Project Title: **DEVELOPMENT OF DENSITY BASED VEHICULAR TRAFFIC CONTROL SYSTEM**

**SUPERVISOR: ENGR O.F ADEBAYO**

STUDENT: GBADAMOSI ABDULSAMAD

1. **PROJECT DESCRIPTION:**

The density based vehicular traffic control system is an experimental rig setup that simulates real life traffic events. The system uses a STM32F401CC microcontroller interfaced with Ultrasonic (UR) sensors, to dynamically adjust signal timings based on real-time traffic density. Implemented at a busy intersection, the system also integrates RFID technology to prioritize emergency vehicles.

1. **COMPONENTS:**

**STM32F401CC MICROCONTROLLER:** This is the brain of the system, responsible for the operation of all the other components of the system.

**ULTRASONIC SENSORS:** This sensor detects vehicle on each lane and gives the lane with the highest number of cars a smooth passage

**RFID READER&TAGS:** It reads the RFID tags corresponding to each emergency vehicle on their respective lane, thereby prioritizing the emergency vehicle first then the normal vehicles

**Signal lights (Red Yellow Green):** These lights are strategically positioned at each approach to the junction, allowing users to clearly see the current status of the traffic signal from a distance.

**Testing Documentation**

1. Unit Testing:

All the components were assessed individually. For example, the RFID reader performance test was conducted to verify that it can detect emergency vehicle on each lane, and the ultrasonic sensors’ performance test was conducted to find vehicle density in its different lanes.

1. Integration Testing:

Thus, all components were tested in isolation, and then combined. To achieve efficient and effective communication and functionality of the RFID reader, the ultrasonic sensors, and the STM32F401CC microcontroller the components were connected and tested on the experimental rig.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case ID | Description | Input | Expected Output | Actual Output | Pass/Fail |
| TC1 | Normal operation of the system | No high numbers of normal vehicle or emergency vehicle detected on a lane | Opposite lanes should be given smooth access | Light changes to green for 8 seconds on the opposite lanes, then alternates to the other lanes | Pass |
| TC2 | Verify detection of normal vehicles | Vehicle enters lane A with the highest density | Ultrasonic sensor detects and changes light to green | Light changes to green for 10 seconds on lane A | Pass |
| TC3 | Confirm the detection of emergency vehicle | RFID tag of emergency vehicle | Traffic light turns green for emergency vehicle | Light changes to green for 6 seconds on the respective lane | Pass |
| TC4 | Simultaneous vehicle detection | Vehicles detected in two lanes | Prioritize the first detected lane, then switch to the next lane | Light changes to green for 10 seconds on the first lane, then to the next lane | Pass |

**User Guide**

**a. Installation Guide**

System Requirements:

STM32F401CC microcontroller

RFID reader (RC522)

Ultrasonic sensors (HCSR204)

Light Emitting Diodes for mimicking traffic signals

Power supply (5V)

Programming computer with Visual Studio Code and PlatformIO IDE.

Steps:

Assemble the hardware: Mount the ultrasonic sensors and RFID reader onto the experimental rig of the microcontroller.

Software: Control logic is coded in C programming language, coding and testing was done using an IDE known as Platform IO whereas development was done using Visual Studio code

**b. Usage Instructions**

Step 1: Firstly, make sure that the system switch is on and that the sensors are placed on the simulated road layout properly.

Step 2: Detection of normal vehicles: Depending on the density of the approaching vehicles, Place a vehicle at the end of any of the lanes. the system shall automatically change the traffic signals on the road

Step 3: Emergency vehicles: This goes with an RFID reader and tags for each lane (A, B, C, D) that makes the traffic signal to switch on the green to ensure the passage of an approaching emergency vehicle.

**Troubleshooting**

Issue: Traffic light not changing.

Solution: Check the connections of the sensors and the microcontroller and also make sure that it is well supplied with power.

Issue: Emergency vehicle not prioritized.

Solution: Look at the RFID reader and confirm that the tag is within the range for it to capture (max 5 cm).

Note: Always make sure the signal lights reset before placing the rfid tag close to the reader

**Additional Information**

Follow these steps to use PlatformIO on Visual Studio Code (VS Code) to program the STM32F401CC:

**Step 1: Install Visual Studio Code**

**Step 2: Install PlatformIO Extension**

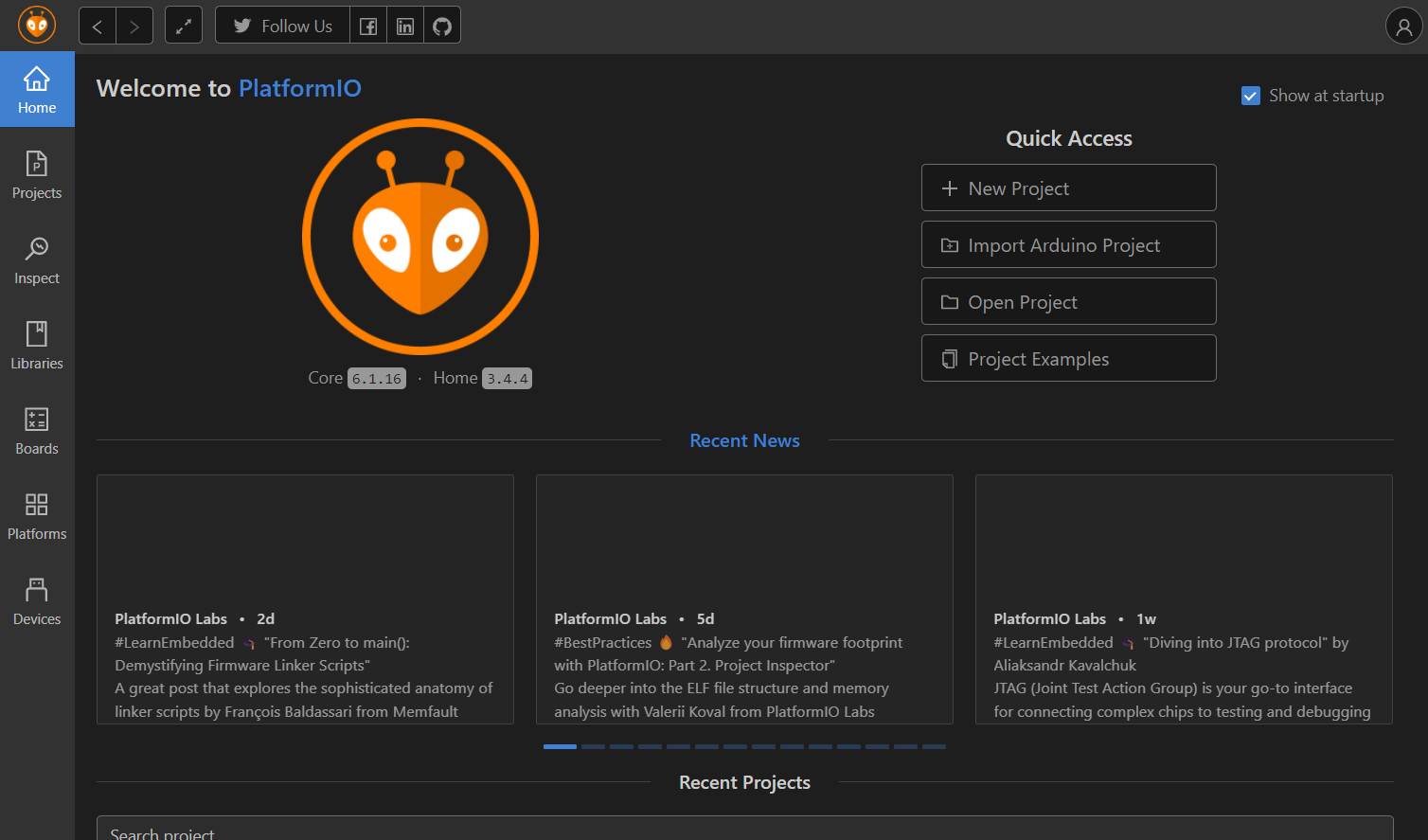
1. Open Visual Studio Code.

2. On the left sidebar, click on the Extensions icon

3. In the search bar at the top, type PlatformIO IDE.

4. Click Install next to the PlatformIO IDE extension.

Once the installation is complete, you should see a PlatformIO icon (a small alien head) on the left sidebar.



**Step 3: Set Up a New Project in PlatformIO**

1. Click on the PlatformIO icon on the left sidebar to open the PlatformIO Home.

2. Click New Project.

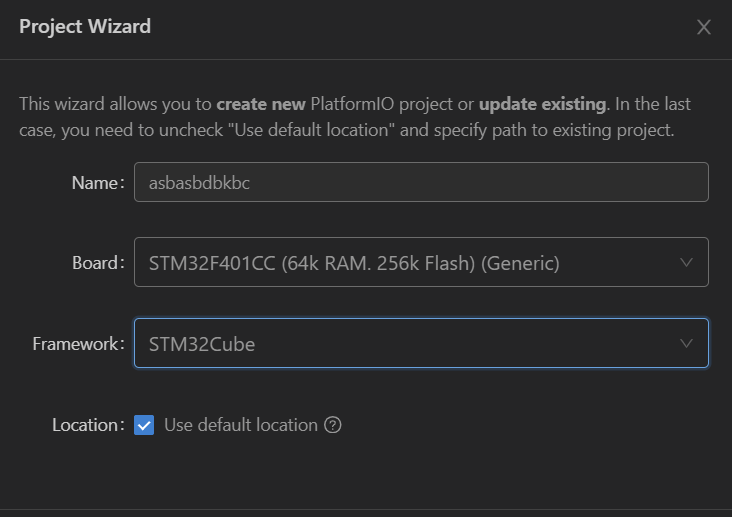
3. In the Project Wizard window:

-Project Name: Give the project a name.

- Board: Select the microcontroller board (STM32F401CC).

- Framework: Choose the preferred development framework (STM32Cube for STM32 boards).

- Location: Specify where you want to save the project.



4. Click Finish. PlatformIO will automatically create the project structure and install the necessary libraries.

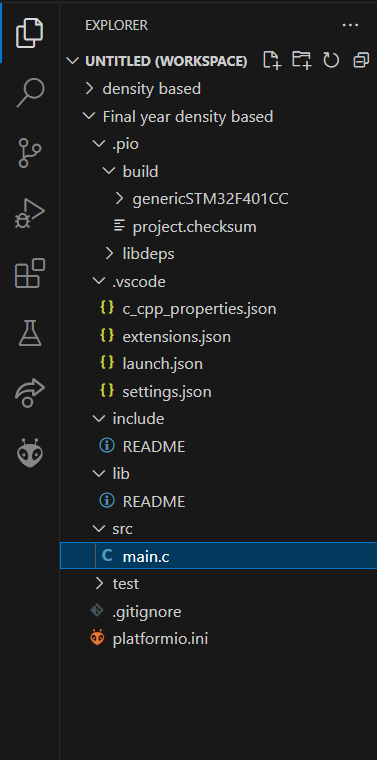
**Step 4: Explore the Project Structure**

PlatformIO organizes the project into folders:

- src: This is where the main source code goes (usually a `.cpp` or `.c` file).

- lib: A folder to place any external libraries needed.

- platformio.ini: The configuration file where you can manage settings for your project, including board type, framework, and libraries.



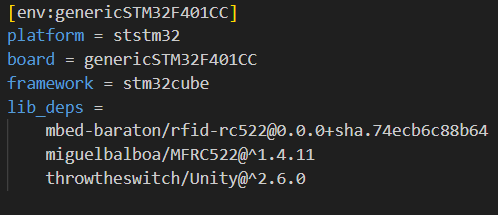
**Step 5: Write/Copy the Code**

1. In the src folder, open `main.cpp` or create a new `.cpp` or `.c` file.

2. Write the microcontroller code as usual or copy it from another editor.

**Step 6: Configure platformio.ini**

Open the `platformio.ini` file to configure your project. It should look like this:



- platform: Specifies the microcontroller platform (for STM32, it's `ststm32`).

- board: Specifies the board model.

- framework: Choose STM32Cube.

- upload\_protocol: Specifies how to upload the code (using stlink, serial, etc.)

**Step 7: Compile and Upload the Code**

1. Compile the Code:

- On the bottom toolbar, click the checkmark icon to build your project.

- PlatformIO will compile the code and show errors or success in the terminal window.

2. Upload the Code:

- Connect the STM32 board to your computer using a USB.

- Click the right arrow icon on the bottom toolbar to upload the code to the microcontroller.

- PlatformIO will flash the code onto the board, and you should see your program running.

**Step 8: Adding Libraries**

1. Click the PlatformIO icon on the sidebar and navigate to Libraries.

2. Search for any external libraries you need (e.g., RFID, sensor libraries).

3. Install the library, and PlatformIO will automatically add it to your project.

