SMART CITY: IoT Advancements in Rail Control, Dam Management & Road Lightning

S074 Kermeen Deboo

S084 Atharva Jadhav

S089 Jyoti Kumbhar

S100 Vedant Patil

S113 Bhumika Shelar

S120 Shweta Todgire

Introduction:

Smart City Project is designed to enhance urban safety, energy efficiency, and reduse man power using IoT technology. It focuses on key areas like water management, smart lighting, and railway safety to improve city infrastructure.

In this project, we have implemented:

1. **Smart Automated Dam** – Controls water levels to prevent flooding and ensures water reuse.
2. **Smart Street Lamps** – Automatically turn ON in darkness and adjust brightness when motion is detected.
3. **Remote-Controlled Train Model** – Demonstrates wireless train movement using a gear motor.
4. **Automated Railway Crossing** – Closes gates automatically when a train approaches to prevent accidents.

For stable power supply, we have used **SMPS (Switched Mode Power Supply)** to prevent voltage fluctuations and ensure smooth operation.

The project aims to reduce human effort, save energy, and improve city safety through automation and IoT-based smart solutions.

Components Used:

1. **SMPS (Switched Mode Power Supply):** SMPS converts AC to DC and provides a stable power supply while preventing voltage fluctuations. It is efficient, compact, and ensures reliable operation of electronic components.

2. **Arduino:** Arduino is a microcontroller that processes sensor data and controls devices like motors and LEDs for automation in various applications.

3. **Arduino Nano:** Arduino Nano is a compact microcontroller used for small-scale automation. It processes sensor data, controls LEDs, motors, and other components while consuming less space and power.

4. **IR Sensor:** It is used for object detection and motion sensing. It detects infrared radiation and is commonly used in automation, security systems, and smart lighting.

5. **LDR Sensor:** It detects light intensity and is used for automatic lighting control. It turns lights ON in darkness and OFF in bright light, helping in energy saving.

6. **LED's:** They are small, energy-efficient lights used for lighting up areas or showing signals. They last longer and use less power than regular bulbs.

7. **Servo Motor:** A servo motor is a motor that can rotate to a specific position. It is used for precise movement like opening doors, controlling gates, or adjusting angles in machines.

8. **Motor Pump:** A motor pump is a device that uses a motor to pump liquids (like water) from one place to another. It's commonly used in applications like water circulation or draining systems.

9. **Motor Pump:** A motor pump is a device that uses a motor to pump liquids (like water) from one place to another. It's commonly used in applications like water circulation or draining systems.

10. **Servo Motor:** A float sensor detects the level of liquids (like water). It floats on the surface of the liquid and sends a signal when the liquid reaches a certain level, used for applications like water level monitoring.

11. **Gear Wheel & Strip:** A gear wheel transfers motion and force between parts, controlling speed and direction. A strip is a narrow material used for connecting parts in mechanical systems.
12. **Gear Motor:** A gear motor is a motor that combines a motor and gear system to reduce speed and increase torque, used for precise control in machines like robots or conveyor belts.

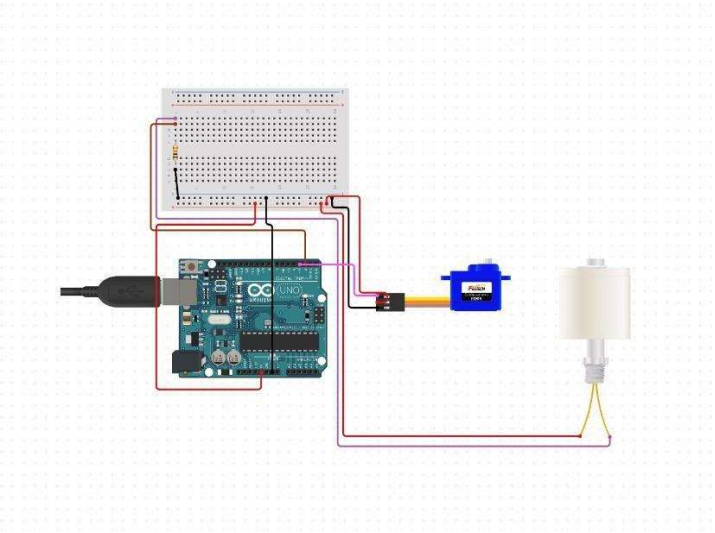
13. **Transmitter - Receiver:** A transmitter sends signals or data wirelessly, while a receiver receives those signals and converts them into a usable form. They are commonly used in wireless communication systems, such as remote controls.

14. **7.4 V Battery:** A 7.4V battery is a rechargeable power source that provides a 7.4 volts output. It's commonly used in devices like RC cars, drones, and small electronics for efficient power supply.

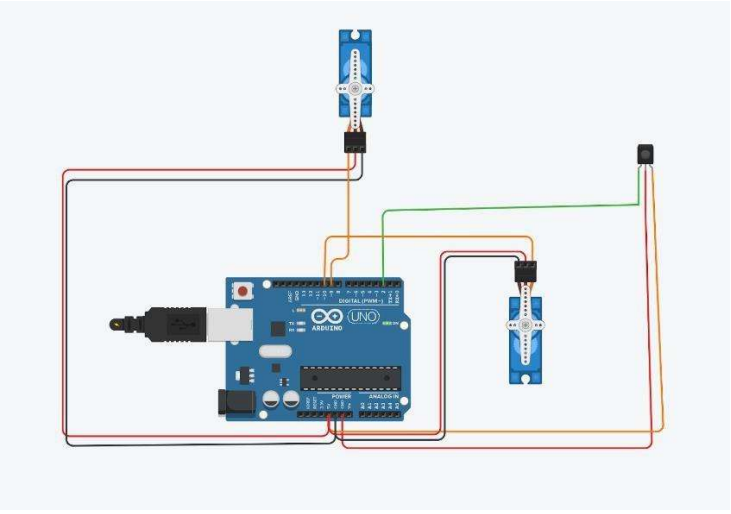
15. **Jumper Wires:** Jumper wires are flexible electrical wires used to connect different components on a breadboard or circuit. They are commonly used in prototyping and testing circuits.

Circuit Diagrams:

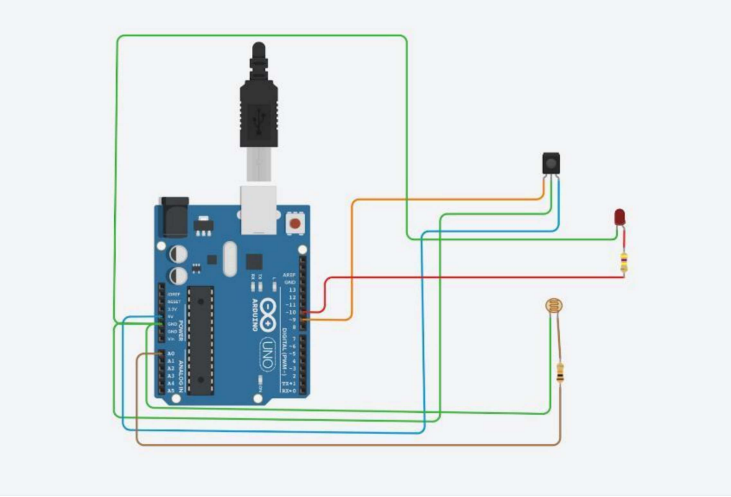
Dam Automation -



Railway Crossing -



Street Lights –



Working:

1. **Automated Dam System:**
    - A float sensor detects the water level in the dam.
    - When the water level rises, the sensor sends a signal to the Arduino Nano.
    - The Arduino triggers the servo motor to open the dam door, allowing excess water to flow out to the river.
    - The water flows through pipes into a container, and an external pump pumps the water back into the dam for reuse.
    - Once the water level lowers, the door is automatically closed.

2. **Smart Street Lighting System:**
    - The LDR sensor detects the surrounding light levels.
    - When it gets dark, the sensor sends a signal to the Arduino, which turns ON the LED street lights.
    - If there's any movement detected by the IR sensor, it adjusts the brightness of the lights.
    - When it gets bright again, the lights turn OFF automatically.

3. **Automated Railway Crossing:**
    - The IR sensor detects an incoming train.
    - The signal is sent to the Arduino, which activates the servo motors to close the railway gates.
    - Once the train passes, the gates open again to allow vehicles to pass safely.

4. **Remote-Controlled Train Model:**
    - The train is controlled wirelessly using a transmitter and receiver.
    - The Receiver in the train receives commands to move forward, backward. Based on the commands, it drives the gear motor to control the movement of the train.

Code:

Dam Automation:

```
#include <Servo.h>

const int floatSensorPin = 2;        // Float sensor connected to pin 2
const int servoPin = 9;         // Servo signal connected to pin 9

Servo myServo;

// Variables for debouncing
int lastSensorState = HIGH;         // Previous state of the float sensor
int currentSensorState;
unsigned long lastDebounceTime = 0;          // Last time the sensor state changed
const unsigned long debounceDelay = 50;         // Debounce delay in milliseconds

// Servo position tracking
int lastServoPosition = 0;

void setup() {
  pinMode(floatSensorPin, INPUT_PULLUP);       // Enable pull-up resistor
  myServo.attach(servoPin);         // Attach the servo
  myServo.write(90);            // Set servo to initial position (90 degrees)
  Serial.begin(9600);
}

void loop() {
  // Read the current state of the float sensor
  int reading = digitalRead(floatSensorPin);

  // Check if the sensor state has changed
  if (reading != lastSensorState) {
    lastDebounceTime = millis();           // Update the debounce timer
```

```
  }

  // If the state is stable for longer than debounceDelay, consider it valid
  if ((millis() - lastDebounceTime) > debounceDelay) {
    if (reading != currentSensorState) {
      currentSensorState = reading;          // Update the current sensor state

      // Perform actions based on the new state
      if (currentSensorState == LOW && lastServoPosition != 0) {
        Serial.println("Float sensor is DOWN - Moving servo to 0 degrees");
        myServo.write(0);                    // Rotate servo to 0 degrees
        lastServoPosition = 0;               // Update servo position
      } else if (currentSensorState == HIGH && lastServoPosition != 90) {
        Serial.println("Float sensor is UP - Moving servo to 90 degrees");
        myServo.write(90);                   // Rotate servo to 90 degrees
        lastServoPosition = 90;              // Update servo position
      }
    }
  }

  // Update the last sensor state
  lastSensorState = reading;

  // Short delay for stability
  delay(10);
}
```

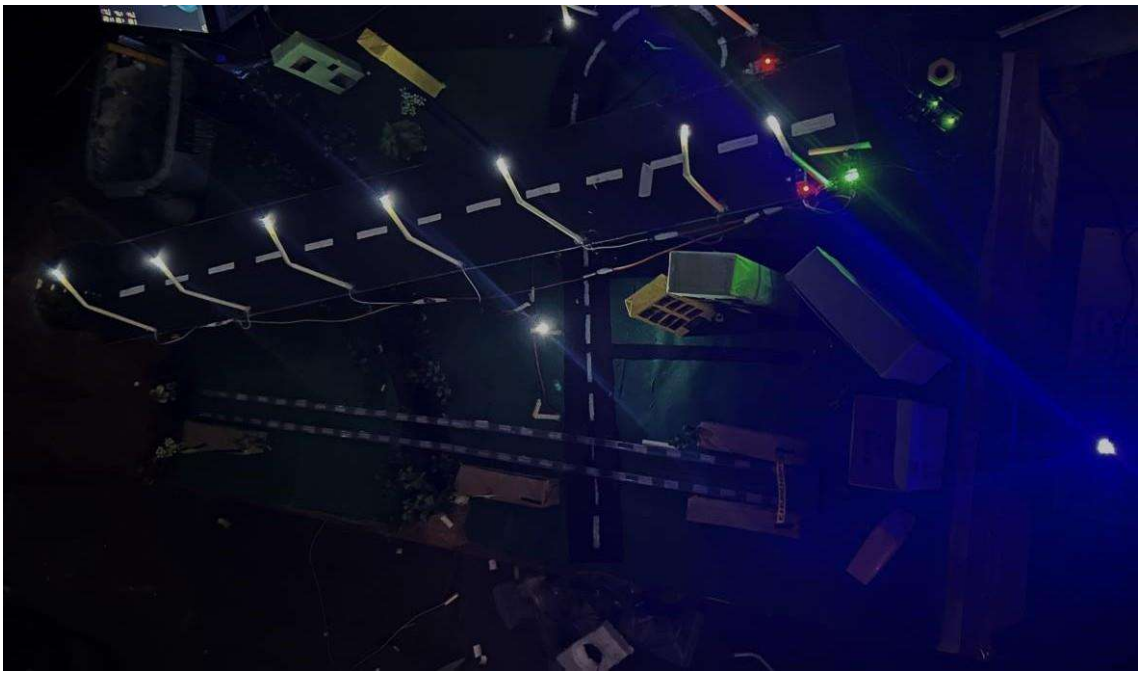
Street Lights & Railway Crossing:

```
#include <Servo.h>

const int ldrPin = A0;           // Pin where LDR is connected
const int ldrThreshold = 500;       // Threshold for LDR to detect day/night
const int irSensor1 = 2;        // IR sensor 1 pin
```

```
const int irSensor2 = 3;       // IR sensor 2 pin
const int irSensor3 = 4;       // IR sensor 3 pin
const int irLed1 = 5;          // LED 1 pin
const int irLed2 = 6;          // LED 2 pin
const int irLed3 = 7;          // LED 3 pin

Servo flapServo;        // Create a servo object to control the flap
int flapIrSensorPin = 2;       // IR sensor for the flap
int flapSensorValue = 0;       // Variable to store sensor reading

void setup() {
  // Setup for LDR and LEDs
  pinMode(ldrPin, INPUT);              // Set LDR pin as input
  pinMode(irSensor1, INPUT);           // Set IR sensor pins as input
  pinMode(irSensor2, INPUT);
  pinMode(irSensor3, INPUT);
  pinMode(irLed1, OUTPUT);             // Set LED pins as output
  pinMode(irLed2, OUTPUT);
  pinMode(irLed3, OUTPUT);

  // Setup for servo and flap IR sensor
  flapServo.attach(9);                 // Attach the servo to pin 9
  pinMode(flapIrSensorPin, INPUT);          // Set the IR sensor pin as input
  flapServo.write(0);                  // Start with the flap closed

  Serial.begin(9600);                // Start serial communication for debugging
}

void loop() {
  int ldrValue = analogRead(ldrPin);            // Read LDR value
  Serial.print("LDR Value: ");                  // Print LDR value
  Serial.println(ldrValue);

  if (ldrValue > ldrThreshold) {                // Day condition
                                                // Check IR and adjust LEDs
```

```
  int ir1 = digitalRead(irSensor1);
  int ir2 = digitalRead(irSensor2);
  int ir3 = digitalRead(irSensor3);

  // Set LED brightness based on IR detection
  analogWrite(irLed1, ir1 == LOW ? 255 : 50);          // Bright if detected, dim
otherwise
  analogWrite(irLed2, ir2 == LOW ? 255 : 50);          // Bright if detected, dim
otherwise
  analogWrite(irLed3, ir3 == LOW ? 255 : 50);          // Bright if detected, dim
otherwise
 } else {                                              // Night condition
                                                       // Turn off all LEDs at night
  digitalWrite(irLed1, LOW);
  digitalWrite(irLed2, LOW);
  digitalWrite(irLed3, LOW);
 }

 // Servo and flap functionality
 flapSensorValue = digitalRead(flapIrSensorPin);       // Read IR sensor for flap

 if (flapSensorValue == HIGH) {                        // Object detected (sensor is HIGH)
  flapServo.write(90);                                 // Rotate the servo to 90 degrees (open flap)
  Serial.println("Object Detected! Flap Opened");
 } else {                                              // No object detected (sensor is LOW)
  flapServo.write(0);                                  // Rotate the servo back to 0 deg (close flap)
  Serial.println("No Object Detected! Flap Closed");
 }

 delay(2500);                                          // Wait for a bit before reading again
}
```

Images:



Summary:

The Smart City IoT project enhances urban safety and efficiency using innovative solutions. It features an automated dam system to prevent flooding, smart street lighting that adjusts brightness to save energy, a remote-controlled train model, and an automated railway crossing to prevent accidents. An SMPS ensures stable power supply and protects against fluctuations. The project aims to improve safety, conserve energy, reduce manpower reliance, and optimize resources through IoT technology.

Reference:

Smart City by Shivam Yadav, SYCS batch 2023-24