Roll Number : **67**

Vivekanand Education Society’s Institute of Technology

Hashu Advani memorial Complex

Collector’s Colony, R C Marg, Chembur Mumbai 400074, Phone Number 022-61532532



**CERTIFICATE**

Certified that **Mr. V Krishnasubramaniam** of **D15A (TE)** semester **VI** has successfully completed necessary experiments in Sensor Network Lab under my supervision in VES Institute of Technology during the academic Year **2021-2022.**

**Prof. Smita Jangale Prof. Smita Jangale**

Lab Incharge Subject Teacher

**Dr. Shalu Chopra**

Principal Head of Department

Table of Contents

| **Sr. No** | **Title** | **Page Number** |
| --- | --- | --- |
| 1 | Assignment 1 | 3 |
| 2 | Assignment 2 | 9 |
| 3 | Assignment 3 | 14 |
| 4 | Project | 20 |

**Assignment 01**

**Aim**

To establish Arduino to Arduino communication using NRF.

**Theory**

**NRF24L01 Module**

The NFR24L01 is a transceiver module which means that it can both send and receive the data.

These modules are very cheap, smaller in size, and has a lot of specifications. Some of the specifications of these modules are as follows

Specifications of NRF24L01 Module

· Power consumption is around 12mA during transmission which is even lesser than the led. · It can operate with baud rates from 250Kbps up to 2 Mbps.

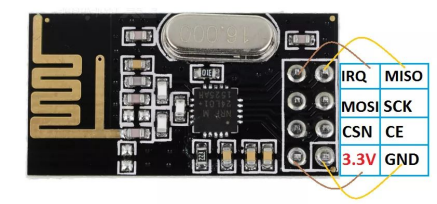
· Its range can reach up to 100 meters if used in open space and with an antenna. · It can both send and receive the data simultaneously.

· Each module can communicate with up to 6 other modules.

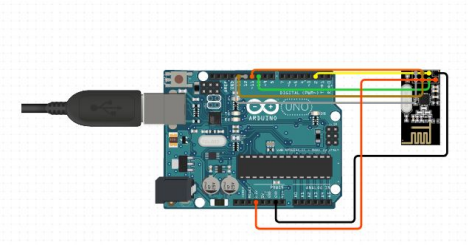
· It uses the 2.4 GHz band.

· It can send 1 to 25 bytes of raw data at the transmission rate of 1 MB. · It has 125 different channels.

**Pinout diagram of NRF24L01 module:**

****

**NRF to Arduino connections:**

****

**Creating a Unidirectional Communication :**

**Arduino 1 (Transmitter) Code:**

#include <SPI.h>

#include <nRF24L01.h>

#include <RF24.h>

RF24 radio(9, 10); // CE, CSN

const byte address[6] = "00011"; //Byte of array representing the address. This is the address where we will send the data. This should be same on the receiving side.

void setup() {

Serial.begin(9600);

radio.begin(); //Starting the Wireless communication

radio.openWritingPipe(address); //Setting the address where we will send the data radio.setPALevel(RF24\_PA\_MIN); //You can set it as minimum or maximum depending on the distance between the transmitter and receiver.

radio.stopListening(); //This sets the module as transmitter

Serial.println("Arduino 1");

}

void loop()

{

const char trans[] = "Ardiono1\_Assignment2\_02\_06\_23";

radio.write(&trans, sizeof(trans));//Sending the message to receiver

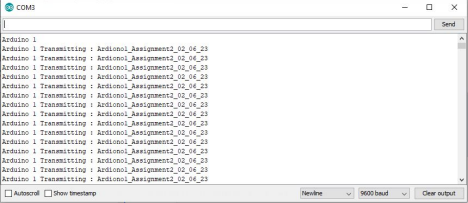
Serial.print("Arduino 1 Transmitting : ");

Serial.println(trans);

delay(100);

}

**Output :**

****

**Arduino 2 (Receiver) Code:**

#include <SPI.h>

#include <nRF24L01.h>

#include <RF24.h>

RF24 radio(9, 10); // CE, CSN

const byte address[6] = "00011";

void setup() {

Serial.begin(9600);

radio.begin();

radio.openReadingPipe(0, address); //Setting the address at which we will receive the data radio.setPALevel(RF24\_PA\_MIN); //You can set this as minimum or maximum depending on the distance between the transmitter and receiver.

radio.startListening(); //This sets the module as receiver

Serial.println("Arduino 2");

}

void loop()

{

if(radio.available()) //Looking for the data.

{

char text[32] = ""; //Saving the incoming data

radio.read(&text, sizeof(text)); //Reading the data

Serial.print("Arduino 2 Receiving : ");

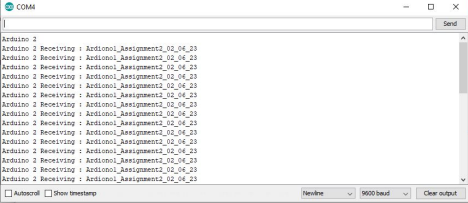
Serial.println(text);

}

delay(100);

}

**Output :**

****

**Creating a Bidirectional Communication:**

**Arduino 1 :**

#include <SPI.h>

#include <nRF24L01.h>

#include <RF24.h>

RF24 radio(9, 10); // CE, CSN

const byte addresses [][6] = {"00001", "00010"}; //Setting the two addresses. One for transmitting and one for receiving

void setup() {

Serial.begin(9600);

radio.begin(); //Starting the radio communication

radio.openWritingPipe(addresses[1]); //Setting the address at which we will send the data radio.openReadingPipe(1, addresses[0]); //Setting the address at which we will receive the data radio.setPALevel(RF24\_PA\_MIN); //You can set it as minimum or maximum depending on the distance between the transmitter and receiver.

Serial.println("Arduino 1 Bidirectional :");

}

void loop()

{

delay(500);

radio.stopListening(); //This sets the module as transmitter

const char trans[] = "Arduino1\_Assignement2\_02\_06\_23";

radio.write(&trans, sizeof(trans)); //Sending the data

Serial.print("Transmitting Data : ");

Serial.println(trans);

delay(500);

radio.startListening(); //This sets the module as receiver

char rec[32] = "";

if(radio.available()) //Looking for incoming data

{

radio.read(&rec, sizeof(rec)); //Reading the data

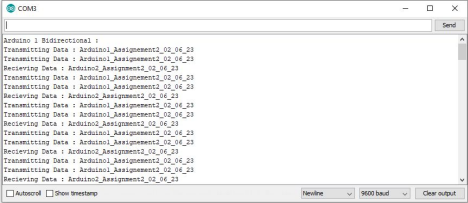
Serial.print("Recieving Data : ");

Serial.println(rec);

}

}

**Output :**

****

**Arduino 2 :**

#include <SPI.h>

#include <nRF24L01.h>

#include <RF24.h>

RF24 radio(9, 10); // CE, CSN

const byte addresses [][6] = {"00001", "00010"}; //Setting the two addresses. One for transmitting and one for receiving

void setup() {

Serial.begin(9600);

radio.begin(); //Starting the radio communication

radio.openWritingPipe(addresses[0]); //Setting the address at which we will send the data radio.openReadingPipe(1, addresses[1]); //Setting the address at which we will receive the data radio.setPALevel(RF24\_PA\_MIN); //You can set it as minimum or maximum depending on the distance between the transmitter and receiver.

Serial.println("Arduino 2 Bidirectional :");

}

void loop()

{

char rec[32] = "";

delay(500);

radio.startListening(); //This sets the module as receiver

if (radio.available()) //Looking for incoming data

{

radio.read(&rec, sizeof(rec));

Serial.print("Recieving Data : ");

Serial.println(rec);

delay(500);

radio.stopListening(); //This sets the module as transmitter

const char trans[] = "Arduino2\_Assignment2\_02\_06\_23";

radio.write(&trans, sizeof(trans)); //Sending the data

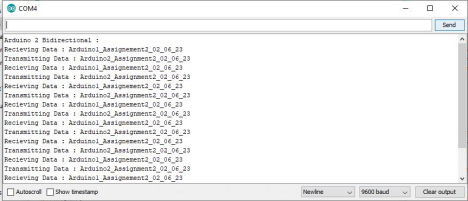
Serial.print("Transmitting Data : ");

Serial.println(trans);

}

}

**Output :**

****

**Conclusion:** Hence, Arduino to Arduino communication using NRF was established.

**Assignment 02**

**Aim**

To establish Arduino to Raspberry pi communication using NRF.

**Theory**

**NRF24L01 Module**

The NFR24L01 is a transceiver module which means that it can both send and receive the data.

These modules are very cheap, smaller in size, and have a lot of specifications. Some of the specifications of these modules are as follows

Specifications of NRF24L01 Module

· Power consumption is around 12mA during transmission which is even lesser than the led. · It can operate with baud rates from 250Kbps up to 2 Mbps.

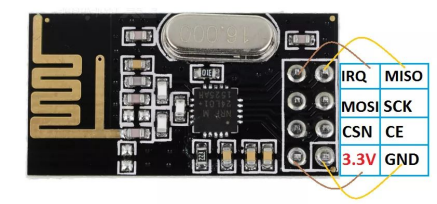
· Its range can reach up to 100 meters if used in open space and with an antenna. · It can both send and receive the data simultaneously.

· Each module can communicate with up to 6 other modules.

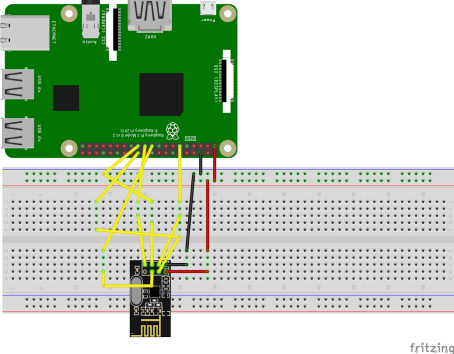
· It uses the 2.4 GHz band.

· It can send 1 to 25 bytes of raw data at the transmission rate of 1 MB. · It has 125 different channels.

**Pinout diagram of NRF24L01 module:**

****

**NRF to Raspberry Pi connections:**

****

**Steps of implementation:**

**After powering up the Raspberry Pi:**

1. In the terminal type:

sudo raspi-config

and press Enter on your PC keyboard.

2. Go to Advanced Options.

3. Go to SPI.

4. Enable SPI Interface by clicking on <Yes>.

5. Reboot the Pi. In the terminal, type:

sudo reboot

6. In the terminal type:

sudo apt-get update

7. After it has rebooted, install Python development tools. In the terminal, type: sudo apt-get install python3-dev. # python3

8. Download this package. In the terminal type:

wget https://github.com/Gadgetoid/py-spidev/archive/master.zip

9. In the terminal type: ls to check that you have master.zip in your current working directory.

10. To unzip, in the terminal type:

unzip master.zip

11. Remove master.zip:

rm master.zip

12. Navigate to py-spidev-master directory(folder)

13. Type in the terminal: ls, to view the contents of the directory(folder).

14. In the terminal run the commands:

sudo python3 setup.py install # python3

15. Create directory NRF24L01 on the Desktop

16. In the terminal type:

git clone https://github.com/Blavery/lib\_nrf24

17. Navigate to directory just downloaded

18. Copy lib\_nrf24.py to the NRF24L01 directory. In the terminal type:

cp lib\_nrf24.py ~/Desktop/NRF24L01/

19. Navigate to the NRF24L01 directory.

20. In Python 3 (IDLE), create a new file.

21. The name could be: ReceiveArduino.py. Save it in the NRF24L01 directory(folder). 22. In the file ReceiveArduino.py, write the following code with comments (line starting with “#”) for clear understanding and save (press Ctrl + S on your PC keyboard) the file.

**Rpi Code :**

import RPi.GPIO as GPIO

from lib\_nrf24 import NRF24

import spidev

GPIO.setmode(GPIO.BCM)

pipes = [[0xE8, 0xE8, 0xF0, 0xF0, 0xE1], [0xF0, 0xF0, 0xF0, 0xF0, 0xE1]]

radio = NRF24(GPIO, spidev.SpiDev())

radio.begin(0, 17)

radio.setPayloadSize(32)

radio.openReadingPipe(1, pipes[1])

radio.setChannel(0x55)

radio.setDataRate(NRF24.BR\_1MBPS)

radio.setPALevel(NRF24.PA\_MIN)

radio.setAutoAck(True)

radio.enableDynamicPayloads()

radio.enableAckPayload()

radio.printDetails()

radio.startListening()

while(1):

while not radio.available(0):

time.sleep(1 / 100)

receivedMessage = []

radio.read(receivedMessage, radio.getDynamicPayloadSize())

print("Received: {}".format(receivedMessage))

print("Translating the receivedMessage into unicode characters")

string = ""

for n in receivedMessage:

# Decode into standard unicode set

if (n >= 32 and n <= 126):

string += chr(n)

print("Out received message decodes to: {}".format(string))

**For the Arduino Setup :**

1. Open Arduino IDE

2. In the above menu bar go to: Sketch -> Include Library -> Manage Libraries… 3. Search for RF24, select RF24 by TMRh20 Version 1.1.6 and click Install. After installation close the library manage.

4. Load the following code in Arduino and run:

**Arduino Code :**

//Send.ino

#include<SPI.h>

#include<RF24.h>

// ce, csn pins

RF24 radio(9, 10);

void setup(void){

Serial.begin(9600);

radio.begin();

radio.setPALevel(RF24\_PA\_MAX);

radio.setChannel(0x55);

radio.openWritingPipe(0xF0F0F0F0E1LL);

radio.enableDynamicPayloads();

radio.powerUp();

}

void loop(void){

const char trans[] = "Arduino\_Assignment3\_02\_06\_23";

radio.write(&trans, sizeof(trans));

Serial.print("Transmitting Data : ");

Serial.println(trans);

delay(2000);

}

**Output :**

**Transmitted Data from Arduino:**

**Conclusion:** Hence, Arduino to Raspberry pi communication using NRF was established.

**Assignment 03**

**Aim**

To understand the usage of Cupcarbon tool for simulating wireless communications

**Theory**

CupCarbon is a Smart City and Internet of Things Wireless Sensor Network (SCI-WSN) simulator. Its objective is to design, visualize, debug and validate distributed algorithms for monitoring, environmental data collection, etc., and to create environmental scenarios such as fires, gas, mobiles, and generally within educational and scientific projects. Not only it can help to visually explain the basic concepts of sensor networks and how they work; it may also support scientists to test their wireless topologies, protocols, etc.

CupCarbon offers two simulation environments. The first simulation environment enables the design of mobility scenarios and the generation of natural events such as fires and gas as well as the simulation of mobiles such as vehicles and flying objects (e.g. UAVs, insects, etc.). The second simulation environment represents a discrete event simulation of wireless sensor networks which takes into account the scenario designed on the basis of the first environment. Networks can be designed and prototyped by an ergonomic and easy to use interface using the OpenStreetMap (OSM) framework to deploy sensors directly on the map. It includes a script called SenScript, which allows to program and to configure each sensor node individually. From this script, it is also possible to generate codes for hardware platforms such as Arduino/XBee. This part is not fully implemented in CupCarbon, it allows us to generate codes for simple networks and algorithms.

CupCarbon simulation is based on the application layer of the nodes. This makes it a real compliment to existing simulators. It does not simulate all protocol layers due to the complex nature of urban networks which need to incorporate other complex and resource-consuming information such as buildings, roads, mobility, signals, etc.

CupCarbon offers the possibility to simulate algorithms and scenarios in several steps. For example, there could be a step for determining the nodes of interest, followed by a step related to the nature of the communication between these nodes to perform a given task such as the detection of an event, and finally, a step describing the nature of the routing to the base station in case that an event is detected. The current version of CupCarbon allows configuring the nodes dynamically in order to be able to split nodes into separate networks or to join different networks, a task which is based on the network addresses and the channel. The energy consumption can be calculated and displayed as a function of the simulated time. This allows clarifying the structure, feasibility, and realistic implementation of a network before its real deployment. The propagation visibility and the interference models are integrated and include the ZigBee, LoRa, and WiFi protocols.

CupCarbon represents the main kernel of the ANR project PERSEPTEUR that aims to develop algorithms for an accurate simulation of the propagation and interference of signals in a 3D urban environment.

**Implementation**

1. Hello World

a. Code

loop

println D15

stop

b. Output



2. Addition

a. Code

loop

set x 10

set y 20

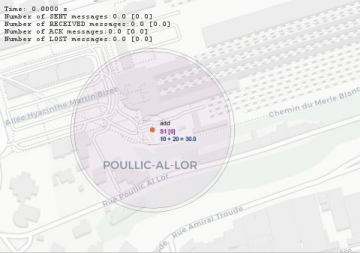
4

plus z $x $y

println $x + $y = $z

stop

b. Output



3. Subtraction

a. Code

loop

set a 20

set b 10

minus c $a $b

println $a - $b = $c

stop

b. Output



5

4. Multiplication

a. Code

set a 13

loop

for b 0 20

mult x $a $b

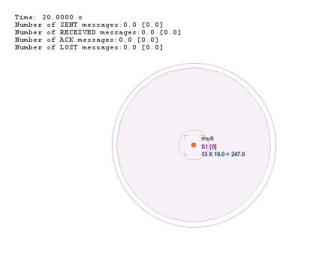
println $a X $b = $x

delay 1000

end

stop

b. Output



5. LED Blinking

a. Code

loop

for i 0 20

led 13 $i

delay 1000

end

6

b. Output



6. Mobile Routing

a. Code

i. Detection.csc

loop

dreadsensor x

if($x == 1)

send A 2

else

send B 2

delay 500

ii. Sink.csc

loop

wait

read x

if($x == A)

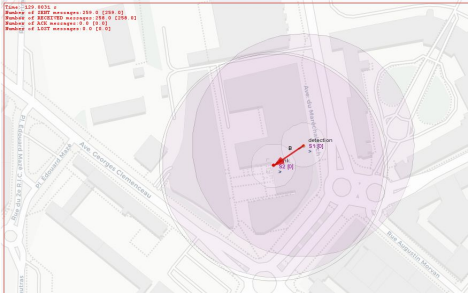
mark 1

else

mark 0

end

b. Output



**Conclusion**

Using the Cupcarbon environment, mobile routing was established between 2 sensors and basics of the SensScript used to program the sensors were understood.

**Room Occupancy Counter**

**Submitted in partial fulfillment of the requirements for the**

**Sensor Lab**

**Submitted by**

1. Aamir Ansari - 01

2. Sreekesh Iyer - 24

3. Jisha Philip - 27

4. V Krishnasubramaniam - 67

Under the Guidance of

**Prof. Smita Jangale**



**Department Of Information Technology**

**Vivekanand Education Society’s Institute of Technology**

**University Of Mumbai**

**2021-2022**

**VIVEKANAND EDUCATION SOCIETY’S INSTITUTE OF TECHNOLOGY (VESIT)**

**Chembur, Mumbai 400074**

****

**CERTIFICATE**

This is to certify that **Aamir Ansari (01), Sreekesh Iyer (24), Jisha Philip (27), V Krishnasubramaniam (67)** have satisfactorily carried out the project work entitled **Room Occupancy Counter** under the head - Wireless Networks Lab at Semester **VI** of **TE** in **Information Technology.**

**Guide Head of Dept.**

Name: **Prof. Smita Jangale** Name: **Dr. Shalu Chopra**

Signature: Signature:

**External Examiner**

**Signature**

**Date: Place: VESIT, Chembur**

**Declaration**

I declare that this written submission represents my ideas in my own words and where others’ ideas or words have been included, I have adequately cited and referenced the original source. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

----------------------------------------

(Signature)

**V Krishnasubramaniam, Roll No. 67**

------------------------------------------------

(Name of the Student and RollNo.)

# Table of Contents

| **Sr No** | **Title** | **Pg No** |
| --- | --- | --- |
| 1 | Abstract | 1 |
| 2 | Introduction | 2 |
| *2a* | Problem Statement | 2 |
| *2b* | Proposed Solution | 2 |
| 3 | Workflow of the System | 3 |
| *3a* | Input | 3 |
| *3b* | Outputs | 3 |
| 4 | Hardware Implementation | 4 |
| *4a* | Hardware Requirements | 4 |
| *4b* | Implementation | 4 |
| *4c* | Pin Configuration | 6 |
| 5 | Software Implementation | 7 |
| 6 | Simulation | 10 |
| 7 | Conclusion | 13 |
| 8 | List of References | 14 |

# Abstract

With COVID-19, rooms in public areas are set to a specific capacity to reduce spreading, and therefore we must keep track of the number of people in a room at a given time. Our project will count people coming in and out, to ensure the total number of people is less than or equal to the allotted max amount. If not, the room will be notified via alarm.

# 

# Introduction

**Problem Statement**

With the advent of the global crisis of COVID-19 there arose a need to have social distancing protocols and rules to be implemented in every shop or closed premises. The rooms in public areas are set to a specific capacity to reduce spreading, and therefore we must keep track of the number of people in a room at a given time.

**Proposed Solution**

Our project used one Arduino Uno, and read in an independent input. This is the total number of people in the room as they are entering and/or exiting it. The total number of people inside a room was counted by using two PIR/IR sensors, both sitting on one side of the entrance: one sensor was closest to the door, and the other sensor was at the same level as the first, but a bit further from the door. The main logic behind this counter is that If a person has entered the room, they would have passed first sensor one (or the sensor closest to the door), and then sensor two (or the sensor further from the door), thus incrementing the counter. If a person who was in the room had exited the room, then they would have passed first sensor two (or the sensor furthest from the door), and then sensor one (or the sensor closest to the door), thus decrementing the counter.

# 

# 

# Workflow of the System

**Input**

PIR/IR Sensors: Detected motion of a person entering or exiting the room. Using two PIR/IR sensors on the same plane, these modules were used to detect motion or obstruction, and depending on the order in which the person passed them, indicated if that person was entering or exiting the room. If the person passed sensor one (Pin A5) and then sensor two (Pin A0), it indicated they were entering the room. On the other hand, if the person passed sensor two (Pin A0) and then sensor one (Pin A5), it indicated they were exiting the room. Depending on the order, the count variable would either increment (entering) or decrement (exiting), keeping track of the total number of occupants, to ensure that the total in the room at any given time did not exceed the standards set by the room. The overall total was displayed on the LCD Display

**Outputs**

LCD Display: Displays room information and conditional messages. This module was used to notify room occupants on information about the room at any given time, that being the total number of people in the room on the top line, and the overall temperature of the room on the bottom lime. The message on the LCD would change based on the room conditions. If the total number of people exceeded the guidelines, the top line would display “Limit Exceeded!”, and the bottom line would continue displaying the number of people in the room

Passive Buzzer: Emitted an alarm. This module was used to act as an alarm. As long as the conditions of the room were not broken (the total number of people in the room at any given time was less than what was set by the guidelines), the passive buzzer would not emit any noise. However, once the room conditions were broken, the buzzer would emit a high pitch, acting as an alarm to notify room inhabitants the room conditions were broken.

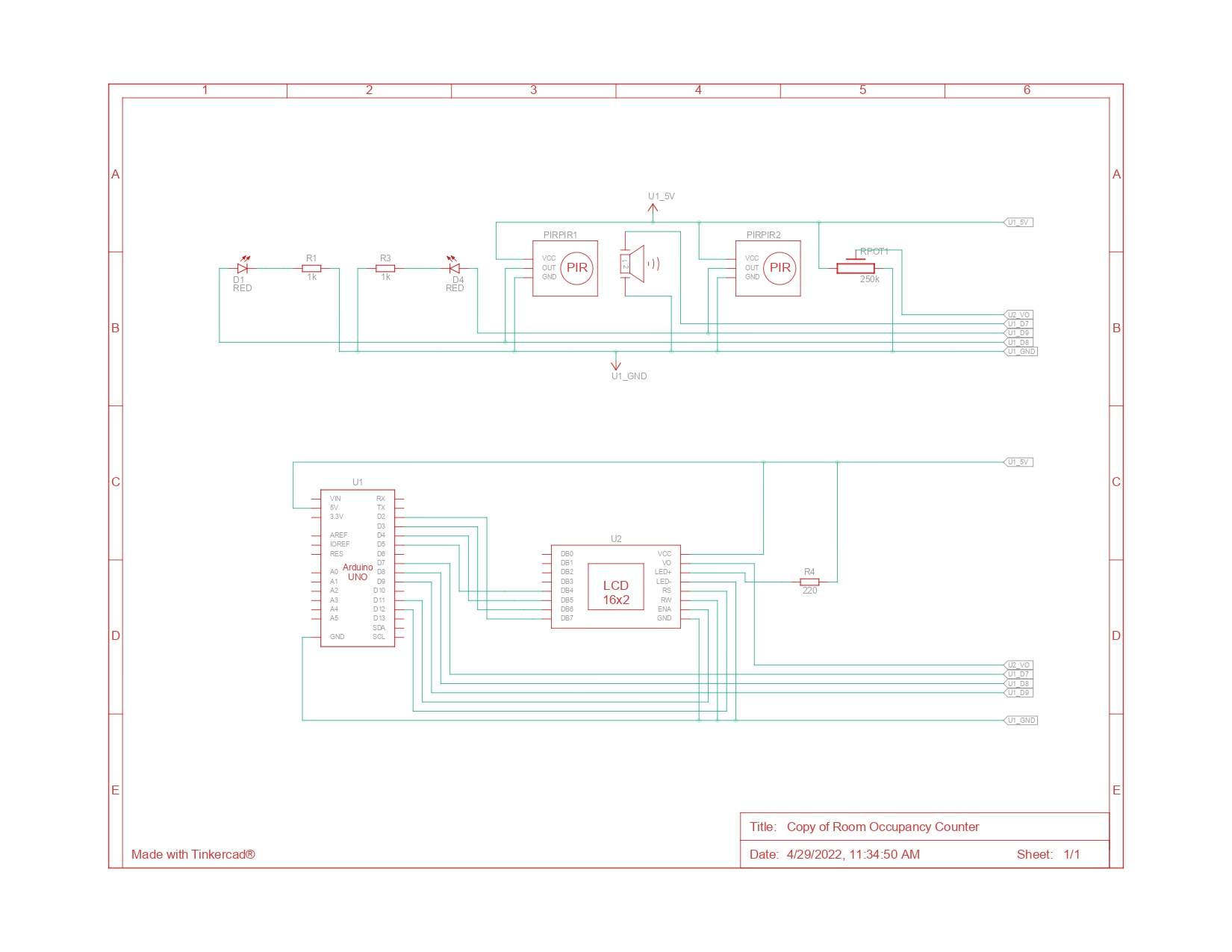
# Hardware Implementation

**Hardware Requirements:**

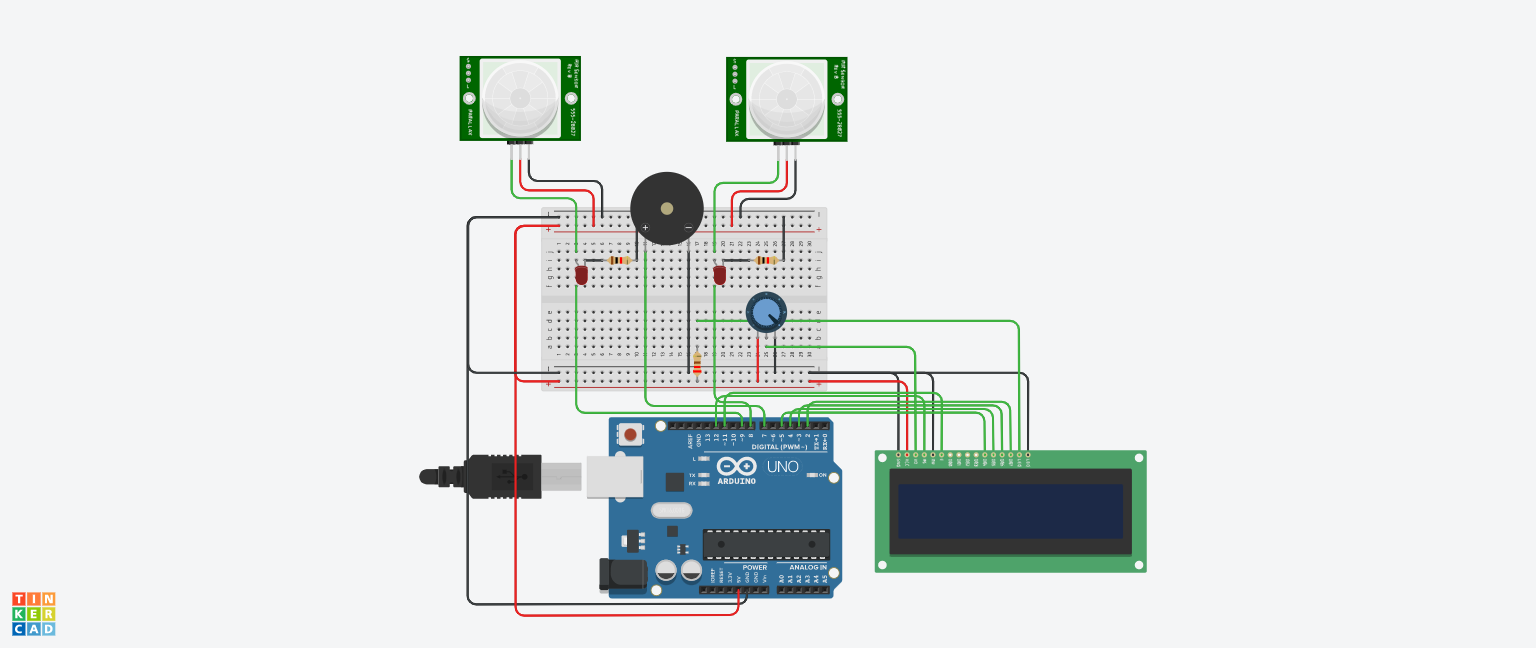
| **Name** | **Quantity** | **Component** |
| --- | --- | --- |
| U1 | 1 | Arduino Uno R3 |
| PIRpir2 | 1 | 18.780288013778318 , -237.7748847591834 , -192.04730281232418 , -202.19626480235138 PIR Sensor |
| R1, R3 | 2 | 1 kΩ Resistor |
| D1, D4 | 2 | Red LED |
| U2 | 1 | LCD 16 x 2 |
| Rpot1 | 1 | 250 kΩ Potentiometer |
| R4 | 1 | 220 Ω Resistor |
| PIEZO1 | 1 | Piezo |
| PIRpir1 | 1 | -1.64 , -237.21738692329558 , -270.77 , -189.97633844875776 PIR Sensor |

**Implementation:**

Schematic Diagram



Final Circuit Diagram



**Pin configuration:**

LCD U2 connection with Arduino Uno U1

| **LCD** | **Arduino Uno** |
| --- | --- |
| DB4 | D5 |
| DB5 | D4 |
| DB6 | D3 |
| DB7 | D2 |
| VCC | 5V |
| LED+ | 5V |
| LED- | GND |
| GND | GND |
| RW | GND |
| ENA | D11 |
| RS | D12 |

PIR1 AND PIR 2 with Arduino Uno U1

| **PIR** | **Arduino Uno** |
| --- | --- |
| VCC | 5V |
| OUT | D8 |
| GND | GND |

# Software Implementation

Code (Arduino Uno R3):

#include <LiquidCrystal.h>

LiquidCrystal lcd(12,11,5,4,3,2);

const unsigned int NOTE\_C4 = 262;

const unsigned int NOTE\_E4 = 330;

const unsigned int NOTE\_G4 = 392;

const int PIR1\_PIN = 8;

const int PIR2\_PIN = 9;

const int PIEZO\_PIN = 7;

const int BUTTON\_PIN = 13;

const bool SERIAL\_PRINT = true;

int visitors = 0;

int lastRIPdetected = 0;

bool b\_PIR1\_active = false;

bool b\_PIR2\_active = false;

void setup() {

lcd.begin(16,2);

pinMode(PIR1\_PIN, INPUT);

pinMode(PIR2\_PIN, INPUT);

pinMode(PIEZO\_PIN, OUTPUT);

pinMode(BUTTON\_PIN, INPUT);

Serial.begin(9600);

DisplayMsg("Visitor counter", "Welcome", SERIAL\_PRINT);

}

// -------- Message display on LCD and Serial (optional) ------

void DisplayMsg(String s1, String s2, bool ab\_serial\_print) {

lcd.clear();

lcd.setCursor(0, 0);

lcd.print(s1);

lcd.setCursor(0, 1);

lcd.print(s2);

if (ab\_serial\_print)

Serial.println(s1 + ". " + s2);

}

// -------- Increase/Decrease visitors ---------

void UpdateVisitorsCounter(int x){

int duration=100;

visitors = visitors + x;

lastRIPdetected = 0; // reset detected PIR

if (x>0)

PlayNote(NOTE\_G4, duration);

else

PlayNote(NOTE\_C4, duration);

}

// -------- Play note ---------

void PlayNote(int note, int duration) {

tone(PIEZO\_PIN, note, duration);

delay(duration \* 1.3);

noTone(PIEZO\_PIN);

}

void loop() {

// ----------- check PIR1 ----------------

if (digitalRead(PIR1\_PIN) == HIGH) {

if ( !b\_PIR1\_active ) {

b\_PIR1\_active = true;

if (lastRIPdetected == 0 && !b\_PIR2\_active) { // new start

lastRIPdetected = 1;

DisplayMsg("In PIR1", "Visit started", SERIAL\_PRINT);

} else if (lastRIPdetected == 2) { // if we were in PIR2 before

UpdateVisitorsCounter(-1);

DisplayMsg("Visitor exited", "Visitors:" + String(visitors), SERIAL\_PRINT);

}

}

}else

b\_PIR1\_active = false ; // reenable PIR1

// ----------- check PIR2 ----------------

if (digitalRead(PIR2\_PIN) == HIGH ) {

if ( !b\_PIR2\_active ) {

b\_PIR2\_active = true;

if (lastRIPdetected == 0 && !b\_PIR1\_active) { // new start

if (visitors > 0) {

lastRIPdetected = 2;

DisplayMsg("In PIR2", "Exit started", SERIAL\_PRINT);

} else

DisplayMsg("No more visitors", "to exit", SERIAL\_PRINT);

} else if (lastRIPdetected == 1) { // if we were in PIR1 before

UpdateVisitorsCounter(1);

DisplayMsg("Visitor entered", "Visitors:" + String(visitors), SERIAL\_PRINT);

}

}

} else

b\_PIR2\_active = false; // reenable PIR2

// ----------- Check Limit condition ----------------

if(visitors>3){

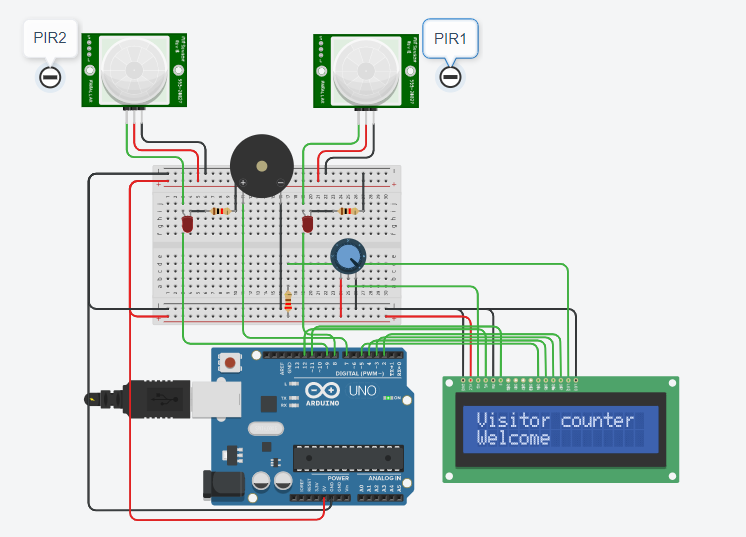
PlayNote(NOTE\_G4, 50);

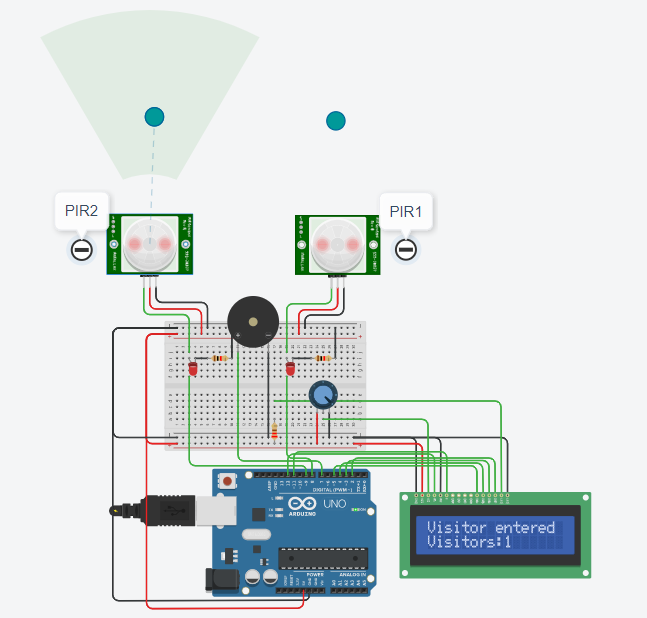
DisplayMsg("ALERT", "Limit Exceeded!", SERIAL\_PRINT);

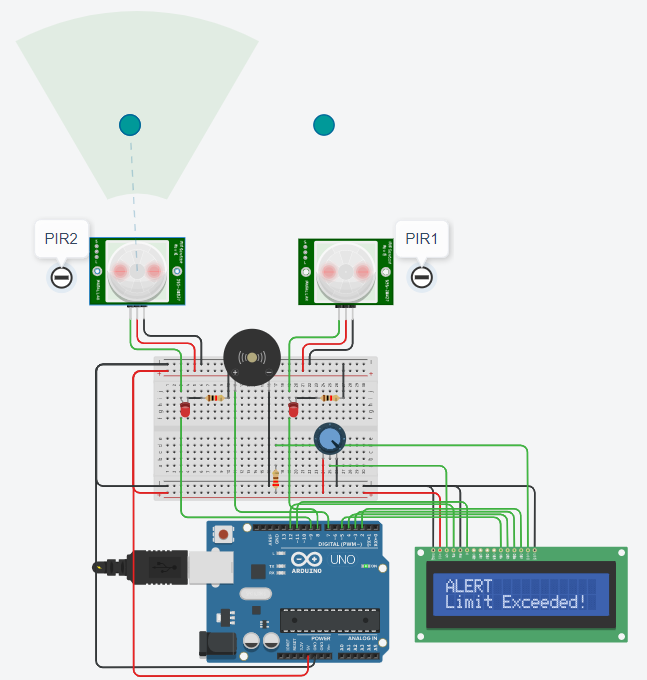
}

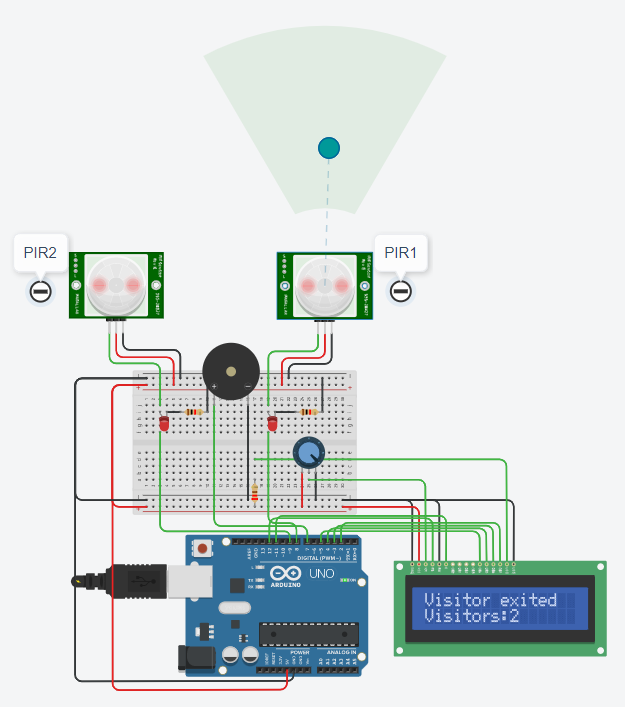
}

# Simulation









# Conclusion

The idea behind making this device is to practice discipline in maintaining the guidelines of social distancing especially after experiencing how things have been during this pandemic, where public places, like stores or closed gatherings, require the presence of people in order to function. This device can help monitor the number of people in that enclosed space at any given time just to ensure that people are in a safe environment.

The guidelines are mostly set the same for everybody, having the uniform number of people in a room no matter its size. That is why we have set a maximum number of people in the room in the code and not accepting it from the user. The purpose behind this is that if the user is given the choice of setting the maximum capacity, they may get tempted to set an arbitrary limit that may never be reached and the purpose of the device is lost.

**List of References**

[1] ElectroPeak. PIR Motion Sensor: How to Use Pirs w/ Arduino & Raspberry Pi. 8 Apr. 2019,

create.arduino.cc/projecthub/electropeak/pir-motion-sensor-how-to-use-pirs-w-arduino-ra spberry-pi-18d7fa.

[2] Khondaker, Mahamudul Karim. Temperature Sensor with Arduino Uno. 17 June 2020, created. arduino.cc/projecthub/m\_karim02/temperature-sensor-with-arduino-uno-9806a3.

[3] Team, The Arduino. Play a Melody Using The Tone() Function. 5 Feb. 2018, www.arduino.cc/en/Tutorial/BuiltInExamples/toneMelody.

[4] Team, The Arduino. Scrolldisplayleft() and Scrolldisplayright() Methods. 5 Feb. 2018, www. arduino.cc/en/Tutorial/LibraryExamples/LiquidCrystalScroll.