Train Control System

Introduction

The project represents an **automatized system for trains control**. It features a control position for the train using **InfraRed sensors** and **Ultrasonic sensors**. To simulate the detection of various trains, a **RFID reader** is used. This way, the railroad switches will move accordingly. Once at the train station, loading and unloading of the freight are carried out using **Braccio**. The transportation of goods is simulated using some **RFID tags**, automating the switching of the railroad.

Motivation

I had a few starting components for the control system as well as for the train modelling. I thought it would be a challenging and interesting project, mostly because I couldn't find many resources regarding the automatization of train models and seemed a unique idea. Most of the projects that you can find online are some digital sets containing a remote controller and a digital station, thus making them very easy to use without the necessity of a technical background.

Hardware Diagram

Hardware Design

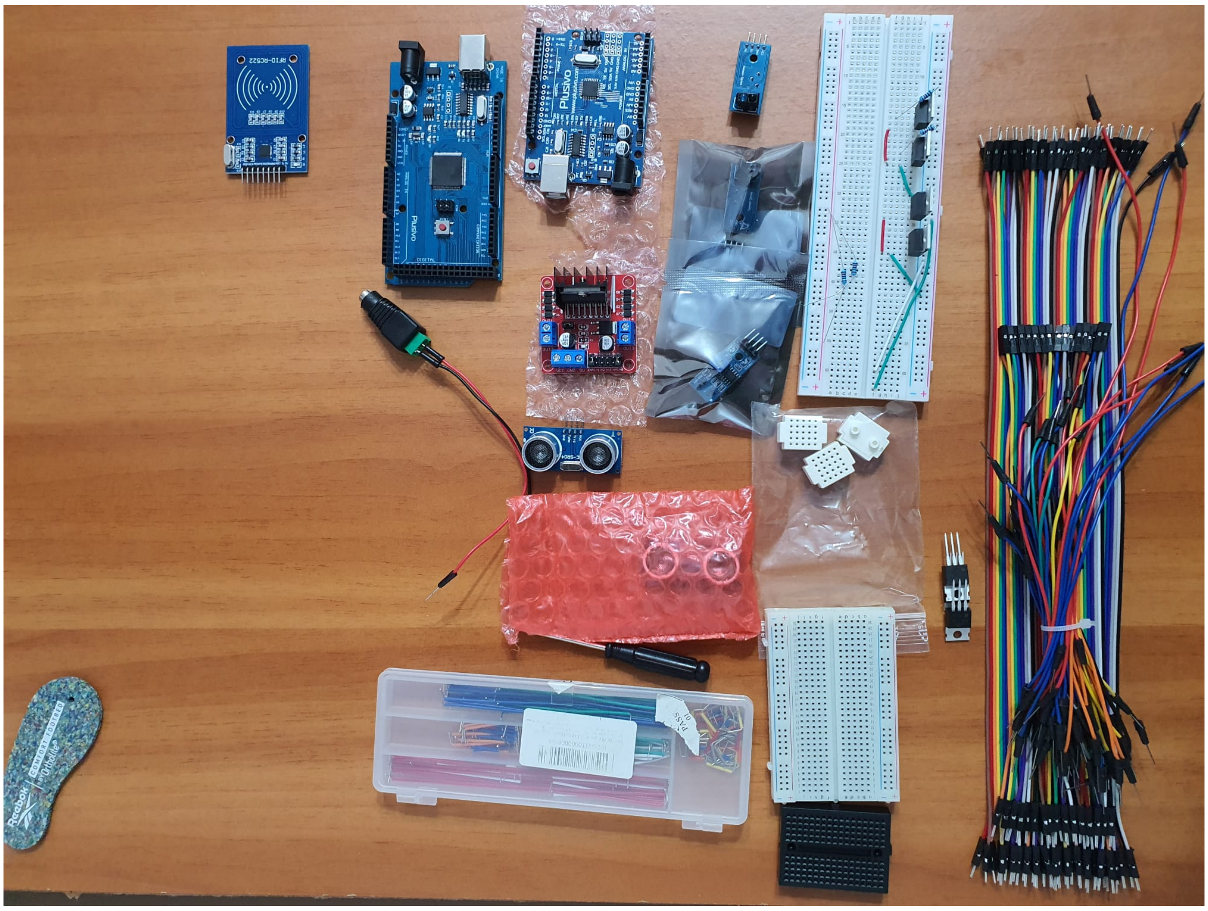
Schematic: Schematic

The schematic for the project was created using the Schematic of MEGA2560_Rev3 and the Pinout Mega2560rev3. Also, for the Uno part, the schematic is here and the pinout is here.

The schematic contains 5 main systems.

- The first one represents the board used: An **Arduino Mega with an ATMega2560**. This is the µC used to control the tracking system.
- The next system is **Direction & Speed Control System**. This is used for controlling the direction and speed of the train. A L298N Dual H-Bridge Motor Driver connected to a power supply represents the interface between the ATMega2560 and the railway.

- Following, we have **Position Control System** - a set of sensors to detect the position of train on the track. This way, the train can stop, change direction and speed.
- Besides these a **Track Switch System** that uses 4 relays controlled by 4 pins is used to change the direction of the track. To simulate more trains, a **RFID reader RC522** detects various tags and commands the rail switch accordingly.
- The last but not least, is the **Arduino Uno with the Braccio shield**. This helps to simulate the idea of “more trains” because it changes RFID tags that help the ATMega2560 with the RFID scanner to make a decision to change the direction of the rail.
- A sheet with components, datasheets and link to buy can be found here:components.xlsx
- An image with the most important components:

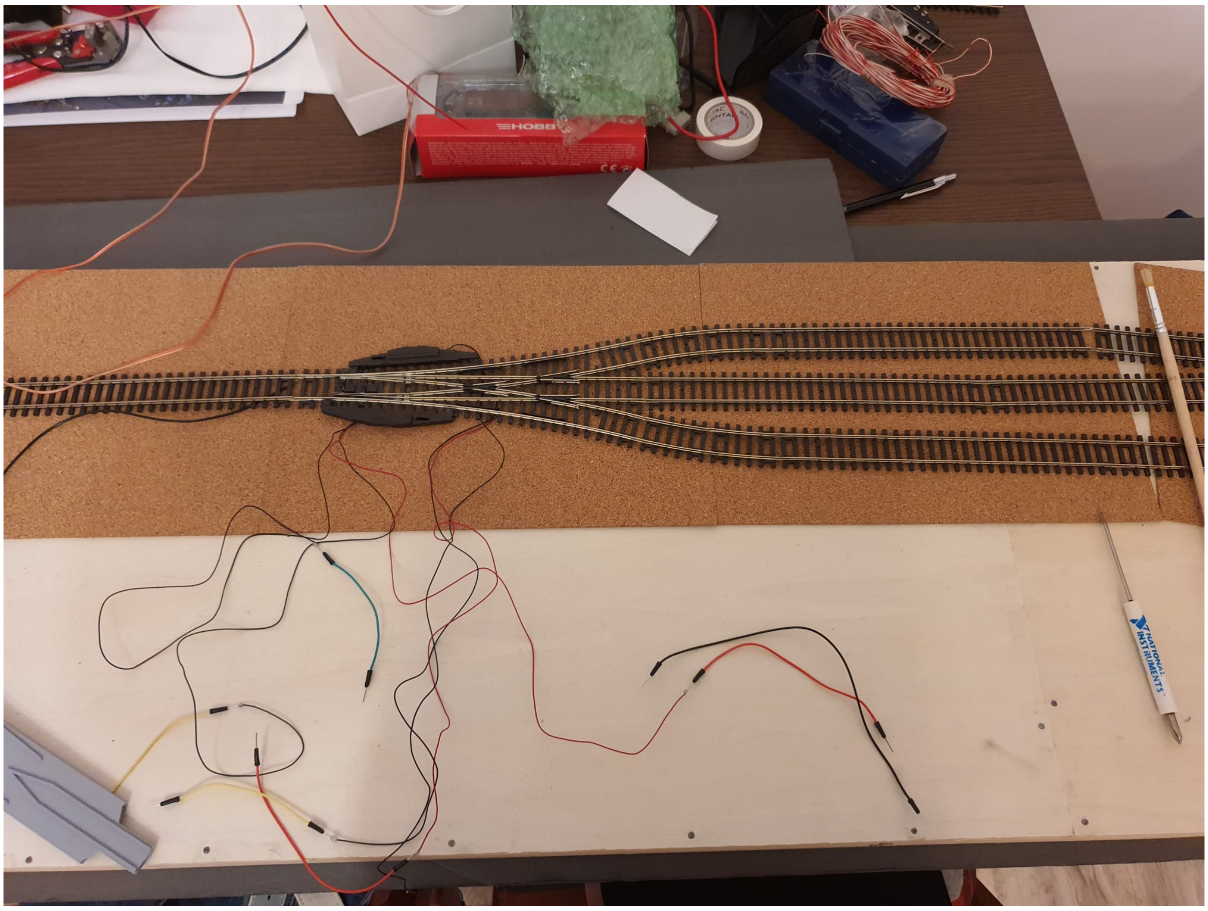


Hardware Description

- I'll start with the ATMega2560. Pins D2(PE4), D3(PE5), D4(PG5), D5(PE3) are outputs for the 4-channel relay module. This system helps changing the direction of the track in one of the three directions, using 2 switch machines that have 2 states. Those are powered by a 14V, 1.1A power supply.
- Next pins used are D6(PH3), D7(PH4) for the **ultrasonic sensor** placed at one of the extremities.
- The pins that are used for the direction and speed of the train are D42(PL7), D43(PL6), D44(PL5). PL5 allows **PWM** so it is perfect for speed modulation. Those are connected to the **l298n motor driver**

alongside a power supply of 12V, 0.5A DC.

- Pins from D49 to D53 are used for the **MFRC522** scanner. It communicates with it using **SPI**.
- For the other extremity of the railway is placed a simple **Infra Red Sensor** that detects when the train reaches a certain position.
- Here **Braccio**, a robotic arm made from 6 servomotors replaces the RFID tags. This is controlled through a **shield placed on Arduino Uno (ATMega328p)**. The communication between the 2 µC is facilitated by **I2C/TWI**. The pins used on the ATMega2560 are PD0 and PD1 and for the ATMega328p PC4 and PC5. Arduino Uno controls Braccio using 6 PWM signals for the 6 servos, PD3, PD5, PD6, PB1, PB2, PB3.



Decision Making

At first, I thought that an Arduino Uno was enough. Quickly I realized that using Braccio not only consumes all of my PWM pins but also overlaps with the RFID scanner. So I used an Arduino Mega instead. To be fair, initially I wanted to add two more infrared sensors and another servo to symbolize an intersection between cars and railway, but cable management and wire connectivity quickly became a problem, so that was the main reason I wanted an Arduino Mega. Also I didn't think that it would improve the project as it just adds a few sensors and one servo. Having both an Arduino Uno and a Mega got me thinking that I could do **I2C communication** between them in order to learn something new. I also thought that I could use **interrupts** to signal Braccio to start to move and that's what I did.

Software Design

The full code can be found on GitHub

Libraries used in the project:

- **MFRC522**: Used to interface with the RFID sensor. It reads RFID cards to determine which train rail to switch to, ensuring the train is directed onto the correct track.
- **SPI**: Required by the RFID reader, as it uses SPI for communication with the microcontroller.
- **Wire**: Used for I2C communication to control the Braccio robotic arm from the Master device. When the train arrives at the station, the Master sends a signal via I2C, triggering Braccio to pick up the old RFID card and replace it with a new one. (The Arduino Mega is the Master and the Uno is the slave).
- **Braccio**: A library used to simplify control of the Braccio robotic arm's six servo motors, allowing for easier/safer coordination of complex movements.
- **Servo**: dependency required by the Braccio library to control individual servo motors.

The trains keeps going until a sensor halts it's movement. Its motion is controlled based on feedback from two sensors:

- An infrared sensor positioned at the starting point.
- An ultrasonic sensor placed at the end of the track.

When the train reaches the end and is detected by the ultrasonic sensor, it automatically reverses direction and moves backward. If the infrared sensor at the starting point detects the train's return, the train stops and waits for further action. At this point, the Braccio robotic arm is triggered via I2C communication (with the Arduino Mega acting as the master and the Uno as the slave). Braccio then performs the following sequence:

- Picks up the old RFID card from the cargo bay.
- Replaces it with a new RFID card.

The new RFID card is then scanned by the RFID sensor, which determines the appropriate track the train should take next based on the card's data.

Bugs and difficulties encountered along the way of building such a challenging project:

- The RFID module can sometimes behave unreliably. Issues observed include intermittent failures to detect cards, entering an unresponsive or sleep state, and inconsistent readings
- Infrared Sensor: readings can be affected by ambient light in the environment. In darker conditions, the sensor tends to return lower values due to reduced infrared interference, while bright lighting or direct sunlight can cause higher or unstable readings. Also depending on the material properties, the sensor will give higher or lower values.
- Electrical Instability: due to imperfect connections, excessive voltage, or high current on the breadboard, one of the track switches overheated to the point where it physically melted. This highlights the importance of ensuring secure, low-resistance connections and staying within the rated voltage and current limits for all components. Breadboards, in particular, are not designed to handle high current and can become unreliable or even dangerous under such conditions.
- Cable management: having long tracks of wires and a big board it is difficult to have a great cable

management especially that some components (like MFRC522) have pins instead of holes.

Youtube videos

Two videos can be found here and here. The first one describes the components and the interaction between them and the second one describes how does the whole project work.

Conclusions and Concepts

- **GPIOs, Interrupts, PWM, ADC, SPI, I2C** directly used in the project through microcontrollers.
- **UART, Multimeter, Oscilloscope, Soldering and other instruments** used for testing, debugging.

.