Project Report

ENEL-351

Microcontroller System Design

E-Commerce Delivery Robot

Author:
Salman Shuaib

Submitted to:
Robert Martens
Dave Duguid

April 11, 2023

Salman Shuaib
200374212
April 11, 2023

Table of Contents

Salman Shuaib
200374212
April 11, 2023

1.0 Introduction

We are required to design an electronic system based on the Nucleo-64 microcontroller the STM32F10RB. The ENEL 384/351 microcontroller board (Nucleo-64) is a widely used platform for developing electronic systems. In this project, students are expected to create a functional electronic system that can accept dynamic inputs from the outside world, provide controlled outputs to the outside world, and solve a real-world problem. This project requires students to incorporate both digital and analog sensors, along with at least two input and output devices in addition to those provided on the Nucleo-64.

2.0 Project Description

A general overview of what the project is supposed to do is to follow a line sensor inside a pre-designed course by the user. I will also be creating a makeshift course for the robot to follow using cardboard with lines inside the course to guide it with the line sensor. This is to operate as simulation for a real-world situation of what the robot could be potentially capable of doing. The robot also has a proximity sensor that is responsible for detecting anything that is within a certain range of the robot which will then stop the robot from proceeding and hitting the obstruction. There will be a couple doors with LEDs on them that will have an LED light this is to signify to the light sensor whether this door requires a package to be delivered. The higher the intensity will mean a package will be delivered and a lower intensity will mean the package does not need to be delivered. The robot will then stop a certain distance from the desired "door" and signify a successful package delivery with and two green LED flashes on the breadboard. If the package is not required or after reading the LED on the door the robot will simply respond with two red LED flashes.

2.1 Main Device Functions

PWM for motors

a. This involves using the STM32 Microcontroller to implement PWM. Involving configuring our clock source for a timer and changing our clock frequency correctly to determine our desired PWM frequency.
b. Then I will configure the timer where I select the PWM output and the timer pre-scaler/period.
c. Next, I will be configuring our PWM channel involving selecting our I/O pin, setting the duty cycle, and then enabling the PWM channel.
d. The motors will be driving off a motor driver module itself the L298N.
e. Lastly, setting off our timer to generate the PWM signal and get our motors moving at our desired speed and frequency.

Line Sensor (Analog)

f. Since we already have ADCs in our microcontroller, we can use this to read analog outputs of our line sensor. We first connect our line sensors to V_{cc} to our 5V power source, GND and the OUT. The OUT pin will be connected to an ADC channel.
g. Once this is connected. I will be configuring the ADC which involves setting the ADC clock, resolution, and the sampling rate.
h. Next involves reading the ADC value to read the analog output of the line sensor. Involves a conversion and then reading the value.

Salman Shuaib
200374212
April 11, 2023

i. Lastly, is to read the ADC value to determine the position of the line sensor. This will include using a comparison of the ADV converted value to a threshold of sorts to determine where the line is (L, R, Center)

Ultrasonic Distance Sensor (Digital)

j. Again, we will begin with connections involving slightly more than the line sensor we had. V_{cc} to our 5V power source, GND, TRIG to a GPIO pin and ECHO to a timer input capture pin.
k. We then configure our pin to generate an ultrasonic pulse. Involves setting the pin mode, output type, speed, and pullup/pulldown configuration
l. Configuring our timer so it can measure the time between the ultrasonic pulse and echo signal. Involves setting timer clock source, pre-scaler, period, and input capture
m. Now we generate our ultrasonic pulse, once this timer is configured, we can generate the pulse by setting the TRIG pin to logic level high.
n. Next, we measure our echo signal for how long it takes for it to return. Involves starting the timer and waiting for it to trigger the input capture event.
o. Lastly, we can measure and find our distance between the sensor and the obstruction based on the time it takes for the pulse to travel.

Motion Sensor (Digital)

p. Make our connections: V_{cc} to 5V power source, GND and OUT to a GPIO pin.
q. Configuring our GPIO pin to detect our signal from the IR sensor. Involves setting the GPIO mode to pull-up/pull-down configuration.
r. Configuring our EXTI interrupt to ensure it can detect the rising edge of the IR sensor. Involves setting our EXTI line, mode, trigger, and its priority.
s. Interrupt handling: When EXTI is triggered the ISR should be executed. In this routine, we can set a flag to indicate when motion is detected (red LED will be prompted to flash until obstruction removed)

Photoelectric Sensor (Analog)

t. Make our connections: V_{cc} to 5V power source, GND and OUT to a GPIO pin.
u. Configure our GPIO pin to an ADC input pin (i.e., PA0).
v. Read the analog value from the ADC and display it using an LED.

Salman Shuaib
200374212
April 11, 2023

2.3 Block Diagram:

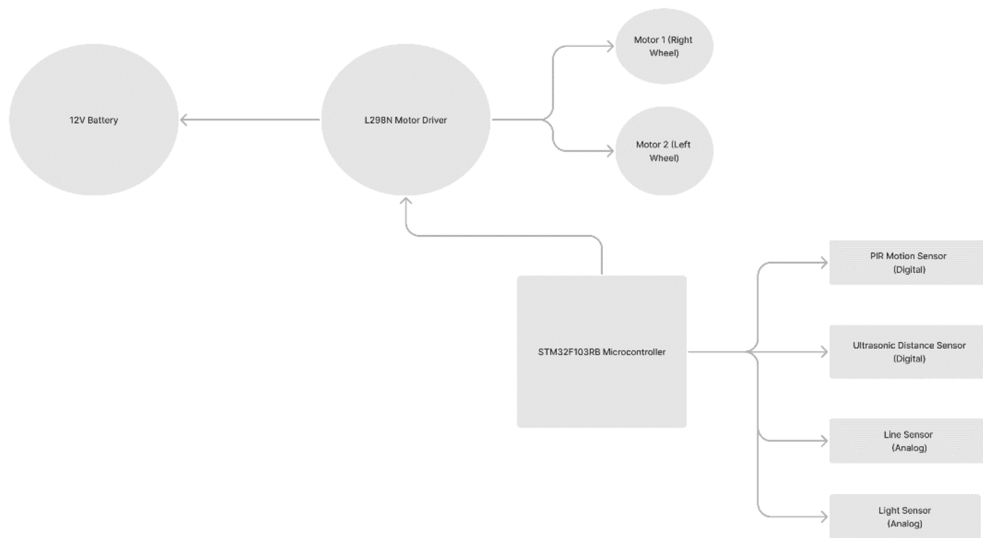

Figure 1. Block Diagram for E-Commerce Robot

2.4 Deviations from original functional specification

I originally planned on making use of the line sensor as my main analog sensor since it was required to navigate the course as well as a proximity sensor. This seemed a little excessive and I could not get this to work and due to time constraints primarily because of other projects in other classes I chose to scrap it. I instead mainly focused on getting my photoresistor to operate as my main analog sensor as well as my ultrasonic sensor as my primary digital sensor. Another notable change is that I did not have time to finish implementing PWM for my motors to control speed control. This would have been very helpful since my robot moves forward too quick and having speed control to remedy this would have been beneficial. The way in which my photoresistor also differs slightly. Originally, I had lights inside my course on the roof of each course pathway which would shine a light on the photoresistor. Depending on the light intensity and the analog value returned, this would then determine whether it would deliver a package or not. I also chose to implement a buzzer as my other output.

3.0 System Operation

3.1 Ultrasonic Sensor, Motors and Buzzer

Once the RESET button on the STM32F103RB is pressed, this will cause the motors to move forward and the soldered LED to turn on. There are essentially two states that the robot will showcase – object detected and no object detected. As the robot moves forward, the Ultrasonic sensor will look to detect something within a 40 cm range. While it does not detect anything it stays within a sleep function until something enters its range of detection. Once an object enters its range, the motors reverse polarity and begin to spin backwards. While this is happening, the soldered LED turns off and the active buzzer goes off until the object is outside of the range.

3.2 Photoresistor and LEDs

The alternate function of the robot is to detect light. This is also activated by hitting the RESET button as well. Although, these functions do not work simultaneously so they must be run separately. Once this function is activated, it is controlled by a single photoresistor and 4 LED's. The LEDs represent a range of approximately 2.7V to 3.7V. These values were chosen by connecting SCOPY to the photoresistor to determine the range of values it was reading when I took my finger off the photoresistor (3.7V) and when I put my finger on the photoresistor (2.7V). It is important to also note these values were determined in the microcontroller lab and the values could differ in another room depending on the light intensity. I chose to read the ADC values starting from 2.7, 2.9, 3.35 and 3.7V. Each of these values turns on an LED respectively. The hex values for each analog value were determined by taking the resolution of the ADC on the STM32F103RB which is 12 bits minus 1 and then taking the reference voltage of 3.3V and dividing them both. We then take the measured value we got off SCOPY and multiply it by the product of the divisors. From this, we receive a decimal value and we can then determine a hex value from this for comparisons.

4.0 Physical Construction

4.1 Drive System and Chassis

The drive system I chose was to use a rear-wheel-drive system with a caster wheel in the center of the robot chassis which is powered by a L298N Motor Driver. This motor driver is capable of handling up to 35V and is suitable if we choose to supply more voltage or an external power supply such as a battery. I think that having a front-wheel-drive system would have been more beneficial since it provides more control especially for turning the robot. If I consider if my line sensor this would have been very helpful since the line sensor requires a bit more turning capability. The drive system can move forward and backwards as well. This is done by reversing polarity and will cause both motors to reverse direction. The chassis of the robot was ordered from Spark Fun and included all parts required for construction as well as the two servo motors that were needed.

4.2 Input and Output Device Connections

The Ultrasonic digital sensor has 4 main pin connections, those being the V_{cc}, TRIG, ECHO, and GND. My V_{cc} and GND connection is made to a breadboard which is powered by the STM32F103RB microcontroller. My ECHO and TRIG pins are connected to general purpose input-output (GPIO) ports on the STM32 – those being PC11 and PC10 respectively.

My photoresistor analog sensor only has 2 main pin connections, those being a V_{cc} connection and a second connection to PA0 a GPIO port. This pin is necessary for ADC conversions and allowed me to read an analog value and create a range to determine comparisons. It is also important to note that the photoresistor is connected to a 10k resistor. This is because by adding a resistor with respect to the photoresistor we can get a variable voltage for it which causes a voltage drop and this allows us to measure the voltage across it using SCOPY.

My two outputs that I made use of were regular LEDs as well as an active buzzer. The LEDs all go to GPIO ports which can be referred to on the CAD schematic. The cathode side goes to GND. As for the buzzer, the positive terminal goes to PB8 and the negative terminal goes to GND.

Salman Shuaib
200374212
April 11, 2023

5.0 Troubleshooting

When performing basic testing, I made sure to test each part separately first to ensure they work on their own. This was in case the whole system was to fail I would still have something to showcase on demo day. A main component of testing was using SCOPY to ensure that values were being measured. This was very important for both my Ultrasonic sensor and Photoresistor. The Ultrasonic sensor TRIG and ECHO waveforms could be measure on SCOPY and we could see exactly when the pins go high which is crucial to get this sensor to work. For the photoresistor, I also heavily used SCOPY in order to determine the range of values that the photoresistor was getting. This ranges from approximately 2.7V to 3.7V.

5.1 First Troubleshoot - Motors

The first issue I ran into was getting my motors to work with L298N motor driver. This was because I did not fully understand the connections required to power the motor driver. I initially had only one GND connection from the motor driver. I also was using an external power supply, that being a 9V battery but ran into current drain issues with it. It was not until I tried directly powering the L298N from the 5V connection from the STM32 microcontroller that the motor driver worked correctly with the motors. I initially did not think of trying this since I feared that it would not draw enough current to drive the motors. I was also afraid of potentially damaging the STM32 and did not want to risk it.

5.2 Second Troubleshoot – Ultrasonic Sensor

This was the second state of troubleshooting and took the longest. Once I finished soldering my sensor on the solderable breadboard, I made sure to check the connectivity points using a DMM. Once this checked out, I began by connecting the ECHO pin of the sensor to SCOPY while also passing a signal into the TRIG pin. This was to make sure that ECHO was reading as intended. I also ran into problems with false triggers at times and made sure to adjust the sensor correctly straight so my robot would not interfere with other objects. Another issue I had was that it was triggering at times I did not need it to. I solved this by adjusting the settings for the TRIG, specifically by adjusting the pulse duration and using a sleep function when the ultrasonic sensor was not detecting something in the programmed range of 40 cm.

5.3 Third Troubleshoot – Photoresistor

The last state of troubleshooting was the analog sensor. This was because I had to determine the correct hex values needed to ensure that the LED outputs were activated. Initially, I was guessing values and this proved to be unsuccessful. After reviewing Lab 3 – Analog to Digital Conversion, I recalled that we connected to PA0 to read values and turn on LEDs. This provoked me to connect a lead from one end of the photoresistor to PA0 and to SCOPY to measure the values. I was able to gather values but did not know how to determine the hex values and once again review Lab 3. The formula I initially was using to determine a decimal value from the measured value was:

$$\frac{Resolution\ of\ the\ ADC}{System\ Voltage} = \frac{ADC\ Reading}{Analog\ Voltage\ Measured}$$

This equation proved to be the correct one and so I calculated our resolution as $2^{12}-1 = 4095$. I used a system voltage of 5V but this turned out to be wrong. I had LEDs connected at this point already so I could easily check if the hex value was accurate. Just by trial and error, I used 3.3V instead of

5V and this proved to work. This gave me a decimal value that I simply converted to hex and used in my read ADC value comparisons.

6.0 Design Improvements

My design did most aspects of my original idea that I wanted to implement but I would improve a lot. One of the main things I would do is to implement PWM successfully into my system. This is because when my motors activate, they move a lot faster than needed. I would implement speed control primarily when moving forward since this would also provide more accuracy in detecting objects with the Ultrasonic sensor. I also definitely would have looked to implement my line sensor as intended. After realizing how simple it was to implement the photoresistor as an analog sensor it made me regret not putting more time into figuring it out. It would help a lot with steering the robot and complete its original task as intended.

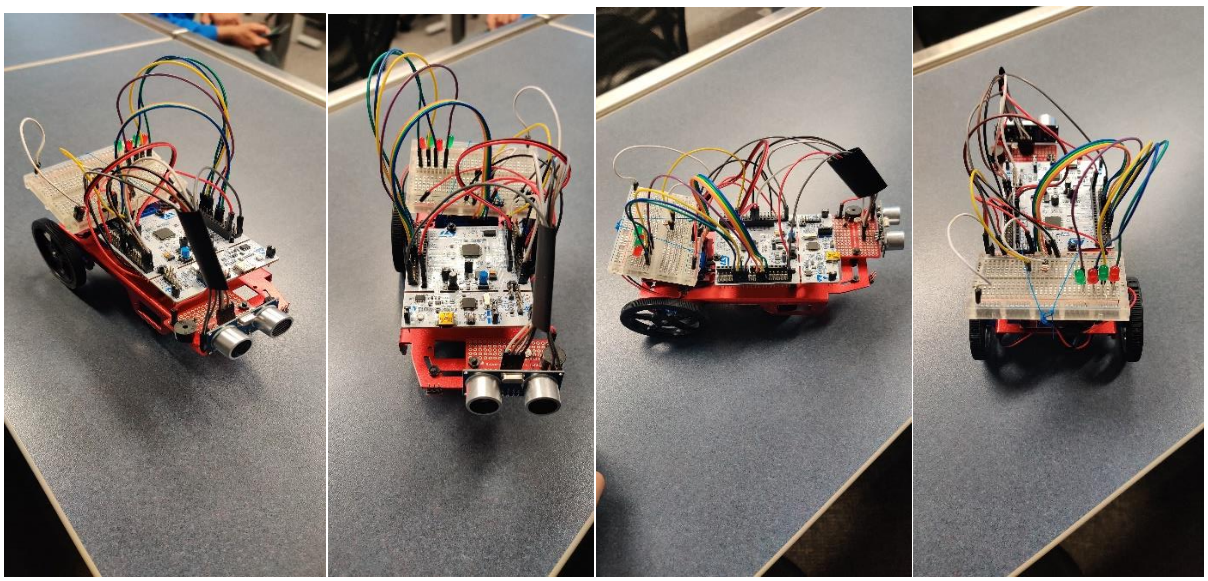

Figure 1. Final Build of E-Commerce Robot

7.0 Conclusion

This report details the design, construction, performance and troubleshooting of my E-Commerce Robot. I made use of different topics covered in both the class and labs, such as ADC, GPIO and a bit of PWM even though I did not implement it. The system incorporates digital and analog sensors, along with at least two input and output devices. The motors, ultrasonic distance sensor, photoresistor were implemented successfully, and the physical construction was also done well with a drive system and chassis, input, and output device connections.

There were a few troubleshooting issues that arose during the project, including problems with the motors, ultrasonic sensor, and photoresistor. These issues were resolved effectively, and the system was able to function properly.

One improvement that could have been made to the project is the addition of more detailed documentation, such as schematics and diagrams, to make the project easier to understand for others. Overall, this project was a great learning experience, and the electronic system designed demonstrates the ability to solve real-world problems using microcontrollers and various sensors.