Reagan Caliendo - rcali3 - rcali3@uic.edu
Munazza Shifa - mshifa2 - mshifa2@uic.edu
Marjea Ahad - mahad4 - mahad4@uic.edu
Chichi Ogugu - cogug3 - cogug3@uic.edu

Holly Jolly Christmas Display

The Holly Jolly Christmas Display is an interactive holiday showcase featuring synchronized lights, music, and special effects. Using four interconnected Arduinos, it combines LED light patterns, music playback, a countdown timer, and confetti effects for a festive atmosphere. Key features include a light sensor for automatic illumination, remote song control, and an emergency shutdown. Bluetooth coordination across the Arduinos ensures unified control, creating a seamless experience. This unique setup transforms a model house into an immersive Christmas display, blending synchronized lighting and sound with interactive elements to bring holiday cheer to life.

1. Overall Description of Project Idea

The Holly Jolly Christmas Display is a festive, interactive light and sound showcase designed to bring the holiday spirit to life on a miniature scale. Our project centers around a model house, where synchronized Christmas lights flash to the beat of pre-recorded holiday music, creating an immersive light show. This display is enhanced by a fan blowing “snow” or confetti for a realistic snowfall effect, an LCD screen displaying holiday messages such as “Happy Holidays”, and a countdown to the fan turning on on a 7-segment display.

This display will consist of four Arduinos, each Arduino will be dedicated to specific functions within the setup to ensure smooth operation. The main Arduino will act as the central control hub, receiving inputs from the remote, power button, and light sensor, while managing communication between other Arduinos through Bluetooth. A second Arduino will control the playback music and LED lights to synchronize with each song. A third and fourth Arduino will manage the special effects, including the fan and LCD messages.

Together, these different components will create a cohesive holiday experience, transforming an ordinary cardboard house model into a captivating Christmas light show. It is designed with the intention of providing family-friendly enjoyment. The Holly Jolly Christmas Display will be an engaging, family-friendly, and hands-on approach to holiday decor. It will showcase our team’s use of multiple Arduinos, I2C connection, and various interactive components into one seamless, original design.

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

The Holly Jolly Christmas Display will operate using four Arduinos, with each one responsible for overseeing various components of the display.

Arduino 1: Lights Synchronization:The first Arduino will control the Christmas lights around the display. This Arduino will manage two LED fairy light strips, adjusting flash patterns according to the beat of the music played on the buzzer. It will receive song data and control commands from the 4th Arduino, allowing the lights to synchronize perfectly with the audio, creating an engaging light show.

Arduino 2: Snow blowing fan: A fan will turn on from a signal from the controller Arduino to blow fake snow around the display. Since this is a special effect that doesn't happen constantly, there will also be a 7 segment display counting down to the fan turning on.

Arduino 3: Holiday Message: The other special effects Arduino will be connected to an LCD screen to display holiday messages to tie the whole display together.

Arduino 4: Control: The fourth Arduino will handle all user or automatic control elements, like the remote control,central power button and the light sensor. Through the remote, users will be able to change songs, adjust volume, and power the display on and off. The light sensor will adjust the brightness of the lights according to the ambient light level and the power button to manually switch the display on or off.

3. Final Plan for Use and Communication between the multiple Arduinos

The communication between the four Arduinos in *The Holly Jolly Christmas Display* will be by Bluetooth modules. Arduino 1 (LED)would take commands from the Arduino 2 (music) allowing for light and music sync. Arduino 1 would also receive signals from Arduino 4(control) for brightness adjustment and switching up control. Arduino 2 (will also receive control signals from Arduino 4, ensuring that users can control the audio settings smoothly. Arduino 4 will send activation signals to Arduino 3 when special effects need to be triggered or when users interact with specific controls. For instance, if a user wants to activate the fan for added visual effects, Arduino 4 will send this command to Arduino 3. Hence, the Arduino 4 serves as the control hub, handling all inputs either manually via the remote control and central power button or automatically via the light sensor.

4. Final Project Design stating Inputs/Outputs

*see diagram for project design

To summarize the project design diagram, there will be four Arduinos. The three inputs (below) will be put into one Arduino board. That board will then send messages to the three remaining Arduinos to determine when certain outputs should be on/off/display certain messages.

INPUTS:

- Remote Control
- Power Button
- Light Sensor

OUTPUT:

- LED Light Strips
- Speaker
- LCD Screen
- Fan
- 7 Segment Display

5. Description of the original work being attempted by your project.

The Holly Jolly Christmas Display is unique, original work because as a holiday-themed project it will combine multiple components into a single synchronized display. Unlike a traditional holiday display setup, our project will combine lights, music, and special effects all in one, these different elements will communicate and work together. We plan on using four

Arduinos and each will have a specific function. There will be a Bluetooth coordinated system that will allow different parts to respond to things such as user controls and environmentals factors. This will make for an original and interactive holiday experience.

A key feature of our project is an automatic and responsive setup. For example, a light sensor will turn on all the lights when it gets dark. Thi is something that is not found on a small scale display. We have also added a countdown timer on a 7-segment display and a fan that will blow confetti that will mimic artificial snow for a festive, holiday effect. There will also be a remote control that will allow users to change songs, adjust the volume, and control the lights; this will add to the convenience and interactive nature of the display that we aim for it to be.

In conclusion, *the Holly Jolly Christmas Display* will be unique and different from a typical holiday setup because of its engaging holiday experience that involves various elements combined into a unified, responsive system.

6. Discussion on how to build your project

To construct the project, we divided tasks among team members, each setting up an Arduino to handle specific functionalities. For the controller, we connected a remote, an IR receiver, a photoresistor, and a button to manage inputs. These components were wired to the Arduino using jumper wires, with the appropriate pins for power, ground, and data communication.

The lighting system was implemented by connecting fairy lights to the Arduino via USB, using a 4-module relay to handle the higher power requirements. The relay module was connected to the Arduino's digital pins for control, with the lights' power circuit wired through the relay channels to allow on/off switching.

For the special effects, we used two additional Arduinos. The first controlled a fan synchronized with a countdown timer. The fan's power supply was connected through a relay module, with the relay triggered by the Arduino to turn the fan on and off. The second Arduino operated an LCD screen, which was wired using I2C connections for efficient communication. This setup enabled the display of festive holiday messages. Careful attention was given to organizing and insulating the wiring to ensure reliability and safety during operation.

7. Discussion on how to your project is to be used

The project is designed to be user-friendly and versatile. Users can manually activate the device using a physical button or a remote control. The display can also automatically turn on when it gets dark by use of a photoresistor. The remote control also allows for customization, enabling users to change the song and light display, update the holiday messages shown on the LCD screen, and turn on the fan to create a dynamic snow-blowing effect around the display.

Timeline

Week 10 Oct 27 - Nov 2	Project Planning and Component Gathering	● Gathered materials needed to complete the project ● Researched code

		implementations for each
Week 11 Nov 3 - Nov 9	Basic Component Programming	• Started programming for each component
Week 12 Nov 10 - Nov 16	System Coordination	• Continued programming components • Tested bluetooth connections between components
Week 13 Nov 17 - Nov 23	Building *Design Presentation on Friday, Nov 22	• Began building the physical components of our project including our model house
Week 14 Nov 24 - Nov 30	Debugging/Final Adjustments	• Ironed out any bugs and perfected the design • Decided to use I2C connection/implemented that
Week 15 Dec 1 - Dec 6	*Project Demonstration on Friday, Dec 6	• Present project! • Completed project report

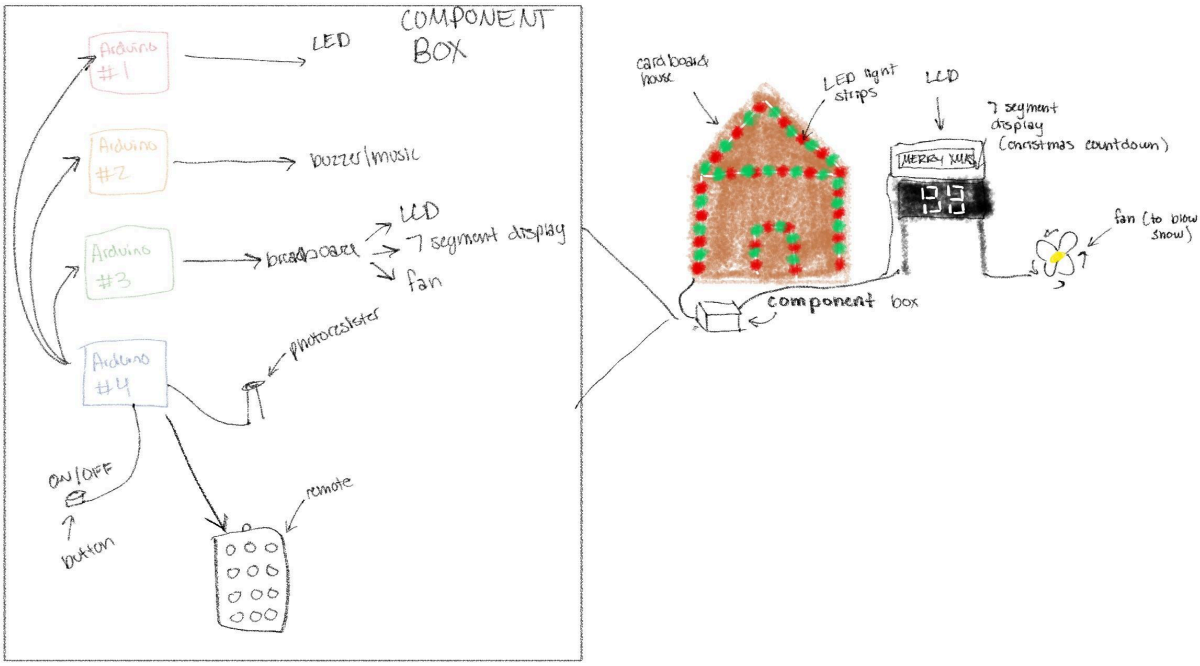
List of Materials Expected to be Needed

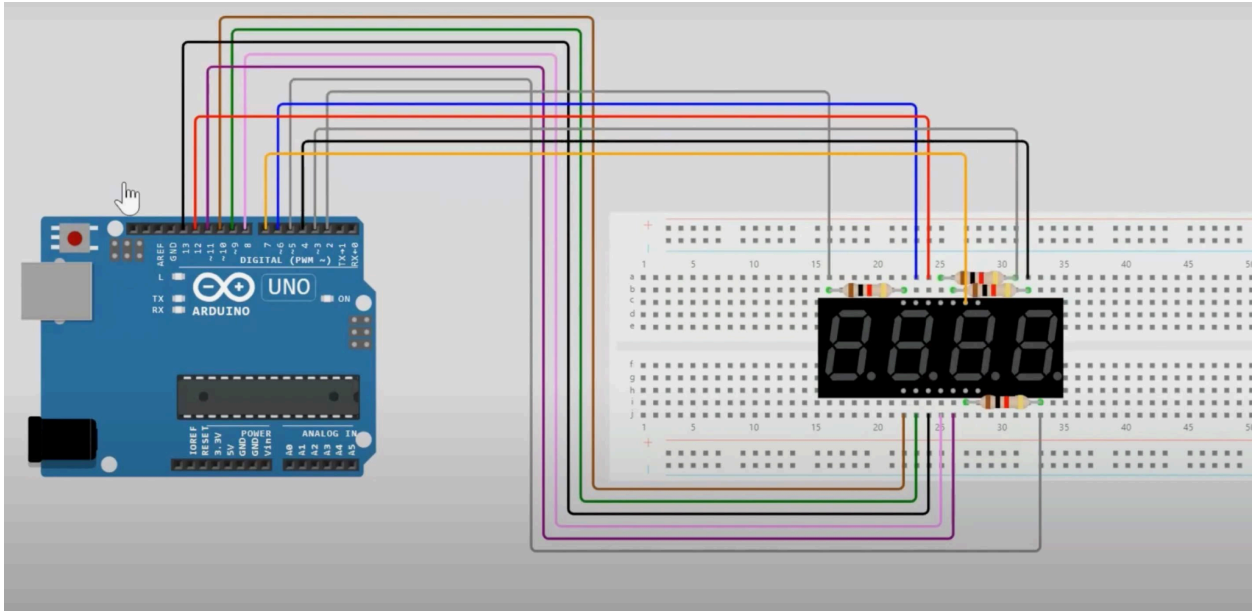
- Arduino Uno x4
- Breadboard x5
- Wires (many)
- 9V battery
- LED fairy lights with USB connection x2
- 4 module relay
- Buzzer x2
- Female USB adapter x2
- 7 Segment Display
- Fan Blade
- 3-6V Motor
- LCD Screen
- Potentiometer
- 220 Ohm Resistor
- 330 Ohm Resistor
- Remote and IR Receiver
- Photoresistor
- Transistor

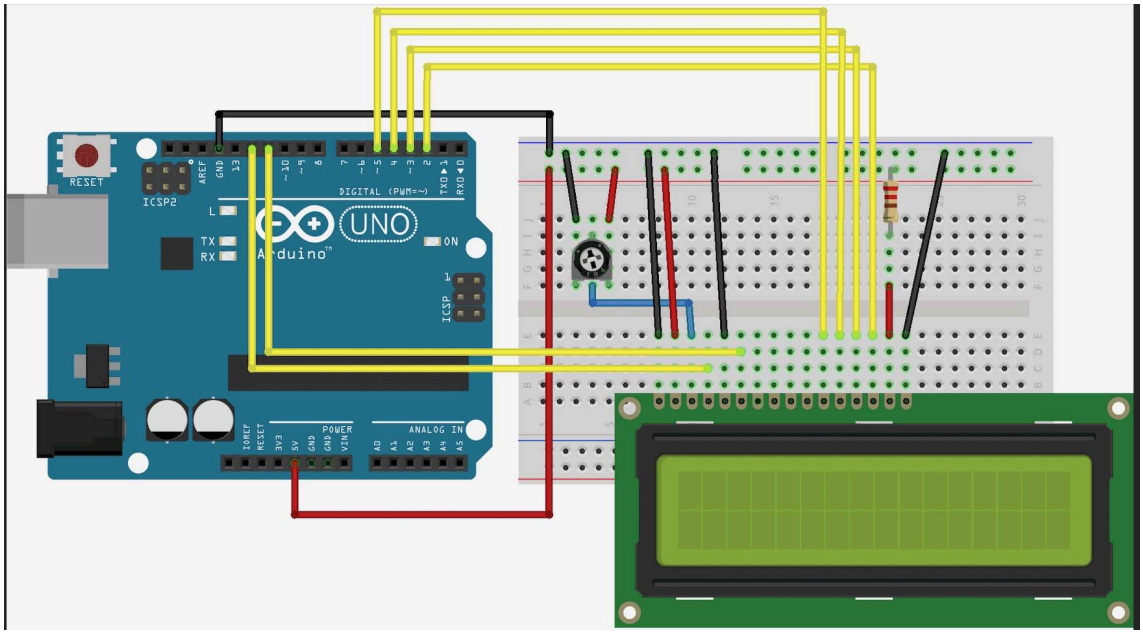
References (ongoing list throughout project)

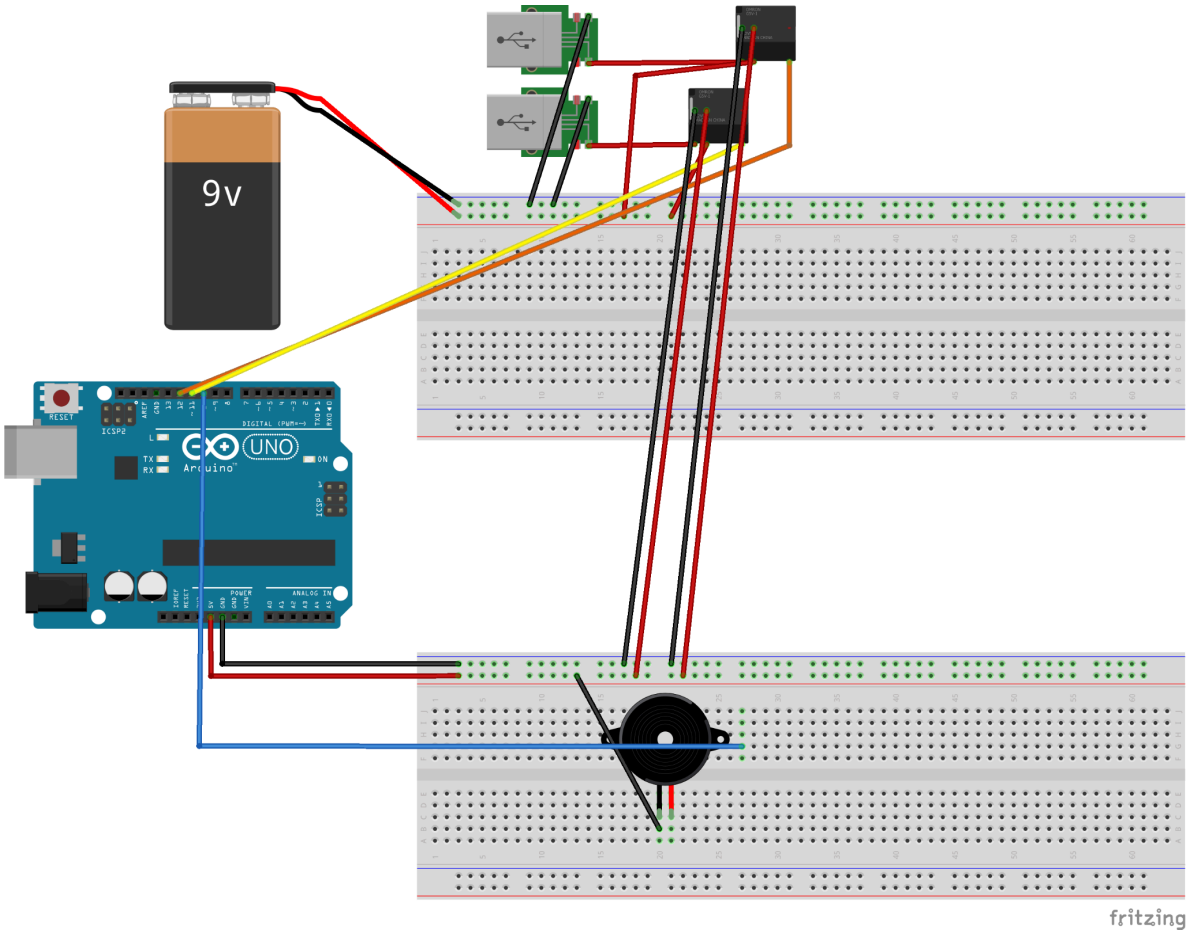
https://youtu.be/MafNmsmBW8s?feature=shared

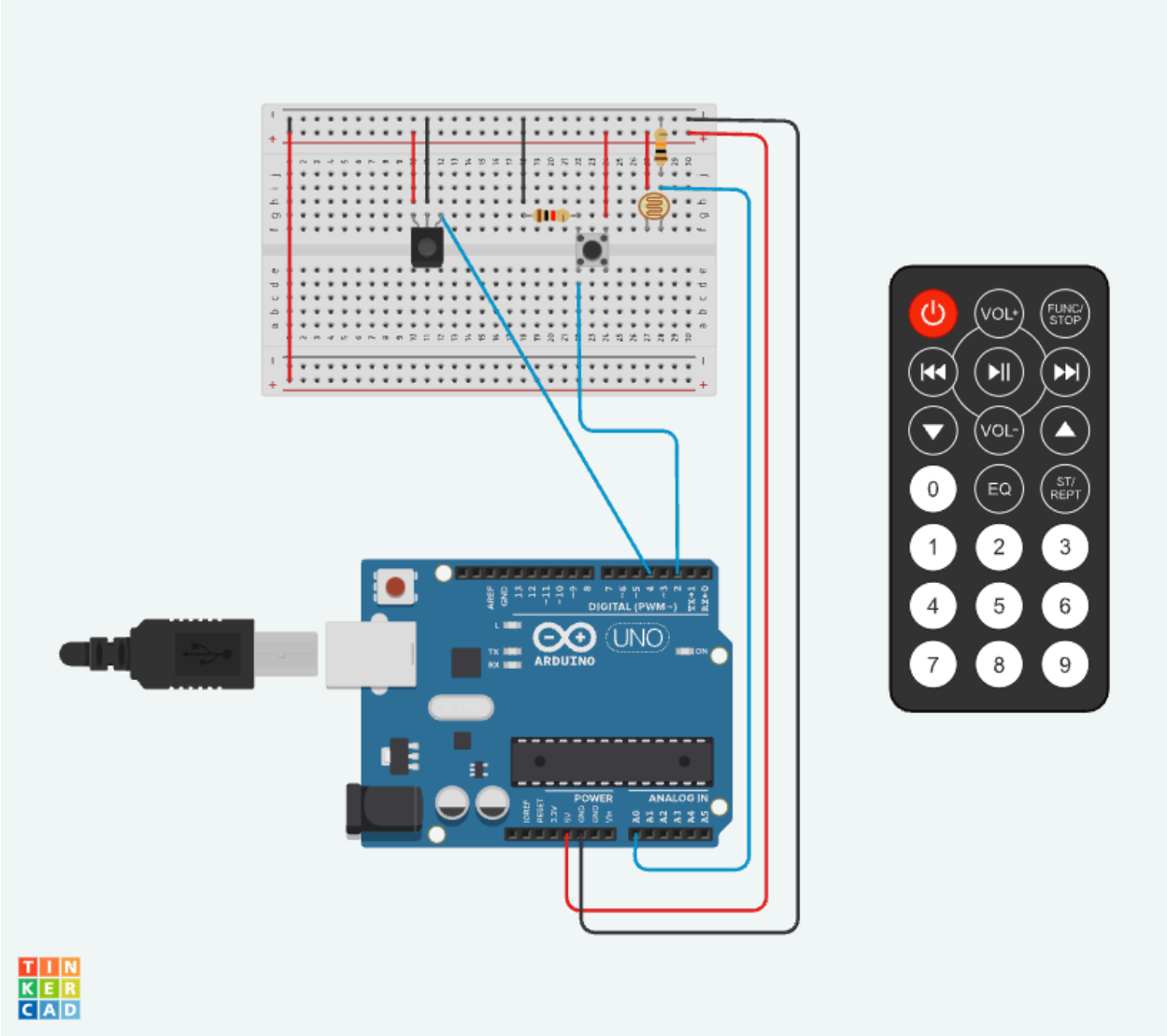
https://projecthub.arduino.cc/joshi/piezo-christmas-songs-c0e7aa
https://mschoeffler.com/2019/12/03/control-fairy-led-strings-with-a-relay-module-and-an-arduino/






RESET
ICSP2
UNO
Arduino
DIGITAL (PWM=~)
POWER
ANALOG IN


9v
RESET
ICSP2
UNO
Arduino
DIGITAL (PWM=~)
POWER
ANALOG IN
fritzing



Code Sketches

LCD SCREEN

```
#include <LiquidCrystal.h>

// Initialize the library by associating the LCD pins with Arduino pins
const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

// Array of Christmas-themed messages split into two lines
const char* messages[][2] = {
  {"Have a Holly", "Jolly Christmas!"},
  {"Tis the season", "to be jolly"},
  {"Wishing you a", "Merry Christmas!"},
```

```
  {"Happy Holidays", "Everyone!"},
  {"Ho Ho Ho!", "Santa's Coming!"}
};
const int numMessages = sizeof(messages) / sizeof(messages[0]);

// Timing variables for switching messages
unsigned long previousMillis = 0;
const long interval = 10000; // 10 seconds interval for each message

int currentMessage = 0; // Index to track the current message

bool startDisplay = false; // Flag to track if the display should start

void setup() {
  Serial.begin(9600); // Start Serial communication at 9600 baud
  lcd.begin(16, 2);    // Initialize the LCD (but keep it off initially)
  lcd.noDisplay();     // Turn off the display
}

void loop() {
  // Check for a signal from the master Arduino
  if (Serial.available() > 0) {
    String command = Serial.readStringUntil('\n');
    command.trim();

    if (command == "START") {
      startDisplay = true;
      lcd.display(); // Turn on the LCD display
      lcd.clear();
      lcd.setCursor(0, 0);
      // lcd.print("Starting Holly");
      lcd.setCursor(0, 1);
      // lcd.print("Jolly Display!");
      delay(2000); // Brief delay before starting the loop
    }
  }

  // Run the display if the signal was received
  if (startDisplay) {
    unsigned long currentMillis = millis();

    // Check if it's time to switch to the next message
    if (currentMillis - previousMillis >= interval) {
      previousMillis = currentMillis;
```

```
    // Move to the next message
    currentMessage++;
    if (currentMessage >= numMessages) {
      currentMessage = 0; // Reset to the first message
    }

    // Clear the screen and display the current message
    lcd.clear();
    lcd.setCursor(0, 0); // Top line
    lcd.print(messages[currentMessage][0]);
    lcd.setCursor(0, 1); // Bottom line
    lcd.print(messages[currentMessage][1]);
   }
  }
}
```

CONTROL ARDUINO

```
#include <Arduino.h>
#include <IRremote.hpp>
#include <Wire.h>

// Define Slave Setup
#define SLAVE_ADDRESS 8
#define SLAVE_ADDRESS1 9
#define ANSWERSIZE 4

// Button setup
#define BUTTON_PIN1 2
volatile byte buttonState = LOW;

// IR receiver setup
const int RECV_PIN = 4;
IRrecv irrecv(RECV_PIN);
bool isSystemOn = false;          // Current system state
bool lastSentState = false;       // Last sent state to the slave
unsigned long received_val = 0;
int song = 1;

// Photoresistor setup
int pResistor = A0;  // Photoresistor input
```

```
unsigned long p_value;

// Timer setup
int interval = 1000;  // 1-second interval
unsigned long start = 0;
bool auto_light = true;

// New variables for handling countdown completion
bool countdownComplete = false;
bool songStarted = false;
unsigned long lastCheckTime = 0;
const int CHECK_INTERVAL = 1000;  // Check countdown status every second

void setup() {
    Serial.begin(9600);
    Wire.begin();  // Join I2C bus as master
    irrecv.enableIRIn();  // Start the IR receiver
}

void loop() {
    // Check IR remote
    IRremote_switch();

    // Handle countdown completion and song triggering
    if (isSystemOn && !songStarted) {
        unsigned long currentTime = millis();
        if (currentTime - lastCheckTime >= CHECK_INTERVAL) {
            String response = slaveResponse();
            Serial.println("Countdown Status: " + response);

            if (response == "Done") {
                Serial.println("Countdown complete, starting song
playback");
                send_song_message("ON");  // Send ON to the song Arduino
                songStarted = true;  // Prevent repeated sending
                Serial.println("End");
            }
            lastCheckTime = currentTime;
        }
    }

    // Check photoresistor and send light state
```

```
    unsigned long curr = millis();
    if (auto_light && curr - start >= interval) {
        p_value = analogRead(pResistor);
        check_lights(p_value);
        start = millis();
    }

    delay(100);
}

// Function to handle light-based system control
void check_lights(unsigned long value) {
    if(value < 600){
        if(!isSystemOn){  // System turns on at Dark
            Serial.print("Photoresistor value: ");
            Serial.println(p_value);
            isSystemOn = !isSystemOn;
            system_state_changed();  // Use new function to handle state
change
        }
    }
}

// New function to handle system state changes
void system_state_changed() {
    if (isSystemOn) {
        send_message("ON");
        songStarted = false;  // Reset song state when system turns on
        countdownComplete = false;  // Reset countdown state
    } else {
        send_message("OFF");
        send_song_message("OFF");  // Turn off song Arduino
        songStarted = false;  // Reset song state
    }
}

void IRremote_switch(){
    if (irrecv.decode()) {
        unsigned long value = irrecv.decodedIRData.decodedRawData;

        if (value == 0xFFFFFFFF) {
            value = received_val;
```

```
            return;
        }

        switch (value) {
            case 0xFA05EF00:  // ON/OFF button
                isSystemOn = !isSystemOn;
                Serial.println(isSystemOn ? "System is ON" : "System is
OFF");
                system_state_changed();  // Use new function to handle
state change
                break;

            case 0xEC13EF00:  // Song 1 button
                if(isSystemOn && song != 1){
                    Serial.println("Play Song 1");
                    send_song_message("Change");
                    song = 1;
                }
                break;

            case 0xED12EF00:  // Song 2 button
                if(isSystemOn && song != 2){
                    Serial.println("Play Song 2");
                    send_song_message("Change");
                    song = 2;
                }
                break;

            case 0xFD02EF00:  // Auto_light setting
                auto_light = !auto_light;
                Serial.println(auto_light ? "Auto_Light is On" :
"Auto_Light is off");
                break;

            default:
                break;
        }
        received_val = value;
        irrecv.resume();
    }
}
```

```
void send_message(String text) {
    Serial.println("Sending to Countdown Slave: " + text);
    Wire.beginTransmission(SLAVE_ADDRESS);
    Wire.write(text.c_str());
    Wire.endTransmission();
}

void send_song_message(String text) {
    Serial.println("Sending to Song Slave: " + text);
    Wire.beginTransmission(SLAVE_ADDRESS1);
    Wire.write(text.c_str());
    Wire.endTransmission();
}

String slaveResponse() {
    String response = "";
    Wire.requestFrom(SLAVE_ADDRESS, ANSWERSIZE);
    while (Wire.available()) {
        char c = Wire.read();
        response += c;
    }
    return response;
}
```

LIGHTS ARDUINO

```
#include <Wire.h>

/************************************************
     PITCHES
  ************************************************/
#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 69qa
#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

//I2C Communication
#define SLAVE_ADDRESS1 9
#define ANSWERSIZE 8

int jingle_bells_melody[] = {
  NOTE_E5, NOTE_E5, NOTE_E5,
  NOTE_E5, NOTE_E5, NOTE_E5,
  NOTE_E5, NOTE_G5, NOTE_C5, NOTE_D5,
  NOTE_E5,
  NOTE_F5, NOTE_F5, NOTE_F5, NOTE_F5,
  NOTE_F5, NOTE_E5, NOTE_E5, NOTE_E5, NOTE_E5,
  NOTE_E5, NOTE_D5, NOTE_D5, NOTE_E5,
  NOTE_D5, NOTE_G5
};

int jingle_bells_tempo[] = {
  8, 8, 4,
  8, 8, 4,
  8, 8, 8, 8,
  2,
  8, 8, 8, 8,
  8, 8, 8, 16, 16,
  8, 8, 8, 8,
  4, 4
};

int  wish_melody[] = {
  NOTE_B3,
  NOTE_F4, NOTE_F4, NOTE_G4, NOTE_F4, NOTE_E4,
  NOTE_D4, NOTE_D4, NOTE_D4,
  NOTE_G4, NOTE_G4, NOTE_A4, NOTE_G4, NOTE_F4,
```

```
  NOTE_E4, NOTE_E4, NOTE_E4,
  NOTE_A4, NOTE_A4, NOTE_B4, NOTE_A4, NOTE_G4,
  NOTE_F4, NOTE_D4, NOTE_B3, NOTE_B3,
  NOTE_D4, NOTE_G4, NOTE_E4,
  NOTE_F4
};

int  wish_tempo[] = {
  4,
  4, 8, 8, 8, 8,
  4, 4, 4,
  4, 8, 8, 8, 8,
  4, 4, 4,
  4, 8, 8, 8, 8,
  4, 4, 8, 8,
  4, 4, 4,
  2
};

#define melodyPin 12
#define lights1_pin 2
#define lights2_pin 3

unsigned long prevNoteTime = 0;  // Time for the last note
unsigned long prevLEDTime = 0;   // Time for the last LED toggle
bool light1State = false;        // Current state of LED 1
int currentNote = 0;             // Current note being played
unsigned long currentTime;
int song = 1;

// System state management
bool isSystemOn = false;    // Tracks the state of the system
bool isPlaying = false;     // Tracks if music is currently
playing


void setup() {
```

```
  Serial.begin(9600);
  Wire.begin(SLAVE_ADDRESS1);
  Wire.onReceive(receiveEvent);

  pinMode(lights1_pin, OUTPUT);
  pinMode(lights2_pin, OUTPUT);
  pinMode(melodyPin, OUTPUT);

  // Initialize all outputs to OFF state
  stopEverything();
}

void loop() {
  currentTime = millis();

  // Only play if system is ON
  if (isSystemOn) {
    play_song(song);
  }
}

// Function to immediately stop all outputs
void stopEverything() {
  // Stop the buzzer
  digitalWrite(melodyPin, LOW);
  // Turn off both lights
  digitalWrite(lights1_pin, HIGH);
  digitalWrite(lights2_pin, HIGH);
  // Reset playback variables
  currentNote = 0;
  prevNoteTime = currentTime;
  prevLEDTime = currentTime;
  light1State = false;
  isPlaying = false;
}

void play_song(int song_num) {
```

```
  int size;
  int* melody;
  int* tempo;

  // Select the appropriate song arrays
  if (song_num == 1) {
    size = sizeof(jingle_bells_melody) / sizeof(int);
    melody = jingle_bells_melody;
    tempo = jingle_bells_tempo;
  } else {
    size = sizeof(wish_melody) / sizeof(int);
    melody = wish_melody;
    tempo = wish_tempo;
  }

  if (currentNote < size) {
    isPlaying = true;
    int noteDuration = 2000 / tempo[currentNote];

    // Play the current note
    if (currentTime - prevNoteTime < noteDuration) {
      buzz(melodyPin, melody[currentNote], noteDuration);
    } else {
      buzz(melodyPin, 0, 0); // Turn off the buzzer
      currentNote++;
      prevNoteTime = currentTime;
    }

    // Handle LED effects
    if (currentTime - prevLEDTime >= noteDuration / 2) {
      prevLEDTime = currentTime;
      light1State = !light1State;
      digitalWrite(lights1_pin, light1State ? LOW : HIGH);
      digitalWrite(lights2_pin, light1State ? HIGH : LOW);
    }
  } else {
    // Reset when song completes
```

```
    currentNote = 0;
    prevNoteTime = currentTime;
    if (isPlaying) {
      stopEverything();
    }
  }
}

void buzz(int targetPin, long frequency, long length) {
  // Only produce sound if system is ON
  if (!isSystemOn) {
    digitalWrite(targetPin, LOW);
    return;
  }

  if (frequency > 0) {
    long delayValue = 1000000 / frequency / 2;
    long numCycles = frequency * length / 1000;
    for (long i = 0; i < numCycles; i++) {
      // Check if system turned OFF during long notes
      if (!isSystemOn) {
        digitalWrite(targetPin, LOW);
        return;
      }
      digitalWrite(targetPin, HIGH);
      delayMicroseconds(delayValue);
      digitalWrite(targetPin, LOW);
      delayMicroseconds(delayValue);
    }
  } else {
    digitalWrite(targetPin, LOW);
  }
}

void receiveEvent(int bytes) {
  char received[ANSWERSIZE];
  int i = 0;
```

```
  while (Wire.available() && i < sizeof(received)) {
    received[i++] = Wire.read();
  }
  received[i] = '\0';

  if (strcmp(received, "ON") == 0) {
    Serial.println("System turning ON");
    isSystemOn = true;
    // Don't reset currentNote so it can continue from where it
was
  }
  else if (strcmp(received, "OFF") == 0) {
    Serial.println("System turning OFF");
    isSystemOn = false;
    stopEverything();  // Immediately stop all outputs
  }
  else if (strcmp(received, "Change") == 0) {
    Serial.print("Changing song from ");
    Serial.print(song);
    song = (song == 1) ? 2 : 1;
    Serial.print(" to ");
    Serial.println(song);

    // Reset playback state for new song
    currentNote = 0;
    prevNoteTime = currentTime;
    prevLEDTime = currentTime;
  }
}
```

```
FAN ARDUINO
#include "SevSeg.h"
#include <Wire.h>

SevSeg sevseg;
```

```
const int buzzer = A0;  // Define buzzer pin at A0
const int fanPin = A1;  // Define fan control pin at A1

//I2C Communication
#define SLAVE_ADDRESS 8
#define ANSWERSIZE 4

bool isSystemOn = false;
bool isCountdownComplete = false;  // New flag to track
countdown completion

void setup() {
  Serial.begin(9600);
  Wire.begin(SLAVE_ADDRESS);
  Wire.onReceive(receiveEvent);
  Wire.onRequest(requestEvent);  // Add handler for master
requests

  byte numDigits = 4;
  byte digitPins[] = {2, 3, 4, 5};
  byte segmentPins[] = {6, 7, 8, 9, 10, 11, 12, 13};
  bool resistorsOnSegments = 0;

  sevseg.begin(COMMON_CATHODE, numDigits, digitPins,
segmentPins, resistorsOnSegments);
  sevseg.setBrightness(90);

  pinMode(buzzer, OUTPUT);
  pinMode(fanPin, OUTPUT);
  digitalWrite(fanPin, LOW);
}

void loop() {
  if (!isSystemOn) {
    // Reset completion flag when system is off
    isCountdownComplete = false;
    digitalWrite(fanPin, LOW);
```

```
    noTone(buzzer);
    sevseg.blank();
    return;
  }

  if (isSystemOn && !isCountdownComplete) {  // Only start
countdown if not already complete
    // Countdown from 10
    for (int count = 10; count > 0; count--) {
      sevseg.setNumber(count, 0);
      tone(buzzer, 1000);
      delay(100);
      noTone(buzzer);

      unsigned long startTime = millis();
      while (millis() - startTime < 1000) {
        sevseg.refreshDisplay();
        if (!isSystemOn) {
          return;
        }
      }
    }

    // Final buzzer and display
    sevseg.setNumber(0, 0);
    tone(buzzer, 1000);
    unsigned long buzzStartTime = millis();
    while (millis() - buzzStartTime < 3000) {
      sevseg.refreshDisplay();
      if (!isSystemOn) {
        return;
      }
    }
    noTone(buzzer);

    // Turn on fan and set completion flag
    digitalWrite(fanPin, HIGH);
```

```
    isCountdownComplete = true;  // Mark countdown as complete

    // Display the date
    int date = 1123;
    sevseg.setNumber(date);

    delay(15000);
  }

  // Keep refreshing display after countdown
  sevseg.refreshDisplay();
}

// Handler for receiving commands from master
void receiveEvent(int bytes) {
  char received[ANSWERSIZE];
  int i = 0;

  while (Wire.available() && i < sizeof(received)) {
    received[i++] = Wire.read();
  }
  received[i] = '\0';

  if (strcmp(received, "ON") == 0) {
    isSystemOn = true;
    isCountdownComplete = false;  // Reset completion flag when
turning on
    Serial.println("System is now ON.");
  } else if (strcmp(received, "OFF") == 0) {
    isSystemOn = false;
    Serial.println("System is now OFF.");
    digitalWrite(fanPin, LOW);
    noTone(buzzer);
    sevseg.blank();
  }
  Serial.println(received);
}
```

```
// Handler for responding to master requests
void requestEvent() {
  if (isCountdownComplete) {
    Wire.write("Done");  // Send "Done" when countdown is
complete
  } else {
    Wire.write("Run ");  // Send "Run " (with space to match 4
chars) when still running
  }
}
```