IOT HOLIDAY ASSESSMENT 2211CS020430

1. **Weather Reporting System**:

#include <Adafruit\_Sensor.h>

#include <DHT.h>

#include <DHT\_U.h>

// Define DHT sensor pin and type

#define DHTPIN 2      // Pin where the DATA pin is connected

#define DHTTYPE DHT22 // DHT22 sensor type

// Initialize DHT sensor

DHT dht(2, DHT22);

void setup() {

**Serial**.begin(9600);

**Serial**.println("Weather Report System");

  // Initialize the DHT sensor

  dht.begin();

**Serial**.println("DHT22 sensor initialized");

}

void loop() {

  // Read temperature and humidity from DHT22

  float temperature = dht.readTemperature();

  float humidity = dht.readHumidity();

  // Check if readings are valid

  if (isnan(temperature) || isnan(humidity)) {

**Serial**.println("Failed to read from DHT22 sensor!");

  } else {

**Serial**.print("Temperature: ");

**Serial**.print(temperature);

**Serial**.println("°C");

**Serial**.print("Humidity: ");

**Serial**.print(humidity);

**Serial**.println("%");

  }

  // Simulate pressure and altitude data (as BMP180 is unavailable)

  float pressure = 1013.25; // Sea level standard atmospheric pressure in hPa

  float altitude = 50.0;    // Simulated altitude in meters

**Serial**.print("Pressure: ");

**Serial**.print(pressure);

**Serial**.println(" hPa");

**Serial**.print("Altitude: ");

**Serial**.print(altitude);

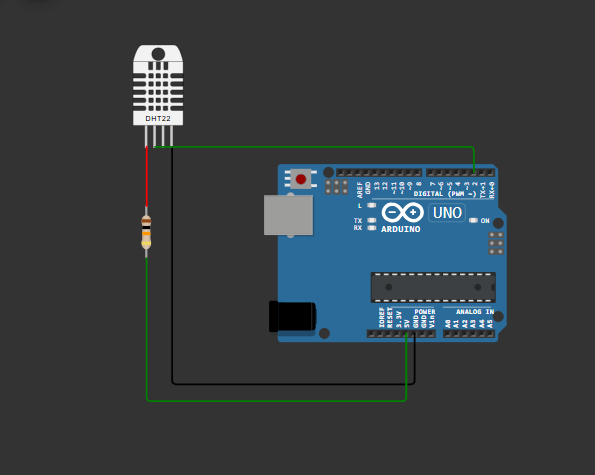
**Serial**.println(" m");

**Serial**.println("-------------------------");

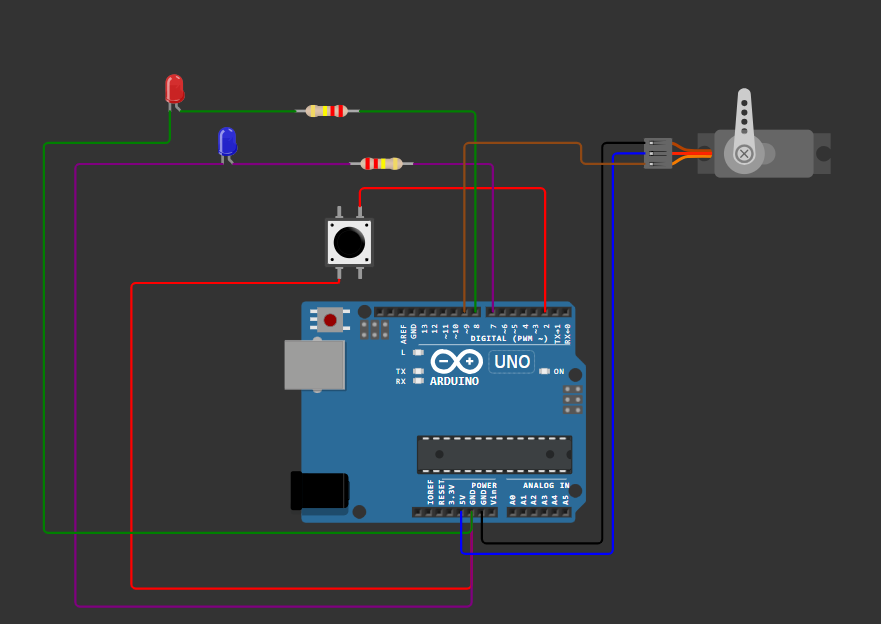
  // Delay before the next reading

  delay(2000);

}



1. **Home Automation System**:
2. #include <Servo.h>
3. #define LIGHT1\_PIN 7  // Pin for Light 1 (LED 1)
4. #define LIGHT2\_PIN 8  // Pin for Light 2 (LED 2)
5. #define BUTTON\_PIN 2  // Pin for the push button
6. #define FAN\_SERVO\_PIN 9  // Pin for the servo motor (Fan)
7. Servo fanServo;          // Servo object for fan simulation
8. bool fanRunning = false; // State of the fan (false = OFF, true = ON)
9. int currentAngle = 90;   // Current angle of the servo
10. int step = 1;            // Step size for continuous movement
11. void setup() {
12. // Pin modes for LEDs
13. pinMode(7, OUTPUT);
14. pinMode(8, OUTPUT);
15. // Pin mode for button with internal pull-up
16. pinMode(2, INPUT\_PULLUP);
17. // Attach the servo and set initial position
18. fanServo.attach(9);
19. fanServo.write(currentAngle); // Start fan at 90° (OFF position)
20. // Turn off LEDs initially
21. digitalWrite(7, LOW);
22. digitalWrite(8, LOW);
23. }
24. void loop() {
25. static bool buttonPressed = false;
26. // Check if the button is pressed and toggle the fan state
27. if (digitalRead(2) == LOW && !buttonPressed) {
28. fanRunning = !fanRunning; // Toggle fan and lights state
29. buttonPressed = true;
30. // Toggle lights
31. if (fanRunning) {
32. digitalWrite(7, HIGH); // Turn Light 1 ON
33. digitalWrite(8, HIGH); // Turn Light 2 ON
34. } else {
35. digitalWrite(LIGHT1\_PIN, LOW);  // Turn Light 1 OFF
36. digitalWrite(LIGHT2\_PIN, LOW);  // Turn Light 2 OFF
37. fanServo.write(90);            // Reset fan to 90° (OFF position)
38. }
39. delay(200); // Debounce delay
40. } else if (digitalRead(BUTTON\_PIN) == HIGH) {
41. buttonPressed = false;
42. }
43. // Move the servo continuously if the fan is ON
44. if (fanRunning) {
45. currentAngle += step;
46. // Reverse direction when reaching bounds
47. if (currentAngle >= 180 || currentAngle <= 90) {
48. step = -step;
49. }
50. fanServo.write(currentAngle);
51. delay(10); // Delay for smooth movement
52. }
53. }



**Air Pollution Monitoring System**:

#include <Wire.h>

#include <LiquidCrystal\_I2C.h>

// Define Pin Assignments

#define AIR\_SENSOR\_PIN A0  // Analog pin for Air Quality Sensor (use potentiometer for simulation)

#define BUZZER\_PIN 8  // Pin for the Buzzer

#define LIGHT\_PIN 9  // Pin for Light (LED)

// LCD I2C Setup (use address 0x27, but try 0x3F if not working)

LiquidCrystal\_I2C lcd(0x27, 16, 2);  // Initialize LCD with I2C address 0x27 and 16 columns, 2 rows

// Thresholds for air quality levels

#define GOOD\_AIR\_QUALITY 700

#define POOR\_AIR\_QUALITY 300

void setup() {

  // Start Serial Communication

**Serial**.begin(9600);

  // Initialize Buzzer and Light pins

  pinMode(8, OUTPUT);

  pinMode(9, OUTPUT);

  // Initialize the LCD

  lcd.begin(16, 2); // Initialize LCD with 16 columns, 2 rows

  delay(1000);  // Wait for 1 second for the LCD to initialize properly

  lcd.backlight();  // Turn on the LCD backlight

  lcd.setCursor(0, 0);  // Set cursor to the first column of the first row

  lcd.print("Air Quality Monitor");  // Display the title

  delay(2000);  // Wait for 2 seconds

  // Test if LCD is working by printing a test message

  lcd.clear();

  lcd.setCursor(0, 0);

  lcd.print("Hello World");

  delay(2000);  // Wait for 2 seconds

}

void loop() {

  // Read the air quality sensor value (simulated by potentiometer)

  int airSensorValue = analogRead(A0);

  // Map the sensor value to a percentage (0-100% for display)

  float airQualityPercentage = map(airSensorValue, 0, 1023, 0, 100);

  // Display the air quality on the LCD

  lcd.clear();  // Clear the screen

  lcd.setCursor(0, 0);  // Set cursor to the first column of the first row

  lcd.print("Air Quality: ");

  lcd.print(airQualityPercentage);

  lcd.print("%");

  // Buzzer and Light activation based on air quality

  if (airSensorValue > 700) {

    digitalWrite(8, LOW);  // Turn off Buzzer

    digitalWrite(9, HIGH);  // Turn on Light (Good air quality)

  } else if (airSensorValue < 300) {

    digitalWrite(8, HIGH);  // Turn on Buzzer

    digitalWrite(9, LOW);   // Turn off Light (Poor air quality)

  } else {

    digitalWrite(8, LOW);  // Turn off Buzzer

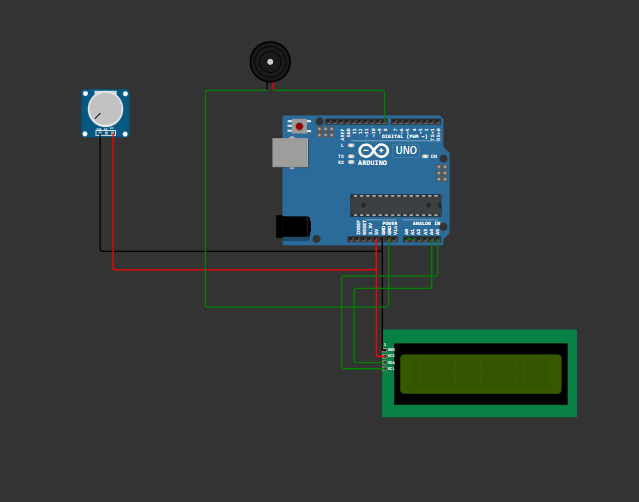
    digitalWrite(9, LOW);   // Turn off Light (Moderate air quality)

  }

  // Delay for a short time

  delay(500);

}



**Smart Irrigation System**:

#include "RTClib.h"

#include "DHT.h"

#define DHTPIN 8

#define DHTTYPE DHT22

DHT dht(DHTPIN, DHTTYPE);

#include <LiquidCrystal\_I2C.h>

#define I2C\_ADDR    0x27

#define LCD\_COLUMNS 20

#define LCD\_LINES   4

LiquidCrystal\_I2C lcd(I2C\_ADDR, LCD\_COLUMNS, LCD\_LINES);

String data;

int relay1=3;

int relay2=4;

int relay3=5;

int relay4=6;

RTC\_DS1307 rtc;

char daysOfTheWeek[7][12] = {"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"};

void setup()

   { {

**Serial**.begin(115200);

**Serial**.println(F("DHT22 example!"));

  dht.begin();

    }

{

**Serial**.begin(115200);

   lcd.init();

  lcd.backlight();

  lcd.setCursor(3,0);

  lcd.print("welcome to");

  lcd.setCursor(2,1);

  lcd.print("SMART FARMING");

  delay(4000);

  pinMode(relay1, OUTPUT);

  pinMode(relay2, OUTPUT);

  pinMode(relay3, OUTPUT);

  pinMode(relay4, OUTPUT);

**Serial**.println("welcome to my project");

  delay(500);

  if (! rtc.begin()) {

**Serial**.println("Couldn't find RTC");

**Serial**.flush();

    abort();

  }

  lcd.clear();

}

   }

void loop () {

   {

  float temperature = dht.readTemperature();

  float humidity = dht.readHumidity();

  // Check if any reads failed and exit early (to try again).

  if (isnan(temperature) || isnan(humidity)) {

**Serial**.println(F("Failed to read from DHT sensor!"));

    return;

  }

**Serial**.print(F("Humidity: "));

**Serial**.print(humidity);

**Serial**.print(F("%  Temperature: "));

**Serial**.print(temperature);

**Serial**.println(F("°C "));

  lcd.setCursor(0,3);

    lcd.print("temp:");

    lcd.println(temperature);

    lcd.setCursor(10,3);

    lcd.print("hum:");

    lcd.println(humidity);

  delay(2000);

}

  DateTime now = rtc.now();

**Serial**.print("Current time: ");

**Serial**.print(now.year(), DEC);

**Serial**.print('/');

**Serial**.print(now.month(), DEC);

**Serial**.print('/');

**Serial**.print(now.day(), DEC);

**Serial**.print(" (");

**Serial**.print(daysOfTheWeek[now.dayOfTheWeek()]);

**Serial**.print(") ");

**Serial**.print(now.hour(), DEC);

**Serial**.print(':');

**Serial**.print(now.minute(), DEC);

**Serial**.print(':');

**Serial**.print(now.second(), DEC);

**Serial**.println();

**Serial**.println();

  delay(3000);

  lcd.setCursor(3,0);

  lcd.print("Time:");

  lcd.print(now.hour(), DEC);

  lcd.print(':');

  lcd.print(now.minute(), DEC);

  lcd.print(':');

  lcd.print(now.second(), DEC);

  if((now.second()> 1) && (now.second()<15))

  {

  lcd.setCursor(0,1);

  lcd.print("Relay1:ON ");

**Serial**.println("relay1 is on");

   digitalWrite(relay1, HIGH);

  }

  else{

    lcd.setCursor(0,1);

    lcd.print("Relay1:Off");

    digitalWrite(relay1,LOW);

  }

   if((now.second()> 20) && (now.second()<30))

  {

  lcd.setCursor(10,1);

  lcd.print("Relay2:ON ");

**Serial**.println("relay2 is on");

   digitalWrite(relay2, HIGH);

  }

  else{

   lcd.setCursor(10,1);

  lcd.print("Relay2:OFF");

  digitalWrite(relay2,LOW);

  }

  if((now.second()> 35) && (now.second()<45))

  {

   lcd.setCursor(0,2);

  lcd.print("Relay3:ON ");

**Serial**.println("relay3 is on");

   digitalWrite(relay3, HIGH);

  }

  else{

    lcd.setCursor(0,2);

  lcd.print("Relay3:OFF");

  digitalWrite(relay3,LOW);

  }

   if((now.second()> 50) && (now.second()<59))

  {

    lcd.setCursor(10,2);

  lcd.print("Relay4:ON ");

**Serial**.println("relay4 is on");

   digitalWrite(relay4, HIGH);

  }

  else{

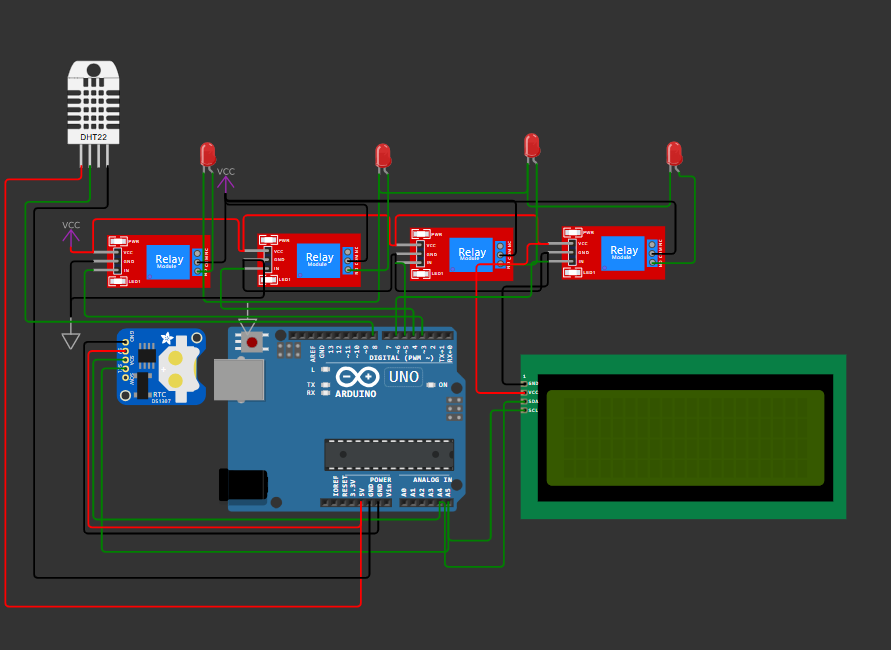
    lcd.setCursor(10,2);

  lcd.print("Relay4:OFF");

  digitalWrite(relay4,LOW);

  }

}



**Smart Alarm Clock**:

/\* ----- C Program for Arduino based Alarm Clock ---- \*/

#include <Wire.h>

#include<EEPROM.h>

#include <RTClib.h>

#include <LiquidCrystal.h>

const int rs = 8;

const int en = 9;

const int d4 = 10;

const int d5 = 11; //DISPLAY

const int d6 = 12;

const int d7 = 13;

LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

RTC\_DS1307 RTC;

int temp,inc,hours1,minut,add=11;

int next=7;

int INC=6;

int set\_mad=5;

#define buzzer 3

int HOUR,MINUT,SECOND;

void setup()

{

**Wire**.begin();

 RTC.begin();

 lcd.begin(16,2);

 pinMode(INC, INPUT);

 pinMode(next, INPUT);

 pinMode(set\_mad, INPUT);

 pinMode(buzzer, OUTPUT);

 digitalWrite(next, HIGH);

 digitalWrite(set\_mad, HIGH);

 digitalWrite(INC, HIGH);

   lcd.setCursor(0,0);

   lcd.print("Real Time Clock");

   lcd.setCursor(0,1);

   lcd.print("Circuit Digest ");

    delay(2000);

 if(!RTC.isrunning())

 {

 RTC.adjust(DateTime(\_\_DATE\_\_,\_\_TIME\_\_));

 }

}

void loop()

{

   int temp=0,val=1,temp4;

   DateTime now = RTC.now();

   if(digitalRead(set\_mad) == 0)      //set Alarm time

   {

     lcd.setCursor(0,0);

     lcd.print("  Set Alarm  ");

    delay(2000);

    defualt();

    time();

    delay(1000);

    lcd.clear();

    lcd.setCursor(0,0);

    lcd.print("  Alarm time ");

    lcd.setCursor(0,1);

    lcd.print(" has been set  ");

    delay(2000);

 }

 lcd.clear();

 lcd.setCursor(0,0);

 lcd.print("Time:");

 lcd.setCursor(6,0);

 lcd.print(HOUR=now.hour(),DEC);

 lcd.print(":");

 lcd.print(MINUT=now.minute(),DEC);

 lcd.print(":");

 lcd.print(SECOND=now.second(),DEC);

 lcd.setCursor(0,1);

 lcd.print("Date: ");

 lcd.print(now.day(),DEC);

 lcd.print("/");

 lcd.print(now.month(),DEC);

 lcd.print("/");

 lcd.print(now.year(),DEC);

 match();

 delay(200);

}

void defualt()

{

  lcd.setCursor(0,1);

  lcd.print(HOUR);

  lcd.print(":");

  lcd.print(MINUT);

  lcd.print(":");

  lcd.print(SECOND);

}

/\*Function to set alarm time and feed time into Internal eeprom\*/

void time()

{

  int temp=1,minuts=0,hours=0,seconds=0;

    while(temp==1)

    {

     if(digitalRead(INC)==0)

     {

      HOUR++;

      if(HOUR==24)

      {

       HOUR=0;

      }

      while(digitalRead(INC)==0);

     }

     lcd.clear();

      lcd.setCursor(0,0);

    lcd.print("Set Alarm Time ");

   //lcd.print(x);

    lcd.setCursor(0,1);

    lcd.print(HOUR);

    lcd.print(":");

    lcd.print(MINUT);

    lcd.print(":");

    lcd.print(SECOND);

    delay(100);

    if(digitalRead(next)==0)

    {

      hours1=HOUR;

**EEPROM**.write(add++,hours1);

     temp=2;

     while(digitalRead(next)==0);

    }

    }

    while(temp==2)

    {

     if(digitalRead(INC)==0)

     {

      MINUT++;

      if(MINUT==60)

      {MINUT=0;}

      while(digitalRead(INC)==0);

     }

     // lcd.clear();

    lcd.setCursor(0,1);

    lcd.print(HOUR);

    lcd.print(":");

    lcd.print(MINUT);

    lcd.print(":");

    lcd.print(SECOND);

    delay(100);

      if(digitalRead(next)==0)

      {

       minut=MINUT;

**EEPROM**.write(add++, minut);

       temp=0;

       while(digitalRead(next)==0);

      }

    }

    delay(1000);

}

/\* Function to chack medication time \*/

void match()

{

  int tem[17];

  for(int i=11;i<17;i++)

  {

    tem[i]=**EEPROM**.read(i);

  }

  if(HOUR == tem[11] && MINUT == tem[12])

  {

   beep();

   beep();

   beep();

   beep();

   lcd.clear();

   lcd.print("Wake Up........");

   lcd.setCursor(0,1);

   lcd.print("Wake Up.......");

   beep();

   beep();

   beep();

   beep();

  }

}

/\* function to buzzer indication \*/

void beep()

{

   digitalWrite(buzzer,HIGH);

   delay(500);

   digitalWrite(buzzer, LOW);

   delay(500);

}

