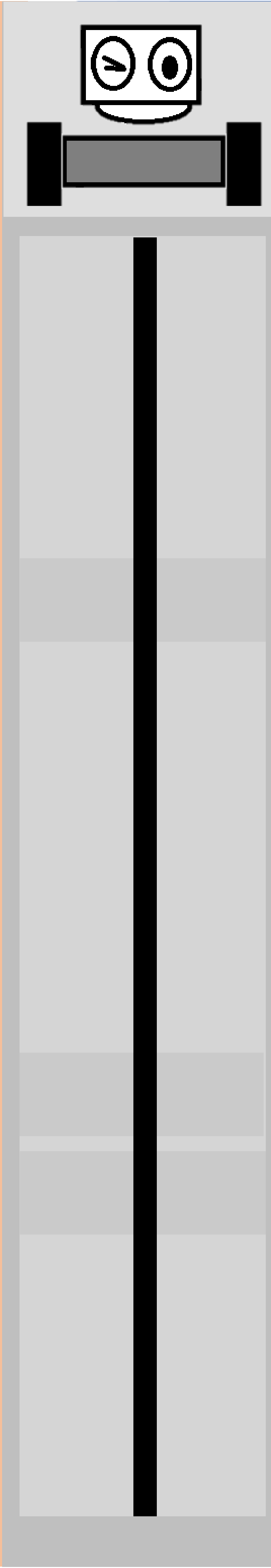
LINE FOLLOWER ROBOT

AHMAD FARIZ BIN FAUZI

Ts. Hj. MOHD NORDIN BIN MOHD JANI

SAIFFUL BAHARI BIN OMAR



LINE FOLLOWER ROBOT

e-book

Ahmad Fariz bin Fauzi

Ts. Hj. Mohd Nordin bin Mohd Jani

Saifful Bahari bin Omar

POLITEKNIK MELAKA

2023

LINE FOLLOWER ROBOT

WRITER
Ahmad Fariz Bin Fauzi (K)
Ts. Hj. Mohd Nordin Bin Mohd Jani
Saifful Bahari Bin Omar

EDITOR
Dr. Rosnani Binti Affandi (K)
Pn. Hairani Binti Ahmad Zainuldin
Tuan Syed Alwi Al-Qudri (InoMa)

DESIGNER
Ahmad Fariz Bin Fauzi

APPLICATION PUBLISHER AND DEVELOPER
Ahmad Fariz Bin Fauzi
Ts. Hj. Mohd Nordin Bin Mohd Jani

TERBITAN EDISI 2023
Perpustakaan Negara Malaysia Cataloguing-in-Publication Data
e-ISBN 978-967-0838-95-3

Published by:
Politeknik Melaka,
Kementerian Pengajian Tinggi,
No.2 Jln Ppm 10,
Plaza Pandan Malim
75250 Melaka
https://www.polimelaka@edu.my
http://celt.edu.my/PAGE-EBOOK-POLYCC

Cataloguing-in-Publication Data

Perpustakaan Negara Malaysia

A catalogue record for this book is available from the National Library of Malaysia

eISBN 978-967-0838-95-3

ACKNOWLEDGMENT

Praise Allah s.w.t for His permission, this e-book has been successfully produced. Thanks also to our friends who are directly and indirectly involved in the preparation of this e-book. Without the high commitment of all parties, especially the Department of Electrical Engineering, the Malacca Polytechnic, this book cannot be realized as a scientific reading material.

All materials used to produce this e-book are in the Malacca Polytechnic. If readers want to produce robots and want to buy essentials they are all for sale in the store as well as online.

Hopefully, this e-book, a little bit can help anyone who wants to start a robot project.

Ahmad Fariz Fauzi, Ts. Hj Mohd Nordin Mohd Jani & Saifful Bahari Omar
Department of Electrical Engineering
Politeknik Melaka

ABSTRACT

This **Line Follower Robot** e-book is a reading material that provides the best input to readers in the quest to produce a robot using NANO's Arduino controller, MX1508 driver motor, and IR TCRT5000 3 array sensor. This book contains a breakdown of chapters detailing each part needed in robot production and coding. Interestingly in this book, all the coding produced is the basic code of the Arduino only and no library is used. The main purpose of using basic coding is to facilitate readers' understanding of how easy it is to create Arduino codes specifically to produce robot line followers. It is undeniable that encoding using the PID control system produces more effective robots, but readers must first understand the basics of analog and digital sensor encoding and PWM encoding. This book contains a tutorial that applies all the necessary coding basics without a PID controller.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

WHAT IS LINE FOLLOWER ROBOT?

Ahmad Fariz bin Fauzi

INTRODUCTION

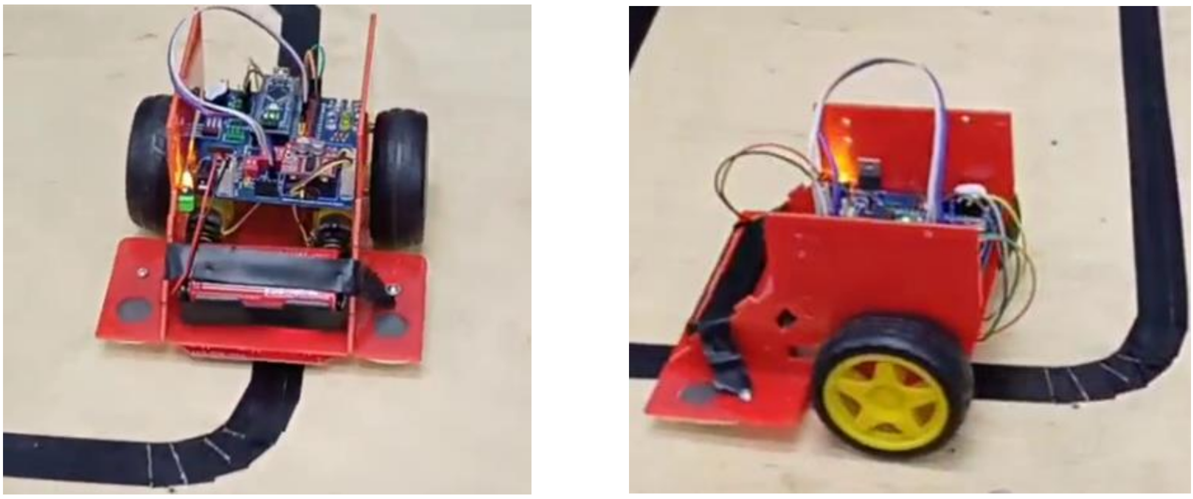


FIGURE 1: LINE-FOLLOWER ROBOT

A line-follower robot is one of the mobile robot applications that is widely used around the world based on its purpose, such as competition, industrial, hobbyist, and education. The basic line-follower robot is shown in **FIGURE 1: LINE-FOLLOWER ROBOT**.

Over the years, there have been more than 3 million innovations and inventions in robotic technologies to make a line-follower mobile robot more accurate in detecting lines for product inspection, product transportation to other places in factories, and automatic guided mobile vehicles (GlobalData, 2023).

The line-follower robot consists of a combination of microcontrollers such as the Arduino, sensors such as an IR sensor, and actuators such as a DC motor. The microcontroller is used to control the actuator based on what the sensors detected before it (Mandal, 2023).

A colour sensor is used to detect colour lines or surfaces. Meanwhile, an IR sensor is used to detect black-and-white lines or surfaces.

The most common colours used as a track or marking line for a line-follower robot are black and white compared to other colours. But in technical terms, black and white are excluded from any colour, even though white includes all the colour spectrum and black is a combination of all colour pigments on paper (Jimmy Presler, 2023).

There are a lot of line follower tracks based on what the industrial production needs or competition requirements. The basic track is divided into a line and a pattern, or combination, as shown in **FIGURE 2A-H: LINE AND PATTERN**.

FIGURE 2A: LINE AND PATTERN (STRAIGHT)

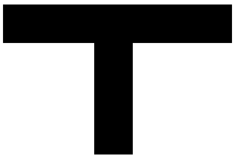


FIGURE 2B: LINE AND PATTERN (JUNCTION)

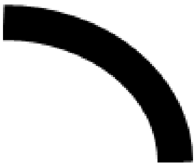


FIGURE 2C: LINE AND PATTERN (ARC)

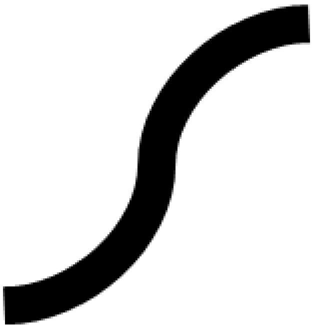


FIGURE 2D: LINE AND PATTERN (CURVED)

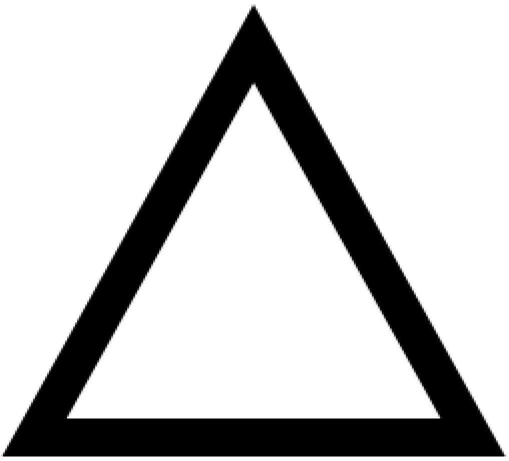


FIGURE 2E: LINE AND PATTERN (TRIANGLE)

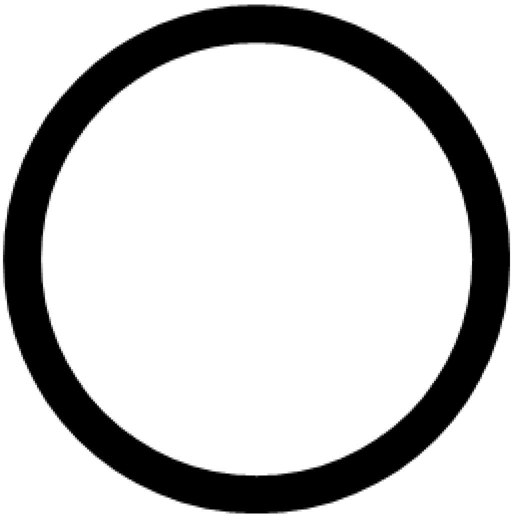


FIGURE 2F: LINE AND PATTERN (CIRCLE)

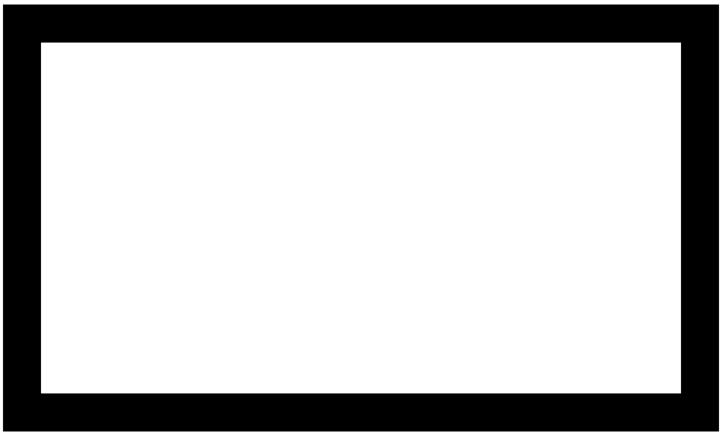


FIGURE 2G: LINE AND PATTERN (RECTANGLE OR SQUARE)

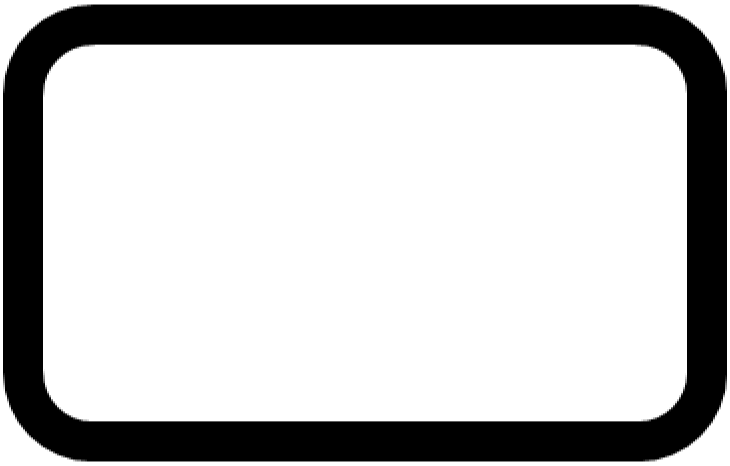


FIGURE 2H: LINE AND PATTERN (ROUND CORNER RECTANGLE OR SQUARE)

Though black and white are the colours that are widely used as a track line, both are not colours when you define them in physics because black and white don't have a specific wavelength (Murmson, 2021). So, what are the spectrum colours that have a specific wavelength?

COLOUR WAVELENGTH

Based on the colour wavelength, visible light is divided into seven specific spectral colours regarding its wavelength (Murmson, 2021).

The corresponding colours from the lowest wavelength to the longest wavelength are VIOLET, INDIGO, BLUE, GREEN, YELLOW, ORANGE, and RED (Volchko, 2018).

All seven colours are the same as a rainbow as shown in **TABLE 1: COLOUR WAVELENGTH**.

TABLE 1: COLOUR WAVELENGTH

COLOR	NAME	WAVELENGTH
🟪	Violet	380 – 450 nanometres
🟪	Indigo	420 - 440 nanometre
🟦	Blue	450 – 495 nanometres
🟩	Green	495 – 570 nanometres
🟨	Yellow	570 – 590 nanometres
🟧	Orange	590 – 620 nanometres
🟥	Red	620 – 750 nanometres

BLACK AND WHITE LINE AND SURFACE

Though the line follower robot sensor can be designed to detect whatever colour of a track, the best track line is black and white. That is why the line follower robot's specific function is to detect a line that is either black or white, that is drawn on any kind of surface that is suitable to the robot (M. Pakdaman, 2010) based on **FIGURE 3: BLACK AND WHITE**.

White
Black

FIGURE 3: BLACK AND WHITE LINE

The main reason for using a black or white line or surface is related to a light wave. The black line or surface absorbs all spectrum light, and the white line or surface reflects all spectrum light (ROBU.IN, 2021) based on **FIGURE 4: WAVELENGTH OF BLACK AND WHITE LINE OR SURFACE**.

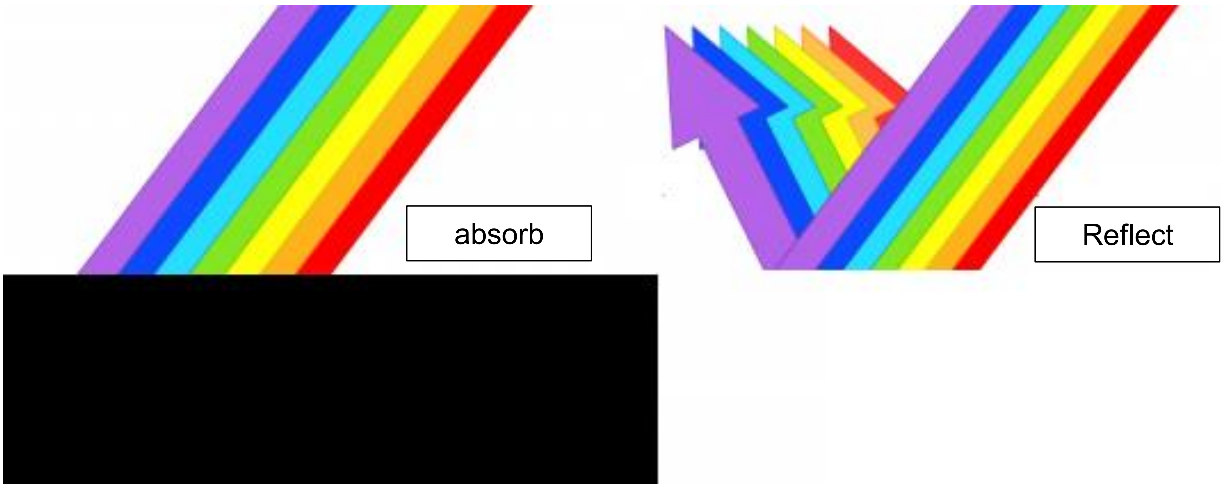


FIGURE 4: WAVELENGTH OF BLACK AND WHITE LINE OR SURFACE

MICROCONTROLLER - ARDUINO NANO

Ahmad Fariz bin Fauzi

INTRODUCTION

All robots need a control system before they can start doing things by manipulating the environment. So, the control system you use in any kind of robot is a microcontroller.

Microcontrollers are the main parts of robotics. A microcontroller is an integrated circuit that has a microprocessor unit, a memory system unit, and some control device pin-out (javatpoint, 2023).

With a microcontroller, you can produce a lot of electronic projects or systems, such as robot line-followers. There are a lot of microcontrollers on the market such, as Arduino, STM32, ATMEL, PIC Microcontroller, and AVR Microcontroller (School, 2019).

In this e-book, you only cover Arduino NANO as a microcontroller to set up the line follower robot program code because of its small dimension board but still covers complex code applications. There are a lot of types of Arduino NANO, such as Arduino Nano 3.0, Arduino Nano Every, Arduino Nano 33 IOT, and Arduino NANO 33 BLE (Kerstin, 2023).

You can use any kind of Arduino NANO to complete the line follower robot task, but which Arduino NANO is the best board you should use for the Line Follower Robot?

It depends on the task and budget. If you are on a low budget and just want to build a simple robot such as the Sumo Robot, or Line Follower Robot without any extraordinary communication, Arduino NANO V3.3 is just enough. But if you are born rich and don’t care about money at all and, at the same time, you want to build a system that uses a lot of extraordinary communication, such as Bluetooth, Wi-Fi and can be programmed using Python, Arduino NANO 33 BLE Sense is the best choice.

All the specifications and the estimated price before buying an Arduino NANO board are based on **TABLE 2: ARDUINO NANO SPECIFICATION AND PRICE**. All estimated prices are based on current prices at Shopee, and all specifications are based on Arduino (Arduino, 2023).

TABLE 2: ARDUINO NANO SPECIFICATION AND PRICE

Type of Arduino NANO	Specification	Estimate Price
Arduino Nano V3.0	• Size: 45 mm (length) x 18 mm (width) • Weight: 5g • Processor: ATMega328P or AtMega168 • Frequency bandwidth: 2.4Ghz • Interface: Micro-USB or Mini-B USB • Regulated Power Supply: 6-20Vdc • Fixed Power Supply = 5Vdc • Number of pins = 30 • Integrated Wi-Fi: No • Integrated Bluetooth: No	RM 10.00
Arduino Nano Every	• Size = 45 mm (length) x 18 mm (width) • Weight: 5g • Processor = ATMega4809 • Interface to the board = Micro-USB • Regulated Power Supply: 7-21Vdc • Fixed Power Supply = 5Vdc • Number of pins = 30 • Integrated Wi-Fi: No • Integrated Bluetooth: No	RM 110.00

Arduino Nano 33 IOT	• Size = 45 mm (length) x 18 mm (width) • Weight: 5g • Processor = SAMD21G18A • Interface to the board = Micro-USB • Regulated Power Supply: 5-18Vdc • Fixed Power Supply = 3.3Vdc • Number of pins = 30Number of pins = 30 • Integrated Wi-Fi: Yes • Integrated Bluetooth: Yes • IMU (accelerometer and gyroscope): Yes	RM 150.00
Arduino NANO 33 BLE	• Size = 45 mm (length) x 18 mm (width) • Weight: 5g • Processor = nRF52840 • Interface to the board = Micro-USB • Regulated Power Supply: 5-18Vdc • Fixed Power Supply = 3.3Vdc • Number of pins = 30 • Integrated Wi-Fi: No • Integrated Bluetooth: Yes • IMU (accelerometer and gyroscope): Yes • Python Support: Yes • Arm Bed OS: Yes	RM60.00
Arduino NANO 33 BLE Sense	• Size = 45 mm (length) x 18 mm (width) • Weight: 5g • Processor = SAMD21G18A • Interface to the board = Micro-USB • Regulated Power Supply: 5-18Vdc • Fixed Power Supply = 3.3Vdc	RM 700.00

	• Number of pins = 30 • Integrated Wi-Fi: Yes • Integrated Bluetooth: Yes • IMU (accelerometer and gyroscope): Yes • Python Support: Yes • Built-in Microphone: Yes • Proximity & Gesture: Yes • Barometric: Yes • Pressure & Temperature: Yes	

So, the microcontroller you use for this Line Follower Robot is the Arduino NANO V3.0 as in **FIGURE 5: ARDUINO NANO V3.0**.

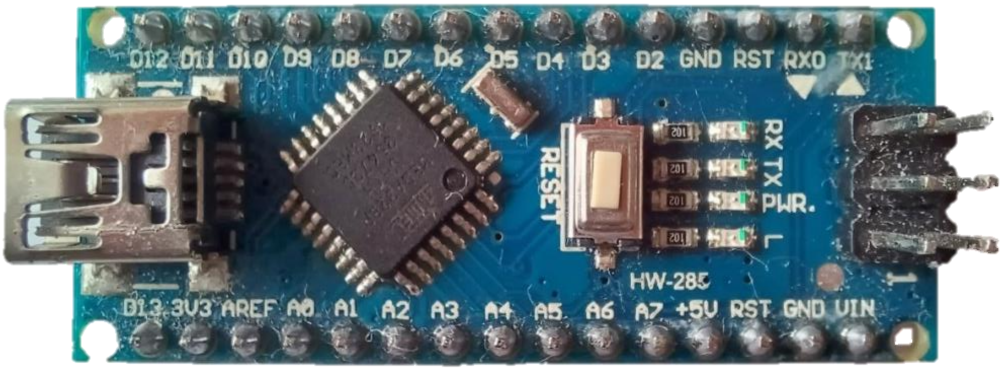

D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 GND RST RX0 TX1
RESET
RX
TX
PWR.
L
HW-285
D13 3V3 AREF A0 A1 A2 A3 A4 A5 A6 A7 +5V RST GND VIN

FIGURE 5: ARDUINO NANO V3.0

The main reason you use the Arduino Nano is because of the cheapest and most affordable price to have. Also, because of its small size board with complete multifunction such as a compact design, easy to use, versatile, power efficient, and capable of integration (Kerstin, 2023).

FEATURES

Now you have your magnificent low-budget Arduino NANO V3.3 board. So, what is the next step? How do you define the board? Which pin should you use and how to connect the pin to the sensors and actuator?

The next step is to familiarize yourself with the pins. After that, it will be easy for us to build any system using the Arduino NANO V3.3. Based on **FIGURE 6: ARDUINO NANO V3.3 PINOUT**, there are two groups of pins, such as primary pins and secondary pins. Both groups are the necessary pinout terminals for us to manipulate based on what system you want to build. The primary pins are digital pins, analog pins, and default pins. Meanwhile, the secondary pin is the communication pin, or you call it the In-Circuit Serial Programming Pin (ICSP).

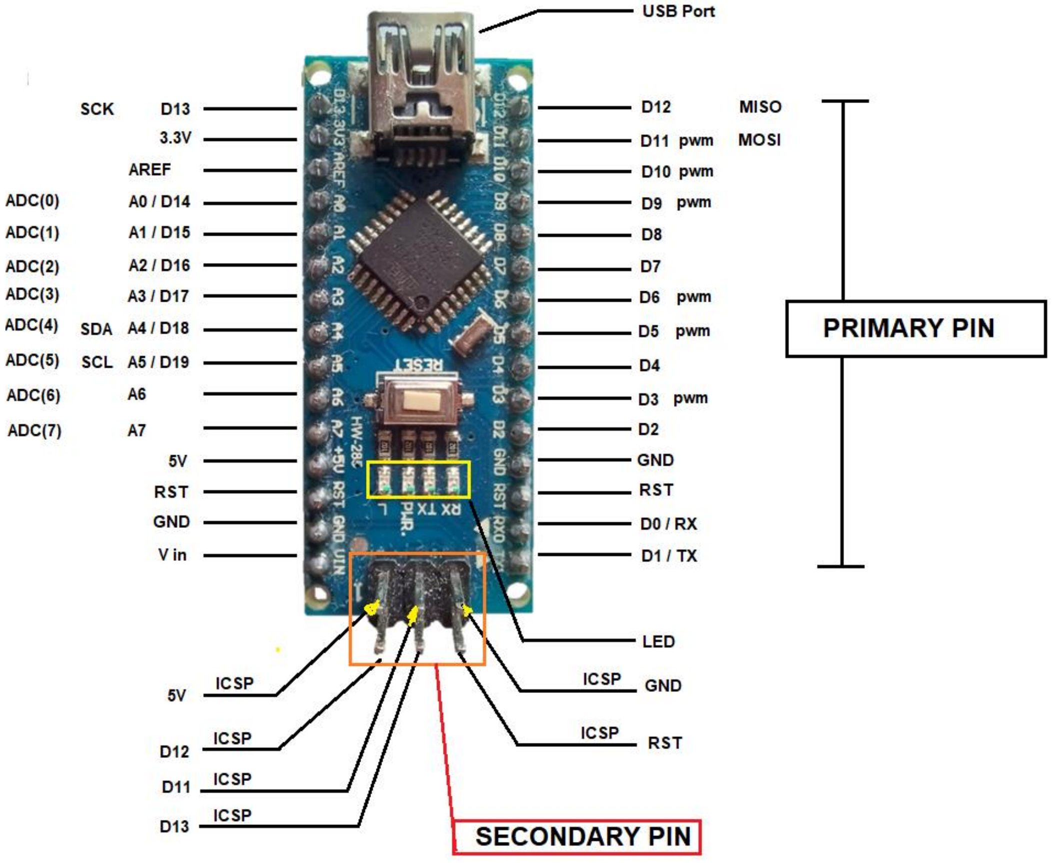


FIGURE 6: ARDUINO NANO V3.3 PINOUT

Based on **FIGURE 7: BUILD-IN SMD LED**, there are also four SMD LEDs on board for indicator information. All the LEDs are shown on board as L for built-in LED (connected to pin D13), PWR for a power indicator that shows the Arduino NANO V3.3 has the required power supply to turn ON, TX for data transmitted from Arduino NANO V3.3 to the computer and RX for the data receive from a computer to the Arduino NANO V3.3. In normal conditions, all the LEDs aren't. L LED is a controllable LED based on your code writing. The rest of the LEDs are automatic LEDs. When the power supply is connected to the Arduino, the power LED will lid. When you do programming and uploading from a computer to the Arduino, the TX will lid, and the RX LED will blink. So, from this LED indicator, you can realize something is not normal when either one LED isn't on or blinking.

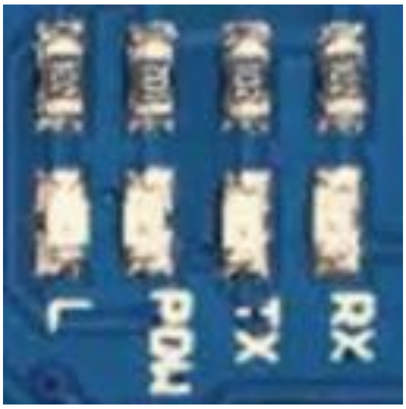
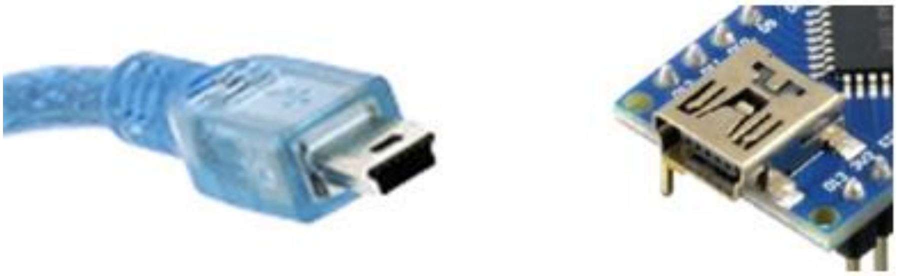


FIGURE 7: BUILD-IN SMD LED

You also must know about the USB Port type for Arduino NANO V3.3. The typical type of USB for Arduino NANO is Mini-B as shown in **FIGURE 8: MINI-B USB TYPE**. Without this port (female) and the connector(male), you can't transfer any code writing from a computer to the Arduino NANO V3.3.



Male   Female

FIGURE 8: MINI-B USB

But nowadays, there are many types of USB ports on Arduino NANO such as USB type C and micro-USB based on **FIGURE 9: MICRO-USB** and **FIGURE 10: TYPE-C**.

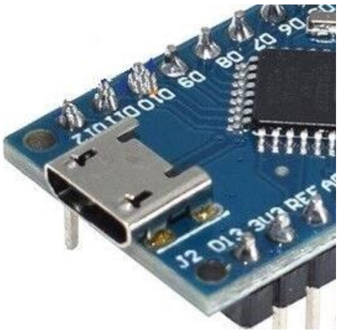


FIGURE 9: MICRO-USB

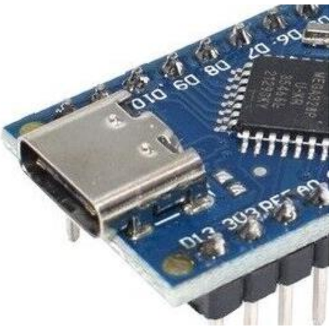


FIGURE 10: TYPE-C

Just forget about microcontroller ports unless you want to know more about the ATMega328 or ATMega198 microcontroller and want to design a new board using the specific microcontroller. By the way, the microcontroller ports are shown in **FIGURE 11: MICROCONTROLLER PORTS.**

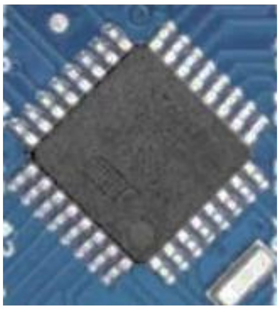


FIGURE 11: MICROCONTROLLER PORTS

COMPONENT – MOTOR DRIVER (MX1508)

Ahmad Fariz bin Fauzi

INTRODUCTION

One of the low-cost motor drivers in the market that can control any kind of DC motor within a 2A current rating. The reason why the MX1508 IC is most suitable to control any kind of DC motor is because its 16-pin IC has an integrated H-bridge designed with power MOSFET. It also prevents any kind of malfunction due to a float input pin with thermal protection on its board (Components101, 2021).

The MX1508 board is shown in **FIGURE 12: MX1508 MOTOR DRIVER.**

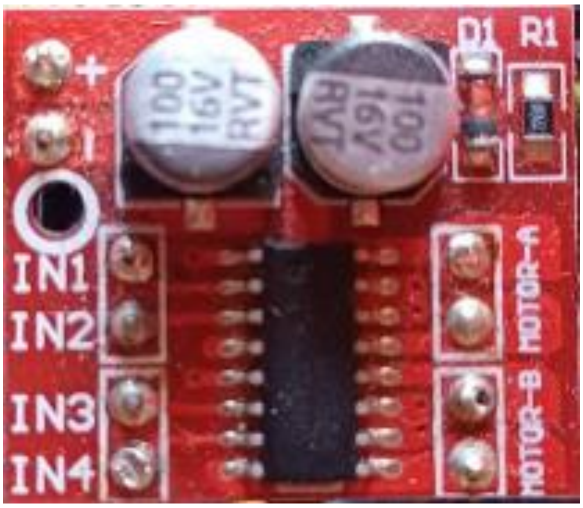
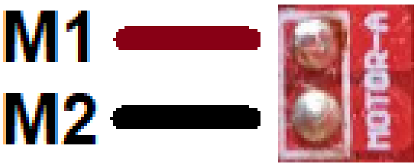
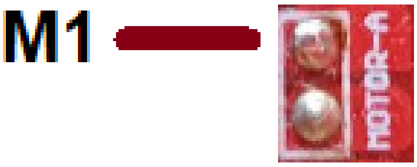
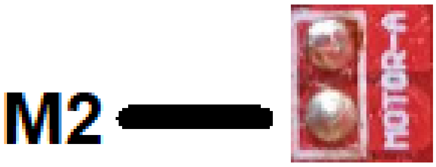


FIGURE 12: MX1508 MOTOR DRIVER

There are four IN pins on the board two pins for Motor A two pins for Motor B and two pins for a 2-10Vdc power supply. This board has an advantage because it is suitable to control two motors at the same time independently. The speed of the motor is controlled using the IN-pin connection with a Pulse Width Modulation (PWM) (Trolove, 2018).

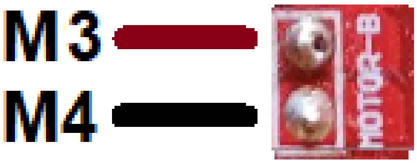
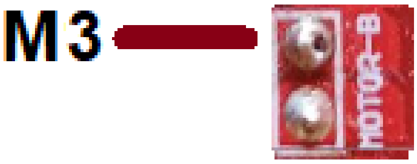
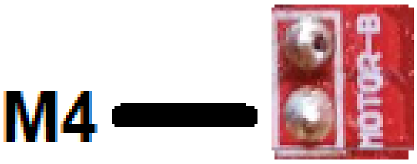
The truth table for the MX1508 based on **TABLE 3: TRUTH TABLE FOR MOTOR A** shows how to run the first DC motor. M1 is connected to a motor terminal that has a red dot and M2 is connected to another motor terminal with no red dot or vice versa based on your design.

TABLE 3: TRUTH TABLE FOR MOTOR A

Motor A Rotation	IN1	IN2
Motor Stop or Brake 	LOW HIGH	LOW HIGH
Motor Forward (M1) 	HIGH	LOW
Motor Reverse (M2) 	LOW	HIGH

Meanwhile, the truth table for the MX1508 based on **TABLE 4: TRUTH TABLE FOR MOTOR B** shows how to run the second DC motor. M3 is connected to a motor terminal that has a red dot or positive polarity and M4 is connected to another motor terminal with no dot or negative polarity.

TABLE 4: TRUTH TABLE FOR MOTOR B

Motor B Rotation	IN3	IN4
Motor Stop or Brake 	LOW HIGH	LOW HIGH
Motor Forward (M3) 	HIGH	LOW
Motor Reverse (M4) 	LOW	HIGH

LOW means 0V for digital condition and HIGH means 5V for digital condition. But for analog conditions, instead of using LOW and HIGH, you use “i” based on the PWM based on **TABLE 5: TRUTH TABLE FOR MOTOR A & B (PWM)**.

TABLE 5: TRUTH TABLE FOR MOTOR A & B (PWM)

Motor A Rotation (PWM)	**IN1**	**IN2**
Motor Forward (PWM) (M1) 	i	LOW
Motor Reverse (PWM) (M2) 	LOW	i
Motor B Rotation	**IN3**	**IN4**
Motor Forward (PWM) (M3) 	i	LOW
Motor Reverse (PWM) (M4) 	LOW	i

FEATURES

This small motor driver specification can be described based on **TABLE 6: MX1508 MOTOR DRIVER SPECIFICATION**.

TABLE 6: MX1508 MOTOR DRIVER SPECIFICATION

No	Details	Specification
1	Power Supply (Vdc)	
	i) DC Voltage Input	2V – 10V
	ii) DC Voltage Output	1.8V – 7V
	iii) Operating DC Current	1.5A
	iv) Peak DC Current	2A
	iv) Standby DC (Low current)	<0.1 micro-Ampere
2	Size	
	Weight	2g
	Length	24.7 mm
	Height	21 mm
	Width	5 mm

From Table 6, you realize that this motor driver is a good motor driver for a small line follower robot because it uses a small amount of DC and space.

COMPONENT – 3 ARRAY IR SENSOR (TCRT5000)

Ahmad Fariz bin Fauzi

INTRODUCTION

The infrared (IR) sensor is a well-known technology used in your daily life and all industries for its specific purpose. It has low power consumption and a lot of features that are suitable especially for mobile robot technology (Robocraze, 2022).

The IR sensor is an active sensor that uses your lighting source to measure distances or objects (AHMAD FARIZ, 2023).

For the line follower robot, you must use a sensor that can detect an object or line in the shortest range. For the time being, the IR sensor is the best sensor used in any mobile robot to detect an object at a very short range (AHMAD FARIZ, 2023).

That is why there are no sensors that can beat the function of an IR sensor when it comes to detecting a very close object.

Another advantage of this sensor is that it can detect black or white surfaces, based on its design range specifications. So, it is convenient for tracking the line or track on a flat surface.

Nowadays, there are many types of IR sensors sold on the market. The basic one is the single channel IR Sensor based on **FIGURE 13: SINGLE CHANNEL IR SENSOR (FOR LINE FOLLOWER)**. This type of sensor is used widely for educational and research purposes or for hobbyists to design a new system.

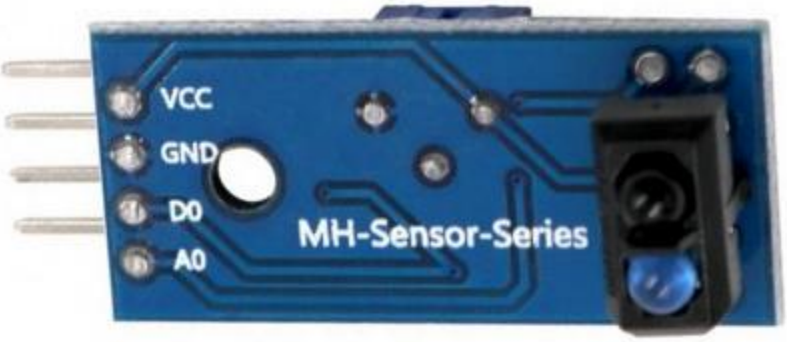
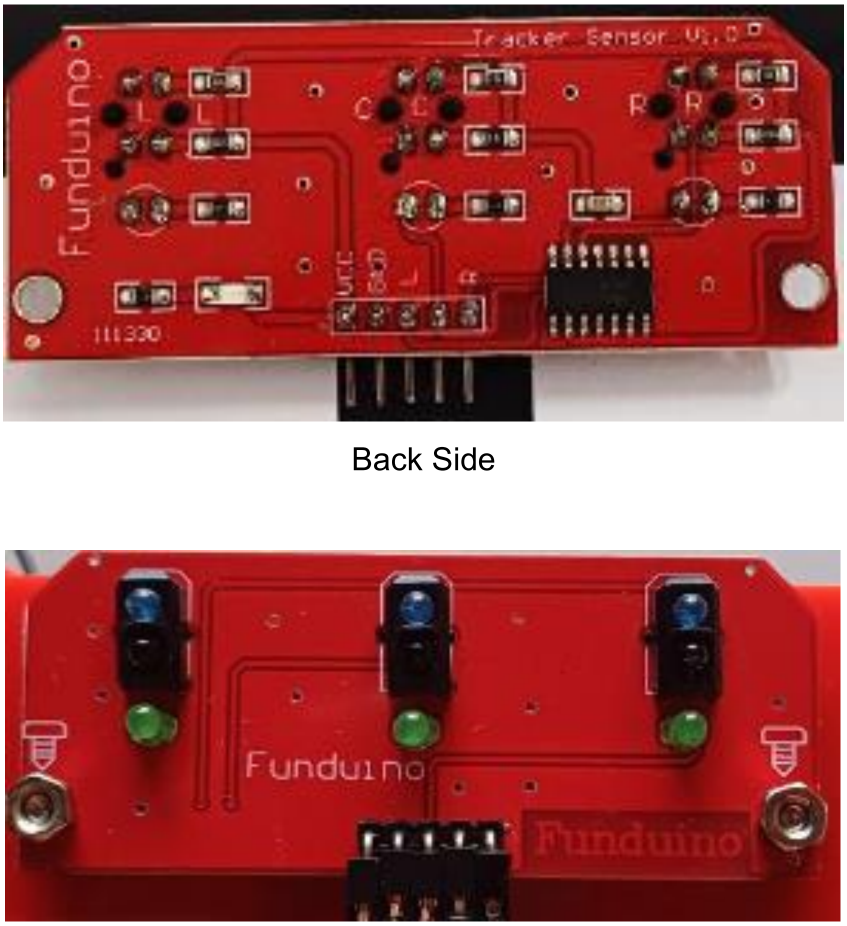


FIGURE 13: SINGLE CHANNEL IR SENSOR (FOR LINE FOLLOWER)

But if you want to use more than one IR sensor for a line-follower mobile robot, the best way is to get a specific multi-array IR sensor module like 3 Array IR Sensor shown in **FIGURE 14: 3 ARRAY IR SENSOR MODULE**.



Back Side

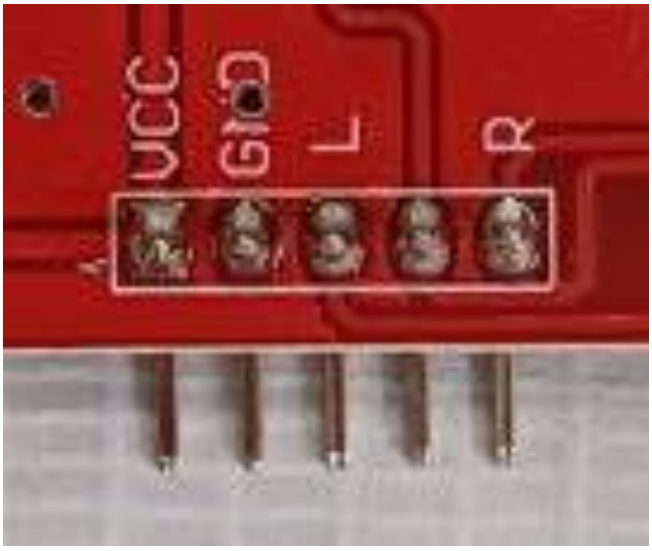
Front Side

FIGURE 14: 3 ARRAY IR SENSOR MODULE

FEATURES

The feature for this IR sensor is based on **TABLE 7: TCRT 3 ARRAY IR SENSOR**.

TABLE 7: TCRT ARRAY IR SENSOR

No	Details	Specification
1	Power Supply (Vdc)	5V
2	TCRT5000 built-in sensor	3 unit
3	Mode	Schmidt Trigger
4	Pins  	5
5	Built-in LED (Green)	3

COMPONENT – WHEEL & TIRE

Ahmad Fariz bin Fauzi

INTRODUCTION

The line follower robot needs a wheel and tire to move forward, reverse, turn right, and turn left. A wheel is an object that rotates on its axis (Merriam-Webster, 2023).

The wheel is round so that it’s more suitable to be used for any type of mechanism that is closely related to movement (Eurofit, 2021).

A round-shaped wheel allows less drag or resistance and becomes smoother movement (Wheel-Talk, 2019).

For this Line Follower Robot, the wheel that you use is a 50 mm outer diameter made of synthetic material based on **FIGURE 15: SYNTHETIC WHEEL**.

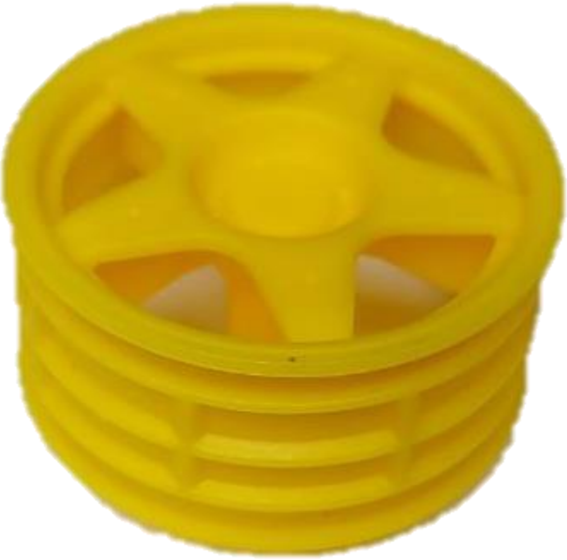


FIGURE 15: SYNTHETIC WHEEL

Even though a wheel is enough to move an object, without tires, movement is quite limited. A tire is made from rubber or rubber compounds based on its purpose of use.

A tire is mounted outside the wheel to make the wheel more efficient for carrying, transmitting, and guiding the movement of an object (Micheline, 2023).

For this line-follower robot, the tire you use is 50mm inner diameter and 65mm outer diameter made of rubber material based on **FIGURE 16: RUBBER TIRE**. The wheel is made from high-quality rubber and gives more grip to the track because of its pattern. The arrow pattern on the tire shows the forward direction.

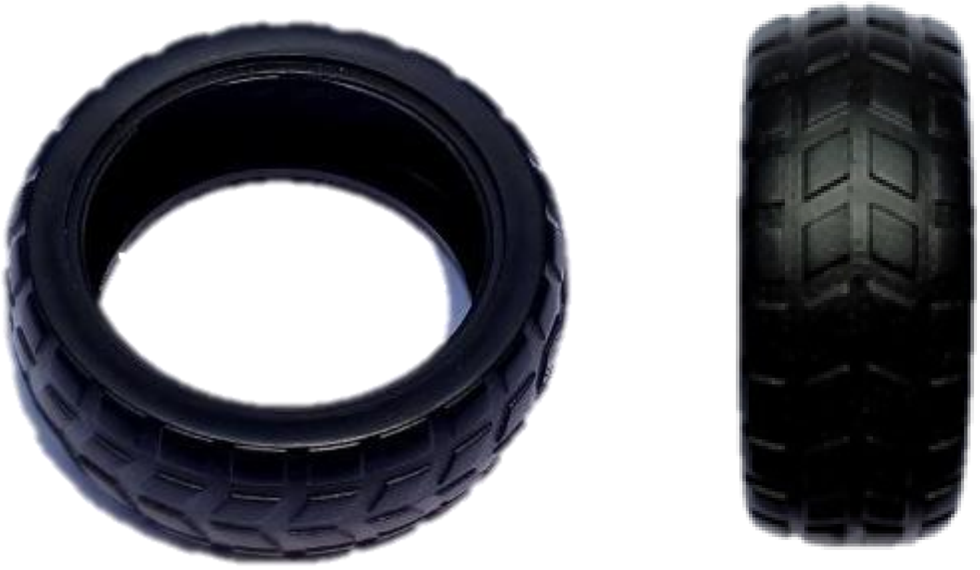


FIGURE 16: RUBBER TIRE

Both tire and wheel for this Line Follower Robot, when you combine them, you can call them by any kind of name, such as ***Smart Car Robot Plastic Tire Wheel***, ***BO Wheel***, ***TT Wheel Robot Tire***, ***Rubber Wheel Robot Tire,*** and ***Robot Car Wheel Plastic Tire***. Just search for it on the internet, and you will be directed to the specific website about the wheel.

The wheel and tire are divided into three sections. The first one is the front section shown in **FIGURE 17: FRONT VIEW**. This section is the most clearly visible when the wheel is attached to the robot.

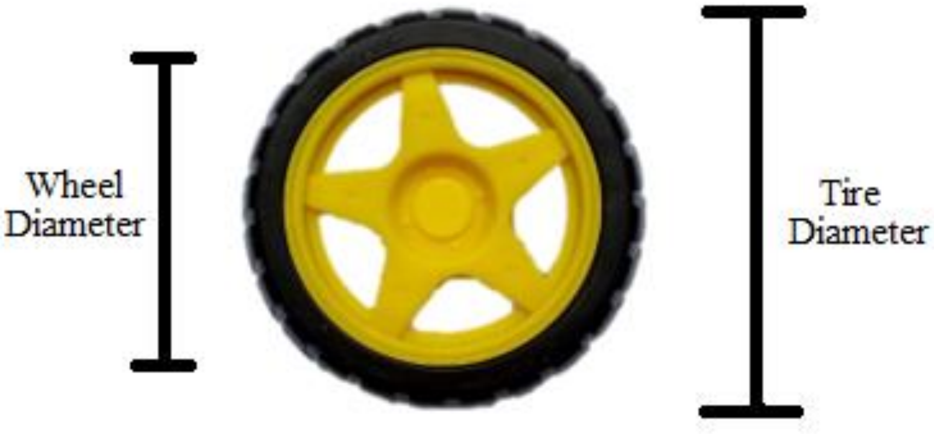


FIGURE 17: FRONT VIEW

The second section is shown in **FIGURE 18: BACK VIEW**. This section has a small hole with two semicircle sides. The function of the hole is to attach the wheel to the DC motor shaft. For this kind of wheel, the semicircle hole has its function as a lock between the shaft and the wheel. Without this semicircle hole, the wheel has less lock if the robot is heavier than a predetermined weight.

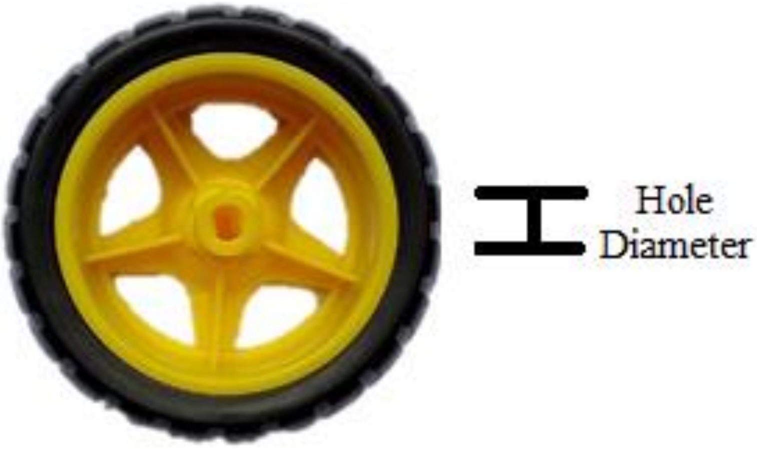


FIGURE 18: BACK VIEW

The third section of the wheel and tire is shown in **FIGURE 19: SIDE VIEW**. The wider the tire width, the stronger the grip on the track. However, it has disadvantages because it is heavier, and the size of the robot becomes wider.

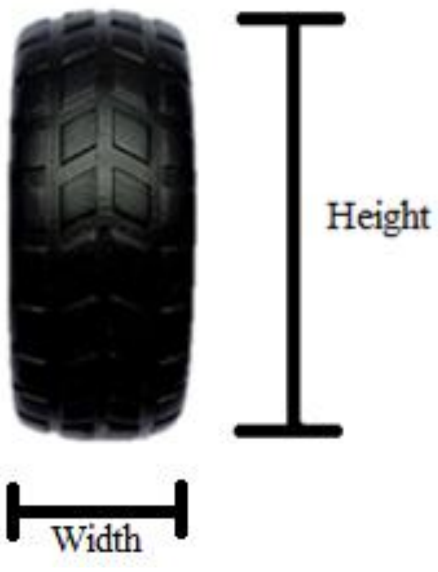


FIGURE 19: SIDE VIEW

FEATURES

Based on the on-field item for this Line Follower Robot, the features for the wheel and the tire are shown in **TABLE 8: WHEEL AND TIRE FEATURE**.

TABLE 8: WHEEL AND TIRE FEATURE

No	Details	Specification
1	Wheel (Yellow, Plastic)	
	v) Weight	34g
	vi) Diameter	50 mm
	vii) Width	25 mm
	viii) Hole Diameter	5 mm
	iv) Loading Capability (Max)	2.5kg
2	Tire (Black, Rubber)	
	Weight	34g
	Inner Diameter = Wheel diameter	50 mm
	Outer Diameter = Height	65 mm
	Width	25 mm

COMPONENT - DC GEARED MOTOR

Ahmad Fariz bin Fauzi

INTRODUCTION

A DC-geared motor is the one that rotates the wheel upon the signal, either clockwise (CW) or counterclockwise (CCW). For this line-follower robot, you use this basic TT 200RPM without an encoder.

Based on **FIGURE 20: DC-GEARED MOTOR**, there are combinations of small gears to reduce the maximum speed to approximately speed.

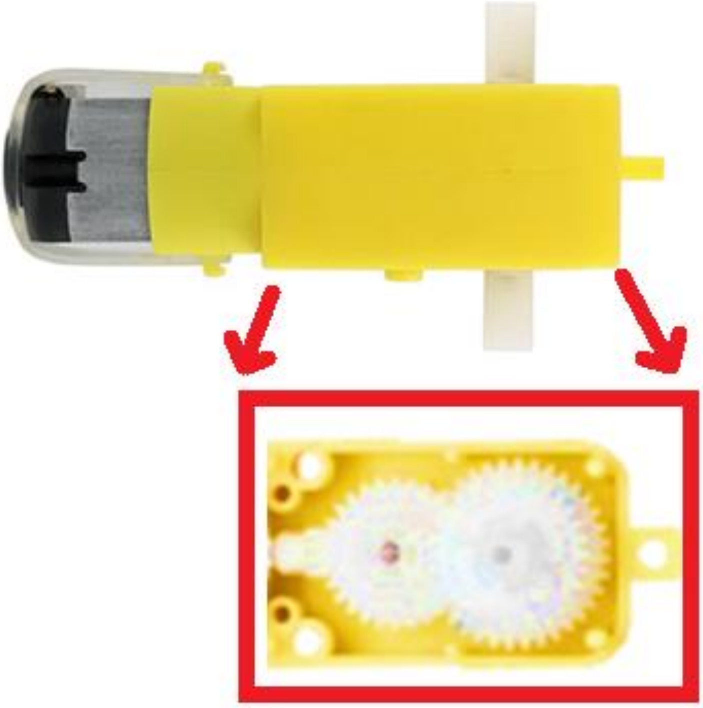


FIGURE 20: DC-GEARED MOTOR

At the bottom of the motor, there are two terminals connected to the coil inside the body. To detect which terminal is terminal 1, there is a red dot showing that terminal. The other without a red dot is terminal 2, based on **FIGURE 21: TERMINAL 1 AND 2**.

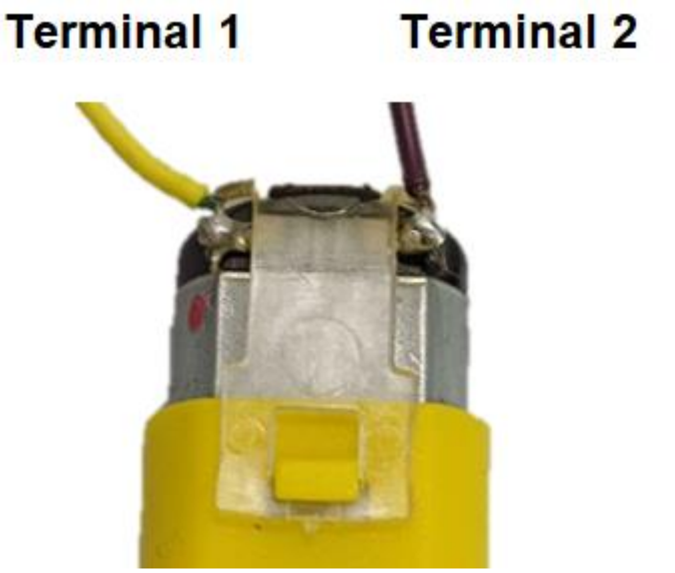


FIGURE 21: TERMINAL 1 AND 2

Terminal 1 means the starting point for the coil, and terminal 2 means the ending point for the coil. By connecting a power supply based on its specification, either 3 Volt, 4.5 Volt, or 6 Volt, the motor will rotate either CW or CCW.

For example, if you connect positive to terminal 1 and negative to terminal 2, the motor will turn CW, as shown in **FIGURE 22: CW**.

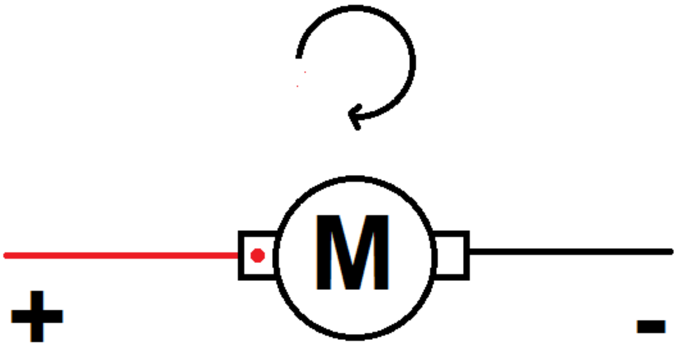


FIGURE 22: CW

So, to turn the motor rotation CCW, just connect terminal 1 to the negative and terminal 2 to the positive, as shown in **FIGURE 23: CCW**.

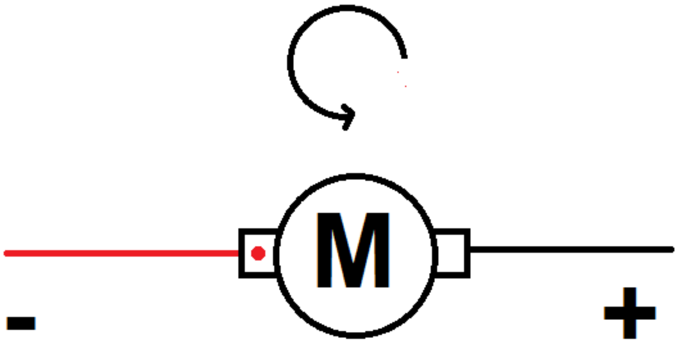


FIGURE 23: CCW

But, to make sure that your robot moves forward or reverse, it depends on how you connect the motor to the board. Though you connect it right to the motor, if you are wrongly connected to the board, your robot will move differently.

The way to assemble the wheel for the DC motor is by looking at the marking point. For the TT DC-geared motor, the marking point is visible on one side of the motor, based on **FIGURE 24: MARKING SPOT**.

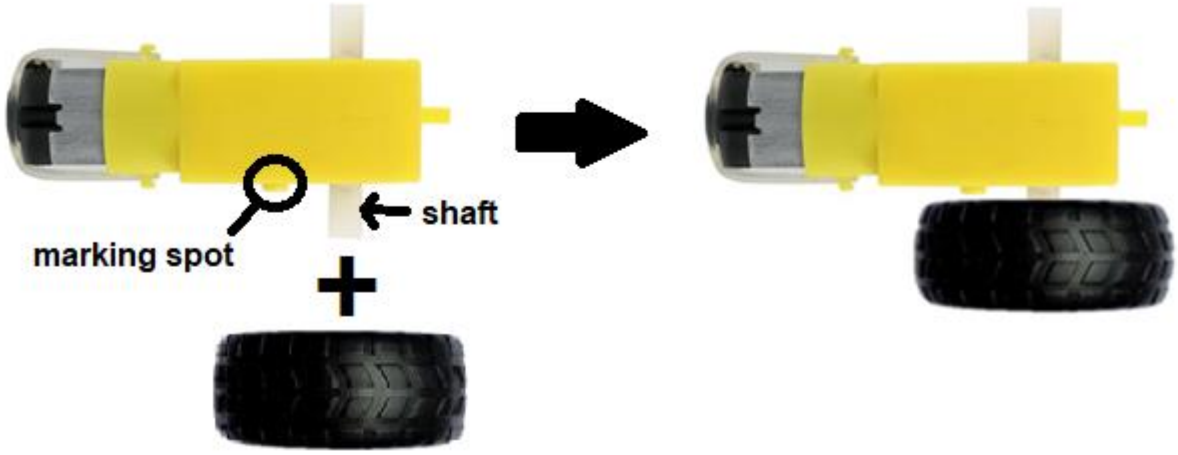


FIGURE 24: MARKING SPOT

The function of this marking spot is to ensure the correct wheel attachment on the motor shaft. Another function is to attach the DC-geared motor through a hole in the robot's chassis or body.

Now you know the correct position to attach the wheel on the DC-geared motor. Let's focus on how to assemble two DC-geared motors with wheels on the robot chassis. Both motors must be asymmetrical to each other, based on **FIGURE 25: ASYMMETRICAL MOTOR POSITION**.

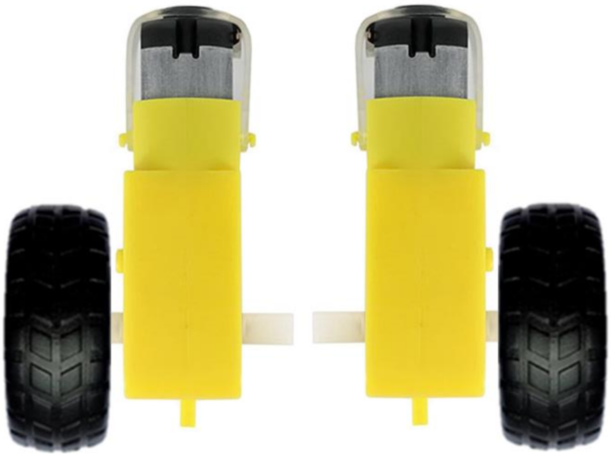


FIGURE 25: ASYMMETRICAL MOTOR POSITION

Make sure both the DC geared motor marking spots are located opposite to each other. Otherwise, you will face a problem during the assembly process. The line-follower robot base can be bought at any DIY online shop, or you can build it on your own. If you have the guts and love to do everything, then you can buy acrylic and start making the base or chassis.

The correct way to assemble the DC-geared motor to the base is shown in **FIGURE 26: DC-GEARED MOTOR TO CHASSIS (TYPE A)**.

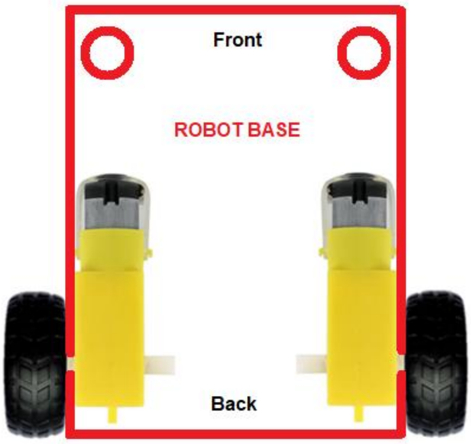


FIGURE 26: DC-GEARED MOTOR TO CHASSIS (TYPE A)

Sometimes you need both tires located at the front of the robot's body, so the best assembly for this purpose is shown in **FIGURE 27: DC GEARED MOTOR TO CHASSIS (TYPE B)**.

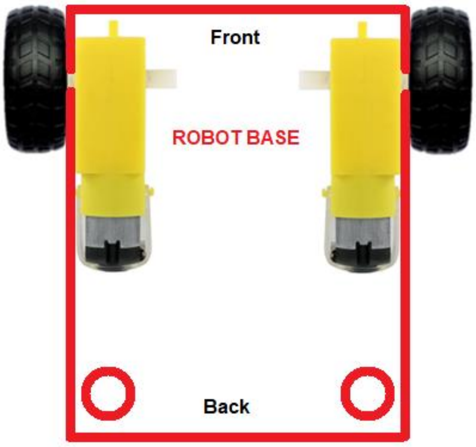


FIGURE 27: DC-GEARED MOTOR TO CHASSIS (TYPE B)

If you want to attach four DC-geared motors to a chassis, the best attachment is shown in **FIGURE 28: DC-GEARED MOTOR TO CHASSIS (4 MOTORS).**

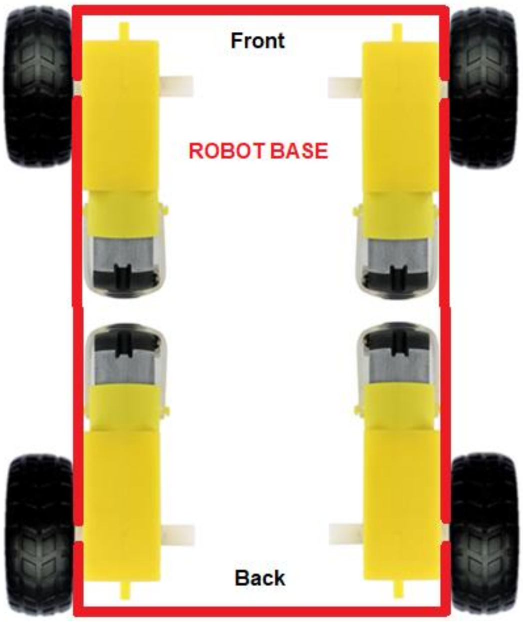


FIGURE 28: DC-GEARED MOTOR TO CHASSIS (4 MOTORS)

The position of the DC Geared Motor attachment depends on the size of your project, the base, and the robot design. There are no strict rules on this matter. All the previous diagrams are about to give an idea of how to attach the DC-geared motor to the body.

FEATURES

The features for the TT DC-geared motor are based on **TABLE 9: DC-GEARED MOTOR**.

TABLE 9: DC-GEARED MOTOR

No	Details	Specification
	Voltage (operated)	3-12V
	Without load current (A)	0.2mA
	Rotation Per Minute (for 6V)	200

COMMISSIONING

Ahmad Fariz bin Fauzi

HOW TO SET THE ROBOT

Based on **FIGURE 29: SETUP THE ROBOT**, attach the two DC-geared motors at the right and left of the chassis. Use a small bolt and nut to tighten all the motors and IR sensors.

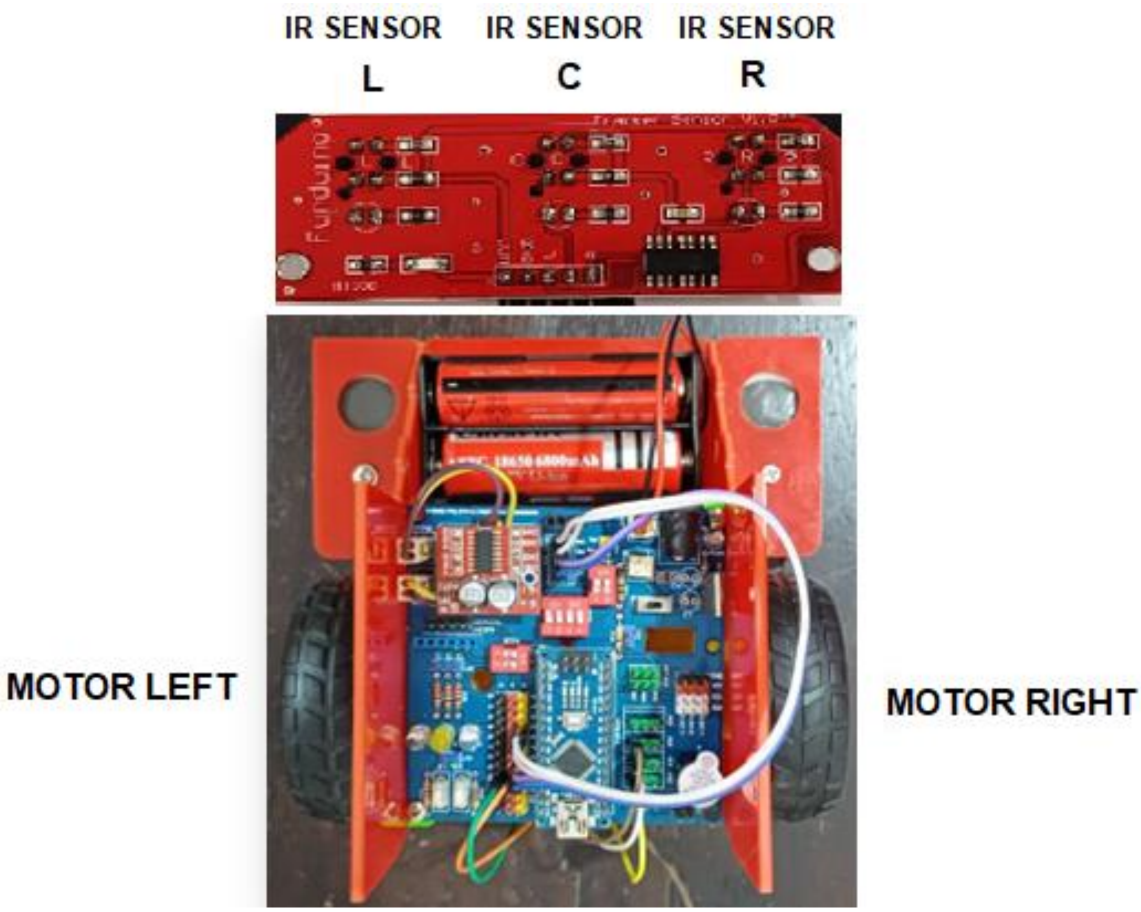


FIGURE 29: SETUP THE ROBOT

Also, attach the IR sensor to the front of the chassis and place it under the chassis. Make sure there is enough space between the IR sensor and the floor or surface based on **FIGURE 30: IR SENSOR LOCATION**.

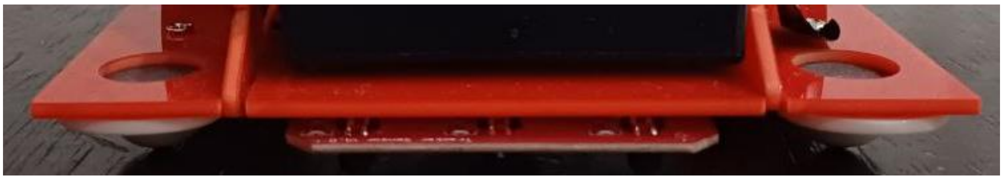


FIGURE 30: IR SENSOR LOCATION

To make the robot less dragged, attach any type of ball caster wheel beside the IR sensor. It is shown in **FIGURE 31: CASTER WHEEL LOCATION**.

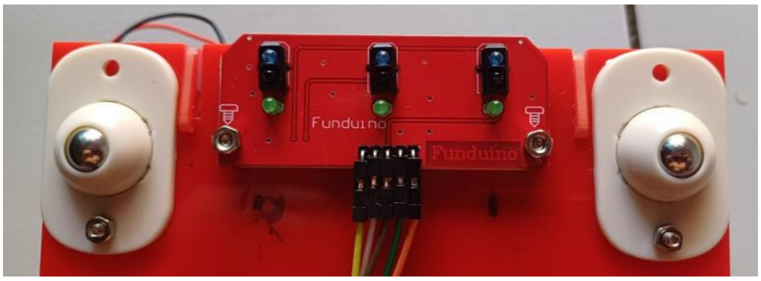


FIGURE 31: CASTER WHEEL LOCATION

HOW TO WIRE THE ROBOT

If you already have a shield board like this robot, just connect the wire from each component to another component based on its pin. To start doing the wiring, the best way is to design the connection based on **TABLE 10: DESIGN FORM**.

TABLE 10: DESIGNED FORM

MOTOR	Motor Terminal	MX1508 pin	Arduino pin
LEFT **MTRA**	1	IN1	D10
	2	IN2	D9
RIGHT **MTRB**	1	IN3	D6
	2	IN4	D5
IR SENSOR	Motor Terminal	TCRT5000 pin	Arduino pin
L Sensor	-	L	A0 or D13
C Sensor	-	C	A1 or D12
R Sensor	-	R	A2 or D11

ARDUINO IDE

Ahmad Fariz bin Fauzi

INTRODUCTION

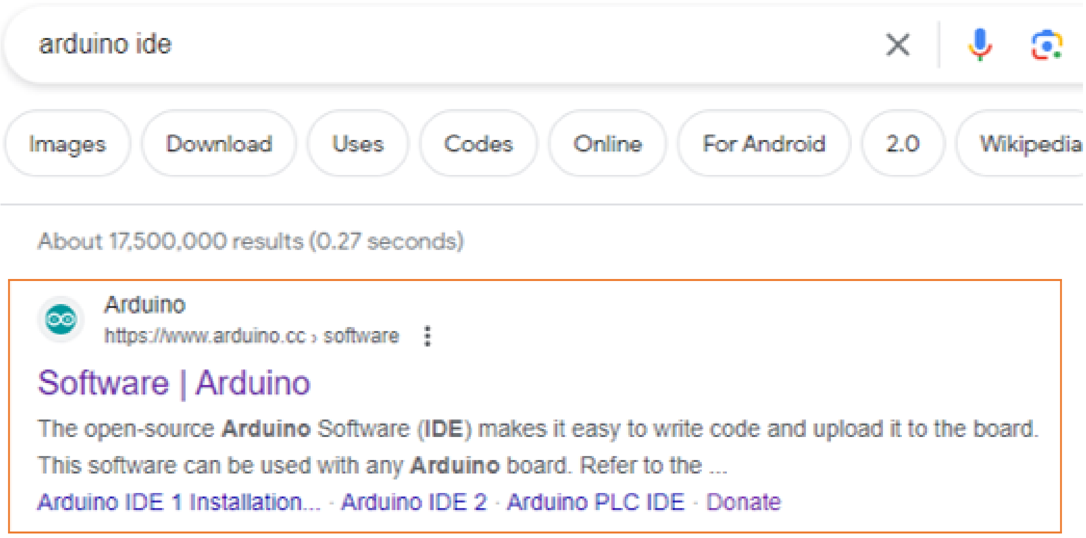
Before executing any microcontroller board, you need a suitable Integrated Development Environment (IDE) for the board. For the Arduino board, you can use the latest open-source Arduino IDE on the website.

While writing for this e-book, the latest version of Arduino IDE is 2.1.1. There is no need to worry if you already installed the older version because Arduino IDE is updatable, and you will be asked to update to a new version. Frankly speaking, the new version is the best one.

HOW TO INSTALL THE ARDUINO IDE

First, turn on your computer and make sure there is an internet connection during the process.

Now, click to open any kind of browser. Then type **Arduino IDE**. The browser will show the result based on **FIGURE 32: SEARCHING FOR ARDUINO IDE**.



arduino ide

About 17,500,000 results (0.27 seconds)

Arduino
https://www.arduino.cc › software

Software | Arduino

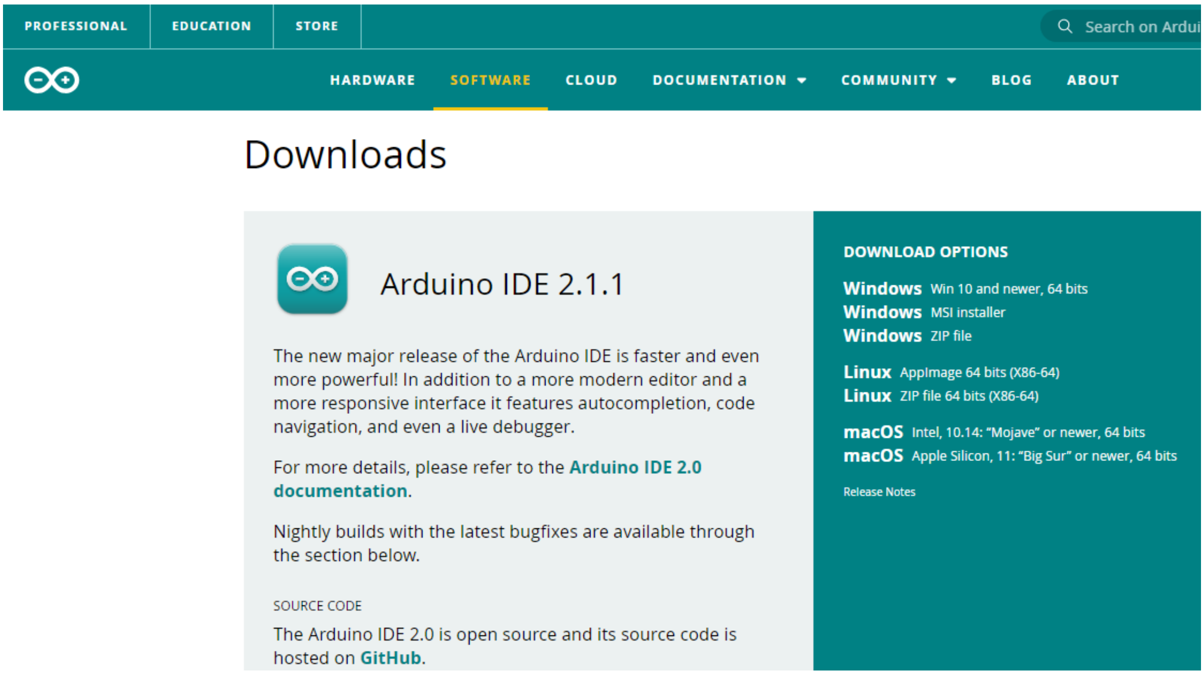
The open-source **Arduino** Software (**IDE**) makes it easy to write code and upload it to the board. This software can be used with any **Arduino** board. Refer to the ...

Arduino IDE 1 Installation... · Arduino IDE 2 · Arduino PLC IDE · Donate

FIGURE 32: SEARCHING FOR ARDUINO IDE

In a second, the browser will send you the result based on what you have typed. Don't worry if the results show millions. Just pick the one that shows **Software | Arduino**. Usually, the top result is the best option to choose. Then, left-click on the **Software | Arduino**.

Now you are on the new Arduino IDE interface shown in **FIGURE 33: ARDUINO IDE DOWNLOADS**.



Downloads

Arduino IDE 2.1.1

The new major release of the Arduino IDE is faster and even more powerful! In addition to a more modern editor and a more responsive interface it features autocompletion, code navigation, and even a live debugger.

For more details, please refer to the **Arduino IDE 2.0 documentation**.

Nightly builds with the latest bugfixes are available through the section below.

SOURCE CODE

The Arduino IDE 2.0 is open source and its source code is hosted on **GitHub**.

DOWNLOAD OPTIONS

Windows Win 10 and newer, 64 bits
Windows MSI installer
Windows ZIP file

Linux AppImage 64 bits (X86-64)
Linux ZIP file 64 bits (X86-64)

macOS Intel, 10.14: "Mojave" or newer, 64 bits
macOS Apple Silicon, 11: "Big Sur" or newer, 64 bits

Release Notes

FIGURE 33: ARDUINO IDE DOWNLOADS

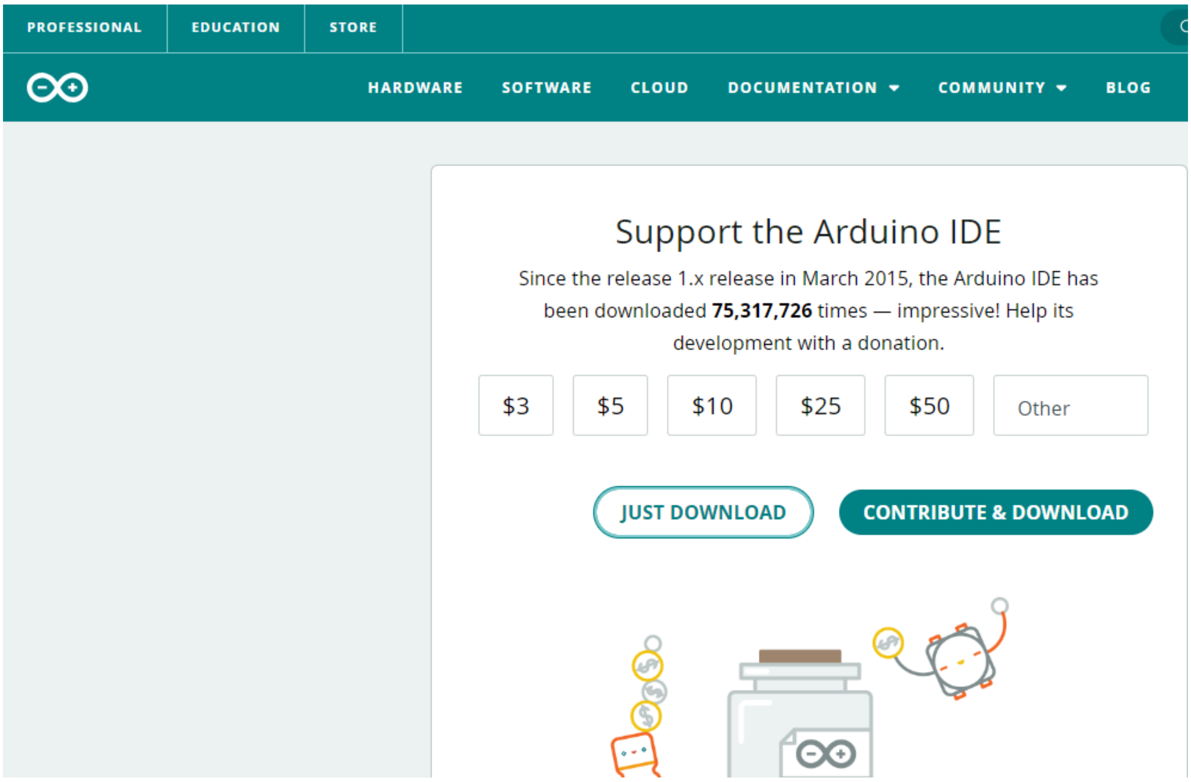
Thanks to all the contributors who support the research and development, the new release of the Arduino IDE version is much better than the older one and more powerful than before. A lot of bugs are fixed. So, it's a good choice to update to the new release IDE version when the new version is available.

To know more about the details before installing or upgrading to the new version, just refer to the **Arduino IDE 2.0 documentation**.

The download options depend on your computer software, either Linux, MacOS, or Windows. It's up to you, but still, your computer needs to meet the minimum requirement for the Arduino IDE.

Just click on your mouse to either one of the **DOWNLOAD OPTIONS** lists, and you will be sent to a new interface.

By the way, you could be one of the contributors that help the development team to upgrade their system by supporting the team by donating a small amount of money based on **FIGURE 34: SUPPORT THE ARDUINO IDE**.



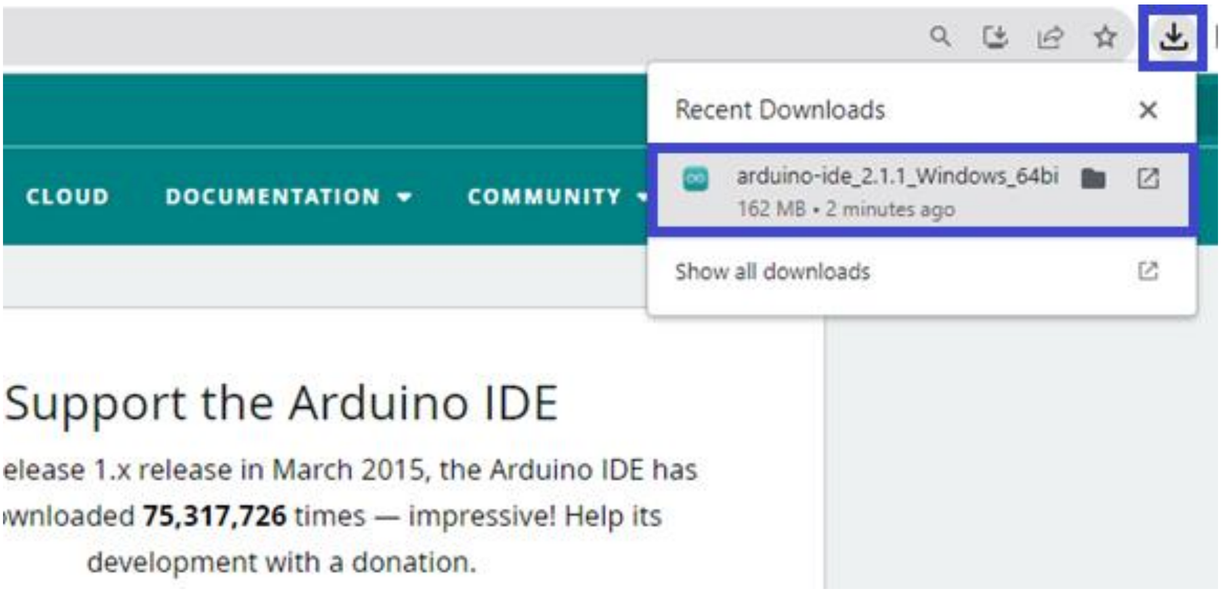
Support the Arduino IDE

Since the release 1.x release in March 2015, the Arduino IDE has been downloaded **75,317,726** times — impressive! Help its development with a donation.

FIGURE 34: SUPPORT THE ARDUINO IDE

Come on guys, the minimum donation is just $3. By supporting it, you are helping the development of the Arduino.

After clicking or CONTRIBUTE & DOWNLOAD, the download process will begin shortly based on **FIGURE 35: RECENT DOWNLOAD**.

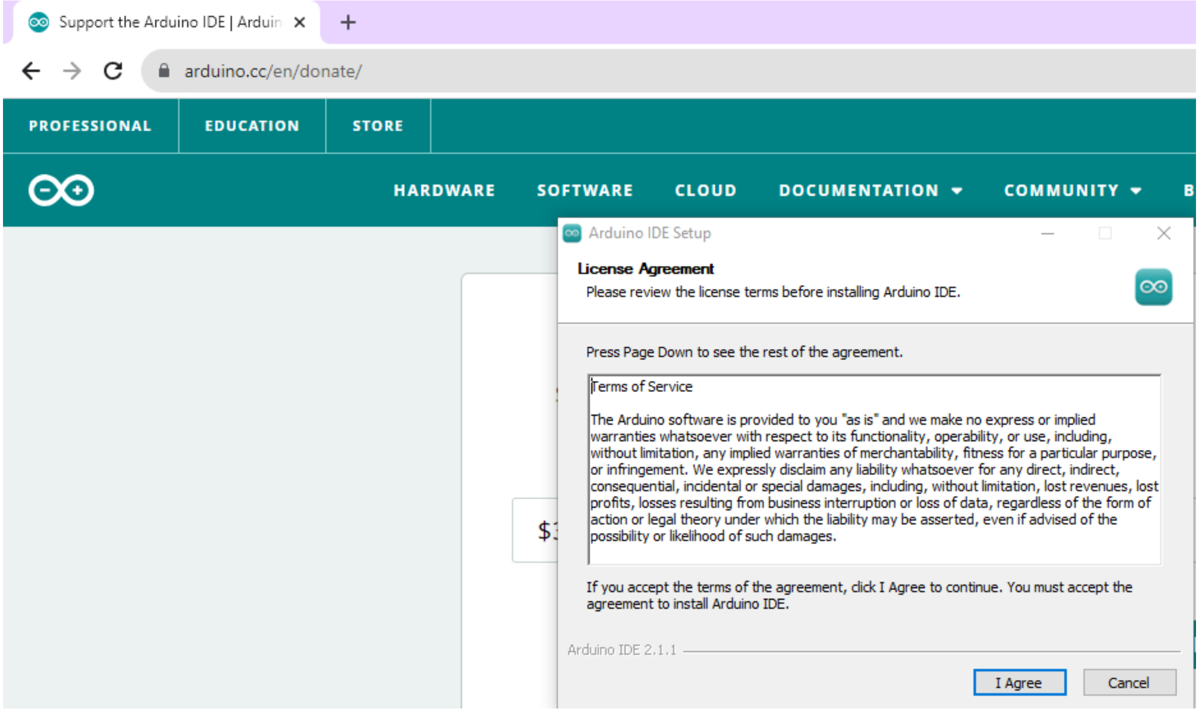


Support the Arduino IDE

elease 1.x release in March 2015, the Arduino IDE has wnloaded **75,317,726** times — impressive! Help its development with a donation.

FIGURE 35: RECENT DOWNLOAD

Then click on the icon 📁 to start installing the software. You will be asked to review the license terms before starting to install the Arduino IDE. If you accept all the terms inside the agreement, then left-click on I Agree. If not, just click Cancel and no need to develop a robot using the Arduino IDE. The step is based on **FIGURE 36: ARDUINO LICENSE AGREEMENT**.



License Agreement

Please review the license terms before installing Arduino IDE.

Press Page Down to see the rest of the agreement.

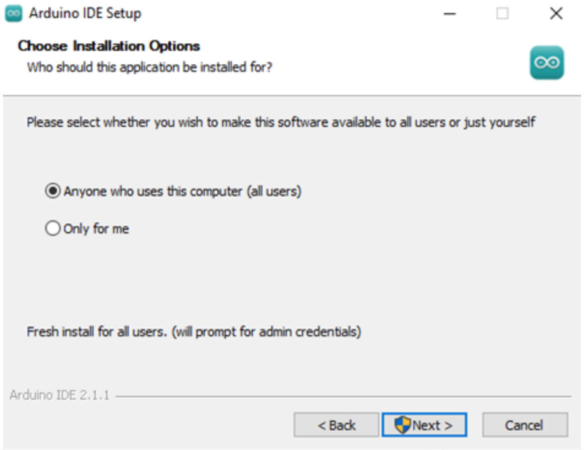
Terms of Service

The Arduino software is provided to you "as is" and we make no express or implied warranties whatsoever with respect to its functionality, operability, or use, including, without limitation, any implied warranties of merchantability, fitness for a particular purpose, or infringement. We expressly disclaim any liability whatsoever for any direct, indirect, consequential, incidental or special damages, including, without limitation, lost revenues, lost profits, losses resulting from business interruption or loss of data, regardless of the form of action or legal theory under which the liability may be asserted, even if advised of the possibility or likelihood of such damages.

If you accept the terms of the agreement, click I Agree to continue. You must accept the agreement to install Arduino IDE.

FIGURE 36: ARDUINO LICENSE AGREEMENT

After agreeing with the License Agreement, you will be asked to choose the installation option. All users who use the computer can access the Arduino IDE or only you can access the Arduino IDE. Everything depends on you. If you like more privacy, then choose ● Only for me. If not, just choose ● Anyone who uses this computer (all users). After choosing the installation Options, left-click on " **FIGURE 37: CHOOSE INSTALLATION OPTIONS**.



Choose Installation Options

Who should this application be installed for?

Please select whether you wish to make this software available to all users or just yourself

● Anyone who uses this computer (all users)

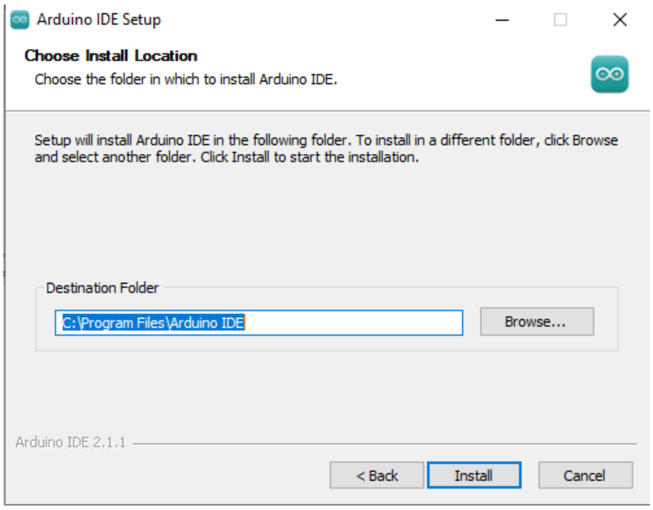
○ Only for me

Fresh install for all users. (will prompt for admin credentials)

Arduino IDE 2.1.1

FIGURE 37: CHOOSE INSTALLATION OPTIONS

After choosing the installation option, Arduino IDE Setup asks where the desirable destination for your Arduino IDE folder is based on **FIGURE 38: CHOOSE INSTALL LOCATION**. The default folder location is in C drive, but the location can be arranged for any drive inside your computer. To select a new destination folder, just left-click on Browse..., it and set your location. After that, left-click on Install to proceed to the destination folder inside the computer. Once installed, you can't change the location unless you uninstall the software and repeat the installation setup.



Choose Install Location

Choose the folder in which to install Arduino IDE.

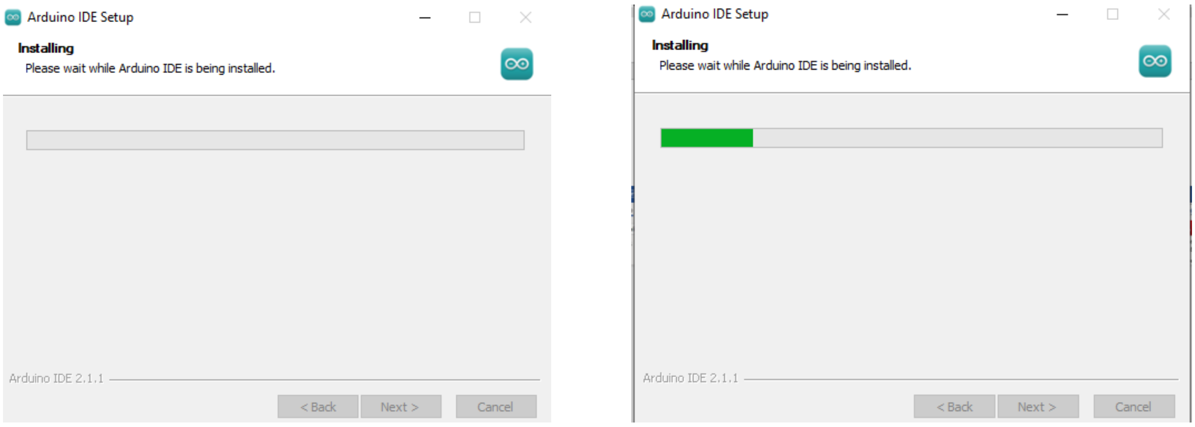
Setup will install Arduino IDE in the following folder. To install in a different folder, click Browse and select another folder. Click Install to start the installation.

Destination Folder: C:\Program Files\Arduino IDE

Arduino IDE 2.1.1

FIGURE 38: CHOOSE INSTALL LOCATION

Now, the Arduino IDE installation process will start. Just let it install automatically until the process is done. It will take a while until the installation is done. During the installation, the horizontal bar shows the green colour increasing from nothing to a full bar based on **FIGURE 39: INSTALLING THE ARDUINO IDE**. When the bar is full of green colour, it means that the installation process is done.



Arduino IDE Setup

Installing

Please wait while Arduino IDE is being installed.

Arduino IDE 2.1.1

Arduino IDE Setup

Installing

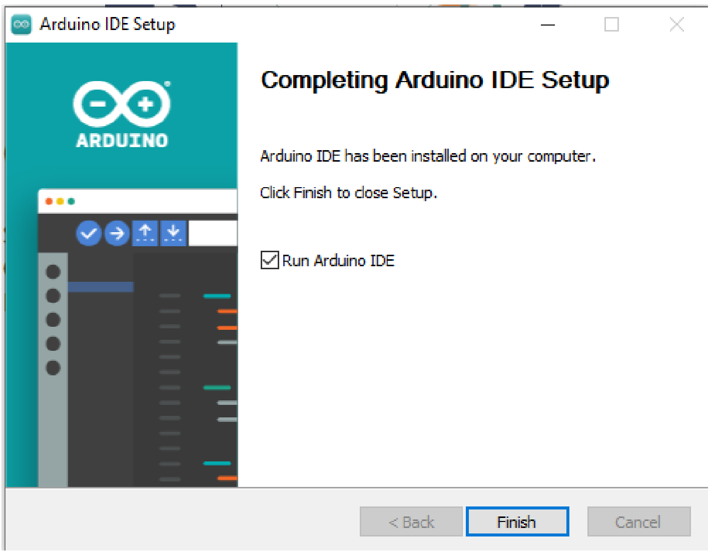
Please wait while Arduino IDE is being installed.

Arduino IDE 2.1.1

start installation during the installation

FIGURE 39: INSTALLING THE ARDUINO IDE

After completing the installation, you will be asked either to Run Arduino IDE immediately or just click Finish to close the process based on **FIGURE 40: COMPLETING ARDUINO IDE SETUP**.



Arduino IDE Setup

Completing Arduino IDE Setup

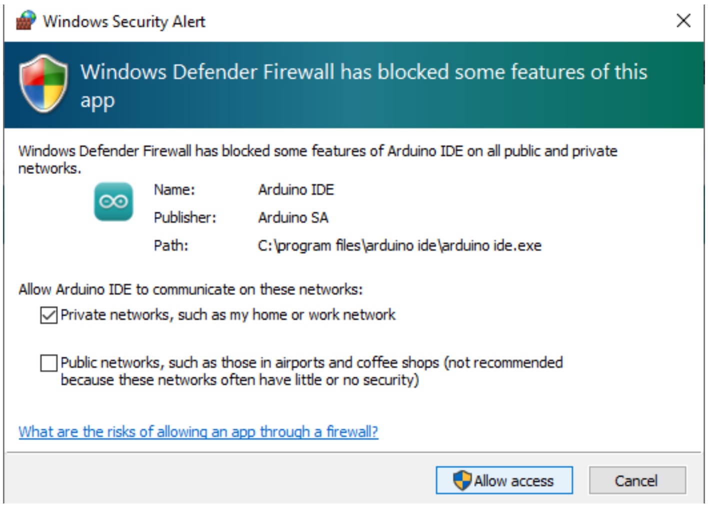
Arduino IDE has been installed on your computer.

Click Finish to close Setup.

☑ Run Arduino IDE

FIGURE 40: COMPETING ARDUINO IDE SETUP

It's a good choice if you just tick the box and click Finish to start the Arduino IDE because you want to know if there is another instruction to follow before you can start doing your programming code using Arduino IDE. Some computers have already activated their firewalls to block some features on any outsourced software based on **FIGURE 41: WINDOWS SECURITY ALERT**.



Windows Security Alert

Windows Defender Firewall has blocked some features of this app

Windows Defender Firewall has blocked some features of Arduino IDE on all public and private networks.

Name: Arduino IDE
Publisher: Arduino SA
Path: C:\program files\arduino ide\arduino ide.exe

Allow Arduino IDE to communicate on these networks:

☑ Private networks, such as my home or work network

□ Public networks, such as those in airports and coffee shops (not recommended because these networks often have little or no security)

What are the risks of allowing an app through a firewall?

FIGURE 41: WINDOWS SECURITY ALERT

So, before you click Allow access, make sure you choose the right choice for your security. For security purposes, click the **Private Network** and click Allow access to give permission, and allows the Arduino IDE to communicate on specific networks.

CODING – BASIC STRUCTURE

Ahmad Fariz bin Fauzi

To learn more about Arduino programming, you need to know more about the basic structure of the Arduino code program itself. The Arduino code program is divided into three main sections, shown in **TABLE 11: ARDUINO MAIN SECTION**

TABLE 11: ARDUINO MAIN SECTION

SECTION	EXPLANATION	CODE PROGRAM
Structure	Two main functions: i) Setup () ii) Loop ()	void **setup ()** {statement 1; statement 2;} void **loop ()** {statement 1; statement 2;}
Value	Variable and constants: i) local variable is described inside a function. ii) global variable is described outside a function.	void setup () {statement1; statement2;} void loop () {int a = 2;} **<-- local variable** int a = 2; **<-- global variable** void setup () {statement1; statement2;} void loop () {statement 1; statement 2;}
Function	i) INPUT All the sensors are INPUT	For Digital Pins: pinMode(pinterminal,INPUT) For Analog Pins: -

	ii) OUTPUT
	All the actuators are OUTPUT
		For digital output:
		digitalWrite(pinterminal,HIGH)
		For analog output:
		analogWrite(pinterminal,ivalue);

TUTORIAL: CODING – LINE FOLLOWER ROBOT

Ahmad Fariz bin Fauzi & Ts. Hj Mohd Nordin bin Mohd Jani & Saifful Bahari bin Omar

BASIC CODING WITHOUT PID & TCRT5000 READ AS DIGITAL SENSOR-Type1

Type 1 is the basic code for the simplest line follower robot using driver motor MX1508 and Arduino NANO with a 3 array IR sensor TCRT5000 without a PID control. All the motor pins are set up as digital output, and the IR sensor pins are set up as digital sensors.

The steps for programming the code into the Arduino NANO board on the line follower robot are very simple:

- The first step is to make sure the robot is connected to the computer's USB port via a USB cable.
- The second step is to open the Arduino IDE on your computer.
- The third step is to do the coding.

Set all the global variables outside the function based on the Arduino Nano PWM pins and MX1508 pin connection. This setup makes your coding easier to understand if something happens to the system.

For this purpose, you use PWM PIN 5 and PIN 6 for the right motor. Meanwhile, PIN 9 and PIN 10 are for the left motor, as shown in **TABLE 12: ARDUINO NANO PINS TO MX1508 PINS (TYPE-1)**.

TABLE 12: ARDUINO NANO PINS TO MX1508 PINS (TYPE-1)

DC Geared Motor	Arduino NANO PWM Pins	MX1508 Pins
Left Motor (Motor-A)	D10	IN1 (M1)
	D9	IN2 (M2)
Right Motor (Motor-B)	D6	IN3 (M3)
	D5	IN4 (M4)

The global variable code is as below:

```
int M1 = 10;                    // MX1508 PIN IN1 to ARDUINO NANO D10 (PWM)
int M2 = 9;                     // MX1508 PIN IN2 to ARDUINO NANO D9 (PWM)
int M3 = 6;                     // MX1508 PIN IN3 to ARDUINO NANO D6 (PWM)
int M4 = 5;                     // MX1508 PIN IN4 to ARDUINO NANO D5 (PWM)
```

Next, do the same thing for the global variable setup based on Arduino Nano Analog pins and a 3-array IR Sensor TCRT5000 pin connection. This setup is suitable for continuous sensor reading and is easy to understand if anything malfunctions with the system.

You can use either digital pins, other than motor pins, or ADC pins. For this line follower robot, you just use digital pins like D13, D12, and D11 shown in **TABLE 13: ARDUINO NANO PINS TO TCRT5000 PINS**.

TABLE 13: ARDUINO NANO PINS TO TCRT5000 PINS

IR Sensor	Arduino NANO Digital Pins	TCRT5000 Pins
Right	D13	IN1 or R
Centre	D12	IN2 or C
Left	D11	IN3 or L

```
int R1 = D13;                   // TCRT5000 PIN IN1 to ARDUINO NANO D13
int C0 = D12;                   // TCRT5000 PIN IN2 to ARDUINO NANO D12
int L1 = D11;                   // TCRT5000 PIN IN3 to ARDUINO NANO D11
```

Now, let’s set up the pin type based on the global variable and Mode type, either as INPUT or OUTPUT based on (1).

void setup () {statement1; statement2; ......;}   (1)

```
void setup ()
{
  Serial.begin(9600);         // setup to 9600 baud for Serial Monitor purpose.
  pinMode(R1, INPUT);        // pin R1 & Mode INPUT
  pinMode(C0, INPUT);        // pin C0 & Mode INPUT
  pinMode(L1, INPUT);        // pin L1 & Mode INPUT
  pinMode(M1, OUTPUT);  // pin M1 & Mode OUTPUT
  pinMode(M2, OUTPUT);  // pin M2 & Mode OUTPUT
  pinMode(M3, OUTPUT);  // pin M3 & Mode OUTPUT
  pinMode(M4, OUTPUT);  // pin M4 & Mode OUTPUT
 }
```

Then, let’s set up the continuous loop program based on (2).

void loop () {statement1; statement2; ......;}   (2)

```
void loop ()
{                                              // open close loop statement
  int L1Value = digitalRead(L1);    // declare the L1Value based on digitalRead(L1)
  Serial.println(L1Value);              // Allow Serial print to show L1Value
  int C0Value = digitalRead(C0);   // declare the C0Value based on digitalRead(C0)
  Serial.println(C0Value);             // Allow Serial print to show C0Value
  int R1Value = digitalRead(R1);   // declare the R1Value based on digitalRead(R1)
  Serial.println(R1Value);             // Allow Serial print to show R1Value

  if (L1Value == 1 && C0Value == 0 && R1Value == 1)          // if statement
  {
    MotorForward();                                                               // motor forward task
    Serial.println("Move Forward");                                          // Serial shows TEXT
  }
else if (L1Value == 0 && C0Value == 0 && R1Value == 0) // else if statement
 {
  MotorStop();                                          // motor stop task
  Serial.println("Stop All Black");                     // Serial shows TEXT
 }
 else if (L1Value == 1 && C0Value == 1 && R1Value == 1)   // else if statement
 {
  MotorStop();                                          // motor stop task
  Serial.println("Stop All White");                     // Serial shows TEXT
 }
 else if (L1Value == 0 && C0Value == 0 && R1Value == 1)   // else if statement
 {
  MotorTurnLeft();                                      // motor turn left task
  Serial.println("Move Left (L1 and C0 on BLACK)");     // Serial shows TEXT
 }
 else if (L1Value == 1 && C0Value == 0 &R1Value == 0)     // else if statement
 {
  MotorTurnRight();                                     // motor turn right task
  Serial.println("Move Right (C0 and R1 on BLACK)");    // Serial shows TEXT
 }
 else if (L1Value == 0 && C0Value == 1 && R1Value == 1)   // else if statement
 {
  MotorTurnLeftStatic();                                // motor turnleftst task
  Serial.println("Move Left(Only L1 on BLACK)");        // Serial shows TEXT
 }
  else if (L1Value == 1 && C0Value == 1 && R1Value == 0) // else if statement
 {
  MotorTurnRightStatic();                               // motor turnrightst task
  Serial.println("Move Right(Only R1 on BLACK)");       // Serial shows TEXT
 }
}                                    // close void loop statement
```

```
void MotorForward()                //MotorForward Task
{
  digitalWrite(M1,HIGH );          //set the M1 as digital & Write the Signal as HIGH
  digitalWrite(M2, LOW);           //set the M2 as digital & Write the Signal as LOW
  digitalWrite(M3, HIGH);          //set the M3 as digital & Write the Signal as HIGH
  digitalWrite(M4, LOW);           //set the M4 as digital & Write the Signal as LOW
}

void MotorTurnLeft()               // Motor TurnLeft Task
{
  digitalWrite(M1, LOW);           //set the M1 as digital & Write the Signal as LOW
  digitalWrite(M2, LOW);           //set the M2 as digital & Write the Signal as LOW
  digitalWrite(M3, HIGH);          //set the M3 as digital & Write the Signal as HIGH
  digitalWrite(M4, LOW);           //set the M4 as digital & Write the Signal as LOW
}

void MotorTurnLeftStatic()  // Motor TurnLeftSt Task
{
  digitalWrite(M1, LOW);           //set the M1 as digital & Write the Signal as LOW
  digitalWrite(M2, HIGH);          //set the M2 as digital & Write the Signal as HIGH
  digitalWrite(M3, HIGH);          //set the M3 as digital & Write the Signal as HIGH
  digitalWrite(M4, LOW);           // set the M4 as digital & Write the Signal as LOW
}

void MotorTurnRight()              // Motor TurnRight Task
{
  digitalWrite(M1, HIGH);          //set the M1 as digital & Write the Signal as HIGH
  digitalWrite(M2, LOW);           //set the M2 as digital & Write the Signal as LOW
  digitalWrite(M3, LOW);           //set the M3 as digital & Write the Signal as LOW
  digitalWrite(M4, LOW);           //set the M4 as digital & Write the Signal as LOW
}
```

```
void MotorTurnRightStatic()          // Motor TurnRightSt Task
{
  digitalWrite(M1, HIGH);            //set the M1 as digital & Write the Signal as HIGH
  digitalWrite(M2, LOW);             //set the M2 as digital & Write the Signal as LOW
  digitalWrite(M3, LOW);             //set the M3 as digital & Write the Signal as LOW
  digitalWrite(M4, HIGH);            //set the M4 as digital & Write the Signal as HIGH
}

void MotorStop()                     // Motor TurnStop Task
{
  digitalWrite(M1, LOW);             // set the M1 as digital & Write the Signal as LOW
  digitalWrite(M2, LOW);             // set the M2 as digital & Write the Signal as LOW
  digitalWrite(M3, LOW);             // set the M3 as digital & Write the Signal as LOW
  digitalWrite(M4, LOW);             // set the M4 as digital & Write the Signal as LOW
}
```

BASIC CODING WITHOUT PID & TCRT5000 READ AS DIGITAL SENSOR-Type 2

This Type 2 coding is the modification code based on Type 1. There are only small differences if compared to Type 1. All motor pins are set as digital pins and all IR sensor pins are set as digital sensors but connected to analog pins.

The steps for programming the code into the Arduino NANO board on the line follower robot are still the same as the previous Type 1. The PWM pin is still the same as the previous Type 1. No need to adjust the wiring connection if all the terminations are connected to PINs 5 and 6 for the Right Motor and PINs 9 and 10 for the Left Motor shown as **TABLE 14: ARDUINO NANO PINS TO MX1508 PINS (TYPE-2)**.

TABLE 14: ARDUINO NANO PINS TO MX1508 PINS (TYPE-2)

DC Geared Motor	Arduino NANO PWM Pins	MX1508 Pins
Left Motor (Motor-A)	D10	IN1
	D9	IN2
Right Motor (Motor-B)	D6	IN3
	D5	IN4

The global variable code is as below:

```
int M1 = 10;                // MX1508 PIN IN1 to ARDUINO NANO D10 (PWM)
int M2 = 9;                 // MX1508 PIN IN2 to ARDUINO NANO D9 (PWM)
int M3 = 6;                 // MX1508 PIN IN3 to ARDUINO NANO D6 (PWM)
int M4 = 5;                 // MX1508 PIN IN4 to ARDUINO NANO D5 (PWM)
```

For this type 2, you must use three ADC pins numbers A0, A1, and A2 to TCRT5000 pins shown as **TABLE 15: ARDUINO NANO PINS TO TCRT5000 PINS**. The reason for using this ADC pin is the ability to set the sensor as digital or analog.

TABLE 15: ARDUINO NANO PINS TO TCRT5000 PINS

IR Sensor	Arduino NANO ADC Pins	TCRT5000 Pins
Right	A0	IN1
Centre	A1	IN2
Left	A2	IN3

```
int R1 = A0;               // TCRT5000 PIN IN1 to ARDUINO NANO A0
int C0 = A1;               // TCRT5000 PIN IN2 to ARDUINO NANO A1
int L1 = A2;               // TCRT5000 PIN IN3 to ARDUINO NANO A2

void setup ()
{
  Serial.begin(9600);      // setup to 9600 baud for Serial Monitor purpose.
  pinMode(R1, INPUT);      // pin R1 & Mode INPUT
  pinMode(C0, INPUT);      // pin C0 & Mode INPUT
  pinMode(L1, INPUT);      // pin L1 & Mode INPUT
  pinMode(M1, OUTPUT);     // pin M1 & Mode OUTPUT
  pinMode(M2, OUTPUT);     // pin M2 & Mode OUTPUT
  pinMode(M3, OUTPUT);     // pin M3 & Mode OUTPUT
  pinMode(M4, OUTPUT);     // pin M4 & Mode OUTPUT
 }
void loop ()
{                                    // open void loop statement
  int L1Value = digitalRead(L1);     // declare the L1Value based on digitalRead(L1)
  Serial.println(L1Value);           // Allow Serial print to show L1Value
  int C0Value = digitalRead(C0);     // declare the C0Value based on digitalRead(C0)
  Serial.println(C0Value);           // Allow Serial print to show C0Value
  int R1Value = digitalRead(R1);     // declare the R1Value based on digitalRead(R1)
  Serial.println(R1Value);           // Allow Serial print to show R1Value
```

```
if (L1Value == 1 && C0Value == 0 && R1Value == 1)          // if statement
{
  MotorForward();                                           // motor forward task
  Serial.println("Move Forward");                           // Serial shows TEXT
}
else if (L1Value == 0 && C0Value == 0 && R1Value == 0) // else if statement
{
  MotorStop();                                              // motor stop task
  Serial.println("Stop All Black");                         // Serial shows TEXT
}
else if (L1Value == 1 && C0Value == 1 && R1Value == 1)     // else if statement
{
  MotorStop();                                              // motor stop task
  Serial.println("Stop All White");                         // Serial shows TEXT
}
else if (L1Value == 0 && C0Value == 0 && R1Value == 1)     // else if statement
{
  MotorTurnLeft();                                          // motor turn left task
  Serial.println("Move Left (L1 and C0 on BLACK)");         // Serial shows TEXT
}

else if (L1Value == 1 && C0Value == 0 &R1Value == 0)       // else if statement
{
  MotorTurnRight();                                         // motor turn right task
  Serial.println("Move Right (C0 and R1 on BLACK)");        // Serial shows TEXT
}

else if (L1Value == 0 && C0Value == 1 && R1Value == 1)     // else if statement
{
  MotorTurnLeftStatic();                                    // motor turnleftst task
  Serial.println("Move Left(Only L1 on BLACK)");            // Serial shows TEXT
}
```

```
   else if (L1Value == 1 && C0Value == 1 && R1Value == 0) // else if statement
  {
   MotorTurnRightStatic();                               // motor turnrightst task
   Serial.println("Move Right(Only R1 on BLACK)");       // Serial shows TEXT
  }
}                                // close void loop statement
```

The difference between this Type-2 programming is in this section. You are changing the motor pin from digital to analog because you want to control the speed. The value of PWM is between 0 to 255. In this case, you put the value to 80 for one motor speed and 100 for both motor speeds.

```
void MotorForward()              // MotorForward Task with PWM equal to 80
{
 analogWrite(M1, 80);            // set the M1 as analog and Write the PWM to 80
 digitalWrite(M2, LOW);          // set the M2 as digital and Write the Signal is LOW
 analogWrite(M3, 80);            // set the M3 as analog and Write the PWM to 80
 digitalWrite(M4, LOW);          // set the M4 as digital and Write the Signal is LOW
}

void MotorTurnLeft()             // Motor TurnLeft Task with PWM equal to 80
{
 digitalWrite(M1, LOW);          // set the M1 as digital and Write the Signal is LOW
 digitalWrite(M2, LOW);          // set the M2 as digital and Write the Signal is LOW
 analogWrite(M3, 80);            // set the M3 as analog and Write the PWM to 80
 digitalWrite(M4, LOW);          // set the M4 as digital and Write the Signal is LOW
}
 void MotorTurnLeftStatic()      // Motor TurnLeftSt Task with PWM equal to 100
{
 digitalWrite(M1, LOW);          // set the M1 as digital and Write the Signal is LOW
 analogWrite(M2, 100);           // set the M2 as analog and Write the PWM to 100
 analogWrite(M3, 100);           // set the M3 as analog and Write the PWM to 100
```

```
  digitalWrite(M4, LOW);             // set the M4 as digital and Write the Signal is LOW
}
void MotorTurnRight()                // Motor TurnRight Task with PWM equal to 80
{
  analogWrite(M1, 80);               // set the M1 as analog and Write the PWM to 80
  digitalWrite(M2, LOW);             // set the M2 as digital and Write the Signal is LOW
  digitalWrite(M3, LOW);             // set the M3 as digital and Write the Signal is LOW
  digitalWrite(M4, LOW);             // set the M4 as digital and Write the Signal is LOW
}
void MotorTurnRightStatic()          // Motor TurnRightSt Task with PWM equal to 100
{
  analogWrite(M1, 100);              // set the M1 as analog and Write the PWM to 100
  digitalWrite(M2, LOW);             // set the M2 as digital and Write the Signal is LOW
  digitalWrite(M3, LOW);             // set the M3 as digital and Write the Signal is LOW
  analogWrite(M4, 100);              // set the M4 as analog and Write the PWM to 100
}
void MotorStop()                     // Motor TurnStop Task
{
  digitalWrite(M1, LOW);             // set the M1 as digital and Write the Signal is LOW
  digitalWrite(M2, LOW);             // set the M2 as digital and Write the Signal is LOW
  digitalWrite(M3, LOW);             // set the M3 as digital and Write the Signal is LOW
  digitalWrite(M4, LOW);             // set the M4 as digital and Write the Signal is LOW
}
```

BASIC CODING WITHOUT PID & TCRT5000 READ AS ANALOG SENSOR-Type 3

This Type 3 coding is the modification code based on the Type 1 and Type 2. There are only small differences if compared to the previous coding. All Motor pins are set as digital pins and all IR sensor pins are set as digital pins but during the loop main structure, all the sensors will be coded as analog sensors.

Make sure to set all the global variable setup outside the function based on Arduino Nano PWM Pins and MX1508 Pins connection. This setup makes your coding easier to understand if something happens to the system. The PWM pin is still the same as the previous Type 1 and Type 2. No need to adjust the wiring connection if all the termination is connected to PIN 5 and 6 for the Right Motor and PIN 9 and 10 for Left Motor shown as **TABLE 16: ARDUINO NANO PINS TO MX1508 PINS (TYPE-3)**.

TABLE 16: ARDUINO NANO PINS TO MX1508 PINS (TYPE-3)

DC Geared Motor	Arduino NANO PWM Pins	MX1508 Pins
Left Motor (Motor-A)	D10	IN1
	D9	IN2
Right Motor (Motor-B)	D6	IN3
	D5	IN4

The global variable code is as below:

```
int M1 = 10;          // MX1508 PIN IN1 to ARDUINO NANO D10 (PWM)
int M2 = 9;           // MX1508 PIN IN2 to ARDUINO NANO D9 (PWM)
int M3 = 6;           // MX1508 PIN IN3 to ARDUINO NANO D6 (PWM)
int M4 = 5;           // MX1508 PIN IN4 to ARDUINO NANO D5 (PWM)
```

For this Type 3, you still use ADC PIN A0, A1, and A2 shown as Figure: Arduino NANO Pins to TCRT5000 Pins shown as **TABLE 17: ARDUINO NANO PINS TO TCRT5000 PINS**. The reason for using this ADC pin is the ability to change the sensor from digital to analog.

Table 17: Arduino NANO Pins to TCRT5000 Pins

IR Sensor	Arduino NANO ADC Pins	TCRT5000 Pins
Right	A0	IN1
Centre	A1	IN2
Left	A2	IN3

```
int R1 = A0;                // TCRT5000 PIN IN1 to ARDUINO NANO A0
int C0 = A1;                // TCRT5000 PIN IN2 to ARDUINO NANO A1
int L1 = A2;                // TCRT5000 PIN IN3 to ARDUINO NANO A2
```

The additional new global variable to store the value from the IR sensor.

```
int R1Value = 0;
int C0Value = 0;
int L1Value = 0;
int threshold = 400;
```

All the IR sensor pins are no longer declared as an input in void setup () because they now become an analog input. But what if all the pins are still declared as a digital input in void setup ()? What will happen to your programming? The answer is the function is still the same if you don’t declare the pin as INPUT but this coding uses more memory.

```
void setup ()
{
  Serial. begin(9600);        // setup to 9600 baud for Serial Monitor purposes.
  pinMode(M1, OUTPUT);  // pin M1 & Mode OUTPUT
  pinMode(M2, OUTPUT);  // pin M2 & Mode OUTPUT
  pinMode(M3, OUTPUT);  // pin M3 & Mode OUTPUT
  pinMode(M4, OUTPUT);  // pin M4 & Mode OUTPUT
 }
```

Inside the void loop function, just change all the digitalRead to analogRead.

```
void loop ()
```

```
{                                    // open void loop statement
  L1Value = analogRead(L1);          // declare the L1Value based on analogRead(L1)
  Serial.println(L1Value);           // Allow Serial print to show L1Value
  C0Value = analogRead(C0);          // declare the C0Value based on analogRead(C0)
  Serial.println(C0Value);           // Allow Serial print to show C0Value
  R1Value = analogRead(R1);          // declare the R1Value based on analogRead(R1)
  Serial.println(R1Value);           // Allow Serial print to show R1Value
  if (L1Value > threshold && C0Value <threshold && R1Value >threshold)   // if statement
  {
    MotorForward();                                        // motor forward task
    Serial.println("Move Forward");                        // Serial shows TEXT
  }
 else if (L1Value <threshold && C0Value <threshold && R1Value <threshold) // else if statement
  {
    MotorStop();                                           // motor stop task
    Serial.println("Stop All Black");                      // Serial shows TEXT
  }
  else if (L1Value >threshold && C0Value >threshold && R1Value >threshold)      // else if statement
  {
    MotorStop();                                           // motor stop task
    Serial.println("Stop All White");                      // Serial shows TEXT
  }
  else if (L1Value <threshold && C0Value<threshold && R1Value >threshold)       // else if statement
  {
    MotorTurnLeft();                                       // motor turn left task
    Serial.println("Move Left (L1 and C0 on BLACK)");      // Serial shows TEXT
  }
  else if (L1Value >threshold && C0Value <threshold &&R1Value <threshold)      //
else if statement
  {
    MotorTurnRight();                                        // motor turn right task
    Serial.println("Move Right (C0 and R1 on BLACK)");       // Serial shows TEXT
  }
  else if (L1Value <threshold && C0Value >threshold && R1Value >threshold)     //
else if statement
  {
    MotorTurnLeftStatic();                                   // motor turnleftst task
    Serial.println("Move Left(Only L1 on BLACK)");           // Serial shows TEXT
  }
    else if (L1Value >threshold && C0Value >threshold && R1Value <threshold)   //
else if statement
  {
    MotorTurnRightStatic();                                  // motor turnrightst task
    Serial.println("Move Right(Only R1 on BLACK)");          // Serial shows TEXT
  }
}                              // close void loop statement
void MotorForward()            // MotorForward Task with PWM equal to 80
{
  analogWrite(M1, 80);          // set the M1 as analog and Write the PWM to 80
  digitalWrite(M2, LOW);        // set the M2 as digital and Write the Signal is LOW
  analogWrite(M3, 80);          // set the M3 as analog and Write the PWM to 80
  digitalWrite(M4, LOW);        // set the M4 as digital and Write the Signal is LOW
}
void MotorTurnLeft()           // Motor TurnLeft Task with PWM equal to 80
{
  digitalWrite(M1, LOW);        // set the M1 as digital and Write the Signal is LOW
  digitalWrite(M2, LOW);        // set the M2 as digital and Write the Signal is LOW
  analogWrite(M3, 80);          // set the M3 as analog and Write the PWM to 80
  digitalWrite(M4, LOW);        // set the M4 as digital and Write the Signal is LOW
```

```
}
 void MotorTurnLeftStatic()         // Motor TurnLeftSt Task with PWM equal to 100
{
  digitalWrite(M1, LOW);            // set the M1 as digital and Write the Signal is LOW
  analogWrite(M2, 100);             // set the M2 as analog and Write the PWM to 100
  analogWrite(M3, 100);             // set the M3 as analog and Write the PWM to 100
  digitalWrite(M4, LOW);            // set the M4 as digital and Write the Signal is LOW
}
void MotorTurnRight()               // Motor TurnRight Task with PWM equal to 80
{
  analogWrite(M1, 80);              // set the M1 as analog and Write the PWM to 80
  digitalWrite(M2, LOW);            // set the M2 as digital and Write the Signal is LOW
  digitalWrite(M3, LOW);            // set the M3 as digital and Write the Signal is LOW
  digitalWrite(M4, LOW);            // set the M4 as digital and Write the Signal is LOW
}
void MotorTurnRightStatic()         // Motor TurnRightSt Task with PWM equal to 100
{
  analogWrite(M1, 100);             // set the M1 as analog and Write the PWM to 100
  digitalWrite(M2, LOW);            // set the M2 as digital and Write the Signal is LOW
  digitalWrite(M3, LOW);            // set the M3 as digital and Write the Signal is LOW
  analogWrite(M4, 100);             // set the M4 as analog and Write the PWM to 100
}
void MotorStop()                    // Motor TurnStop Task
{
  digitalWrite(M1, LOW);            // set the M1 as digital and Write the Signal is LOW
  digitalWrite(M2, LOW);            // set the M2 as digital and Write the Signal is LOW
  digitalWrite(M3, LOW);            // set the M3 as digital and Write the Signal is LOW
  digitalWrite(M4, LOW);            // set the M4 as digital and Write the Signal is LOW
}
```

QUESTION

QUESTION 1:

Show the code to move a motor forward if all the motor pin is set to digital.

ANSWER

```
void loop ()
{
digitalWrite(M1,HIGH );         //set the M1 as digital & Write the Signal as HIGH
digitalWrite(M2, LOW);          //set the M2 as digital & Write the Signal as LOW
digitalWrite(M3, HIGH);         //set the M3 as digital & Write the Signal as HIGH
digitalWrite(M4, LOW);          //set the M4 as digital & Write the Signal as LOW
}
```

QUESTION 2:

Show the code to move a motor turn left if all the motor pin is set to digital.

ANSWER

```
void loop ()
{
digitalWrite(M1, LOW);          //set the M1 as digital & Write the Signal as LOW
digitalWrite(M2, LOW);          //set the M2 as digital & Write the Signal as LOW
digitalWrite(M3, HIGH);         //set the M3 as digital & Write the Signal as HIGH
digitalWrite(M4, LOW);          //set the M4 as digital & Write the Signal as LOW
}
```

QUESTION 3:

Show the code to stop a motor if the L1, C0 and R1 sensors are below the threshold.

ANSWER

```
void loop ()
{
  L1Value = analogRead(L1);       // declare the L1Value based on analogRead(L1)
  C0Value = analogRead(C0);       // declare the C0Value based on analogRead(C0)
  R1Value = analogRead(R1);       // declare the R1Value based on analogRead(R1)

if (L1Value <threshold && C0Value <threshold && R1Value <threshold) // else if statement
  {
  digitalWrite(M1, LOW);          // set the M1 as digital and Write the Signal is LOW
  digitalWrite(M2, LOW);          // set the M2 as digital and Write the Signal is LOW
  digitalWrite(M3, LOW);          // set the M3 as digital and Write the Signal is LOW
  digitalWrite(M4, LOW);          // set the M4 as digital and Write the Signal is LOW
  }
}
```

REFERENCES

AHMAD FARIZ, F. (2023). ANALISIS PENGGUNAAN KUASA(W) IR SENSOR MENGAWAL LED DAN/TANPA PENGAWAL MIKRO UNTUK APLIKASI ROBOTIK. *Icrepe 2023*.

Arduino. (2023). *Doc*. Retrieved from Arduino. cc: https://docs.arduino.cc

Components101. (2021, May 31). *MX1508 DC Motor Driver with PWM Control*. Retrieved from Components 101: https://components101.com/

Eurofit. (2021). *The reason car tires are circles and why it matters*. Retrieved from Eurofit /Home /News: https://www.eurofitgroup.com

GlobalData. (2023, June 6). *Robotics innovation: Leading companies in line follower robots*. Retrieved from Verdict: https://www.verdict.co.uk

javatpoint. (2023). *What is Microcontroller?* Retrieved from java T point: https://www.javatpoint.com

Jimmy Presler, J. M. (2023). *Understanding black and white as colours.* Retrieved from Adobe: https://www.adobe.com

Kerstin. (2023, August 4). *Understanding Arduino Nano – A Comprehensive Guide to Features, Uses, and Comparisons*. Retrieved from IBE Electronics: https://www.pcbaaa.com

M. Pakdaman, M. M. (2010). A line follower robot from design to implementation: Technical issues and problems. *Computer and Automation Engineering (ICCAE), 2010 The 2nd International Conference onVolume: 1* (pp. 5-9). Singapore: 2010 The 2nd International Conference on Computer and Automation Engineering (ICCAE).

Mandal, P. (2023, February 27). *Line Follower Robot*. Retrieved from Educba: https://www.educba.com

Merriam-Webster. (2023). *Wheel*. Retrieved from Merriam-Webster Dictionary: https://www.merriam-webster.com

Micheline. (2023). *Functions of the tire*. Retrieved from Michelin : The tire digest: https://thetiredigest.michelin.com

Murmson, S. (2021). *Why Do We Not List Black and White as Colors in Physics?* Retrieved from Seattle.PI: https://education.seattlepi.com

Murmson, S. (2021). *Why Do We Not List Black and White as Colors in Physics?* Retrieved from Seattle Pi: https://education.seattlepi.com/

Robocraze. (2022, June 20). *IR Sensor Working.* Retrieved from Robocraze: https://robocraze.com/

ROBU.IN. (2021). *how-to-make-a-line-follower-robot-using-arduino-connection-code*. Retrieved from ROBU.IN: https://robu.in

School, E. (2019, Jan 29). *Different types of Microcontroller Programming used in Embedded Systems*. Retrieved from Elysium Embedded School: https://embeddedschool.in

Trolove, H. (2018, April). *Using the MX1508 Brushed DC Motor Driver.* Retrieved from www.techmonkeybusiness.com: https://www.techmonkeybusiness.com

Volchko, J. (2018). *Visible Light Spectrum: From a Lighting Manufacturer's Perspective*. Retrieved from LUMITEX: https://www.lumitex.com/blog/visible-light-spectrum

Wheel-Talk. (2019, January 28). *Why Are Wheels Circular?* Retrieved from Santa Ana Wheel: https://www.santaanawheel.com

LINE FOLLOWER ROBOT

AHMAD FARIZ BIN FAUZI
Ts. Hj. MOHD NORDIN BIN MOHD JANI
SAIFFUL BAHARI BIN OMAR

This **Line Follower Robot** e-book is a reading material that provides the best input to readers in the quest to produce a robot using NANO's Arduino controller, MX1508 driver motor, and IR TCRT5000 3 array sensor. This book contains a breakdown of chapters detailing each part needed in robot production and coding. Interestingly in this book, all the coding produced is the basic code of the Arduino only and no library is used. The main purpose of using basic coding is to facilitate readers' understanding of how easy it is to create Arduino codes specifically to produce robot line followers. It is undeniable that encoding using the PID control system produces more effective robots, but readers must first understand the basics of analog and digital sensor encoding and PWM encoding. This book contains a tutorial that applies all the necessary coding basics without a PID controller.

LINE FOLLOWER ROBOT

e ISBN 978-967-0838-95-3

9 789670 838953

POLITEKNIK MELAKA

(online)