ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Bachelor Thesis

Research and construction of system for remote supervizing and management rural facilities (rural Pomones).

Του φοιτητή
Κουκογιώργου Βασίλειου
Αρ. Μητρώου: 164683

Επιβλέπων
Ονοματεπώνυμο Γιακουμής Άγγελος
Βαθμίδα Λέκτορας

3 Σεπτεμβρίου 2022

Τίτλος Δ.Ε Μελέτη και κατασκευή συστήματος για την απομακρυσμένη επίβλεψη και διαχείριση αγροτικών εγκαταστάσεων
Κωδικός Διπλωματικής Έργασιας 22106
Ονοματεπώνυμο φοιτητή Κουκογιώργος Βασίλειος
Ονοματεπώνυμο εισηγητή Γιακουμής ’Αγγελος
Ημερομηνία ανάληψης Δ.Ε 27-01-2022
Ημερομηνία περάτωσης Δ.Ε. 31/08/2022

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

«Αφιέρωση»

Πρόλογος

Η πτυχιακή εργασία ”Μελέτη και κατασκευή συστήματος για την απομακρυσμένη επίβλεψη και διαχείριση αγροτικών εγκαταστάσεων (αγροτικές πομόνες)” περατώθηκε στο πλαίσιο των κανονισμών και των προυποθέσεων ολοκλήρωσης των σπουδών για το τμήμα ”Μηχανικών Πληροφορικής Ηλεκτρονικών Συστημάτων” του Διεθνούς Πανεπιστημίου Ελλάδος. Η ανάληψη τόσο και η ανάπτυξη του θέματος συνέβη υπο την επίβλεψη και καθοδήγηση του καθηγητή, κύριου Άγγελου Γιακουμή. Σχετικά με την τελική ανάληψη και την ολοκλήρωση του θέματος, έγινε με σκοπό την περαιτέρω ανάπτυξη περιορισμένων υπαρκών λύσεων σχετικά με τον έλεγχο και τον προγραμματισμό της πιο αποδοτικής και στοχευμένης άρδευσης που υπάγετται στα πλαίσια της ”έξυπνης γεωργίας.Το συγκεκριμένο πλάισιο ανάπτυξης και υλοποίησης αποτελεί στόχο της ανάπτυξης μιας αποδοτικής λύσης που θα είναι σε θέση να βελτιώσει την αγροτική παραγωγή, να εξοικονομεί τόσο χρόνο οσο κόστοι και πόρους.

Περίληψη

Στο πλάισιο αυτής της πτυχιακής είναι να αναπτυχθεί και να δημιουργηθεί ένα σύστημα το οποίο θα είναι σε θέση να διαχειριστεί, να ενημερώνει και να παρέχει άμεσα πληροφορίες στον διαχειριστή του με σκόπο την πρόληψη και την σύνεχιση της διαδικασίας της άρδευσης. Η δημιουργία του συστήματος χωρίστηκε σε επιμέρους βήματα τα οποία ήταν «αριθμος».Αρχικά σχεδιάστηκε το κύκλωμα, ποιά η λειτουργία του, σε ποιές περιπτώσεις χρησιμεύει και η λογική κάτω απο την οποία θα σχεδιαστεί.Αφότου αναπτυχθήκε το σχέδιο επιλέχθηκαν τα κατάλληλα δομοστοιχεία που θα το αποτελλούσαν μερικά έξ αυτών *RaspberryPI,Esp8266,LoraSX1276,wireboards etc* . Ακολούθησε η σύνδεση τους και η ανάπτυξη κώδικα για την κάθε λειτουργεία που θα έιναι σε θέση να εκτελεστεί είτε ενέργειας *Start,Stop* έιτε διαγνωστικές *Diagnostics,email.* Έπειτα τέθηκε σε λειτουργεία με σκόπο την δοκιμή και την λήψη αποτελεσμάτων.Μερικές εκ των παρατηρήσεων-αποτελεσμάτων ήταν πως επιτεύχθηκε, ο απομακρυνσμένος έλεγχος και πρόσβαση, η έγκαιρη και άμεση ενημέρωση του διαχειριστή τοσο και η ανάκαμψη της διαδικασίας της άρδευσης κατόπιν αιτήματος μετά απο διακοπή.

«Research and Construction for remote supervision and management of rural facilities.»

Vasileios Koukogiorgos

Abstract

Systems are build in order to provide help and efficiency in any working environment or procedure. Smart Agriculture doesn't make an exception. A system is developed for improving productivity increasing efficiency while reducing costs and time spent. This system involves various technological devices from a mini computer acting as a server (Raspberry Pi) to micro controllers like NodeMCU 8266 with modules connected to it providing added functionalities. The schema developed provides functionalities like on demand Start,Stop,Diagnostics and Email commands where the user is in place Starting or stopping a procedure, get information for the state and ask for personalized information to his email. A Raspberry Pi server manages and executes the commands instructed from the user. Commands execute their traversal which is bidirectional. The result is that the user managing the system can have a clear picture of the condition the system is , can manage in advance when and how the procedures will happen , save time from manual travels in order to execute the commands , get the picture of the system and improve efficiency and productivity almost to ideal results.

Ευχαριστίες

Η εκπόνηση μιας πτυχιακής εργασίας είναι ενα σημαντικό σχεδόν πάντα χρονοβόρο έργο για τον εκάστοτε φοιτητή. Το να υπάρχουν άνθρωποι γύρω μας το καθιστά πιο εύκολο και σαφέστατα πιο ευχάριστο. Αυτό ίσως ηταν και το μεγαλύτερο μάθημα αυτής της εργασίας που έμεινε και σε μένα με την πάροδο του χρόνου και την προόδο του θέματος. Κάπου εδω ο κάθε φοιτητής γράφει τις ευχαριστίες του προς γονείς,παππούδες,φίλους τον κοινωνικο του περίγυρο. Ούτε αυτό το ευχαριστώ θα ειναι ιδιαίτερα διαφορετικό ομώς έχει μεγάλη βαρύτητα στο να ειπώθει.Ευχαριστώ!!!

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Συντομογραφίες

Π.Ε.	Πτυχιακή Εργασία
ΔΙΠΑΕ	Διεθνές Πανεπιστήμιο Ελλάδος
Π.Ε.	Πτυχιακή Εργασία
A.I	Artificial Intelligence
SoC	System on a Chip
GPIO	General Purpose Input/Output
MCU	Micro Controller Unit
AP	Access Point
RF	Radio Frequency
GSM	Global System for Mobile Communication
LoRa	Long Range
MQTT	Message queue Telemetry Transport
HTTP	Hyper Text Transfer Protocol
IP	Internet Protocol

Chapter 1: Introduction

1.1 History

Making agriculture more productive and efficient with the help of technology in order to spend less time and produce steadily more isn't a new concept.It made its first Appearance on 1929 when "Linsley and Bayer" developed the first map for studying the variability of the "pH" of the ground [1].Although a more systematic and considerable approach and research bringing up the term of "Smart Agriculture" was introduced in the decade between 1980 to 1990.It was there that the first ground sensors were built which can be considered as the ancestors of modern Internet of Things sensors.As years progressed more technologies starting to find to be developed and improved with a great example being the Global Positioning System also known as (GPS) by "Roger Lee Easton".In years 1997 and 2005 for the first time European and Asian conferences were organized with the topic being no other than Smart Agriculture,branch which ever since has made a global impact in interest and investment so as from government entities to businesses.Lastly in 2015 farm bots made their first appearances with fully automated systems of irrigation.

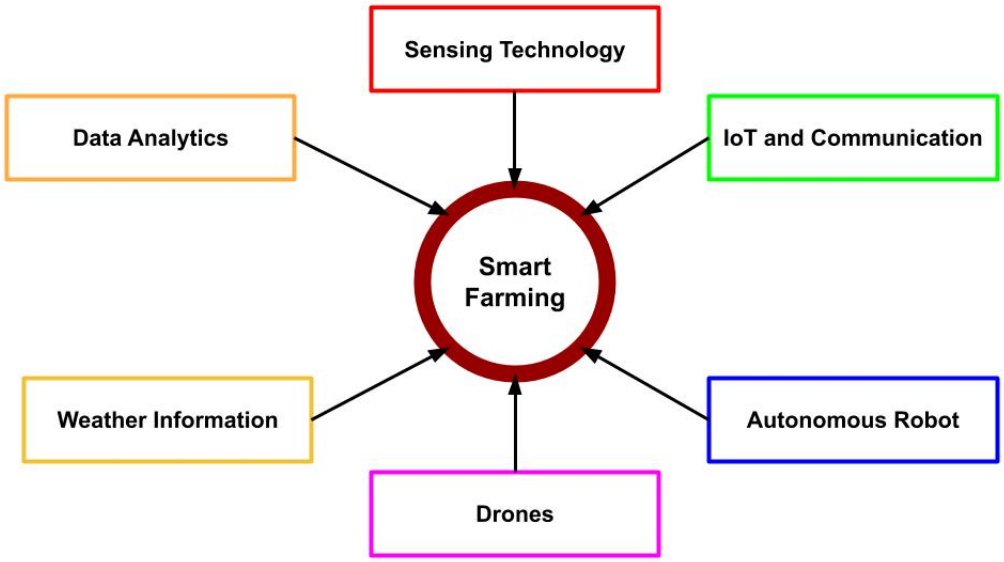

Figure 1.1: Smart Agriculture Key Technologies
Source

1.2 Smart Agriculture

Smart Agriculture represents the amplification of modern information Technologies and communications in Agriculture , which leads to the movement that can be called as third green revolution.Right after the revolution of plant reproduction and genetics,this third revolution started to be applied and enforced in the rural world in conjunction with a variety of equipment that can produce precise results such as internet of things,sensors and alerts,GPS,big data and analytics , Unarmed area Vehicles also known as "UAV" and even robotics.Smart agriculture has the real capability to provide a much more productive and efficient use of resources.However even though in United States 20-80 percent of farmers use at least one technological resource of Smart Agriculture in Europe this statistic is shrinking by the margin where it can vary between 0-24 percent. From the farmer perspective Smart Agriculture should be able

to provide added value by the means that the farmer will be in the right position to make better decisions or have a more efficient usage and management of his resources.

Figure 1.2: Smart Agriculture
Source

1.3 Three Sectors

Smart Agriculture is well linked with three technology sectors.First and foremost there are the management information systems which collect,edit,store and transmit data in the appropriate form that is needed for the execution of programmed tasks and functions which are necessary for a rural business to operate. The second sector is ”Precision Farming” which manages spacial and chronically variation in order to improve the economic efficiency in combination with the reduction of influx and climate effect. Includes Decision Support Systems (DSS) for complete management of rural resources so it can maximize the efficiency of influxes while maintaining the resources [4]. Lastly there are Rural automation and robotics which apply automatic control and ”AI” in every level of the rural production including farmbots and farmdrones [5].

1.4 Systems Developed

Three basic technological sectors are well linked and mentioned. For the first one discussed which is the management information systems which are capable of managing information,editing, collecting , storing and more system have been developed over time. Most of them can be found on the market on different prices. One that will be discussed is the ”n2016” which is produced by a Greek company and provides some descent features and some weak spots [2]. This system is in place to provide remote management and collect information about the system that it will be connected with the use of cellular networking. A ”SIM” card with a number owned by the user is inserted into the ”SIM” slot and the system can provide information through SMS or direct calls to the user.It provides Economical design by having compact rail design so that makes the installation even to an electrical structure much easier. It is powered with 12Vdc and has an embedded battery capable of lasting 4 hours in case of an outage. Through the mobile application that was developed up to 4 mobile numbers can be stored so that a security measure can be set than no more than specific numbers can be able to access and declare commands to the system. Another security feature made is the of the 4 number password that it uniquely provided with the device so in case of stealing the device can be deactivated. Even though this system provides some advantageous features , it also has some weakness. Starting with GSM developed systems which even though provide ”SLAs”

and coverage on tremendous of occasions almost all are ”SIM” reluctant. With that said a Mobile number has to be purchased and most of the times has to be refilled its balance in order to be in place to provide bidirectional communication which in term becomes crucial if a user needs to be informed about the state of its system, which in the long term provides only extra costs. Another weakness is its scalability. In larger systems where automation need to be applied simultaneously in dozens of even hundreds of systems summarized commands can be given.

Figure 1.3: N2016
Source

Figure 1.4: GSM Relay Switch
Source

1.4.1 Comparison

Comparisons some times need to be made when building a system because provides different ideas and structures which in term can be developed or even merged so that the user can have the ”best of both worlds”. Starting with the protocols which are different the system discussed above uses ”GSM” while the system that is developed is built on ”LoRa” and ”Mqtt”. ”GSM” has the upper hand because of the whole infrastructure of telecommunications is being developed for ages so for that reason it can be accessible from almost anywhere, But for same reason user becomes a dependant to ”SMS” and call costs [3]. In contrast LoRa is free which in the Long term provides cost efficiency . To Solve some distance

issues the user can declare commands through a website which is also cost effective solution to provide access. Scalability is the next topic and can really fast exceed the system capabilities.If more than some systems are installed in locations and need to be programmed simultaneously each and every one of them should receive a command in order to start. That isn't developed for the time being. On the other hand even hundreds of systems were connected to one management Node Costs and effectiveness would be same.

Chapter 2: Models Protocols and Modules

2.1 Introduction

Building up an IoT system can mean that many tools,protocols,modules can be involved.This system is no exception,to be built some protocols were used like LoRa and Mqtt.A micro controller was used to build and connect everything together.This system is build including the use of ”NodeMCU 8266”.Modules were also connected and specifically ”Lora SX1276” providing LoRa end to end connectivity.Another Module used is a current sensor(”ACS712”) for providing measurements about the current and the status of the system,a relay is also used for start,stop purposes. At last a ”Raspberry PI” was used acting as a server managing and keep auditing of actions interacting with the system. The system is split in three parts.The first part is a Raspberry PI for deploying managing and auditing actions that have to do with the system.Second is an ”ESP NodeMCU 8266” connected with a Lora Module acting as an intermediate node between the server and the Endpoint and lastly another ”ESP NodeMCU 8266” that can send information and execute commands using LoRa,start stop a motor with the use of a relay,and collect current measurements with the use of a sensor.In order for everything to function properly and scale to the system needs code had to be developed using Arduino.

2.2 Arduino

Using Arduino is simple and affordable that is one of its many reasons why it is used on thousands of projects,applications and systems,its software its easy to use to amateur users so as the experienced ones [6].It can run in multiple platforms for macOS to Ubuntu and of course Windows[33][34]. Some of its usual characteristics why someone use Arduino are summarized below :

- Open source : everyone can contribute and develop code which can be extended by someone experienced or so.
- Cross Platform : It is used by the majority of software (Windows,MacOs,Ubuntu)
- Understanding is easy: Experienced programmers or even amateurs can understand and use Arduino IDE
- It is cheap: no expenses need to start coding and most of its boards are low priced.

2.2.1 ESP NodeMCU 8266

As Mentioned the micro controller used is ”*ESP NodeMCU 8266*”.It is a System on Chip (SoC) and is produced by a Chinese company named *Espressif* 42110432.it consists from a 32-bit ”*Tensilica* L106” micro controller unit (MCU), a WiFi transceiver and 12 GPIO (General Purpose Input/Output). It can work fluently in many conditions and it can withstand temperatures between -40C up 125C.This micro controller can be programmed by Arduino IDE and connect to WiFi so as the internet and work as an access point (AP).In addition can host a server or connect to one. Another Honorable Mention that has

been published is the ”ESP32” which it has 2 cores, has lower power consumption and provides WiFi and Bluetooth on its Capabilities[34].

Figure 2.1: ESP8266-NodeMCU
Source

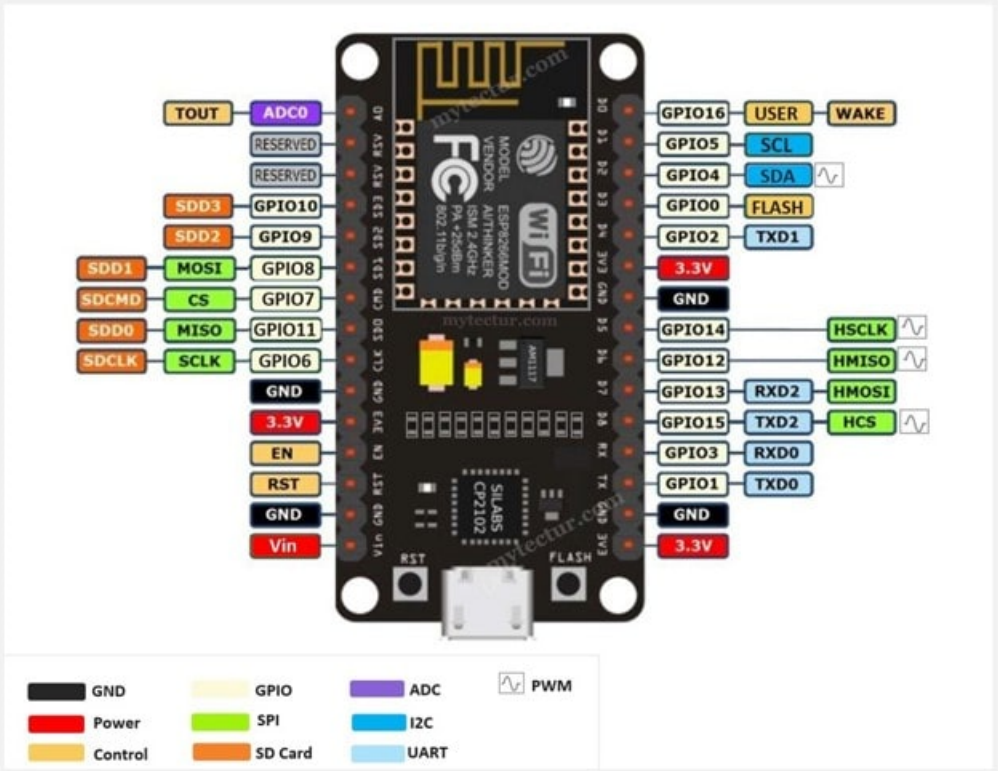

Figure 2.2: ESP8266-NodeMCU GPIO
Source

Πίνακας 2.1 ESP8266 NodeMCU Specifications

Voltage	3,3V
Flash Memory	512K up to 16Mb
Processor	32bit Tensilica L106
Processor Clock	80-160MHz
Consumption	up to 170mA
RAM	32K + 80K
GPIO Pins	17
802.11 connectivity	b/g/n/d/e/i/k/r
Max TCP connections	5

0	GPIO16
1	GPIO5
2	GPIO4
3	GPIO0
4	GPIO2
5	GPIO14
6	GPIO12
7	GPIO13
8	GPIO15
9	GPIO3
10	GPIO1
11	GPIO9
12	GPIO10

Πίνακας 2.2 ESP8266 NodeMCU GPIO

2.2.2 ESP NodeMCU 8266 To Arduino Ide

ESP NodeMCU 8266 can work with Arduino IDE.For this to happen the firmware of the microcontroller needs to be installed on the Arduino IDE interface [6].The Walkthrough is relatively simple and it can be summarized in 4 steps.

- **Step 1** Navigate to File Preferences.
- **Step 2**: Paste the following link to Additional Board Manager URLs *http://Arduino.esp8266.com/stable/package_esp8266com_index.json*
- **Step 3**: Navigate to Tools → Board → Boards Manager

- **Step 4**: Search for ESP8266 and install.

2.3 LoRa Protocol

LoRa stands from Long Range.It is a physical proprietary radio communication technique.It is based on spread spectrum modulation techniques.The development was made by ”Cycleo” which was later acquired by ”SemTech”.”LoRaWAN” defines the software communication protocol and system architecture. The continuous development of the ”LoRaWAN” protocol is managed by ”LoRa Alliance”, of which SemTech is one of its founding members.”LoRa” operates in the sub ”GHz” spectrum which makes it transparent so it can cover larger areas in contrast for example with WiFi or Bluetooth which can cover some Meters.Generally most of its advantages are summarized below:

- Low Power
- Secure
- Standardized
- Geo location
- Mobile
- High Capacity
- Low cost

Using ”LoRa” is ideal for circumstances when low power fits the purpose using sub ”GHz” Radio Frequencies. ”LoRa Alliance” has established and using 3 frequency bands depending each region. LoRa is one of the technologies that make LoRaWAN possible. LoRaWAN is an abbreviation for Long Range Wide Area Network. designed for the Internet of Things (IoT). LoRaWAN uses a star network topology—similar to a Local Area Networks. All the end nodes connect to LoRaWAN gateways, which connect to one central network server. LoRaWAN, IoT manufacturers can build or buy their own network infrastructure or find a provider that serves the area they want to deploy. LoRa is not limited to LoRaWAN and it is not the same thing. It differentiates in that although it uses LoRa but refers to the network and how data transmissions travel through it.

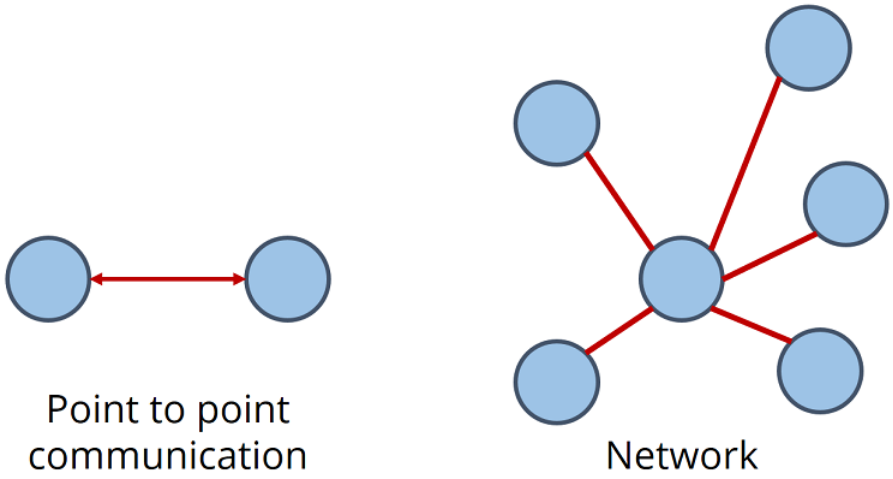

Figure 2.3: LoRa Design Methods - Source
Source

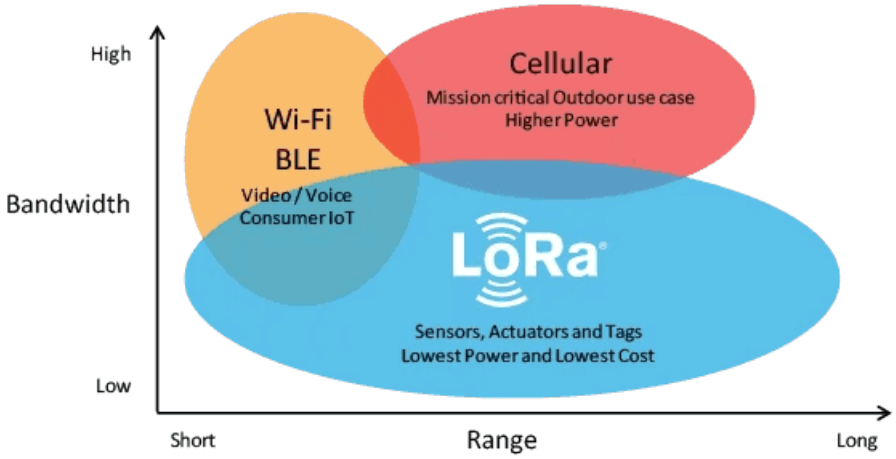

Figure 2.4: LoRa versus Wi-Fi Bluetooth - Source
Source

Πίνακας 2.3 LoRa Frequency Table.

Europe	868MHz
USA	433MHz
China	915MHZ

2.3.1 SX1276 RF Module

LoRa in Order to be implemented needs it's corresponding hardware. Modules were developed so that cause can be achieved.Introducing the RF module SX1276.This module can be connected to Arduino,Esp or similar micro controllers so that in can provide communication using the LoRa protocol [12] [13].It can work on different RF frequencies depending on location and standards applied to each region.For example

GND	Antenna interface
DIO5	GND
REST	DIO3
NSS	DIO4
SCK	VCC
MOSI	DIO0(IRQ)
MISO	DIO1
GND	DIO2

Europe has Established 868MHz, USA has 433MHz and China has 915MHz of the Radio Spectrum.In order to Transmit and Receive the use of an antenna is necessary in Half Duplex mode.It consists of 16 connections points which of them 3 are grounds, 1 is VCC , 6 are Digital Input,Output (including DIO0 which is used to IRQ) and 6 connections which are used so that connection to Micro controller is made. Some applications can be found in remote control and data collection systems.

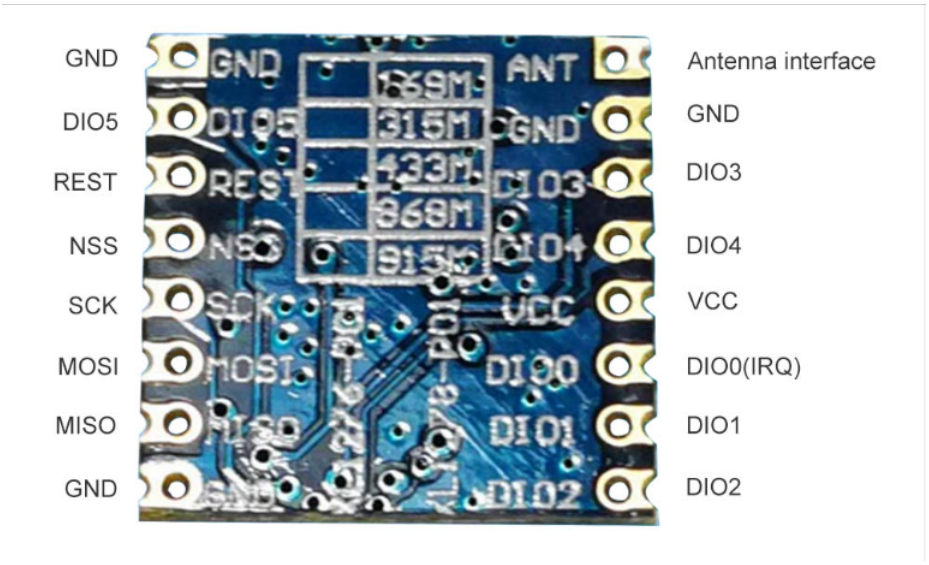

Figure 2.5: SX1276 RF Module
Source

2.4 Relay

In order to provide more flexibility and control to your systems for better management we use an electrical switch named Relay.Relays are switches that are programmed and controlled by micro controllers and can provide an On/Off state to the device connected to by using voltage or current. Acts as an intermediate

between high voltage devices and Arduino. Relays consist of two groups of pins, those with high Voltage and those with low Voltage. High Voltage group has 3 pins mostly in screw terminal.

- COM : Used both in Normally open and closed mode
- NO : Used in normally open mode
- NC : Used in normally closed mode

  Low Voltage group has also 3 pins and in this side there are the pins which connect to the Micro controller.

- GND : Needs to be connected to GND (0V)
- VCC : Needs to be connected to VCC (5V)
- NC : controls the received signals from Micro controller

Figure 2.6: 5V 1 channel Relay Module - Source
Source

2.5 Liquid Crystal Displays (LCD)

Liquid-Crystal displays are flat panel which use light modulated optical devices that make use of liquid crystals in combination with polarizers. It is often misunderstood that Liquid-Crystals emit light. The answer is that they don't.In Contrary they use back lights or reflectors in order to make the images in colour or monochrome. LCD are used in a variety of applications and sizes some of them are Televisions,computers screens and some more. Small LCD screens are commonly used in portable devices like smartphones,digital cameras,watches,calculators and more. LCD screens have become more of a standard by replacing and proving their authority over the bulky cathode-ray tube known as "CRT" providing

more coverage in sizes and costs.As the time was passing by slowly they get replaced by oLED which can be manufactured in different and every kind of a shape and have lower response times,wider colour,lower weight and slimmer profile.

2.5.1 LCD 1602 Module

With the growth of Arduino both LCD screens so as oLED found room to be applied and used in various projects.One module that was developed so that it can be compatible with Arduino is the LCD1602 Module. LCD1602 is a kind of dot matrix module which is capable of showing numbers,characters and letters. It's composed of 5X7 or 5X11 dot matrix positions, every position displays one character. There is a dot pitch between two and a space between lines so it separates characters and line accordingly. The model is named "1602" because it displays 16 characters in 2 lines.In general "LCD1602" its ports are parallel so it can take advantage and use several pins simultaneously. "LCD1602" has two categories 4 port and 8 port connections.If the 8 port connection is used more digital ports are occupied and that can be a problem in case more sensors need connection. If more sensors need to be connected, there possibly will be no ports available. Therefore, the four-port connection is used here for better application[7].

Figure 2.7: LCD Screen Module
Source

2.5.2 LCD Display I2C Interface Module

The four-port connection can be achieved with a LCD display I2C Interface Module which is a serial I2C connection. I2C is a synchronous, multi slave, multi master packet switched, single-ended serial bus [8]. multiple chips can be connected to the same bus. I2C uses only two bidirectional open collector or open drain lines, Serial Data Line (SDA) and Serial Clock Line (SCL), pulled up with resistors. The contrast can be controlled by the LCD display by rotating the POT on the module. Moreover there is a jumber fixed on the module. When the Jumber is removed the back light of the LCD display will go OFF.

Figure 2.8: I2C LCD Module
Source

2.6 ACS712

The Allegro ACS712 provides economical and precise solutions for AC or DC current sensing in industrial, commercial, and communications systems [9]. The device package allows for easy implementation by the customer. Typical applications include motor control, load detection and management, switched-mode power supplies, and over current fault protection. he device consists of a precise, low-offset, linear Hall sensor circuit with a copper conduction path located near the surface of the die. Applied current flowing through this copper conduction path generates a magnetic field which is sensed by the integrated Hall IC and converted into a proportional voltage. Device accuracy is optimized through the close proximity of the magnetic signal to the Hall transducer. A precise, proportional voltage is provided by the low-offset, chopper-stabilized BiCMOS Hall IC, which is programmed for accuracy after packaging. Some of its features are summarized below :

- 66 to 185 mV/A output sensitivity
- Low-noise analog signal path
- Extremely stable output offset voltage
- Nearly zero magnetic hysteresis
- 5.0 V, single supply operation

Depending the needs of each applications and demands ”ACS712” is produced in some variations. Broadly used are those with 5,20 and 30 Amps.

Πίνακας 2.6 ACS712 Models

Part Number-Model	Package Type	Temperature
ACS712ELCTR-05B-T	8-lead SOIC	-40°C to 85°C
ACS712ELCTR-20A-T	8-lead SOIC	-40°C to 85°C
ACS712ELCTR-30A-T	8-lead SOIC	-40°C to 85°C

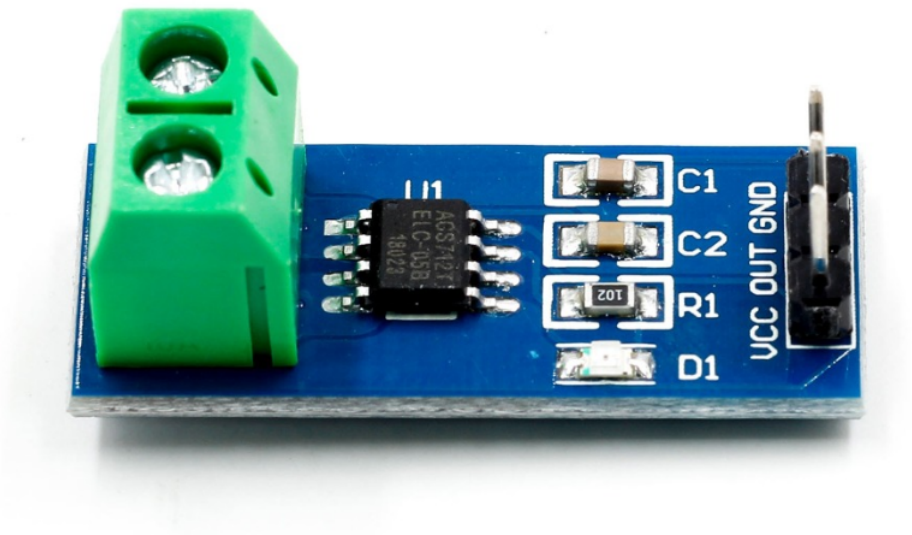

Figure 2.9: ACS712 Module
Source

2.7 Power Redundancy

2.7.1 Batteries

Redundancy is an issue that concerns everyone when talking about functionality especially in the case that there no on spot supervision. For that reason a power back up should be considered and build to support everything on LoRa Endpoint. The most usual and efficient if not the only way to provide power even when there is an outage is through stored power inside of batteries. They are capable of providing power for hours even in some cases days which perfectly can fit system needs. Batteries that are gonna be used are ”Li-ion” cells ”18650” which can be recharged and have capacity of 3000mah.

Figure 2.10: Battery 18650 Li-ion

2.7.2 Battery Charger and Boost Converter

This module is a small single cell lithium battery charging module which also includes a 1A step-up (boost) converter for powering a large range of applications. The module will charge most types of single cell (3.7) LiPo/Li-Ion batteries from either 4 to 7.5V power supply input, or from a standard 5V USB port/adapter. A battery charge and standby LED is also included for visual indication.

Besides battery charging capabilities this module also includes an adjustable boost converter which is capable of stepping up the attached battery voltage from 4.5 to 24V with a maximum supply current of 2A. It is a good match for Micro controller purposes because of low voltages it provides plus it can power up a battery for redundancy purposes.

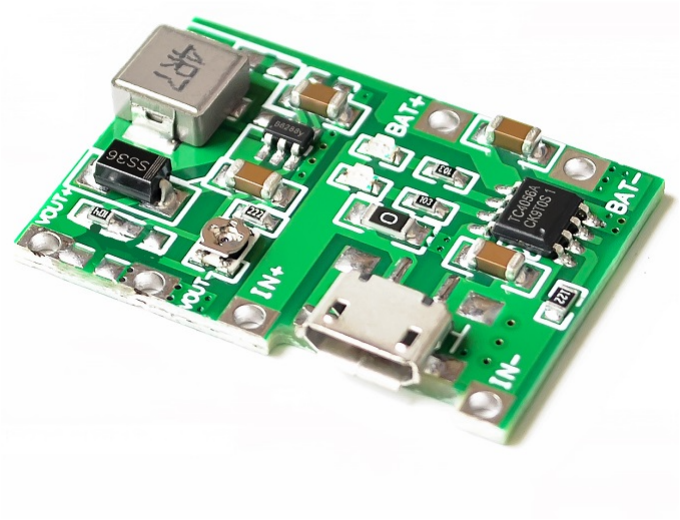

Figure 2.11: Li-ion Battery Charger Boost Converter
Source

2.8 Raspberry Pi

Raspberry Pi is a series of small single board computers.Broadcom in association with Raspberry Pi Foundation were the first who started the development. It's original purpose was to promote the teaching of computer science but rapidly because popular exceeded expectations and target group [10].The Addition of USB and HDMI standards made it usable for daily use.The first generation (**Raspberry Pi Model B**) was released in early 2012, followed by the simpler and cheaper Model A. In 2014 an improved design, **Raspberry Pi Model B**+ hit the stores. It got the size of a credit card and featured ARM11 processors.By 2015 **Raspberry Pi2** initially featured a 900 MHz 32-bit quad-core ARM Cortex-A7 processor with 1 GB RAM.Next Year **Raspberry Pi 3 Model B** was released in February 2016 with a 1.2 GHz 64-bit quad core ARM Cortex-A53 processor, on-board 802.11n Wi-Fi, Bluetooth and USB boot capabilities. On "Pi Day" of 2018, the **Raspberry Pi 3 Model B+** was launched with a faster 1.4 GHz processor, a three-times faster gigabit Ethernet (throughput limited to ca. 300 Mbit/s by the internal USB 2.0 connection), and 2.4/5 GHz dual-band 802.11ac Wi-Fi (100 Mbit/s). Lastly on 2019 **Raspberry Pi 4 Model B** was released with a 1.5 GHz 64-bit quad core ARM Cortex-A72 processor, on-board 802.11ac Wi-Fi, Bluetooth 5, full gigabit Ethernet (throughput not limited), two USB 2.0 ports, two USB 3.0 ports, 1–8 GB of RAM, and dual-monitor support via a pair of micro HDMI (HDMI Type D) ports for up to 4K resolution. The version with 1 GB RAM has been abandoned and the prices of the 2 GB version have been reduced. An addition powering through a USB-C port enabling additional power for downstream peripherals [29].

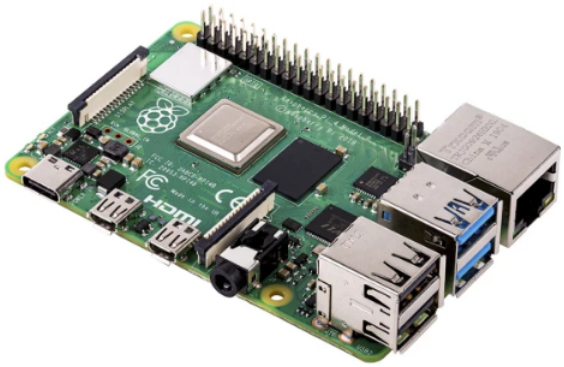

Figure 2.12: Raspberry Pi 4 Model B
Source

2.9 Mqtt

MQTT stands from MQ Telemetry Transport. It is lightweight publish-subscribe M2M (machine to machine) network Protocol. The design was made for connections on remote locations with devices which may have limited resources or network bandwidth. TCP/IP suite is recommended to be used

because it provides lossness, bi-directional connections.It is an open standard and ISO referred (ISO/IEC 20922). It's first version of the Protocol was authored in 1999 by Andy Stanford-Clark and Arlen Nipper. Its purpose was to monitor oil pipelines within the "SCADA" industrial control system. Its goal was to create a protocol that doesn't consume too much bandwidth and consumes the least amount of energy. MQ in the MQTT came from IBM MQ product which means Message queue. In contrast the protocol uses publish-subscribe messaging system which in term means that no queues are used. Later version 3.1 of the protocol was referred to as "MQ" Telemetry Transport" by IBM. Version 5 adding more several features was released on 2019. By an overview it can be observed that MQTT defines two network entities the message broker and its clients. Broker is the server that receives the messages and routes them to its corresponding destination. Client is a device that has MQTT library and connect through network to the broker. Information uses topics to be populated and build its hierarchy. If a publisher has an item of data to send it just sends a control message and the broker distributes the data to clients that are subscribed on that topic. If a message received on a topic with no subscribers it just discards the topic except from the occasion that the message was designated to be retained. Then broken stores the last retained message plus its QoS of its selected topic. If a clients subscribed and a retained message exists if immediately receives the message. Only one message can stored as retained per topic. The minimum control message sent by MQTT is small as 2 bytes and the maximum a control message can be is 256 megabytes. There are 14 defined message types which can vary from connect to disconnect client from a broker, to publish, acknowledge receipts of data published and to supervise the client server connections. MQTT mostly relies on TCP in order to transmit data. A variant named MQTT-SN is used for transmits over UDP or Bluetooth. One Important thing that's worth noting is that MQTT is not secure because it sends credentials in plain text. TLS can be used so that it can provide encryption ,ensure that no interception or modification has occured. MQTT uses port 1883 for unencrypted connections and port 8883 for encrypted connections. The MQTT broker is software that runs on a computer that exist on premises or even the cloud. The Broker can be paradigmatized as a post office . Clients don't directly connect of the recipient but use the subject line which is called topic. Every client that subscribe can receive a copy of the messages for that topic. Both One broker to many clients and many brokers to one client can occur. Every client is capable or either publishing or subscribing. The main advantages of a MQTT broker are summarized below :

- Can easily Scale from numbered to many devices
- Reduces Network strain without compromising the security
- Eliminates vulnerable client connections
- Manages and tracks client states

2.10 MQTT vs HTTP

When structuring a system what protocols should possibly be used should be thought in advance. Building an IoT system most of the times concludes to either HTTP or MQTT. MQTT has lower power consumption which makes it preferable if not perfect for IoT systems.Secondly HTTP has high bandwidth consumption making it insufficient for larger deployments and more devices. Lastly HTTP has bigger overheads which makes it more complex and slower than MQTT [11].

	MQTT	HTTP
Design Orientation	Data centric	Document centric
Pattern	Publish/Subscribe	Request/Response
Complexity	Simple	More complex
Message size	Small,with a compact binary header 2 bytes size	Larger, partly because status detail is text-based
Service levels	Three quality of service settings	All messages get the same level of service
Extra libraries	Libraries for C (30KB) and Java (100KB)	Depends on the application (JSON,XML) but larger
Data Distribution	Supports 1 to 0, 1 to 1 , 1 to many	1 to 1 only

2.11 Node Red

Node-RED is a a flow-based development tool for visual programming developed originally by *IBM* for wiring together hardware devices, APIs and online services. It provides a browser-based flow editor named **Node-RED Dashboard** UI that simplifies connection in a declarative way to flows using the wide range of nodes in the palette. Flows can be then deployed to the runtime almost immediately.It is built on *Node.Js* taking full advantage of its event-driven, non-blocking model which in turn makes it efficient on low cost devices with a great example being Raspberry Pi or in docker instances or even the cloud. Moreover because of its easy imports-exports was appalling to social development [14] [15] [16][31].

2.11.1 Installing NodeRED

NodeRED is ideal for systems like Raspberry Pi because is not heavy resourceful and can be used in projects were servers are involved ,so for that reason it suits Raspberry Pi. To proceed to installation the package with the repo should be requested with the command ”bash <(curl -sL https://raw.githubusercontent.com/node-red/linux-installers/master/deb/update-nodejs-and-nodered)”. A good suggestion after the package is installed is to enable the service starting on each and every boot so that unnecessary time is saved . This is accomplished by executing the ”sudo systemctl enable nodered.service” command. NodeRED can then be accessed by typing the hostname IP which in most cases is the one of Raspberry Pi following with ”:” plus the port 1880 , so the format is the one of ”http://<hostname>:1880”.

Chapter 3: Connections and Diagrams

3.1 Connections

In order for a system to be built be efficient but mostly understandable from the designer to the developer or even the viewer a design pattern,a logical diagram must be handcrafted. This section explains connections made between the devices and their respective modules. Starting with Raspberry Pi which it doesn't have any module connected to it's physical layer. Raspberry acts as the back-bone of the system.It runs RaspbianOS as the operating software because it lightweight and fits the purpose for this system. It hosts NodeRed Framework in which the whole BackEnd is developed in addition with a database which stores and displays data inserted for status changes from the other end so as a Mqtt Server Publish/subscribing to Mqtt topics. Moreover it provides a simple UI to display data and execute commands through Node-RED Dashboard UI. It is connected to Wi-Fi in order to connect with the NodeMCU 8266 and uses "MQTT" as a protocol to provide connection with it [26].As an intermediate device a Node-MCU 8266 is used connected with an LoRa Module "SX1276" which is connected to Wi-Fi and uses "MQTT" protocol so that persists a connection with Raspberry Pi and publishes or listens to MQTT messages being sent by 2 MQTT topics [17]. Every message in which is destined to the other NodeMCU 8266 which acts as an endpoint uses LoRa Protocol in order to transmit commands sent as messages or listen for incoming replies. Lastly another Node MCU 8266 is soldered with a second LoRa Module SX1276 and makes possible the End to End connectivity. In addition it has connected a relay which uses when a command is ordered (Start,Stop,Diagnostics) and a current sensor so that measurements are taken and provide live results.

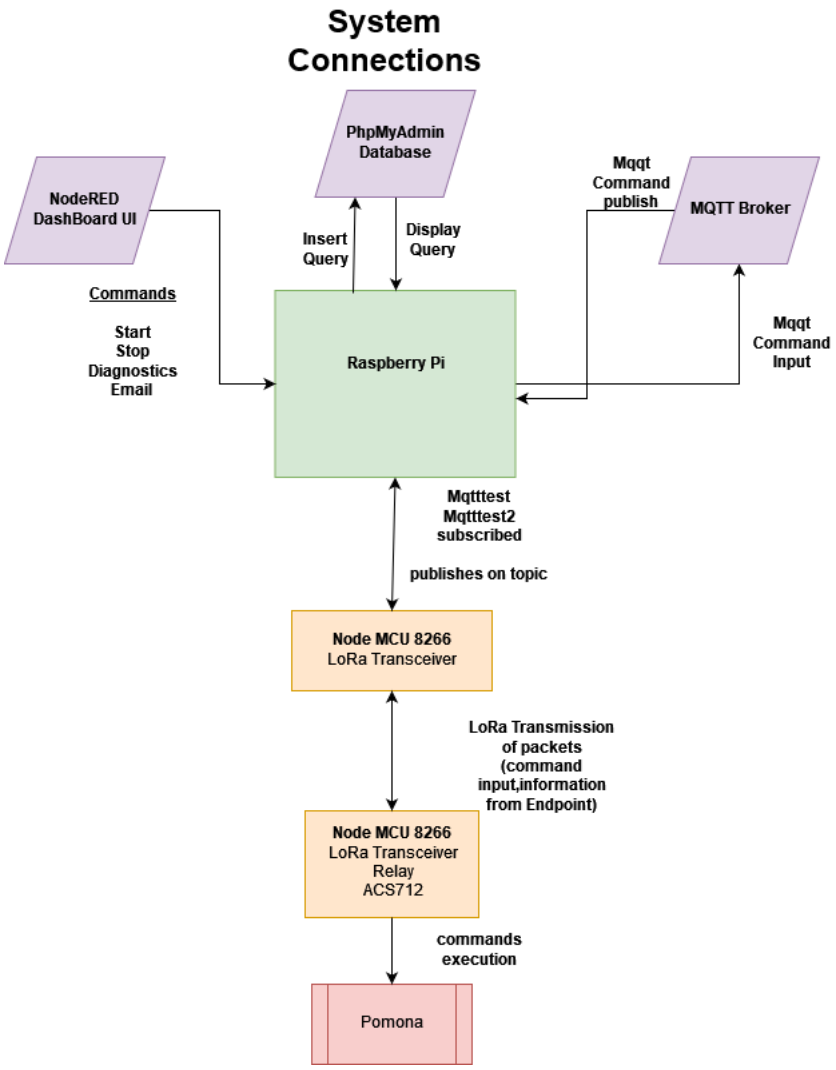

Figure 3.1: System Diagram

3.1.1 NodeMCU 8266 with LoRa SX1276 Sender

LoRa SX1276 Module is not friendly for direct connection on the NodeMCU8266 because it hasn't its own extensions and for that reason it must be soldered [15]. In order to accomplish this some male Straight Header Strip were soldered to LoRa SX1276 Module. On a PCB development board were soldered Female header jumper connectors so that both NodeMCU 8266 and LoRa SX1276 Module could be attached. From the other side of the board connections were soldered. Connections pin to pin follow the table below. In order for the "NodeMCU 8266" to communicate properly and execute the corresponding commands tasked by the user code was written in "C"programming language [30]. Fortunately libraries exist and make coding more easy, customized and functional. A LoRa library was including and was in place to create manage and transmit LoRa messages.

NodeMCU	SX1276
GND	GND
3.3V	VCC
D8	NSS
D7	MOSI
D6	MISO
D5	SCK
D0	RST
D2	DIO0

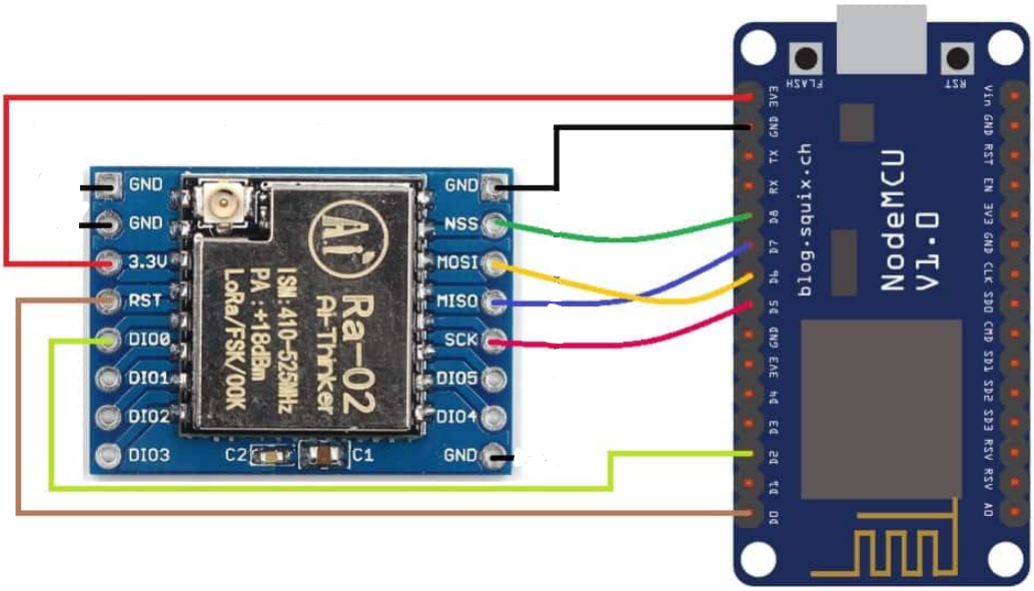

Figure 3.2: LoRa Transceiver Schema
Source

Figure 3.3: LoRa Transceiver Front View
Source

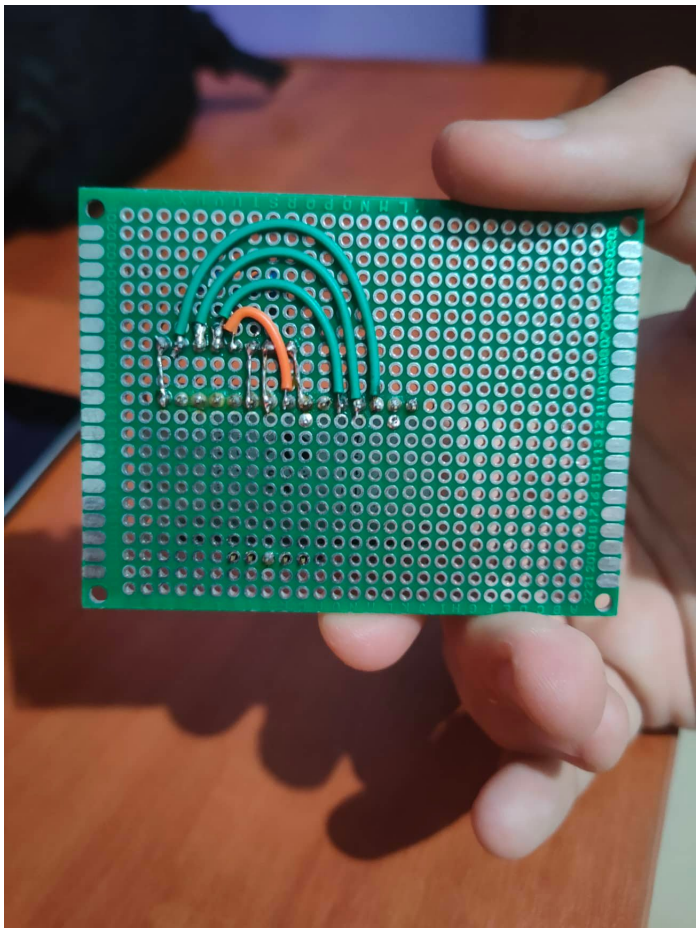

Figure 3.4: LoRa Transceiver Rear View
Source

3.1.2 NodeMCU 8266 Endpoint

Following the same pattern for the LoRa Endpoint side of the system SX1276 LoRa Module should be soldered with male Straight Header strips so then can be connected with Female Header jumpers. Next the relay both the ACS712 needed to be placed on the PCB development Board. Unfortunately both relay and ACS712 current sensor had their responding connection pins soldered upside up. For comfort reason their connections headers were soldered upside down so that they can have direct connection with the development board , plus the benefit of standing as a whole upon the PCB development Board. Moreover some corner jumpers were soldered upon the board so that the 4 pins needed for the LCD Module could be accessed underneath the NodeMCU. The connections pin to pin are more and some micro adjustments had to be done in order for everything to be fitted in without having disconnected pins[32]. Code was written in the same logic as the first ”NodeMCU 8266” with LoRa library. Moreover Library for the ”LCD” module was used so as a relay for making true the On/off purpose of the system.

Mapping of the pins is described in the following table.

NodeMCU8266	Modules
D7	LoRa MOSI
D6	LoRa MISO
D5	LoRa SCK
D0	LoRa DIO0
D3	LoRa RST
D4	LoRa NSS
D1	I2C SCL
D2	I2C SDA
GND	I2C GND
Vin	I2C VCC
GND	Relay GND
Vin	Relay VCC
SD3	Relay In
GND	ACS712 GND
Vin	ACS712 Vin
AD0	ACS712 OUTPUT

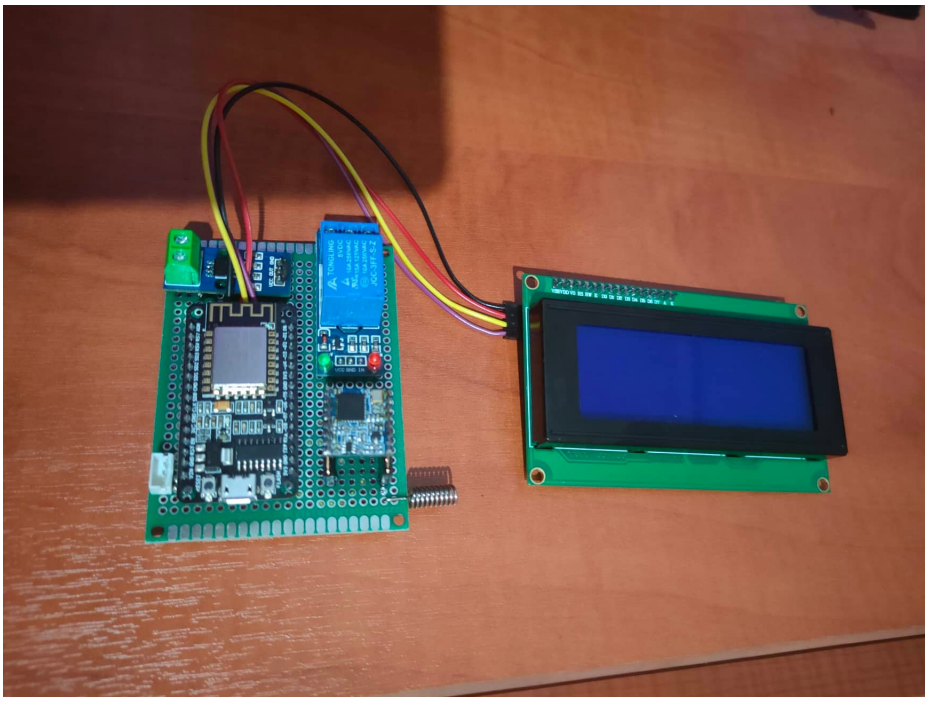

Figure 3.5: LoRa EndPoint Front View

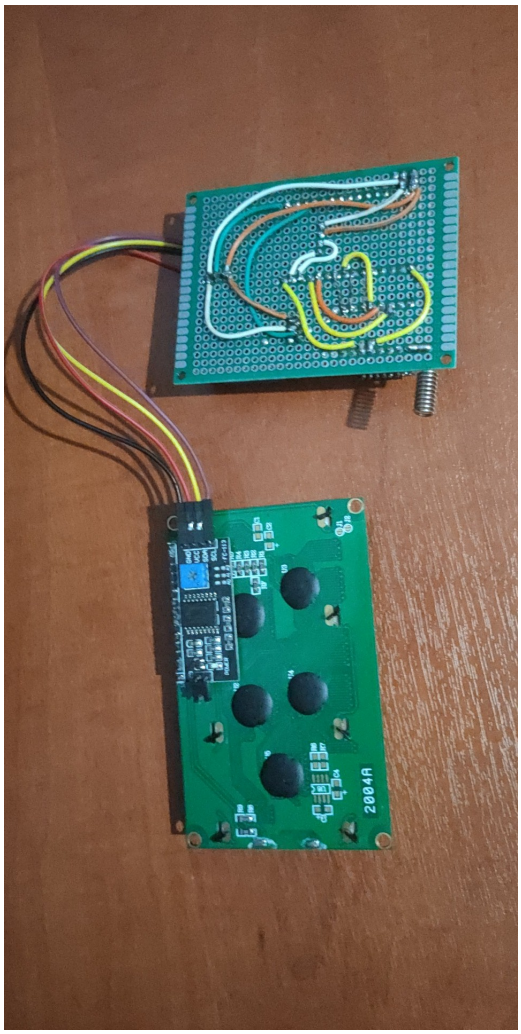

Figure 3.6: LoRa EndPoint Rear View

3.2 MQTT Broker to Raspberry to NodeMCU 8266

NodeRED works and cooperates really well with MQTT so that's one of the reason that was preferred over http for the constructions of this system. MQTT needs 2 entities in order to work a broker and clients [18]. Raspberry Pi can easily host a MQTT broker by just installing it. Right after the installation is complete and the MQTT broker has started to run, NodeRED can connect to this broker by just inserting MQTT nodes from the palette and declaring the topic. On the Other end NodeMCU 8266 needs to have its appropriate library and just subscribe to the same topic. After these configurations are done NodeRED can send control commands to the MQTT broker and the broker can publish them to NodeMCU8266. The reverse traversal can both happen though.

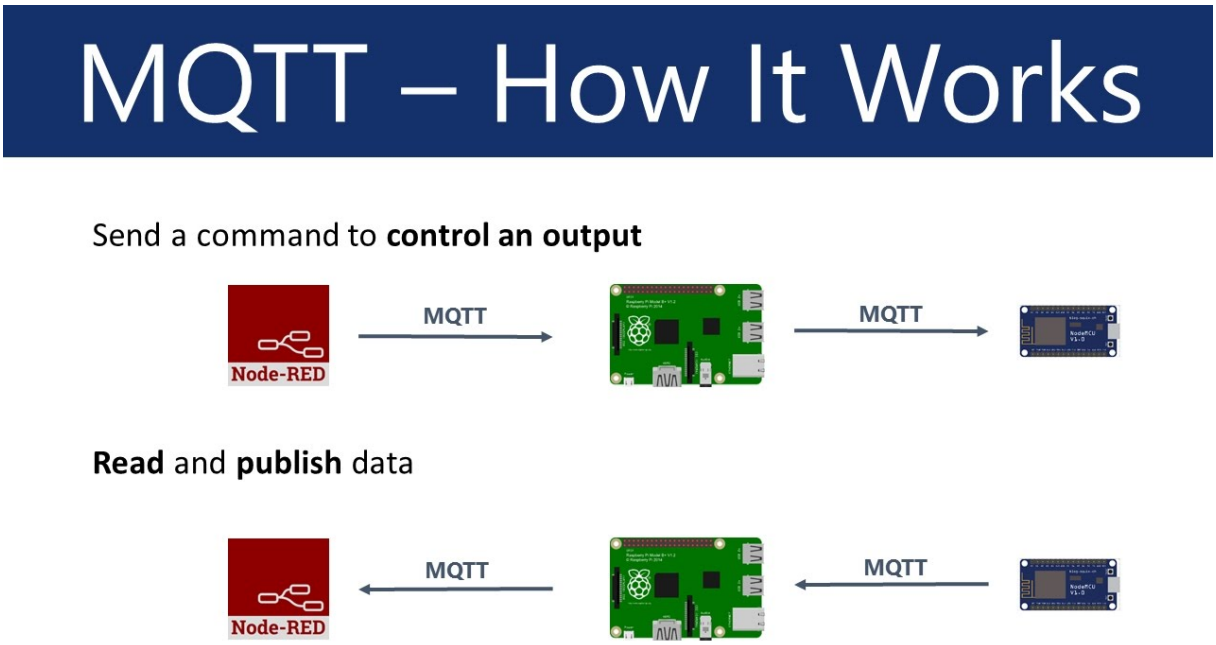

Figure 3.7: MQTT flow
Source

3.3 Logical Diagrams

Logical Diagrams are created for each command and its traversal so that every flow is clear and understandable. Four Diagrams are created. The first is one to describe and display the process of how the system provides ”start” to the motor. The second one explains the reverse scenario of how the the system provides ”stop” to the motor. Next is the ”Diag” flow displays the path in order to get state and voltage from the motor. Lastly the ”Email” flow shows the step needs to be configured in order for an Email to be sent so that the users can have diagnostics on their end.

3.3.1 Start Command

Start Command follows a flow. Starting by the user, start button on PomonaDashBoard UI is pressed.Immediately a publish message message is created on topic Mqtttest with the value ”start”. Node MCU LoRa Transceiver is subscribed to topic Mqtttest and receives the message. It compares it if it matches the acceptable values,if it does it creates a LoRa packet with the value ”start” and transmits it. The second NodeMCU which acts as an Endpoint receives it. Accordingly after it has received the message makes the comparison if the value received matches certain criteria . If it does, it closes the Relay so that the motor can start its working process. Endpoint Creates a LoRa approval message and transmits it back. After it is received , Node MCU creates publishes a message back to Raspberry Pi which is subscribed to the topic ”Mqtttest2”. Raspberry receives it and it is used by NodeRed. NodeRED makes a query which is an insertion to the ”PomonaDatabase” and a second one so that it can display the data of the table named ”Pomona”.NodeRED DashBoard UI is responsible of loading and displaying the table.

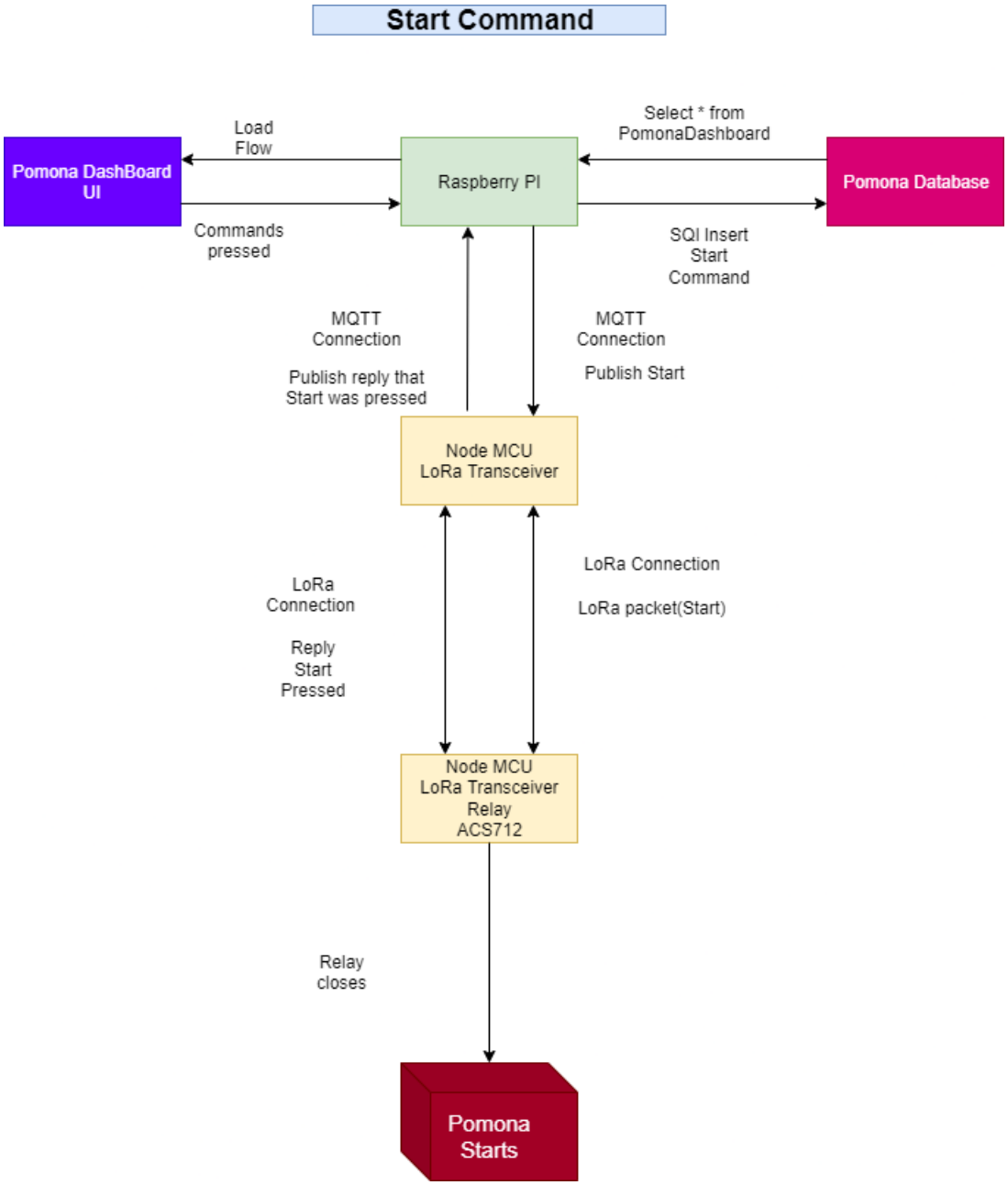

Figure 3.8: Start Command

3.3.2 Stop Command

Stop Command follows a flow. Starting by the user, Stop button on PomonaDashBoard UI is pressed.Immediately a publish message message is created on topic Mqtttest with the value ”stop”. Node MCU LoRa Transceiver is subscribed to topic Mqtttest and receives the message. It compares it if it matches the acceptable values,if it does it creates a LoRa packet with the value ”stop” and transmits it. The second NodeMCU which acts as an Endpoint receives it. Accordingly after it has received the message makes the comparison if the value received matches certain criteria . If it does, it Opens the Relay so that the motor can stop its working process. Endpoint Creates a LoRa approval message and transmits it back. After it is received , Node MCU creates publishes a message back to Raspberry Pi which is subscribed to the topic ”Mqtttest2”. Raspberry receives it and it is used by NodeRed. NodeRED makes a query

which is an insertion to the ”PomonaDatabase” and a second one so that it can display the data of the table named ”Pomona”.NodeRED DashBoard UI is responsible of loading and displaying the table.

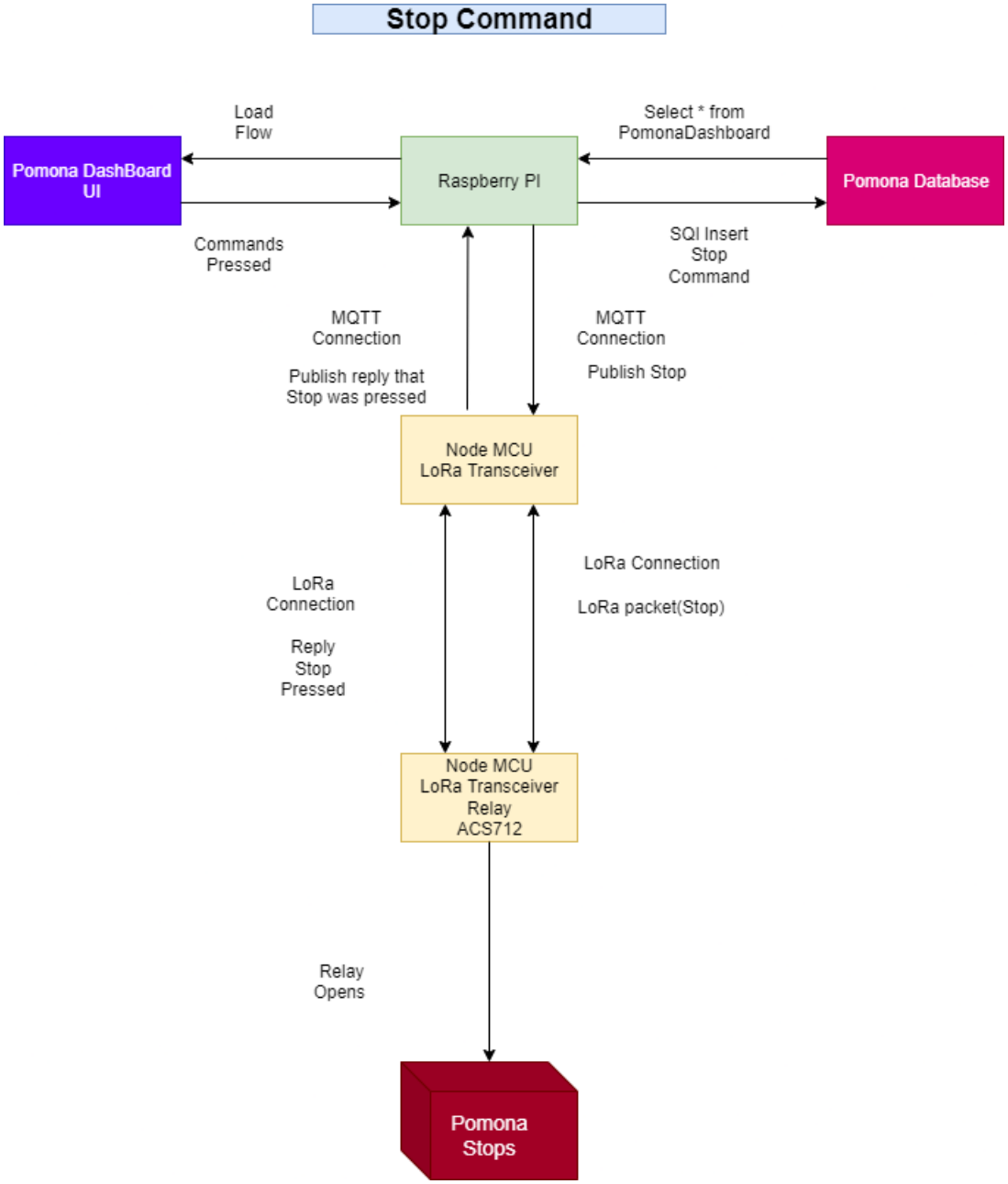

Figure 3.9: Stop Command

3.3.3 Diag Command

Diagnostics Command follows a flow. Starting by the user, Diagnostics button on PomonaDashBoard UI is pressed.Immediately a publish message message is created on topic Mqtttest with the value ”Diag”. Node MCU LoRa Transceiver is subscribed to topic Mqtttest and receives the message. It compares it if it matches the acceptable values,if it does it creates a LoRa packet with the value ”Diag” and transmits it. The second NodeMCU which acts as an Endpoint receives it. Accordingly after it has received the message makes the comparison if the value received matches certain criteria . If it does , Endpoint creates

a LoRa approval message with the state of the motor (On/Off) and the Voltage value received from the current sensor and transmits it back. After it is received , Node MCU creates and publishes a message back to Raspberry Pi which is subscribed to the topic ”Mqtttest2”. Raspberry receives it and it is used by NodeRed. NodeRED makes a query which is an insertion to the ”PomonaDatabase” and a second one so that it can display the data of the table named ”Pomona”.NodeRED DashBoard UI is responsible of loading and displaying the table.

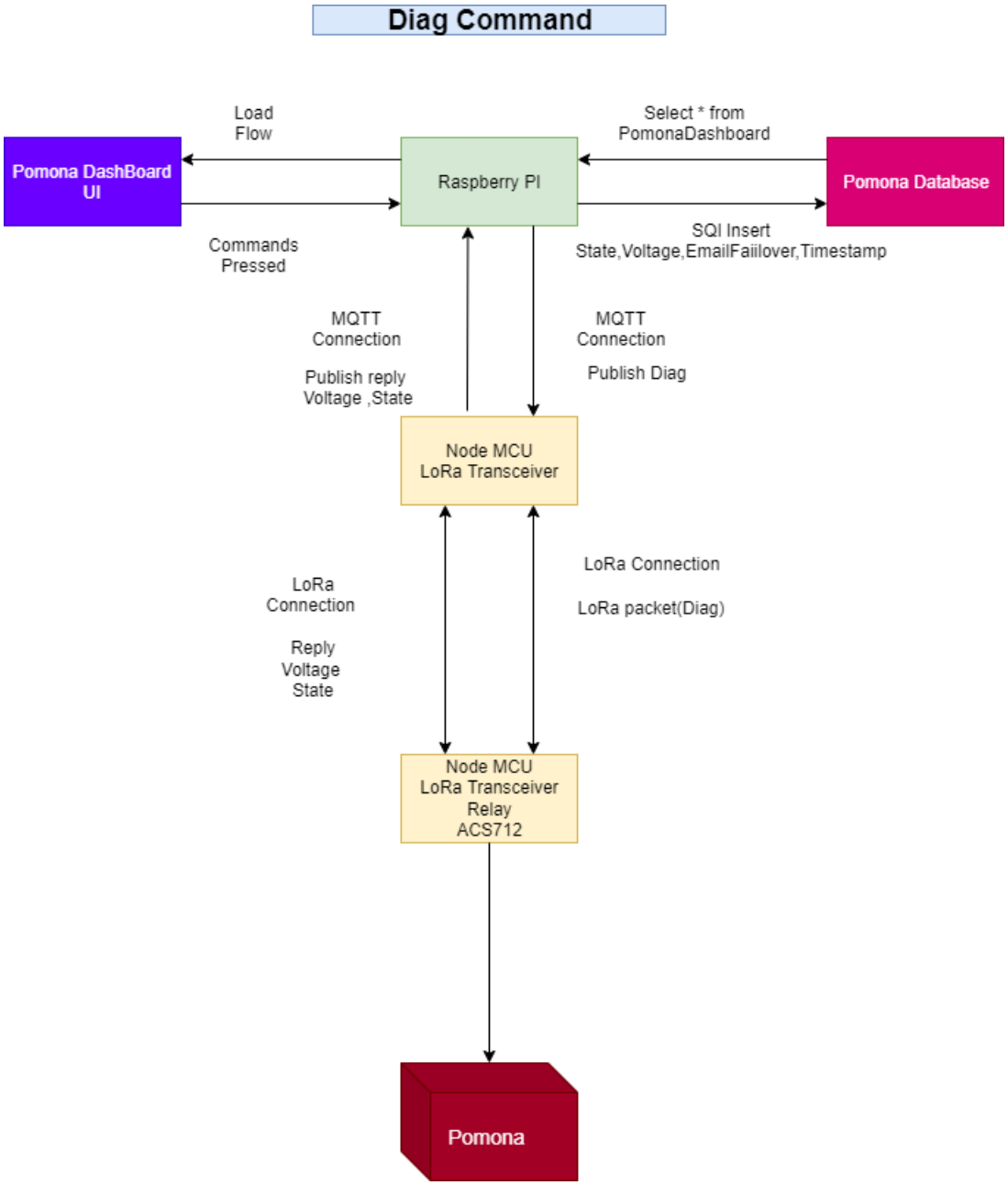

Figure 3.10: Diag Command

3.3.4 Email Command

Email Command follows a different flow. It is solidly performed in the back end on NodeRED. If Email button is pressed on NodeRED Dashboard UI then NodeRED searches for the attachment that will be

appended to the Email that will be sent on the predetermined path,else it will create the file which is a csv file named logs. Next step is the creation of the Email with its name subject and its information.Lastly it forwards to the Email configuration where receivers Email,server and port are declared accordingly with the sender Email plus credentials.

Figure 3.11: Email Command

3.3.5 LoRa Sender Logical Diagram

In order LoRa sender to be in position so that manage LoRa,MQTT,WiFi and every other protocol code had to be written. Using Arduino Ide code was developed followed by logical diagram below. Starting by including libraries and declaring variables.Next connections with WiFi ,MQTT and LoRa had to be established. The next thing if every connection occurs then when input of message occurs with ”Start,Stop,Diag” then a LoRa packet with the message is created. If a LoRa packet is parsed then depending on the command message the value is transfered to MQTT client for further management.

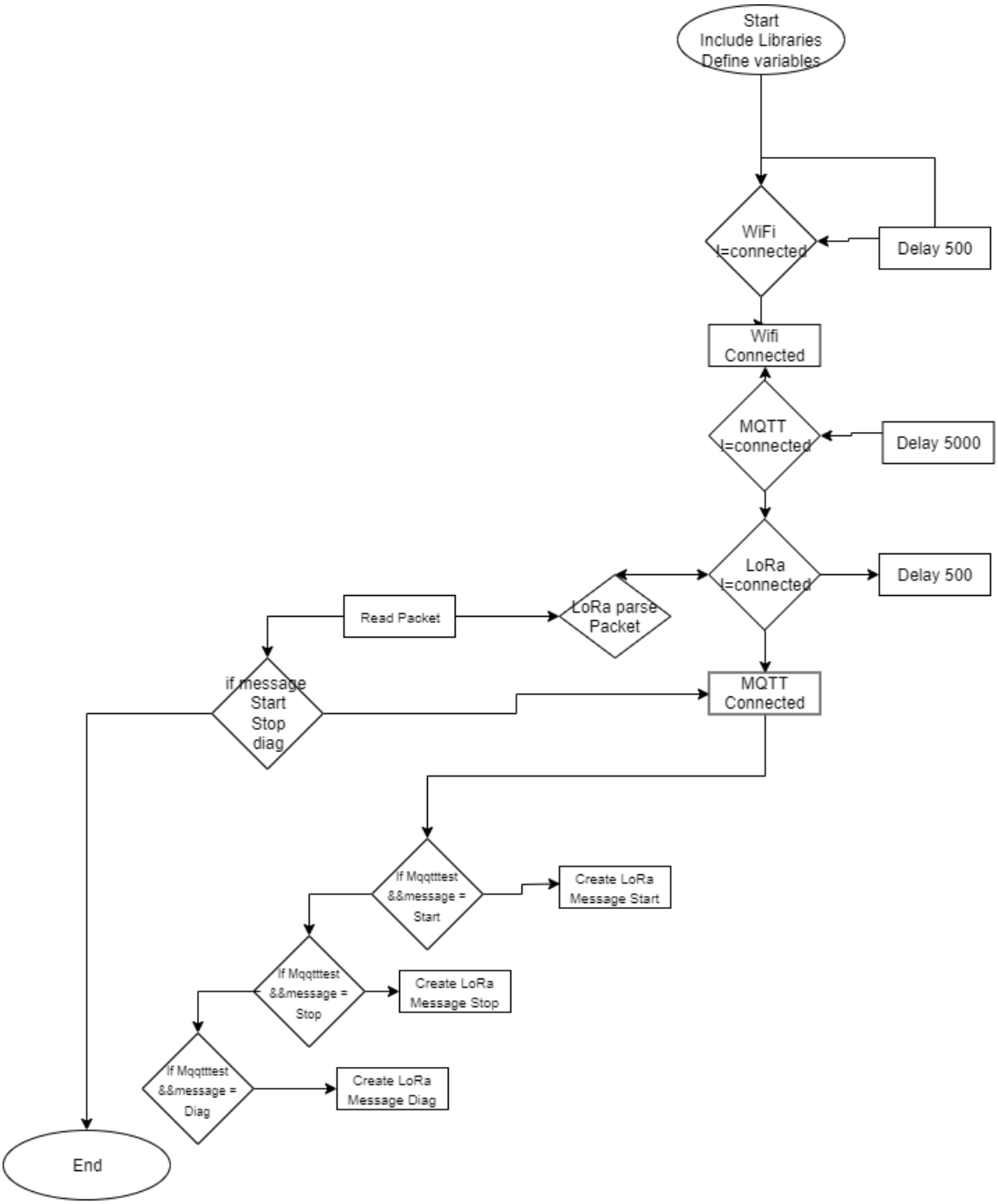

Figure 3.12: LoRa Sender Logical Diagram

3.3.6 LoRa Endpoint Logical Diagram

In order LoRa Endpoint to be in position so that manage LoRa and every other module code had to be written. Using Arduino Ide code was developed followed by logical diagram below. Starting by including libraries and declaring variables.Next connections with LoRa and every module had to be established. After LoRa connection occurs then it awaits input of message with ”Start,Stop,Diag” commands. In case of start relay normally closes and a LoRa packet is sent with value ”Start Pressed”.In case of Stop relay normally Opens and a LoRa packet is sent with value ”Stop Pressed”.In ”Diagnostics case a voltage measurement is stored and sent through a LoRa Packet according with the state of the system.

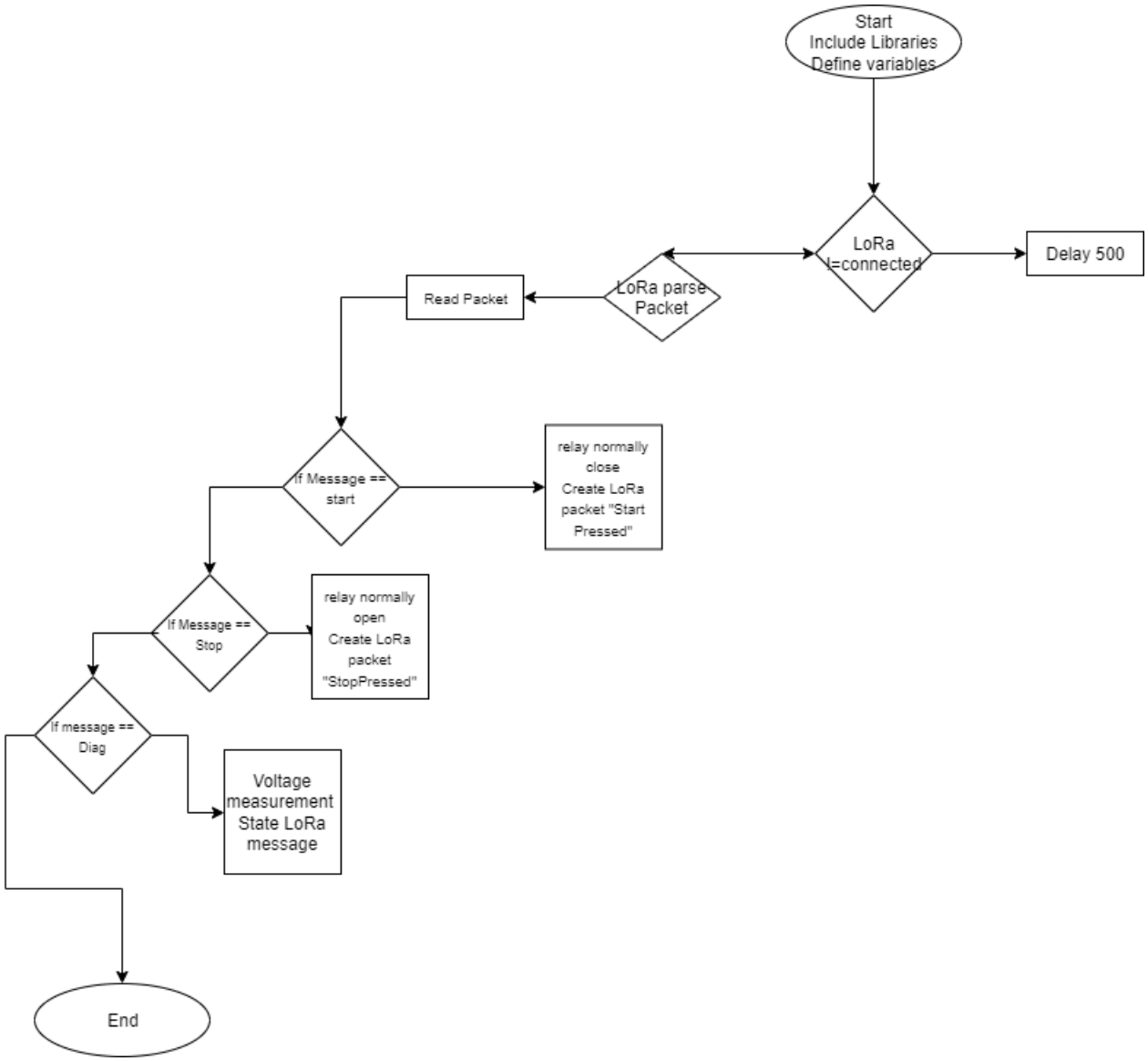

Figure 3.13: LoRa EndPoint Logical Diagram

3.4 NodeRED Schema

NodeRED is pretty plain and simple on creating a flow which can consists of many nodes. Nodes can be drag n dropped into the palette and in order to be connected it just need to be wired together through clicking. As it is projected on the flow the Schema consists of 4 buttons which are start, stop, diagnostics and email, 2 Mqtt topics, one Database which stores and displays information , 2 functions one for writing the email and one splitting the Mqtt input so that it can be stored right after to the Database, a template in order to visualize the table output and data, a helping text field which displays the last command executed, a csv node which creates and updates a csv file with data and lastly the Email Configurator node which does exactly as the name suggests and configures to who a diagnostic Email should be sent.

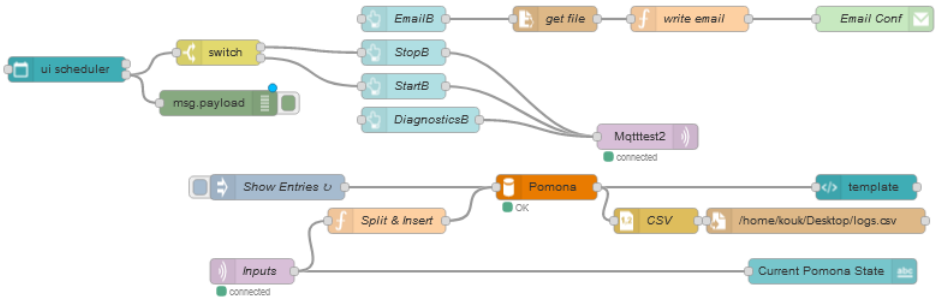

Figure 3.14: BackEnd Flow

3.4.1 Database Node

After a Database Node is dropped into the palette it must be configured in order to be working properly.Double clicking onto the Node displays the properties window. The parameters that should be configured are the IP address of host , the port in which the Database is listening to, the username of the user according with his password and the Database name. If different Charset was used into the database creation Charset field should be modified accordingly.

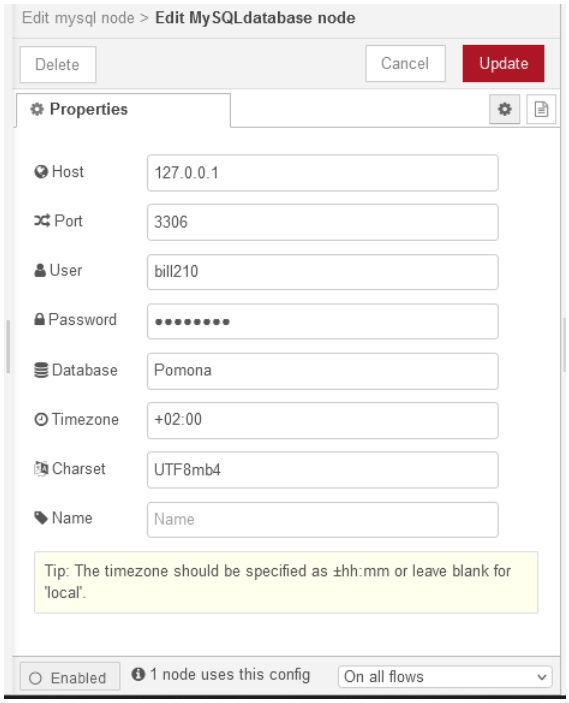

Figure 3.15: Database Configuration

3.4.2 Database Template

A template Node is inserted into the palette in order to provide a means to display Database data through a table by using HTML and CSS properties. The table consists of 5 headers which are ID,State, CurrentAC, Email Failover and Timestamp. For each message payload it fills its associated field on the table.

Figure 3.16: Database Template

3.4.3 Inject Node

An Inject Node is inserted into the palette in order to provide a more direct way to provide information and values to other Nodes. The payload that it is transfered can vary from values to objects to even SQL queries to be executed against a Database. An inject Node can like any Node inserted have a name. In the menu can be declared what type is needed to be transfered. It can vary from payload even to the whole topic and from the drop down menu by selecting a to z it is declared that the value will be in string type. In the text field the value is filled which is an SQL query ”select * from PomonaDashboard order by Id desc limit 50”. Below more options are available. Lastly the repeat option gives the choice of repeatable executions in a specific amount of time.

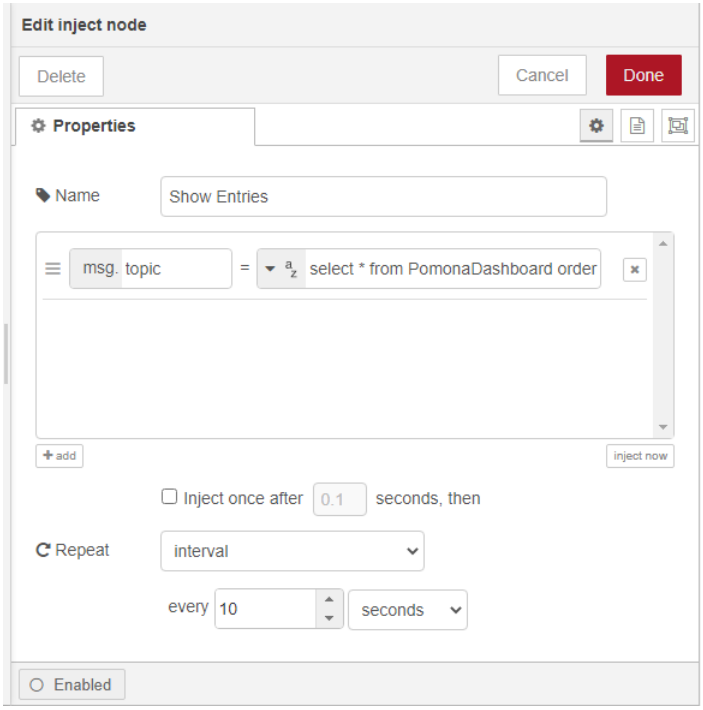

Figure 3.17: Inject Node

3.4.4 Functions

An Function Node is inserted into the palette in order to provide more complexity and distribution to the needs of the system. Functions are pretty simple in their structure. By double clicking to the Node it just opens an large text field which works as an editor. Inserting code inside of it provides execution according to the specific needs. Two functions were built for the system. Starting with the first one being ”Split and Insert”. In the editor code is typed so that stores the values came from the MQTT topic and upon receivable splits the payload into values whenever the char ”:” is found . Right after the split reaches the end of the payload a Query is created which inserts into every table column its corresponding value. The Second Function that was created is named ”Write Email”. This Function creates the values inserted into the Email. Starting with the Subject following with the text that will exist into the Email.Next it gets the attachment in which case it is a csv file named logs.Lastly a variable with Date synchronized to specific Timezone is declared so the information of when the Email was created is visible.

Figure 3.18: Split and Insert Function

Figure 3.19: Write Email Function

3.4.5 Email Node

Emails can be used in NodeRED so Email nodes are created.Email Nodes are not present by default so for that reason must be installed. From manage palette node-red-node-email should be typed and the install button should be pressed. Right after the installation is complete a restart of NodeRED is considered advisable. Nodes can be drag and dropped into the palette. In order for Emails to be sent some configuration should be done in advance. By double clicking to the Email Node the properties menu is displayed so that it can be configured. The first field is the recipients Email. Next is the server that the email is gonna be sent, Smtp.gmail.com is the one the gmails use. Next is the port declaration in which it is typed 465 for gmail[16]. The following 2 fields hold the Sender Credentials. The TLS field which is the next field is a good strategy to be checked. Lastly a Name for Node can be declared.

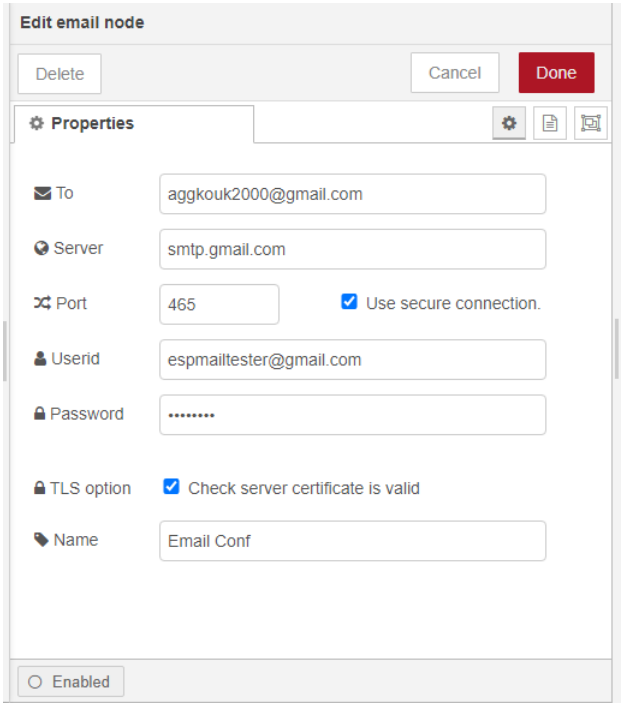

Figure 3.20: Email Node Configuration

3.4.6 Mqtt Input Node

Mqtt Nodes in order to work in NodeRED have a prerequisite. A Mqtt server must be installed first into the Raspberry pi. Mqtt server has its IP address and that is how a connection is made with this server. After a Mqtt input Node is dropped into the palette, it needs to configured so that it can work as expected. There are some fields that need to be filled. Starting with the Server which is the IP address of the Mqtt server in combination with the port which the default for Mqtt is 1883. Next from the drop down menu we click the option ”Subscribe to a single topic” so that it subscribes to the topic and waits for information. Next is the topic where it is declared the topic name in this Schema is Mqtttest from which the node is subscribed.

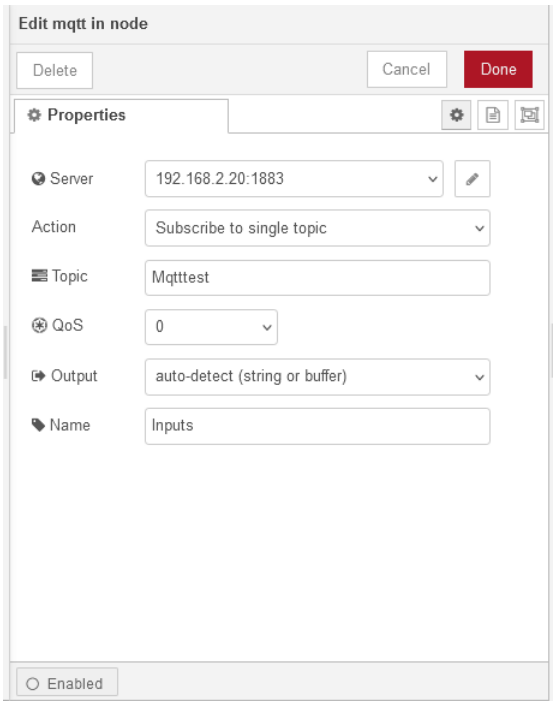

Figure 3.21: Mqtt Input Node

3.4.7 Mqtt Output Node

Much as a Mqtt Input so as the Output a Prerequisite is the same. A Mqtt server must be installed first into the Raspberry pi. Mqtt server has its IP address and that is how a connection is made with this server. When a Mqtt Output Node is dropped into the palette, it needs its configuration so that it can work as expected. The fields that need to be filled are, a Server field which is the IP address in combination with the port, which the default for Mqtt is 1883. Next is the topic where it is declared the topic name Mqtttest2 from which the node is publishing information. QoS and retain are more optional fields which provide a more reliant transmission of the published message.

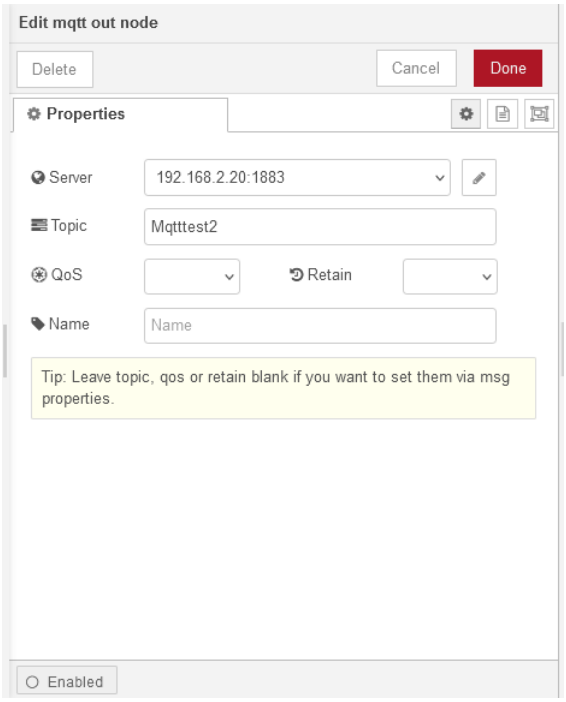

Figure 3.22: Mqtt Output Node

3.5 UI Scheduler

NodeRED is Open Source and developed in NodeJs. Having your code and flows publicly available means that more people depending on their needs can use it to meet their demands. Moreover brings plurality because more and more variants and solutions will come up to the surface given time from a specific project,flow to even something completely different. One Node developed and it is used in this system is ”UI-time-scheduler”. It can be installed with 2 ways. First by directly using the ”Manage Palette” menu in the Node-RED interface or alternatively on the NodeRED install file by executing the command ”npm install node-red-contrib-ui-time-schedule”[19]. It can be accessed on the Dashboard by clicking the plus button and setting the desired times that the user may want to set. In system functionality provides more diversity in options because it gives user the option to declare commands on specific period of time or even repeated events without being present and declaring the commands by hand on the ”NodeRED DashBoard UI”.

3.6 NodeRED DashBoard UI

NodeRED Dashboard UI is the representing deployment of the BackEnd and the nodes deployed into the palette. From the Schema 4 buttons are deployed Stop,Email,Start,Diagnostics. Each button follows its certain path and executes its respective programmed commands.From the Template Node a Table is presented filled with data which were queried from the ”Pomona” Table on Pomona Database. A Textfield presents the state of the system if it is started,stopped or a Diagnostic button was issued.Lastly from the palette is deployed and presented an UI Scheduler in which times can be set so that actions can take place like Start,Stop [20].

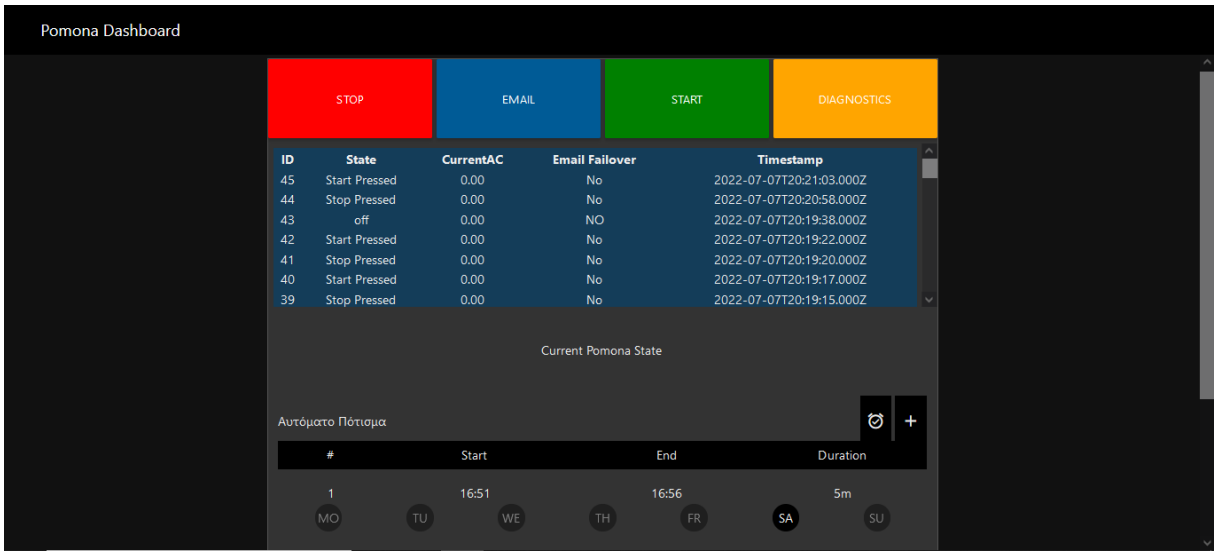

Figure 3.23: NodeRED DashBoard UI

3.7 phpMyAdmin

phpMyAdmin is a management open source code tool capable of managing Databases through ”MySQL” and ”MariaDB”. It provides portable web application written mostly in PHP.The main purpose of phpMyAdmin is to handle the administration of MySQL over the web. It is one if not the most popular applications for creating, updating, drop, alter, delete, import, export for MySQL database tables by using this software. PhpMyAdmin also supports a wide range of operations like managing databases, relations to tables, columns, indexes, permissions, users, etc both on MySQL and MariaDB. These operations can be performed managing the user interface, while we still have are in the position to execute any SQL statement. It provides a web-based interface and can run to almost any server. Since it is web-based, it can be accessed from any computer [22] [28].

3.7.1 Installing phpMyAdmin

In order to install phpMyAdmin some steps need to be executed. Starting with step one being installing the package with the command ”sudo apt install phpmyadmin”. After this command is executed phpMyAdmin will start to install to Raspberry Pi. Some inputs are required for the installation to proceed. Next ”apache2” is selected and in the following window asking about the connection of phpMyAdmin to MYSQL server ”Yes” is selected. A password is asked as the next step for phpMyAdmin itself. Note that this password is gonna be used to connect to the ”MYSQL”. Right after that the installation process is complete. In this condition phpMyAdmin is installed and has one user the one called root. It is highly recommended to to create a user aside from root. With that principle in mind the following command is issued in order to create a user ”sudo mysql -u root -p”. Moreover one command is issued so that the new created user can access all the databases and have permissions on them ”GRANT ALL PRIVILEGES ON *.* TO ’username’@’localhost’ IDENTIFIED BY ’password’ WITH GRANT OPTION;”. Lastly so that we can load phpMyAdmin interface the apache configuration file should be modified. The command ”sudo nano /etc/apache2/apache2.conf” in order for this step to be achieved. The next command ”Include

/etc/phpmyadmin/apache.conf" is being added to the bottom of the config file , it is declared that the phpmMyAdmin configuration of "apache" is added to the "apache" configuration. Lastly executing the command "sudo service apache2 restart" we restart the apache server in order for the new configuration takes place. PhpMyadmin can now be accessed through a web browser just typing the Raspberry Pi IP address followed by a slash and adding phpmyadmin in the end of the URL. The login page is gonna appear connecting to <ip address>/phpmyadmin/index.PHP to the user asking for its credentials. Logging is done by providing Username and Password in its corresponding field[23].

Figure 3.24: phpMyAdmin Login Screen

3.7.2 Database and Tables

Considering phpMyAdmin has been installed properly and the user has been authenticated by its corresponding credentials the user is in front the main page. User selects Databases from the menu. In the following create database field users types a name for the database that wants to be created. By clicking create the Database is created. On the left pane the Database should be visible , clicking on it will load this database and its fields [23]. A Table is then created with the following fields as its columns. Starting with an auto increment field named "ID" which serves as a primary key to this table and uniquely identifies each row inserted into the table with an increment number. Second field's name is "State" structured as Varchar type and it will be used in order to store and display the Status that were defined ,in most cases "on/off" or "Start/Stop Pressed".Third is a Varchar type field named "CurrentAC" and on it, it is stored and displayed the value in double format of the current that was sent as information from the LoRa Endpoint. The next field is named "Email". This field is also Varchar and can have values "yes/no" and the usage is to provide the information if the user is informed with an Email if the system made an action. Lastly a field named "Timestamp" with type datetime holds the current time that the command was inserted into the table. For State, Currentac, Email Null attribute is enabled.

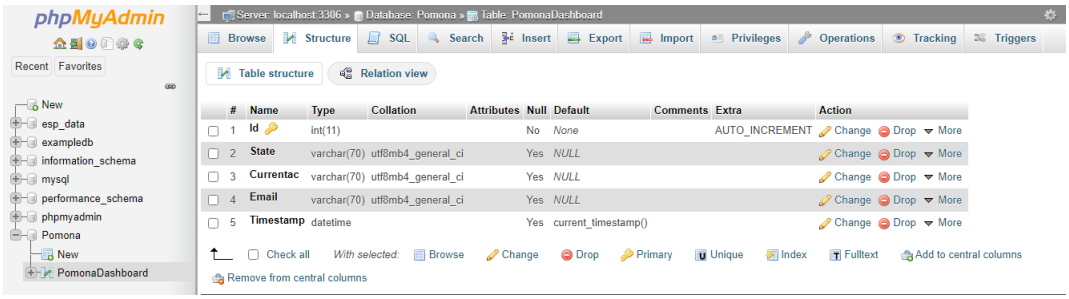

Figure 3.25: Pomona Table

3.8 ZeroTier

ZeroTier is a software that combines the capabilities of VPN and SD-WAN and by doing that it can simplify the network management. It is hardware so as software independant, and can be setup really fast even in minutes with remote and automated deployment. Another benefit is it provides flexibility because it is in position to emulate layer 2 Ethernet with multipath multicast and bridging. Security in one of its stronger features because provides scalable security with 256-bit end-to-end encryption [24].

3.8.1 ZeroTier Install

ZeroTier can be accessed through web so an account needs to be created. User can provide authentication by using its email from google, Microsoft, GitHub or by just creating an account on the main screen by pressing register. After the account creation or login user is displayed the main page where a button named ”Create A Network” is visible. Pressing the button Creates a network with a name , with a unique NetworkID,subnet, nodes that are connected and the created field which indicates when the network was created [24] [25]. Heading back to Raspberry Pi the command that should be executed in order to get the repository containing the ZeroTier Package is ***curl*** *https://raw.githubusercontent.com/zerotier/ZeroTierOne/master/doc/contact40zerotier.com.gpg* | ***gpg*** *–dearmor* | ***sudo*** *tee /usr/share/keyrings/zerotierone-archive-keyring.gpg >/dev/null*. Next the *$RELEASE = (lsb_release -cs)* is executed so the OS shell Variable is stored. Afterwards, the repository is added through the command ***echo*** *deb[signed-by=/usr/share/keyrings /zerotierone-archive-keyring.gpg http://download.zerotier.com/debian/$RELEASE $RELEASE main* | ***sudo*** *tee /etc/apt/sources.list.d/zerotier.list*. The next command is the installation of ZeroTier. By executing ***sudo*** *apt install zerotier-one*. After the installation is complete client which in case is Raspberry Pi should join the previously connected Network. That is accomplished by executing ***sudo*** *zerotier-cli join <NETWORKID>*. Last step is necessary for security reasons. Although the device has successfully joined the network it must be authenticated, so heading back to the web to the address ”https://my.zerotier.com/network/<NETWORKID>”, the user scrolls down until finds the device and clicks the auth checkbox. Connection can be checked that it is established and working by entering the command ***sudo*** *zerotier-cli status* and ***sudo*** *zerotier-cli listnetworks* for checking what networks are connected to [26].

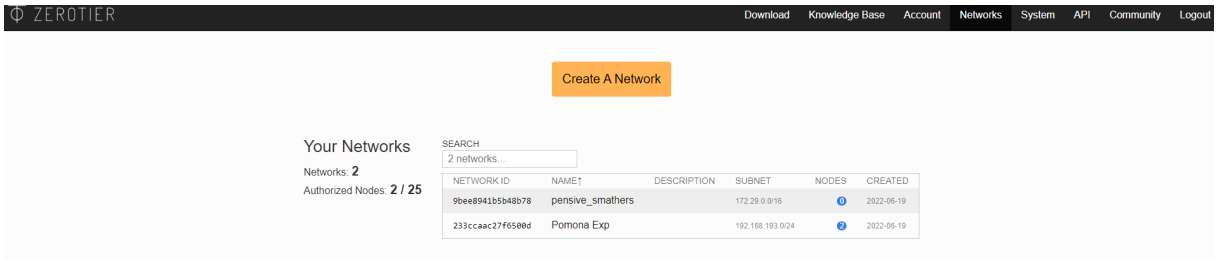

Figure 3.26: ZeroTier Networks

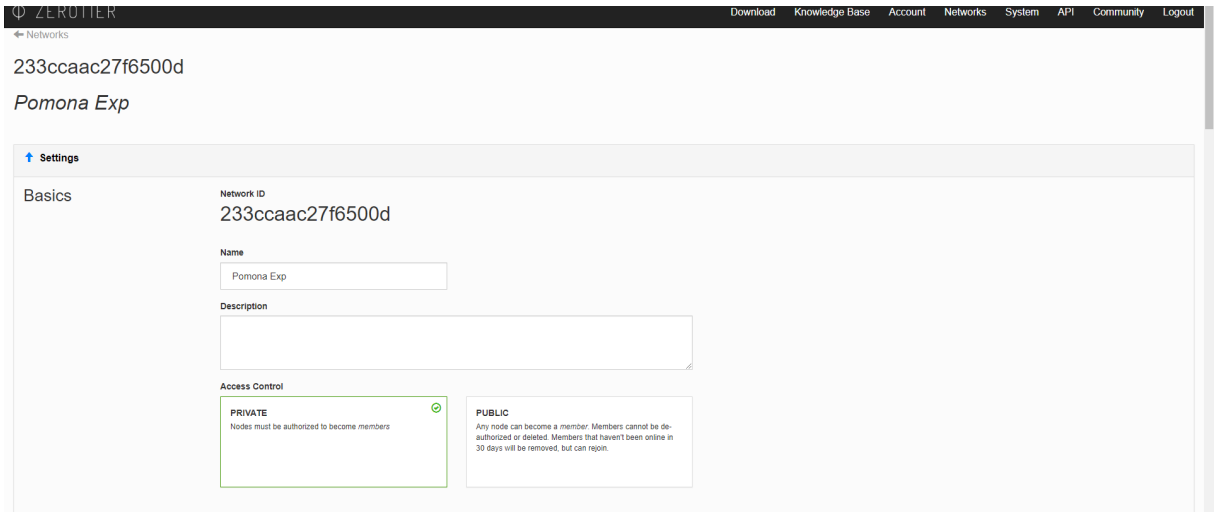

Figure 3.27: ZeroTier Network Configuration

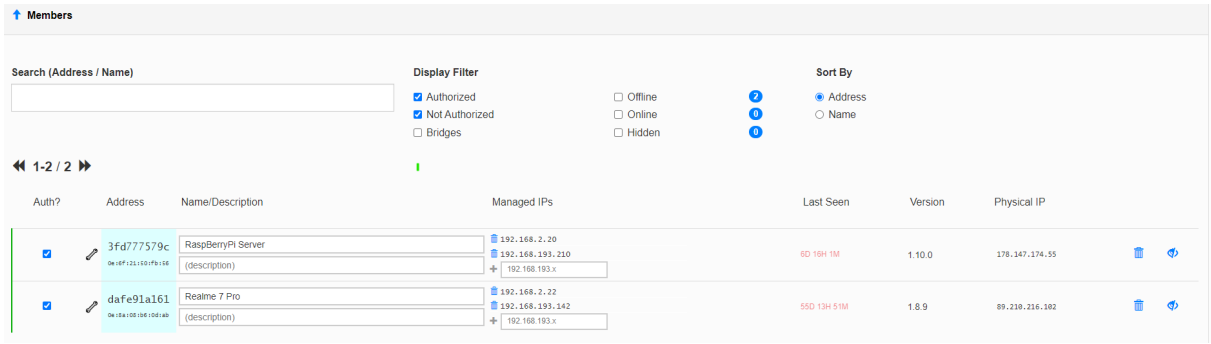

Figure 3.28: ZeroTier Devices

Chapter 4: Conclusion

4.1 Strong Points

Summarizing things up building a system can be tricky. Multiple variables are entangled with each other so that the best solution may arise. It is easy understandable that a well crafted plan of construction and its parameters are useful. For that reason Logical diagrams were created so the picture of which module, micro controllers or even Servers like Raspberry Pi are connected to Each other. Another Crucial aspect is for the correct materials to be selected. In this system needs a variety of materials are connected. It's strong points are that they are really flexible .For example almost any module with jump wires can be connected with a "NodeMCU 8266" or wireless through WiFi or Bluetooth which in this system case is not used. Assembling this system became easier with a solid Backend which can be used from one to many systems providing scalability to larger needs. Schedulers , buttons ,functions can work with the same outcome independently from the number of Endpoints connected to the Backend. Cost is another factor that usually plays its role in development and creation of systems. This system cost is relatively affordable within reach almost for everyone. With current pricing the system can be build with less than one hundred euros while almost every solution available in market is can be acquired with more than 150 euros.Another cost factor that can be tricky especially in the long term is the the SIM provider cost in order for SMS or call services to be provided. SIM numbers should be regularly topped up their balance in cases of expiring or if the balance is depleted. LoRa is free with no costs attached aside from the purchase and connection of modules. The next is Maintenance which is another huge factor affecting long term reliability and access to the construction of the system. Market solutions doesn't provide easy replacement of a broken module in contrast with the system in which most of the modules can be referred as "plug n play". Lastly power redundancy secures the continuous up time of the system while exceeds in time most if not all market solutions available.

4.2 Weak Points

Nobody is perfect . It is easily understandable that this system makes no exception. Unlike "GSM" solutions which with the infrastructure that was developed though the years it gives the feeling that has unlimited range and coverage. On the other hand using LoRa for the time this takes place has some restrictions. Although it provides uninterrupted signal transmission into the Sub GHz spectrum for ranges up to kilometres , its infrastructure due to recency is not on the same level as the ones of "GSM" which in term makes WAN LoRa connections much more difficult to be achieved in hundreds of kilometres of distance.Thus giving the feeling that it is not unlimited. System is designed for a point to point connection which is a different design of connection from WAN but the same restrictions occur for both WAN and point to point connections.

4.3 Overall

Overall this system is recommended for long term usage because it provides reduced costs compared to GSM solutions more flexibility in implementation and bigger redudunccy because of the battery capacity towards consumption ratio. Can be used to single or with many Endpoints providing scalability. One

of the things that should be considered is the distance between the Node transceiver and the Endpoint because it heavily relies on the Antennas used on each Node respectively.

Chapter 5: Conclusions and Suggestions for Improvement

Every System has room for development from bug fixing where abnormalities to user experience occur to adding features functionalities and providing better and more efficient capabilities.

5.1 Telegram Chat Bot

Telegram is a channel message platform just as messenger , Whatsup , Singal and many more. A great feature which can provide another channel of communication to the system is the addition of an Telegram Bot Feature to the Back end of the System. The immediate benefit will be that user can have the choice of accessing his system and declaring actions in the level of functionality though a multi million user chat app which works in harmony with the backbone of NodeRED which system is Developed. A possible use case can start from the User typing the command and a reply message can be received with the result of the action. Telegram is sincerely secure and a reliable channel with little close to zero downtime. So with that perspective in mind great availability can be achieved through secure channel.

5.2 Addition of Relays

Another addition which can be easily implemented is the addition of Relays .Relays are generally cheap and easy in the respective installation. Installing more Relays bring the added benefit of having multiple switches to the system .This in term provides bigger coverage in functionality with multiple On/offs. Choices include 2,4,8 or even 16 relays.

5.3 Improved Statistics

It is Important for every user of a system to have a clean and transparent image of the system that he is managing. Transparency brings much better control and improves by a large margin the efficiency of almost every system. Metrics of the time Spent On/Off and wattage consumption can provide a better outlook of costs and predictions of a good working systems. The display of those could be achieved though Nodes in the Back-end of NodeRED for example a gauge Node can easily be installed and used easily.

5.4 Login Credential System

Security is a huge sector of systems working as intended properly and effectively. Unauthorized access can cause serious trouble and even costs to users when usage is abnormal. A feature that is almost a must to every system is user Authentication and Authorization. A reliable and resilient means to provide these two combined is a login system. Login systems are implemented with a home welcome page with text boxes where a user can fulfill its credentials. Information provided by the User is matched against the Database information and if the Credentials much one of a legit user Login page is passed and the system with the allowed features is ready to use.

Chapter 6: Bibliography

References

[1] https://el.wikipedia.org/wiki/

[2] https://www.neropoulos.gr/tilechirismos-meso-kinitou-tilefonou-gsm-ragas-n2016r/

[3] https://plantron.gr/index.php/gsm-s130-3g-4g-sms-remote-controller-alarm-2din-2do-usb.html?ref=bestprice.gr__bpgid=MW9BVVM4T2VhK1gsMUt+TGRtSGZoQmo=

[4] https://www.smart-akis.com/index.php/el/network-el/what-is-smart-farming-el/

[5] https://www.techtarget.com/iotagenda/definition/smart-farming

[6] https://create.arduino.cc/projecthub/electropeak/getting-started-w-nodemcu-esp8266-on-arduino-ide-28184f

[7] http://wiki.sunfounder.cc/index.php?title=LCD1602_Module

[8] https://grobotronics.com/lcd-display-i2c-interface-module.html

[9] https://www.hellasdigital.gr/electronics/sensors/current-sensors/hall-current-sensor-module-acs712-20a-model-for-arduino/

[10] https://en.wikipedia.org/wiki/Raspberry_Pi

[11] https://www.hivemq.com/blog/mqtt-vs-http-protocols-in-iot-iiot/

[12] https://randomnerdtutorials.com/esp32-lora-rfm95-transceiver-arduino-ide/

[13] https://how2electronics.com/lora-sx1278-esp8266-transmitter-receiver/

[14] https://flows.nodered.org/node/node-red-node-mysql

[15] https://nodered.org/docs/user-guide/messages

[16] https://flows.nodered.org/node/node-red-node-email

[17] https://appcodelabs.com/introduction-to-iot-build-an-mqtt-server-using-raspberry-pi

[18] http://stevesnoderedguide.com/configuring-the-mqtt-publish-node

[19] https://flows.nodered.org/node/node-red-contrib-ui-time-scheduler/in/590bc13ff3a5f005c7d2189bbb563976

[20] https://flows.nodered.org/node/node-red-dashboard

[21] https://docs.phpmyadmin.net/en/latest/setup.html

[22] https://www.cloudways.com/blog/install-phpmyadmin/

[23] https://wpengine.com/support/run-query-phpmyadmin/

[24] https://zerotier.atlassian.net/wiki/spaces/SD/pages/29065282/Command+Line+Interface+zerotier-cli

[25] https://zerotier.atlassian.net/wiki/spaces/SD/pages/6848513/Join+a+Network

[26] Rui Santos , Sara Santos ”Ultimate Guide for Arduino Sensors/Modules”

[27] Rui Santos , Sara Santos ”ESP 8266 WEB SERVER WITH ARDUINO IDE”

[28] Marc Delisle , ”Mastering phpMyAdmin 3.4 for Effective MySQL Management”

[29] Matt Richardson (Author), Shawn Wallace (Author) , ”Getting Started with Raspberry Pi (Make: Projects) ”

[30] Brian W. Kernighan (Author), Dennis M. Ritchie (Author) , ”C Programming Language, 2nd Edition”

[31] Jonathan Wexler (Author), Get Programming with Node.js

[32] Kraig Mitzner (Author), ”Complete PCB Design Using OrCAD Capture and PCB Editor”

[33] Simon Monk (Author) ,”Programming Arduino : Getting Started with Sketches , Second Edition”

[34] Neil Cameron (Author), ”Electronics Projects with the ESP8266 and ESP32: Building Web Pages, Applications, and WiFi Enabled Devices”

Chapter A: LoRa Sender

```
//include the required Libs
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
#include <WiFiClient.h>
#include <SPI.h>
#include <LoRa.h>
#include <PubSubClient.h>
#include <ESP_Mail_Client.h>


//define the pins used by the transceiver module
#define ss 2
#define rst 0
#define dio0 4

// Network SSID & PASSWORD
const char* ssid     = "KoukHome";
const char* password = "echost0rm210koukoS2";
// RASPBERRYPI ADDRESS & MQTT SERVER ADDRESS & Topics
const char* mqtt_server = "192.168.2.20";
//const char* mqtt_server = "192.168.193.210";
char*  Topic = "Mqtttest" ;
char*  Topic2 = "Mqtttest2" ;

//variables for subdeviding the Received Lora Packet
String LoRaData = "";
String LoRaInput = "";
String LoRaState = "";
String LoRaCurrent = "";
String LoRaEmail = "";
String LoRaTimeStamp = "";

// Initializes the espClient
WiFiClient espClient;
PubSubClient client(espClient);
char msg[70];
//char msgState[10];
//char msgCurrent[10];
//char msgEmail[30];
//char msgTimeStamp[30];
int value = 0;
String payload = "";

//initializes the SMTP
#define SMTP_HOST "smtp.gmail.com"
#define SMTP_PORT 465

/* The sign in credentials */
#define AUTHOR_EMAIL "espmailtester@gmail.com"
#define AUTHOR_PASSWORD "ycmgmloilaheowpx"

```

```
/* Recipient's email*/
#define RECIPIENT_EMAIL "aggkouk2000@gmail.com"

/* The SMTP Session object used for Email sending */
SMTPSession smtp;


// This functions connects your ESP8266 to your router
void setup_wifi() {
  delay(10);
  // We start by connecting to a WiFi network
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("WiFi connected - ESP IP address is : ");
  Serial.println(WiFi.localIP());
}

//function running on loop checking and storing messages received from mqtt
void callback(String topic, byte* message, unsigned int length) {
  Serial.print("Message arrived on topic: ");
  Serial.print(topic);
  Serial.print(". Message: ");
  String messageTemp;

  for (int i = 0; i < length; i++) {
    Serial.print((char)message[i]);
    messageTemp += (char)message[i];
  }
  Serial.println();


  // If a message is received on topic Mqtttest and message is matching send Lora
    packet!
  if(topic=="Mqtttest2" && messageTemp=="Start"){
      Serial.println("Sending Start Message to Lora Endpoint");
      LoRa.beginPacket();
      LoRa.print("Start");
      LoRa.endPacket();
      }
      else if(topic=="Mqtttest2" && messageTemp=="Stop"){
        Serial.println("Sending Stop Message to Lora Endpoint");
        LoRa.beginPacket();
        LoRa.print("Stop");
        LoRa.endPacket();
      }
      else if(topic=="Mqtttest2" && messageTemp=="Diagnostics"){
```

```
104         Serial.println("Sending Diagnostics Message to Lora Endpoint");
105         LoRa.beginPacket();
106         LoRa.print("Diagnostics");
107         LoRa.endPacket();
108       }
109
110 }
111 //function for reconnecting to Mqtt
112 void reconnect() {
113   // Loop until we're reconnected
114   while (!client.connected()) {
115     Serial.print("Attempting MQTT connection...");
116     // Attempt to connect
117     if (client.connect("ESP8266Client")) {
118       Serial.println("connected");
119       // Subscribe or resubscribe to a topic
120       client.subscribe("Mqtttest");
121       client.subscribe("Mqtttest2");
122     } else {
123       Serial.print("failed, rc=");
124       Serial.print(client.state());
125       Serial.println(" try again in 5 seconds");
126       // Wait 5 seconds before retrying
127       delay(5000);
128     }
129   }
130 }
131
132 //fucntion Deviding Lora Received message
133 void devide(){
134   LoRaInput = LoRaData.substring(1,6);
135   Serial.println(LoRaInput);
136   LoRaState = LoRaData.substring(17,19);
137   Serial.println(LoRaState);
138   LoRaCurrent = LoRaData.substring(28,34);
139   Serial.println(LoRaCurrent);
140   //LoRaTimeStamp = LoRaData.substring(46,55);
141   //Serial.println(LoRaTimeStamp);
142   LoRaEmail = LoRaData.substring(58,61);
143   Serial.println(LoRaEmail);
144
145 }
146
147 //function publishing if Lora message Received is input for the database table
148 void Input_check(){
149   if(LoRaInput=="Input"){
150     Serial.println("Publishing to be stored in DB");
151     //LoRaState.toCharArray(msgState, 10);
152     //LoRaCurrent.toCharArray(msgCurrent, 10);
153     //LoRaEmail.toCharArray(msgEmail, 30);
154     //LoRaTimeStamp.toCharArray(msgTimeStamp, 30);
155     LoRaData.toCharArray(msg, 70);
156     if (client.publish(Topic,msg)) {
```

```
157       Serial.println("Publish ok");
158     }
159     else {
160       Serial.println("Publish failed");
161     }
162     //delay(20000);
163  }
164 }
165
166 void Email_Failover(){
167  if(LoRaEmail== "YES"){
168
169   /** Enable the debug via Serial port
170    * none debug or 0
171    * basic debug or 1
172   */
173   smtp.debug(1);
174
175     /* Declare the session config data */
176   ESP_Mail_Session session;
177
178   /* Set the session config */
179   session.server.host_name = SMTP_HOST;
180   session.server.port = SMTP_PORT;
181   session.login.email = AUTHOR_EMAIL;
182   session.login.password = AUTHOR_PASSWORD;
183   session.login.user_domain = "";
184
185   /* Declare the message class */
186   SMTP_Message message;
187
188   /* Set the message headers */
189   message.sender.name = "Pomona Dashboard";
190   message.sender.email = AUTHOR_EMAIL;
191   message.subject = "Pomona Check";
192   message.addRecipient("bill", RECIPIENT_EMAIL);
193
194
195   //Send raw text message
196   String textMsg = "Hey its Pomona Diag Here! Pomona just stopped..checking
        Dashboard would be recommended! ";
197   message.text.content = textMsg.c_str();
198   message.text.charSet = "us-ascii";
199   message.text.transfer_encoding = Content_Transfer_Encoding::enc_7bit;
200
201   message.priority = esp_mail_smtp_priority::esp_mail_smtp_priority_low;
202   message.response.notify = esp_mail_smtp_notify_success |
        esp_mail_smtp_notify_failure | esp_mail_smtp_notify_delay;
203
204
205   /* Connect to server with the session config */
206   if (!smtp.connect(&session))
207     return;
```

```
208
209   /* Start sending Email and close the session */
210   if (!MailClient.sendMail(&smtp, &message))
211     Serial.println("Error sending Email, " + smtp.errorReason());
212   LoRaEmail = "NO";
213   Serial.println(LoRaEmail);
214  }
215 }
216
217
218 void setup() {
219
220
221   //initialize Serial Monitor
222   Serial.begin(115200);
223   setup_wifi();
224   client.setServer(mqtt_server, 1883);
225   client.setCallback(callback);
226   while (!Serial);
227   Serial.println("LoRa Sender");
228
229   //setup LoRa transceiver module
230   LoRa.setPins(ss, rst, dio0);
231
232   //replace the LoRa.begin(---E-) argument with your location's frequency
233   //433E6 for Asia
234   //866E6 for Europe
235   //915E6 for North America
236   while (!LoRa.begin(866E6)) {
237     Serial.println(".");
238     delay(500);
239   }
240    // Change sync word (0xF3) to match the receiver
241   // The sync word assures you don't get LoRa messages from other LoRa transceivers
242   // ranges from 0-0xFF
243   LoRa.setSyncWord(0xF3);
244   Serial.println("LoRa Initializing OK!");
245
246
247 }
248
249 void loop() {
250
251   // put your main code here, to run repeatedly:
252   if (!client.connected()) {
253     reconnect();
254   }
255   if(!client.loop())
256
257   client.connect("ESP8266Client");
258
259     // try to parse packet
260   int packetSize = LoRa.parsePacket();
```

```
261   if (packetSize) {
262     // received a packet
263     Serial.print("Received packet '");
264
265     // read packet
266     while (LoRa.available()) {
267       LoRaData = LoRa.readString();
268       Serial.print(LoRaData);
269     }
270
271     // print RSSI of packet
272     Serial.print("' with RSSI ");
273     Serial.println(LoRa.packetRssi());
274
275     devide();
276     Input_check();
277     Email_Failover();
278
279
280   }
281
282 }
```

Listing 1: Example of the LoRa Sender

Chapter B: LoRa EndPoint

```

#include <LoRa.h>
#include <SPI.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include "ACS712.h"

#define relayPin 10

//define LoRa Pins
#define ss 2
#define rst 0
#define dio0 16

String State = "off";
String EmailFailover = "NO";
double Currentac = 0;

//pinMode(relayPin, OUTPUT);    // sets the digital pin 10 as output

// initialize the LCD library with I2C address and LCD siz
LiquidCrystal_I2C lcd (0x27, 16, 2);

// ACS712 30A uses  66 mV per A
ACS712  ACS(A0, 5.0, 1023, 66);
volatile int mA;

volatile int interrupts;

// Timer Interrupt Function, called each second
void ICACHE_RAM_ATTR onTime() {
  //interrupts++;
  //Serial.print("Total Interrupts: ");
  //Serial.println(interrupts);
  //
  mA = ACS.mA_AC();
  Serial.print("mA: ");
  Serial.print(mA);
  Serial.print(". Form factor: ");
  Serial.println(ACS.getFormFactor());
  timer1_write(5000000);


}

void setup() {
  //digitalWrite(relayPin,LOW);
  //initialize Serial Monitor
  Serial.begin(115200);

```

```
52   //Setup Timer Interrupt
53   timer1_attachInterrupt(onTime); // Add ISR Function
54   timer1_enable(TIM_DIV16, TIM_EDGE, TIM_SINGLE);
55   timer1_write(5000000);
56
57   // Initialize the LCD connected
58   lcd.begin();
59   lcd.clear();
60   lcd.print("    LoRa  Pomona");
61   lcd.setCursor(0, 1);
62   lcd.print(" LoRa Initiallizing");
63
64   //setup LoRa Transceiver
65   LoRa.setPins(ss, rst, dio0);
66   //Europe's Freq is 868MHz
67   while (!LoRa.begin(866E6)) {
68     Serial.println(".");
69     delay(500);
70   }
71   // ranges from 0-0xFF
72   LoRa.setSyncWord(0xF3);
73   Serial.println("LoRa Initializing OK!");
74   lcd.setCursor(0, 2);
75   lcd.print("     LoRa  OK");
76
77   //ACS Initiallize
78   ACS.autoMidPoint();
79   Serial.print("MidPoint: ");
80   Serial.print(ACS.getMidPoint());
81   Serial.print(". Noise mV: ");
82   Serial.println(ACS.getNoisemV());
83
84
85 }
86
87 void loop() {
88   // try to parse packet
89   int packetSize = LoRa.parsePacket();
90   if (packetSize) {
91     // received a packet
92     Serial.print("Received packet '");
93
94     // read packet
95     while (LoRa.available()) {
96       String LoRaData = LoRa.readString();
97       //Serial.print(LoRaData);
98       if (LoRaData == "Start") {
99         Serial.print(LoRaData);
100        LoRa.beginPacket();
101        LoRa.print(" Input : ");
102        LoRa.print(" State : ");
103        LoRa.print(" Start Pressed");
104        LoRa.print(": Current : ");
```

```
        LoRa.print(Currentac);
        LoRa.print(": Email Failover : ");
        LoRa.print("No");
        LoRa.endPacket();
        digitalWrite(relayPin, HIGH);
        Serial.print(" Current flow");
      }
      else if (LoRaData == "Stop") {
        Serial.print(LoRaData);
        LoRa.beginPacket();
        LoRa.print(" Input : ");
        LoRa.print(" State : ");
        LoRa.print(" Stop Pressed");
        LoRa.print(": Current : ");
        LoRa.print(Currentac);
        LoRa.print(": Email Failover : ");
        LoRa.print("No");
        LoRa.endPacket();
        digitalWrite(relayPin, LOW);
        Serial.print(" Current not flowing");
      }
      else if (LoRaData == "Diagnostics") {
        Serial.print(LoRaData);
        LoRa.beginPacket();
        LoRa.print(" Input : ");
        LoRa.print(" State : ");
        LoRa.print(State);
        LoRa.print(": Current : ");
        LoRa.print(Currentac);
        LoRa.print(": Email Failover : ");
        LoRa.print(EmailFailover);
        LoRa.endPacket();
      }


      Serial.print("' with RSSI ");
      Serial.println(LoRa.packetRssi());
    }
  }
}
```

Listing 2: Example of the LoRa EndPoint

Chapter C: NodeRED Backend


```
[
    {
        "id": "65de697399c108a4",
        "type": "tab",
        "label": "Flow 1",
        "disabled": false,
        "info": "[{\"id\":\"599740b7.efde9\",\"type\":\"http response\",\"z\":\"
    b01416d3.f69f38\",\"name\":\"\",\"statusCode\":\"200\",\"headers\":{},\"x
    \":420,\"y\":689,\"wires\":[]},{\"id\":\"1618a829.76f638\",\"type\":\"json\",\"z
    \":\"b01416d3.f69f38\",\"name\":\"\",\"property\":\"payload\",\"action\":\"obj
    \",\"pretty\":true,\"x\":410,\"y\":809,\"wires\":[[\"d0089cc7.d25ac\"]]},{\"id
    \":\"c7410fa2.1c2fa\",\"type\":\"debug\",\"z\":\"b01416d3.f69f38\",\"name
    \":\"\",\"active\":true,\"tosidebar\":true,\"console\":false,\"tostatus\":false
    ,\"complete\":\"false\",\"x\":850,\"y\":709,\"wires\":[]},{\"id\":\"75a22f74.
    f1aba\",\"type\":\"ui_text\",\"z\":\"b01416d3.f69f38\",\"group\":\"2b7ac01b.
    fc984\",\"order\":1,\"width\":0,\"height\":0,\"name\":\"\",\"label\":\"Sensor
    Name\",\"format\":\"{{msg.payload}}\",\"layout\":\"row-spread\",\"x\":860,\"y
    \":769,\"wires\":[]},{\"id\":\"1c8f9093.8bc2bf\",\"type\":\"ui_gauge\",\"z\":\"
    b01416d3.f69f38\",\"name\":\"\",\"group\":\"2b7ac01b.fc984\",\"order\":2,\"width
    \":0,\"height\":0,\"gtype\":\"gage\",\"title\":\"Temperature\",\"label\":\"ºC
    \",\"format\":\"{{value}}\",\"min\":0,\"max\":\"38\",\"colors\":[\"#00b500\",\"#
    e6e600\",\"#ca3838\"],\"seg1\":\"\",\"seg2\":\"\",\"x\":850,\"y\":829,\"wires
    \":[]},{\"id\":\"a5bd2706.54e108\",\"type\":\"ui_gauge\",\"z\":\"b01416d3.f69f38
    \",\"name\":\"\",\"group\":\"2b7ac01b.fc984\",\"order\":3,\"width\":0,\"height
    \":0,\"gtype\":\"gage\",\"title\":\"Humidity\",\"label\":\"%\",\"format\":\"{{
    value}}\",\"min\":0,\"max\":\"100\",\"colors\":[\"#0080ff\",\"#0062c4\",\"#002
    f5e\"],\"seg1\":\"\",\"seg2\":\"\",\"x\":840,\"y\":889,\"wires\":[]},{\"id
    \":\"105ac2cc.7b3cfd\",\"type\":\"ui_gauge\",\"z\":\"b01416d3.f69f38\",\"name
    \":\"\",\"group\":\"2b7ac01b.fc984\",\"order\":4,\"width\":0,\"height\":0,\"
    gtype\":\"gage\",\"title\":\"Pressure\",\"label\":\"hPa\",\"format\":\"{{value
    }}\",\"min\":0,\"max\":\"1200\",\"colors\":[\"#b366ff\",\"#8000ff
    \",\"#440088\"],\"seg1\":\"\",\"seg2\":\"\",\"x\":840,\"y\":949,\"wires
    \":[]},{\"id\":\"d0089cc7.d25ac\",\"type\":\"function\",\"z\":\"b01416d3.f69f38
    \",\"name\":\"JSON or URL Encoded\",\"func\":\"var msg0 = { payload: msg.payload
    .api_key };\\nvar msg1 = { payload: msg.payload.sensor };\\nvar msg2 = { payload
    : msg.payload.value1 };\\nvar msg3 = { payload: msg.payload.value2 };\\nvar msg4
     = { payload: msg.payload.value3 };\\n\\nreturn [msg0, msg1, msg2, msg3, msg4
    ];\",\"outputs\":5,\"noerr\":0,\"x\":610,\"y\":809,\"wires\":[[\"c7410fa2.1c2fa
    \"],[\"75a22f74.f1aba\"],[\"1c8f9093.8bc2bf\"],[\"a5bd2706.54e108\"],[\"105ac2cc
    .7b3cfd\"]]},{\"id\":\"5d9ab0d2.66b92\",\"type\":\"http in\",\"z\":\"b01416d3.
    f69f38\",\"name\":\"\",\"url\":\"update-sensor\",\"method\":\"post\",\"upload\":
    false,\"swaggerDoc\":\"\",\"x\":200,\"y\":740,\"wires\":[[\"599740b7.efde9\",\"
    c7410fa2.1c2fa\",\"1618a829.76f638\"]]},{\"id\":\"7f5cf345.63f56c\",\"type\":\"
    http response\",\"z\":\"b01416d3.f69f38\",\"name\":\"\",\"statusCode
    \":\"200\",\"headers\":{},\"x\":540,\"y\":420,\"wires\":[]},{\"id\":\"6530621.95
    b429c\",\"type\":\"http in\",\"z\":\"b01416d3.f69f38\",\"name\":\"\",\"url\":\"/
    get-sensor\",\"method\":\"get\",\"upload\":false,\"swaggerDoc\":\"\",\"x
    \":180,\"y\":600,\"wires\":[[\"9471d1a0.68588\"]]},{\"id\":\"5ddc9f47.4b555\",\"
    type\":\"http response\",\"z\":\"b01416d3.f69f38\",\"name\":\"\",\"statusCode
    \":\"200\",\"headers\":{},\"x\":540,\"y\":560,\"wires\":[]},{\"id\":\"9471d1a0
```

```
    .68588\",\"type\":\"function\",\"z\":\"b01416d3.f69f38\",\"name\":\"\",\"func
    \":\"msg.payload = {\\\"value1\\\":24.25, \\\"value2\\\":49.54, \\\"value3
    \\\":1005.14};\\nreturn msg;\",\"outputs\":1,\"noerr\":0,\"x\":350,\"y\":600,\"
    wires\":[[\"5ddc9f47.4b555\",\"13aea59.7430e5a\"]]},{\"id\":\"13aea59.7430e5a
    \",\"type\":\"debug\",\"z\":\"b01416d3.f69f38\",\"name\":\"\",\"active\":true,\"
    tosidebar\":true,\"console\":false,\"tostatus\":false,\"complete\":\"false\",\"x
    \":550,\"y\":628,\"wires\":[]},{\"id\":\"e71c7a7d.e7c598\",\"type\":\"debug\",\"
    z\":\"b01416d3.f69f38\",\"name\":\"\",\"active\":true,\"tosidebar\":true,\"
    console\":false,\"tostatus\":false,\"complete\":\"false\",\"x\":550,\"y\":500,\"
    wires\":[]},{\"id\":\"c7807102.3f433\",\"type\":\"http in\",\"z\":\"b01416d3.
    f69f38\",\"name\":\"\",\"url\":\"/update-sensor\",\"method\":\"get\",\"upload\":
    false,\"swaggerDoc\":\"\",\"x\":190,\"y\":460,\"wires\":[[\"60410cde.562a34
    \"]]},{\"id\":\"60410cde.562a34\",\"type\":\"function\",\"z\":\"b01416d3.f69f38
    \",\"name\":\"\",\"func\":\"msg.payload = msg.payload.temperature;\\nreturn msg
    ;\",\"outputs\":1,\"noerr\":0,\"x\":390,\"y\":460,\"wires\":[[\"e71c7a7d.e7c598
    \",\"7f5cf345.63f56c\"]]},{\"id\":\"2b7ac01b.fc984\",\"type\":\"ui_group\",\"z
    \":\"\",\"name\":\"SENSORS\",\"tab\":\"99ab8dc5.f435c\",\"disp\":true,\"width
    \":\"6\",\"collapse\":false},{\"id\":\"99ab8dc5.f435c\",\"type\":\"ui_tab\",\"z
    \":\"\",\"name\":\"HTTP\",\"icon\":\"dashboard\",\"order\":1,\"disabled\":false
    ,\"hidden\":false}]\n",
        "env": []
    },
    {
        "id": "5a81f3be.561bdc",
        "type": "subflow",
        "name": "Time Stamp",
        "info": "",
        "category": "",
        "in": [
            {
                "x": 80,
                "y": 100,
                "wires": [
                    {
                        "id": "682ac304.0590d4"
                    }
                ]
            }
        ],
        "out": [
            {
                "x": 640,
                "y": 180,
                "wires": [
                    {
                        "id": "682ac304.0590d4",
                        "port": 0
                    }
                ]
            },
            {
                "x": 640,
                "y": 230,
```

```
                "wires": [
                    {
                        "id": "682ac304.0590d4",
                        "port": 0
                    },
                    {
                        "id": "3db4df41.634848",
                        "port": 0
                    }
                ]
            },
            {
                "x": 640,
                "y": 280,
                "wires": [
                    {
                        "id": "682ac304.0590d4",
                        "port": 0
                    }
                ]
            }
        ],
        "env": [],
        "color": "#FF8888",
        "outputLabels": [
            "For use in log files",
            "msg.time",
            "For filename use"
        ],
        "icon": "node-red/timer.svg"
    },
    {
        "id": "9ecad9c1f3fd2447",
        "type": "mqtt-broker",
        "name": "",
        "broker": "192.168.2.20",
        "port": "1883",
        "clientid": "",
        "autoConnect": true,
        "usetls": false,
        "protocolVersion": "4",
        "keepalive": "60",
        "cleansession": true,
        "birthTopic": "",
        "birthQos": "0",
        "birthPayload": "",
        "birthMsg": {},
        "closeTopic": "",
        "closeQos": "0",
        "closePayload": "",
        "closeMsg": {},
        "willTopic": "",
        "willQos": "0",
```

```
        "willPayload": "",
        "willMsg": {},
        "sessionExpiry": ""
    },
    {
        "id": "6fd35aff7495ef2e",
        "type": "ui_tab",
        "name": "Pomona Dashboard",
        "icon": "dashboard",
        "disabled": false,
        "hidden": false
    },
    {
        "id": "bf6bf64c9316602d",
        "type": "ui_base",
        "theme": {
            "name": "theme-custom",
            "lightTheme": {
                "default": "#0094CE",
                "baseColor": "#fb60d9",
                "baseFont": "-apple-system,BlinkMacSystemFont,Segoe UI,Roboto,Oxygen
-Sans,Ubuntu,Cantarell,Helvetica Neue,sans-serif",
                "edited": true,
                "reset": false
            },
            "darkTheme": {
                "default": "#097479",
                "baseColor": "#710f73",
                "baseFont": "Arial,Arial,Helvetica,sans-serif",
                "edited": true,
                "reset": false
            },
            "customTheme": {
                "name": "Provisional",
                "default": "#4B7930",
                "baseColor": "#000000",
                "baseFont": "-apple-system,BlinkMacSystemFont,Segoe UI,Roboto,Oxygen
-Sans,Ubuntu,Cantarell,Helvetica Neue,sans-serif",
                "reset": false
            },
            "themeState": {
                "base-color": {
                    "default": "#25ebef",
                    "value": "#000000",
                    "edited": true
                },
                "page-titlebar-backgroundColor": {
                    "value": "#000000",
                    "edited": false
                },
                "page-backgroundColor": {
                    "value": "#111111",
                    "edited": false
```

```
146                 },
147                 "page-sidebar-backgroundColor": {
148                     "value": "#333333",
149                     "edited": false
150                 },
151                 "group-textColor": {
152                     "value": "#000000",
153                     "edited": true
154                 },
155                 "group-borderColor": {
156                     "value": "#555555",
157                     "edited": false
158                 },
159                 "group-backgroundColor": {
160                     "value": "#333333",
161                     "edited": false
162                 },
163                 "widget-textColor": {
164                     "value": "#eeeeee",
165                     "edited": false
166                 },
167                 "widget-backgroundColor": {
168                     "value": "#000000",
169                     "edited": false
170                 },
171                 "widget-borderColor": {
172                     "value": "#333333",
173                     "edited": false
174                 },
175                 "base-font": {
176                     "value": "-apple-system,BlinkMacSystemFont,Segoe UI,Roboto,
    Oxygen-Sans,Ubuntu,Cantarell,Helvetica Neue,sans-serif"
177                 }
178             },
179             "angularTheme": {
180                 "primary": "indigo",
181                 "accents": "blue",
182                 "warn": "red",
183                 "background": "grey",
184                 "palette": "light"
185             }
186         },
187         "site": {
188             "name": "Pomona Dashboard",
189             "hideToolbar": "false",
190             "allowSwipe": "false",
191             "lockMenu": "true",
192             "allowTempTheme": "true",
193             "dateFormat": "DD/MM/YYYY",
194             "sizes": {
195                 "sx": 100,
196                 "sy": 100,
197                 "gx": 6,
```

```
                "gy": 6,
                "cx": 6,
                "cy": 6,
                "px": 0,
                "py": 0
            }
        }
    },
    {
        "id": "5cd6b4a4a6ab7a95",
        "type": "ui_group",
        "name": "Default",
        "tab": "6fd35aff7495ef2e",
        "order": 1,
        "disp": false,
        "width": "8",
        "collapse": false,
        "className": ""
    },
    {
        "id": "02c572ed1ea1e7d1",
        "type": "MySQLdatabase",
        "name": "",
        "host": "127.0.0.1",
        "port": "3306",
        "db": "Pomona",
        "tz": "+02:00",
        "charset": "UTF8mb4"
    },
    {
        "id": "c810aa73.a5706",
        "type": "change",
        "z": "5a81f3be.561bdc",
        "name": "TOPIC",
        "rules": [
            {
                "t": "move",
                "p": "payload",
                "pt": "msg",
                "to": "time",
                "tot": "msg"
            }
        ],
        "action": "",
        "property": "",
        "from": "",
        "to": "",
        "reg": false,
        "x": 220,
        "y": 230,
        "wires": [
            [
                "3db4df41.634848"
```

```
            ]
        ]
    },
    {
        "id": "be4fd533.c9abb",
        "type": "change",
        "z": "5a81f3be.561bdc",
        "name": "Save",
        "rules": [
            {
                "t": "set",
                "p": "payload",
                "pt": "flow",
                "to": "payload",
                "tot": "msg"
            }
        ],
        "action": "",
        "property": "",
        "from": "",
        "to": "",
        "reg": false,
        "x": 220,
        "y": 140,
        "wires": [
            [
                "c7ad4b3c.3a5368"
            ]
        ]
    },
    {
        "id": "3db4df41.634848",
        "type": "change",
        "z": "5a81f3be.561bdc",
        "name": "Get",
        "rules": [
            {
                "t": "set",
                "p": "payload",
                "pt": "msg",
                "to": "payload",
                "tot": "flow"
            }
        ],
        "action": "",
        "property": "",
        "from": "",
        "to": "",
        "reg": false,
        "x": 440,
        "y": 230,
        "wires": [
            []
```

```
        ]
    },
    {
        "id": "682ac304.0590d4",
        "type": "switch",
        "z": "5a81f3be.561bdc",
        "name": "check topic",
        "property": "topic",
        "propertyType": "msg",
        "rules": [
            {
                "t": "eq",
                "v": "TIMESTAMP",
                "vt": "str"
            },
            {
                "t": "else"
            }
        ],
        "checkall": "true",
        "repair": false,
        "outputs": 2,
        "x": 200,
        "y": 100,
        "wires": [
            [],
            [
                "be4fd533.c9abb"
            ]
        ]
    },
    {
        "id": "c7ad4b3c.3a5368",
        "type": "change",
        "z": "5a81f3be.561bdc",
        "name": "TimeStamp",
        "rules": [
            {
                "t": "set",
                "p": "payload",
                "pt": "msg",
                "to": "",
                "tot": "date"
            }
        ],
        "action": "",
        "property": "",
        "from": "",
        "to": "",
        "reg": false,
        "x": 200,
        "y": 180,
        "wires": [
```

```
            []
        ]
    },
    {
        "id": "40def674ee2e662a",
        "type": "ui_button",
        "z": "65de697399c108a4",
        "name": "StartB",
        "group": "5cd6b4a4a6ab7a95",
        "order": 0,
        "width": "2",
        "height": "1",
        "passthru": false,
        "label": "Start",
        "tooltip": "On Click Starts the Pomona",
        "color": "white",
        "bgcolor": "Green",
        "className": "",
        "icon": "",
        "payload": "Start",
        "payloadType": "str",
        "topic": "Start",
        "topicType": "str",
        "x": 150,
        "y": 340,
        "wires": [
            [
                "3bde247e493354b4"
            ]
        ]
    },
    {
        "id": "62b890ad906008a7",
        "type": "ui_button",
        "z": "65de697399c108a4",
        "name": "StopB",
        "group": "5cd6b4a4a6ab7a95",
        "order": 1,
        "width": "2",
        "height": "1",
        "passthru": false,
        "label": "Stop",
        "tooltip": "On click Stops the Pomona",
        "color": "white",
        "bgcolor": "red",
        "className": "",
        "icon": "",
        "payload": "Stop",
        "payloadType": "str",
        "topic": "Mqtttest",
        "topicType": "msg",
        "x": 150,
        "y": 300,
```

```
        "wires": [
            [
                "3bde247e493354b4"
            ]
        ]
    },
    {
        "id": "385fb18a82d517b0",
        "type": "ui_button",
        "z": "65de697399c108a4",
        "name": "DiagnosticsB",
        "group": "5cd6b4a4a6ab7a95",
        "order": 0,
        "width": "2",
        "height": "1",
        "passthru": false,
        "label": "Diagnostics",
        "tooltip": "On click Receives Current State of the Pomona",
        "color": "white",
        "bgcolor": "Orange",
        "className": "",
        "icon": "",
        "payload": "Diagnostics",
        "payloadType": "str",
        "topic": "Diagnostics",
        "topicType": "str",
        "x": 130,
        "y": 380,
        "wires": [
            [
                "3bde247e493354b4"
            ]
        ]
    },
    {
        "id": "3577dcd8ea5d1ad2",
        "type": "mysql",
        "z": "65de697399c108a4",
        "mydb": "02c572ed1ea1e7d1",
        "name": "",
        "x": 560,
        "y": 460,
        "wires": [
            [
                "f94a7dae62533e06",
                "c79a04fa980d5b09"
            ]
        ]
    },
    {
        "id": "c6631068757927c9",
        "type": "inject",
        "z": "65de697399c108a4",
```

```
        "name": "Show Entries",
        "props": [
            {
                "p": "topic",
                "vt": "str"
            }
        ],
        "repeat": "10",
        "crontab": "",
        "once": false,
        "onceDelay": 0.1,
        "topic": "select * from PomonaDashboard order by Id desc limit 50",
        "x": 120,
        "y": 460,
        "wires": [
            [
                "3577dcd8ea5d1ad2"
            ]
        ]
    },
    {
        "id": "f94a7dae62533e06",
        "type": "ui_template",
        "z": "65de697399c108a4",
        "group": "5cd6b4a4a6ab7a95",
        "name": "",
        "order": 4,
        "width": 0,
        "height": 0,
        "format": "<style>\ntable\n{\n    background:#143d59;\n}\n.main\n{\n
    height:200px;\n}\n</style>\n<div class=\"main\">\n<table style=\"width:100%\">\n
        <tr>\n        <th>ID</th>\n        <th>State</th>\n        <th>CurrentAC</th
    >\n        <th>Email Failover</th>\n        <th>Timestamp</th>\n        \n    </
    tr>\n    <tr ng-repeat=\"x in msg.payload | limitTo:100\">\n        <td style=\"
    text-align:center\">{{msg.payload[$index].Id}}</td>\n        <td style=\"text-
    align:center\">{{msg.payload[$index].State}}</td>\n        <td style=\"text-
    align:center\">{{msg.payload[$index].Currentac}}</td>\n        <td style=\"text-
    align:center\">{{msg.payload[$index].Email}}</td>\n        <td style=\"text-
    align:center\">{{msg.payload[$index].Timestamp}}</td>\n        \n    </tr>\n</
    table>\n</div>",
        "storeOutMessages": true,
        "fwdInMessages": true,
        "resendOnRefresh": true,
        "templateScope": "local",
        "className": "",
        "x": 820,
        "y": 460,
        "wires": [
            []
        ]
    },
    {
        "id": "b24c892d9bcbf904",
```

```
        "type": "mqtt in",
        "z": "65de697399c108a4",
        "name": "Inputs",
        "topic": "Mqtttest",
        "qos": "0",
        "datatype": "auto",
        "broker": "9ecad9c1f3fd2447",
        "nl": false,
        "rap": false,
        "inputs": 0,
        "x": 90,
        "y": 560,
        "wires": [
            [
                "0e218344b703b8e1",
                "16cb4dd5a07dbbf4"
            ]
        ]
    },
    {
        "id": "0e218344b703b8e1",
        "type": "ui_text",
        "z": "65de697399c108a4",
        "group": "5cd6b4a4a6ab7a95",
        "order": 4,
        "width": 0,
        "height": 0,
        "name": "",
        "label": "Current Pomona State",
        "format": "{{msg.payload}}",
        "layout": "col-center",
        "className": "",
        "x": 680,
        "y": 560,
        "wires": []
    },
    {
        "id": "16cb4dd5a07dbbf4",
        "type": "function",
        "z": "65de697399c108a4",
        "name": "Split & Insert",
        "func": "var myDataArray = msg.payload.split(':');\nvar State = myDataArray
    [2];\nvar Currentac = myDataArray[4];\nvar Email = myDataArray[6];\nmsg.topic =
    \"INSERT INTO `PomonaDashboard` (`State`,`Currentac`,`Email`)VALUES ('\"+State
    +\"','\"+Currentac+\"','\"+Email+\"')\";\nreturn msg;",
        "outputs": 1,
        "noerr": 0,
        "initialize": "",
        "finalize": "",
        "libs": [],
        "x": 310,
        "y": 500,
        "wires": [
```

```
            [
                "3577dcd8ea5d1ad2"
            ]
        ]
    },
    {
        "id": "3bde247e493354b4",
        "type": "mqtt out",
        "z": "65de697399c108a4",
        "name": "",
        "topic": "Mqtttest2",
        "qos": "",
        "retain": "",
        "respTopic": "",
        "contentType": "",
        "userProps": "",
        "correl": "",
        "expiry": "",
        "broker": "9ecad9c1f3fd2447",
        "x": 560,
        "y": 400,
        "wires": []
    },
    {
        "id": "c5cace63ad0cfc24",
        "type": "e-mail",
        "z": "65de697399c108a4",
        "server": "smtp.gmail.com",
        "port": "465",
        "secure": true,
        "tls": true,
        "name": "aggkouk2000@gmail.com",
        "dname": "",
        "x": 750,
        "y": 180,
        "wires": []
    },
    {
        "id": "f19678c3e9fa9cf1",
        "type": "ui_button",
        "z": "65de697399c108a4",
        "name": "EmailB",
        "group": "5cd6b4a4a6ab7a95",
        "order": 5,
        "width": "2",
        "height": "1",
        "passthru": false,
        "label": "Email",
        "tooltip": "On Click Sends Email To the Administrator",
        "color": "white",
        "bgcolor": "#005b96",
        "className": "",
        "icon": "",
```

```
        "payload": "Here are The latest stats from your Pomona",
        "payloadType": "str",
        "topic": "topic",
        "topicType": "msg",
        "x": 110,
        "y": 180,
        "wires": [
            [
                "17b64aae834480a2"
            ]
        ]
    },
    {
        "id": "17b64aae834480a2",
        "type": "file in",
        "z": "65de697399c108a4",
        "name": "get file",
        "filename": "/home/kouk/Desktop/logs.csv",
        "format": "",
        "chunk": false,
        "sendError": false,
        "encoding": "none",
        "allProps": false,
        "x": 270,
        "y": 180,
        "wires": [
            [
                "2904a0ac8332bed5"
            ]
        ]
    },
    {
        "id": "2904a0ac8332bed5",
        "type": "function",
        "z": "65de697399c108a4",
        "name": "write email",
        "func": "file = msg.filename;    // create local file variable for
    convenient reference\nvar d = new Date();     // create current date object for
    the time string\n    d.setHours(d.getHours() + 2);\n\nvar tstring = d.toString()
    .substring(0,4) + d.toString().substring(15,21);\n\nmsg.attachments =\n     {
    filename : file.substring(file.lastIndexOf('/')+1,file.length),\n        content
     : msg.payload };    // content should be a file binary buffer\n        \nmsg.
    topic = \"Pomona Report \" + tstring; // email subject\n\nmsg.payload = \"Pomona
     Requested Recap : `\" + msg.attachments.filename + \"`\"; // email body\n\
    nfunction addHours(date, hours) {\n  const newDate = new Date(date);\n  newDate.
    setHours(newDate.getHours() + hours);\n  return newDate;\n}\n\n\n\n\nreturn msg
    ;",
        "outputs": 1,
        "noerr": 0,
        "initialize": "",
        "finalize": "",
        "libs": [],
        "x": 490,
```

```
        "y": 180,
        "wires": [
            [
                "c5cace63ad0cfc24"
            ]
        ]
    },
    {
        "id": "152f71e8fa8d2626",
        "type": "file",
        "z": "65de697399c108a4",
        "name": "",
        "filename": "/home/kouk/Desktop/logs.csv",
        "appendNewline": false,
        "createDir": true,
        "overwriteFile": "true",
        "encoding": "none",
        "x": 940,
        "y": 520,
        "wires": [
            []
        ]
    },
    {
        "id": "c79a04fa980d5b09",
        "type": "csv",
        "z": "65de697399c108a4",
        "name": "CSV",
        "sep": ",",
        "hdrin": true,
        "hdrout": "all",
        "multi": "one",
        "ret": "\\r\\n",
        "temp": "Id,State,Currentac,EmailFailover,Timestamp",
        "skip": "0",
        "strings": true,
        "include_empty_strings": true,
        "include_null_values": true,
        "x": 730,
        "y": 500,
        "wires": [
            [
                "152f71e8fa8d2626"
            ]
        ]
    }
]
```


Listing 3: Example of the NodeRED Backend