Microcontrollers Design and Implementation

Railway Administration and Information Logical System

RAILS for Model Railroads

David Bristow
Version 2.0.0
October 27, 2025

Contents

Chapter 1

Microcontrollers for RAILS

1.1 Introduction

Railway Administration and Information Logical System (RAILS) is a software model and implementation of an automated system to assist the model railroader achieve realism in the operation of a model railroad. There are four user interface Single Page Application (SPA) that provide different aspects of rails they are:

- Rollingstock RFID Manager (RSRM) allows the user to match a rfid tag to a rollingstock's road name and number;
- Model Railroad Inventory Manager (MRIM) allows the user to create, update and delete model railroad assets, such as rolling stock;
- Model Projects and Purchase Manager (MPPM) allows the user to enter information about their projects and purchases; and
- Model Railroad Layout Manager (MRLM) allows the user to enter information about their layout and control elements of it.

1.2 RSRM Components

The implementation of RSRM consists of the following micro-services components:

- Radio Frequence Identification (RFID) Controller is a micro-controller that processes RFID tags obtained from a RFID reader and then publishes Internet of Things (IoT) messages to the Message Queuing Telemetry Transport (MQTT) Broker;
- MQTT Broker is responsible for receiving RFID and micro info messages, filtering them, posting to designated topics and sending messages to clients subscribing to topics. The subscribers and publishers bridge the MQTT elements with the GUI applications. The broker handles IoT messages;
- IoT Subscriber RFID Services (ISRS) subscribes to RFID messages and pushes them via a web-socket to the rsm component;

- IoT Subscriber Micro-controller Services (ISMS) that subscribes to micro controller startup and heartbeat messages, updating the micros collection via Railroad Layout Data Services (RLDS);
- RLDS provides Representational State Transfer (REST) access to model railroad layout collections including micros;
- Railroad Inventory Data Services (RIDS) provides REST access to railroad inventory collections including rollingstock;
- MongoDB a NoSQL database program that stores data records as documents which are gathered in collections. A database stores one or more collections of documents;
- Model Railroad (MR) Data is the document repository, used by MongoDB, to store complete collections of items such as rollingstock, industries (producers and consumers), track elements, turnouts, projects, purchases, etc. in support of RAILS; and
- RSRM is the SPA that allows a user to match a RFID tag to a rollingstock's road name and number.

Figure 1.1 depicts the micro-services used to create the rolling stock RFID management subsystem.

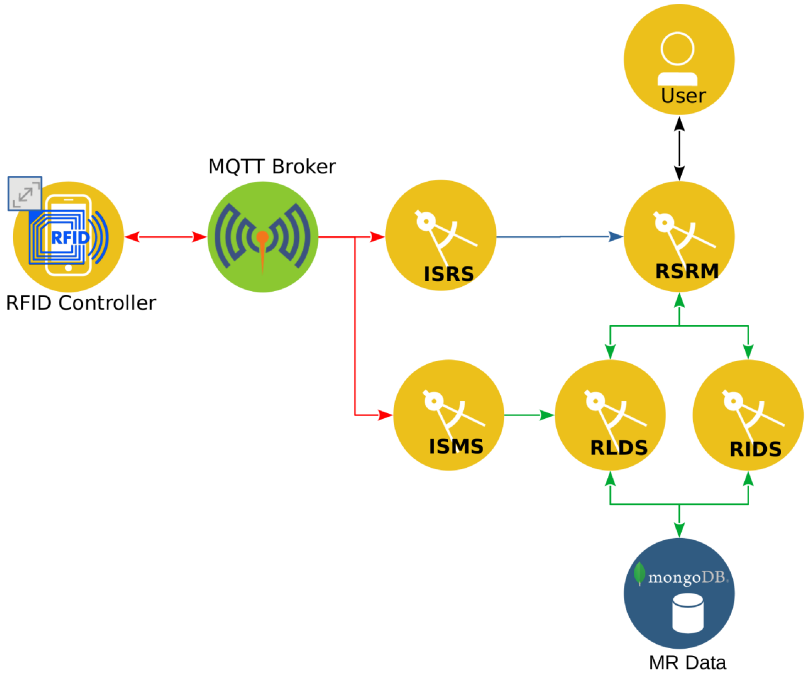

Figure 1.1: Microservices Components

Three system components that make up this subsystem:

- RFID Controller is a micro-controller that processes RFID tags obtained from a RFID reader and then publishes IoT messages to the MQTT Broker;
- Network is a Transmission Control Protocol/Internet Protocol (TCP/IP) communication medium that connects the RFID Controller, MQTT Broker and RSRM components; and

- Host is a computer that runs the MQTT Broker and other RSRM micro-services components.

Figure 1.2 depicts the systems components used to create the rolling stock RFID management subsystem.

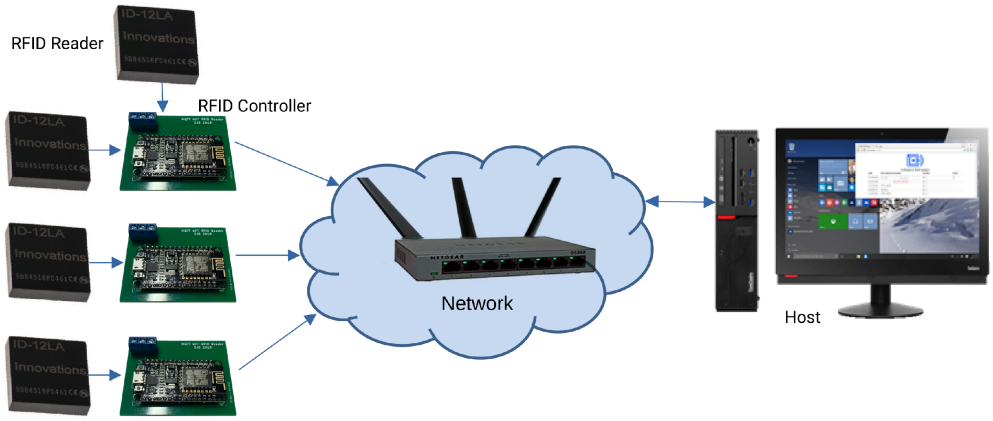

Figure 1.2: RSMS System Components

1.3 MRLM System Components

Figure 1.3 depicts the micro-services used to create the MRLM subsystem and figure 1.4 depicts it's logical architecture.

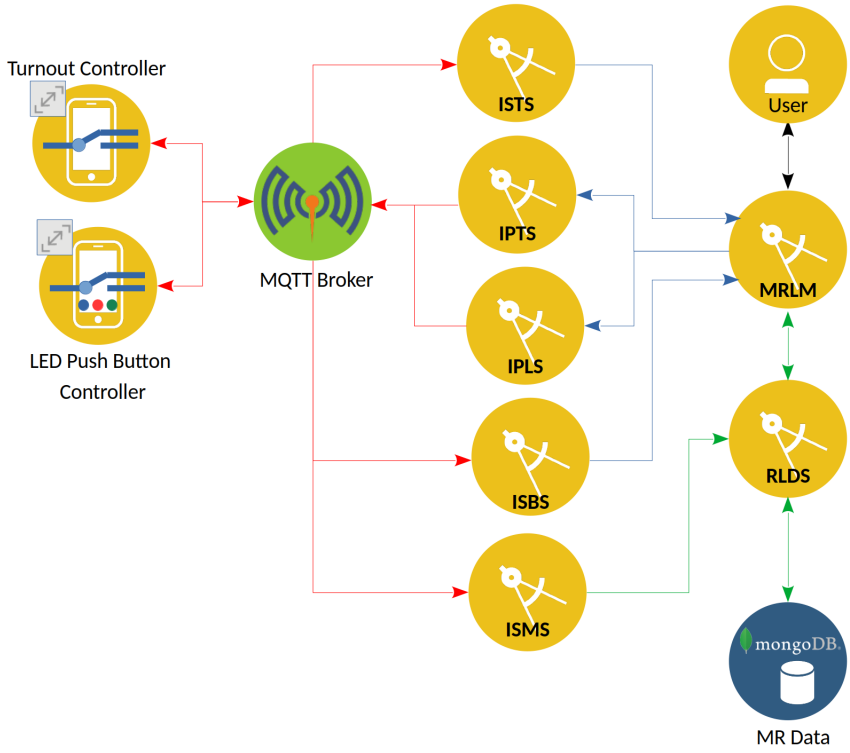

Figure 1.3: Microservices Components

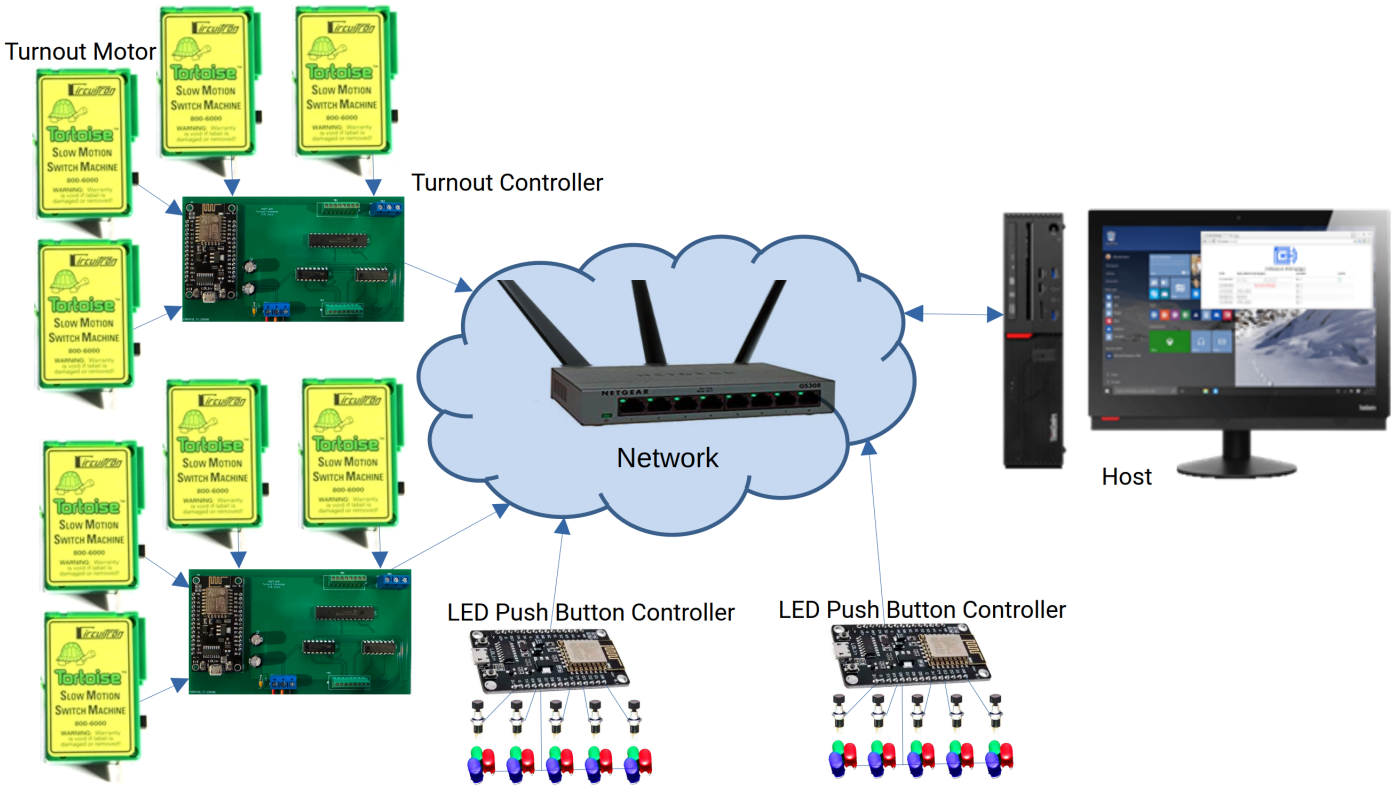

Figure 1.4: MRLM System Components

Chapter 2

RFID Microcontroller

2.1 Introduction

The RFID microcontroller is a versatile and compact system designed to interface with RFID readers and process tag information efficiently. It acts as a bridge between RFID hardware and software applications, enabling seamless communication and data exchange. Built around the NodeMCU, which is powered by the ESP8266 chip, the microcontroller offers robust processing capabilities and integrated Wireless Fidelity (Wi-Fi) connectivity. This makes it ideal for IoT applications where RFID data needs to be transmitted over a network for further processing, monitoring, or storage. The system is designed to be reliable, scalable, and easy to integrate into various use cases, including access control, inventory management, asset tracking, and smart automation systems.

2.2 Hardware

2.2.1 Schematic

The schematic diagram in Figure 2.1 illustrates the design of the RFID microcontroller circuit. It shows the connections between componentse and the terminal blocks for one or two RFID readers. Each component is carefully integrated to ensure reliable recieving the tag information from the RFID readers.

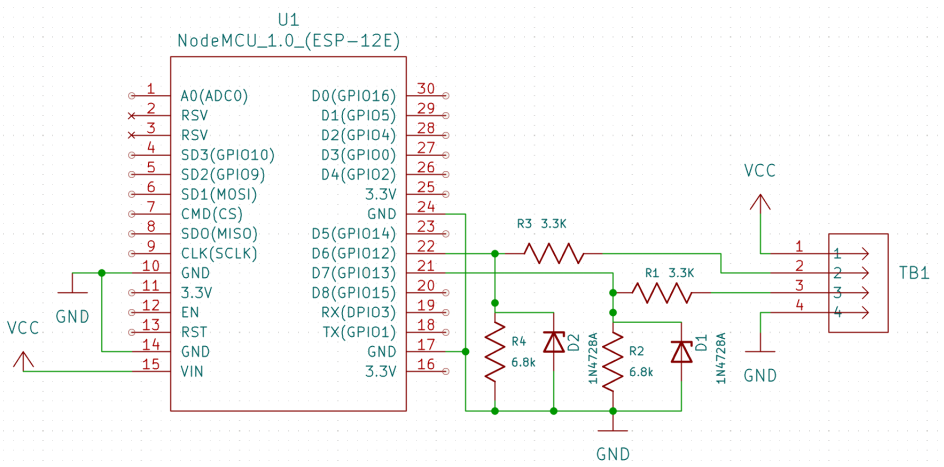

Figure 2.1: RFID Microcontroller Schematic

The hardware used in the RFID microcontroller circuit consists of several components that work together to enable the RFID readers. The main components are:

- NodeMCU (U1) is the central microcontroller, an ESP8266, which will manage communication with the RFID readers.
- Resistors (R1, R2, R3, R4) are resistors crucial for voltage level shifting.
- Zener Diodes (D1, D2) are diodes, along with the resistors, form a level shifting circuit.
- TB1 Connector is the 4-pin connector where the RFID readers are attached.

The NodeMCU (U1) is programmed to handle the communication with the RFID readers through the Inter-Integrated Circuit (I2C) protocol. The resistors (R1, R2, R3, and R4) are used in a level shifting circuit to ensure that the voltage levels between the NodeMCU and the RFID readers are compatible. The diodes (D1 and D2) are used to protect against voltage spikes and ensure proper operation of the level shifting circuit. The TB1 connector is where the RFID readers are connected to the microcontroller.

2.2.2 PCB Design

The printed circuit board (PCB) design for the RFID microcontroller is shown in Figure 2.2. The PCB layout is designed to accommodate all the components mentioned in the schematic, ensuring proper connections and efficient routing of signals.

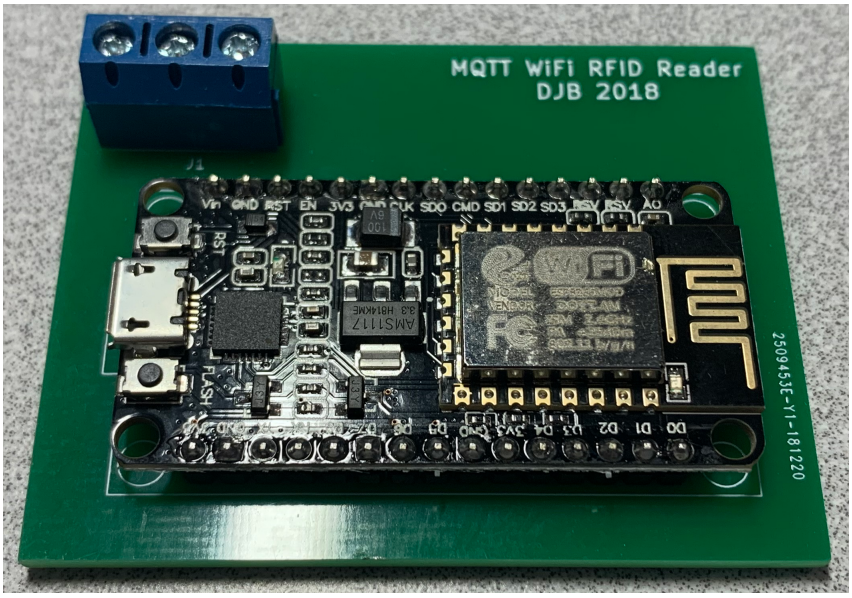

Figure 2.2: RFID Microcontroller PCB

2.2.3 Connectivity

The wiring diagram in Figure 2.3 illustrates the connections between the RFID readers and the microcontroller. The RFID readers are connected to the TB1 terminal block, which uses a RS232 connection to the NodeMCU. The connections are made using standard wiring techniques, ensuring reliable and secure connections. The use of terminal blocks allows for easy connection and disconnection of the Tortoise machines, making maintenance and troubleshooting easier.

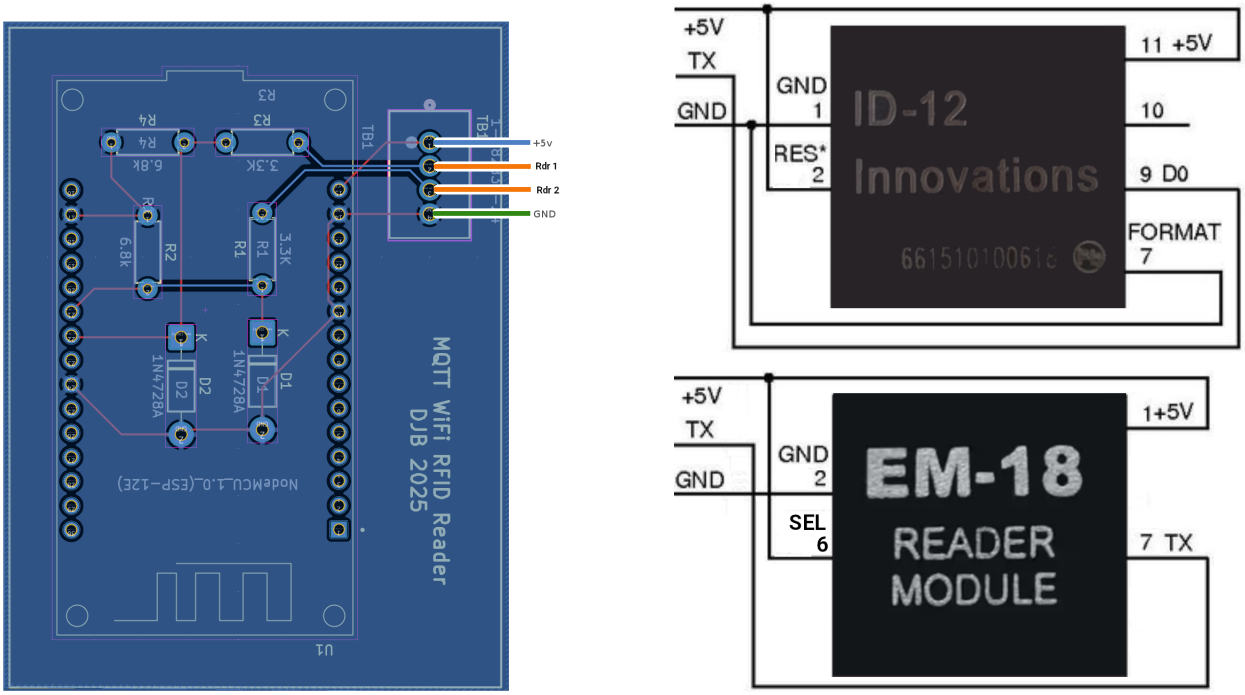

Figure 2.3: RFID Microcontroller - RFID Reader Connections

2.3 Software

The software for the RFID microcontroller is developed using the Visual Studio Code (VS Code), which provides a user-friendly environment for programming microcontrollers. The code is written in C/C++ and utilizes various libraries to facilitate communication with the components. The software implementation includes the following key functionalities:

- Initialization of the NodeMCU and configuration of the GPIO pins.
- Initialization of the Wi-Fi connectivity and connect to the MQTT broker. This allows the microcontroller to send and receive messages over the network. At the finish of the initialization process, the microcontroller, it will publish a message to the topic “micros” indicating its readiness and status to the MQTT broker. The message is in JavaScript Object Notation (JSON) format as follows:
{“et”:“1590462747”,“mcntrlr”:“RfidRdr01”,“msgType”:“initial”,“ip”:“192.168.0.19”}
- Setup of the serial communication protocol to interface with the RFID reader module. This allows for reading RFID tags.
- Handling RFID tag detection and processing the data received from the RFID reader. reads values from a single RFID reader, formats the results as a JSON string, gets Epoch time from an NTP server and then publishes the JSON String to the topic “sensors/rfid” on the MQTT broker. The message is in JSON format as follows:
{“et”:“1590463450”,“mcntrlr”:“RfidRdr01”,“reader”:“1”,“rfid”:“1C0044CF23”}
- Sending periodic heartbeat messages to the MQTT broker to indicate that the microcontroller is operational. This can be used for monitoring and debugging purposes. The message are

in JSON format as follows:
{“et”:“1590462747”,“mcntrlr”:“RfidRdr01”,“msgType”:“heartbeat”}

The software is designed to be modular and easily maintainable, allowing for future enhancements and bug fixes. The use of libraries such as “PubSubClient.h” for MQTT messaging simplifies the implementation and ensures compatibility with the hardware components.

Chapter 3

Turnout Microcontroller

3.1 Introduction

A turnout microcontroller is a specialized electronic device designed to control railway turnouts, also known as switches. These devices are essential in model railroading and railway automation, where precise and reliable control of turnouts is required. By using a microcontroller, it is possible to automate the operation of turnouts, integrate them into larger control systems, and provide feedback on their status. This chapter discusses the design and implementation of a microcontroller-based system for controlling Tortoise switch machines, which are commonly used in model railroads.

3.2 Hardware

3.2.1 Schematic

The schematic diagram in Figure 3.1 illustrates the design of the turnout microcontroller circuit. It shows the connections between componentse and the terminal blocks for the Tortoise switch machines. Each component is carefully integrated to ensure reliable operation and efficient control of the turnouts. The design also includes capacitors for power supply stabilization and feedback connections for monitoring the switch positions.

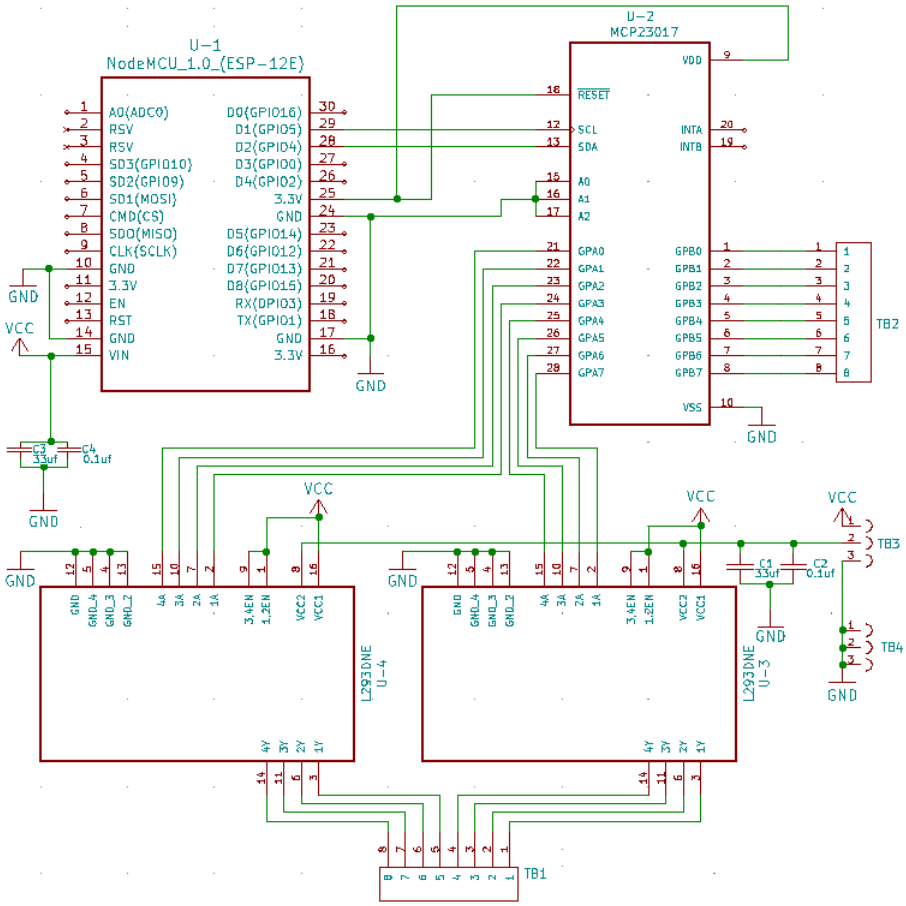

Figure 3.1: Turnout Microcontroller Schematic

The hardware used in the turnout microcontroller circuit consists of several components that work together to control the Tortoise switch machines. The main components are:

- NodeMCU (U-1) is an ESP8266-based development board, which controls the other components.
- MCP23017 (U-2) is a 16-bit input/output (I/O) expander. It increases the number of General Purpose Input/Output (GPIO) pins available from the NodeMCU.
- L293DNE (U-3 and U-4) are dual H-bridge motor drivers. They are used to control the direction of current flow to the Tortoise machines. Tortoise machines require a change in polarity to change the direction of the switch.
- TB1 is a terminal block connector where the Tortoise switch machines, up to four, are connected to the circuit.
- TB2 is a terminal block connector where the Tortoise switch machines internal contacts are connected to the circuit.
- TB3 and TB4 are additional terminal blocks, for power distribution.

- Capacitors (C1, C2, C3, and C4) are used for filtering and smoothing power supply, ensuring stable operation.

The NodeMCU is programmed to control the MCP23017, which in turn controls the L293DNE drivers. The drivers are responsible for controlling the direction of current flow to the Tortoise machines. The capacitors are used to filter and smooth the power supply to ensure stable operation of the circuit. The internal contacts of the Tortoise machines are connected to provide feedback on the switch position.

3.2.2 PCB Design

The PCB design for the turnout microcontroller is shown in Figure 3.2. The PCB layout is designed to accommodate all the components mentioned in the schematic, ensuring proper connections and efficient routing of signals.

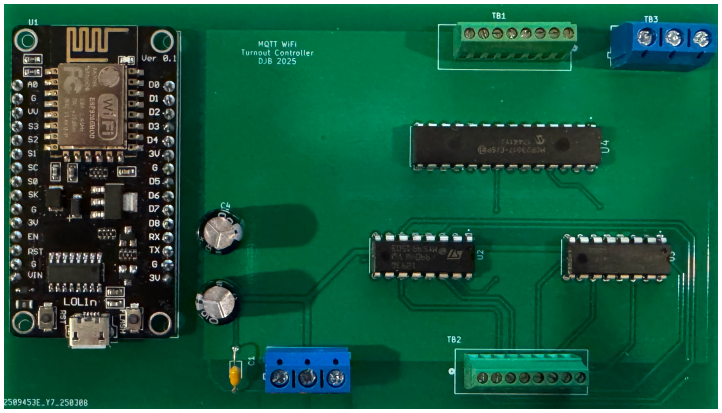

Figure 3.2: Turnout Microcontroller PCB

3.2.3 Connectivity

The wiring diagram in Figure 3.3 illustrates the connections between the Tortoise switch machines and the microcontroller. The Tortoise machines are connected to the TB1 terminal block, which is wired to the L293DNE motor drivers. The internal contacts of the Tortoise machines are connected to the TB2 terminal block, which is wired to the MCP23017 I/O expander. This allows the microcontroller to monitor the position of the switch machines and control their operation. The connections are made using standard wiring techniques, ensuring reliable and secure connections. The use of terminal blocks allows for easy connection and disconnection of the Tortoise machines, making maintenance and troubleshooting easier.

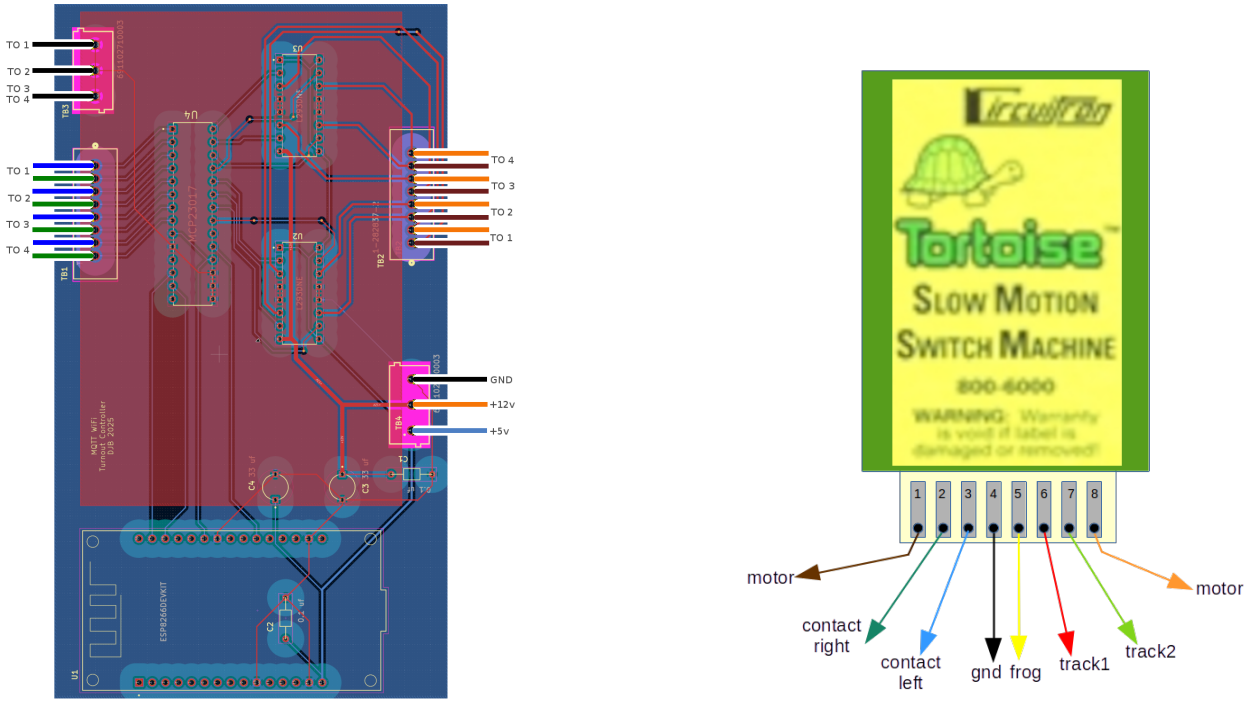

Figure 3.3: Turnout Microcontroller - Tortoise Connections

3.3 Software

The software for the turnout microcontroller is developed using the VS Code, which provides a user-friendly environment for programming microcontrollers. The code is written in C/C++ and utilizes various libraries to facilitate communication with the components. The software implementation includes the following key functionalities:

- Initialization of the NodeMCU and configuration of the GPIO pins.
- Setup of the I2C communication protocol to interface with the MCP23017 I/O expander. This allows for the expansion of GPIO pins beyond what is available on the NodeMCU.
- Initialization of the Wi-Fi connectivity and connect to the MQTT broker. This allows the microcontroller to send and receive messages over the network. At the finish of the initialization process, the microcontroller, it will publish a message to the topic “micros” indicating its readiness and status to the MQTT broker. The message is in JSON format as follows:
{“et”:“1590462747”,“mcntrlr”:“TrnCntlr01”,“msgType”:“initial”,“ip”:“192.168.0.19”}
- Subscribes to incoming MQTT messages to the topic “acts/to/TrnCtlrxx” where xx is the this controller’s number. This includes parsing commands to throw or close the turnouts or the status of the turnout. The commands are in JSON format as follows:
{“cntrlr”:“TrnCntlr01”,“to”:“1|2|3|4”,“cmd”:“throw|close|status”}
- Communication with the MCP23017 to read the status of the inputs and control the outputs.

- Implementation of the logic to control the L293DNE motor drivers based on the commands received. This includes setting the appropriate GPIO pins to control the direction and state of the Tortoise machines.
- Control of the L293DNE motor drivers to change the polarity of the current flow to the Tortoise machines, thereby controlling their position.
- Monitoring the internal contacts of the Tortoise machines to provide feedback on their position.
- Handling of any errors or exceptions that may occur during operation, ensuring the system remains stable and responsive.
- Publishing the status of the turnout positions to an MQTT topic for integration with other systems, such as a model railroad control system.The message are in JSON format as follows: {“et”:“1588827073”,“cntrlr”:“trnCtlr01”,“to”:“1”,“state”:“THROWN|CLOSED|ERR|INVLD”} where ‘state‘ can be ‘THROWN‘, ‘CLOSED‘, ‘ERR‘ (for error), or ‘INVLD‘ (for invalid state). This allows other components of the system to monitor the status of each turnout in real-time.
- Sending periodic heartbeat messages to the MQTT broker to indicate that the microcontroller is operational. This can be used for monitoring and debugging purposes. The message are in JSON format as follows:
{“et”:“1590462747”,“mcntrlr”:“TrnCntlr01”,“msgType”:“heartbeat”}

The software is designed to be modular and easily maintainable, allowing for future enhancements and bug fixes. The use of libraries such as “Wire.h” for I2C communication and “PubSubClient.h” for MQTT messaging simplifies the implementation and ensures compatibility with the hardware components.

Chapter 4

LED Push Button Microcontroller

4.1 Introduction

A light-emitting diode (LED) push button microcontroller is a specialized electronic device designed to request the direction of a turnout be changed and to display the satus of the turnouts assigned to the panel. These devices are essential in model railroading and railway automation, where precise and reliable control of turnouts is required. By using a microcontroller, it is possible to automate the operation of turnouts, integrate them into larger control systems, and provide feedback on their status. This chapter discusses the design and implementation of a microcontroller-based system for the lights and push buttons of a turnout panel.

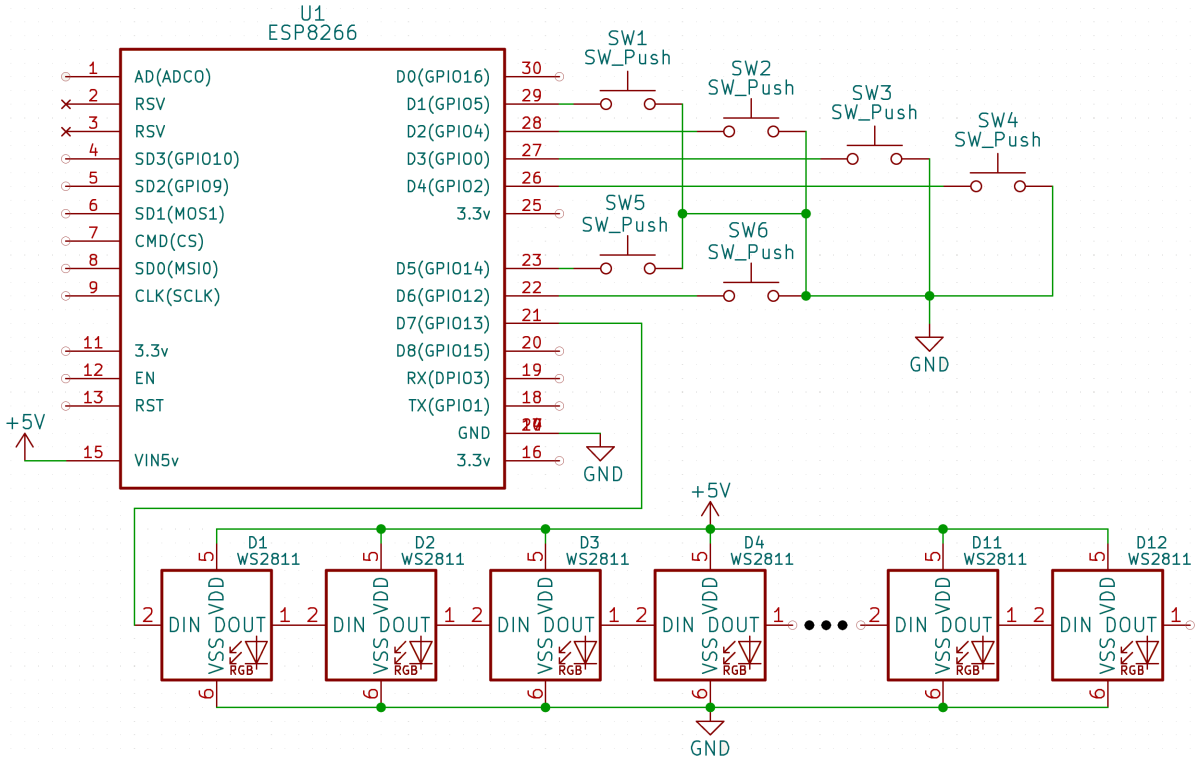

Figure 4.1: LED Push Button Microcontroller Schematic

The hardware used in the LED push button microcontroller circuit consists of several components that work together to control a chain of 12 LEDs and react to 6 push buttons. The main components are:

- NodeMCU (U1) is an ESP8266-based development board, which controls the other components.
- Buttons (SW1 - SW6) There are six momentary push buttons. Each button connects a specific GPIO pin on the ESP8266 to ground (GND). When a button is pressed, it creates a connection to ground, and the microcontroller detects this change. Each button press initiates a request to change the direction of the associated turnout.
- LED Strip (D1 - D12) The circuit is designed to control a chain of 12 addressable LEDs, of the WS2811 type. Each pair of LEDs represents the status of an associated turnout.

The ESP8266 microcontroller reads the state of the buttons, which represents a specific turnout. The microcontroller controls the color of the LEDs.

4.2 Software

The software for the LED push button microcontroller is developed using the VS Code, which provides a user-friendly environment for programming microcontrollers. The code is written in C/C++ and utilizes various libraries to facilitate communication with the components. The software implementation includes the following key functionalities:

- Initialization of the NodeMCU and configuration of the GPIO pins.
- Initialization of the Wi-Fi connectivity and connect to the MQTT broker. This allows the microcontroller to send and receive messages over the network. At the finish of the initialization process, the microcontroller will publish a message to the topic “micros” indicating its readiness and status to the MQTT broker. The message is in JSON format as follows:
{“et”:“1590462747”:“mcntrlr”:“LpbCntlr01”:“msgType”:“initial”:“ip”:“192.168.0.19”}
- Subscribes to the topic acts/tpl/LpbCntlrxx where xx is a specific controller from the MQTT broker. The message is in JSON format as follows:
{“cntrlr”:“LpbCntlrxx”:“lamp”:“1|2|3...6”:“color”:“RED|GREEN|BLUE|YELLOW”}
- Reads up to 6 buttons (count set in “params.h”) using simple debouncing and publishes a JSON message on press to the topic “sensors/pb” to the MQTT broker. The message is in JSON format as follows:
{“et”:“1590462747”:“mcntrlr”:“LpbCntlrxx”:“pb”:“1|2|3...6”}
- Locks that button until a command arrives on the command topic with a matching “lamp” number is recieved.
- Sending periodic heartbeat messages to the MQTT broker to indicate that the microcontroller is operational. This can be used for monitoring and debugging purposes. The message are in JSON format as follows:
{“et”:“1590462747”,“mcntrlr”:“LpbCntlrxx”,“msgType”:“heartbeat”}

The software is designed to be modular and easily maintainable, allowing for future enhancements and bug fixes. The use of libraries such as “Adafruit_NeoPixel.h” for control of WS2811 LED strip and “PubSubClient.h” for MQTT messaging simplifies the implementation and ensures compatibility with the hardware components.

Acronyms

ground (GND) 17

General Purpose Input/Output (GPIO) 12, 14, 15, 17

input/output (I/O) 12, 13, 14

Inter-Integrated Circuit (I2C) 7, 14, 15

Internet of Things (IoT) refers to a vast network of physical devices embedded with sensors, software, and other technologies that allows them to connect and exchange data with other devices and systems over the internet or other communication networks 2, 3, 6

IoT Subscriber Micro-controller Services (ISMS) 3

IoT Subscriber RFID Services (ISRS) 2

JavaScript Object Notation (JSON) is a lightweight data-interchange format that uses human-readable text to transmit data objects consisting of attribute–value 1 pairs and arrays 9, 10, 14, 15, 17

light-emitting diode (LED) a semiconductor diode which glows when a voltage is applied 16, 17, 18

Model Projects and Purchase Manager (MPPM) 2

Message Queuing Telemetry Transport (MQTT) is a lightweight messaging protocol designed for machine-to-machine (M2M) communication in resource-constrained environments, like those found in the Internet of Things (IoT) 2, 3, 4, 9, 10, 14, 15, 17, 18

Model Railroad (MR) 3

Model Railroad Inventory Manager (MRIM) 2

Model Railroad Layout Manager (MRLM) 2, 4

printed circuit board (PCB) 7, 13

Railway Administration and Information Logical System (RAILS) 2, 3

Representational State Transfer (REST) is a set of architectural principles for designing web services. It's not a protocol itself (like HTTP), but rather a guideline for creating web services 3

Radio Frequence Identification (RFID) is technology that uses radio waves to wirelessly identify and track objects 2, 3, 4, 6, 7, 8, 9

Railroad Inventory Data Services (RIDS) 3

Railroad Layout Data Services (RLDS) 3

Rollingstock RFID Manager (RSRM) 2, 3, 4

Single Page Application (SPA) a web application that loads a single HTML page in the user's browser and dynamically updates the content based on user interaction, without reloading the entire page. This creates a more fluid and responsive user experience, similar to what you might expect from a native mobile app 2, 3

Transmission Control Protocol/Internet Protocol (TCP/IP) a suite of communication protocols that governs how data is exchanged over the internet or a private network. It is the foundation of any data exchange on the Web 3

Visual Studio Code (VS Code) 9, 14, 17

Wireless Fidelity (Wi-Fi) 6, 9, 14, 17