**SMART PUBLIC RESTROOM**

**FINAL PART**

Designing a final module for a smart public restroom involves integrating various features that prioritize hygiene, efficiency, and user convenience. Here are components and features that could be part of this module:

**Occupancy Monitoring:**

* Sensors to detect occupancy and display availability.
* Green (vacant) or red (occupied) indicators outside the restroom.

**Automated Entry/Exit:**

* Motion sensors or touchless systems for door opening/closing.
* RFID or smartphone-based access for authorized users.

**Hygiene Maintenance:**

* Automatic flush for toilets and urinals.
* Touchless faucets and soap dispensers.
* Automated sanitizer or disinfectant dispensers for surfaces.
* Self-cleaning mechanisms for toilet seats and floors.

**Air Quality Control:**

* Air fresheners or purifiers to maintain good air quality.
* Motion-activated exhaust fans for efficient odor control.

**Maintenance Alerts:**

* Sensors to monitor soap, toilet paper, and paper towel levels, sending alerts for refill.
* Maintenance alerts for cleaning or repairs needed.

**Energy Efficiency:**

* LED lighting with motion sensors for energy conservation.
* Water-saving fixtures to minimize usage.

**User Feedback Mechanism:**

* Digital feedback systems for users to rate cleanliness or report issues.

**Universal Accessibility:**

* Design for users with disabilities, including grab bars, wider stalls, and lower sinks.
* Voice-activated controls for those with mobility issues.

**Data Analytics and Remote Monitoring:**

* Sensors to collect data on usage patterns, foot traffic, and supply needs.
* Remote monitoring for efficient maintenance and management.

**Security and Safety:**

* CCTV cameras for security.
* Emergency call buttons or systems.

**Adaptability and Modularity:**

* Modular design allowing easy integration or replacement of components.
* Upgradable systems to incorporate future technologies.

**Privacy Measures:**

* Soundproofing or white noise features to enhance privacy within the restroom.

**Sustainability:**

* Integration of sustainable materials and designs in construction.

**COVID-19 Adaptations**:

* Touchless temperature scanners or health check kiosks at the entrance.
* UV sanitization mechanisms for high-touch surfaces.

**Integration with Mobile Apps**:

* Integration with mobile apps for real-time restroom status and navigation.

**Community Engagement:**

* Interactive displays for public health awareness or local community messages.

**Aesthetics and User Experience:**

* Pleasant aesthetics and design for a comfortable user experience.

**Maintenance and Cleaning Schedule Optimization:**

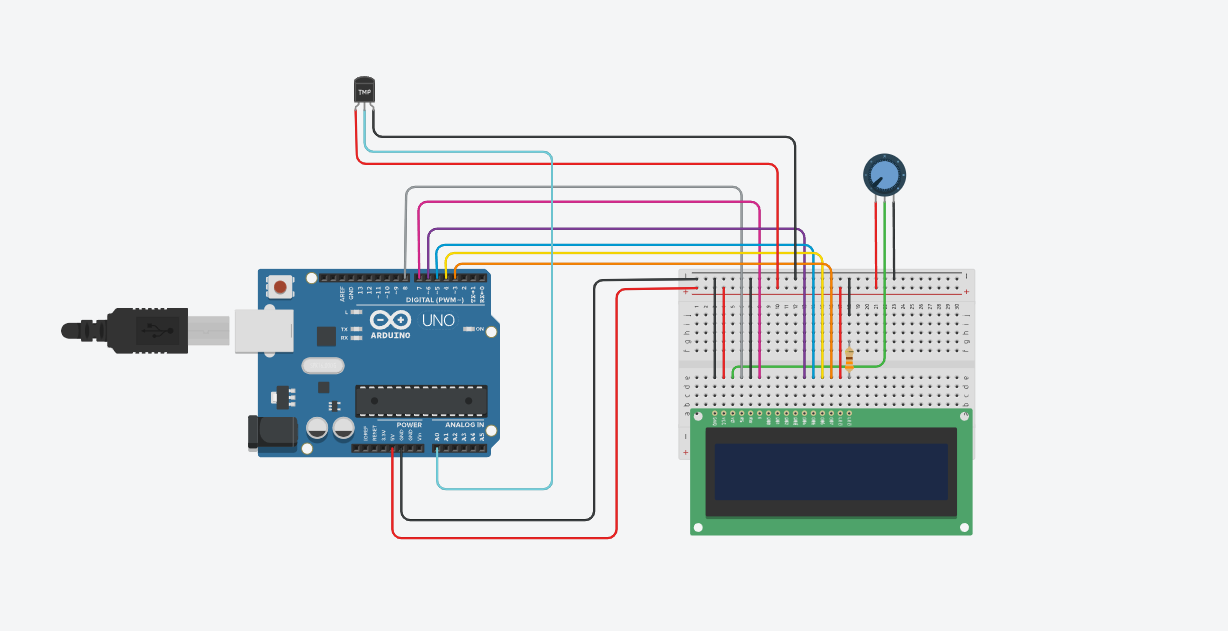
* AI-driven systems to predict usage peaks for cleaning and maintenance scheduling.

**Emergency Response Protocols:**

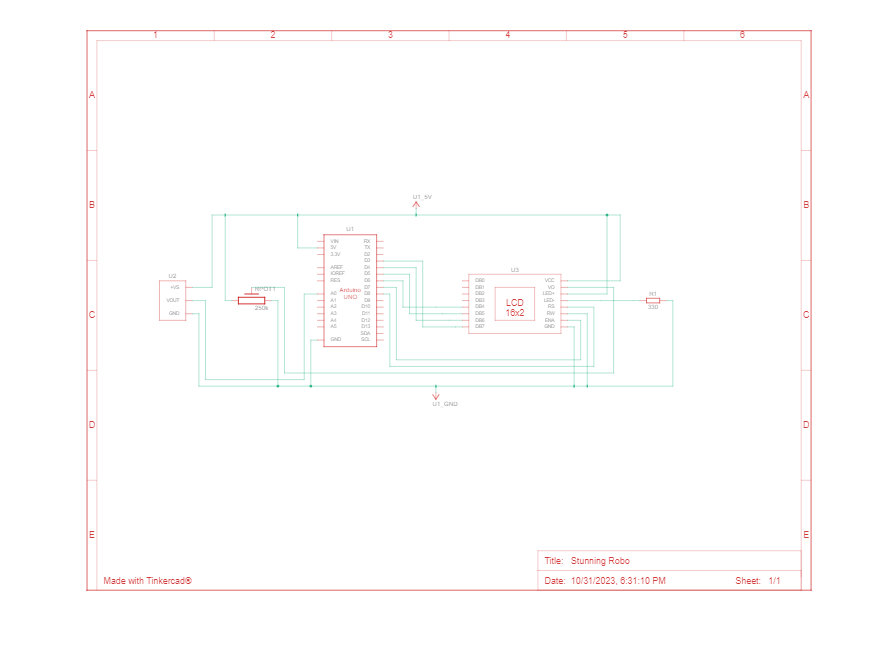
* Clear protocols for emergency situations, easily accessible to users.

Creating a final module for a smart public restroom involves combining these features to enhance hygiene, accessibility, and overall user experience, keeping in mind the needs of diverse user groups while prioritizing cleanliness, efficiency, and sustainability.

**IMPLEMENTATION OF TEMPERATURE SENSOR**

****

**SCHEMATIC DIAGRAM:**

****

**PROGRAM:**

#include "LiquidCrystal.h"

LiquidCrystal lcd(8,7,6,5,4,3);

int sensorPin = 0;

void setup()

{

Serial.begin(9600);

lcd.begin(16,2);

}

void loop()

{

int reading = analogRead(sensorPin);

float voltage = reading \* 4.68;

voltage /= 1024.0;

float temperatureC = (voltage - 0.5) \* 100;

Serial.print(temperatureC);

Serial.println(" degrees C");

lcd.setCursor(0,0);

lcd.print("Temperature Value ");

lcd.setCursor(0,1);

lcd.print(" degrees C");

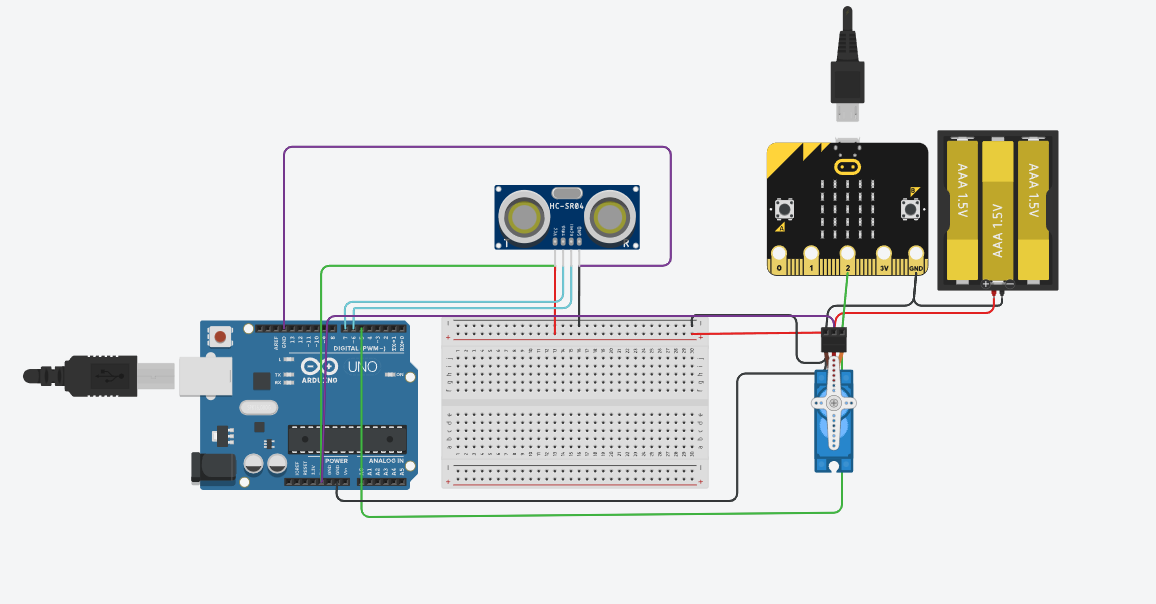
lcd.setCursor(11,1);

lcd.print(temperatureC);

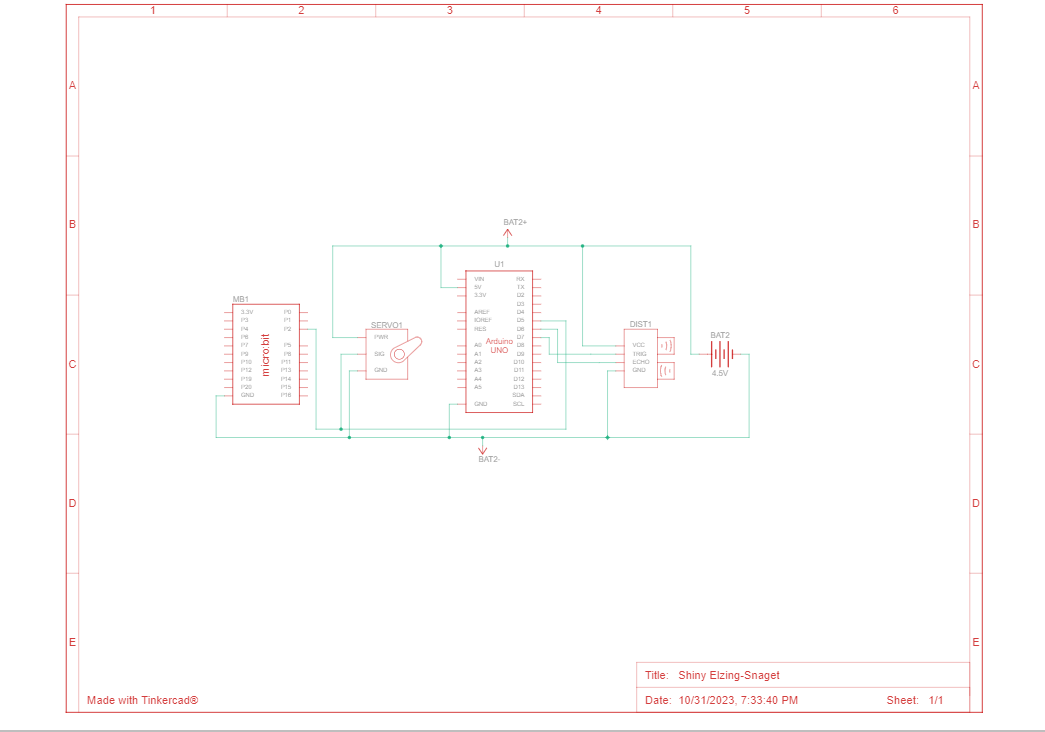
delay(100);

}

**IMPLEMENTATION OF AUTOMATIC FLUSHING TOILET**

****

**SCHEMATIC DIAGRAM:**

****

**PROGRAM:**

#include <Servo.h>

const int trigPin = 2; // Define Trig pin

const int echoPin = 3; // Define Echo pin

Servo servo; // Create a servo object

int distance; // Variable to hold distance value

void setup() {

Serial.begin(9600);

servo.attach(9); // Attach servo signal to pin 9

pinMode(trigPin, OUTPUT);

pinMode(echoPin, INPUT);

}

void loop() {

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

distance = pulseIn(echoPin, HIGH) / 58; // Calculate distance in centimeters

Serial.print("Distance: ");

Serial.print(distance);

Serial.println(" cm");

if (distance < 10) { // If the distance is less than 10 cm (adjust as needed)

servo.write(90); // Rotate the servo to simulate flushing

delay(1000); // Wait for the flush to complete

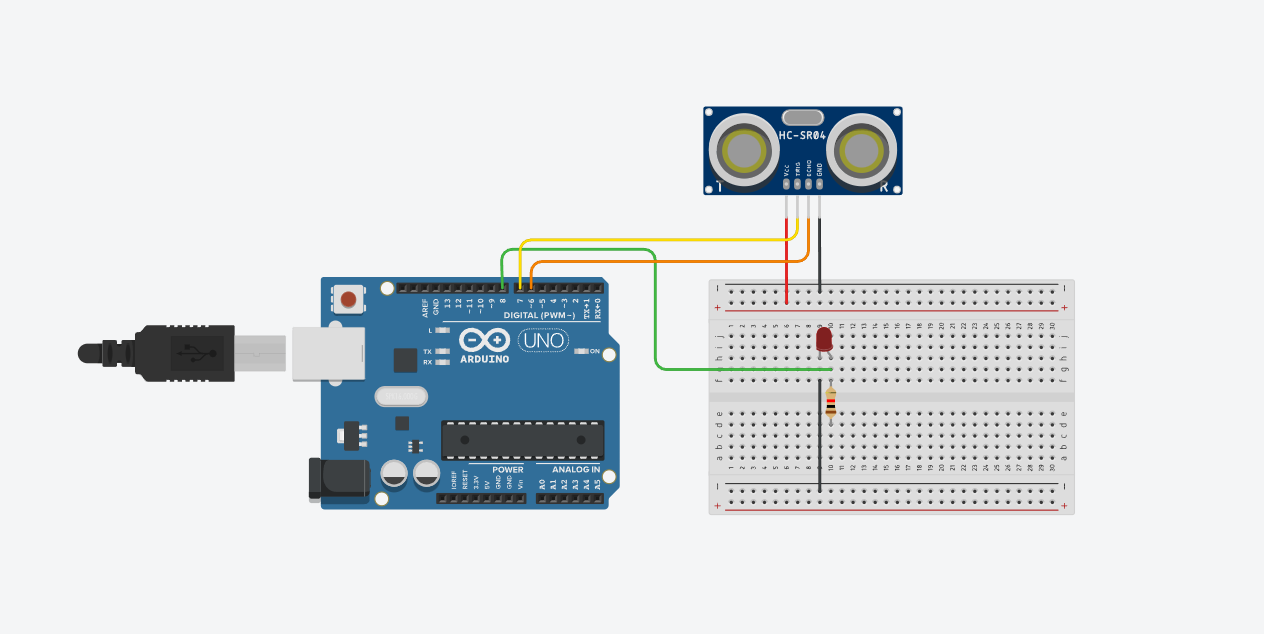
servo.write(0); // Return the servo to its initial position

}

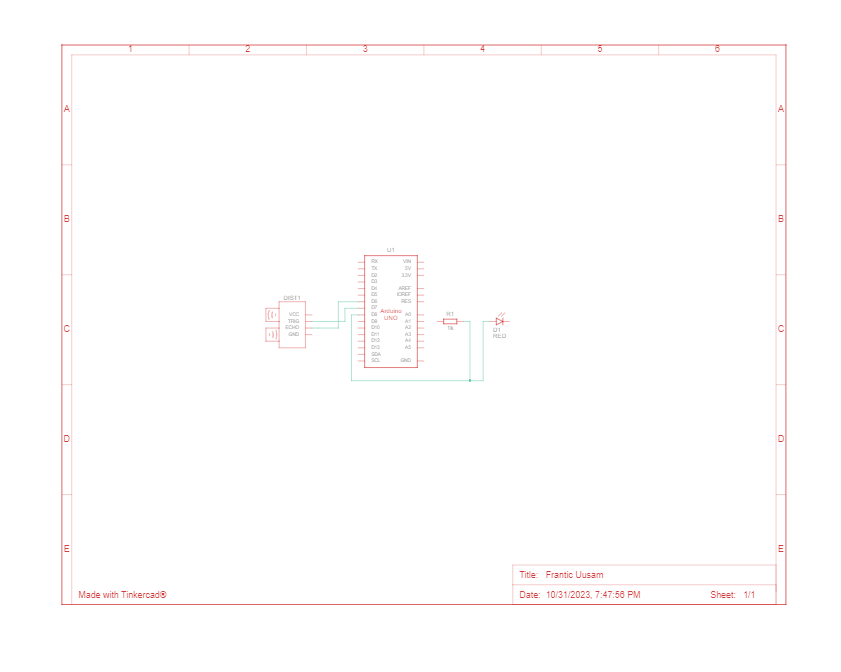
delay(100); // Delay for stability

}

**IMPLEMENATION OF OCCUPANCY MONITORING**

****

**SCHEMATIC DIAGRAM:**



**PROGRAM:**

const int trigPin = 2; // Trig pin of the ultrasonic sensor

const int echoPin = 3; // Echo pin of the ultrasonic sensor

const int ledPin = 7; // Pin to control the LED

long duration;

int distance;

void setup() {

pinMode(trigPin, OUTPUT);

pinMode(echoPin, INPUT);

pinMode(ledPin, OUTPUT);

Serial.begin(9600);

}

void loop() {

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

duration = pulseIn(echoPin, HIGH);

distance = duration \* 0.034 / 2;

Serial.print("Distance: ");

Serial.println(distance);

if (distance < 50) { // Change the threshold according to your setup

digitalWrite(ledPin, HIGH); // LED indicating occupancy

} else {

digitalWrite(ledPin, LOW); // LED indicating vacancy

}

delay(1000); // Adjust the delay as necessary

}