HC-SR04 Dead Rail block signal Arduino Code

```
/* Using the HC-RS04 Ultrasonic sensor.

  Use for garden railroad or dead rail block signal detection.
  Trains are detected by a HC-RS04 at each end of the block
  section. Code is written so the occupied block will not clear
  until trains leave the block and a clear signal is established.

  This circuit can be used for railroad crossings by activating
  road crossing lights, as train approaches and turns off after
  the train clears the road crossing.

  Displays results can be viewed on Serial Monitor

  Baldy & Palms Railroad
  Ron Karlsson 2021

*/

// Code for HC-RS04 Garden Railroad or Dead Rail block detection signal circuit

const int TRIGGER_PIN_1 = 5; // West sensor
const int ECHO_PIN_1 = 7;    // West sensor
const int TRIGGER_PIN_2 = 8; // East sensor
const int ECHO_PIN_2 = 10;   // East sensor

int ledG = 11; // Green signal light
int ledR = 12; // Red signal light
// int ledY = 13; // Yellow signal light (Initiate if running three aspect signals)
int dt = (500); // delay half second

float duration1, distance1; // Stores calculated distance in cm for First Sensor (WEST Sensor)
float duration2, distance2; // Stores calculated distance in cm for Second Sensor (EAST Sensor)

void setup() {

  pinMode(TRIGGER_PIN_1, OUTPUT);
  pinMode(ECHO_PIN_1, INPUT);
  pinMode(TRIGGER_PIN_2, OUTPUT);
  pinMode(ECHO_PIN_2, INPUT);

  pinMode(ledG, OUTPUT);
  pinMode(ledR, OUTPUT);
  //pinMode(ledY, OUTPUT); // Initiate if running three aspect signals
  Serial.begin (9600);
}
enum BLOCK1  // Name of variable of track occupancy block1. Identifies the four signal states
{
  GREEN_CLR,
  RED1,
  RED2,
  GRNYLW,
};

BLOCK1 sigState = GREEN_CLR; // This identifies the BLOCK1 default signal state

void loop() { // Activating HC-RS04 sensors to begin sensing and calculating distance data

  digitalWrite(TRIGGER_PIN_1, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIGGER_PIN_1, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIGGER_PIN_1, LOW);

  duration1 = pulseIn(ECHO_PIN_1, HIGH);
  distance1 = (duration1 * .034) / 2;
  delay(100); // For program stability

  digitalWrite(TRIGGER_PIN_2, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIGGER_PIN_2, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIGGER_PIN_2, LOW);

  duration2 = pulseIn(ECHO_PIN_2, HIGH);
  distance2 = (duration2 * .034) / 2;
  delay(100); // For program stability

  // Serial Print code to view and adjust the HC-RS04 sensor readings.
  Serial.print("distance1 west "), Serial.print(distance1), Serial.println(" cm");
  Serial.print("distance2 east "), Serial.print(distance2), Serial.println(" cm");
  delay(100);

  switch (sigState)
    // sigState is the variable of BLOCK1, program will look at each state and act
    // accordingly based on the reading of sensor 1 (WEST) and sensor 2 (EAST)
  {
    case GREEN_CLR:
      grnsig(distance1, distance2);
      break;
    case RED1:
      redsig1(distance1, distance2);
      break;
    case RED2:
      redsig2(distance1, distance2);
      break;
    case GRNYLW:
      gysig(distance1, distance2);
      break;
  }
}

void grnsig(int distance1, int disance2) { // default state of BLOCK1 with all clear Green Signal
  digitalWrite(ledG, HIGH);
  digitalWrite(ledR, LOW);
  // Distance measured in centimeters, HC-RS04 is sensing from 2 cm to 100 cm
  // HC-RS04 will trigger when an object is detected within 10cm.  Adjust these
  // trigger values to suit your needs.

  if (distance1 <= 10 && distance2 >= 30) { // train enters from west through WEST sensor
(redsig1)
    sigState = RED1;
  }
  else if (distance1 >= 30 && distance2 <= 10) { // train enters from east through EAST sensor
(redsig2)
    sigState = RED2;
  }
}
```

```
void redsig1(int distance1, int distance2) { // BLOCK1 in a Red Signal state until train exits
  digitalWrite(ledG, LOW);
  digitalWrite(ledR, HIGH);

  if (distance1 >= 30 && distance2 <= 10) { // train exits BLOCK1 through EAST sensor
    sigState = GRNYLW;
  }
}
void redsig2(int distance1, int distance2) {
  digitalWrite(ledG, LOW);
  digitalWrite(ledR, HIGH);

  if (distance1 <= 10 && distance2 >= 30) { // train exits BLOCK1 through WEST sensor
    sigState = GRNYLW;
  }
}
// Adding three aspect lighting is easy by just changing gysig case.
// Initiate yellow LED code and have GRNYLW go to ledY HIGH instead of the
// two aspect blinking sequence before clearing to default state of grnsig.

void gysig(int distance1, int distance2) {
  digitalWrite(ledG, HIGH); // change to LOW if running three aspect signals
  digitalWrite(ledR, HIGH); // change to LOW if running three aspect signals
  //digitalWrite(ledY, HIGH); (Initiate if running three aspect signals)
  //delay(5000); // Yellow light HIGH for 5 seconds before claring to default state GREEN_CLR
  delay(dt);
  digitalWrite(ledR, LOW);
  delay (dt);
  digitalWrite(ledR, HIGH);
  delay(dt);
  digitalWrite(ledR, LOW);
  delay(dt);
  digitalWrite(ledR, HIGH);
  delay(dt);
  digitalWrite(ledR, LOW);
  //delay(dt); removed this delay and added 5 second to stop premature signal change.
  delay(5000);

  if (distance1 >= 20 && distance2 >= 20) {
    sigState = GREEN_CLR;
    }
  }
```