ITCS447: Embedded Systems and Internet of Things

Railway guard MkII

Servo motor + ultrasonic sensor

Submit to

Asst. Prof. Dr. Thitinan Tantidham

Presented by

Chancheep	Mahacharoensuk	6288092	section 1
Kantapong	Matangkarat	6288160	section 1

Faculty of Information and Communication Technology

Mahidol University

2022

Table of contents

Servo motor + ultrasonic sensor 1

Introduction 3

COMPONENTS AND SUPPLIES (From start) 4

Why we chose this topic. 4

How do we take advantage of it? 6

Flowchart(1st version) 7

What we did (step-by-step visualization). 8

ESP 32 Version. 8

- Step 1: Start in Fritzing 8
- Step 2: Connect Ultrasonic sensor 8
- Step 3: Connect Ultrasonic sensor with Led light 8
- Step 4: Connect Ultrasonic sensor and Led light to switch 9
- Step 5: Coding 9
- Step 6: Upload and simulate. 10

Fritzing ESP 32. 11

Result. 12

Project 2 13

What we do. 13

What's useful 14

Flowchart (2th version) 15

Steps to do 16

Challenging 16

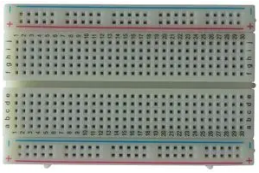
Add-on 16

Video URL. 20

Introduction

This project has something exciting inside this. You can use proximity to actually control some kind of mechanical work. I want you guys to look further into this project. I have written this project in accordance with the thought that we can make Motor do some mechanical work. We will show you how to modify and bring knowledge from various labs to make our project something unique, such as adding equipment to the board to make more functions.

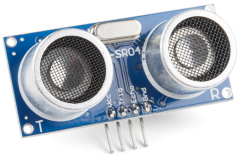
COMPONENTS AND SUPPLIES (From start)



Breadboard (generic)

Female/Female Jumper Wires

Male/Female Jumper Wires



Ultrasonic Sensor - HC-SR04 (Generic)

SG90 Micro-servo motor

Why we chose this topic.

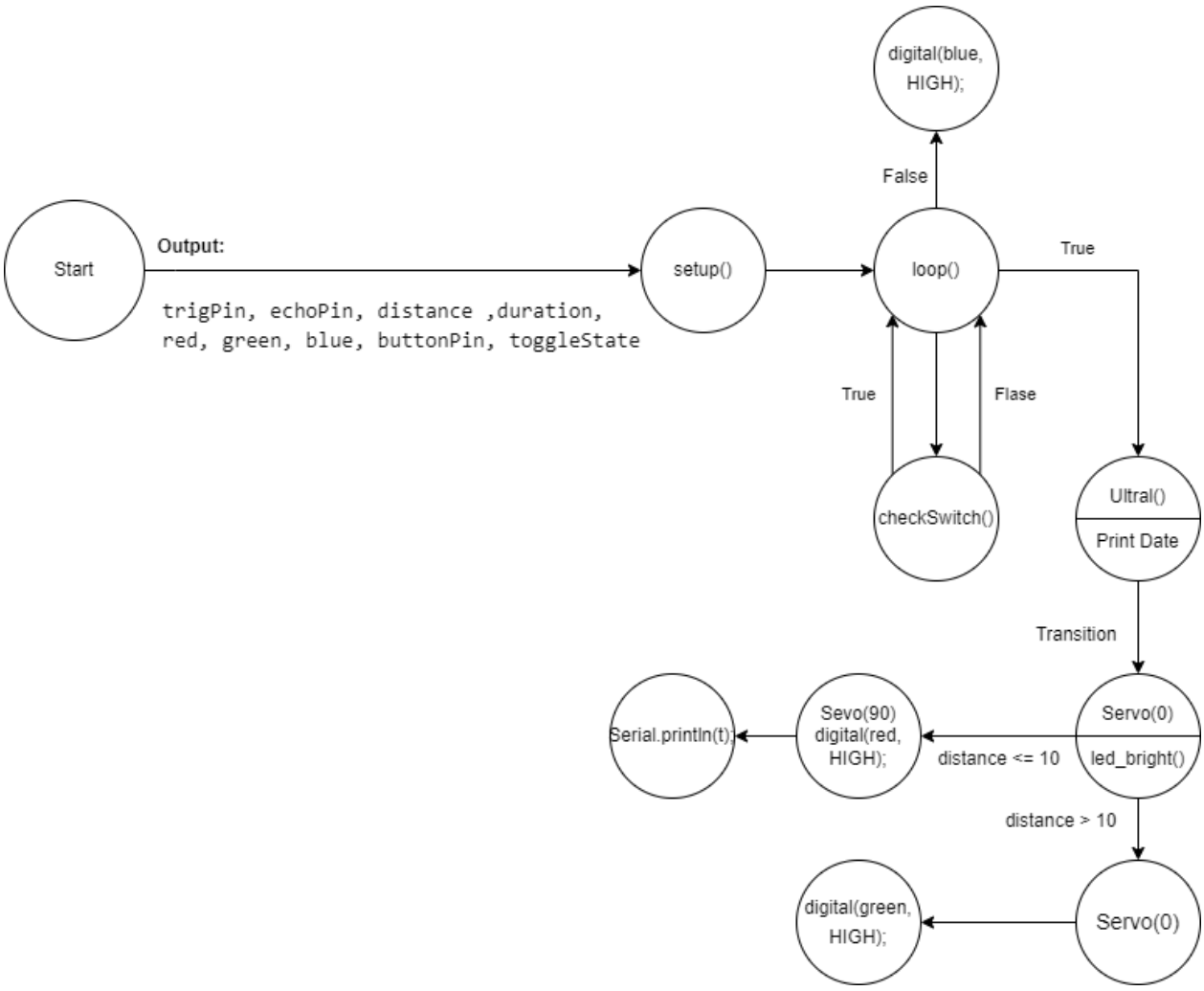
An ultrasonic sensor is a device that can measure the distance to an object by the use of sound waves while a servo motor is built for precision control of the angular or linear position, velocity, and acceleration. We were immediately drawn to the topic, probably because we knew what we were going to do with it after we got the topic. Perhaps we will be able to add or mix up the equipment that is assigned to it and the equipment that we already have experience with.

How do we take advantage of it?

From the board that we have connected The device includes an ultrasonic rangefinder, servo motor, RGB light, and RTC. All of these may not be useful in real life, but they could be further developed in the future. For example, if we have a servo motor and an ultrasonic rangefinder, it may be used as a gate or barrier for your little one when he is not yet mature enough to take care of himself. Whenever something comes close to your rangefinder it will cause the servo motor to run and come down to block it. There is also an RGB light to show how close the distance is, such as if it's red, it's closer than 10 centimeters, or if it's too far away, it's green. The power switch also functions as the name suggests, turning the device on and off when you don't need it. If you press it, it will show a blue light. The last one is RTC to show the time to keep when something makes the lights red or blue.

Initiating this invention, many people, especially children and the elderly may not be careful around them. Thus, it makes them encounter some serious situations. Such as a railroad barricade, normally, only functions as a barrier, but if it has a kid who is smaller than a stick, or an elderly person who doesn't notice walking past, perhaps the IoT defense will be able to meet these people.

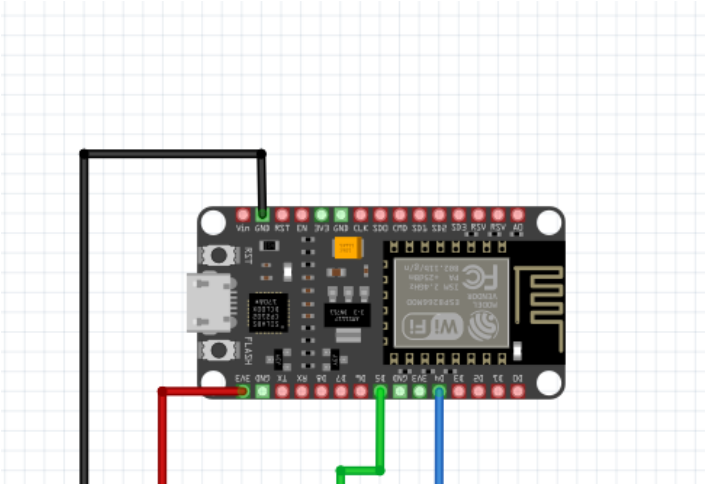
Flowchart(1st version)



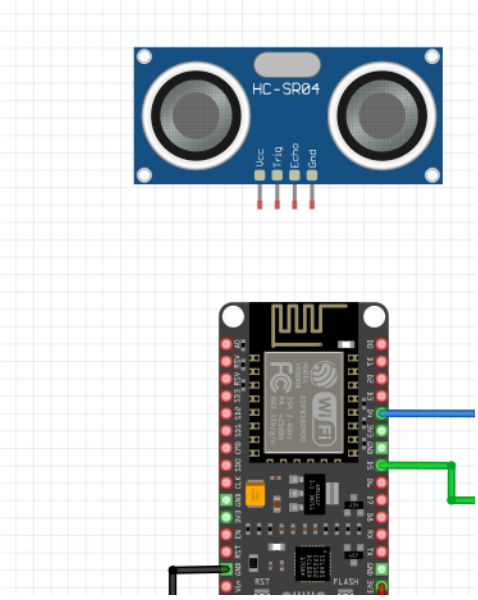
What we did (step-by-step visualization).

ESP 32 Version.

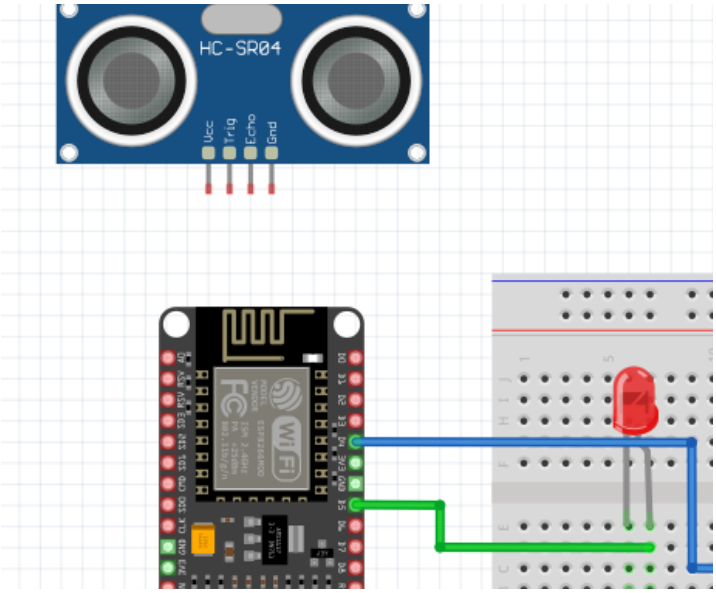
Step 1: Start in Fritzing



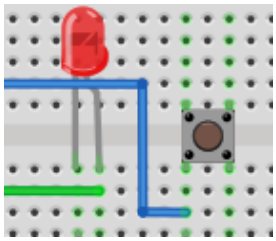
Step 2: Connect Ultrasonic sensor



Step 3: Connect Ultrasonic sensor with Led light

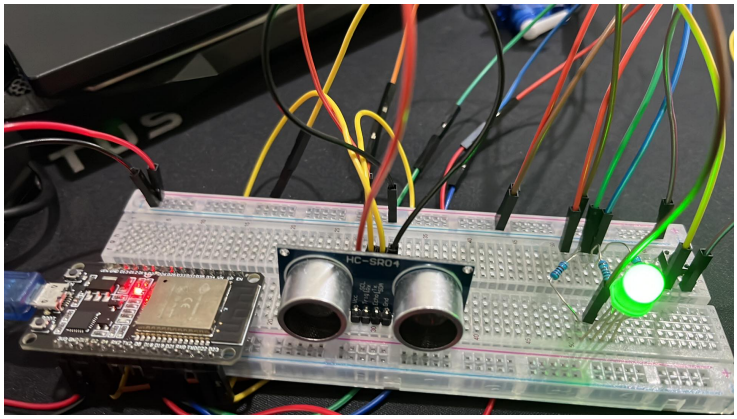


Step 4: Connect Ultrasonic sensor and Led light to switch

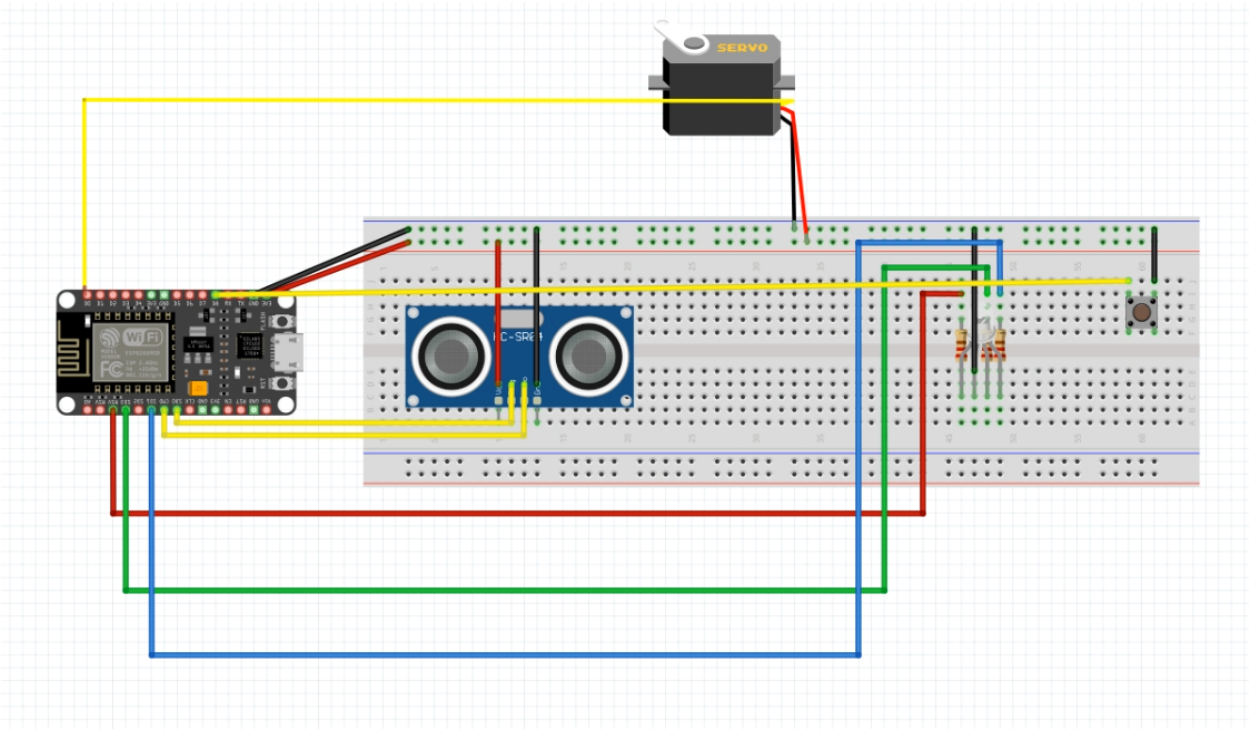


Step 5: Coding

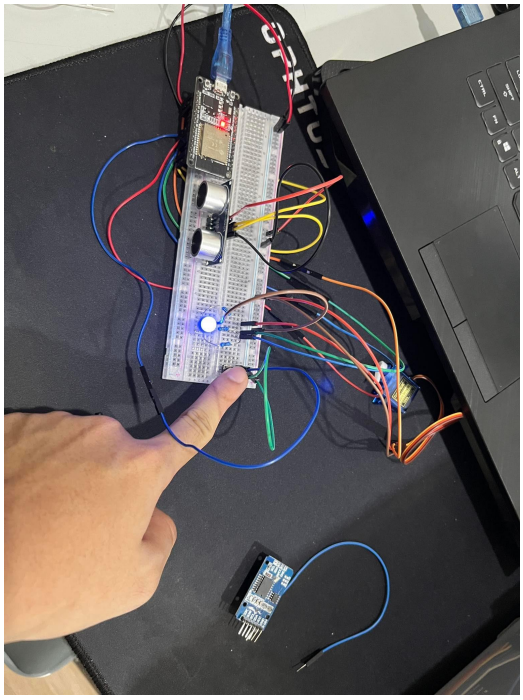
Step 6: Upload and simulate.

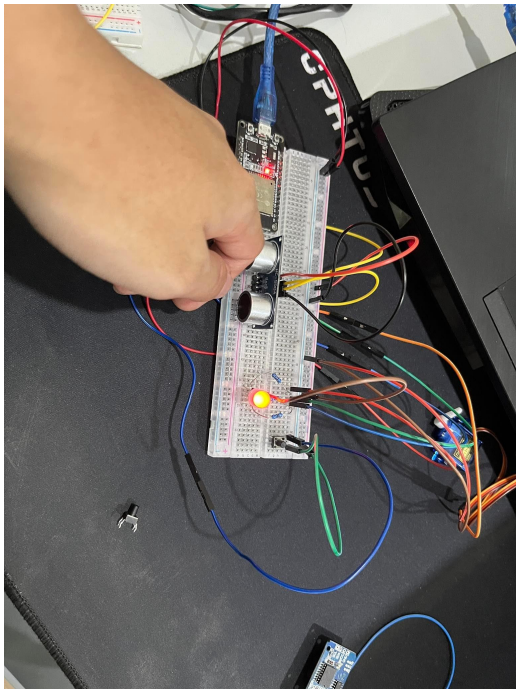


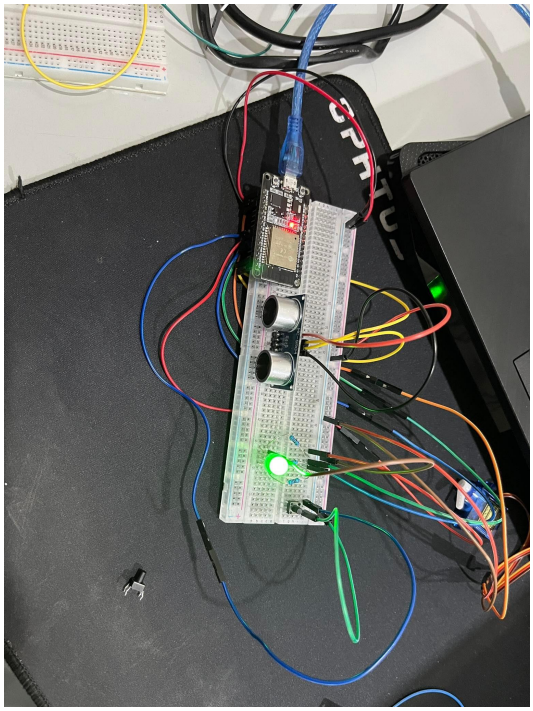
Fritzing ESP 32.



Result.







Project 2

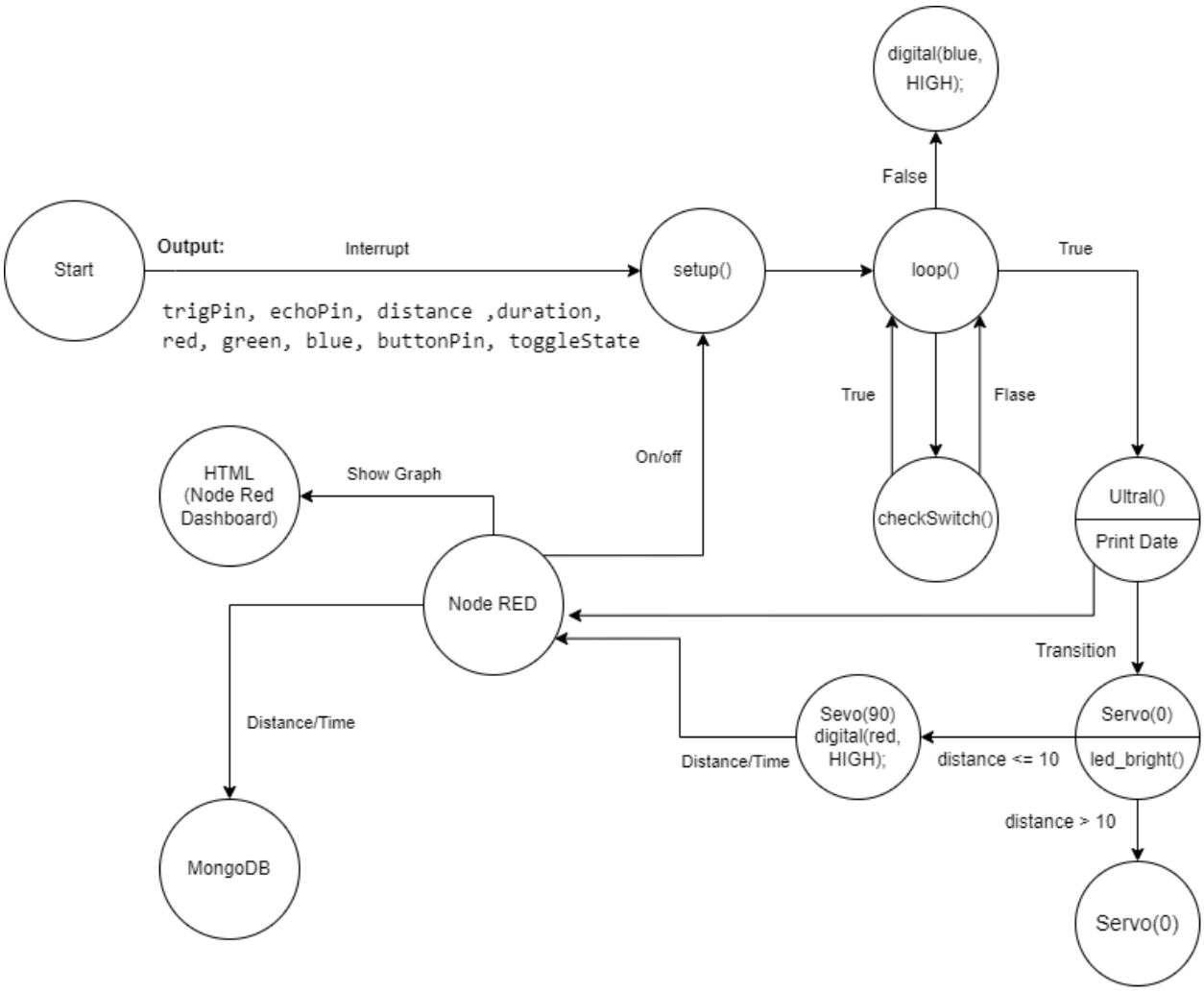
What we do.

What we did was take the system from Project 1 that was done in the first half of the semester. Continuing with the integration of mongoDB database and control over the Node-RED system. In order to make it look closer to being IOT in everyday use even more.

What's useful

As described in the first project, utilizing what we invented may not be possible now. It's just a simulated device. If talking about the benefits that can be gained from it, it would be like I said above, that is, bringing it as a device to block or obstruct when something is closer than we set. Connecting this device to the Node RED adds functionality to it so that it can be controlled via a web page. A graph showing time per distance can also be viewed on the web page.

Flowchart (2th version)



Steps to do

First, we have to connect the ESP32 to the laptop and after that we have to open node-RED and MongoDB in Ubuntu(VM). After we do all this thing, upload the code and here we are. You can watch the result and control the system on Node-RED dashboard and you can watch the data in console of the VM

Challenging

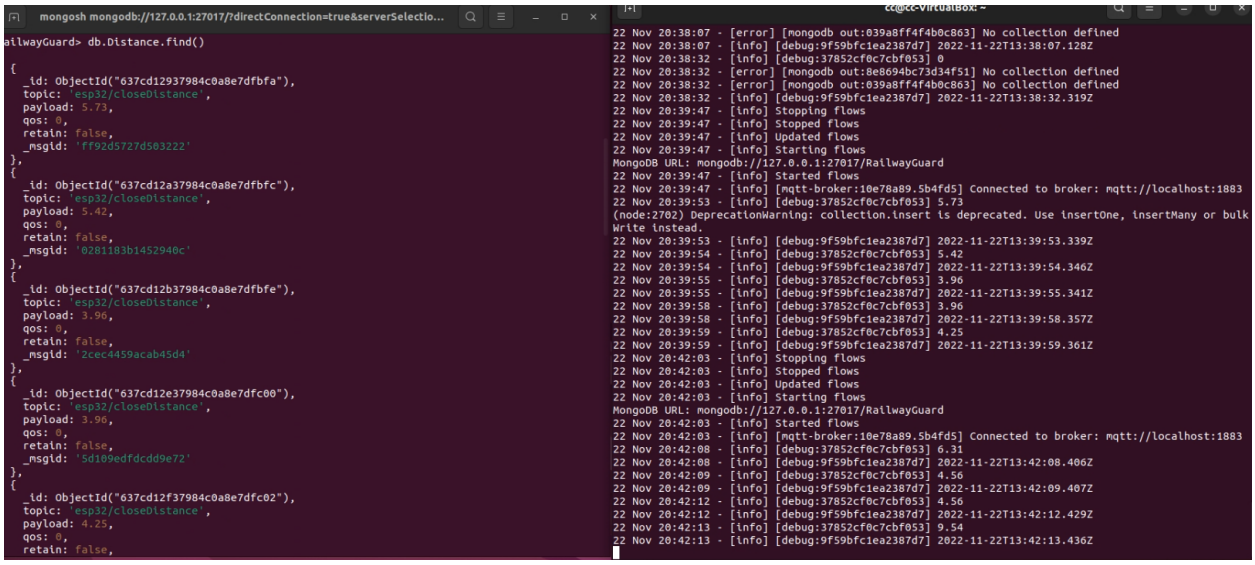
The challenge of this work is to add elements from the system that was created in the project1. Since we don't plan on bringing it forward to the next project, it's necessary to update the code in many parts to be in line with this round of development. Most importantly, this time we extract the time data and store it in the database via the function in node-RED so we don't have the same RTC problem as last time.

Add-on

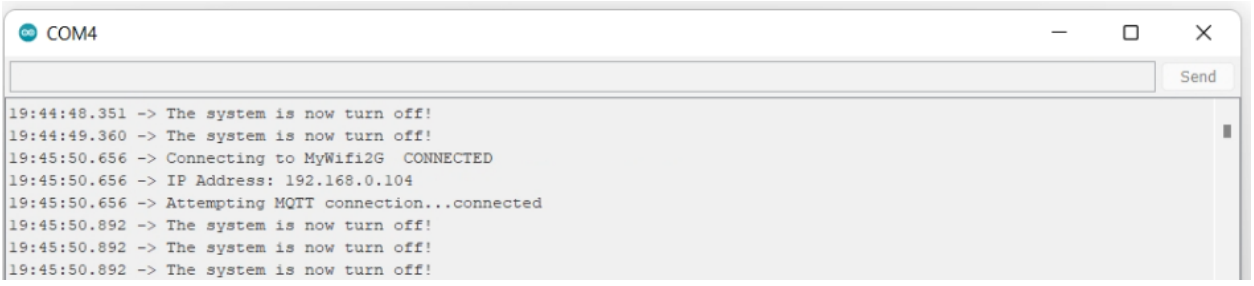
We use everything from the library in arduino or palette in node-RED like the standard in the class eg: node_RED dashboard, WiFi.h, PubSubClient.

Output capture

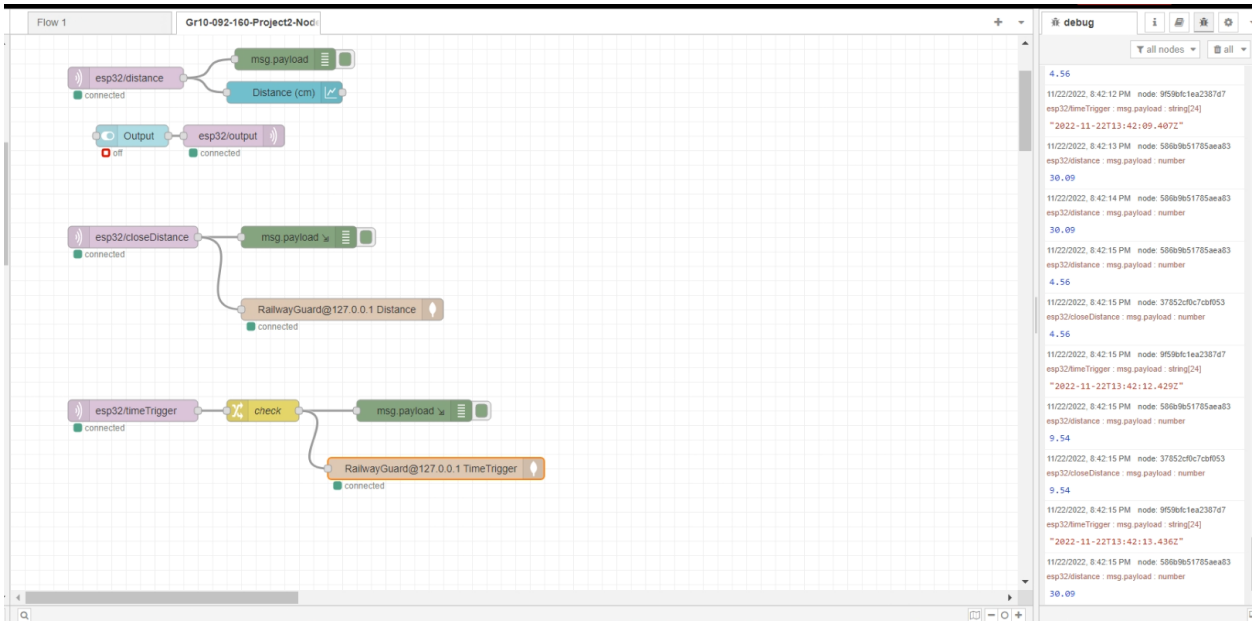
```
test> show dbs
Activity13db  316.00 KiB
RailwayGuard   16.00 KiB
admin          40.00 KiB
config         60.00 KiB
local          72.00 KiB
test> use RailwayGuard
switched to db RailwayGuard
RailwayGuard> db.Distance.find()
```



```
{
  _id: ObjectId("637cd1b037984c0a8e7dfc05"),
  topic: 'esp32/timeTrigger',
  payload: '2022-11-22T13:42:08.406Z',
  qos: 0,
  retain: false,
  _msgid: 'f629fabde8bf45bb'
},
{
  _id: ObjectId("637cd1b137984c0a8e7dfc07"),
  topic: 'esp32/timeTrigger',
  payload: '2022-11-22T13:42:09.407Z',
  qos: 0,
  retain: false,
  _msgid: 'c8ffd0fac236780d'
},
{
  _id: ObjectId("637cd1b437984c0a8e7dfc09"),
  topic: 'esp32/timeTrigger',
  payload: '2022-11-22T13:42:12.429Z',
  qos: 0,
  retain: false,
  _msgid: '7df441ec5d78099b'
},
{
  _id: ObjectId("637cd1b537984c0a8e7dfc0b"),
  topic: 'esp32/timeTrigger',
  payload: '2022-11-22T13:42:13.436Z',
  qos: 0,
  retain: false,
  _msgid: '6af981251f3e9bd5'
}
```



```
20:42:07.417 -> Message arrived on topic: esp32/output. Message: on
20:42:07.417 -> Changing output to on
20:42:07.417 -> 1
20:42:07.417 -> on
20:42:07.417 -> distance: 30.09 cm
20:42:08.396 -> 1
20:42:08.396 -> on
20:42:08.396 -> distance: 6.31 cm
20:42:09.408 -> 1
20:42:09.408 -> on
20:42:09.408 -> distance: 4.56 cm
20:42:10.421 -> 1
20:42:10.421 -> on
20:42:10.421 -> distance: 30.09 cm
20:42:11.430 -> 1
20:42:11.430 -> on
20:42:11.430 -> distance: 30.09 cm
20:42:12.442 -> 1
20:42:12.442 -> on
20:42:12.442 -> distance: 4.56 cm
20:42:13.452 -> 1
20:42:13.452 -> on
20:42:13.452 -> distance: 9.54 cm
20:42:14.429 -> 1
20:42:14.429 -> on
20:42:14.463 -> distance: 30.09 cm
20:42:15.441 -> Message arrived on topic: esp32/output. Message: off
20:42:15.441 -> Changing output to off
20:42:15.441 -> The system is now turn off!
20:42:16.456 -> The system is now turn off!
```



Video URL.

---

https://drive.google.com/file/d/1ZypbkKjtcg5UdDR7cFa6bBC7o_v5fVFc/view?usp=share_link