Electronics and Communication Engineering Department
Shri Ramdeobaba College of Engineering & Management, Nagpur

Teachers Assessment Activity

MICROCONTROLLERS AND INTERFACING
ECT 353

5th Semester B.Tech. Session-2023-24

Project Report

Title: Automated Rail Gate Control

Project Group Students: **1.Gaurav Agrawal(42)**
2.Priyansh Nigam (52)

Course Coordinator: **Prof Bhumika A.Neole**

Automated rail gate control

COMPONENT USED WITH SPECIFICATION:

1. Microcontroller (e.g., ATmega328): 8-bit, 16 MHz, 32 KB flash memory, 2 KB SRAM; controls sensors and motor driver for gate operation.

2. IR Transmitter and Receiver (IR Pair Module): Operating range 5-30 cm; wavelength around 940 nm; detects train by sensing interruptions in the IR beam.

3. L293D Motor Driver IC: Dual H-bridge; operates at 4.5V-36V; output current of 600mA per channel; allows bidirectional control of gate motor.

4. Motor (DC Motor): Operating voltage 12V, torque rating around 5 kg-cm; drives the opening and closing of the gate barrier.

5. Red and Green LEDs: 5mm, operating voltage 2V (Red), 3V (Green); serves as visual indicators for vehicles and pedestrians.

6. Bell/Buzzer: Operating voltage 5V-12V, sound output around 80dB; provides an audible warning when a train approaches.

7. Capacitors (e.g., 100µF, 0.1µF) and Resistors (e.g., 220Ω): Stabilizes the circuit, filters noise, and limits current to LEDs and other components.

8. Power Supply: 12V, 2A DC adapter; supplies consistent power to microcontroller, motor, and other components.

These components, with their specific features, make up an effective and safe automated railway gate control system that operates efficiently based on the detection of train movement.

SOFTWARE SPECIFICATIONS :

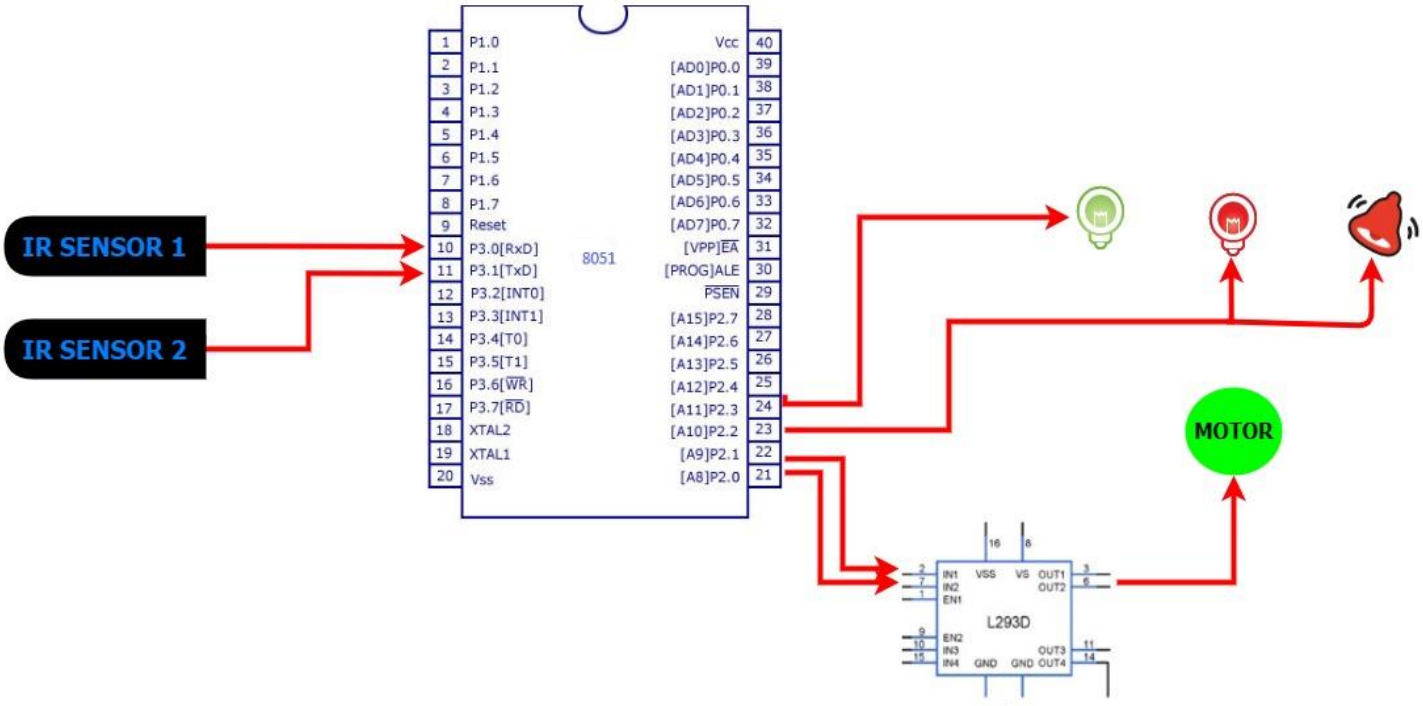
- Keil uVision5: IDE software used for coding and debugging microcontroller programming.

- 10. Proteus Software: Electronic design software for simulating and testing circuit performance before physical implementation.

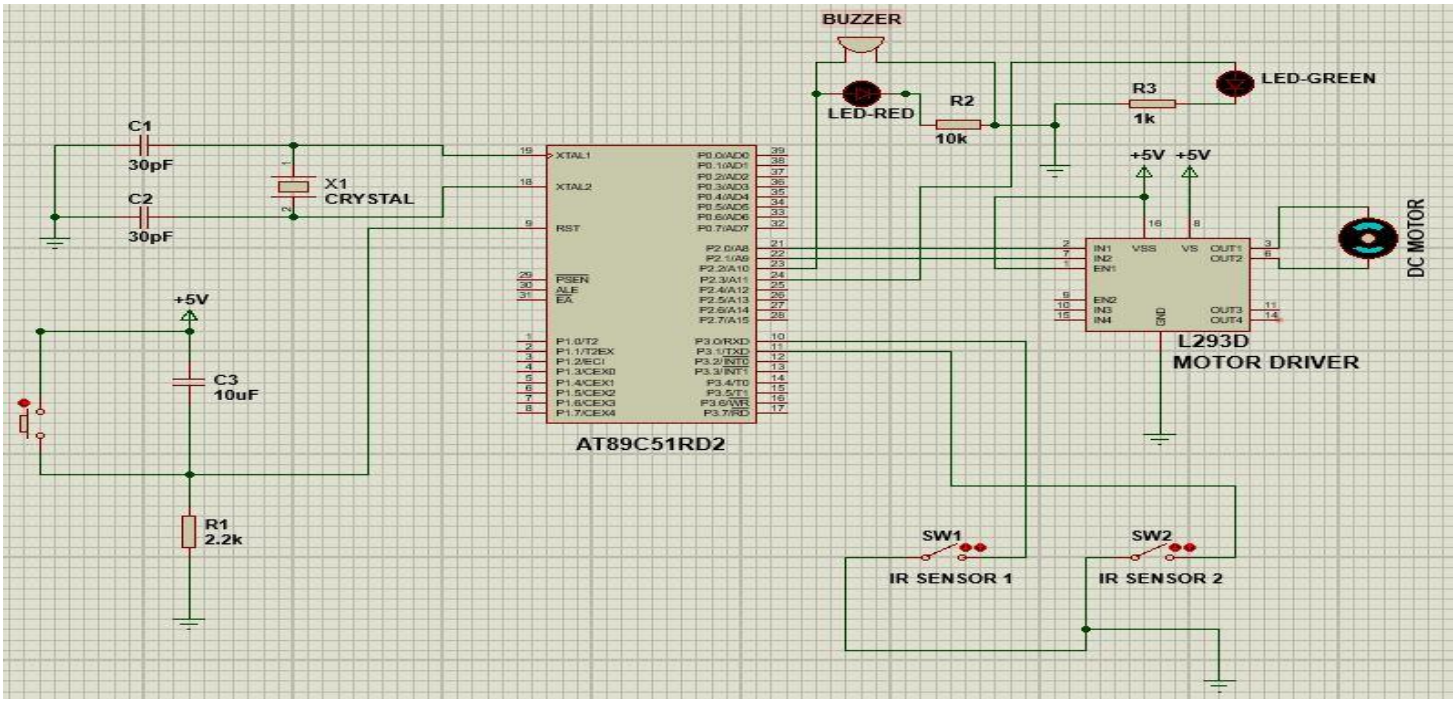
Purpose:

The purpose of this automated rail gate control system is to improve both safety and operational efficiency at railway crossings through real-time, intelligent gate control. The system automatically monitors approaching and departing trains using strategically placed IR sensors on either side of the crossing, eliminating the need for manual gate operations. When a train is detected by the first sensor, the system immediately activates warning signals (LEDs and buzzers) and closes the gates to prevent vehicles and pedestrians from crossing. The gates remain closed until the train clears both sensors, ensuring complete train passage before reopening. Advanced logic within the system addresses specific scenarios, such as trains temporarily stopping between sensors or a single engine passing through, to avoid unsafe gate operations. This reduces the likelihood of accidents, prevents traffic congestion at crossings, and increases reliability by minimizing human error, offering a more robust, automated solution for railway safety. Additionally, this system can be further enhanced by integrating IoT or remote monitoring for centralized control and data collection, enabling predictive maintenance and status monitoring.

Block Diagram:



Circuit Diagram:



Methodology:

1. Create a new project in the Keil uVision5 software, search for the microchip AT89C51RD2 and add the STARTUP.A51 file to the project.
2. Create a new empty document and write all the code is that. Save the file with extension ".asm" and add file to the Source Group
3. In Project menu, select Options for Target and go to the Output tab.
4. Check Create HEX File and click OK button.
5. Thus Translate, Build and Rebuild the project, making sure that there are no errors.
6. Create the project on Proteus Software, and implement the circuit.
7. Copy the destination of the HEX file and insert it in the Edit Component window of the AT89C51RD2 component in the Proteus Software.
8. Thus, we are ready with the simulation of the working project.

Working:

To design an automated rail gate control system that ensures safe operation of gates at a railway crossing, let's go through the detailed working process of the project. This project will rely on a sequence of IR sensors, a microcontroller for logic processing, and motors to operate the gates. Here's a breakdown of each component's role and how they work together to handle various scenarios.

System Logic and Working Mechanism

The project logic is implemented within the microcontroller in order to handle different train crossing scenarios. The goal is to ensure the gates open only when the train has entirely cleared the crossing.

Initial State

- The system starts with both IR1 and IR2 in an unblocked state
- The gate is in the **Open** position, allowing vehicles and pedestrians to cross the tracks.

Case 1: Long Train Passing Through

1. **Train Approaches and Blocks IR1**:
    - When IR1 is blocked (indicating the approaching train), the microcontroller detects this and activates the **Gate Close** sequence.
    - The warning LED and buzzer are turned on to alert nearby vehicles and pedestrians.
    - The gate closes, preventing vehicles and pedestrians from crossing the tracks.
2. **Train Continues and Blocks IR2**:
    - As the train moves forward, it blocks IR2 while still blocking IR1.
    - The microcontroller detects that both IR1 and IR2 are blocked, meaning the train is still occupying the crossing.
    - The gate remains closed.
3. **Train Clears IR1 but is Still Blocking IR2**:
    - When the train clears IR1 (meaning IR1 is now unblocked), but IR2 remains blocked, the microcontroller knows the train has partially passed but is still on the crossing.
    - The gate remains closed until the train completely clears IR2.
4. **Train Clears IR2**:
    - When the train finally clears IR2, both IR1 and IR2 are unblocked.

- The microcontroller recognizes this state as safe for reopening the gate.
- The gate opens, the LED and buzzer are turned off, and vehicles and pedestrians can safely cross.

Case 2: Train Stops Between IR1 and IR2

1. **Train Blocks IR1**:
   - As in Case 1, the microcontroller closes the gate when IR1 is blocked and activates the warning system.
2. **Train Stops Before Blocking IR2**:
   - If the train stops between IR1 and IR2, only IR1 is blocked.
   - The gate remains closed, as IR2 has not detected the train's passage yet.
3. **Train Moves and Eventually Blocks IR2**:
   - Once the train resumes moving and blocks IR2, the system continues to keep the gate closed.
4. **Train Clears Both IR1 and IR2**:
   - When the train eventually clears both sensors, the gate reopens, allowing safe passage.

Case 3: Single Engine or Partial Train Between IR1 and IR2

1. **Train Blocks IR1, then Unblocks IR1 Without Blocking IR2**:
   - A train or engine could trigger IR1 and then unblock it without reaching IR2 (e.g., if a single engine passes by or stops in between).
   - If both sensors are unblocked after an initial IR1 block, the microcontroller checks for a **block-unblock sequence** from both IR1 and IR2.
2. **Gate Remains Closed Until Full Passage is Confirmed**:
   - The gate will only open if it detects a block-unblock transition from both sensors (IR1 and IR2), confirming the train has fully exited.
   - If there is no transition from both sensors, the gate remains closed to avoid any potential danger.

3. Implementation Steps

1. **Sensor Inputs**: The IR sensors provide digital signals (HIGH or LOW) to the microcontroller, which reads these to determine whether a train is approaching, passing, or has cleared the crossing.

2. **Microcontroller Logic**: The code in the microcontroller continuously monitors IR1 and IR2. It applies conditional logic based on the block-unblock states of the sensors to decide gate operations.
3. **Motor Control for Gate Operation**: The microcontroller sends a control signal to the motor, which operates the gate (either opening or closing it).
4. **Warning System**: The LED and buzzer are activated in sync with the gate's closing. They remain active while the gate is closed and turn off when the gate reopens.

4. Simulation in Proteus Software

- **Create Circuit in Proteus**: Add the microcontroller (e.g., AT89C51RD2), IR sensors (represented as switches in simulation), motors, LEDs, and buzzer in Proteus to simulate the circuit.
- **Load HEX File**: Compile your microcontroller code in Keil uVision to generate a HEX file. Load this HEX file into the microcontroller component in Proteus.
- **Test Scenarios**: Run the simulation in Proteus and test each case (long train, train stopping, and single engine) to verify the logic and ensure safe operation of the gate.

Code:

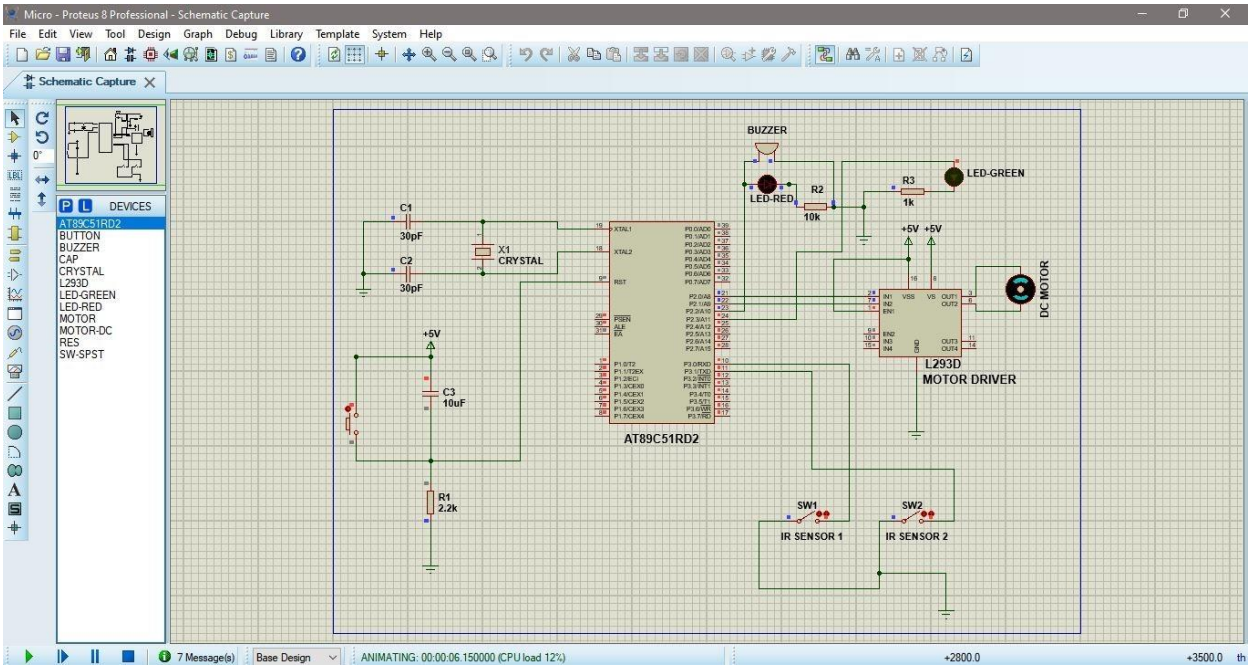
```
Start: MOV P3, #0FFH; IR connected to Port3. No block, IR□s output is logica1. ff makes P3 input port
CLR P2.0 ;connected to IN1 of L293D, clearing P2 makes it output
CLR P2.1 ;connected to IN2 of L293D, clearing P2 makes it output
CLR P2.2 ; Red traffic light and Alarm
SETB P2.3 ; Green on for all time except for the time Red is switched on
CheckIR: ACALL DELAY
                MOV R7, P3 ;IR (1) read.
                MOV A, R7
                RRC A
                JNC MotorForward; ;Jump when CY=0 i.e. IR is blocked.
                SJMP CheckIR
BACK1: MOV R7, P3 ;IR(2) read.
        MOV A, R7
        RRC A
        RRC A
        JNC SecondIRDetected
        SJMP BACK1
```

```
MotorForward:SETB P2.2 ; Red light on
                    CLR P2.3 ; Green light off
                    ACALL DELAY1
                    ACALL DELAY1
                    SETB P2.0 ;This loop closes the gate.
                    CLR P2.1
                    ACALL DELAY1
                    ACALL DELAY1
                    ACALL DELAY
                    CLR P2.0
                    SJMP BACK1
SecondIRDetected:MOV R6,P3 ;Train is cutting IR(2) and is going to pass through it.
                         MOV A, R6
                         RRC A
                         RRC A
                         JC MotorReverse_Check_FirstIR ; Check IR (1) when train crosses
IR (2)
                         SJMP SecondIRDetected ; Wait till train crosses IR (2)
MotorReverse_Check_FirstIR:MOV R5,P3
                                   MOV A,R5
                                   RRC A
                                   JNC MotorReverse_Check_FirstIR ;Keep looping
when IR (1) is cut. Go down once train crosses IR (1)
SecondIR_Check:MOV R4,P3 ;Check IR (2) after making sure that train has crossed IR(1)
                    MOV A,R4
                    RRC A
                    RRC A
                    JNC SecondIR_Check ;Keep looping when IR(2) is cut. Go down when
train crosses IR(2)
                    CLR P2.0 ;This loop opens the gate.
                    SETB P2.1
                    ACALL DELAY1
                    ACALL DELAY1
                    ACALL DELAY
                    CLR P2.1
                    CLR P2.2 ;Red light and Alarm turned off
                    SETB P2.3 ;Green light back on
                    SJMP Start
DELAY:MOV R7,#0FFH ;Delay program 1
TOP:MOV R6,#0FFH
```

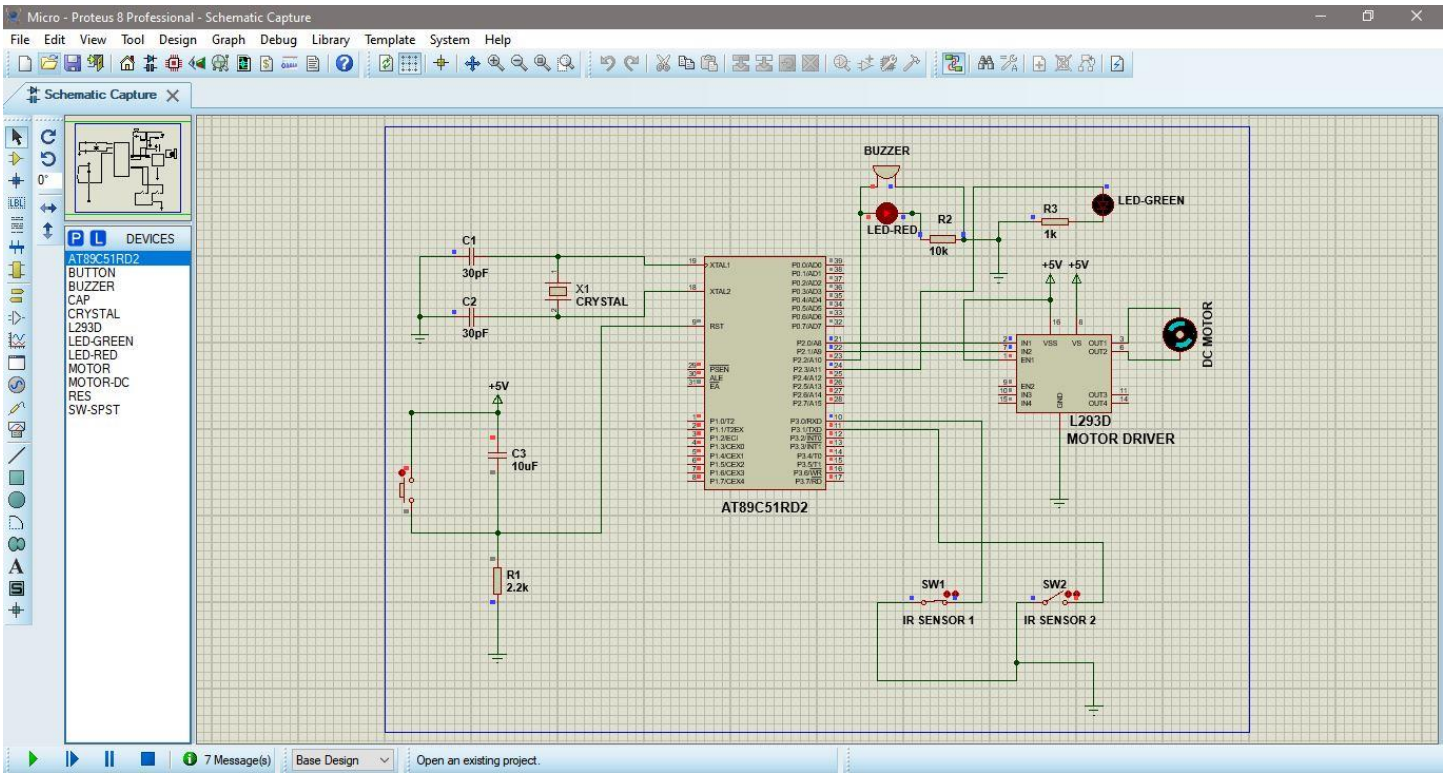
```
        MOV R5,#0FFH
BACK:DJNZ R5,BACK
UP:DJNZ R6,UP
  DJNZ R7,TOP
  RET
DELAY1:MOV R4,#14H ;Delay program 2
OneSec:MOV TMOD,#01H
          MOV TL0,#0AFH
          MOV TH0,#3CH
          SETB TR0
WAIT:JNB TF0,WAIT
        CLR TR0
        CLR TF0
        DJNZ R4,OneSec
        RET
        END
```

Result:

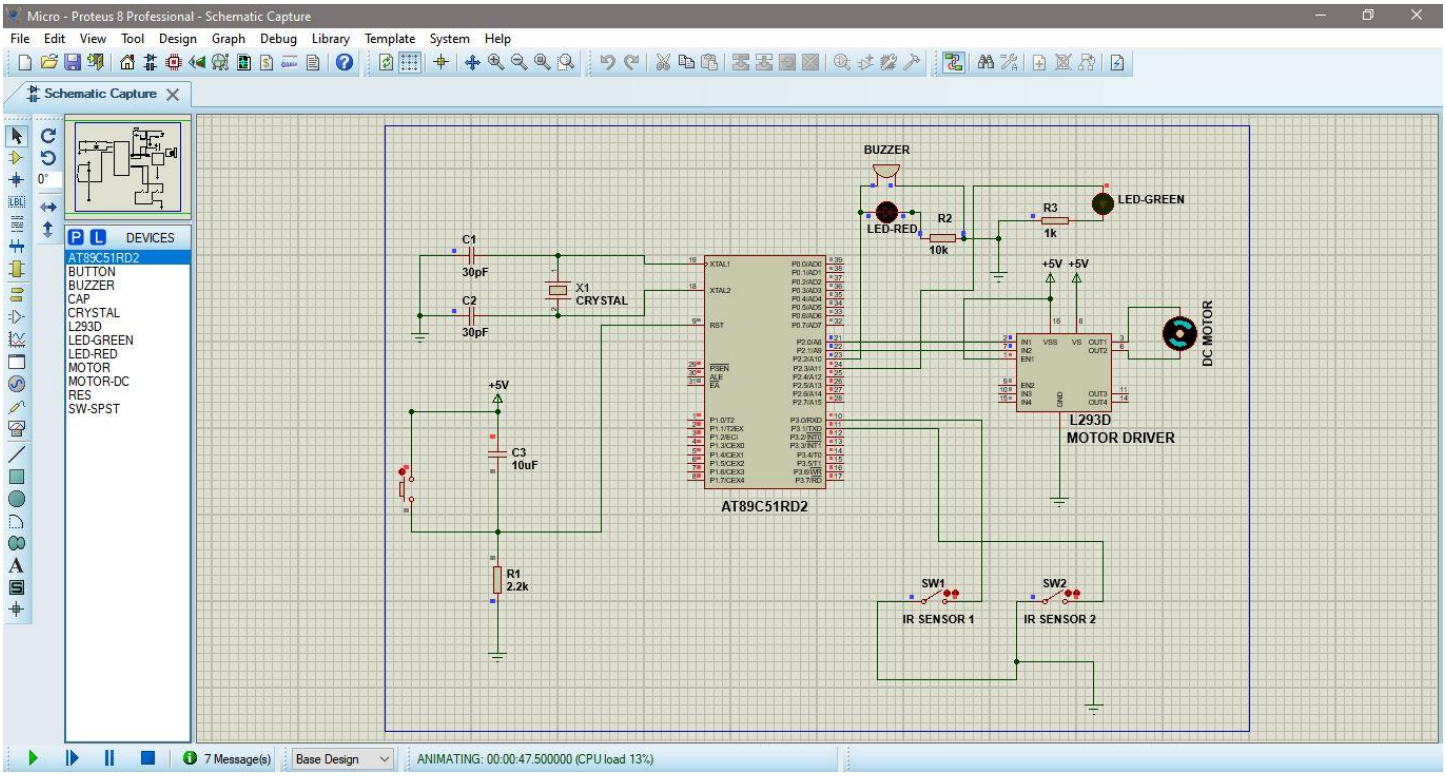
1. Green light ON, Train has not arrived yet.



2. Train arrives/is between two sensors, Gate closes/motor rotates clockwise, Green (OFF), Red (ON), Buzzer (ON)



3. Train completely passes the IR sensor 2, Gate opens (motor rotates anti-clockwise), RED(OFF), Buzzer (OFF), GREEN(ON)



Reference:

- https://www.youtube.com/watch?v=K-KO-GA_Y1M&ab_channel=FarmaanSyed
- https://www.electronicshub.org/automatic-railway-gate-controller/
- https://www.youtube.com/watch?v=pTGWT_fr_e8&ab_channel=ElectronicsHub
- https://www.researchgate.net/publication/324529019_Automatic_Railway_Gate_Control_System_Using_8051micro_Controller