GROUP 59

Security System Music Player

MILESTONE 7

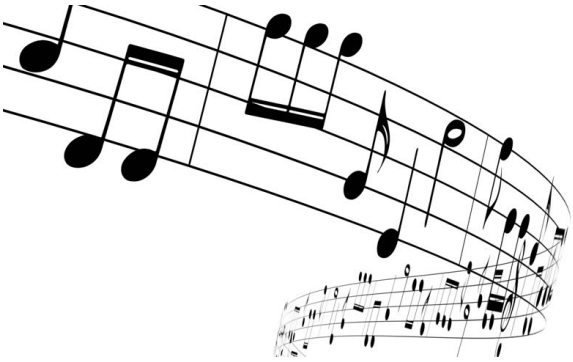
Prepared by:

+ Son Nguyen - snguye49 - snguye49@uic.edu
+ Quan Tran - qtranm2 - qtranm2@uic.edu
+ Akshant Jain - ajain78 - ajain78@uic.edu

GROUP 59

ABSTRACT OF THE PROJECT

Since the invention of basic lock and key and other less complex safety mechanisms, modern anti-theft systems nowadays have been developed, including a variety of high-tech security devices such as cameras and biometric identification. However, users do not have options to personalize the security system like changing the sound of the alarm as this function on devices currently available on the market is limited. Therefore, we want to introduce a customizable anti-theft product that will play the user's choice of music when the device detects a potential threat around the user's house.



GROUP 59

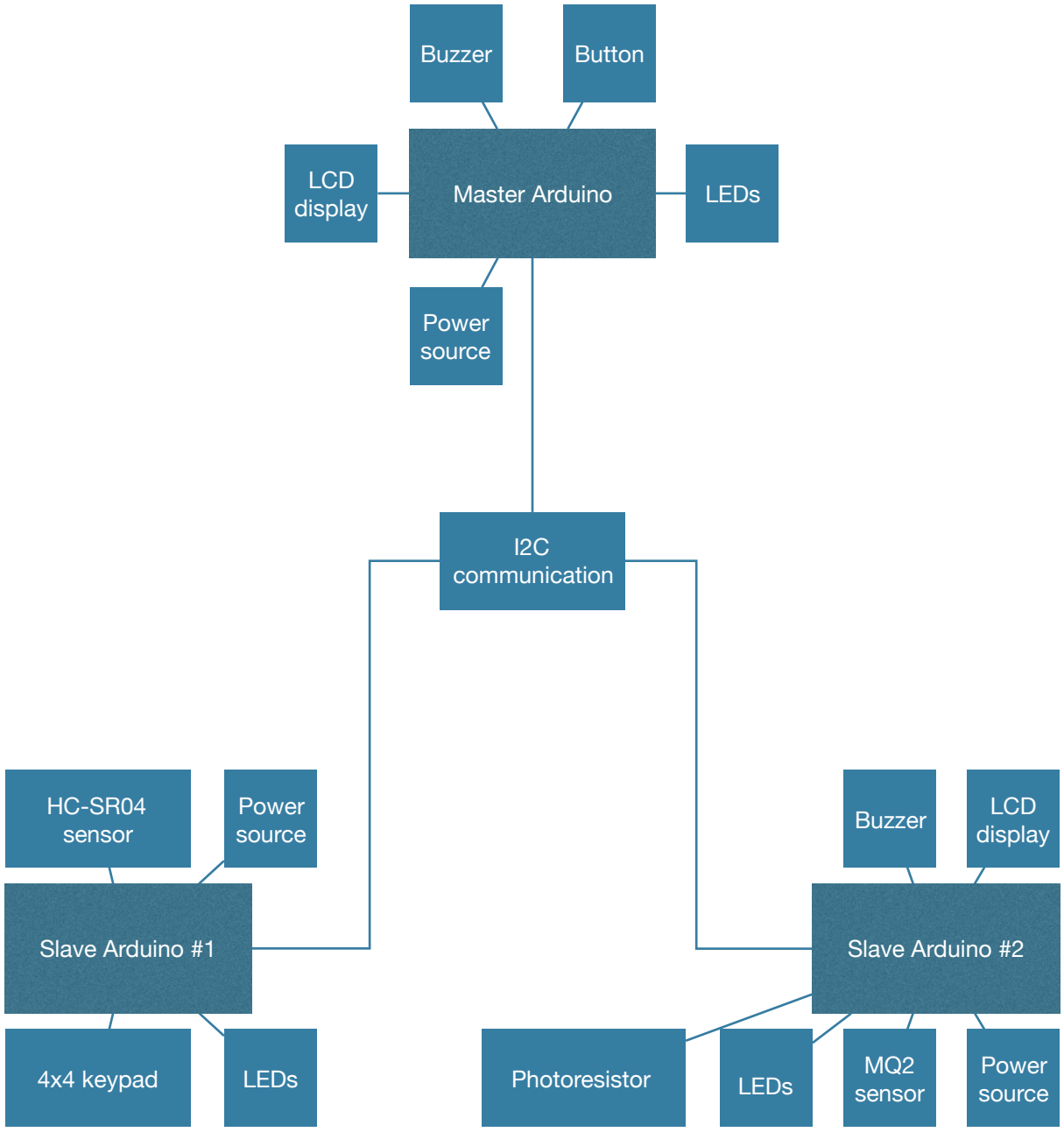
DETAILED PROJECT IDEAS

1. Overall Description of Project Idea

The main idea for our Security System Music Player project is to create a responsive auditory and visual security system that alerts the user of any credible safety threats within the user's home. We use I2C communication to connect multiple slave Arduinos to a single master Arduino and have the master control them. Regarding the mechanical aspects of this system, there will be several different components responsible for the various functions being performed.

The primary function of this system is to play the music or melody that the user has pre-selected in response to a safety threat. Sound will be played at a very loud volume so that the user can aware of the situation. In addition to the musical component of the system, we plan on implementing several accompanying forms of visual alerts. Our system can be activated and deactivated by utilizing a master passcode, which can be entered from the keypad. Since the primary function of the system is playing music based on the user's preference after the alarm is triggered, the user will first need to set up the system by inputting a password, then select a melody to be played from a list of pre-installed melodies.

Besides playing music, our system includes elementary sensors such as an HC-SR04 ultrasonic sensor to detects the distance of an object, a photoresistor to detects a warning level of luminous intensity in the house and an MQ2 sensor to detect a gas leak, smoke or high level of carbon monoxide. Our main goal for this project is to maximize the safety of the user and protecting them from a variety of possible dangers when combining with different sensors.



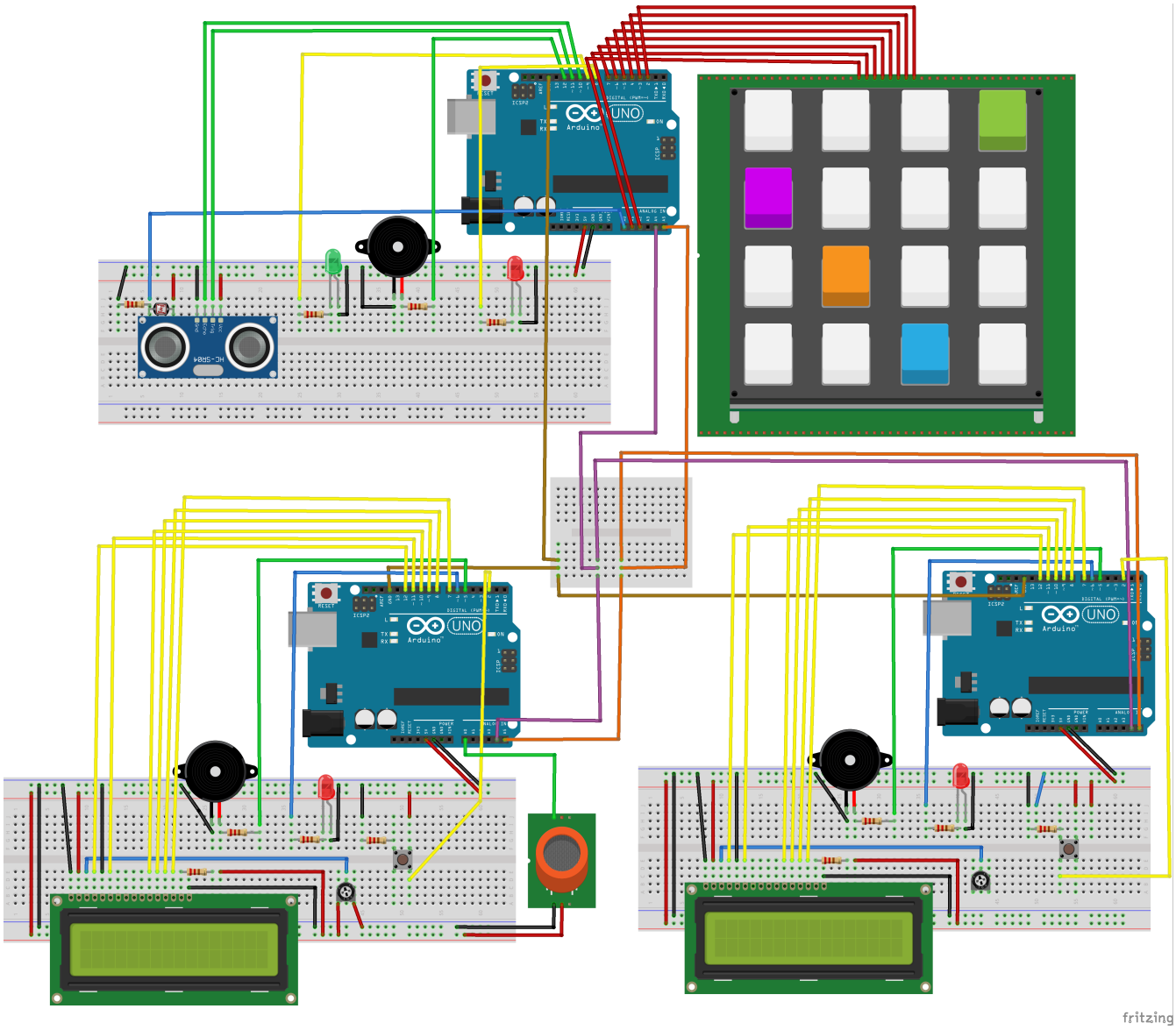
Overview of Project Idea

2. Final Project Design stating how multiple Arduinos will be used

The first Arduino will provide the system functionalities with the ultrasonic sensor, photoresistor, and a 4x4 keypad. It will be placed outside the house of the user to detect unwanted strangers by sending a signal to the master Arduino when someone is standing in front of the ultrasonic sensor for too long or entering the wrong password for the system too many times. The second Arduino, which behaves as the master, has a buzzer, LEDs and a 16x2 LCD as output devices. When it detects a threat or receives any information transmitted by the other two Arduinos, the buzzer will be triggered by buzzing a melody that the user has pre-selected in the processing of setting up the environment. It also has a button to stop the alarm and return the system to the normal state. The third Arduino is connected to an MQ2 gas sensor and a buzzer, which is triggered simultaneously with the master Arduino. The wiring diagram is shown below.

3. Final Plan for Use and Communication between multiple Arduinos

To allow three different Arduinos to communicate with another and carry out their function, we will utilize the Inter-Integrated Circuit (I2C) communication. This system will feature a wired connection that enables three Arduino microcontrollers to share information and relay signals back and forth throughout the system. Communication should and will always occur between a master Arduino and a slave Arduino. The advantage of I2C communication is that more than one slave can be connected to a master, and also it can have multiple masters. For our project, one Arduino will be the master and the other two will be the slaves. The wiring diagram is shown below.


UNO
Arduino
HC-SR04
fritzing

4. Final Project Design stating Expected Inputs/Outputs

- **Expected Inputs:**
  - 4x4 keypad should receive the correct digits when the user input the password into the system.
  - Button should turn off the LED, stop the buzzer and return the system to its normal state when it is pressed.
- **Expected Outputs:**
  - Ultrasonic sensor should send a warning after detecting an object for the same 15 seconds to indicate there is an unwanted strangers near the house.
  - Photoresistor should output value with a range between 0 and 1023 based on the brightness of the environment.
  - Gas sensor should return values larger than 20000 to indicate there is a potential fire hazard or gas leak in the house.
  - Red LED should be turned on when a wrong passcode was entered. Green LED should be turned on when a correct passcode was entered.
  - LCD Display should display a “welcome to the system” message when the system is turned on. It should display a message when any of the sensor is activated to indicate a danger nearby. It should also display if the system is currently activated or deactivated.
  - Buzzer should be playing the tone according to user’s preference when turned on.

5. Final Description of the original work being attempted by our project

Most security systems and intruder alert systems are built separately and are limited in the number of sound options available. For the original work attempted in the project, our group merge two functions of detecting unwanted strangers and smoke/fire/carbon monoxide hazards into a single device package, using an MQ2 gas sensor, ultrasonic sensor, photoresistor and connecting those with the Arduinos. We believe having a 2-in-1 customizable security device will make it more convenient for the user as the user only have to set up the app kit once and they are all set. We also modify the buzzer for our project to work with no less than ten different customizable sounds so more complicated sounds will be produced and the user will have more sound options to choose from. We currently have three different melodies but those can be easily added with later updates. Additionally, we have applied everything that we learn in CS 362 labs into this project from the very basic such as LED blinking, LCD screen displaying to more advanced topics such as the serializable connection between two Arduinos and utilizing interrupts for the Arduino.

6. Discussion on how to build our project

- **1st Arduino build:**
  - Step 1: Pin 0 and 1 are used for serial communication so we don't use those.
  - Step 2: Pin 2 to 7, A1 and A2 are used for receiving input from the keypad.
  - Step 3: Pin 10 and 11 are for receiving input from the ultrasonic sensor.
  - Step 4: Pin 12 is for triggering the buzzer.
  - Step 5: Pin 8 and 9 are for the red and green LEDs respectively.
  - Step 6: Pin A0 is for receiving value from the photoresistor.
  - Step 7: Pin A4 and A5 are for I2C communication between other Arduinos.
- **2nd Arduino build:**
  - Step 1: Pin 0 and 1 are used for serial communication so we don't use those.
  - Step 2: Pin 2 is for receiving input from a button and also the button can be attached to an interrupt.
  - Step 3: Pin 5 is for triggering the buzzer.
  - Step 4: Pin 6 is for the red LED.
  - Step 5: Pin 7 to 12 is for connecting with the LCD.
  - Step 76 Pin A4 and A5 are for I2C communication between other Arduinos.
- **3rd Arduino build:**
  - Step 1: Pin 0 and 1 are used for serial communication so we don't use those.
  - Step 2: Pin 2 is for receiving input from a button and also the button can be attached to an interrupt.
  - Step 3: Pin 5 is for triggering the buzzer.

- Step 4: Pin 6 is to turn on the LED.
- Step 5: Pin 7 to 12 is for connecting with the LCD.
- Step 6: Pin A0 is for the MQ2 gas sensor.
- Step 7: Pin A4 and A5 are for I2C communication between other Arduinos.

7. Discussion on how our project is to be used (User Guide)

I. Choose a place to setup your Arduino units

- Select a convenient location near both an electrical wall outlet and the front door of your house. Using the wiring cord provided, plug one end into the back of the 1st Arduino unit (Slave #1) and one end into your wall jack.
- Choose an area of your home in which you spend the most time, or in the safe area like your bedroom. Put the 2nd Arduino unit (Master) on a stand so it doesn't oscillate. Be sure that the electrical outlet you select to plug in this Arduino cannot be turned off by a wall switch.
- For the 3rd Arduino unit (Slave #2), we recommend you to put it near the kitchen as it is designed for gas leak detection. Plug one end of the wiring cord provided into the back of this Arduino and the other end into a power source, also provided in the box.

II. Set up the environment and start the system

- All three Arduino units should be turn on at the same time for synchronization.
- User sets up the passcode for the system from the keypad.
- User chooses a melody to be played to the buzzers.
- The photoresistor will signal when it detects a warning amount of light longer than 15s.
- The ultrasonic sensor will signal when it detects someone near the sensor longer than 15s.
- The gas sensor will signal when it detects smoke, gas or a high amount of carbon monoxide in the house.
- The LEDs will blink rapidly to indicate there is an ongoing threat.
- The LCD will start displaying messages about an ongoing threat.
- The buzzer will play the sound that the user has chose before.
- User presses the button in one of the Arduinos to deactivate the alarm.

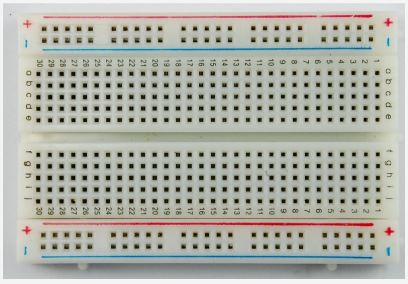
GROUP 59

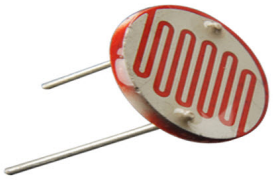
REQUIRED SUPPORTING MATERIALS

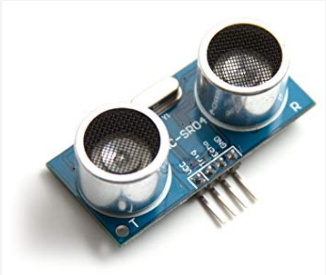
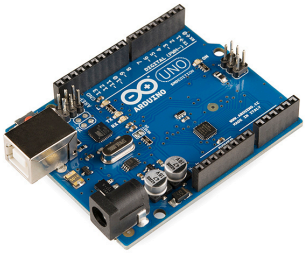
1. Timeline of development given in a week-by-week format of completed items

WEEK NUMBER	COMPLETED ITEMS
Week 1	Form groups and start sharing ideas for the project
Week 2	Get familiar with the Arduino IDE and Arduino skeleton code
Week 3	Learn how to wire a LED
Week 4	Learn how to set up push buttons and LED with the Arduino
Week 5	Learn how to use more complicated forms of output such as LCD display
Week 6	Learn how to get our photoresistor wired correctly and print the output to serial monitor
Week 7	Learn how to have Arduino do unrelated things at the same time
Week 8	Learn how to print date and time from the user via serial monitor in Arduino IDE
Week 9	Learn how to communicate two Arduinos together using Serial communication
Week 10	Learn how to write program for the Arduino that utilizes interrupts
Week 11	Learn how to use Processing for graphing stuffs
Week 12	Start applying what we have learned in the lab to our project and finalizing our project structure
Week 13	Watch tutorials online on efficient ways to connect multiple Arduinos together and start some basic coding
Week 14	Finish wiring all the IO devices that will be used for the project and connect them to our Arduinos
Week 15	Finish the code for the project and demonstrate the project as a whole to the T.A.
Week 16	Wrap up and reflect back on what we have learned through out the whole semester

2. Final List of Materials Needed

	Wire	
3 pcs	Breadboard	
3 pcs	Battery	
5 pcs	LED	
2 pcs	Button	

2 pcs	16x2 LCD screen	
2 pcs	Potentiometer	
1 pcs	Photoresistor	
1 pcs	MQ2 gas sensor	
1 pcs	4x4 generic keypad	

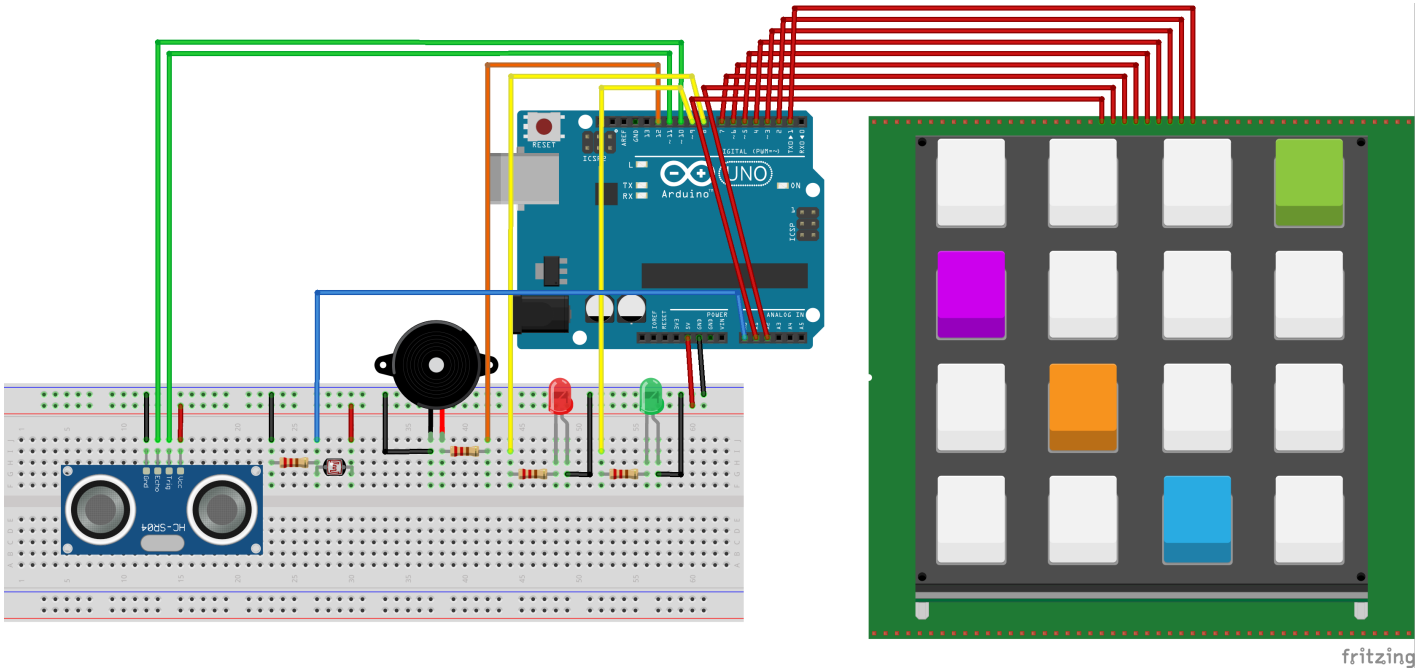
1 pcs	HC-SR04 ultrasonic sensor	
3 pcs	Arduino Uno microcontroller	
3 pcs	Buzzer	

3. Final List of References

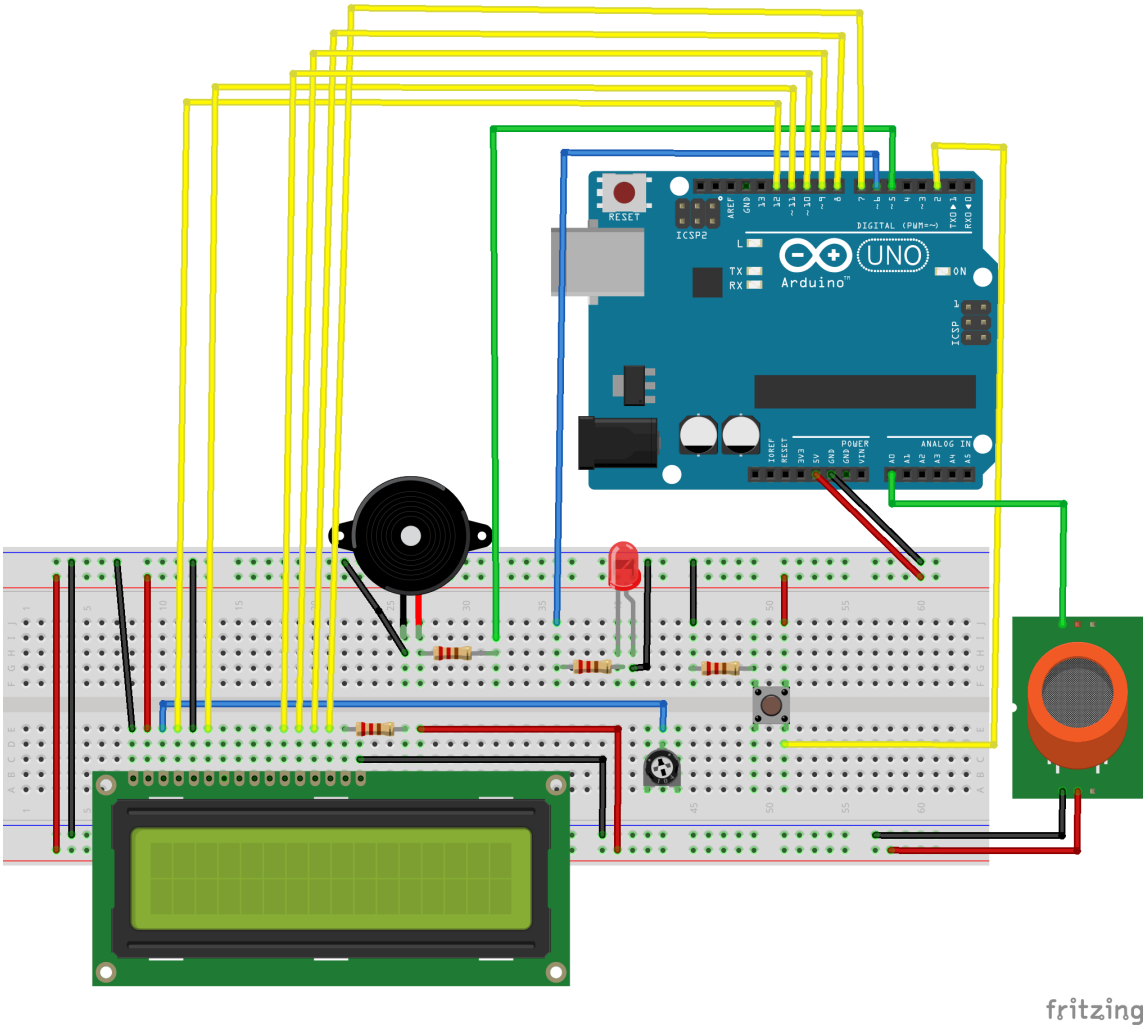
- https://arduino.cc/en/Tutorial/Blink
- https://www.arduino.cc/en/Tutorial/Button
- https://www.arduino.cc/en/Tutorial/LiquidCrystalScroll
- https://playground.arduino.cc/Learning/PhotoResistor/
- https://www.arduino.cc/en/Reference/AnalogWrite
- https://www.arduino.cc/reference/en/language/functions/communication/serial/
- https://iotguider.in/arduino/serial-communication-between-two-arduino-boards/
- https://arduino.cc/en/Reference/attachInterrupt
- https://learn.sparkfun.com/tutorials/connecting-arduino-to-processing#introduction
- https://circuitdigest.com/microcontroller-projects/arduino-audio-music- player
- https://create.arduino.cc/projecthub/fradirosa00/arduino-fire-alarm-4da798
- https://randomnerdtutorials.com/complete-guide-for-ultrasonic-sensor-hc-sr04/
- https://create.arduino.cc/projecthub/techmirtz/using-4x4-keypad-with-arduino-2d22e9
- https://electronicsprojectshub.com/how-to-connect-mq2-gas-sensor-to-arduino/
- https://learn.sparkfun.com/tutorials/i2c/all
- https://forum.arduino.cc/index.php?topic=445737.0
- https://www.youtube.com/watch?v=PnG4fO5_vU4
- https://www.youtube.com/watch?v=yQ15Hi1E7I4
- https://www.arduino.cc/en/Tutorial/ToneMelody?from=Tutorial.Tone

4. Final Arduino Diagrams

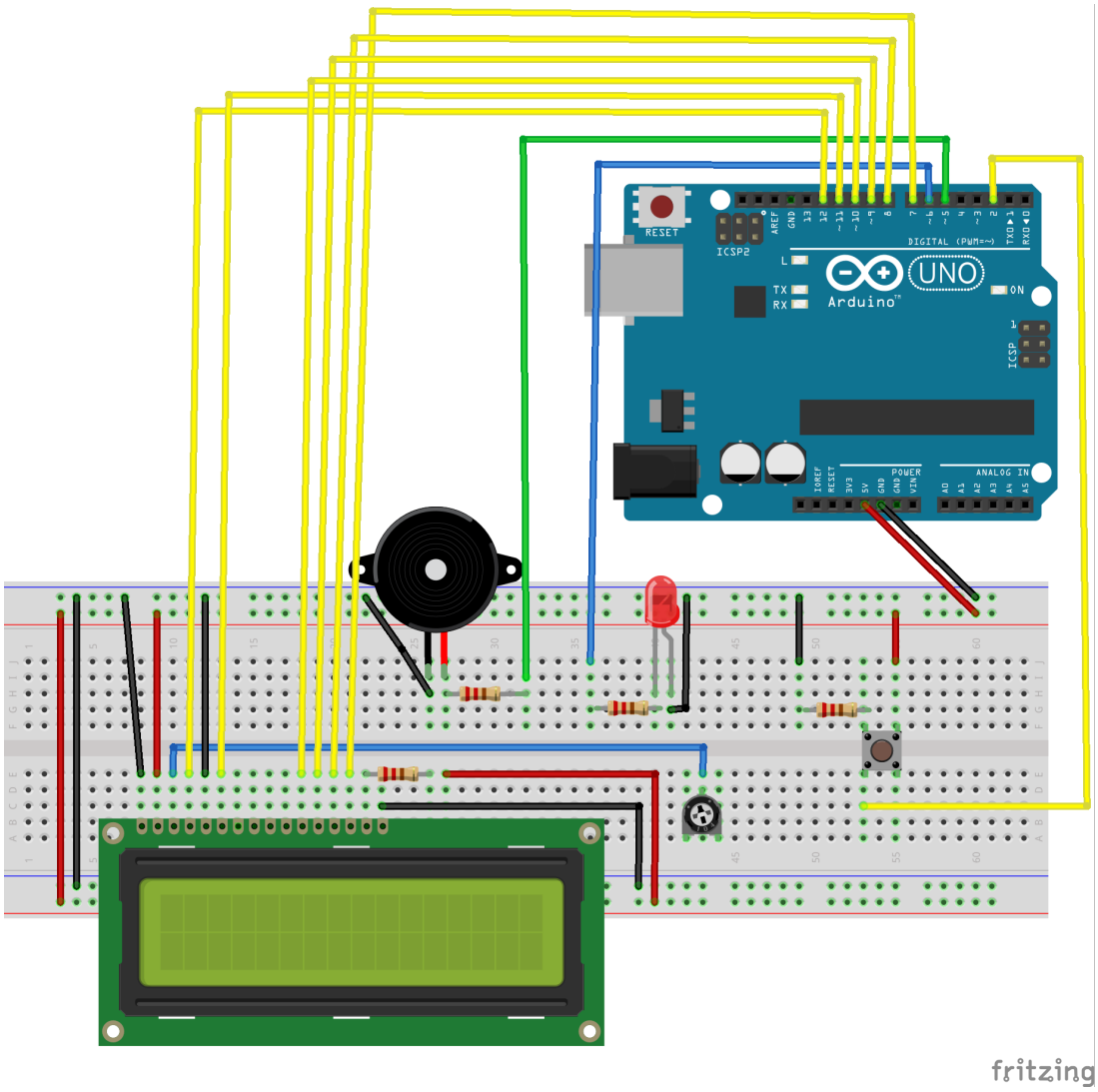
I. 1st Arduino diagram



II. 2nd Arduino diagram



III. 3rd Arduino diagram



5. Final code sketches

- First Arduino sketch

```
// Group 59 - Security System Music Player (CS362 - Fall19)
// + Quan Tran - qtranm2
// + Son Nguyen - snguye49
// + Akshant Jain - ajain78

// First Arduino (SLAVE_1)
// This one should be put near the entrance of the user's house
// Connects with 4x4 keypad, ultrasonic sensor, photoresistor, buzzer and LEDs

// PINS:
// 0: none
// 1: none
// 2: keypad
// 3: keypad
// 4: keypad
// 5: keypad
// 6: keypad
// 7: keypad
// 8: red LED
// 9: green LED
// 10: HC-SR04 echo
// 11: HC-SR04 trigger
// 12: buzzer
// 13: none
// A0: photoresistor
// A1: keypad
// A2: keypad
// A3: none
// A4: I2C
// A5: I2C

#include <Keypad.h>   // for 4x4 Keypad
#include <NewPing.h>  // for HCSR04 ultrasonic sensor
#include <Wire.h>     // for I2C
#include "pitches.h"  // pre-defined frequencies

typedef unsigned int uint;

/* ------------ HC-SR04 ------------ */
const uint HCSR04_TRIGGER_PIN = 11;
const uint HCSR04_ECHO_PIN = 10;
const uint HCSR04_MAX_DISTANCE = 200;  // 2m max

NewPing hcsr04(HCSR04_TRIGGER_PIN, HCSR04_ECHO_PIN, HCSR04_MAX_DISTANCE);

/* ------------ KEYPAD ------------ */
const byte KP_ROWS = 4;  // number of rows of keypad
const byte KP_COLS = 4;  // number of columns of keypad

const char keymap[KP_ROWS][KP_COLS] = {
    {'1', '2', '3', 'A'},
    {'4', '5', '6', 'B'},
    {'7', '8', '9', 'C'},
    {'*', '0', '#', 'D'}};

byte rowPins[KP_ROWS] = {7, 6, 5, 4};    // rows 0 to 3
byte colPins[KP_COLS] = {3, 2, A1, A2};  // columns 0 to 3
```

```
Keypad keypad = Keypad(makeKeymap(keymap), rowPins, colPins, KP_ROWS, KP_COLS);

/* ------------ OTHER ------------ */
const int MASTER = 0x0;
const int SLAVE_1 = 0x1;
const int SLAVE_2 = 0x2;

const uint RED_LED_PIN = 8;
const uint GREEN_LED_PIN = 9;
const uint BUZZER_PIN = 12;
const uint PHOTO_PIN = A0;

const uint DISTANCE_THRESHOLD = 10;
const uint LUMINANCE_THRESHOLD = 900;

String userInputPassword;  // to store the input password from user

// turn on the LED in certain amount of time
unsigned long ledTime;

// handle distance get too low in certain amount of time
unsigned long distanceTime;
bool distanceTooNear = false;

// handle luminance get too high in certain amount of time
unsigned long luminanceTime;
bool luminanceTooBright = false;

bool setupFirstTime = true;  // setup first time
volatile bool systemActivated = false;

// STATE:
// 0: idle, waiting for '*'
// 1: '*' received, waiting user to input passcode
// 2: '#' received, user confirmed passcode, send to the master
// 3: waiting respond from master
// 4: alarm activated
// 5: alarm deactivated -> state 0
volatile uint state = 0;

/* ------------ MELODY ------------ */
int m1[] = {NOTE_C4, NOTE_G3, NOTE_G3, NOTE_A3, NOTE_G3, 0, NOTE_B3, NOTE_C4};
int n1[] = {4, 8, 8, 4, 4, 4, 4, 4};

int m2[] = {NOTE_E3, NOTE_E3, NOTE_F3, NOTE_G3, NOTE_G3, NOTE_F3, NOTE_E3, NOTE_D3, NOTE_C3, NOTE_C3,
NOTE_D3, NOTE_E3, NOTE_E3, 0, NOTE_D3, NOTE_D3};
int n2[] = {4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 2, 2};

/* ------------ SETUP ------------ */
void setup() {
  Wire.begin(SLAVE_1);           // join I2C bus with address #1
  Wire.onReceive(receiveEvent);  // register receive event

  pinMode(RED_LED_PIN, OUTPUT);    // red LED
  pinMode(GREEN_LED_PIN, OUTPUT);  // green LED

  Serial.begin(9600);
  Serial.println("[DEBUG] First Arduino (SLAVE_1) initialized");
}

/* ------------ MAIN ------------ */
void loop() {
  if (state == 0) {
```

```
  if (keypad.getKey() == '*') {
    state = 1;
    userInputPassword += '*';
    Serial.println("[DEBUG] * received, state = 1");
  }
}

if (state == 1) {
  char input = keypad.getKey();
  if (input) {
    Serial.print("[DEBUG] input: ");
    Serial.println(input);

    // 0 to 9
    if (isdigit(input)) {
      userInputPassword += input;
    }

    // 'A', 'B', 'C', 'D'
    if (isalpha(input)) {
      size_t len = userInputPassword.length();
      if (len == 5) {
        userInputPassword += input;
      } else if (len < 5) {
        Serial.println("[WARN] You have to put 4-digit passcode first");
      } else {
        Serial.println("[WARN] You already chose a mode, press # to confirm");
      }
    }

    // user confirmed with #
    if (input == '#') {
      // inputPwd should be *1234A#, length = 6
      if (userInputPassword.length() == 6) {
        userInputPassword += input;
        state = 2;
        Serial.print("[DEBUG] # received, state = 2, inputPwd: ");
        Serial.println(userInputPassword.c_str());
      } else {
        Serial.println("[WARN] # received but password is not in valid format");
        state = 0;
        userInputPassword = "";
        Serial.println("[DEBUG] password not set, reset to state = 0");
      }
    }
  }
}

if (state == 2) {
  // send password to master
  String pwd = "pass:" + userInputPassword;

  Wire.beginTransmission(MASTER);
  Wire.write(pwd.c_str());
  Wire.endTransmission();

  userInputPassword = ""; // clear the password

  if (setupFirstTime) {
    state = 0;
    setupFirstTime = false;
    Serial.println("[DEBUG] first time setup, password sent to master, cleared, state = 0");
  } else {
    state = 3;
```

```
    Serial.println("[DEBUG] password sent to master, cleared, state = 3");
  }
}

if (state == 3) {
  // Serial.println("[DEBUG] Waiting response from master");
  // do nothing
}

if (state == 4) {
  activateAlarm(true);
}

if (state == 5) {
  activateAlarm(false);
  state = 0;
}

// get the distance from the HC-SR04 sensor
if (state != 3) {
  if (!systemActivated) return;

  const uint distance = hcsr04.ping_cm();
  if (distance > 0 && distance < HCSR04_MAX_DISTANCE) {
    if (distance < DISTANCE_THRESHOLD) {
      if (!distanceTooNear) {
        distanceTooNear = true;
        distanceTime = millis();
      }
      // if distance is too near in 3s
      if (distanceTooNear) {
        if (millis() - distanceTime > 3000) {
          Wire.beginTransmission(MASTER);
          Wire.write("dist");
          Wire.endTransmission();
          distanceTooNear = false;
          state = 3;
        }
      }
    } else {
      distanceTooNear = false;
    }
  }

  // get the luminance from the photoresistor
  const int luminance = analogRead(PHOTO_PIN);
  if (luminance > 0) {
    if (luminance > LUMINANCE_THRESHOLD) {
      if (!luminanceTooBright) {
        luminanceTooBright = true;
        luminanceTime = millis();
      }
      // if light is too bright in 3s
      if (luminanceTooBright) {
        if (millis() - luminanceTime > 3000) {
          Wire.beginTransmission(MASTER);
          Wire.write("lumi");
          Wire.endTransmission();
          luminanceTooBright = false;
          state = 3;
        }
      }
    } else {
      luminanceTooBright = false;
```

```
      }
    }

    // turn off the LEDs after 3s
    if (digitalRead(GREEN_LED_PIN) == HIGH || digitalRead(RED_LED_PIN) == HIGH) {
      if (millis() - ledTime > 3000) {
        digitalWrite(GREEN_LED_PIN, LOW);
        digitalWrite(RED_LED_PIN, LOW);
      }
    }
  }
}

void activateAlarm(bool shouldAlarm) {
  if (shouldAlarm) {
    // blinking both LEDs
    digitalWrite(RED_LED_PIN, HIGH);
    digitalWrite(GREEN_LED_PIN, HIGH);
    delay(200);
    digitalWrite(RED_LED_PIN, LOW);
    digitalWrite(GREEN_LED_PIN, LOW);
    delay(200);
  } else {
    // turn off the LEDs
    digitalWrite(RED_LED_PIN, LOW);
    digitalWrite(GREEN_LED_PIN, LOW);
  }
}

void receiveEvent(int howMany) {
  // read the incoming data
  String data;
  while (howMany--) {
    data += Wire.read();
  }
  Serial.print("data: ");
  Serial.println(data.c_str());

  // handle response from master after sending password
  if (data.indexOf("pwd") >= 0) {
    digitalWrite((data == "pwd correct") ? GREEN_LED_PIN : RED_LED_PIN, HIGH);
    ledTime = millis();
    state = 0;
  }

  // activate alarm
  if (data.indexOf("alarm") >= 0) {
    state = (data == "activate alarm") ? 4 : 0;
  }

  // activate system
  if (data.indexOf("system") >= 0) {
    systemActivated = (data == "system activated") ? true : false;
  }
}
```

- Second Arduino sketch

```
// Group 59 - Security System Music Player (CS362 - Fall19)
// + Quan Tran (qtranm2)
// + Son Nguyen (snguye49)
// + Akshant Jain (ajain78)

// Second Arduino (MASTER)
// This one should be safely put inside the user's house (bedroom,...)
// Connects with LCD, LED, buzzer and button

// PINS:
// 0: none
// 1: none
// 2: button
// 3: none
// 4: none
// 5: buzzer
// 6: red LED
// 7: LCD
// 8: LCD
// 9: LCD
// 10: LCD
// 11: LCD
// 12: LCD
// 13: none
// A0: none
// A1: none
// A2: none
// A3: none
// A4: I2C
// A5: I2C

#include <LiquidCrystal.h>  // for LCD
#include <Wire.h>           // for I2C
#include "pitches.h"        // pre-defined frequencies

typedef unsigned int uint;

/* ------------ OTHER ------------ */
const int MASTER = 0x0;
const int SLAVE_1 = 0x1;
const int SLAVE_2 = 0x2;

const uint BUTTON_PIN = 2;
const uint BUZZER_PIN = 5;
const uint RED_LED_PIN = 6;

LiquidCrystal lcd(12, 11, 10, 9, 8, 7);

String receivedPassword;      // to store received password
uint wrongPasswordCount = 0;  // number of wrong password input

unsigned long lcdTime;

// user can change password from Serial Monitor
String PASSWORD = "1234";

String threatMessage;  // threat status
String mainMode;       // to store the mode

volatile bool systemActivated = false;
```

```
// handle input from serial monitor
const byte numChars = 32;
char receivedData[numChars];  // to store the input from Serial Monitor
char data[numChars];          // make a copy of the received input
boolean newData = false;      // new data is received from the Serial Monitor
boolean parsed = false;       // data was parsed and validated successfully

bool printed = false;  // printed the message
bool welcome = false;  // printed the welcome message

// STATE:
// 0: idle, waiting data from slave
// 1: check password
// 2: alarm activated
// 3: alarm deactivated -> state 0
// 4: setup the password
volatile uint state = 4;

/* ------------ MELODY ------------ */
int m1[] = {NOTE_C4, NOTE_G3, NOTE_G3, NOTE_A3, NOTE_G3, 0, NOTE_B3, NOTE_C4};
int n1[] = {4, 8, 8, 4, 4, 4, 4, 4};

int m2[] = {NOTE_E3, NOTE_E3, NOTE_F3, NOTE_G3, NOTE_G3, NOTE_F3, NOTE_E3, NOTE_D3, NOTE_C3, NOTE_C3,
NOTE_D3, NOTE_E3, NOTE_E3, 0, NOTE_D3, NOTE_D3};
int n2[] = {4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 2, 2};

int m3[] = {NOTE_C3, NOTE_E3, NOTE_G3, NOTE_C4};
int n3[] = {4, 4, 4, 4};

/* ------------ SETUP ------------ */
void setup() {
  Wire.begin(MASTER);            // join I2C bus with address #2
  Wire.onReceive(receiveEvent);  // register receive event

  // initialize LCD
  lcd.begin(16, 2);
  lcd.clear();

  pinMode(RED_LED_PIN, OUTPUT);  // red LED
  pinMode(BUTTON_PIN, INPUT);    // button
  pinMode(BUZZER_PIN, OUTPUT);   // buzzer

  attachInterrupt(0, buttonPressed, CHANGE);  // attach interrupt for button

  Serial.begin(9600);
  Serial.println("[DEBUG] Second Arduino (MASTER) initialized");
}

/* ------------ MAIN ------------ */
void loop() {
  if (state == 0) {
    if (!welcome) {
      printToLcd("WELCOME TO", "SSMP");
      delay(2000);
      welcome = true;
    }
    if (systemActivated) {
      printToLcd("SYSTEM", "ACTIVATED");
      delay(200);
    } else {
      printToLcd("SYSTEM", "DEACTIVATED");
      delay(200);
    }
    // do nothing
```

```
}

if (state == 1) {
  // verify password
  if (receivedPassword == PASSWORD) {
    // notify SLAVE_1 that password is correct
    printToLcd("PASSWORD IS", "CORRECT");
    Serial.println("[DEBUG] Password is correct");

    Wire.beginTransmission(SLAVE_1);
    Wire.write("pwd correct");
    Wire.endTransmission();

    if (mainMode == "A") {
      // notify all slaves that system is activated
      systemActivated = true;

      Wire.beginTransmission(SLAVE_1);
      Wire.write("system activated");
      Wire.endTransmission();

      Wire.beginTransmission(SLAVE_2);
      Wire.write("system activated");
      Wire.endTransmission();
    } else if (mainMode == "B") {
      // notify all slaves that system is deactivated
      systemActivated = false;

      Wire.beginTransmission(SLAVE_1);
      Wire.write("system deactivated");
      Wire.endTransmission();

      Wire.beginTransmission(SLAVE_2);
      Wire.write("system deactivated");
      Wire.endTransmission();
    }
  } else {
    // notify SLAVE_1 that password is incorrect
    printToLcd("PASSWORD IS", "NOT CORRECT");
    Serial.println("[DEBUG] Password is incorrect");
    wrongPasswordCount++;

    Wire.beginTransmission(SLAVE_1);
    Wire.write("pwd incorrect");
    Wire.endTransmission();
  }
  if (wrongPasswordCount == 2) {
    state = 2;
  }
  receivedPassword = "";
  state = 0;
}

if (state == 2) {
  if (systemActivated) {
    announceThreat(true);
    activateAlarm(true);
  } else {
    state = 0;
  }
}

if (state == 3) {
  announceThreat(false);
```

```
    activateAlarm(false);
    state = 0;
  }

  if (state == 4) {
    if (!printed) {
      printToLcd("ENTER PASSWORD", "FORMAT: *1234A#");
      Serial.println("[DEBUG] Please setup the system password");
      Serial.println("[DEBUG] in following format: <*1234A#>");
      printed = true;
    }
  }
}

void buttonPressed() {
  // deactivate the alarm
  if (state == 2) {
    state = 3;
    printToLcd("ALARM IS", "DEACTIVATED");
  }
}

void announceThreat(bool shouldAnnounce) {
  // announce all slaves that it should activate/deactivate alarm
  Wire.beginTransmission(SLAVE_1);
  Wire.write((shouldAnnounce) ? "activate alarm" : "deactivate alarm");
  Wire.endTransmission();

  Wire.beginTransmission(SLAVE_2);
  Wire.write((shouldAnnounce) ? "activate alarm" : "deactivate alarm");
  Wire.endTransmission();
}

void playMusic(int index) {
  if (index == 0) {
    for (int i = 0; i < 8; i++) {
      // to calculate the note duration, take one second divided by the note type
      int noteDuration = 1000 / n1[i];
      tone(BUZZER_PIN, m1[i], noteDuration);

      // to distinguish the notes, set a minimum time between them
      // the note's duration + 30% seems to work well
      int pauseBetweenNotes = noteDuration * 1.30;
      delay(pauseBetweenNotes);

      // stop the tone playing
      noTone(BUZZER_PIN);
    }
  } else if (index == 1) {
    for (int i = 0; i < 15; i++) {
      // to calculate the note duration, take one second divided by the note type
      int noteDuration = 1000 / n2[i];
      tone(BUZZER_PIN, m2[i], noteDuration);

      // to distinguish the notes, set a minimum time between them
      // the note's duration + 30% seems to work well
      int pauseBetweenNotes = noteDuration * 1.30;
      delay(pauseBetweenNotes);

      // stop the tone playing
      noTone(BUZZER_PIN);
    }
  } else {
    for (int i = 0; i < 4; i++) {
```

```
      // to calculate the note duration, take one second divided by the note type
      int noteDuration = 1000 / n3[i];
      tone(BUZZER_PIN, m3[i], noteDuration);

      // to distinguish the notes, set a minimum time between them
      // the note's duration + 30% seems to work well
      int pauseBetweenNotes = noteDuration * 1.30;
      delay(pauseBetweenNotes);

      // stop the tone playing
      noTone(BUZZER_PIN);
    }
  }
}

void activateAlarm(bool shouldAlarm) {
  if (shouldAlarm) {
    // blinking LED and buzzing
    digitalWrite(RED_LED_PIN, HIGH);
    delay(200);
    digitalWrite(RED_LED_PIN, LOW);
    delay(200);

    playMusic(rand() % 3);
  } else {
    // turn off the LED and stop buzzing
    digitalWrite(RED_LED_PIN, LOW);
    noTone(BUZZER_PIN);
  }
}

void printToLcd(String first, String second) {
  threatMessage = first + "," + second;
  if (first.length() > 16) {
    Serial.println("[WARN] First string print to LCD too long");
  }
  if (second.length() > 16) {
    Serial.println("[WARN] Second string print to LCD too long");
  }

  // padding for the string (center alignment)
  int pad1 = (16 - first.length()) / 2;
  int pad2 = (16 - second.length()) / 2;
  if (pad1 < 0) pad1 = 0;
  if (pad2 < 0) pad2 = 0;

  String tmp1, tmp2;
  for (int i = 0; i < pad1; i++) tmp1 += " ";
  tmp1 += first;
  for (int i = 0; i < pad2; i++) tmp2 += " ";
  tmp2 += second;

  // print to LCD
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(tmp1.c_str());
  lcd.setCursor(0, 1);
  lcd.print(tmp2.c_str());

  lcdTime = millis();
}

void receiveEvent(int howMany) {
  // read the incoming data
```

```
String data;
while (howMany--) {
  data += Wire.read();
}
//Serial.print("data: ");
//Serial.println(data.c_str());

// "pass:*1234A#"
int passIdx = data.indexOf("pass");
if (passIdx >= 0) {
  // contains password
  passIdx += 6;  // now idx at first digit
  String pass = data.substring(passIdx, 4);
  passIdx += 4;  // now idx at mode
  String mode = data.substring(passIdx, 1);
  mainMode = mode;

  if (state != 4) {
    // received the password from keypad
    receivedPassword = pass;
    state = 1;
  } else {
    // state = 4, setup the master password
    PASSWORD = pass;
    state = 0;
  }

  Serial.print("PASS: ");
  Serial.println(pass.c_str());
  Serial.print("MODE: ");
  Serial.println(mode.c_str());
  //printToLcd("PASS: " + pass, "MODE: " + mode);
  //delay(2000);
}

if (state == 4) {
  return;  // ignore incoming data
}

// different hazards
if (data.indexOf("dist") >= 0) {
  printToLcd("SUSPICIOUS", "PERSON DETECTED");
  state = 2;
}

if (data.indexOf("fire") >= 0) {
  printToLcd("FIRE RISK", "DETECTED");
  state = 2;
}

if (data.indexOf("lumi") >= 0) {
  printToLcd("BRIGHT LIGHT", "DETECTED");
  state = 2;
}

if (data.indexOf("gas") >= 0) {
  printToLcd("GAS LEAK", "DETECTED");
  state = 2;
}

// SLAVE_2 sent deactivate signal
if (data.indexOf("deactivate") >= 0) {
  state = 3;
  printToLcd("ALARM IS", "DEACTIVATED");
```

```
    delay(1000);
  }
}
```

- Third Arduino sketch

```
// Group 59 - Security System Music Player (CS362 - Fall19)
// + Quan Tran (qtranm2)
// + Son Nguyen (snguye49)
// + Akshant Jain (ajain78)

// Third Arduino (SLAVE_2)
// This one should be put near the kitchen inside the user's house
// Connects with gas sensor, photoresistor, LCD, buzzer

// PINS:
// 0: none
// 1: none
// 2: button
// 3: none
// 4: none
// 5: buzzer
// 6: red LED
// 7: LCD
// 8: LCD
// 9: LCD
// 10: LCD
// 11: LCD
// 12: LCD
// 13: none
// A0: MQ2 gas sensor
// A1: none
// A2: none
// A3: none
// A4: I2C
// A5: I2C

#include <LiquidCrystal.h>  // for LCD
#include <MQ2.h>            // for MQ2 gas sensor
#include <Wire.h>           // for I2C
#include "pitches.h"        // pre-defined frequencies

typedef unsigned int uint;

/* ------------ MQ2 ------------ */
const uint GAS_PIN = A0;
MQ2 mq2(GAS_PIN);

const int LPG_THRESHOLD = 3000;     // liquefied petroleum gas
const int CO_THRESHOLD = 20000;     // carbon monoxide
const int SMOKE_THRESHOLD = 20000;  // smoke

/* ------------ OTHER ------------ */
const int MASTER = 0x0;
const int SLAVE_1 = 0x1;
const int SLAVE_2 = 0x2;

const uint BUTTON_PIN = 2;
const uint BUZZER_PIN = 5;
const uint RED_LED_PIN = 6;

LiquidCrystal lcd(12, 11, 10, 9, 8, 7);

unsigned long lcdTime;

// state
// 0: idle, read value from sensors
```

```
// 1: waiting response from master
// 2: alarm activated
// 3: alarm deactivated -> state 0
volatile uint state = 0;

bool systemActivated = false;

/* ------------ SETUP ------------ */
void setup() {
  Wire.begin(SLAVE_2);           // join I2C bus with address #2
  Wire.onReceive(receiveEvent);  // register receive event

  lcd.begin(16, 2);  // initialize LCD
  lcd.clear();
  printToLcd("WELCOME TO", "SSMP");

  pinMode(RED_LED_PIN, OUTPUT);  // red LED
  pinMode(BUTTON_PIN, INPUT);    // button
  pinMode(BUZZER_PIN, OUTPUT);   // buzzer

  attachInterrupt(0, buttonPressed, CHANGE);  // attach interrupt for button

  Serial.begin(9600);
  Serial.println("[DEBUG] Third Arduino (SLAVE_2) initialized");

  mq2.begin();
}

/* ------------ MAIN ------------ */
void loop() {
  if (state == 1) {
    // Serial.println("[DEBUG] Waiting response from master");
    // do nothing
  }

  if (state == 2) {
    activateAlarm(true);
  }

  if (state == 3) {
    activateAlarm(false);
    state = 1;
  }

  if (state != 1) {
    // if system has not been activated, do nothing
    if (!systemActivated) return;

    // get the value from the gas sensor
    int lpg = mq2.readLPG();
    int co = mq2.readCO();
    int smoke = mq2.readSmoke();

    if (lpg >= LPG_THRESHOLD || co >= CO_THRESHOLD || smoke >= SMOKE_THRESHOLD) {
      Wire.beginTransmission(MASTER);
      Wire.write("gas");
      Wire.endTransmission();
      state = 1;
    }

    if (millis() - lcdTime > 3000) {
      printToLcd("WELCOME TO", "SSMP");
    }
  }
```

```
}

void buttonPressed() {
  // deactivate the alarm
  if (state == 2) {
    state = 3;
    printToLcd("ALARM IS", "DEACTIVATED");
  }
}

void printToLcd(String first, String second) {
  if (first.length() > 16) {
    Serial.println("[WARN] First string print to LCD too long");
  }
  if (second.length() > 16) {
    Serial.println("[WARN] Second string print to LCD too long");
  }

  // padding for the string (center alignment)
  int pad1 = (16 - first.length()) / 2;
  int pad2 = (16 - second.length()) / 2;
  if (pad1 < 0) pad1 = 0;
  if (pad2 < 0) pad2 = 0;

  String tmp1, tmp2;
  for (int i = 0; i < pad1; i++) tmp1 += " ";
  tmp1 += first;
  for (int i = 0; i < pad2; i++) tmp2 += " ";
  tmp2 += second;

  // print to LCD
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(tmp1.c_str());
  lcd.setCursor(0, 1);
  lcd.print(tmp2.c_str());

  lcdTime = millis();
}

void activateAlarm(bool shouldAlarm) {
  if (shouldAlarm) {
    // blinking LED and buzzing
    digitalWrite(RED_LED_PIN, HIGH);
    delay(200);
    digitalWrite(RED_LED_PIN, LOW);
    delay(200);

    // notes in the melody:
    int melody[] = {
        NOTE_C4, NOTE_G3, NOTE_G3, NOTE_A3, NOTE_G3, 0, NOTE_B3, NOTE_C4};

    // note durations: 4 = quarter note, 8 = eighth note, etc.:
    int noteDurations[] = {
        4, 8, 8, 4, 4, 4, 4, 4};
    for (int thisNote = 0; thisNote < 8; thisNote++) {
      // to calculate the note duration, take one second divided by the note type.
      //e.g. quarter note = 1000 / 4, eighth note = 1000/8, etc.
      int noteDuration = 1000 / noteDurations[thisNote];
      tone(BUZZER_PIN, melody[thisNote], noteDuration);

      // to distinguish the notes, set a minimum time between them.
      // the note's duration + 30% seems to work well:
      int pauseBetweenNotes = noteDuration * 1.30;
```

```
      delay(pauseBetweenNotes);
      // stop the tone playing:
      noTone(BUZZER_PIN);
    }
  } else {
    // turn off the LED and stop buzzing
    digitalWrite(RED_LED_PIN, LOW);
    noTone(BUZZER_PIN);
  }
}

void receiveEvent(int howMany) {
  // read the incoming data
  String data;
  while (howMany--) {
    data += Wire.read();
  }
  //Serial.print("data: ");
  //Serial.println(data.c_str());

  // activate alarm
  if (data.indexOf("alarm") >= 0) {
    state = (data == "activate alarm") ? 2 : 0;
  }

  // activate system
  if (data.indexOf("system") >= 0) {
    systemActivated = (data == "system activated") ? true : false;
  }
}
```

- “pitches.h” from ToneMelody Arduino tutorial to generate different frequencies for the buzzer

```
#define NOTE_B0 31
#define NOTE_C1 33
#define NOTE_CS1 35
#define NOTE_D1 37
#define NOTE_DS1 39
#define NOTE_E1 41
#define NOTE_F1 44
#define NOTE_FS1 46
#define NOTE_G1 49
#define NOTE_GS1 52
#define NOTE_A1 55
#define NOTE_AS1 58
#define NOTE_B1 62
#define NOTE_C2 65
#define NOTE_CS2 69
#define NOTE_D2 73
#define NOTE_DS2 78
#define NOTE_E2 82
#define NOTE_F2 87
#define NOTE_FS2 93
#define NOTE_G2 98
#define NOTE_GS2 104
#define NOTE_A2 110
#define NOTE_AS2 117
#define NOTE_B2 123
#define NOTE_C3 131
#define NOTE_CS3 139
#define NOTE_D3 147
#define NOTE_DS3 156
#define NOTE_E3 165
#define NOTE_F3 175
#define NOTE_FS3 185
#define NOTE_G3 196
#define NOTE_GS3 208
#define NOTE_A3 220
#define NOTE_AS3 233
#define NOTE_B3 247
#define NOTE_C4 262
#define NOTE_CS4 277
#define NOTE_D4 294
#define NOTE_DS4 311
#define NOTE_E4 330
#define NOTE_F4 349
#define NOTE_FS4 370
#define NOTE_G4 392
#define NOTE_GS4 415
#define NOTE_A4 440
#define NOTE_AS4 466
#define NOTE_B4 494
#define NOTE_C5 523
#define NOTE_CS5 554
#define NOTE_D5 587
#define NOTE_DS5 622
#define NOTE_E5 659
#define NOTE_F5 698
#define NOTE_FS5 740
#define NOTE_G5 784
#define NOTE_GS5 831
#define NOTE_A5 880
#define NOTE_AS5 932
#define NOTE_B5 988
```

```
#define NOTE_C6 1047
#define NOTE_CS6 1109
#define NOTE_D6 1175
#define NOTE_DS6 1245
#define NOTE_E6 1319
#define NOTE_F6 1397
#define NOTE_FS6 1480
#define NOTE_G6 1568
#define NOTE_GS6 1661
#define NOTE_A6 1760
#define NOTE_AS6 1865
#define NOTE_B6 1976
#define NOTE_C7 2093
#define NOTE_CS7 2217
#define NOTE_D7 2349
#define NOTE_DS7 2489
#define NOTE_E7 2637
#define NOTE_F7 2794
#define NOTE_FS7 2960
#define NOTE_G7 3136
#define NOTE_GS7 3322
#define NOTE_A7 3520
#define NOTE_AS7 3729
#define NOTE_B7 3951
#define NOTE_C8 4186
#define NOTE_CS8 4435
#define NOTE_D8 4699
#define NOTE_DS8 4978
```

THANK YOU FOR READING!