Department of Computer Engineering
Faculty of Engineering
University of Peradeniya

Smart Traffic Light System

Project Report

Group 06

E/18/098 Fernando K.A.I.

E/18/100 Fernando K.N.A.

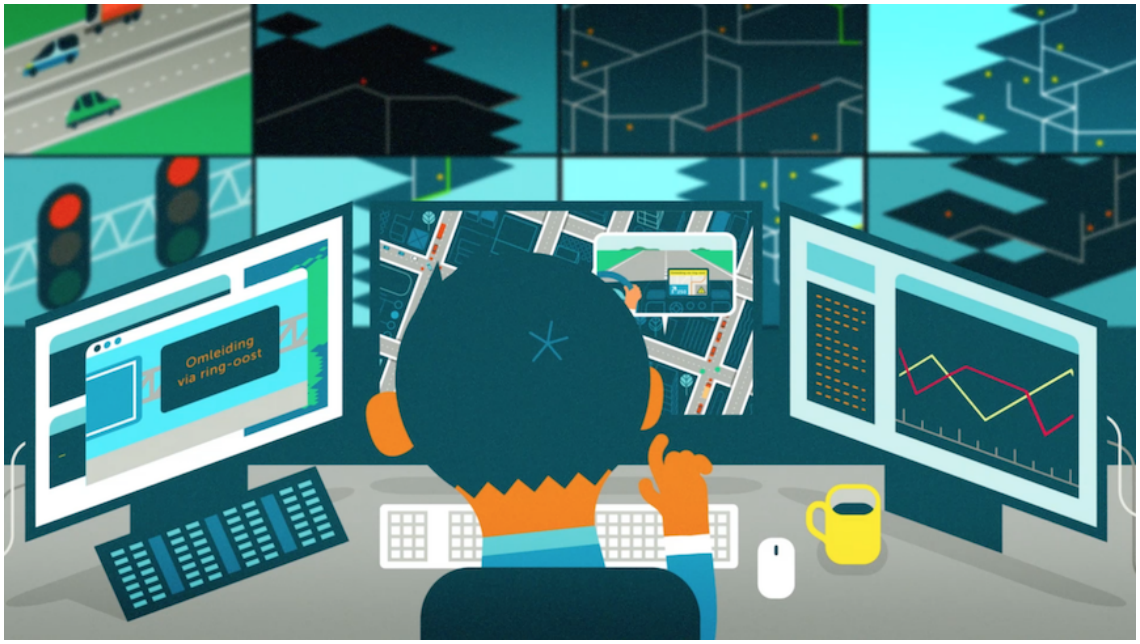
E/18/155 Jayasundara J.W.K.R.B.

TABLE OF CONTENTS

SYSTEM OVERVIEW

Introduction

We have developed an IoT-based traffic management system that can effectively allocate time for vehicles in a lane based on the number of vehicles present. Traffic congestion is a common issue faced by most urban areas, causing delays and inconvenience for commuters. The current traffic management systems use a fixed time-based approach, which often leads to traffic congestion during peak hours and empty lanes during off-peak hours. To address this issue, an intelligent traffic management system is needed that can dynamically allocate time for each lane based on the real-time traffic flow. The system we have developed is using IoT-based sensors to detect the number of vehicles in each lane and communicate with the central controller to allocate time accordingly. This system will help to reduce traffic congestion, improve traffic flow, and reduce waiting times for commuters. The project aims to design and develop a prototype system that can be easily integrated with the existing traffic management infrastructure.



In addition to the ultrasonic sensors and microcontrollers, SCADA (Supervisory Control and Data Acquisition) is an integral component of the smart road traffic control system project. SCADA software is used to collect and process real-time traffic data as well as manual control of the traffic control system.

Moreover, the SCADA system is integrated with the traffic signal system, allowing for automatic adjustments of traffic lights at intersections based on the traffic flow data received. As an extension to the project The SCADA system can also be used to provide alerts to drivers and commuters about traffic conditions, suggesting alternative routes, and other relevant information. This is not the primary goal of this project.

The smart road traffic control system project, with the integration of SCADA, is expected to have a significant impact on road traffic management.

HARDWARE IMPLEMENTATION

Required Components

- ESP32 Microcontroller

We have used ESP32 as the microcontroller for our project. ESP32 offers built-in features and connectivity options, such as support for Bluetooth, Wi-Fi, and Ethernet, which was useful to intergrate the system with our SCADA system. This microcontroller is capable of handling the tasks of the smart road traffic control system project, which requires handling traffic data collected by multiple ultrasonic sensors, controlling the LEDs accordingly and sending the captured data to the SCADA system through MQTT.

ESP32 will be powered using a USB Phone charger for the demo. A phone charger can output 1A usually. ESP32 will use about 240mA when it's active. 1A - 240mA = 760mA can be used by the other devices in this system.

- HCSR04 UltraSonic Sensor

The HC-SR04 is an ultrasonic sensor that is a common sensor that is used for distance measurement in robotics and automation applications. It works by emitting high frequency sound waves, and then detecting the time it takes for the sound waves to bounce back after hitting an object. Based on this time delay, the sensor can calculate the distance to the object.

The HC-SR04 is relatively low-cost and easy to use, making it a suitable sensor for projects like this. It requires a power supply of 5V DC and can be interfaced with microcontrollers such as ESP32 using a simple 4-pin interface.

This sensor is used to get an approximation of how many vehicles are there in queue to cross the road using the color-lights.

The distance is provided and can be read by the microcontroller connected to the sensor which then will be converted to an approximation of the number of vehicles.

One sensor uses 15mA when it's active. As we have 2 of these. Both of them use 30mA when they are active. 760mA - 30mA = 730mA is left for the other devices.

- LED

For simulating traffic signals in our project, small and low-power 5mm LEDs were used. These LEDs are available in red, yellow, and green colors, which correspond to the colors used in traffic lights. The LEDs were connected to the microcontroller through resistors to limit the current and prevent damage to the LEDs. The resistor value calculations are shown below. The microcontroller then controls the LEDs to turn on and off and simulate the different colors of the traffic signals.

LED's act as the actuators which control the vehicle traffic based on the sensor measurements and if necessary, based on the SCADA system as well.

Red LED Resistor

$V = IR$

$3.3 - 3.1 = 30\text{mA}*R$

$R = 6.66\ \Omega$ or higher

Green LED Resistor

$V = IR$

$3.3 - 2.1 = 20\text{mA}*R$

$R = 60\ \Omega$ or higher

Orange LED Resistor

$V = IR$

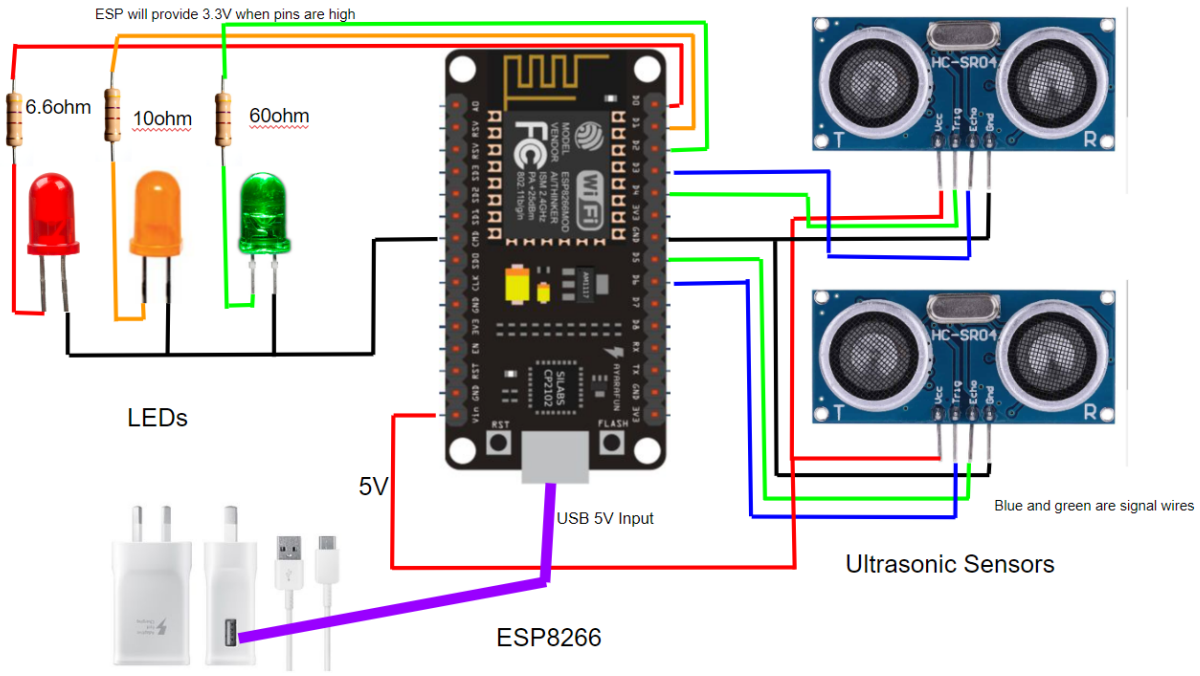
$3.3 - 3.0 = 30\text{mA}*R$

$R = 10\ \Omega$ or higher

The 3 LEDs will consume a maximum of 80mA if they are lit simultaneously. All three leds will never be lit at once. At most two leds will be lit at any given time. Therefore, the maximum current usage of the LEDs will be about 60mA. 730mA - 60mA = 670mA can be used by the other devices in the system.

If we use a 5V 1A USB phone charger. It will be loaded up to 330mA. That is 33% of its rated value and it should be able to supply this current all day.

Circuit Design



SOFTWARE IMPLEMENTATION

Firmware

The code utilized several libraries, including WiFi and PubSubClient, to establish a seamless connection between the smart traffic light system and the SCADA via MQTT. This connection facilitated real-time data exchange and remote control of the traffic lights. To ensure accurate timekeeping, the code incorporated time synchronization functionality by connecting to an NTP server and adjusting the local time accordingly.

The code included functions to retrieve readings from the ultrasonic sensors. The getUltrasonic1Reading() and getUltrasonic2Reading() functions activate the respective sensors, and calculate the vehicle status in the queue. These sensor readings served as crucial input for the traffic light control logic.

In terms of traffic light control, the code employed three LEDs representing the standard red, orange, and green signals. Dedicated functions, such as TurnOnRED(), TurnOffRED(), TurnOnORANGE(), and so on, were implemented to facilitate the control of individual LEDs. The currentLED variable was used to track the currently active LED color.

The code subscribed to the "group7/control" MQTT topic, enabling it to receive control commands for the traffic lights. Upon receiving a message, the code processed its content to determine the desired state of the traffic lights. Messages with the payloads "Red," "Orange," or "Green" triggered the corresponding LED control functions to change the traffic light color accordingly. This allowed for dynamic and remote control of the traffic lights based on real-time traffic conditions.

To provide insights into traffic patterns and congestion levels, the code periodically publishes sensor data to the MQTT broker, which is used to monitor using the SCADA system. The sensor readings, along with the current LED color and timestamp, were bundled into an MQTT message and transmitted to the

"group7/sensors" topic. This data could be collected and visualized by the SCADA for further processing, and decision-making.

Complete Code Implementation : https://github.com/cepdnaclk/e18-co326-Smart-Traffic-Light-System

MQTT

MQTT (Message Queuing Telemetry Transport) was utilized to facilitate communication between SCADA system and hardware node and control the traffic lights. Specifically, we connected to the MQTT broker hosted on test.mosquitto.org. This broker is a publicly available Eclipse Mosquitto MQTT server that provides a lightweight and efficient publish/subscribe model for machine-to-machine messaging.

The MQTT broker operates on various ports to support different configurations. The available ports include 1883 for unencrypted and unauthenticated MQTT communication, 1884 for unencrypted but authenticated MQTT, 8883 for encrypted MQTT without authentication, and 8884 for encrypted MQTT requiring client certificate authentication. Additionally, there are ports dedicated to MQTT over WebSockets, including unencrypted and encrypted variants.

Authentication and topic access on the broker are managed through usernames and passwords. Unauthenticated clients have publishing access to all topics, except for the literal "#" topic. Clients with authentication can have different levels of access, such as read/write, read-only, or write-only permissions on the topic hierarchy.

MQTT broker on test.mosquitto.org is primarily intended for testing purposes. As it runs on low-power hardware, occasional restarts and slower response times can be expected.

While using the MQTT broker for our project, we took into consideration its limitations and designed our client application to handle potential broker restarts and fluctuations in performance.

Database

In our project, we integrated MongoDB as our database solution to store and manage the data received from the hardware node through MQTT. Specifically, we utilized MongoDB Atlas, a fully managed cloud database service, to ensure scalability, reliability, and ease of administration.

The data structure stored in our MongoDB database consists of various fields representing the information collected from the hardware node. Each record in the database is identified by a unique ObjectId, and the relevant fields include:

deviceName: This field represents the name of the traffic lights device that generated the data.

timeMeasured: It captures the timestamp when the data was measured, in this case, "June 29, 2023, 11:58:11".

distanceSensor1: This field stores the value recorded by the first distance sensor, which in our case is "154".

distanceSensor2: It holds the value obtained from the second distance sensor, which is "195" in this instance.

currentColor: This field indicates the current color state of the traffic lights, with "81" representing a specific color code.

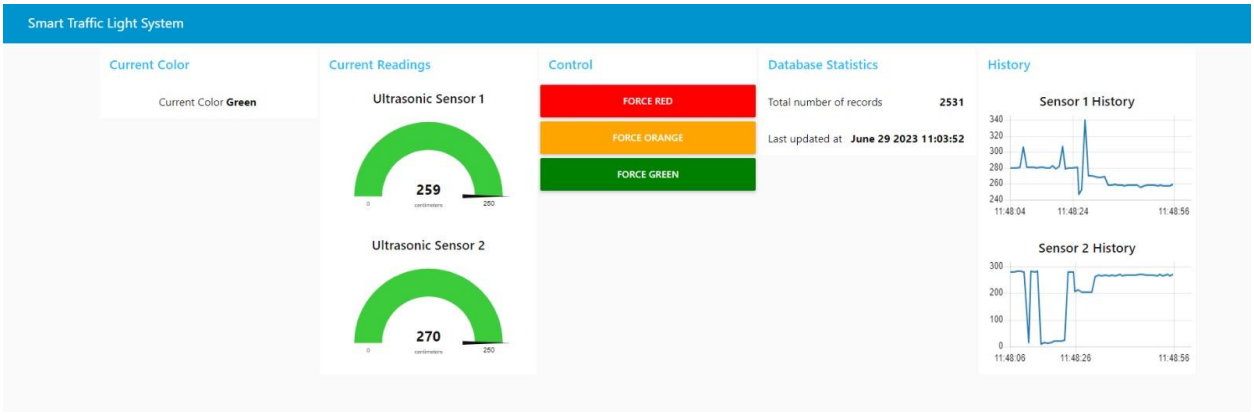
NodeRedReceivedTime: This field signifies the time when the data was received by the Node-RED platform, with the timestamp being "2023-06-29T06:28:12.274+00:00".

SCADA System

To enhance the monitoring and control capabilities of our smart traffic control system, we have successfully implemented a SCADA (Supervisory Control and Data Acquisition) system using Node-RED. This integration enables us to effectively manage and supervise the entire traffic control infrastructure while providing real-time insights into the system's performance.

With our SCADA system in place, we gain the ability to exert control over the traffic control behavior when required. Moreover, our SCADA system empowers us to monitor the current status of the traffic control system comprehensively. By integrating with the sensor readings obtained from the hardware nodes, Node-RED provides us with real-time data visualization and analysis.

In addition to its monitoring and control features, Node-RED offers seamless integration with other systems and services. We have utilized this capability to integrate our SCADA system with the MQTT broker and MongoDB Atlas database. This integration ensures a seamless flow of data between the traffic control system, the SCADA system, and the database, enabling efficient data storage, retrieval, and analysis.



PROTOCOLS

WIFI

In our smart traffic control system, we have employed WiFi as the primary protocol to establish internet connectivity for the microcontroller units. WiFi (Wireless Fidelity) offers a reliable and efficient means of wireless communication, enabling seamless data exchange between the microcontrollers and SCADA system via the internet.

WiFi offers several advantages for microcontroller-based internet connectivity. It provides high data transfer rates, enabling efficient communication between the microcontroller and internet services. WiFi networks are widely available, making it convenient to establish connections in various environments. The protocol also supports encryption and security mechanisms, ensuring the integrity and confidentiality of data transmitted over the wireless connection.

REFERENCES

1) ElecFreaks (2011). Ultrasonic Ranging Module HC -SR04. [online] Available at: https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pd
2) Red LED datasheet https://panda-bg.com/datasheet/802-090011-LED-3mm-L-314SRT-SUPER-RED-transparent.pdf
3) Green LED datasheet https://www.farnell.com/datasheets/1671514.pdf
4) Orange LED datasheet https://www.farnell.com/datasheets/1660998.pdf
5) Last Minute Engineers. (2018). Insight Into ESP32 Sleep Modes & Their Power Consumption. [online] Available at: https://lastminuteengineers.com/esp32-sleep-modes-power-consumption/#:~:text=ESP32%20Active%20Mode
6) Mohammed, Bahaa Kareem. “Design and implement a smart traffic light controlled by the internet of things.” Periodicals of Engineering and Natural Sciences (PEN), http://pen.ius.edu.ba/index.php/pen/article/viewFile/2351/973
7) TRAFFIC CONGESTION CONTROL SYSTEM USING ADVANCED IOT IRJET, https://www.irjet.net/archives/V8/i7/IRJET-V8I7735.pdf
8) Traffic Light System using magicbit-(ESP-32).” Hackster.io, 24 August 2022, https://www.hackster.io/magicbit0/traffic-light-system-using-magicbit-esp-32-0cbe48