with this code seemingly now work, my next thought is what if I put something I already knew into it, this being pointers. As a result, this is done below. The code is shown below and placed into the file folder.

Through this process like the tasks before numerous mistakes were made. Some of the mistakes were

- Incorrected initialisation of pointer (the [0] was not used till several iterations later)
- Incorrect incrementation of the pointer ("stackPointer++/queuePointer++" had the incorrect incrementation (wrong variable was used to increment))
- Printing: the wrong variable was used to print the stack and queue and this lead to numerous compiler issues. This was fixed through looking at resources that used pointers within stack and queue.

```
// Stack size and pointer for the stack
const int stackSize = 10;
int stack[stackSize];
int* stackPtr = &stack[0];
int* stackTop = &stack[0];

// Queue size and pointer for the queue
const int queueSize = 10;
int queue[queueSize];
int* queuePtr = &queue[0];
int* queueFront = &queue[0];
int* queueRear = &queue[0];
```

```

void pushStack(int value) {
    // Check if the stack is not full
    if (stackPtr < stack + stackSize) {
        *stackPtr = value;
        stackPtr++;
    } else {
        // Stack overflow, handle as needed (e.g., pop elements to make space)
    }
}

void enqueueQueue(int value) {
    // Check if the queue is not full
    if (queuePtr < queue + queueSize) {
        *queuePtr = value;
        queuePtr++;
    } else {
        // Queue overflow, handle as needed (e.g., dequeue elements to make space)
    }
}

void printStack() {
    Serial.print("Stack: ");
    int* ptr = stack;
    while (ptr < stackPtr) {
        Serial.print(*ptr);
        Serial.print(" ");
        ptr++;
    }
    Serial.println();
}

void printQueue() {
    Serial.print("Queue: ");
    int* ptr = queueFront;
    while (ptr < queuePtr) {
        Serial.print(*ptr);
        Serial.print(" ");
        ptr++;
    }
    Serial.println();
}

```

Security Door

Door Status:

Turn On LED

Turn Off LED

Open door

Close Door

Play music

Quick Open

Created by Phebe Le in Session 2 2023

Outputted Serial monitor after beign successful.

```

Printing stack:
Stack: 122
Printing Queue:
Queue: 122
Printing stack:
Stack: 122 119
Printing Queue:
Queue: 122 119
Printing stack:
Stack: 122 119 120
Printing Queue:
Queue: 122 119 120
Printing stack:
Stack: 122 119 120 120
Printing Queue:
Queue: 122 119 120 120
Printing stack:
Stack: 122 119 120 120 120
Printing Queue:
Queue: 122 119 120 120 120
Printing stack:
Stack: 122 119 120 120 120 119
Printing Queue:
Queue: 122 119 120 120 120 119
Printing stack:
Stack: 122 119 120 120 120 119 126
Printing Queue:
Queue: 122 119 120 120 120 119 126

```

Due to this success, the implementation of the stack and queue code with the pointers was then placed within the webserver code to replace the quickOpen function. This meant that when the webserver button for quickOpen() was pressed it would initiate the stack and queue code (that contained the movement of the servo). Below is the Serial Monitor output of the stack and queue.

```

GET /?button3.2 HTTP/1.1

Printing stack:
Stack: 452
Printing Queue:
Queue: 452

```

Bibliography:

[programming - Creating a queue in Arduino? - Arduino Stack Exchange](#)

[queue - Difference between "enqueue" and "dequeue" - Stack Overflow](#)

[enqueue and dequeue in C - Coding Ninjas](#)

[SMFSW/Queue: Queue handling library \(designed on Arduino\) \(github.com\)](#)

[programming - Creating a queue in Arduino? - Arduino Stack Exchange](#)

[Arduino_AdvancedAnalog - enqueue\(\) - Arduino Reference](#)

[Arduino Playground - QueueArray Library](#)

[Data-structure/ENQUEUEQUEUE.C at main · Likithapoojari/Data-structure \(github.com\)](#)

[EinarArnason/ArduinoQueue: A lightweight linked list type queue implementation, meant for microcontrollers. \(github.com\)](#)

[Introduction to Queue - Data Structure and Algorithm Tutorials - GeeksforGeeks](#)

[Introduction to Stack - Data Structure and Algorithm Tutorials - GeeksforGeeks](#)

[Complete Guide for Ultrasonic Sensor HC-SR04 with Arduino | Random Nerd Tutorials\](#)

[nc/Robotics Session 2 2023/Health Monitoring System/heartMonitoringSystem_part1.ino at main · musictena/nc \(github.com\)](#)

[Arduino - Webserver with an Arduino + Ethernet Shield | Random Nerd Tutorials](#)

[How to Wire and Program a Button | Arduino Documentation](#)

[Security Door](#)

Full code below:

```
part_1_securitySystem.ino
10  #include <Servo.h>
11
12  //defining sensors
13  const int motionSensor = 2;
14  const int forceSensor = A0;
15
16  //defining outputs
17  Servo servoMotor;
18  const int servoMotor = 3;
19  const int motionLight = 4;
20
21  //values and status of sensors
22  bool motionDetection = false;
23  bool forceDetection = false;
24
25  const int forceThreshold = 500;
26
27  //stack setup
28
29  //queue setup
30
31  void setup() {
32      Serial.begin(9600);
33
34      //setup sensors and outputs
35      servoMotor.attach(servoMotor);
36      pinMode(motionSensor, INPUT);
37      pinMode(motionLight, OUTPUT);
```

```

39 //start by having the door closed
40 closedDoor();
41 }
42
43 void loop() {
44 // check motion sensor first
45 motionDetection = digitalRead(motionSensor);
46 // check force sensor
47 int forceValue = analogRead(forceSensor);
48 forceDetection = (forceValue && forceThreshold);
49
50 // check to see if motion sensor and forc sensor have been activated
51 if (motionDetection && forceDetection){
52     openedDoor();
53 }else{
54     closedDoor();
55 }
56 }
57
58 // function to open door and set motion light on
59 void openedDoor(){
60     Serial.print("Door is open!");
61     pinMode(motionLight, HIGH);
62     servo.write(90);
63 }
64
65 // function to close door and set motion light off
66 void closedDoor(){
67     Serial.print("Door is closed");
68     pinMode(motionLight, LOW);
69     servo.write(0);
70 }
71

```


Section 2: Inside the Smart Home

For this section the focus will be to implement a variety of different aspects which will be combined. These being a MIC system, a monitoring system and a LED system.

Light system:

The light system will be based around creating a multi-dimensional array that incrementally turns on as an object moves closer to an ultrasonic sensor. This will include 6 LEDs and an ultrasonic sensor.

Based on a CoderDojo website, I was able to use the main functionality of the structure of using an ultrasonic sensor with an LED. Yet, due to my need to implement multi-dimensional arrays, much of the code had to be rearranged and edited to suit this need. Below is the original code with a few edits. This will be the base template for my code architecture.

[Ultrasonic sensor with Arduino - Coderdojo Athlone](#)

```
1 // created by phebe le sl
2 //initialisation of main variables
3 const int pins = 8;
4 const int trigPin = 9;
5 const int echoPin = 10;
6
7 // the leds
8 const int numLeds = 6;
9 const int ledPins[] = {2, 3, 4, 5, 6, 7};
10
11 void setup() {
12   Serial.begin(9600);
13
14   //set up inputs and outputs
15   pinMode(trigPin, OUTPUT);
16   pinMode(echoPin, INPUT);
17   for (int i = 0; i < numLeds; i++) {
18     pinMode(ledPins[i], OUTPUT);
19   }
20 }
21
22 void loop() {
23   // create variables limited to loop (opposite of global , google)
24   long duration;
25   long distance;
26
27   ultraSetup();
28
29   duration = pulseIn(echoPin, HIGH);
30   distance = duration * 0.034 / 2; // this number is taken from another website, to ensure accuracy of calculation
31
32   // create the mapping of the leds in regards to the ultrasonic sensor
33   for (int i = 0; i < numLeds; i++) {
34     if (distance >= (i * 10) && distance < ((i + 1) * 10)) {
35       for (int j = 0; j <= i; j++) {
36         digitalWrite(ledPins[j], HIGH);
37       }
38       for (int j = i + 1; j < numLeds; j++) {
39         digitalWrite(ledPins[j], LOW);
40       }
41     }
42   }
43
44   // serial print to debug and check
45   Serial.print("Distance: ");
46   Serial.print(distance);
47   Serial.println(" cm");
48   delay(100);
49 }
50
51 // used to initialise the beginning of the ultrasonic sensor
52 void ultraSetup(){
53   digitalWrite(trigPin, LOW);
54   delayMicroseconds(2);
55   digitalWrite(trigPin, HIGH);
56   delayMicroseconds(10);
57   digitalWrite(trigPin, LOW);
58 }
```

After a successful test, the next step is the place multidimensional arrays in. Yet through this process many syntax and architecture problems occurred. Some errors included forgetting another 'for' statement within the mapping of the LEDs, missing semicolons, brackets, digitalWrite's. Through these errors, this caused the distance measurements to work and print but the LEDs didn't turn on.

```
const int numLeds = 6;
const int ledPins[2][3] = {{2, 3, 4}, {5, 6, 7}};

void setup() {
  Serial.begin(9600);

  //set up inputs and outputs
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  for (int i = 0; i < numLeds; i++) {
    for (int j = 0; j < numLeds; j++) {
      pinMode(ledPins[i][j], OUTPUT);
    }
  }
}

// create the mapping of the leds in regards to the ultraonic sensor
for (int i = 0; i < numLeds; i++) {
  if (distance >= (i * 10) && distance < ((i + 1) * 10)) {
    for (int j = 0; j <= i; j++) {
      for (int k = 0; k <= i; k++) {
        digitalWrite(ledPins[j][k], HIGH);
      }
    }
    for (int j = 0; j <= i; j++) {
      for (int k = 0; k <= i; k++) {
        digitalWrite(ledPins[j][k], LOW);
      }
    }
  }
}
```

Yet this test result also failed to work and resulted in a distance measurement of 0. This error may have occurred because the placement of the multi-dimensional array within the 'for' statement. Due to this, I will remove the [j] and [k] and move them into separate for statements. This will make the code unconventional in the sense that it will be less optimised.

```
void loop() {
  // create variables limited to loop (opposite of global , google)
  long duration;
  long distance;

  ultraSetup();

  duration = pulseIn(echoPin, HIGH);
  distance = duration * 0.034 / 2; // this number is taken from another website, to ensure accuracy

  // create the mapping of the leds in regards to the ultrasonic sensor
  for (int i = 0; i < numLeds; i++) {
    if (distance >= (i * 10) && distance < ((i + 1) * 10)) {
      for (int j = 0; j <= i; j++) {
        digitalWrite(row[j], HIGH);
        for (int k = 0; k <= i; k++) {
          digitalWrite(col[k], HIGH);
        }
      }
      for (int j = i + 1; j < numLeds; j++) {
        digitalWrite(row[j], LOW);
        for (int k = 0; k <= i; k++) {
          digitalWrite(col[k], LOW);
        }
      }
    }
  }

  // serial print to debug and check
  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.println(" cm");
  delay(100);
}

// used to initialize the beginning of the ultrasonic sensor
void ultraSetup() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
}
```

This new iteration showed a positive result that allowed three of the LEDs to work. This is a result of not initialising the column row. As a result, this will be fixed by adding this functionality into the code.

A later problem after seeing this section work was that the column and the row were placed in the wrong order (for statements showed them in incremental order, but it should be the 2nd variable before the first as the for statement iterates the inside loop before the outside loop). As a result, this will be fixed by rearranging the order.

```
// the leds
const int numLeds = 6;
const int row[] = {2,3,4};
const int col[] = {5,6,7};
int ledPins[3][3];

void setup() {
  Serial.begin(9600);

  //set up inputs and outputs
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);

  // initialization
  for (int i = 0; i < numLeds; i++) {
    pinMode(col[i], OUTPUT);
    pinMode(row[i], OUTPUT);
  }
}

void loop() {
  // create variables limited to loop (opposite of global , google)
  long duration;
  long distance;

  ultraSetup();

  duration = pulseIn(echoPin, HIGH);
  distance = duration * 0.034 / 2; // this number is taken from another website, to ensure accuracy of

  // create the mapping of the leds in regards to the ultrasonic sensor
  for (int i = 0; i < numLeds; i++) {
    if (distance >= (i * 10) && distance < ((i + 1) * 10)) {
      for (int j = 0; j < 3; j++) {
        digitalWrite(col[j], HIGH);
        for (int k = 0; k < 3; k++) {
          digitalWrite(row[k], HIGH);
        }
      }
      for (int j = i + 1; j < numLeds; j++) {
        digitalWrite(col[j], LOW);
        for (int k = i + 1; k < 8; k++) {
          digitalWrite(row[k], LOW);
        }
      }
    }
  }

  // serial print to debug and check
  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.println(" cm");
  delay(100);
}

// used to initialize the beginning of the ultrasonic sensor
void ultraSetup() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
```

Since this only uses 10%, a library would be good. Yet in the worst-case scenario, this will be simply implemented into the main code

[Control an 8x8 matrix of LEDs. | Arduino Documentation](#)

[Arduino - Multi-Dimensional Arrays | Tutorialspoint](#)

[Arduino - Multi-Dimensional Arrays | Tutorialspoint](#)

[Create Your Own Arduino Library - The Robotics Back-End \(roboticsbackend.com\)](#)

[Writing a Library for Arduino | Arduino Documentation](#)

[Guide for Microphone Sound Sensor Arduino | Random Nerd Tutorials](#)

Voice system:

Aim: The voice system is to create a mic that can activate sound. This will be done by mic sensor that will have to HIGH alongside a button for sound to be validated. This reduces the risk of having sound play every time there is a slight bit of sound. This task is not essential as it is now not required. This code will be tested but will only be implemented to main code or made into a library if time permits

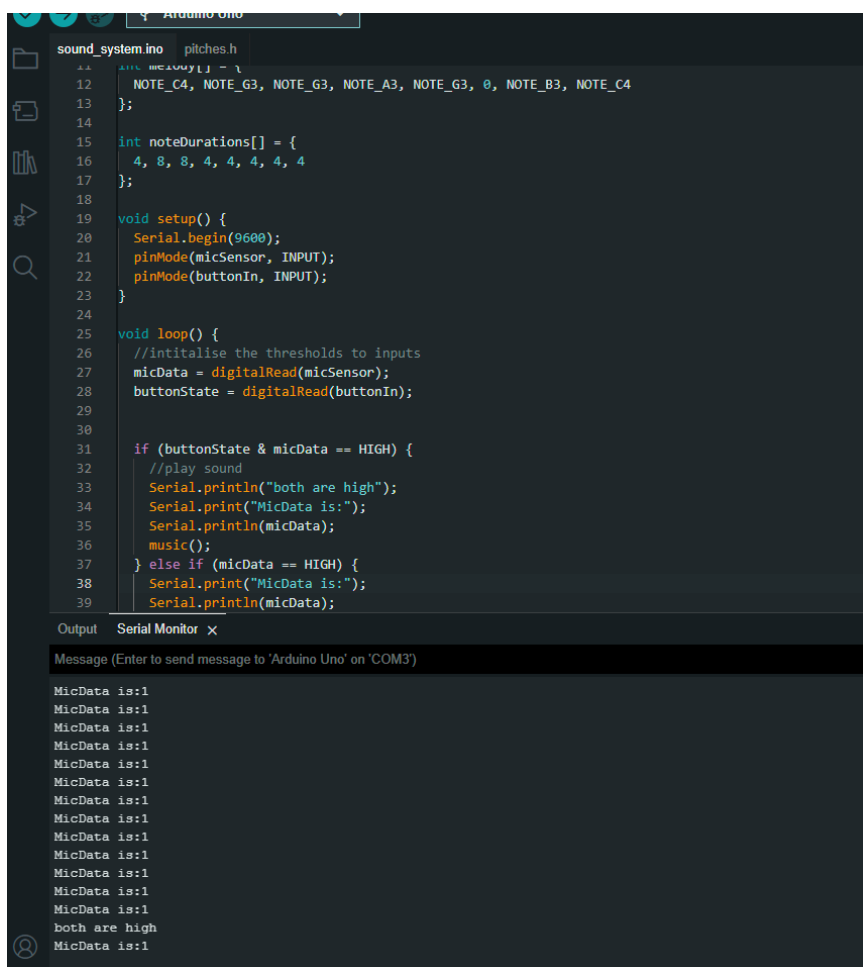
```
sound_system.ino  pitches.h
1  const int micSensor = 9;
2  const int buttonIn = 10;
3  const int speaker = 11;
4
5  int buttonState = 0;
6  boolean micData = 0;
7
8  // notes in the melody:
9  int melody[] = {
10     NOTE_C4, NOTE_G3, NOTE_G3, NOTE_A3, NOTE_G3, 0, NOTE_B3, NOTE_C4
11 };
12
13 int noteDurations[] = {
14     4, 8, 8, 4, 4, 4, 4, 4
15 };
16
17
18 void setup() {
19     Serial.begin(9600);
20     pinMode(micSensor, INPUT);
21     pinMode(buttonIn, INPUT)
22 }
23
24
25 void loop() {
26     //initialise the thresholds to inputs
27     micData = digitalRead(micSensor);
28     buttonState = digitalRead(buttonIn);
29
30
31     if (buttonState == HIGH)&(micData == HIGH){
32         //play sound
33         Serial.println(micData)
34         music()
35     }
36 }
37
38
39 void music() {
40     for (int thisNote = 0; thisNote < 8; thisNote++) {
41         int* p_speaker = &speaker;
42         int noteDuration = 1000 / noteDurations[thisNote];
43         tone(*p_speaker, melody[thisNote], noteDuration);
44         int pauseBetweenNotes = noteDuration * 1.30;
45         delay(pauseBetweenNotes);
46         // stop the tone playing:
47         noTone(*p_speaker);
48     }
49 }
50
51
```

This code will be implemented into the code. As a result, the setup within the void setup was created into a function to reduce the size of the void setup for later purposes whilst implementing it in the webserver.

```
void lightSystem(){
    //set up inputs and outputs
    pinMode(trigPin, OUTPUT);
    pinMode(echoPin, INPUT);

    // initialization
    for (int i = 0; i < numLeds; i++) {
        pinMode(col[i], OUTPUT);
        pinMode(row[i], OUTPUT);
    }
}

// used to initialize the beginning of
```



Through this test, I was able to evaluate that the Mic and the button code was working. Within this test I faced many errors with the incorrect circuitry of the button, not including the required structure for C (';' or capital lettering). When these were fixed, they all worked. This test all showed the buttons and the mic outputted binary numbers instead of digits depending on the loudness of the sound. This seemed strange, and many sources said that it was possible so I will test to try and achieve this.

Through testing I found my error, this being that by putting the mic sensor through an if statement the value of the input changed to 'one' and 'zero' because 'high=1' and 'low = 0'. As a result, I removed the mic from the if statement and this relayed values. Another error within this was that I initialised the pin as digitalWrite instead of analogWrite.

```
32 //play sound
33 Serial.println("both are high");
34 Serial.print("The value of the mic is:");
35 Serial.println(sensorValue);
36 music();
37 }
38 }
39
40
41
42 void music() {
43   for (int thisNote = 0; thisNote < 8; thisNote++) {
44     int* p_speaker = &speaker;
45     int noteDuration = 1000 / noteDurations[thisNote];
46     tone(*p_speaker, melody[thisNote], noteDuration);
47     int pauseBetweenNotes = noteDuration * 1.30;
48     delay(pauseBetweenNotes);
49     // stop the tone playing:
50     noTone(*p_speaker);
51   }
52 }
53
```

Output Serial Monitor X

Message (Enter to send message to 'Arduino Uno' on 'COM3')

```
The value of the mic is:8
both are high
The value of the mic is:128
both are high
The value of the mic is:642
both are high
The value of the mic is:8
both are high
The value of the mic is:423
both are high
The value of the mic is:676
both are high
The value of the mic is:246
both are high
The value of the mic is:601
```

Library for sound (first iteration):

To create the sound library, I attempted to try and follow the techniques used to create class whilst copying and pasting from the workig file which I had. This seemed to fit the visual brief perfectly.

soundSystem.ino	SoundSystem.cpp	SoundSystem.h	pitches.h
1	#include "SoundSystem.h"		
2			
3	const int sensorPin = A0;		
4	const int buttonIn = 11;		
5	const int speaker = 8;		
6			
7	SoundSystem SoundSystem(sensorPin, buttonIn, speaker);		
8			
9	void setup() {		
10	SoundSystem.setup();		
11	}		
12			
13	void loop() {		
14	SoundSystem.loop();		
15	}		
16			

soundSystem.ino	SoundSystem.cpp	SoundSystem.h	pitches.h
1	#include "SoundSystem.h"		
2	#include "pitches.h"		
3			
4	SoundSystem::SoundSystem(int sensorPin, int buttonIn, int speaker) {		
5	_sensorPin = sensorPin;		
6	_buttonIn = buttonIn;		
7	_speaker = speaker;		
8	_buttonState = 0;		
9			
10	// music array		
11	_melody[8] = (_melody, (int[]) { NOTE_C4, NOTE_G3, NOTE_G3, NOTE_A3, NOTE_G3, 0, NOTE_B3, NOTE_C4 }, sizeof(_melody));		
12	_noteDurations[8] = (_noteDurations, (int[]) { 4, 8, 8, 4, 4, 4, 4, 4 }, sizeof(_noteDurations));		
13	}		
14			
15	void SoundSystem::setup() {		
16	pinMode(_sensorPin, INPUT);		
17	pinMode(_buttonIn, INPUT);		
18	Serial.begin(9600);		
19	}		
20			
21	void SoundSystem::loop() {		
22	int sensorValue = analogRead(_sensorPin);		
23	_buttonState = digitalRead(_buttonIn);		
24			
25	if (_buttonState) {		
26	Serial.println("both are high");		
27	Serial.print("The value of the mic is:");		
28	Serial.println(sensorValue);		
29	music();		
30	}		
31	}		
32			
33	void SoundSystem::music() {		
34	for (int thisNote = 0; thisNote < 8; thisNote++) {		
35	int noteDuration = 1000 / _noteDurations[thisNote];		
36	tone(_speaker, _melody[thisNote], noteDuration);		
37	int pauseBetweenNotes = noteDuration * 1.30;		
38	delay(pauseBetweenNotes);		
39	noTone(_speaker);		
40	}		
41	}		
42			

soundSystem.ino	SoundSystem.cpp	SoundSystem.h	pitches.h
1		<code>#ifndef SoundSystem_h</code>	
2		<code>#define SoundSystem_h</code>	
3			
4		<code>#include "Arduino.h"</code>	
5			
6		<code>class SoundSystem {</code>	
7		<code>public:</code>	
8		<code>SoundSystem(int sensorPin, int buttonIn, int speaker);</code>	
9		<code>void setup();</code>	
10		<code>void loop();</code>	
11			
12		<code>private:</code>	
13		<code>int _sensorPin;</code>	
14		<code>int _buttonIn;</code>	
15		<code>int _speaker;</code>	
16		<code>int _buttonState;</code>	
17		<code>int _melody[8]</code>	
18		<code>int _noteDurations[8];</code>	
19			
20		<code>void music();</code>	
21		<code>};</code>	
22			
23		<code>#endif</code>	
24			

soundSystem.ino	SoundSystem.cpp	SoundSystem.h	pitches.h
1			<code>1 *****</code>
2			<code>2 Public Constants</code>
3			<code>3 *****/</code>
4			
5			<code>5 #define NOTE_B0 31</code>
6			<code>6 #define NOTE_C1 33</code>
7			<code>7 #define NOTE_CS1 35</code>
8			<code>8 #define NOTE_D1 37</code>
9			<code>9 #define NOTE_DS1 39</code>
10			<code>10 #define NOTE_E1 41</code>
11			<code>11 #define NOTE_F1 44</code>
12			<code>12 #define NOTE_FS1 46</code>
13			<code>13 #define NOTE_G1 49</code>
14			<code>14 #define NOTE_GS1 52</code>
15			<code>15 #define NOTE_A1 55</code>
16			<code>16 #define NOTE_AS1 58</code>
17			<code>17 #define NOTE_B1 62</code>
18			<code>18 #define NOTE_C2 65</code>
19			<code>19 #define NOTE_CS2 69</code>
20			<code>20 #define NOTE_D2 73</code>
21			<code>21 #define NOTE_DS2 78</code>
22			<code>22 #define NOTE_E2 82</code>
23			<code>23 #define NOTE_F2 87</code>
24			<code>24 #define NOTE_FS2 93</code>
25			<code>25 #define NOTE_G2 98</code>
26			<code>26 #define NOTE_GS2 104</code>
27			<code>27 #define NOTE_A2 110</code>
28			<code>28 #define NOTE_AS2 117</code>
29			<code>29 #define NOTE_B2 123</code>
30			<code>30 #define NOTE_C3 131</code>
31			<code>31 #define NOTE_CS3 139</code>
32			<code>32 #define NOTE_D3 147</code>
33			<code>33 #define NOTE_DS3 156</code>
34			<code>34 #define NOTE_E3 165</code>

Whilst the implementation of this seemed to be valid, when testing the button could only be clicked once to output data, and the sound failed to work, I presume this is because the sound is wrong and is looping the section and not allowing it to run. When the sketch was placed into the class format, the class definition failed to work due to the private variable for the array. As a result, I decided to place the array that was contained within the cpp file into the header file (`_melody[8] = {}`). There was also an error in not including the `itches.h`, so that header file for music notes was included. Yet this failed to work so the array was placed back into cpp file and placed a 'for' statement into the code to initialise it. Yet this showed an error for the different array variables due to not having the scope declared. Reading the error message, I found it aligned with how the `_song` and `_notetime` didn't align with the header file, so `_melody` and `_song` was swapped around in the file, similarly to `_noteTime` and `_noteDuration`. Alongside the datatype was added, which resolved the errors as well. This showed to be successful.

```
// music array
_melody[] = { NOTE_C4, NOTE_G3, NOTE_G3, NOTE_A3, NOTE_G3, 0, NOTE_B3, NOTE_C4 };
_noteDurations[] = { 4, 8, 8, 4, 4, 4, 4, 4 };

for (int i = 0; i < 8; i++) {
    _song[i] = melody[i];
    _noteTime[i] = noteDurations[i];
}

// music array
int song[] = { NOTE_C4, NOTE_G3, NOTE_G3, NOTE_A3, NOTE_G3, 0, NOTE_B3, NOTE_C4 };
int noteTime[] = { 4, 8, 8, 4, 4, 4, 4, 4 };

for (int i = 0; i < 8; i++) {
    _melody[i] = song[i];
    _noteDurations[i] = noteTime[i];
}

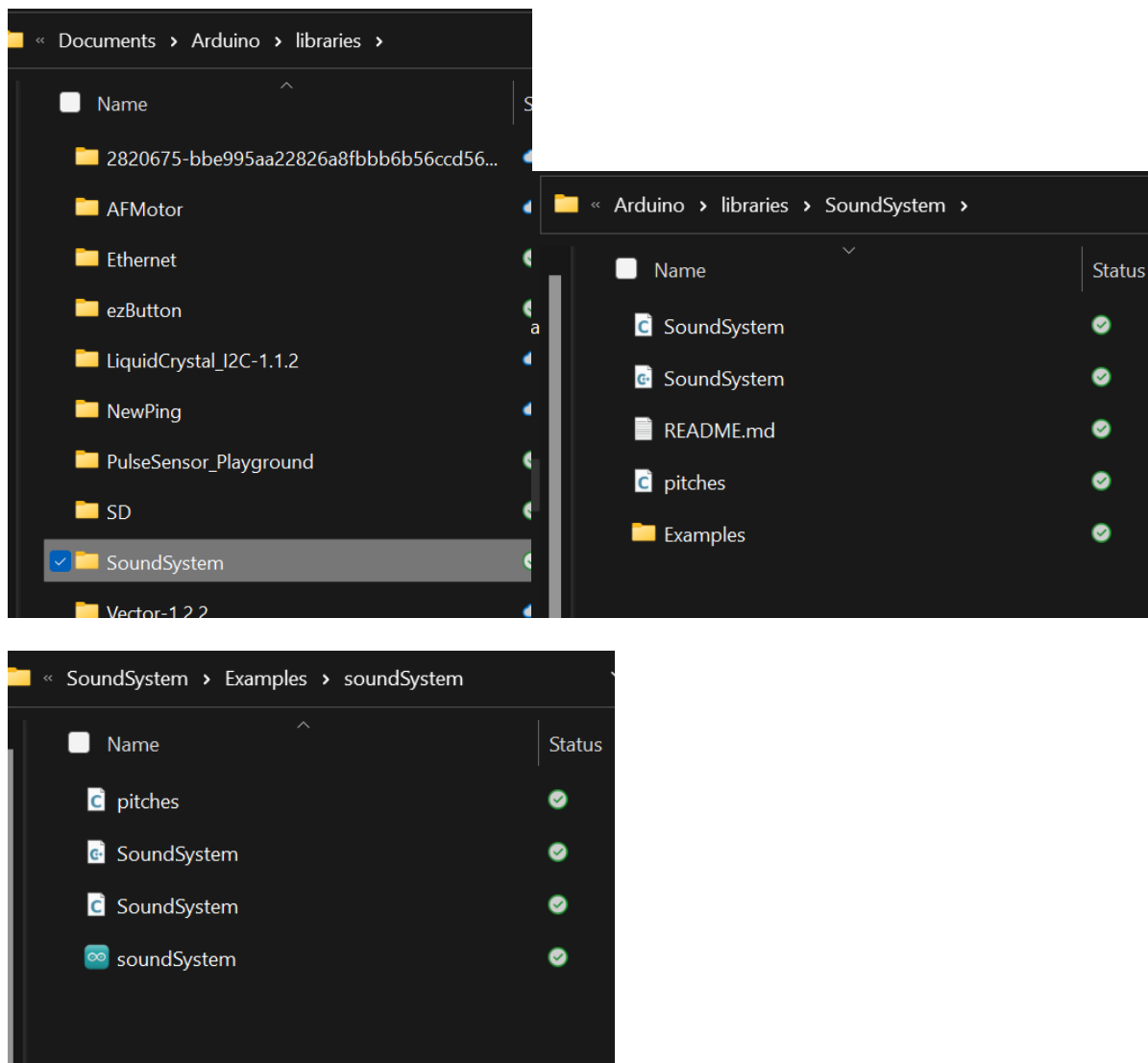
void SoundSystem::setup() {
```

This is the output

for the MIC:

```
The value of the mic is:0
both are high
The value of the mic is:601
both are high
The value of the mic is:935
both are high
The value of the mic is:183
both are high
The value of the mic is:51
both are high
The value of the mic is:221
both are high
The value of the mic is:855
```

To turn it into a library the `ino` file will be removed and all code will be put into the folder within the library



[\(4\) Arduino Scream Machine - YouTube](#)

[programming - Initializing an array within a class - Arduino Stack Exchange](#)

[Arduino Sound Sensor Tutorial - How Sound Sensor works and How to Interface it with Arduino \(circuitdigest.com\)](#)

[Liquid Crystal Displays \(LCD\) with Arduino | Arduino Documentation](#)

[Arduino - Multi-Dimensional Arrays | Tutorialspoint](#)

[Create Your Own Arduino Library - The Robotics Back-End \(roboticsbackend.com\)](#)

LCD monitoring and SD Card Monitoring System:

The LCD and micro-SD card will be created with the use of the sound data which is portrayed. Through this process I was able to integrate, LCD and Sd card monitoring and the light system with each other. This resulted in an output of data, this is shown within the data.txt file.

Combination of all aspects:

The combination of all aspects will be copying and pasting the previous systems and placing them into one file.

The first integration was the sound and the light system which showed to be successful when all aspects were tested. As seen below.

```
Distance: 153 cm
Both MIC and button are available
The value of the mic is: 508
Distance: 154 cm
Distance: 153 cm
Distance: 47 cm
Distance: 69 cm
Distance: 154 cm
Distance: 153 cm
Distance: 154 cm
Distance: 155 cm
Distance: 154 cm
Both MIC and button are available
The value of the mic is: 803
Distance: 156 cm
```

The next test was the sound, light and LCD integration. This also was successful in achieving all aspects that all systems included. Below, the integration of the LCD into the main code can be seen. Yet here, there was a slight problem with the functionality of the LCD. Here the LCD, seemed to have data being written over the top of each iteration of new data. As a result, I chose to add a clearScreen for the beginning of the LCD in the loop that used the LCD (the ultrasonic sensor function). I also printed 'cm' in the LCD for the audience to view the data measurement of the ultrasonic sensor. Below the order of the image can be seen as the initial code written and the new iteration for the LCD clear screen.

```

// serial print to debug and check
Serial.print("Distance: ");
Serial.print(distance);
Serial.println(" cm");
delay(100);

lcd.setCursor(2, 0);    // Set the c
lcd.print("Distance:"); // Print the
lcd.setCursor(2, 1);    //Set the cu
lcd.print(distance);
}

```

```

// serial print to debug and check
Serial.print("Distance: ");
Serial.print(distance);
Serial.println(" cm");
delay(100);

lcd.clear();
lcd.setCursor(2, 0);    // Set the cu
lcd.print("Distance:"); // Print the s
lcd.setCursor(2, 1);    //Set the cur
lcd.print(distance);
lcd.print("cm");

```

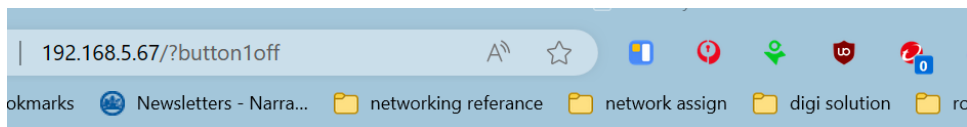
The integration of the web server and the code:

Through this task, I found numerous errors. This is documented below with the way that it was solved each time.

- Problem:
 - Inclusion of the Ethernet: the wrong SPI pin was used, but this was not known until several hours later of testing and debugging other aspects such as incorrect placement of multidimensional array
 - Faulty referencing within the function quickopen() and the local and global variables
 - The use of the SPI pin for speaker and button, which led to them being HIGH, constantly

- Unable to display data, because of the continuous number of outputs from other outputs
- Due to my inability to display data through the LED, I thought the best way to solve it was by creating a new library for the lights
- This was successful, after several iterative debugging
- After being able to integrate the methods of the light System, I tried to integrate the LCD in as, the LCD relied on the distance variable that was inly used in the quickopen() function.
- Yet this failed to work and I wasn't able to integrate LCD in due to the LCD being a part of a class and two classes cannot be put within each other As a result I chose to break the library and the function back up, so that the function quickOpen was used in the main file whilst the initiation of variables was used through the classes.
- As a result, I was required to match the function up with the initiation of the class

The connection to the web server, was valid yet



IOT Home

Door Status:

Turn On LED

Turn Off LED

Created by Phebe Le in Session 2 2023

```
Distance: 5 cm
Distance: 5 cm
Distance: 5 cm
Distance: 5 cm
Distance: 7 cm
Distance: 7 cm
Distance: 6 cm
Distance: 6 cm
Both MIC and button are available
The value of the mic is: 194
Distance: 6 cm
Distance: 6 cm
Distance: 6 cm
Distance: 6 cm
Distance: 6 cm
```

soundSystem.ino	SoundSystem.cpp	SoundSystem.h	pitches.h
1	#include "SoundSystem.h"		
2			
3	const int sensorPin = A0;		
4	const int buttonIn = 11;		
5	const int speaker = 8;		
6			
7	SoundSystem SoundSystem(sensorPin, buttonIn, speaker);		
8			
9	void setup() {		
10	SoundSystem.setup();		
11	}		
12			
13	void loop() {		
14	SoundSystem.loop();		
15	}		
16			

soundSystem.ino	SoundSystem.cpp	SoundSystem.h	pitches.h
1	#include "SoundSystem.h"		
2	#include "pitches.h"		
3			
4	SoundSystem::SoundSystem(int sensorPin, int buttonIn, int speaker) {		
5	_sensorPin = sensorPin;		
6	_buttonIn = buttonIn;		
7	_speaker = speaker;		
8	_buttonState = 0;		
9			
10	// music array		
11	_melody[8] = (_melody, (int[]) { NOTE_C4, NOTE_G3, NOTE_G3, NOTE_A3, NOTE_G3, 0, NOTE_B3, NOTE_C4 }, sizeof(_melody));		
12	_noteDurations[8] = (_noteDurations, (int[]) { 4, 8, 8, 4, 4, 4, 4, 4 }, sizeof(_noteDurations));		
13	}		
14			
15	void SoundSystem::setup() {		
16	pinMode(_sensorPin, INPUT);		
17	pinMode(_buttonIn, INPUT);		
18	Serial.begin(9600);		
19	}		
20			
21	void SoundSystem::loop() {		
22	int sensorValue = analogRead(_sensorPin);		
23	_buttonState = digitalRead(_buttonIn);		
24			
25	if (_buttonState) {		
26	Serial.println("both are high");		
27	Serial.print("The value of the mic is:");		
28	Serial.println(sensorValue);		
29	music();		
30	}		
31	}		
32			
33	void SoundSystem::music() {		
34	for (int thisNote = 0; thisNote < 8; thisNote++) {		
35	int noteDuration = 1000 / _noteDurations[thisNote];		
36	tone(_speaker, _melody[thisNote], noteDuration);		
37	int pauseBetweenNotes = noteDuration * 1.30;		
38	delay(pauseBetweenNotes);		
39	noTone(_speaker);		
40	}		
41	}		
42			

soundSystem.ino	SoundSystem.cpp	SoundSystem.h	pitches.h
1		<code>#ifndef SoundSystem_h</code>	
2		<code>#define SoundSystem_h</code>	
3			
4		<code>#include "Arduino.h"</code>	
5			
6		<code>class SoundSystem {</code>	
7		<code>public:</code>	
8		<code>SoundSystem(int sensorPin, int buttonIn, int speaker);</code>	
9		<code>void setup();</code>	
10		<code>void loop();</code>	
11			
12		<code>private:</code>	
13		<code>int _sensorPin;</code>	
14		<code>int _buttonIn;</code>	
15		<code>int _speaker;</code>	
16		<code>int _buttonState;</code>	
17		<code>int _melody[8]</code>	
18		<code>int _noteDurations[8];</code>	
19			
20		<code>void music();</code>	
21		<code>};</code>	
22			
23		<code>#endif</code>	
24			

soundSystem.ino	SoundSystem.cpp	SoundSystem.h	pitches.h
1			<code>1 *****</code>
2			<code>2 Public Constants</code>
3			<code>3 *****/</code>
4			
5			<code>5 #define NOTE_B0 31</code>
6			<code>6 #define NOTE_C1 33</code>
7			<code>7 #define NOTE_CS1 35</code>
8			<code>8 #define NOTE_D1 37</code>
9			<code>9 #define NOTE_DS1 39</code>
10			<code>10 #define NOTE_E1 41</code>
11			<code>11 #define NOTE_F1 44</code>
12			<code>12 #define NOTE_FS1 46</code>
13			<code>13 #define NOTE_G1 49</code>
14			<code>14 #define NOTE_GS1 52</code>
15			<code>15 #define NOTE_A1 55</code>
16			<code>16 #define NOTE_AS1 58</code>
17			<code>17 #define NOTE_B1 62</code>
18			<code>18 #define NOTE_C2 65</code>
19			<code>19 #define NOTE_CS2 69</code>
20			<code>20 #define NOTE_D2 73</code>
21			<code>21 #define NOTE_DS2 78</code>
22			<code>22 #define NOTE_E2 82</code>
23			<code>23 #define NOTE_F2 87</code>
24			<code>24 #define NOTE_FS2 93</code>
25			<code>25 #define NOTE_G2 98</code>
26			<code>26 #define NOTE_GS2 104</code>
27			<code>27 #define NOTE_A2 110</code>
28			<code>28 #define NOTE_AS2 117</code>
29			<code>29 #define NOTE_B2 123</code>
30			<code>30 #define NOTE_C3 131</code>
31			<code>31 #define NOTE_CS3 139</code>
32			<code>32 #define NOTE_D3 147</code>
33			<code>33 #define NOTE_DS3 156</code>
34			<code>34 #define NOTE_E3 165</code>

```
// music array
_melody[] = { NOTE_C4, NOTE_G3, NOTE_G3, NOTE_A3, NOTE_G3, 0, NOTE_B3, NOTE_C4 };
_noteDurations[] = { 4, 8, 8, 4, 4, 4, 4, 4 };

for (int i = 0; i < 8; i++) {
  _song[i] = melody[i];
  _noteTime[i] = noteDurations[i];
}
```

Section 3: The housing of the circuit

Aim: The housing of the circuit will be designed based through a house structure. The side of the housing will contain the door, motion sensor and light for the security system of the house. Whilst in the inside an LCD, ultrasonic sensor and an LED will be used placed.

Material: Originally, the material was planned to be created through LEGO, yet due to limited supply and the force required for the door to be opened, cardboard will be used instead. Other external materials include scissors, hot glue, tape,

Structure of the house:



The background information is that it will be in an open space with one room. The right wall will contain the door. The middle wall will contain the LCD screen. The roof will contain the LED and voice sensor. Lastly the inside of the right wall will contain the ultrasonic sensor. The Arduino and Ethernet shield will be placed on top of the circuit.

Security system:



Inside of the home:



The process of iteration occurred, and this is the final model and product.

