CS983: Embedded, Cyber Physical Systems and IoT Security

GROUP 7

Assignment: Build Your Smart Home

Jordan is a wealthy business tycoon who lives a life of indulgence and doesn't shy away from an extravagant display of his financial prowess. This time he has set his eyes on augmenting his home with automation to improve his lifestyle, uplift his societal status, and feed his laziness. More specifically, Jordan wants to automate his door, fans, and lights. He contacted "Home Corp.", a renowned smart-home startup, and finalised the lucrative deal. You are responsible for leading the project, and this is your time to shine and showcase your technical expertise.
You need an ultrasonic distance sensor installed on the door to detect if someone is at the door and a servo motor to control the door movement. The house interior will have light sensors to see illumination and lighting to toggle the light switch appropriately. We need to turn on the lights when Lux < 400 in the room. A temperature and humidity sensor will detect how hot and humid the room is and regulate the fan speed according to the table below. Please use this simulator to plan and execute your project.

Temperature	Humidity	Fan speed
25 and below		0
25-29	40-60	2
25-29	61-80	3
25-29	81-100	4
30-34	40-60	3
30-34	61-80	4
30-34	81-100	5
35-39	40-60	4
35-39	61-100	5
40 and above	40-100	5

Equipments Required from the Simulator:

1. Temperature sensor and Humidity sensor
2. Servo motor
3. IR proximity sensor
4. Light sensor
5. Arduino board
6. Stepper motor
7. LCD Display

Expected functionality from your smart home simulation:

1. Temperature and humidity(DHT22) sensor has sliders to adjust the readings. The fan speed(a stepper motor) should change with the change in temperature and humidity values according to the rules defined in the table above.

2. Turn ON the lights (LED) based on the illumination in the room. You can use LDR to measure illumination.

3. Use an ultrasonic distance sensor to detect the presence of a person at the doorway. When the measured distance is less than or equal to 200 centimetres, rotate the servo motor by 90-degree, simulating the motion of the door opening. After a certain period of time, initiate the closure of the simulated door by returning the servo motor to its original position.

4. Once the simulation has started, it should run in a loop. Reflecting any changes to the sensor values automatically by modifying the Servo motor position, LED state (ON, OFF) and display temperature and humidity values on LCD without restarting the simulation. Refresh the readings on the LCD in every iteration. You can use some delay time inside the loop so that displayed values don’t change too quickly

Group Members

Name	Roll No	email
Pritesh Krishna Kadam	23157049	pkkadam23@iitk.ac.in
Chaudhari Harshal Pandurang	23157014	harshalp23@iitk.ac.in
Jaiesh Pandey	23157019	Jaieshnp23@iitk.ac.in
Ojasvi Ashish Chauhan	23157041	achauhan23@iitk.ac.in
Pradeep Kumar C	23157045	pradeepkc23@iitk.ac.in

Contribution from each member

Project Area	Contributor	Percentage
Arduino Mega 2560	Jaiesh Pandey	100%
DHT22 Sensor + Bipolar stepper motor	Pritesh Krishna Kadam	100%
LCD2004 Display	Ojasvi Ashish Chauhan	100%
Photoresistor (LDR) sensor + LED(5mm)	Chaudhari Harshal Pandurang	100%
HC-SR04 Ultrasonic Distance Sensor + Servo motor	Pradeep Kumar C	100%

Overview

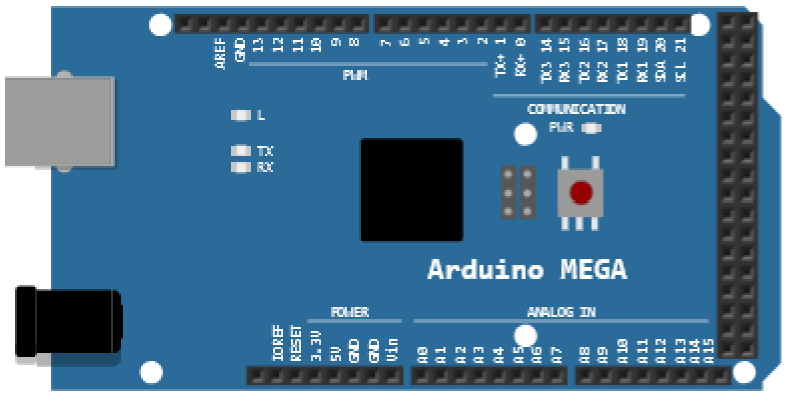
We are designing this project using the Wowki online electronics simulator.
Based on the sensors/actuator required for the project, we divided them among each of the group members and explored them separately using Wowki documentation. Below, we have mentioned the details about the project area explored by each member and the sample codes that we referred to for its implementation . The final code is the integrated version of these separately explored areas.

Arduino Mega 2560

Overview

The Arduino Mega 2560 is a microcontroller board based on the ATmega2560. It has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; based on our requirements of the pins required to support Temperature sensor and Humidity sensor , servo motor , ultrasonic sensor, light sensor, stepper motor and LCD display,
The Arduino Mega 2560 is simulated using the [AVR8js Library]() that covers the libraries which are required for the solution.

Circuit Schematics



Pins 0 to 53 are digital GPIO pins. Pins A0 to A15 double as analog input pins, in addition to being digital GPIO pins.
There are five ground pins: GND.1 (next to pin 13), GND.2/GND.3 (next to the Vin pin), and GND.4/GND.5 (at the bottom of the dual-row female header connector)
Pins VIN / 5V are connected to the positive power supply. There are also two additional power supply pins, 5V.1/5V.2, at the top of the dual-row female header connector
Pins 3.3V / IOREF / AREF / RESET are not available in the simulation.
Digital pins 2 … 13, 44, 45, and 46 have hardware PWM support (total of 15 PWM channels).
The board includes four LEDs:

L	Connected to digital pin 13
RX	Serial RX Activity

TX	Serial TX Activity
ON	Power LED. Always on while the simulation is running

Sample code

Following code controls the on-board "L" LED.

```
void setup() {
  // Initialize digital pin LED_BUILTIN (13) as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```

DHT22 Sensor + Bipolar stepper motor

DHT22 Overview

DHT22 is a Digital Humidity and Temperature sensor. Its temperature measuring range is from -40 to +125 degrees Celsius with +-0.5 degrees accuracy. Its humidity measuring range is from 0 to 100% with 2-5% accuracy

Working Principle

It consists of a humidity sensing component, a NTC temperature sensor (or thermistor), and an IC on the back side of the sensor.

The humidity sensing component has two electrodes with moisture holding substrates between them. So as the humidity changes, the conductivity of the substrate changes, or the resistance between these electrodes changes. This change in resistance is measured and processed by the IC which makes it ready to be read by a microcontroller.

NTC(Negative Temperature Coefficient) temperature sensor or a thermistor is actually a variable resistor that changes its resistance with change of the temperature. These sensors are made by sintering of semiconductive materials such as ceramics or polymers in order to provide larger changes in the resistance with just small changes in temperature.

Circuit Schematics

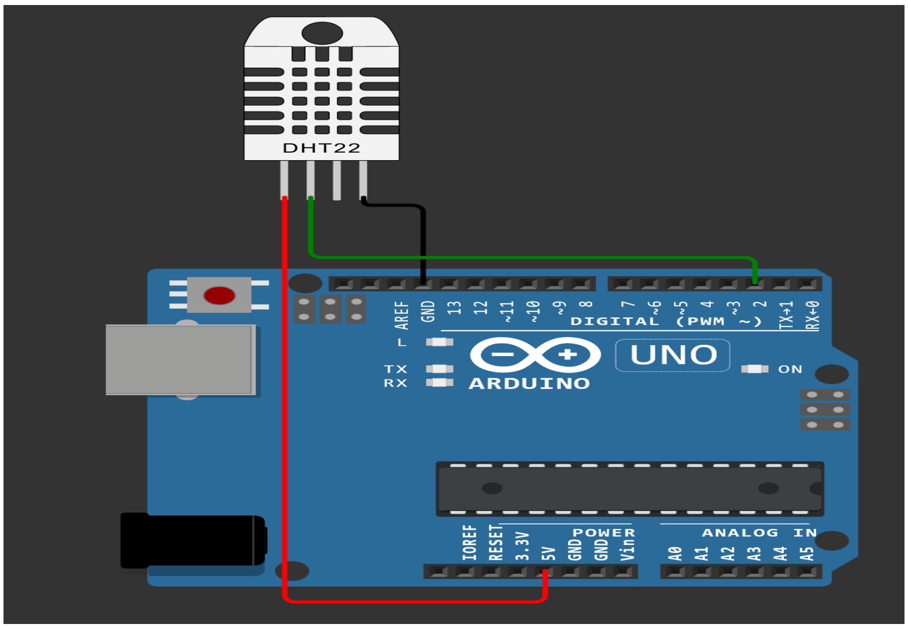
It has four pins

- VCC : Positive Voltage
- SDA : Digital data pin (input/output)
- NC : Not connected
- GND : Ground

Controlling the temperature

You can change the temperature and humidity values while the simulation is running. Click on the DHT22 sensor and a small popup window will open. Use the temperature and humidity sliders to change the value

Sample Simulation



Sample code

```
#include "DHT.h"

#define DHTPIN 2     // Digital pin connected to the DHT sensor

DHT dht(DHTPIN, DHT22);

void setup() {
  dht.begin();
}

void loop() {
  // Wait a few seconds between measurements.
  delay(2000);

  // Reading temperature or humidity takes about 250 milliseconds!
  // Sensor readings may also be up to 2 seconds
  float h = dht.readHumidity();
  // Read temperature as Celsius (the default)
  float t = dht.readTemperature();
}
```

Bipolar stepper motor

Overview

A stepper motor is a unique type of brushless DC motor which position can be precisely controlled even without any feedback

Working principle

The working principle of a stepper motor is based on magnetic fields. It has two main components, a stator and a rotor. The rotor is usually a permanent magnet and it’s surrounded by some coils on the stator. When we energise or let current flow through the coils, particular magnetic fields are generated in the stator that either attract or repel the rotor. By activating the coils, step by step, one after another in a particular order, we can achieve continued motion of the rotor, but also, we can make it stop at any position.

Circuit Schematics

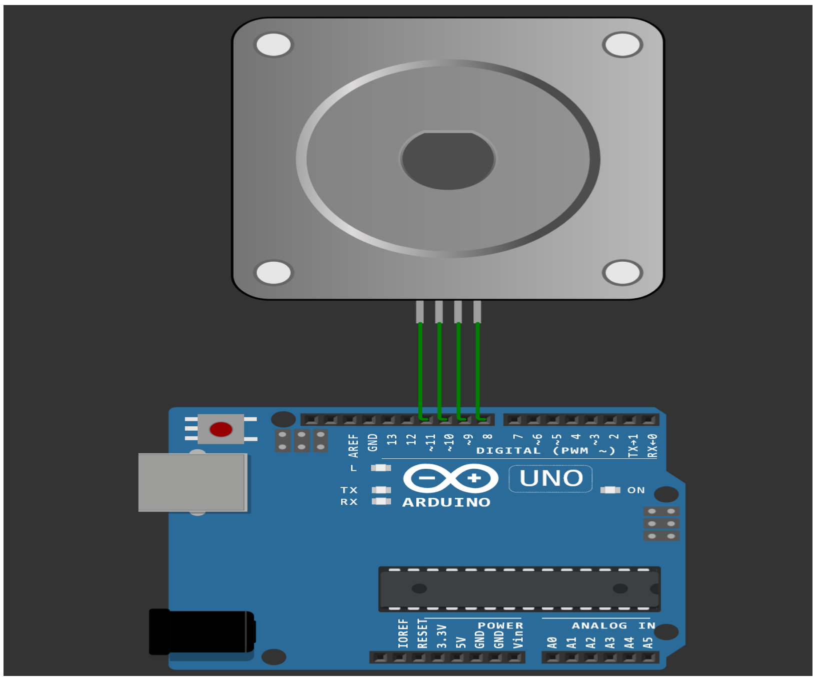
It has 4 pins :

- A-Coil : A negative signal
- A+Coil : A positive signal
- B+Coil : B positive signal
- B-Coil : B negative signal

Simulation Behaviour

The stepper motor moves 1.8 degrees per step (200 steps per revolution). The motor also supports half-stepping (0.9 degrees per step / 400 steps per revolution). You can even use smaller microsteps (e.g. 1/4 or 1/8 step), but the simulated motor only displays the angle in half-step resolution.

Sample simulation



Sample code for above simulation:

```
#include <Stepper.h>
// initialise the stepper library on pins 8 through 11:
Stepper myStepper(200, 8, 9, 10, 11);
void setup() {
  // set the speed at 60 rpm:
  myStepper.setSpeed(60);
}
void loop() {
  // step one revolution  in one direction:
  myStepper.step(stepsPerRevolution);
  delay(500);
 // step one revolution in the other direction:
  myStepper.step(-stepsPerRevolution);
  delay(500);
}
```

LCD2004 Display

Overview & Working:

The LCD2004, is a display module designed to show letters, numbers, and symbols using a grid of tiny dots. Think of it as a screen made up of small building blocks that can form different characters and information.

This display consists of rows and columns of dots, and each dot can light up to create a character. The "2004" in its name indicates that it can display 20 characters in each of its 4 lines. So, it's like having four lines of text, with each line capable of showing up to 20 characters.

Hello World!
VSS VDD V0 RS RW E D0 D1 D2 D3 D4 D5 D6 D7 A K
1
16

The LCD2004 works like a clever blend of science and technology. It has a special substance inside called liquid crystals that can change their behaviour when an electric current runs through them. These liquid crystals are organised in a grid of tiny dots on the screen.
When we send controlled electric signals to specific points on this grid, the liquid crystals respond by adjusting their alignment. This adjustment affects how light passes through them, which creates characters, numbers, and symbols that we can read on the screen. To make these characters visible, there's a light source behind the liquid crystals, sort of like a flashlight.

In essence, the LCD2004 uses the interaction between liquid crystals and electricity to craft the messages and information we see on its display, making it a smart way to show data in various devices and projects.

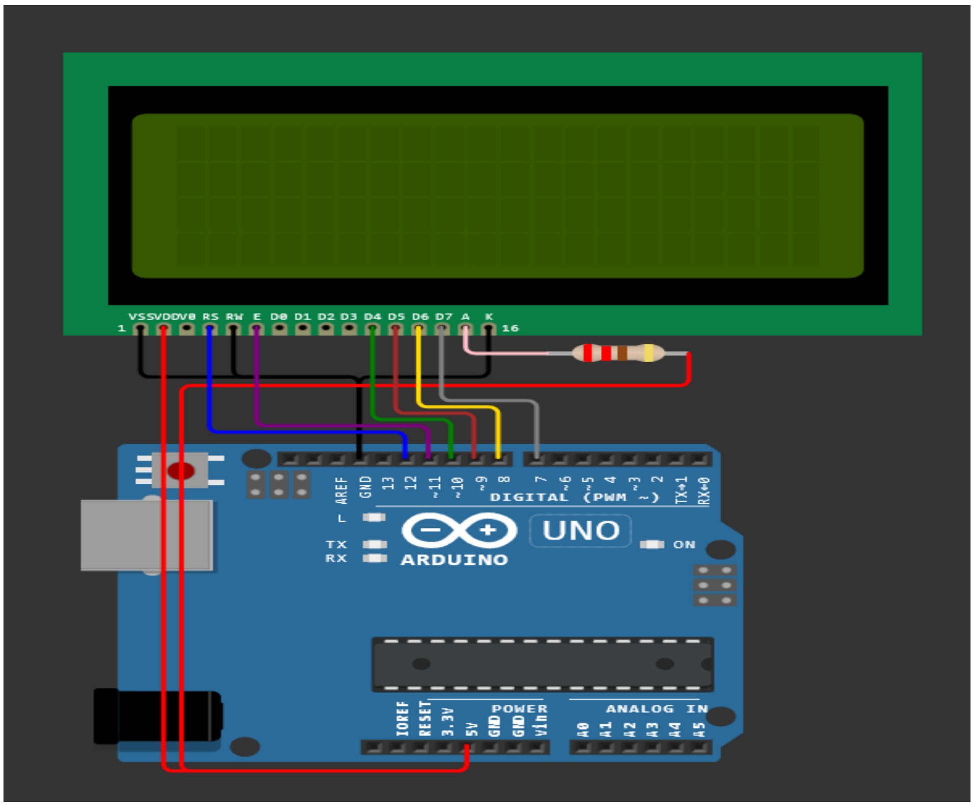
Circuit Schematics

Here's a general overview of how you connect an LCD2004 to an Arduino:

- VSS (GND): Connect this pin to the ground (GND) of the Arduino.
- VDD (VCC): Connect this pin to the 5V power supply of the Arduino.
- V0 (Contrast): Connect this pin to a variable resistor or a potentiometer to control the contrast of the display.
- RS (Register Select): Connect this pin to a digital pin on the Arduino, usually used to select between command and data mode.

- RW (Read/Write): Connect this pin to GND to set the LCD in write mode.
- E (Enable): Connect this pin to a digital pin on the Arduino, used to enable or disable the data sent to the LCD.
- D0 to D7 (Data Lines): Connect these pins to digital pins on the Arduino for sending data to the LCD.
- A (Anode) and K (Cathode): These pins are for the backlight. Connect A to a positive voltage source and K to GND to enable the backlight.

Simulation



Sample Code :

```
#include <LiquidCrystal.h>

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

void setup() {
  lcd.createChar(1, pacman);
  lcd.createChar(2, dot);
  lcd.begin(20, 4);
  lcd.setCursor(3, 0);
  lcd.print("wokwi-lcd2004");
}
```

Photoresistor (LDR) sensor + LED(5mm)

Overview

A photoresistor (LDR) is a light-sensitive resistor that changes its resistance based on the intensity of light it's exposed to. This simple and cost-effective sensor finds applications in various fields where light detection or control is needed.

LDR belongs to the passive resistive sensor category which means it doesn't require an external power source to operate. Instead, its resistance value changes based on the amount of light it detects.

The LED automatically applies gamma correction. This means that even a very short burst of current will result in some visible light, similar to how physical LEDs work, so you get more accurate simulation in the following cases:

Working principle

Photoresistor (LDR)

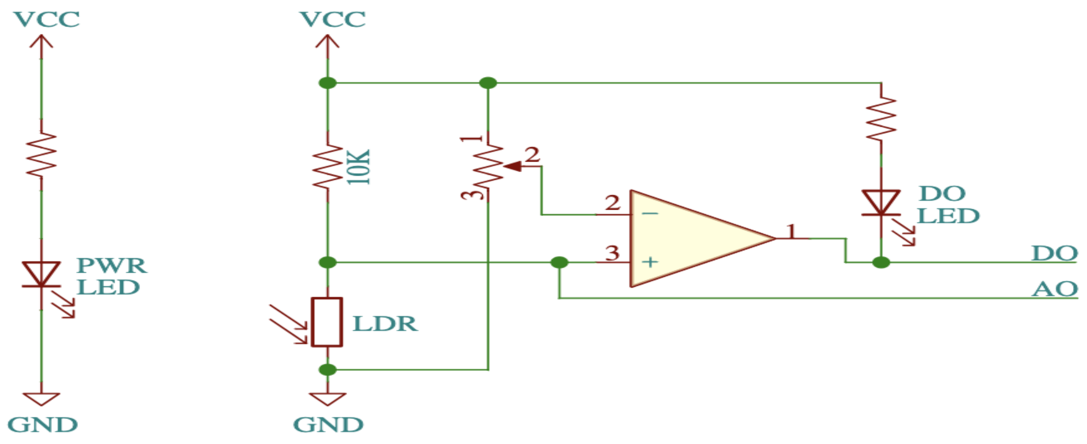
- Photoresistors are typically made from a semiconductor material. These materials have a property called the photoelectric effect, which means that when photons (light particles) strike the material's surface, they can release electrons from the atoms within the material.
- When light of a specific wavelength strikes the surface of the semiconductor material, the photons from the light are absorbed by the atoms in the material. This absorption of photons creates electron-hole pairs. Electrons are excited to higher energy levels, leaving behind positively charged holes in the material's atomic structure.
- The presence of these electron-hole pairs affects the conductivity of the semiconductor material. In the absence of light, the material has a higher resistance because there are fewer electron-hole pairs to contribute to the flow of electric current. This higher resistance results in less electrical current passing through the material.
- When the photoresistor is exposed to light, the creation of electron-hole pairs reduces the resistance of the material. As a result, the resistance of the photoresistor decreases when it's exposed to more intense light.
- The photoresistor is typically connected in a circuit with a voltage source. As the resistance of the photoresistor changes due to varying light levels, the current passing through the circuit changes accordingly. This change in current can be measured and used to determine the intensity of light falling on the photoresistor.
- A photoresistor (LDR) can be used to trigger actions based on light conditions, such as turning on lights, adjusting camera settings, or activating security alarms.

- Photoresistors have a certain response time, meaning it takes a short while for their resistance to stabilize after a change in light intensity. Additionally, their response can be affected by factors such as the material's composition, temperature, and the wavelength of the incident light.

LED:

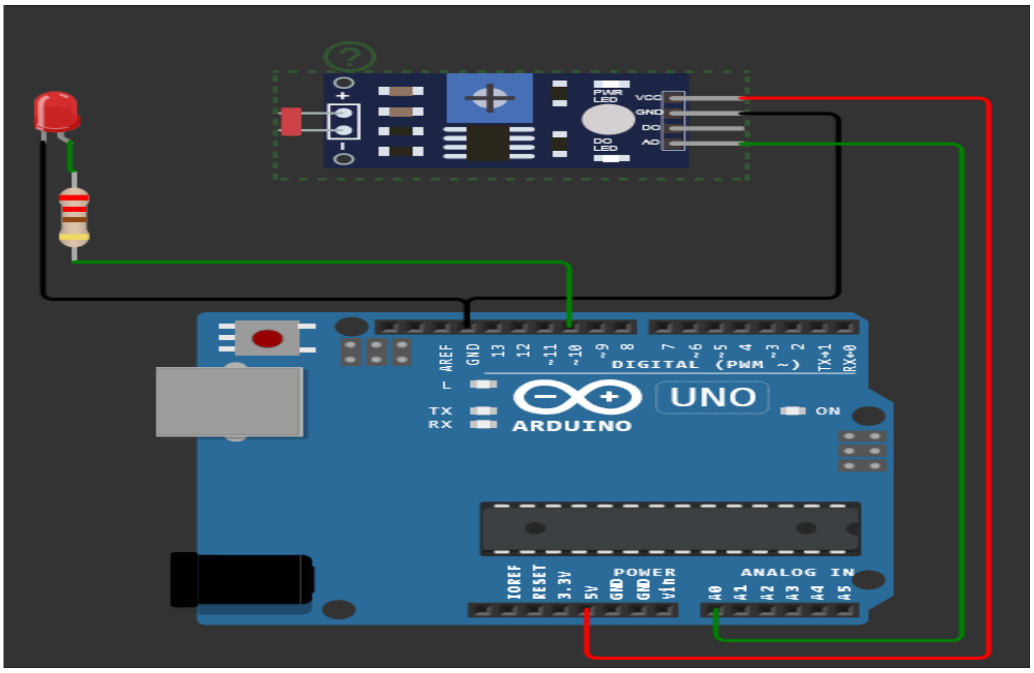
- An LED consists of a P-N junction, where the P-type material (rich in positive charge carriers or "holes") is connected to the anode (positive terminal), and the N-type material (rich in negative charge carriers or electrons) is connected to the cathode (negative terminal).
- The photon released during electron-hole recombination has a specific energy corresponding to its wavelength, which determines the color of the emitted light. Different semiconductor materials produce different colors of light when they emit photons. The intensity of light emitted by an LED can be controlled by adjusting the current flowing through it.

Circuit Schematics



- VCC: The power supply voltage. This is the voltage that the circuit runs on.
- LDR: The light dependent resistor. This resistor changes its resistance depending on the amount of light it is exposed to.
- LED: The light emitting diode. This diode emits light when current flows through it.
- R: The current limiting resistor. This resistor limits the current that flows through the LED, preventing it from burning out.
- GND: The ground connection. This is the common point for all the voltages in the circuit.

Simulation



Sample code

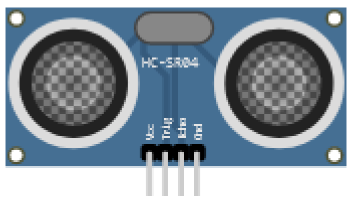
```
#define LED_PIN 10
// LDR Characteristics
const float GAMMA = 0.7;
const float RL10 = 50;

void setup() {
  pinMode(LED_PIN, OUTPUT);
}

void loop() {
  int analogValue = analogRead(A0);
  float voltage = analogValue / 1024. * 5;
  float resistance = 2000 * voltage / (1 - voltage / 5);
  float lux = pow(RL10 * 1e3 * pow(10, GAMMA) / resistance, (1 / GAMMA));

  if(lux < 200)  {
    digitalWrite(LED_PIN, HIGH);
  } else{
    digitalWrite(LED_PIN, LOW);
  }
}
```

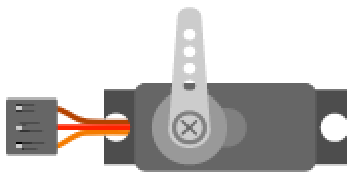
Ultrasonic Distance Sensor + Servo motor

Overview

As we learned, while IR sensors are more appropriate for short-range applications and perform well with reflective surfaces, ultrasonic sensors are better suited for longer distance measurements and operate well with a variety of materials. The ultrasonic sensor would be the right choice for the application on which we are working here, and to match the requirement the available sensor was HC-SR04. The HC-SR04 is an affordable and easy to use distance measuring sensor that has a range of 2 cm to 400cm (about an inch to 13 feet). The sensor is composed of two ultrasonic transducers. One is a transmitter that outputs ultrasonic sound pulses, and the other is a receiver that listens for reflected waves. Although soft materials like cloth can be challenging to detect acoustically, it is important that sunlight or dark materials do not affect the operation.

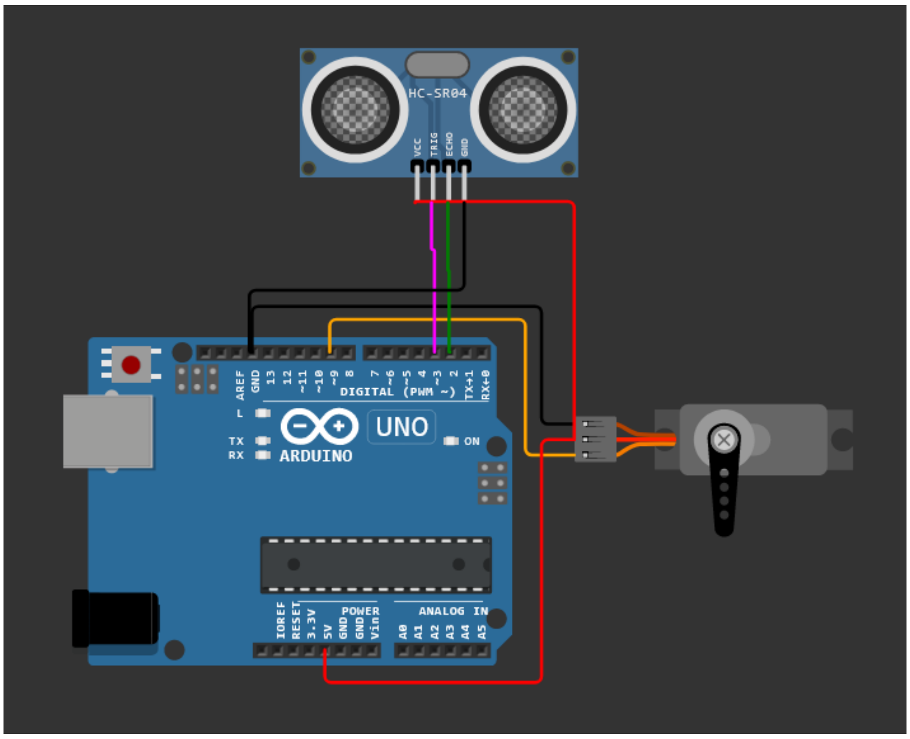
The sensor has 4 pins. VCC and GND go to 5V and GND pins on the Arduino, and the Trig and Echo go to any digital Arduino pin. Using the Trig pin we send the ultrasound wave from the transmitter, and with the Echo pin we listen for the reflected signal.





As we know, a servo motor is a closed-loop system that uses position feedback to control its motion and final position. There are many types of servo motors, and their main feature is the ability to precisely control the position of their shaft.
We have used a simple standard Micro Servo Motor(3 pin Servo control signal, voltage and RND) to close and open the door when the ultrasonic sensor detects the person is present.
A servo motor is controlled by sending a series of pulses through the signal line. The frequency of the control signal should be 50Hz or a pulse should occur every 20ms. The width of pulse determines angular position of the servo and these type of servos can usually rotate 180 degrees

Simulation

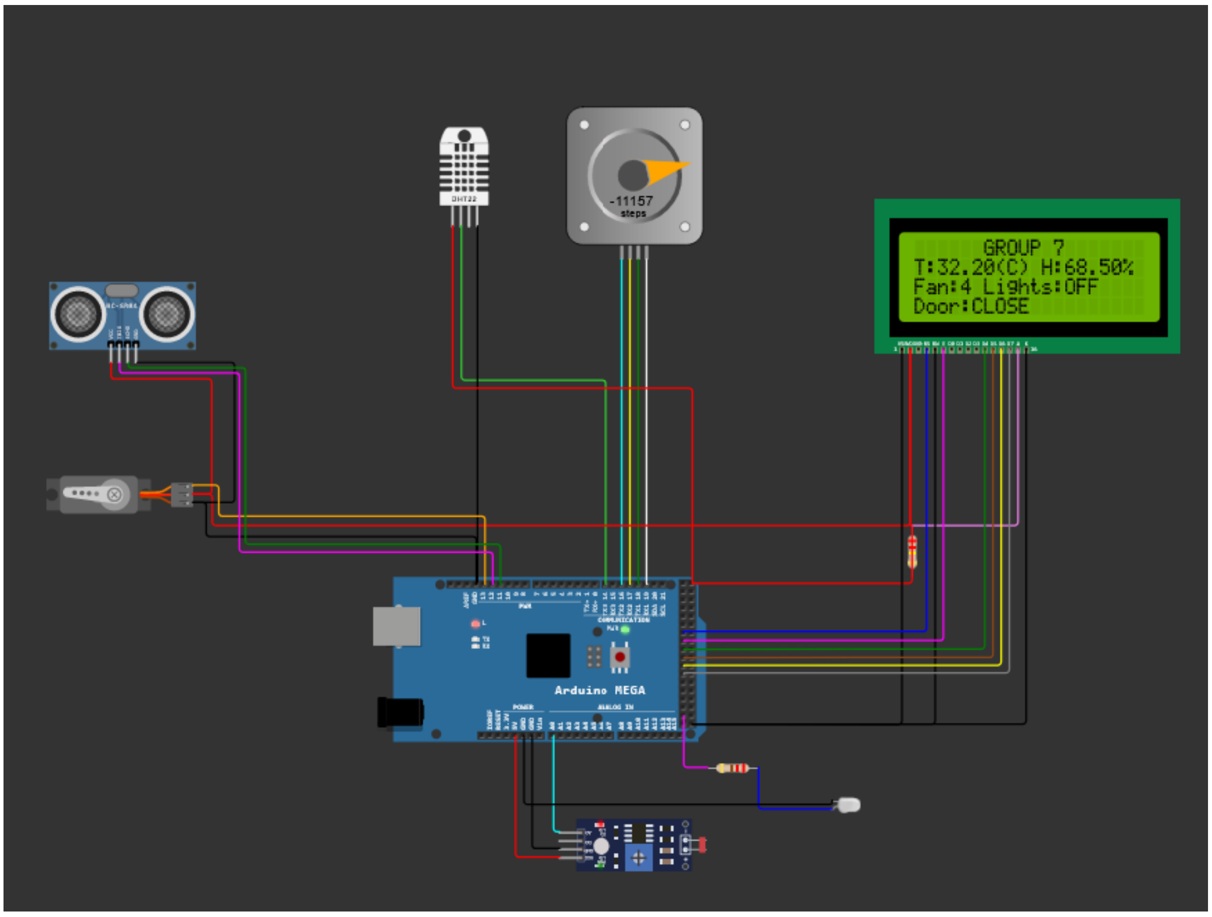


Sample code

```
#include <Servo.h>

#define ECHO_PIN 2
#define TRIG_PIN 3

Servo myservo;  // create servo object to control a servo
// twelve servo objects can be created on most boards

int pos = 0;    // variable to store the servo position

float readDistanceCM() {
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);
  int duration = pulseIn(ECHO_PIN, HIGH);
  return duration * 0.034 / 2;
}
```

```
void setup() {
  myservo.attach(9);  // attaches the servo on pin 9 to the servo object
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
}

void loop() {
  float distance = readDistanceCM();

  bool isNearby = distance < 100;
  if(isNearby)
  {
    for ( ;pos <= 180; pos += 1) { // goes from 0 degrees to 180 degrees
      // in steps of 1 degree
      myservo.write(pos);              // tell servo to go to position in
variable 'pos'
      delay(15);                       // waits 15ms for the servo to reach
the position
    }
  }else
  {
    for ( ;pos >= 0; pos -= 1) { // goes from 180 degrees to 0 degrees
      myservo.write(pos);              // tell servo to go to position in
variable 'pos'
      delay(15);                       // waits 15ms for the servo to reach
the position
    }
  }
}
```

Final Simulation



```
GROUP 7
T:32.20(C) H:68.50%
Fan:4 Lights:OFF
Door:CLOSE
```

Final Code

```
#include <Servo.h>
#include <dht.h>
#include <Stepper.h>
#include <LiquidCrystal.h>

#define PIN_TRIG 12
#define PIN_ECHO 11
#define DHT22_PIN 14
#define LED_PIN 52
#define SERVO_PIN 10
```

```
// LCD for display
LiquidCrystal lcd(32, 34, 36, 38, 40, 42);

// DHT for temp & Humidity sensor
dht DHT;

// Stepper motor for Fan.
Stepper myStepper(2000, 16, 17, 18, 19);

// Servo for Door control
Servo myservo;
int servo_position = 90;

// Global variable for display
bool g_doorOpen = false;
bool g_lightOn = false;
int g_fanSpeed = 0;
float g_temp = 0;
float g_humid = 0;

void setup()
{
  // Attach the servo pin to the servo object
  myservo.attach(SERVO_PIN);

  // Set input/output pins for Ultrasonic Distance Sensor
  pinMode(PIN_TRIG, OUTPUT);
  pinMode(PIN_ECHO, INPUT);
  pinMode(LED_PIN, OUTPUT);

  myStepper.setSpeed(200);

  lcd.begin(20, 4);
  lcd.setCursor(6,0);
  lcd.print("GROUP 7");
  lcd.setCursor(3,1) ; //sets cursor to second line first row
  lcd.print("IOT Assignment");
  delay(1000);
}

/// Shows parameters on LCD.
void refershDisplay()
{
    lcd.setCursor(0,1) ;
    lcd.print("T:");
    lcd.print(g_temp);
    lcd.print("(C) H:");
    lcd.print(g_humid);
    lcd.print("% ");
    lcd.setCursor(0,2);
    lcd.print("Fan:");
    lcd.print(g_fanSpeed);
    lcd.print(" Lights:");
    lcd.print(g_lightOn?"ON ":"OFF");
    lcd.setCursor(0,3);
    lcd.print("Door:");
    lcd.print(g_doorOpen?"OPEN ":"CLOSE");

}

//===================================================================
// Door Movement

// Detects is someone is at the door(<200cm distance) using Ultrasonic
Distance Sensor
// return true is distance from sensor < 200 cm.
//else return false.
```

```
bool IsSomeoneAtDoor()
{
  long duration = 0;
  int distance = 0;

  digitalWrite(PIN_TRIG, LOW);
  delayMicroseconds(2);

  digitalWrite(PIN_TRIG, HIGH);
  delayMicroseconds(10);
  digitalWrite(PIN_TRIG, LOW);

  // Reads the sound wave travel time in microseconds
  duration = pulseIn(PIN_ECHO, HIGH);
  // Calculating the distance
  distance = (duration * 0.034 / 2);

  return (distance < 200);
}



// If someone is at door then open the door
void controlDoor(){

  if(IsSomeoneAtDoor())
  {
    // Open the door
    g_doorOpen = true;
    refershDisplay();

    if(servo_position > 0)
    {
      servo_position -= 5;
      myservo.write(servo_position);
    }
  }
  else
  {
    // Close the door
    g_doorOpen = false;
    if(servo_position < 90){
      servo_position += 5;
      myservo.write(servo_position);
    }
  }
}
// - Door Movement
//====================================================================

//====================================================================
// Light On/Off

// Detect illumination(lux) using the light sensor.
//if lux < 400 then turn on the lights(LED)
void controlLights()
{
  const float GAMMA = 0.7;
  const float RL10 = 50;

  // Convert the analog value into lux value:
  int analogValue = analogRead(A0);
  float voltage = analogValue / 1024. * 5;
  float resistance = 2000 * voltage / (1 - voltage / 5);
  float lux = pow(RL10 * 1e3 * pow(10, GAMMA) / resistance, (1 / GAMMA));

  if(lux<400)
  {
    g_lightOn = true;
    digitalWrite(LED_PIN, HIGH);   // turn on the lights
  }
  else
  {
    g_lightOn = false;
    digitalWrite(LED_PIN, LOW);    // turn off the lights
  }
  refershDisplay();
}

// - Light On/Off
//===================================================================

//===================================================================
// Fan Speed control
/*
Adjust fan speed according to following parameters.
Temperature   Humidity  Fan speed
25 and below          0
25-29       40-60       2
25-29       61-80       3
25-29       81-100      4
30-34       40-60       3
30-34       61-80       4
30-34       81-100      5
35-39       40-60       4
35-39       61-100      5
40 and above  40-100      5

Note we have added following fan speed for
humidity less than 40  :
Temperature   Fan speed
25-29           1
30-34           2
```

```
35-39           3
40 & above      4
*/
void controlFan()
{
  uint32_t start = micros();
  int chk = DHT.read22(DHT22_PIN);
  uint32_t stop = micros();

  g_temp = DHT.temperature;
  g_humid = DHT.humidity;

  if(g_temp<=25)
  {
    myStepper.step(0);
    g_fanSpeed = 0;
  }
  else if(g_temp > 25 and g_temp <= 29)
  {
    if(g_humid >= 40 and g_humid <= 60){
      myStepper.step(-10);
      g_fanSpeed = 2;
    }
    else if(g_humid > 60 and g_humid <= 80)
    {
      myStepper.step(-50);
      g_fanSpeed = 3;
    }
    else if(g_humid > 80 and g_humid <= 100){
      myStepper.step(-100);
      g_fanSpeed = 4;
    }
    else{
      myStepper.step(-5);
      g_fanSpeed = 1;
    }
  }
  else if(g_temp > 29 and g_temp <= 34)
  {
    if(g_humid >= 40 and g_humid <= 60)
    {
      myStepper.step(-50);
      g_fanSpeed = 3;
    }
    else if(g_humid > 60 and g_humid <= 80)
    {
      myStepper.step(-100);
      g_fanSpeed = 4;
    }
    else if(g_humid > 80 and g_humid <= 100)
    {
      myStepper.step(-150);
      g_fanSpeed = 5;
    }
    else{
      myStepper.step(-10);
      g_fanSpeed = 2;
    }
  }
  else if(g_temp > 34 and g_temp <= 39)
  {
    if(g_humid >= 40 and g_humid <= 60)
    {
      myStepper.step(-100);
      g_fanSpeed = 4;
    }
    else if(g_humid > 60 and g_humid <= 100)
    {
      myStepper.step(-150);
      g_fanSpeed = 5;
    }
    else
    {
      myStepper.step(-50);
      g_fanSpeed = 3;
    }
  }
  else if(g_temp > 39)
  {
    if(g_humid >= 40 and g_humid <= 100)
    {
      myStepper.step(-150);
      g_fanSpeed = 5;
    }
    else
    {
      myStepper.step(-100);
      g_fanSpeed = 4;
    }
  }
}

// - Fan speed
//===================================================================



void loop() {
  controlDoor();
  controlLights();
  controlFan();
  refershDisplay();
}
```

Simulation Link :

https://wokwi.com/projects/373212386958823425

References:

https://wokwi.com/
https://howtomechatronics.com/