**Project Report**

**Density controlled traffic light system**

By:

Khushal Shroff(23BCP137)

**Abstract**

This project aims to create a smart traffic light system that adjusts based on how many vehicles are on the road. Instead of using fixed timers, the system will use sensors or cameras to count the cars at an intersection. When traffic is heavy, the light will stay green longer to let more cars pass. When traffic is light, the system will shorten the wait time. By adjusting the lights based on real-time traffic, the system will help reduce congestion, save time, and improve traffic flow at busy intersections.

**Project Description**

Title**:** Density Controlled traffic light system

Aim: To design and implement a smart traffic light system that adjusts signal timings based on real-time traffic density

Components used:

* Arduino Uno
* IR Sensors
* Shift register (74HC595)
* Breadboard
* LEDs
* Resistors (470 Ω)
* Jumper wires (for connection)

Component Description:

* **Arduino Uno**: A popular microcontroller board used for building electronics projects. It has input/output pins that allow it to interact with sensors, motors, LEDs, and other components. It is programmable via a USB connection and is often used for automation, control systems, and prototyping.
* **IR Sensors**: Infrared sensors are used to detect objects or measure distances by emitting infrared light. They work by sending out an infrared beam and detecting how much of that light is reflected back. They are commonly used in obstacle detection, remote controls, and motion sensing.
* **Shift Register (74HC595)**: A shift register is an integrated circuit that expands the number of output pins available on a microcontroller. It allows data to be shifted in serially and output in parallel, making it useful for controlling multiple devices like LEDs, displays, or relays with fewer microcontroller pins.
* **Breadboard:** A tool for building and testing circuits without soldering. Components and wires are inserted into its holes for easy, temporary connections.
* **LEDs**: Small lights that emit light when current flows through them, commonly used as indicators or for lighting in electronics.
* **Resistors :** Components that limit the flow of current, protecting other parts of the circuit, like LEDs, from too much electricity.
* **Jumper Wires**: Flexible wires used to connect different components on a breadboard, allowing easy setup and changes in a circuit.

Algorithm and implementation procedure:

Step 1: Gather Materials

Step 2: Set Up the IR Sensors for Vehicle Detection

1. \*\*Position the IR Sensors:\*\* Place IR sensors at each lane of the intersection where they will count the vehicles.

2. \*\*Connect Sensors to Arduino:\*\* Connect the output pins of each IR sensor to the digital pins on the Arduino. Ensure the sensors are positioned correctly to detect vehicle presence in each lane.

3. \*\*Power the Sensors:\*\* Connect each sensor’s VCC to the Arduino's 5V pin and GND to the Arduino's ground.

### Step 3: Configure the Traffic Lights with LEDs and Shift Registers

1. \*\*Design LED Circuit for Traffic Lights:\*\* Arrange the LEDs for each traffic light in red, yellow, and green configurations. Each light will correspond to a traffic signal direction.

2. \*\*Use Shift Registers for Multiple LEDs:\*\* Connect each LED’s cathode to ground through a resistor, and connect each anode to the shift register. The 74HC595 shift register will expand the number of outputs for the Arduino, allowing it to control multiple LEDs.

3. \*\*Link Shift Registers to Arduino:\*\* Connect the shift register pins (data, clock, and latch) to digital pins on the Arduino. Follow the datasheet for correct wiring.

4. \*\*Test the LED Circuit:\*\* Run a test code on Arduino to ensure each LED lights up as expected when powered through the shift register.

### Step 4: Write the Code for Traffic Density Detection and Light Control

1. \*\*Define Sensor and LED Pins:\*\* In the code, assign the correct pins for each IR sensor and LED.

2. \*\*Set Up the Timing Logic:\*\* Implement a function to read data from each IR sensor to detect traffic density. Count how long the IR sensors are blocked, indicating the presence of vehicles.

3. \*\*Adjust Traffic Light Timing Based on Sensor Input:\*\*

- If the IR sensor detects continuous traffic (indicating high density), extend the green light duration.

- If there’s no vehicle for a set time (indicating low density), reduce the green light time.

- Implement a balanced timing for the yellow and red lights to ensure smooth transitions.

### Step 5: Implement the Main Control Logic

1. \*\*Loop Through Intersection Phases:\*\* Using a `loop` function, create cycles for each direction (north-south and east-west).

2. \*\*Monitor Sensor Input in Real-Time:\*\* Within each cycle, continuously check sensor inputs to adjust green light duration dynamically.

3. \*\*Control the LEDs Accordingly:\*\* For each phase, turn on the appropriate LEDs (green, yellow, red) using the shift registers to manage the timing.

### Step 6: Test and Debug

1. \*\*Simulate Traffic Conditions:\*\* Place objects in front of the IR sensors to simulate vehicles and observe how the timing of the lights adjusts.

2. \*\*Refine Code:\*\* Make adjustments in the code to optimize green light timing for realistic traffic flow.

3. \*\*Check All Connections:\*\* Ensure that all components are correctly connected and powered to prevent any hardware issues.

### Step 7: Finalize and Deploy

1. \*\*Optimize and Clean Up the Circuit:\*\* If all functions are working as expected, tidy up the wiring on the breadboard and ensure secure connections.

2. \*\*Enclose in a Housing (Optional):\*\* To protect the sensors and Arduino, consider creating an enclosure that replicates a traffic signal model.

3. \*\*Observe Long-Term Performance:\*\* Test the system over an extended period to ensure reliable performance under varying traffic conditions.

This smart traffic light system will now dynamically adjust based on the real-time vehicle count at the intersection, improving traffic flow and reducing congestion effectively.

CODE:

#define NOP \_\_asm\_\_("nop")

// LED pattern

unsigned char pattern = 0b00000000; // Initial state: all RED

int sensorCount = 4;

// Time constants

const long baseGreenDelay = 50; // Base time for green light in milliseconds

const long redDelay = 50;   // Time for red light in milliseconds

// Sensor Pins

int sensorPin[4] = {A0, A1, A2, A3};

// SHR control

int SER = 2;     //pin-14 in 595

int RST = 3;    //pin-10 in 595

int inClk = 4;    //pin-11 in 595

int latchpin = 5;   //pin-12 in 595

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void init\_Port() {

    for (int i = 0; i < sensorCount; i++) {

        pinMode(sensorPin[i], INPUT);

    }

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void init\_Light() {

    pinMode(SER, OUTPUT);

    pinMode(RST, OUTPUT);

    pinMode(inClk, OUTPUT);

    pinMode(latchpin, OUTPUT);

    digitalWrite(SER, LOW);

    digitalWrite(RST, LOW);

    delay(500);

    digitalWrite(RST, HIGH);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void write\_LED(unsigned char val) {

    digitalWrite(latchpin, LOW);

    NOP;

   // digitalWrite(inClk, LOW);

    NOP;

    shiftOut(SER, inClk, MSBFIRST, val);

    NOP;

    digitalWrite(latchpin, HIGH);

    NOP;

    NOP;

    NOP;

    //digitalWrite(latchpin, LOW);

    NOP;

    //digitalWrite(inClk, LOW);

    NOP;

}

// Function to display the current LED signal

void displaySignal() {

    write\_LED(pattern);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void updatePattern(int state) {

    // Reset the pattern to RED for all

    pattern = 0b00000000;

    // Update pattern based on the current state

    switch (state) {

        case 0: // North Green

            pattern |= 0b01010110; // Set North to Green

            break;

        case 1: // East Green

            pattern |= 0b01011001; // Set East to Green

            break;

        case 2: // South Green

            pattern |= 0b01100101; // Set South to Green

            break;

        case 3: // West Green

            pattern |= 0b10010101; // Set West to Green

            break;

    }

}

// Function to read the vehicle density from sensors

int readDensity(int sensorId) {

    return analogRead(sensorPin[sensorId]); // Adjust according to your sensor type

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void setup() {

    Serial.begin(9600);

    Serial.println(" ------- Automatic Traffic Light Control based on Density --------");

    // SHR Init

    init\_Light();

    init\_Port();

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void loop() {

    for (int state = 0; state < sensorCount; state++) {

        // Read the density for the current state

        int density = readDensity(state);

        Serial.print("density: ");

        Serial.println(density);

        // Calculate green light duration based on density

        long greenDelay = baseGreenDelay;

        if (density > 800) {

            greenDelay += 60000; // High traffic: increase green duration

        }

        else if(density > 400){

            greenDelay += 2000;

        }

        updatePattern(state);

        Serial.print("state: ");

        Serial.println(state);

        Serial.print("p: ");

        Serial.println(pattern);

        displaySignal();

        delay(greenDelay); // Dynamic green light duration

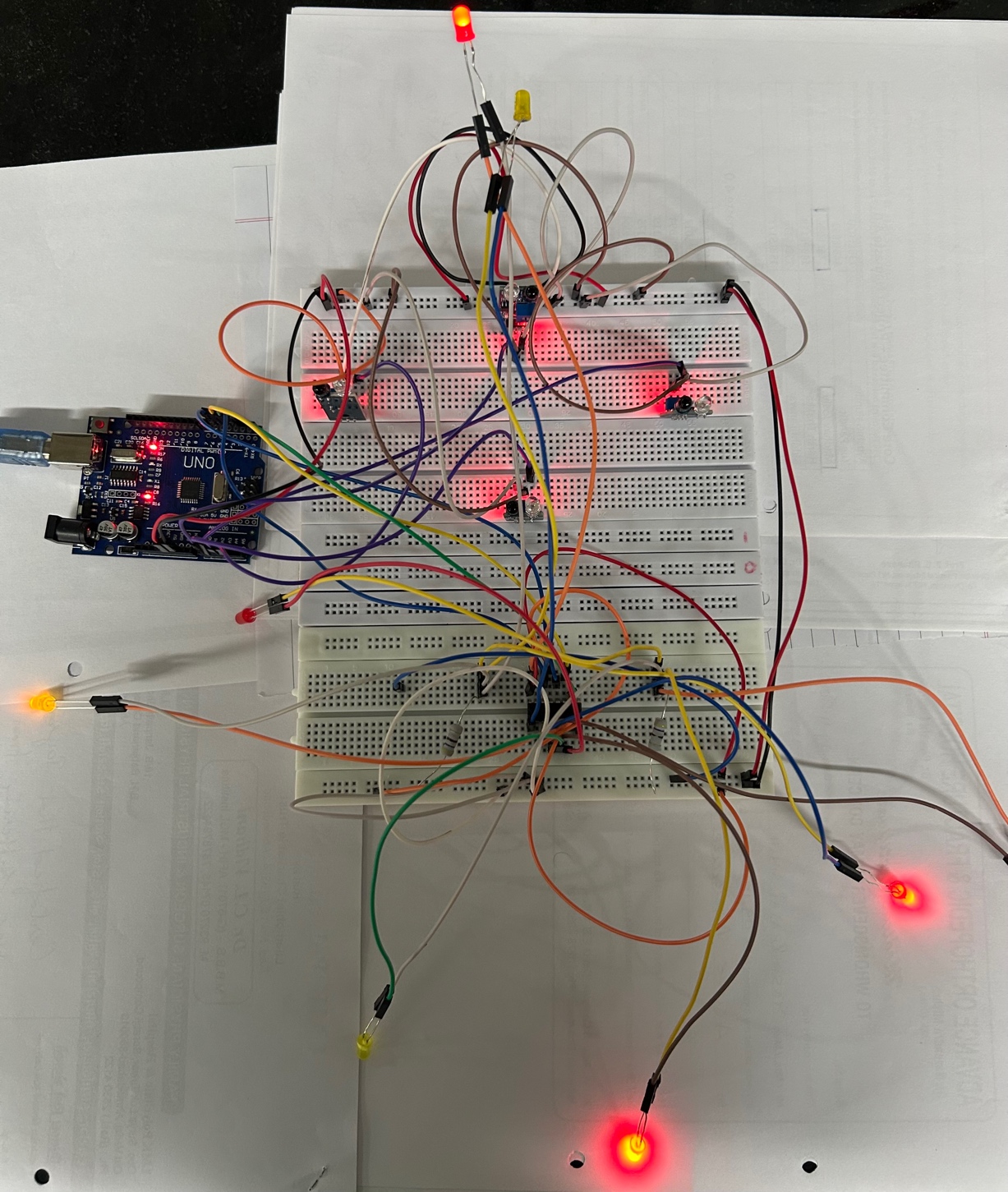
        displaySignal();

        delay(redDelay); // Fixed time for red light

    }

}

Project photo:



Circuit Explanation:

In this project, the circuit connections are made using a **breadboard**, which serves as the foundation for connecting all components without soldering. The **Arduino Uno** microcontroller is connected to a laptop via USB for power and data transfer, supplying **5V DC** to the board and circuit components.

The Arduino Uno is connected to the laptop for power and control signals. The power rails on the breadboard are connected to the Arduino’s **5V** and **GND** pins to distribute power to the components.

Each IR sensor is placed on the breadboard and connected to one of the input pins of the Arduino. The sensors are powered by the 5V and GND lines on the breadboard, while their output pins are connected to the corresponding Arduino pins to relay traffic density information.

A **shift register** (74HC595) is used to expand the Arduino’s output capabilities, allowing it to control multiple LEDs. The shift register is connected to the Arduino via three pins: data, clock, and latch. It receives data from the Arduino and distributes it to the LEDs. The shift register is also powered by the breadboard’s power rails.

The LEDs are placed on the breadboard and connected to the shift register. They receive signals from the shift register to simulate traffic lights at the intersection, lighting up based on real-time traffic density.

Jumper wires are used to make the necessary connections between the Arduino, sensors, shift register, and LEDs.

Circuit Working:

1. **Traffic Detection**: The IR sensors detect traffic density by monitoring vehicle presence in each lane. Each sensor sends a signal to the Arduino based on whether a vehicle is detected.
2. **Data Processing**: The Arduino processes the input from the sensors and determines the traffic density at each lane. The lane with the highest density is prioritized for a longer green light.
3. **Shift Register Control**: The processed traffic information is sent from the Arduino to the shift register, which controls the traffic lights (LEDs). The shift register expands the Arduino’s output capacity, allowing it to control multiple LEDs while using fewer pins.
4. **LED Management**: The LEDs, representing traffic lights, are powered on and off based on the shift register’s output. The green LED for a lane with heavy traffic will stay on longer, while the red LED will be on for lanes with less traffic.
5. **Dynamic Traffic Adjustment**: The system continuously adjusts traffic light timings in real-time based on input from the IR sensors, dynamically managing the traffic flow and minimizing delays.

Project Result:

This project successfully implements a density-controlled traffic light system using an Arduino Uno, IR sensors, shift register, LEDs, and a breadboard. By detecting real-time traffic density through sensors, the system dynamically adjusts traffic signal timings, prioritizing lanes with higher vehicle volumes. This approach reduces congestion, minimizes waiting times, and improves traffic flow efficiency at intersections. The use of a shift register allows to extend the number inputs on Arduino board and efficient control of multiple traffic lights, making the system scalable and adaptable for real-world applications.

Conclusion:

In this group project, we gained hands-on experience with Arduino, IR sensors, and shift registers for real-time applications. We developed skills in coding for dynamic timing control, integrating multiple components, and troubleshooting complex circuits. This project also enhanced our understanding of sensor-based automation and embedded systems, illustrating how technology can optimize real-world tasks like traffic management.