INDIVIDUAL PROGRESS REPORT FOR ECE 445

By

Feng Zhao

March 2023

Contents

1. Introduction

The main content of this individual progress report is about the progress of designing and building up the prototype of the Sensor Subsystem and the crib model that we will use for our demo, which is a modified cardboard box. In addition, I was also responsible for the crib control subsystem, whose function was to accept data from the sensor and use the data to compile a bit code and send it to the receiving end through Bluetooth.

2. Design & Verification

2.1. Crib Control Subsystem

The microcontroller is crucial in controlling the entire system. It collects and interprets data from various sensors and makes decisions about the actions the receiver should take based on the data. For instance, it can calculate the distance between the baby and the crib using ultrasonic sensors, collect analog voltage output data from pressure sensors to determine the pressure state of each point on the bed by converting the analog signal to digital signals using an Analog-to-Digital Converter, and use a sound sensor to judge the baby's posture by transmitting an electronic pulse when the sound exceeds the threshold.

More specifically, the microcontroller can determine whether the baby is attempting to climb the crib, which we define as a "dangerous action," based on this sensor data. It can also determine whether the baby is awake, crying, or moving restlessly. After processing all the sensor information, the microcontroller determines the baby's current state and sends a trigger signal to the receiver to sound an alarm. The transmission process relies on the Bluetooth module, an essential component of another control subsystem.

The Bluetooth module communicates data through the digital interface of the MCU. After analyzing the data sent by the sensor subsystem, the MCU sends the information and the required next steps to the receiving end, ensuring timely information processing. One significant advantage of the Bluetooth module is its ability to provide real-time updates, ensuring that parents can track their baby's current status on the screen of the receiving end at any time, avoiding delays in responding to unexpected situations.

2.2. Crib Sensor Subsystem Design & Prototype

2.2.1. Pressure Sensor

In the baby crib design described, pressure sensors detect the baby's weight on the crib's surface. And the pressure sensor we use is SEN0294 from DFRobot. This sensor has a very

fast response time, less than 0.01 seconds, and a trigger weight of 20g, meaning it can detect changes in pressure as small as 0.2N [1].

The pressure sensor array in the crib design consists of three arrays of five pressure sensors each, for a total of fifteen pressure sensors. An N-channel MOSFET controls each array as an amplification and switching transistor. The MOSFET acts as a switch that allows the MCU to activate the sensors in each array one at a time.

The MOSFET is connected to the digital pins of the MCU, which send an activation signal to each row in a serial manner. Once the gate of the MOSFET receives a TTL signal, it will become a pull-down device and generate current from the drain to the source. This will power all five pressure sensors in the array and create a resistance array. The voltage array will be generated correspondingly and sent to the analog pins of the MCU through the connections from col0 to col4.

The current formula for the N-channel MOSFET is [2]:

$$I = K * (V_{GS} - V_{TH})^2$$

where I is the current flowing through the MOSFET, V_{GS} is the voltage between the gate and the source, V_{TH} is the threshold voltage, and K is a constant.

In this design, the MOSFET is being used as a switch, so the current flowing through it will be either 0 or a very small value. The MOSFET's ability to turn on and off quickly and reliably allows the MCU to activate each row of pressure sensors and read the resulting voltage array.

In Layout, the MOSFET I use is FQP30N06L, an N-channel MOSFET with a maximum drain current of 30A and a maximum drain-source voltage of 60V [3]. The pinout of the MOSFET is shown in the image, with the gate pin connected to the digital pin of the MCU, the drain pin connected to the pressure sensors, and the source pin connected to the ground.

In summary, pressure sensors are used in the baby crib design to detect the baby's weight on the crib's surface. The pressure sensor array consists of three arrays of five pressure sensors each, with each array controlled by an N-channel MOSFET.

When I moved the pressure sensor subsystem to a breadboard, I encountered some layout

issues due to the small size of the breadboard. To simplify the wiring layout and make it fit on the breadboard, I made some modifications and added the necessary extension wires.

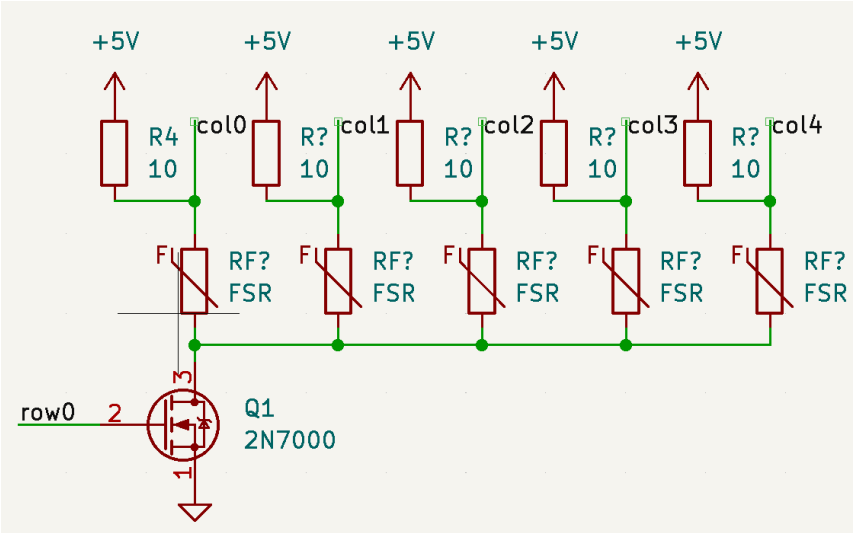

Figure 1. Pressure Sensor Array Layout with N-Channel MOSFET Control

2.2.2. Ultrasonic Sensor

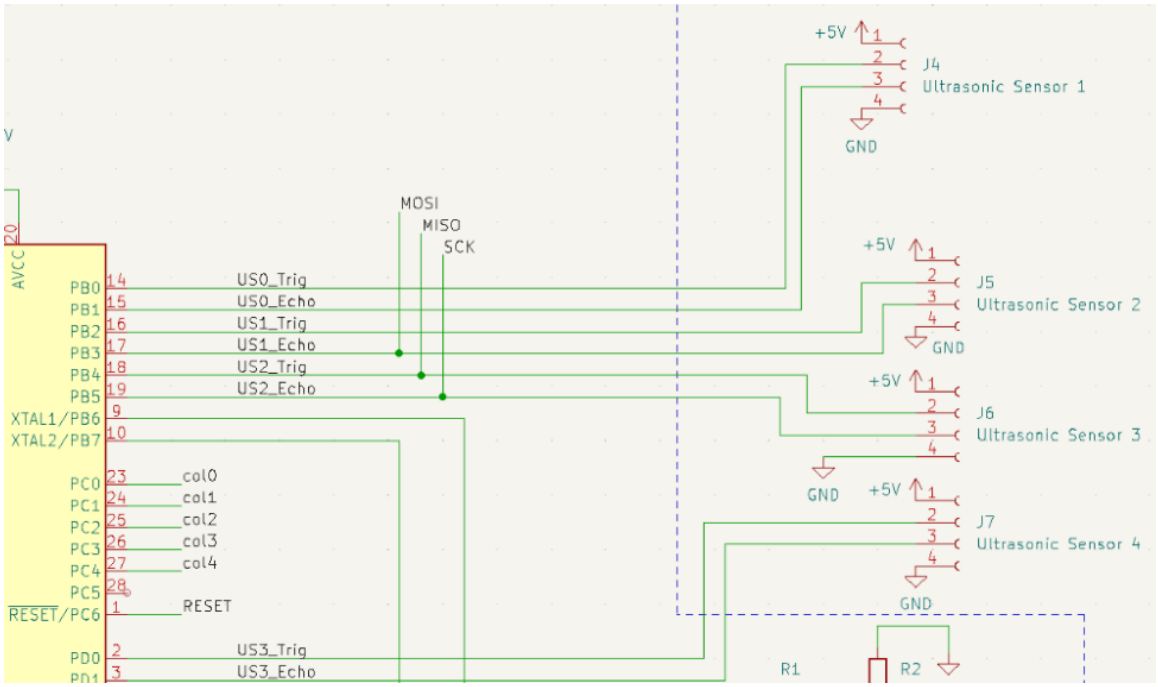

Figure 2. Assembly of Ultrasonic Sensors for Crib Monitoring System

In the sensor subsystem described, ultrasonic sensors measure the distance between themselves and the closest object in front of them. This is achieved by sending out an ultrasound and

measuring the time it takes for the wave to bounce back to the sensor. The MCU in the control subsystem then calculates the distance based on the time recorded and the speed of sound.

The ultrasonic sensors used in the design are the HC-SR04 sensors by EPLZON [4]. These sensors are mounted along the short side of the crib at the height of 20 inches ± 0.5 inches from the bottom of the crib. They are programmed to measure the distance from one short side of the crib to another and alert the MCU if the distance measured is equivalent to 34 inches ± 1 inch. If the distance measured is out of range, the LCD screen will display the "Baby is climbing out of the crib" alert.

The original layout shows that the four ultrasonic sensors are connected to the Arduino board in parallel. Each sensor is connected to the Trig (transmit pulse) and Echo (receive pulse) pins of the Arduino board through a 220-ohm resistor.

Specifically, each ultrasonic sensor's Trig pin is connected to a digital pin of the Arduino board (D2, D3, D4, or D5) through a 220-ohm resistor. Each sensor's Echo pin is connected to an analog pin of the Arduino board (A0, A1, A2, or A3) to read the echo time of the ultrasonic signal. In addition, each ultrasonic sensor is also connected to the 5V power supply and GND pins of the Arduino board to provide power and ground.

When the MCU sends a command to the sensor, it sets the Trig pin to a high state to trigger the transmission of an ultrasound signal. The sensor then sends the signal and waits for it to be reflected. When the reflected sound wave is detected, the sensor sets the Echo pin to a high state. The MCU then detects this high state on the Echo pin and calculates the distance based on the duration of the high state. Similarly, when the MCU receives the distance measurement from the sensor, it sets the digital pins to a high or low state to signal whether the measurement is within an acceptable range. If the distance is within the acceptable range, the MCU may set a digital pin to a high state to indicate that everything is normal. However, if the distance exceeds a specific limit, the MCU may set the digital pin to a low state to trigger an alarm or other corrective action.

Through these connections, the Arduino board can send ultrasonic signals to the sensors and calculate the distance by measuring the echo time. By connecting to digital and analog pins, the Arduino board can read and process the data collected by the sensors and execute corresponding control logic.

2.2.3. Sound Sensor

The sound sensor in the sensor subsystem is the Sparkfun SEN-14262, which detects nearby noise amplitude. The sensor sends an analog signal to the MCU in the crib control subsystem. An analog output type is required to connect the sound sensor to the correct MCU pin. In my design, we utilize a three-pin female connector with the sensor.

According to the datasheet and schematic of the Sparkfun SEN-14262 sound sensor, I decided to connect the Envelope pin to the fifth analog pin of the MCU while connecting the Gate pin to the thirteenth digital pin [5]. This means that the analog voltage signal from the sound sensor's envelope pin is converted to a digital value by the ADC on the MCU and can be read by the MCU as a numerical value. On the other hand, the digital output from the Gate pin indicates when the sound level exceeds a certain threshold and can be read by the digital pin of the MCU as a high or low signal. The MCU can then use this information to trigger alerts or take other actions based on the sound level detected by the sensor.

Once the noise level reaches its threshold, the sensor will send a TTL signal through a digital pin to the MCU, which considers the signal as the baby is crying. The threshold is usually represented by the voltage gain inside the amplifier, which can be tuned manually by a potentiometer or soldering an additional resistance onto the board. The sound sensor's analog output will be converted to a digital signal with an analog-to-digital converter (ADC) on the MCU. In my design, the digital output type is a TTL signal, which is considered a “1” when the voltage is between 2.4 V and 5 V and a “0” when the voltage is between 0 V and 0.4 V.

To verify that the sound sensor works correctly, a consistent sound source needs to be placed at 34 in ± 1 in from the sensor, and a decibel meter is placed within 1 in from the sound sensor. The sound source's volume is adjusted to ensure that the readings from the decibel meter are above 86 dB. We can then verify that the LCD screen shows the “Baby is crying” alert, indicating that the sound sensor correctly reports to the MCU when a sound above 86 dB is measured at 34 in ± 1 in from the sensor.

In summary, the sound sensor is an essential part of the sensor subsystem and provides information about the baby's crying behavior to the MCU. The sound sensor detects the nearby noise amplitude and sends an analog signal to the MCU, converted to a digital signal with an ADC. The sensor sends a TTL signal through a digital pin to the MCU once the noise level reaches its threshold, indicating that the baby is crying.

2.3. Arduino IDE Coding Design

2.3.1. Ultrasonic Sensor Code

```ino
const int trigPin0 = 8;
const int echoPin0 = 2;
const int trigPin1 = 12;
const int echoPin1 = 4;
const int trigPin2 = 13;
const int echoPin2 = 7;

long duration, distance, Distance0, Distance1, Distance2;

void setup() {
pinMode(trigPin0, OUTPUT);
pinMode(echoPin0, INPUT);
pinMode(trigPin1, OUTPUT);
pinMode(echoPin1, INPUT);
pinMode(trigPin2, OUTPUT);
pinMode(echoPin2, INPUT);
Serial.begin(9600);
}
void loop() {
  //multiple ultrasonic sensors
  SonarSensor(trigPin0, echoPin0);
  Distance0 = distance;
  SonarSensor(trigPin1, echoPin1);
  Distance1 = distance;
  SonarSensor(trigPin2, echoPin2);
  Distance2 = distance;
  Serial.print(Distance0);
  Serial.print(" - ");
  Serial.print(Distance1);
  Serial.print(" - ");
  Serial.println(Distance2);
}
```

```ino
void SonarSensor(int trigPin, int echoPin){
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  distance = (duration/2) / 29.1;

  if (distance <= 86.36){
    digitalWrite(trigPin, LOW);
    Serial.println("Warning");
  }
  else{
  }
}
```

Figure 3 & 4. SonarSensor Function & Loop Call

This program defines three sets of pins for three ultrasonic sensors, with two pins for each sensor, one for sending a trigger signal (trigPin) and the other for receiving an echo signal (echoPin). The code also initializes some variables and sets up the Serial communication with a baud rate of 9600.

In the loop function, the SonarSensor function is called three times for each sensor, which sends out a trigger signal, receives an echo signal, and calculates the object's distance from the sensor. The distance is then stored in three different variables (Distance0, Distance1, and Distance2) for three different sensors. Finally, the distances are printed on the Serial Monitor, separated by a dash.

The SonarSensor function sets the trigger pin to LOW for 2 microseconds, then sets it to HIGH for 10 microseconds, and then sets it to LOW again. It then uses the pulseIn function to measure the duration of the echo pulse in microseconds and calculates the distance of the object from the sensor using the formula [6]:

$$distance = (\frac{duration}{2 * 29.1})$$

The function also checks if the distance is less than or equal to 86.36 cm (which is the maximum distance we need for the sensor to detect an object because it is the length of the crib), and if so, it sets the trigger pin to LOW and prints a warning message on the Serial Monitor.

When a baby approaches the ultrasonic sensor, the sensor emits an ultrasonic signal, which reflects from the baby and is received by the sensor. The sensor measures the duration of the echo signal and calculates the distance of the baby from the sensor. The calculated distance is stored in the Distance0, Distance1, and Distance2 variables and sent to the MCU for further processing. The MCU can then use this information to trigger alarms or activate other devices based on the distance measured. The trigger pins will go from LOW to HIGH and back to LOW, and the echo pins will receive the echo signals used to calculate the distance.

2.3.2. Pressure Sensor Code

```
void loop() {
  //int a,b,c,d,e,f,g,h,i,j,k,l,m,n,o;
  int a[3][5];
  int b[3][5];
    //first 1*5
  digitalWrite(trigPin0, HIGH);
  digitalWrite(trigPin1, LOW);
  digitalWrite(trigPin2, LOW);

  delay(100);
  a[0][0]=analogRead(sensorPin0);
  a[0][1]=analogRead(sensorPin1);
  a[0][2]=analogRead(sensorPin2);
  a[0][3]=analogRead(sensorPin3);
  a[0][4]=analogRead(sensorPin4);

  Serial.print("TrigPin0High: ");
  Serial.print(a[0][0]);
  Serial.print("-");
  Serial.print(a[0][1]);
  Serial.print("-");
  Serial.print(a[0][2]);
  Serial.print("-");
  Serial.print(a[0][3]);
  Serial.print("-");
  Serial.println(a[0][4]);

  delay(100);
```

```
for(int i = 0; i <= 2; i++){
  for(int j = 0; j<= 4; j++){
    if(b[i][j] <= 900){
      b[i][j] = 1;
    }else{
      b[i][j] = 0;
    }
    Serial.println(b[i][j]);
    delay(100);
  }
}

int count = 0;

for(int i = 0; i <= 2; i++){
  for(int j = 0; j<= 4; j++){
    int displacement = a[i][j] - b[i][j];
    if(displacement != 0){
      count++;
    }
    delay(100);
  }
}
Serial.print("Count: ");
Serial.println(count);
delay(500);
```

Figure 5 & 6. Pressure Sensor Read Loop & Count Incrementation

This code reads data from a pressure sensor using multiple analog pins. The sensor is divided into three rows, with five elements in each row. Therefore, three rows are read one at a time by the code.

The code reads data from the pressure sensor using the analogRead() function in the loop function and stores it in a 3 * 5 matrix. The trig pins select a particular row of the sensor to read. The code sets one trig pin to high and two to low to select the reading row. After reading the data from a row, the code converts the analog value to a digital value (0 or 1) based on a threshold of 900. Once the code has read the data from all three rows, it compares the two data sets and calculates the number of changed elements between them. This count can be used to detect pressure changes on the sensor.

When a baby applies pressure to the sensor, the analogRead() function will measure the change in voltage across the sensing elements in the matrix. The MCU will convert these analog values to digital values (between 1 to 1023) and compare them with the previous readings to detect any pressure changes on the sensor. The MCU will determine the pressure on a particular sensor when the change in value exceeds a certain proportion.

2.3.3. Sound Sensor Code

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}

//sound sensor
//ama_val = analogRead(PIN_ANALOG_IN);
dig_val = digitalRead(PIN_GATE_IN);
//Serial.println(ana_val);
//Serial.print("Detected Voice: ");
//Serial.println(dig_val);

if(dig_val == 1){
  bitSet(states,1); //set the third bit for sound sensor
}

//delay (20);
```

Figure 7. Sound Sensor Pin Read and Bit Set

The code is related to the sound sensor. It first reads the digital input value from the sound sensor and stores it in the "dig_val" variable. Then, it checks if the "dig_val" variable equals 1. If it is, it sets the second bit of the "states" variable using the "bitSet" function to indicate that sound has been detected. Finally, it includes a delay of 20 milliseconds before executing the following line of code.

2.3.4. Verification

Firstly, I conducted tests on the ultrasonic sensors. I placed an object in front of the receiver microphone, exactly 4 meters away, the maximum detection distance indicated in the datasheet [7]. Then, I slowly approached the object until I reached a distance of around 86cm, the expected length of the crib. During this process, the team member used a ruler to measure the distance between the object and the surface of the microphone and ensure there was no object between the test object and the microphone. At this point, the program gave a prompt indicating that the distance had fallen below the set value, indicating that the baby had either stood up or reached the height of the crib railing, and an alarm needed to be triggered.

Next, I tested the pressure sensor. Since we were testing on a breadboard, our initial goal was to test whether the sensor matrix could reflect the changes in each sensor's data in the program. I pressed each pressure sensor with the same force one by one in the first column and observed the changes in each value in the 3*5 matrix constructed in the program. I found that the variable values of the corresponding sensors did change as I pressed them, and the difference in the values after the change was not significant, indicating that the accuracy of the sensors was acceptable.

As I had previously tested and controlled my pressing force to be approximately 5N on the pressure sensor, each change in the sensor value was incremented by one in the counter. Since the condition set in the program was to trigger an alarm when the count was greater than four, I continued to test the case where four sensors were simultaneously pressed, and the program successfully issued a reminder.

For the sound sensor, For the sound sensor, I played a baby's cry at a distance of 86cm from the sensor, which is the length of the crib, and the volume of the cry was measured at 105 decibels at zero distance. The testing requirement was that the sound sensor detects the sound, judge if it exceeds the threshold, and then show the warning message on the screen. However, the test results showed that the sensor did not always trigger an alarm. I analyzed the environmental

conditions and concluded that the presence of other objects in the area, even without obstruction, could absorb some of the sound waves, reducing the sound intensity reaching the microphone at the receiving end of the sensor. In actual use, pillows and blankets will be on the baby bed, absorbing some sound waves. Therefore, I suggested adjusting the sensor's threshold to make it more sensitive. The latest test also found that the sound sensor may have suffered from some physical malfunctions.

Upon testing the sound sensor, we discovered some unexpected issues. When there is no noise in the nearby environment, the digital output continuously returns a value of "1," and the analog output returns around 60. We have identified this as a malfunction in the sensor. We may need to buy a brand-new sound sensor or adjust the if statement in the code.

I also tested whether the program can simultaneously process the input data of multiple sensors. Also, use a 3-bit array to represent all eight cases, including the trigger/not trigger cases for three sensors. It makes it convenient for data transmission through Bluetooth later in the project.

I asked the program to output a variable for each sensor based on the trigger criteria we previously set for each sensor. If the corresponding condition is triggered, bitSet the value to 1; otherwise bitClear the value back to zero. Each sensor occupies one bit of code, so the three sensors correspond to three bits of code, which are 2*2*2 = 8 cases in total. Therefore, the receiving end can recognize the corresponding case after receiving the Bluetooth signal.

Arranged as Ultrasonic--Pressure--Sound	
0-0-0	Read as case #0
1-0-0	Read as case #1
0-1-0	Read as case #2
1-1-0	Read as case #3
0-0-1	Read as case #4
1-0-1	Read as case #5
0-1-1	Read as case #6
1-1-1	Read as case #7

2.4. Crib Model Design & Modification

I have made some cuts to the model of the Crib, which is a cardboard box of dimensions specified in our Design Document, and marked the position of each sensor in the pressure sensor matrix. Similarly, I have marked and drilled holes for the ultrasonic sensor array. In addition, I have shortened the distance between the two ultrasonic sensors in the middle from 15cm to 10cm. This adjustment is beneficial to ensure that all the space is scanned and reduces gaps.

I also cut the paper box according to the style of the baby crib, cut off the top surface, and then taped one of the openings on both sides, leaving the other opening for easy installation of the pressure sensor matrix and necessary adjustments. Then, together with Yuhao, we accurately marked the installation positions of each pressure sensor on the bottom edge of the paper box for easy installation later. I also marked the installation positions of the subsequent circuit case and ultrasonic sensors on the outer wall of the paper box.

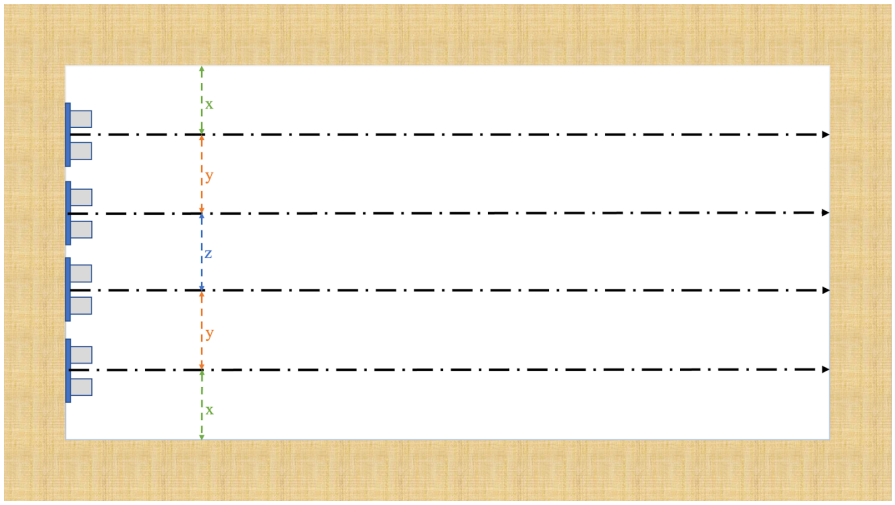

Figure 8. Ultrasonic Sensor Array

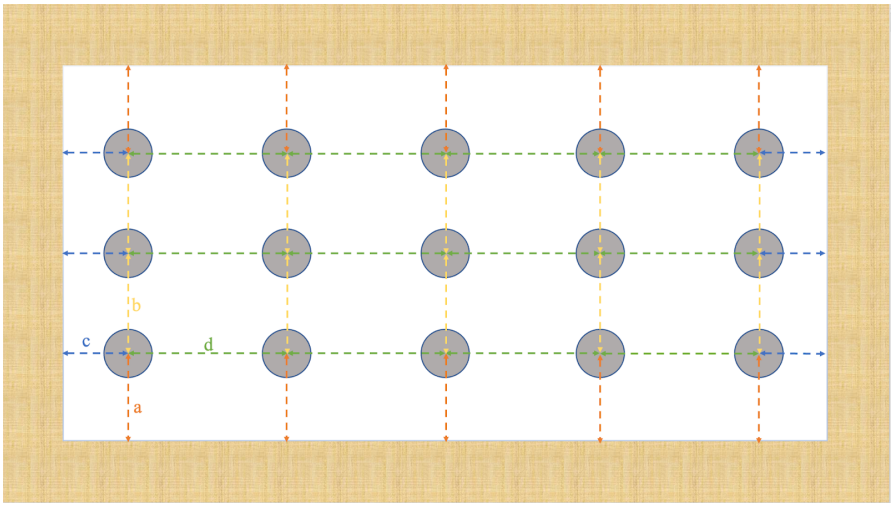

Figure 9. Pressure Sensor Matrix

Figure 10. Crib Model (Paper Box Version)

3. Conclusion

In the coming days, I will continue to calibrate and experiment with the parameters of the sensor subsystem to ensure that the MCU can accurately determine the experimental conditions and correctly encode the bit code. As Yuhao and I will be responsible for the PCB soldering of the crib subsystem, I will work with him to complete this part of the project. Additionally, I will promptly complete the crib model drilling, the circuit case installation, and the sensors' fixation. Here is my schedule for the next few weeks:

3/26-4/1	Welding PCB with other team members. Yuhao and I are responsible for the welding of Crib PCB

4/2-4/8	By adding additional resistors to the sound sensor module to increase its gain, the threshold can be adjusted to ensure that the sensor only sends back pulses when it detects sounds that are louder than a certain decibel level, higher than that of a baby's crying.
4/9-4/15	Experiment with the parameters of the sensor subsystem to ensure it's correctly responding to different cases. Put subsystems together, and prepare for the mock demo
4/16-4/22	Prepare for the final demo
4/23-4/29	Write the Final Report

Table 1. Timeline for Project Completion

As a reminder of ethical conduct, I will consistently assist other team members with their responsibilities if they encounter issues they cannot resolve independently.

References

[1] DFRobot. (n.d.). *SEN0294 DFRobot Digital D7S Seismic Sensor.* [Online]. Available at: https://media.digikey.com/pdf/Data%20Sheets/DFRobot%20PDFs/SEN0294_Web.pdf. [Accessed: 28-Mar-2023].

[2] University of California, Berkeley. (2005). *EE 105 Discussion 5: MOSFET Small-Signal Model.* [Online]. Available at: https://inst.eecs.berkeley.edu/~ee105/fa05/handouts/discussions/Discussion5.pdf. [Accessed: 28-Mar-2023].

[3] Fairchild Semiconductor. (2013). FQP30N06L Datasheet. [Online]. Available at: https://cdn.sparkfun.com/datasheets/Components/General/FQP30N06L.pdf. [Accessed: 28-Mar-2023].

[4] ETC. (2018). HC-SR04 Datasheet. [Online]. Available at: https://datasheetspdf.com/pdf-file/1380136/ETC/HC-SR04/1. [Accessed: 28-Mar-2023].

[5] Texas Instruments. (2007). LMV324 Low-Voltage Rail-to-Rail Output Operational Amplifiers

Datasheet. [Online]. Available at:
https://cdn.sparkfun.com/datasheets/Sensors/Sound/LMV324.pdf.
[Accessed: 28-Mar-2023].

[6] Pi and More. (2016, October 4). *Arduino: HC-SR04 Ultrasonic Distance Sensor with Visuino.* [Online]. Available at:
https://piandmore.wordpress.com/2016/10/04/arduino-hc-sr04-ultrasonic-distance.
[Accessed: 28-Mar-2023].

[7] ElectroSchematics. (2013). HCSR04 Datasheet. [Online]. Available at:
https://www.electroschematics.com/wp-content/uploads/2013/07/HCSR04-datasheet-version-1.pdf. [Accessed: 28-Mar-2023].