**Team name: Brute Fives**

**Traffic Signal lights simulator**

**Introduction:**

Traffic Signal lights simulator uses 8 led bulbs to simulate a working traffic signal with a green and a red bulb per lane. We have considered 4 lanes for this project. A lane’s duration of green signal can be pre fed as a global variable which will in turn calculate the duration a lane has to wait ie., stay in red. When a lane has a green light, others need to be waiting for their turn to be green while staying at red.

**Components:**

LED bulbs: Green X4, Red X4

Arduino Uno board

Jumper wires

Bread board

**Code:**

|  |
| --- |
| #defineIFGREEN(num)num=='g'?1:0 |
| |  |  | | --- | --- | |  | #define IFHIGH(val) val>0?'HIGH':'LOW' | |  | int lane1[2] ={0,1}; | |  | int lane2[2] ={2,3}; | |  | int lane3[2] ={4,5}; | |  | int lane4[2] ={6,7}; | |  |  | |  |  | |  | void setColor(char color, int highOrLow) | |  | { | |  | if(IFGREEN(color)) | |  | { | |  | digitalWrite(lane1[1],IFHIGH(highOrLow)); | |  | digitalWrite(lane2[1],IFHIGH(highOrLow)); | |  | digitalWrite(lane3[1],IFHIGH(highOrLow)); | |  | } | |  | else | |  | { | |  | digitalWrite(lane1[0],IFHIGH(highOrLow)); | |  | digitalWrite(lane2[0],IFHIGH(highOrLow)); | |  | digitalWrite(lane3[0],IFHIGH(highOrLow)); | |  | } | |  |  | |  |  | |  | } | |  |  | |  | void sequence(int ,int); | |  | void setup() { | |  | // put your setup code here, to run once: | |  | pinMode(lane1[0], OUTPUT); | |  | pinMode(lane1[1], OUTPUT); | |  | pinMode(lane2[0], OUTPUT); | |  | pinMode(lane2[1], OUTPUT); | |  | pinMode(lane3[0], OUTPUT); | |  | pinMode(lane3[1], OUTPUT); | |  | pinMode(lane4[0], OUTPUT); | |  | pinMode(lane4[1], OUTPUT); | |  | setColor('r',1); | |  | setColor('g',0);  int duration\_lane1 = 2, duration\_lane2 = 2, duration\_lane3 = 2, duration\_lane4 = 2; | |  |  | |  | } | |  |  | |  | void loop() { | |  | // put your main code here, to run repeatedly: | |  |  | |  |  | |  | sequence(1, duration\_lane1); | |  | sequence(2, duration\_lane2); | |  | sequence(3, duration\_lane3); | |  | sequence(4, duration\_lane4); | |  |  | |  | //signal(SIGINT, sigIntHandler); | |  | } | |  | void sequence(int lane,int duration) | |  | { | |  | if(lane==1) | |  | { | |  | digitalWrite(lane1[0], LOW); | |  | digitalWrite(lane1[1], HIGH); | |  | delay((1000\*duration)); | |  | digitalWrite(lane1[1], LOW); | |  | digitalWrite(lane1[0], HIGH); | |  | } | |  | else if(lane==2) | |  | { | |  | digitalWrite(lane2[0], LOW); | |  | digitalWrite(lane2[1], HIGH); | |  | delay((1000\*duration)); | |  | digitalWrite(lane2[1], LOW); | |  | digitalWrite(lane2[0], HIGH); | |  | } | |  | else if(lane==3) | |  | { | |  | digitalWrite(lane3[0], LOW); | |  | digitalWrite(lane3[1], HIGH); | |  | delay((1000\*duration)); | |  | digitalWrite(lane3[1], LOW); | |  | digitalWrite(lane3[0], HIGH); | |  | } | |  | else | |  | { | |  | digitalWrite(lane4[0], LOW); | |  | digitalWrite(lane4[1], HIGH); | |  | delay((1000\*duration)); | |  | digitalWrite(lane4[1], LOW); | |  | digitalWrite(lane4[0], HIGH); | |  | } | |  |  | |  | } | |
|  |