Contents

[Things used in this project 1](#_Toc111390837)

[Hardware 1](#_Toc111390838)

[Software 2](#_Toc111390839)

[Lego Pieces 2](#_Toc111390840)

[Story 2](#_Toc111390841)

[How it works 2](#_Toc111390842)

[Step 1 Setup Raspberry Pi 3](#_Toc111390843)

[Install Rocrail on Raspberry Pi 3](#_Toc111390844)

[Install Mosquitto on Raspberry Pi 3](#_Toc111390845)

[Step 2 Add sensors 4](#_Toc111390846)

[Modify Lego to hold sensors 4](#_Toc111390847)

[Connect sensors to ESP 8266 4](#_Toc111390848)

[Step 4 Upload firmware 4](#_Toc111390849)

[Upload Mattzo firmware (MLC) to ESP 8266 4](#_Toc111390850)

[Upload Mattzo firmare (MTC4BT) to ESP 32 5](#_Toc111390851)

[Step 5 Configure Rocrail and Rocview 5](#_Toc111390852)

[Mosquitto configuration 4](#_Toc111390853)

[Demo 17](#_Toc111390854)

[Code 19](#_Toc111390855)

[References 19](#_Toc111390856)

# Things used in this project

## Hardware

* [Raspberry Pi 4 Computer Modell B, 4GB RAM](https://www.berrybase.de/en/raspberry-pi/raspberry-pi-computer/boards/raspberry-pi-4-computer-modell-b-4gb-ram) x 1
* [Argon ONE V2 Case für Raspberry Pi 4](https://www.berrybase.de/en/new/argon-one-v2-case-f-252-r-raspberry-pi-4) x 1
* [Official Raspberry Pi USB-C Power Supply 5,1V / 3,0A, EU, black](https://www.berrybase.de/en/raspberry-pi/raspberry-pi-computer/power-supply/power-supply-units-for-the-socket/offizielles-raspberry-pi-usb-c-netzteil-5-1v/3-0a-eu-schwarz) x 1
* [ESP32 NodeMCU Module Dev Kit C Dev Board with CP2102](https://www.amazon.de/gp/product/B071P98VTG/ref=ppx_yo_dt_b_asin_title_o02_s00?ie=UTF8&th=1) x 1
* [NodeMCU v2 - ESP8266 Development Board, CP2102](https://www.berrybase.de/en/new/nodemcu-v2-esp8266-development-board-cp2102) x 1
* [GPS-14A, size 2,5x14mm](https://www.amazon.de/-/en/gp/product/B07Y9SW7DD/ref=ppx_yo_dt_b_asin_title_o00_s00?ie=UTF8&psc=1) x 4
* [USB cables](https://www.berrybase.de/en/raspberry-pi/raspberry-pi-computer/cable-adapter/usb-cable-adapter/usb-2.0-hi-speed-kabel-a-stecker-150-micro-b-stecker-wei-223) x 2
* [USB 4 Port Power Supply / 5V / 5.0A with Auto-ID white](https://www.berrybase.de/en/raspberry-pi/raspberry-pi-computer/power-supply/power-supply-units-for-the-socket/usb-4-port-netzteil/ladeadapter-5v/5-0a-mit-auto-id-wei-223) x 1

## Software

* Rocrail
* Mosquitto
* Mattzobricks Firmware
  + MTC4BT (V0.5)
  + MLC (V0.5)
* Arduino IDE
* VS Code
  + Platform IO

## Lego Pieces

* [Lego set #75955 – Hogwarts Express train](https://www.lego.com/en-de/product/hogwarts-express-75955)
* Lego train track straight x 8
* Lego train track curved x 4
* [Custom coal car v2.7](https://www.bricklink.com/v3/studio/edit.page?idModel=298270)
* [Extra wagons](https://www.bricklink.com/v3/studio/edit.page?idModel=298296) x 2
* Modified locomotive
* [Lego Plate, Modified 1 x 1 with Light Attachment - Thick Ring](https://www.bricklink.com/v2/catalog/catalogitem.page?P=4081b&C=85%23T=C&C=85) x 4
  + No: 4081b  Alternate Item No: 41632
  + Dark bluish gray

# Story

While the Lego Power Up Functions iPad app works great for getting your trains running on a layout quickly, it does not allow for the level of automation I wanted. I began researching what other AFOL have done and stumbled upon Mattzobrick’s YouTube channel. I was impressed with the flexibility and the collection of Lego trains. They have been constantly updating the project and it seems to be going in a good direction.

One of the latest changes was to firmware version 0.5, in which they added the MTCBT. This is what I am using to control the locomotives. This allows the ESP 32 board to connect to the Lego Powered Up Bluetooth hub, just like one would with and iPad or the Lego controller. This worked great and as I couldn’t get the older version working.

My first goal was to design the minimum viable layout, where I could test enough of the pieces to have a running system. While the Rocrail and Mattzo websites, forums, and documentation are extremely detailed, I wanted to still document the layout I had done. I hope it too can help others interested in trying to automate their trains, because there is no end to the creativity or customization you can two with these products.

# How it works

The Mattzo team has a great diagram of the overall [system architecture](https://mattzobricks.com/automation/overall-architecture).

The oval is the simplest layout to create in Rocrail and what most Lego trains come with originally. So that is what I modeled in Rocrail. In order for this to work, you need to have a minimum of 2 stations (or blocks) in the layout.

Graphical user interface

Description automatically generated

**Step Overview**

1. Setup Raspberry Pi
2. Add sensors
3. Upload firmware
4. Configure Rocrail and Rocview

## Step 1 Setup Raspberry Pi

### Install Rocrail on Raspberry Pi

The first thing to do is [Install Rocrail server on RPI 4](https://wiki.rocrail.net/doku.php?id=download-int). You can choose to run the Rocrail server on your PC, but I chose to have it on the Raspberry Pi. This allowed it to be always running.

It is also a good idea to setup the [Auto start on RPI](https://wiki.rocrail.net/doku.php?id=rocrail-linux-user-en) so that the Rocrail server boots up, when the Raspberry Pi starts.

The Rocrail software allows for server & client architecture. Since the RPi is running the server, you have to setup the client on a PC. I set up Rocview on my Windows 10 PC.

### Install Mosquitto on Raspberry Pi

Follow instructions for setting up a Mosquitto server on Mattzobricks and Rocrail websites.

mosquitto version 1.5.7 starting

Mosquitto configuration

Where are the logs located? - /var/log/mosquitto

How to check that it is working

topic ‘#’

Testing mosquitto

mosquitto\_sub -d -t testTopic

mosquitto\_sub -d -t rocrail/service/command

mosquitto\_pub -d -t testTopic -m "Hello world!"

sudo systemctl stop mosquitto.service

sudo systemctl start mosquitto.service

sudo systemctl restart mosquitto.service

https://wiki.rocrail.net/doku.php?id=rocrailini-service-en#mqtt\_service

## Step 2 Add sensors

### Modify Lego to hold sensors

The sensors, that I found on Amazon.de ended up having too large of a diameter for the Lego pieces. Therefore, I had to drill out the Lego pieces, in order for them to fit. I had purchased all of them from a seller on Brick Link. I first drilled out the hole to 3.5 mm and then to 4.0 mm. Then the sensors were able to fit.

### Connect sensors to ESP 8266

The layout controller can handle more than 4 sensors, but this was the bare minimum needed to have 2 blocks in Rocrail going a single direction. The sensors do not have polarity. They should all have 1 lead connected to ground and the other connected to an input on the development board.

<https://mattzobricks.com/automation/sensors>

## Step 4 Upload firmware

There currently a view different firmware versions available. For this project I used version 0.5.

### Upload Mattzo firmware ([MLC](https://mattzobricks.com/controllers/mattzolayoutcontroller)) to ESP 8266

To upload the firmware I used the Arduino IDE.

### Upload Mattzo firmare ([MTC4BT](https://mattzobricks.com/controllers/mtc4bt)) to ESP 32

For this firmware, the Mattzo team has created a workspace for Visual Studio Code. With the help of some extensions VS Code provides a great IDE for working with the ESP32.

## Step 5 Configure Rocrail and Rocview

Here are some helpful videos on how to setup your Rocrail layout.

<https://www.youtube.com/watch?v=V0APPfda9xA&t=113s>

<https://www.youtube.com/watch?v=tf_V-Ghm0YE&t=8s>

Firstly, you need to configure the Rocrail server properties.

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

The key parameters are under the ‘Service’ tab for the hostname and port for the MQTT service. You should enter the host name or ip address of the Raspberry pi. The default port for Mosquitto is 1883.

Graphical user interface

Description automatically generated

On the ‘Controller’ tab you need to create at least 1 controller. You can see the settings in the following figure.

Graphical user interface, text, application

Description automatically generated

Next you need to setup at least one locomotive. In Rocrail, you can also setup trains, but in this case it isn’t necessary.

Graphical user interface, text, application, table, Word

Description automatically generated

Graphical user interface, application

Description automatically generated

The ‘Speed’ of the locomotive can be set under the locomotive or under

Table

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Then we should setup some sensors. The more sensors the better. The absolute minimum could be two (2) sensors. However, the documentation recommends for two sensors before each block. The first sensor that will be tripped for example rs1 will trigger the ‘*enter*’ event type and then the next sensor rs2, which is closest to the block will trigger the ‘*in*’ event.

Most times the magnets on the Lego train buffers will not trigger the sensor to switch. However, the Lego motor was consistent enough for my scenario, and this is what I used in this case. There are some suggestions on the Mattzobricks website on how to add magnets to your wagons to have a sensor reading for the first and last wagon in the train. I would definitely go this route, if setting up a more complex layout with multiple trains, directions, and varying lengths of the trains.

The address of the sensor

Graphical user interface, application, Word

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Next we can setup the blocks.

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

The reason we needed to blocks was to be able to setup a schedule between the two. It is not possible in Rocrail to have a single block and make it go in a loop. The destination can’t equal the current location. In the schedule below, you can see that we start in block 1 then go to block 2 then go to block 1 again.

Graphical user interface, text, application

Description automatically generated

Graphical user interface

Description automatically generated

## Demo

Graphical user interface

Description automatically generated with low confidenceA picture containing indoor

Description automatically generatedA picture containing text, electronics, circuit

Description automatically generatedA calculator on a table

Description automatically generated with low confidenceA picture containing floor, chair, wooden

Description automatically generated

## Code

The code needed for this project is from a few different sources.

The code need for Rocrail can be found [here](https://mattzobricks.com/controllers/firmware-history-and-roadmap).

The firmware for the development boards can be found [here](https://wiki.rocrail.net/doku.php?id=win-en).

My modified configuration files for the Mattzo controllers can be found [here](https://github.com/jbettenh/smart-lego-trains/tree/trunk/hello_world).

## References

<https://mattzobricks.com/>

https://wiki.rocrail.net/doku.php?id=start