BOT BUILDERS

Basic Robotics Course

Robotics AI Lab India

CHAPTER 01

INTRODUCTION TO ROBOTICS

ROBOTS

WHAT IS A ROBOT?

Robot is a machine - one that can be programmed by a computer. It is capable of carrying out complex series of actions. A robot can be guided by an external control device like humans, or the control can be embedded within, for autonomous functioning.

Most robots are task-performing machines that are designed to solve a specific problem.

ROBOTICS

WHAT IS ROBOTICS?

The branch of technology that deals with the design, construction, operation, and applications of robots, as well as computer systems for their control and information processing is called robotics.

HOW ROBOTS SOLVE OUR DAY TO DAY PROBLEMS?

- Perform monotonous tasks - **Less Fatigue**
- Perform tasks with accuracy - **Less Errors**
- Perform tasks at remote places where humans can’t reach - **Accessibility**
- Perform tasks that can be dangerous for a human to do **- Safety**
- Save time - **Increase Productivity**

ROBOTS

INDUSTRIAL ROBOTS

ROBOT IN A CAR FACTORY

ROBOT IN A PHONE FACTORY

ROBOTS

SERVICE ROBOTS

SELF ORDERING MACHINE

CAR CLEANING BOT

DELIVERY ROBOT

ROBOTS

MEDICAL ROBOTS

AUTONOMOUS ROBOT PERFORMING MEDICAL PROCEDURE

MEDICAL ROBOTS ASSISTING DOCTORS

ROBOTS

EXPLORATION ROBOTS

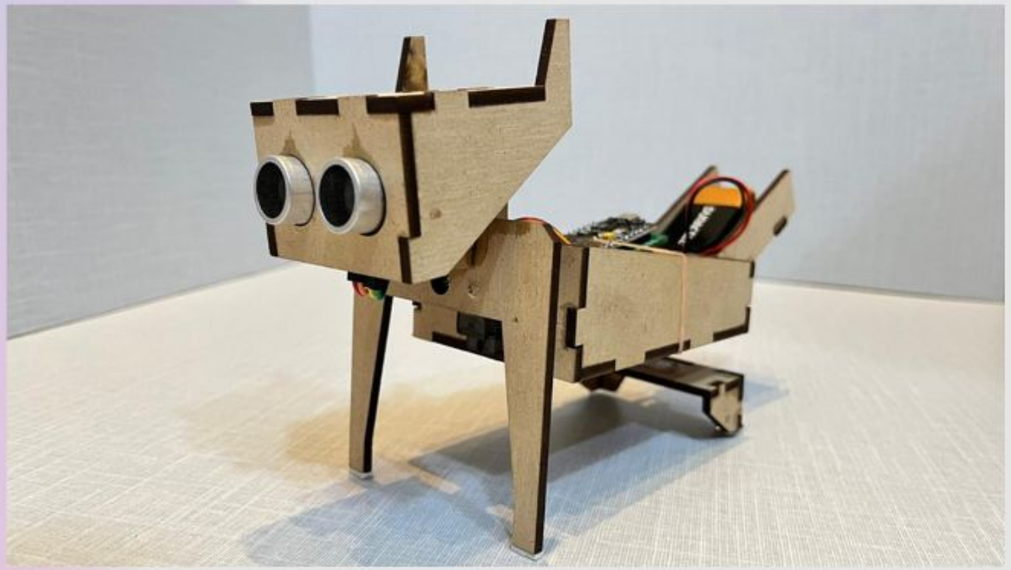
MARS ROVER EXPLORING
SURACE OF MARS

SEA EXPLORING ROBOT

ROBOTS

EDUCATIONAL ROBOTS

SCOUT 4WD



DOGO

LET’S DISCUSS

WHERE ELSE CAN ROBOTS BE USED?

- Agricultural Robots
- Home Assistance
- Entertainment

WHAT DAY-TO-DAY TASKS CAN ROBOTS HELP US WITH?

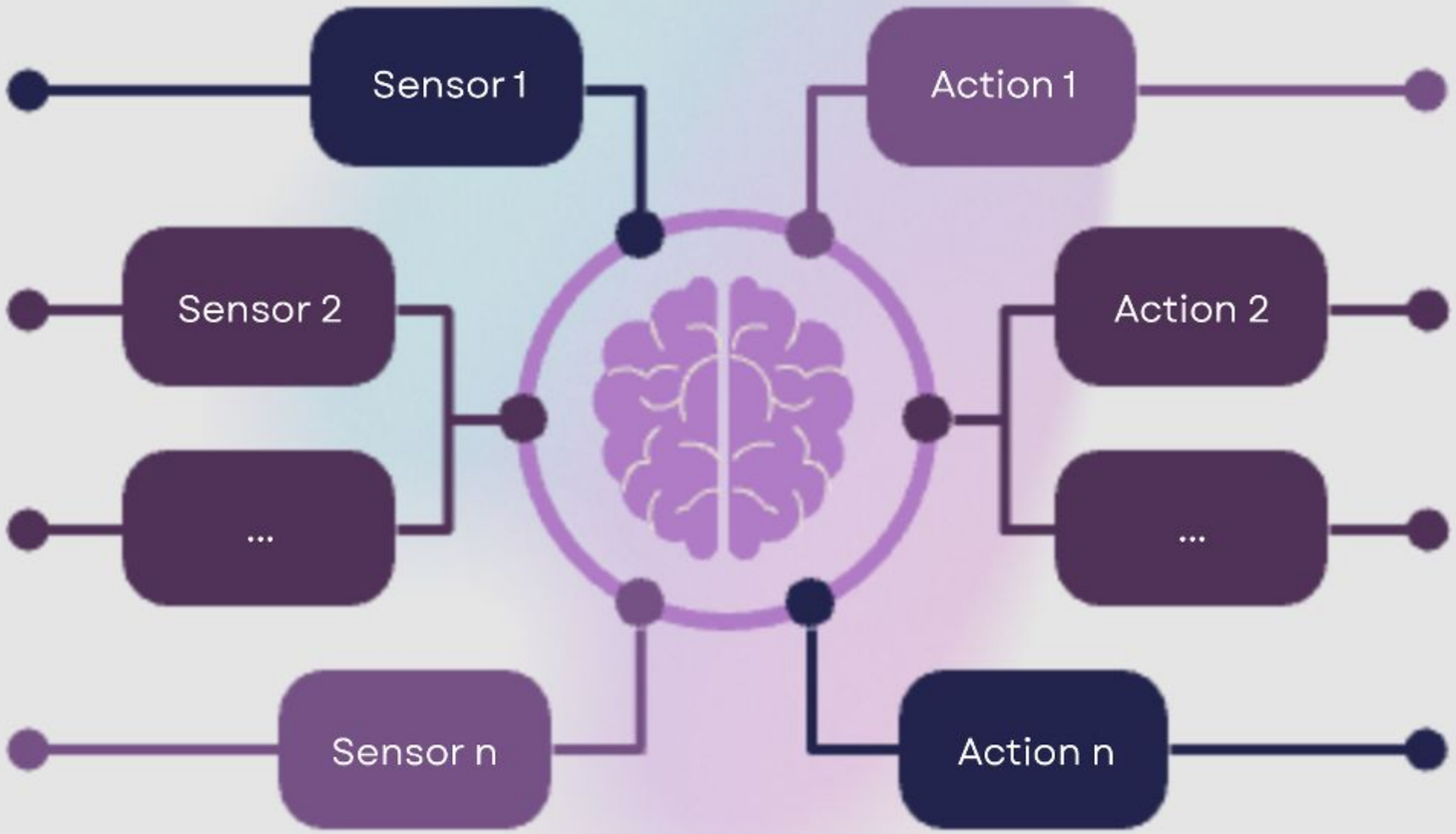
CHAPTER 02

INTRODUCTION TO MICROCONTROLLER

ROBOT OPERATION

HOW DO ROBOTS OPERATE?

- Sense
- Process
- Act



ROBOT OPERATION

LET US UNDERSTAND THIS IN AN INTUITIVE MANNER

Sense
It's **HOT**!!!

Process to make a decision
"Take your hand off"

Act
Hand pulled away 😅

ROBOT OPERATION

Before we learn about sensing, first let’s jump to microcontroller

MICROCONTROLLER

WHAT ARE MICROCONTROLLERS?

Tiny computers that control robots and gadgets, an Integrated Circuit that can be programmed to conduct specific task.

What makes robots smart?

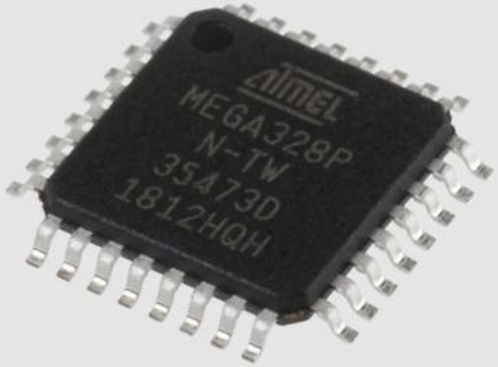
Did you know that robots have tiny brains? These brains are called microcontrollers. They act like computers but are small enough to fit in your palm.

- They have the capability to sense the environment
- They have the capability to process information and take decision
- They have the capability to send instruction

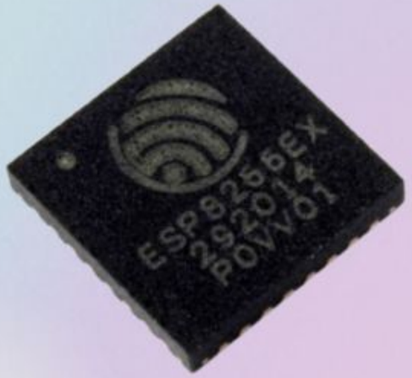
Digital Brain

MICROCONTROLLER

EXAMPLES



ATmega 328P
used on Arduino Uno



ESP8266EX
used on Mercury Board

RP2040
used on Raspberry Pi Pico

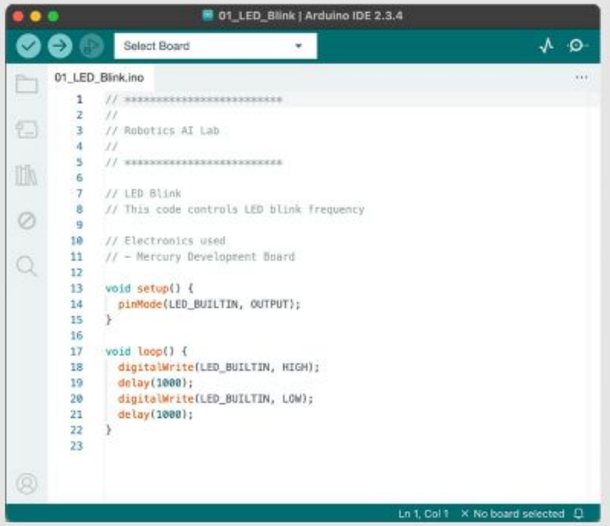
ARDUINO®

WHAT IS ARDUINO?

- Arduino is an open-source electronics platform.
- Based on easy-to-use hardware and software.
- Used for building interactive electronics projects.
- Ideal for beginners, students, and engineers.

Arduino Hardware

ARDUINO™



Arduino Software

ARDUINO®

WHAT IS ARDUINO® PLATFORM IN GENERAL?

Arduino combines easy-to-use hardware and software to interact with the external environment by reading inputs from sensors, such as a light sensor, and controlling outputs, such as running a motor or turning on an LED.

WHAT IS ARDUINO® INTERFACE USED FOR?

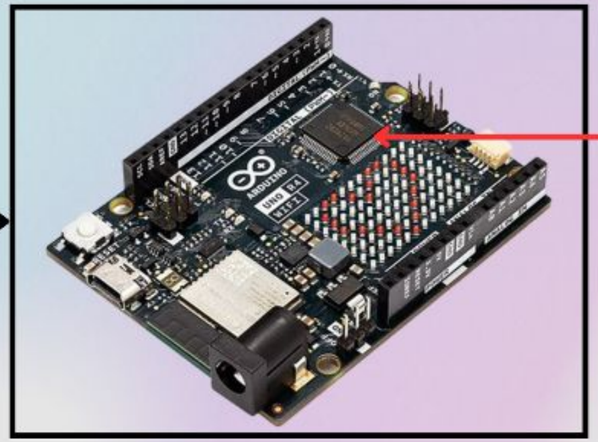
The interface is used to achieve the following tasks:

- Write code for a robot
- Compile the code (translate) it into a set of instructions that the machine will understand
- Finally upload (put) the code on the microcontroller
- Sometimes you can also use this interface to interact with the machine

ARDUINO® HARDWARE

INSIDE ARDUINO HARDWARE?

- Common Arduino Hardware Types: Arduino Uno, Arduino Mega, Arduino Due, Arduino MKR Series.
- Arduino Hardware is a **development board comprising a microcontroller**



Development Board

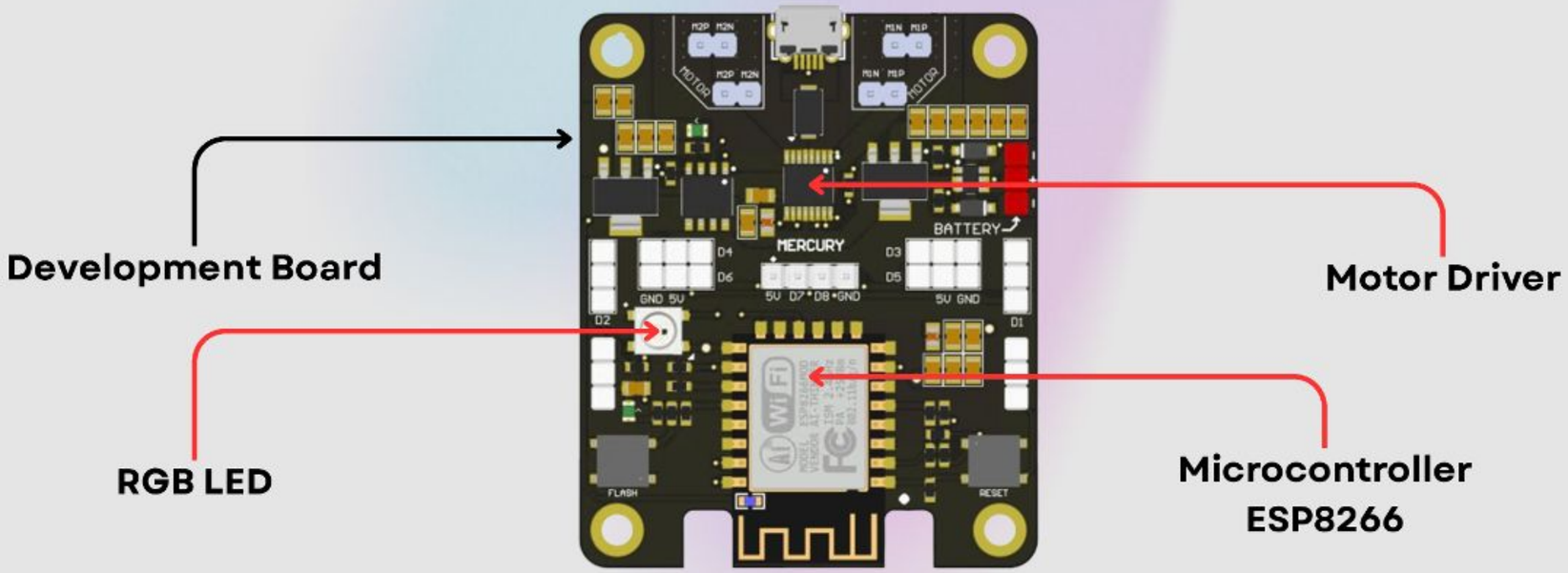
Microcontroller
ATmega 328

MERCURY DEVELOPMENT BOARD

WHAT IS MERCURY?

Mercury is **similar to Arduino but supercharged with more features.**
Example:

- Mercury has an onboard motor driver unlike Arduino
- Mercury has an onboard RGB LED unlike Arduino



CHAPTER 03

ARDUINO® IDE

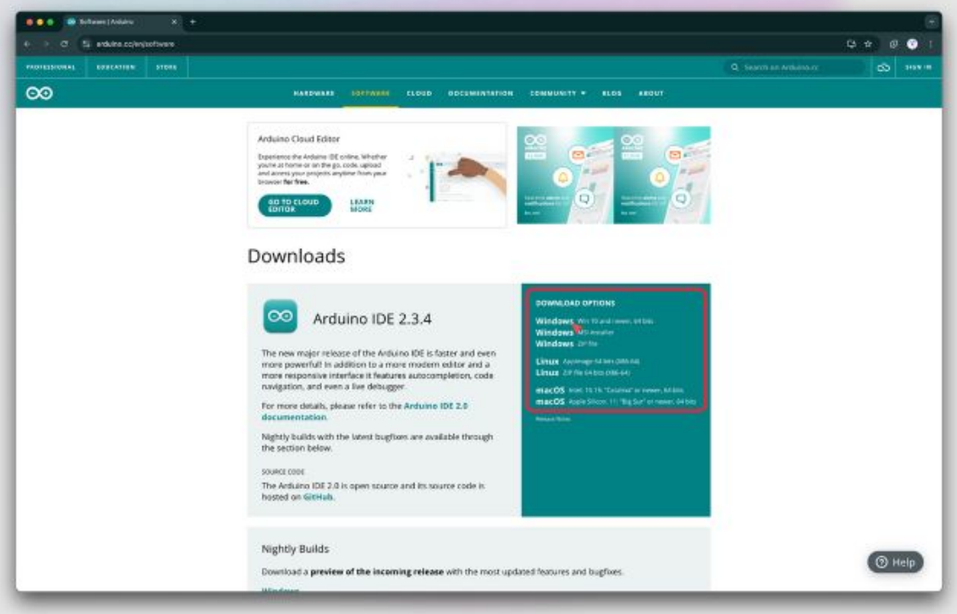
DOWNLOAD AND INSTALL ARDUINO

Download

Download the latest version of application from the following link:
https://www.arduino.cc/en/software

INSTALL

Follow the installation steps to successfully get the software running on your computer.



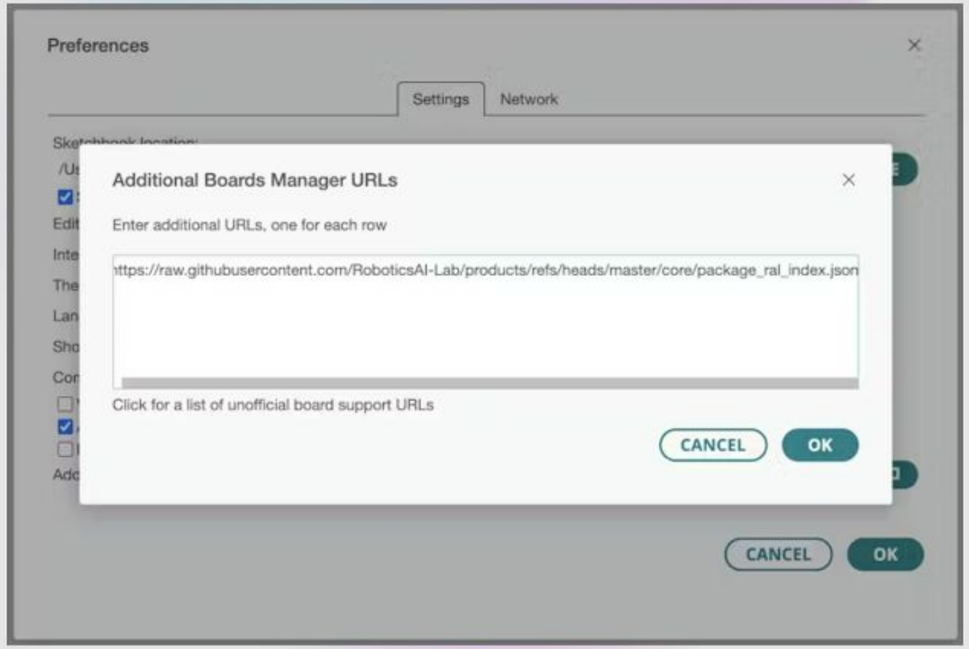
CHAPTER 04

MERCURY BOARD SETUP

MERCURY BOARD SETUP

BOARD FILE

- Navigate to “*Settings*”
- Copy-paste this file link into “*Additional Boards Manager URLs*”
- *https://raw.githubusercontent.com/RoboticsAI-Lab/products/refs/heads/master/core/package_ral_index.json*

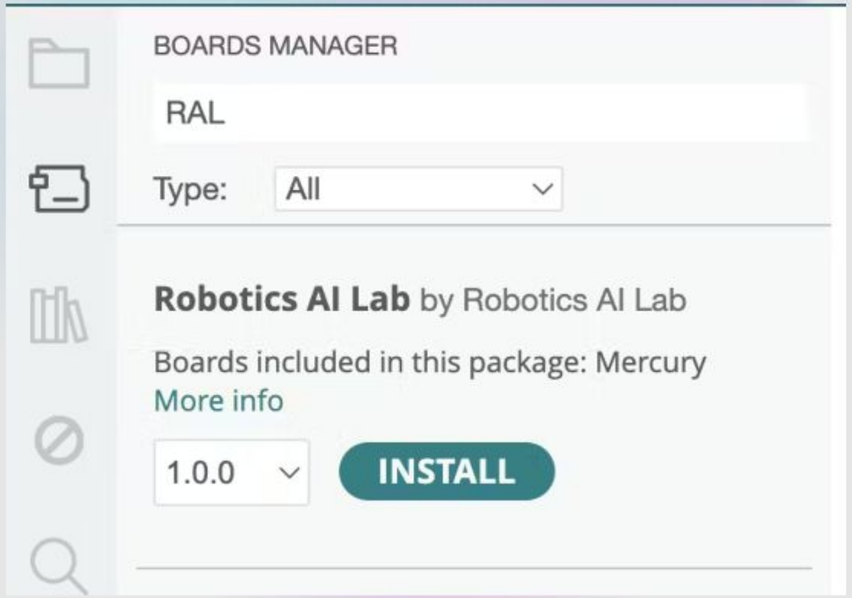


Additional Boards Manager URLs

MERCURY BOARD SETUP

BOARD MANAGER

- Navigate to “Tools >>> *Board Manager*” or click on “*Board Manager*” icon in sidebar
- Search for “Robotics AI Lab”
- Install latest version of the package

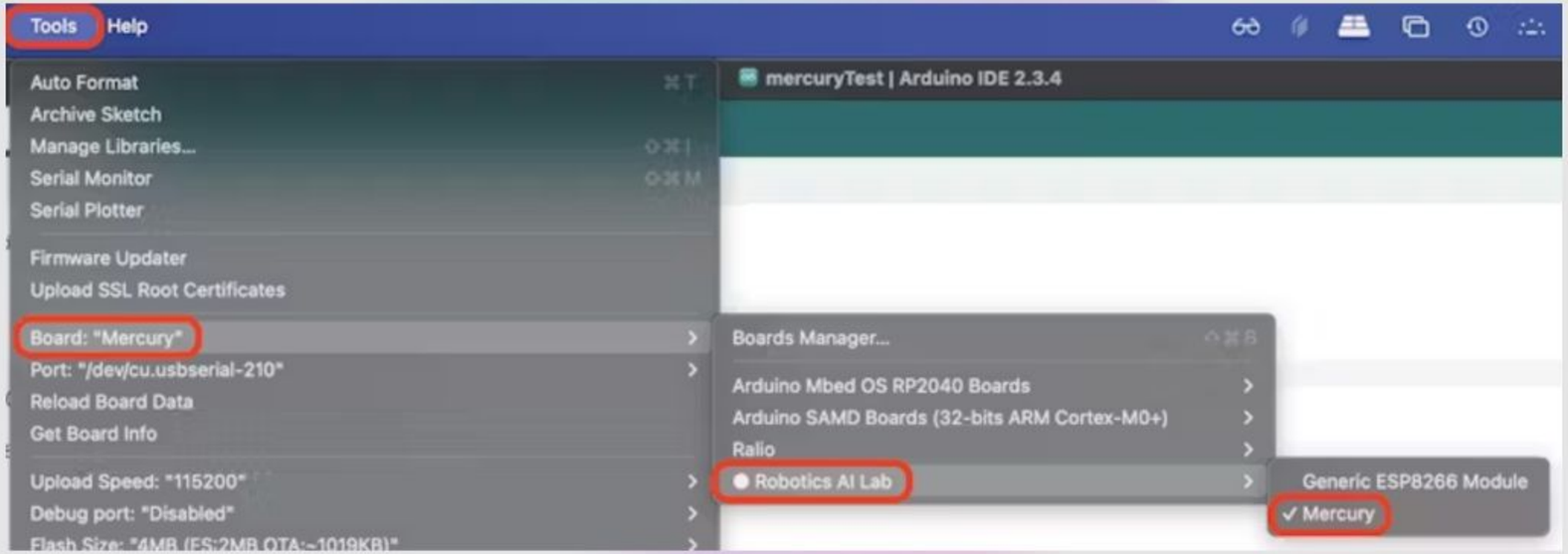


Robotics AI Lab: Package

MERCURY BOARD SETUP

BOARD SELECTION

- Navigate to “Tools >>> *Board*”
- Select "Mercury" board from Tools > Board menu after installation.



Robotics AI Lab: Mercury Board

CHAPTER 05

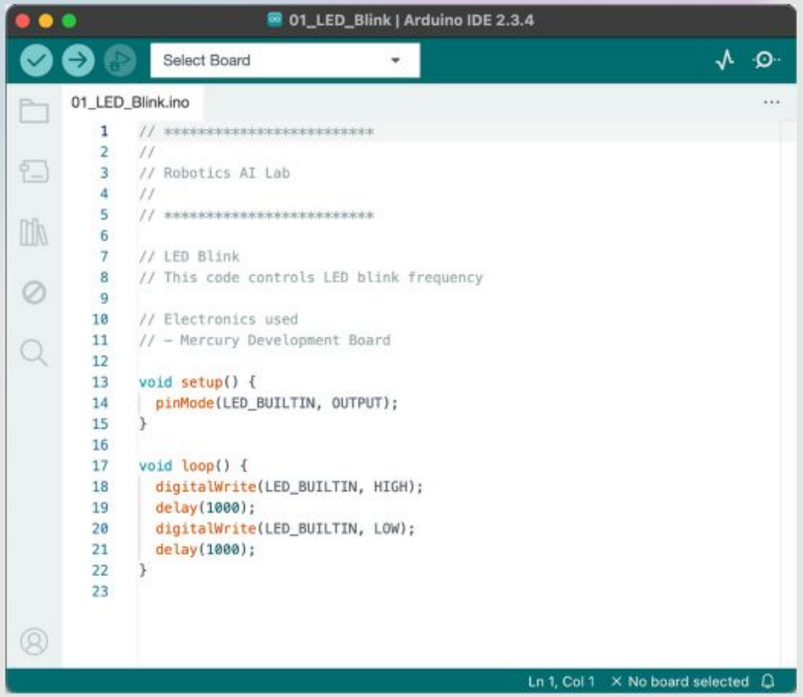
UPLOADING FIRST PROGRAM

UPLOADING FIRST PROGRAM

CODE DOWNLOAD

- Navigate to the following link and copy the code
- https://github.com/RoboticsAI-Lab/products/blob/master/projects/02_Smart_Car_4WD/01_LED_Blink/01_LED_Blink.ino

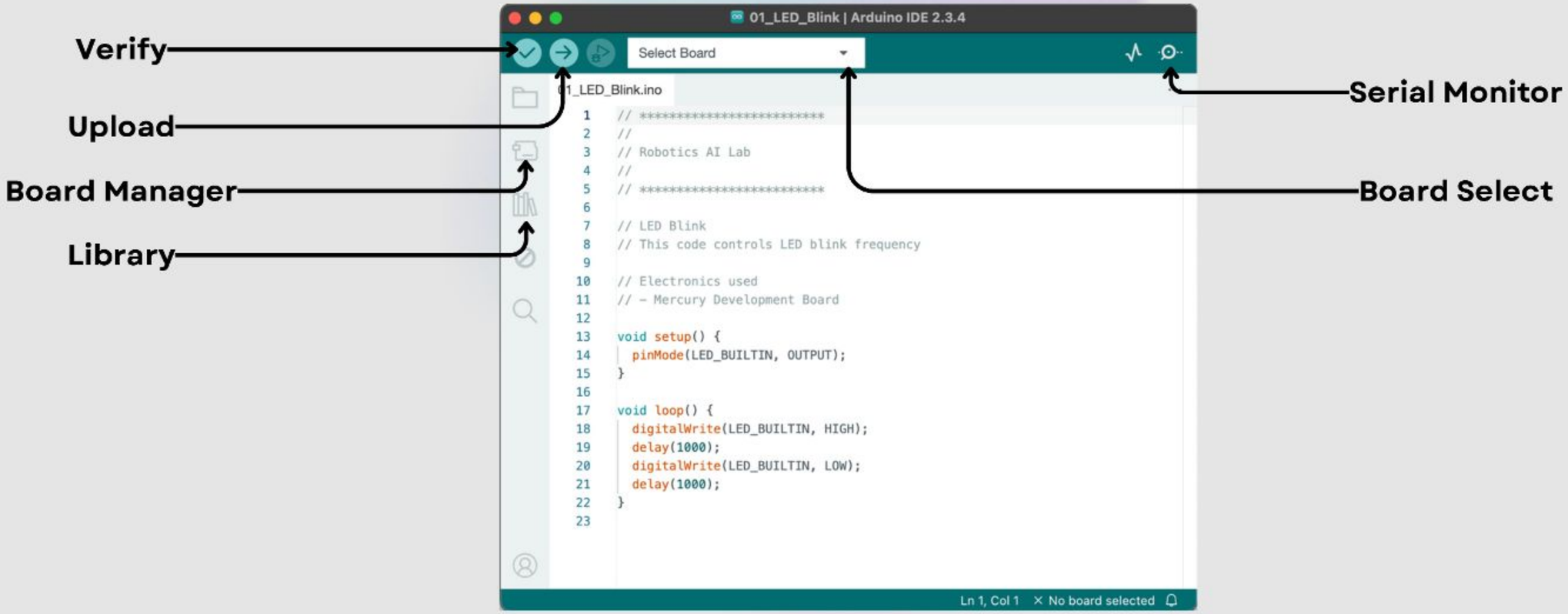
For now, let us just focus on getting the first program on the board.
Later on we will understand the code and other nuances in detail.



CODING WITH ARDUINO®

WHAT IS IDE?

- Integrated Development Environment
- It's an application that helps build a program that can later be uploaded on the microcontroller

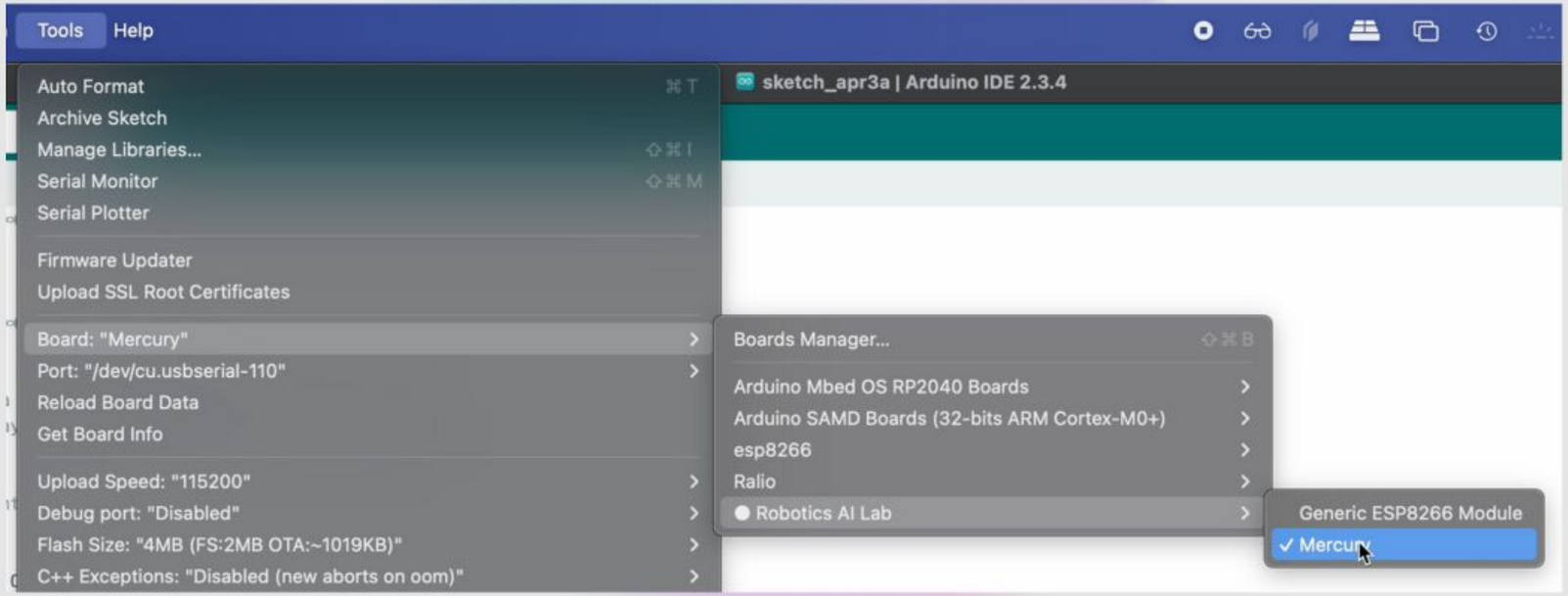


Arduino IDE

UPLOADING FIRST PROGRAM - LED BLINK

BOARD SELECT

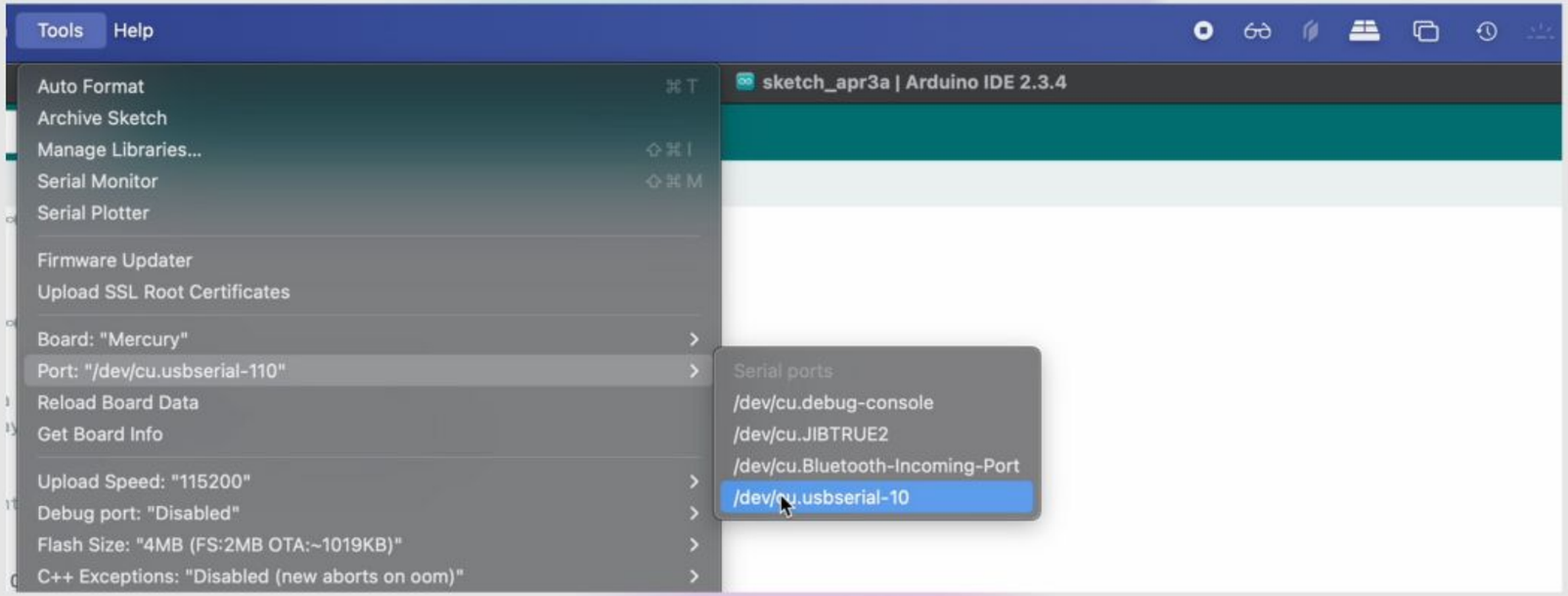
- Navigate to “Tools >>> Board >>> Robotics AI Lab >>> Mercury”
- Select the appropriate version of the board



UPLOADING FIRST PROGRAM

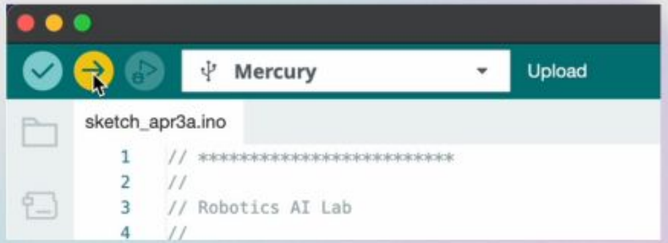
PORT SELECT

- Navigate to “Tools >>> Port >>> USB port”
- Select the appropriate port. It’s generally a USB port

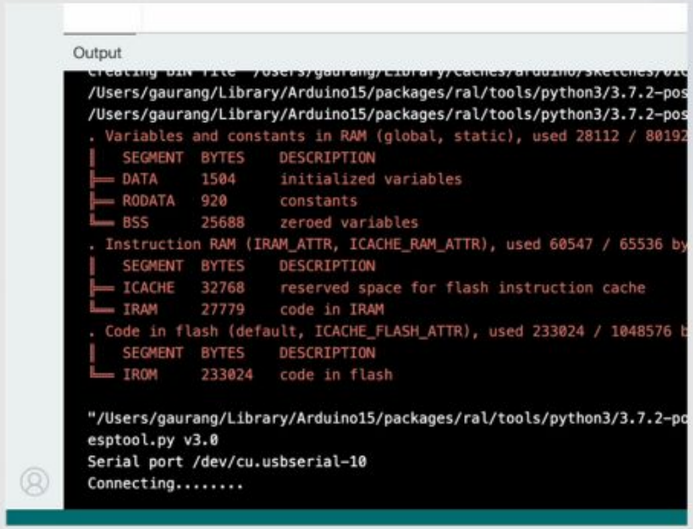


UPLOADING FIRST PROGRAM

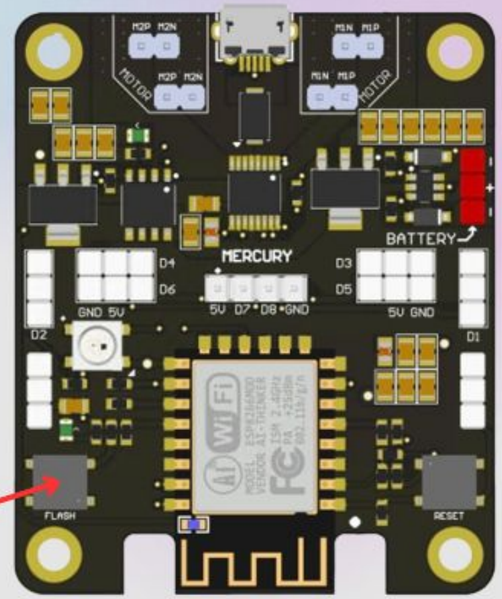
CODE UPLOAD

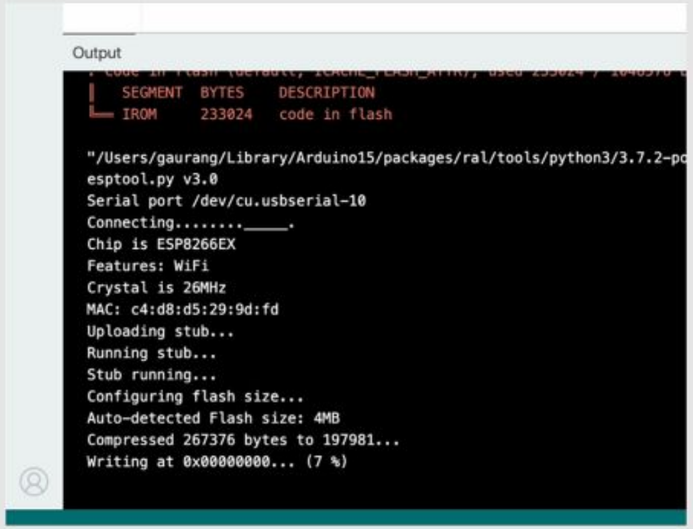


Click *Upload*



Press and Hold *Flash* Button
when you see “Connecting...”



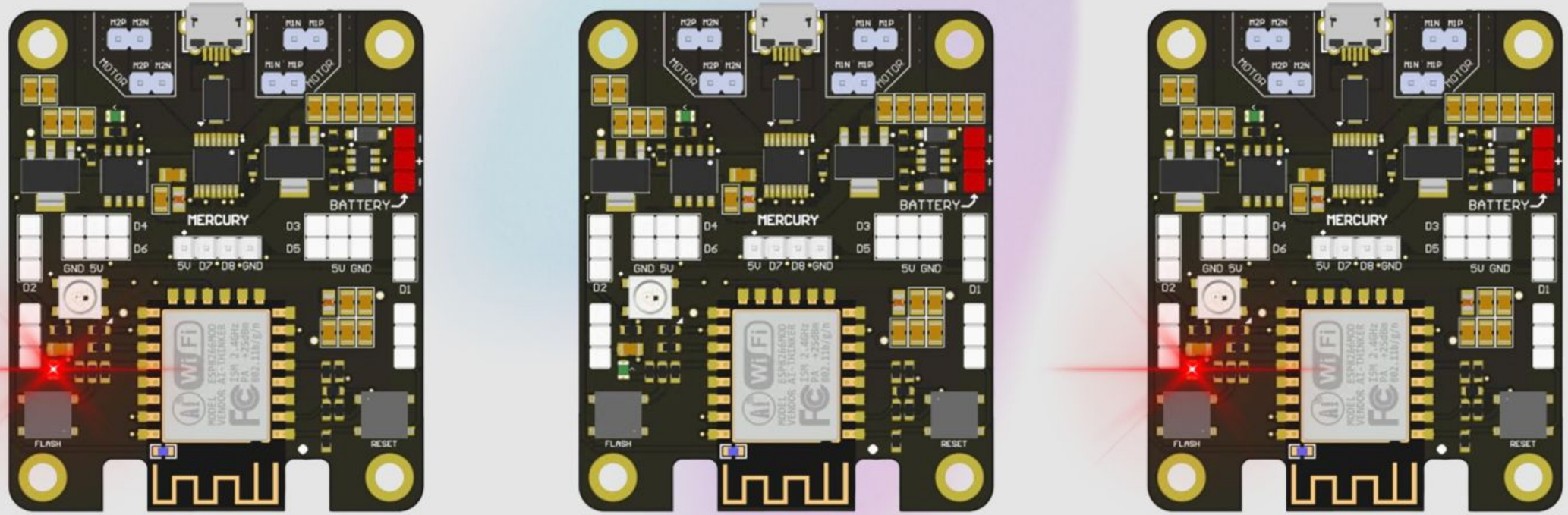


Release *Flash* Button
after code starts to upload

UPLOADING FIRST PROGRAM

SUCCESSFUL UPLOAD

- The LED should start blinking after successful code upload



CHAPTER 06

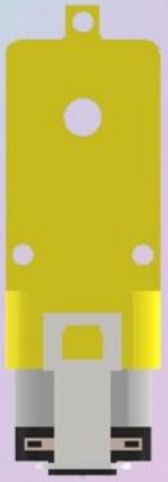
UPLOADING LED - OUTPUT DEVICE

OUTPUT DEVICES

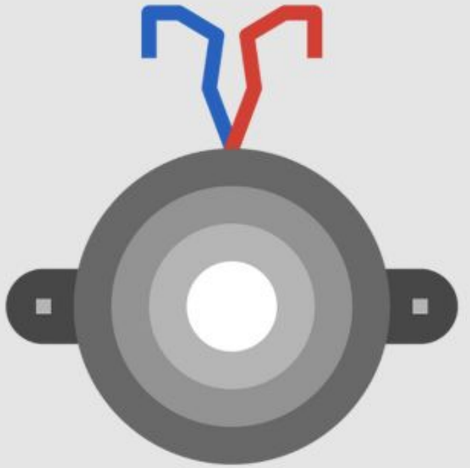
WHAT ARE OUTPUT DEVICES?

- Components or devices controlled by the microcontroller are considered to be output devices

LED



Motor



Buzzer

LED

WHAT IS AN LED?

- Light Emitting Diode (also known as LED)
- A source of light that glows when electricity flows through it in specific direction.
- Now a days commonly used in TVs, Home Lighting, Decor Lighting, etc.

LED

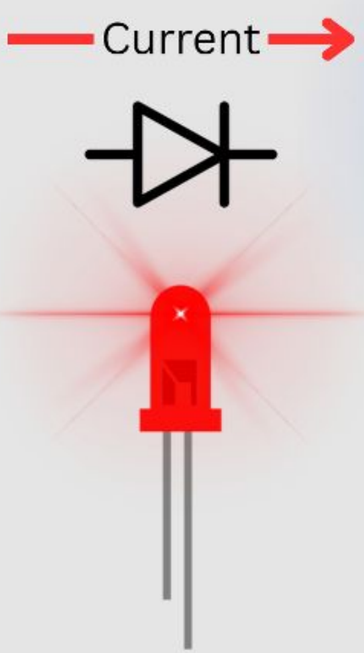
Home Lighting

Decorative Lighting

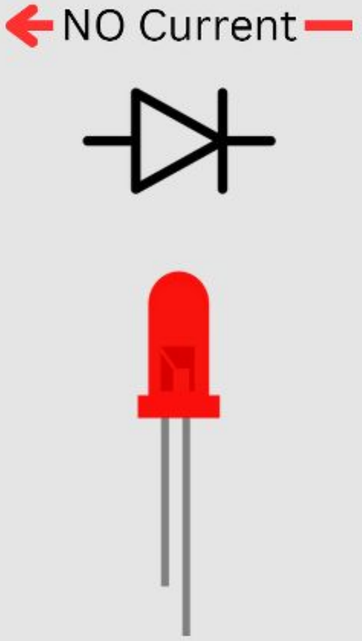
DIODE

THE THEORY BEHIND LED OPERATION?

- Diode only allow current to flow in on direction, Anode to Cathode
- Thus, electron flow from Cathode to Anode
- During this, electrons travel through a special medium when they loose energy in form of light.
- Thus “Light getting emitted through a Diode”



LED ON



LED OFF

FUN FACT!

THE FIRST LED WAS INVENTED BY NICK HOLONYAK JR. IN THE YEAR 1962

Source: https://ece.illinois.edu/newsroom/51161

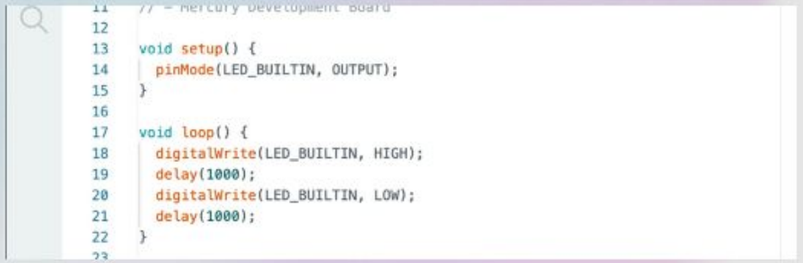
Nick Holonyak, Jr., the John Bardeen Endowed Chair Emeritus in Electrical and Computer Engineering and Physics

CHAPTER 07

BASIC ARDUINO® CODING

LED BLINK CODE

CODE OVERVIEW



```
// - Mercury Development Board

void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {
  digitalWrite(LED_BUILTIN, HIGH);
  delay(1000);
  digitalWrite(LED_BUILTIN, LOW);
  delay(1000);
}
```

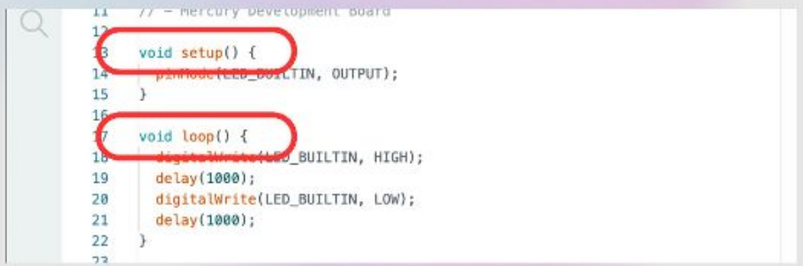
SKETCH

WHAT IS SKETCH?

A sketch is a set of instructions written in the Arduino® programming language that tells the microcontroller board what to do. It's like a recipe that tells a chef how to make a dish.

Basic structure of a sketch

- setup()
- loop()



SETUP()

WHAT IS SETUP?

The setup() function is like the beginning of a school day when you get ready for your lessons.

- It's where you set up everything you need to run your program.
- This part runs once when the microcontroller board is Powered-ON or Reset.
- It’s like telling the board what different components will be connected on it.
- This part is also responsible for “waking up” different parts of the board - example Serial, Wire, etc.

LOOP()

WHAT IS LOOP?

The loop() function is like the main part of your school day, where you do your activities repeatedly.

This part of the sketch runs over and over again until you turn OFF the microcontroller board.

```
16
17  void loop() {
18    digitalWrite(LED_BUILTIN, HIGH);
19    delay(1000);
20    digitalWrite(LED_BUILTIN, LOW);
21    delay(1000);
22  }
23
```

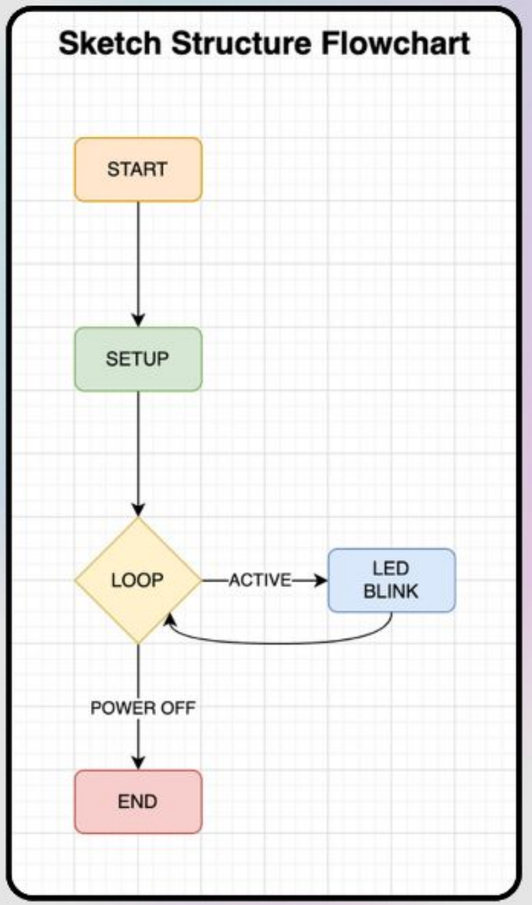
Blinks LED ON-OFF
until Microcontroller is shutdown

```
77  void loop() {
78    stop_distance = measureDistance();
79    if (stop_distance > 5) {
80      dogo_cat_walk();
81    } else {
82      dogo_pause();
83    }
84  }
```

Dog Robot keeps walking and avoiding Obstacle
until Microcontroller is shutdown

SKETCH

SKETCH FLOWCHART



PINMODE()

WHAT IS PINMODE() ?

- Defined as: **pinMode(pin, mode);**
- Setting up the pinMode is like telling the microcontroller, what job each pin has to do.
- There are two types of basic mode: **INPUT** and **OUTPUT**.
- This line in the setup() function, sets the built-in LED pin to work as an OUTPUT.
- Think of it like setting up a light switch that you can turn on and off.

```
13  void setup() {
14    pinMode(LED_BUILTIN, OUTPUT);
15  }
```

DIGITALWRITE()

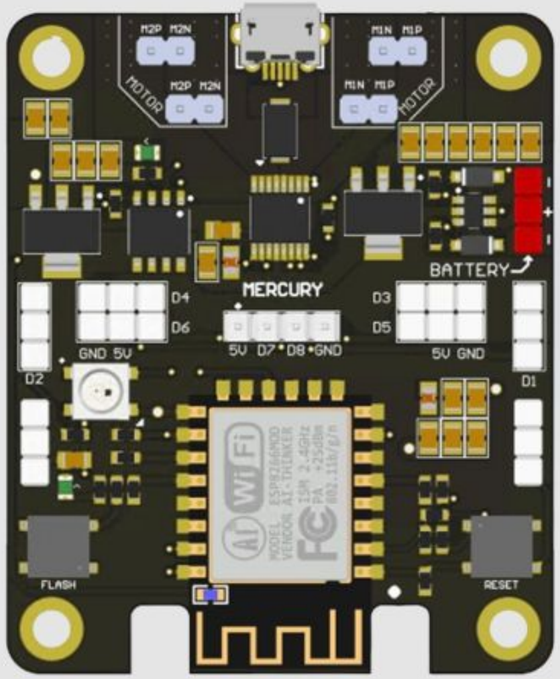
WHAT IS DIGITALWRITE() ?

- Defined as: **digitalWrite(pin, value);**
- Think of digitalWrite as the action of **turning the light ON and OFF**.
- There are two types of value that can be written to a pin, **HIGH** an **LOW**.
- If the pin is set as **OUTPUT with pinMode**, then setting the value of pin **HIGH or '1'** is like telling the microcontroller to **turn ON** the component connected to that pin.
- Similarly, setting the value of pin **LOW or '0'** is like telling the microcontroller board to **turn OFF** the component connected to that pin.

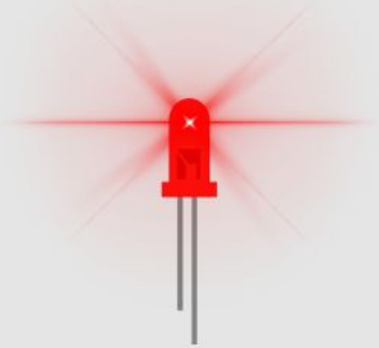
```
17  void loop() {
18    digitalWrite(LED_BUILTIN, HIGH);
19    delay(1000);
20    digitalWrite(LED_BUILTIN, LOW);
21    delay(1000);
22  }
```

DIGITALWRITE()

DIFFERENT DIGITALWRITE() VALUE TO LED PIN



digitalWrite(LED_PIN, HIGH)



LED ON

digitalWrite(LED_PIN, LOW)

LED OFF

DIGITAL AND BINARY

WHAT IS THE WORD “DIGITAL”?

- Use numbers to represent information

THEN WHAT IS THE WORD “BINARY”?

- A FORM OF DIGITAL SYSTEM THAT ONLY USES ‘0’ AND ‘1’ TO REPRESENT INFORMATION.

Think of a light switch in your home. The switch can be either **ON** or **OFF**. These two positions (ON and OFF) are like the ‘1' and ‘0’ respectively in the digital world.

‘1’ and ‘0’ are also referred to as ‘**HIGH**’ and ‘**LOW**’ in embedded world.

DELAY()

WHAT DOES THE DELAY() FUNCTION ACCOMPLISH?

The delay() function tells the Arduino to **pause and wait** for some give milliseconds before doing the next thing.

1 millisecond = 1second / 1,000

TRAFFIC LIGHT EXAMPLE

Imagine you're programming a traffic light with an Arduino.

- First, the red light turns on.
- But you don't want it to change right away, correct? The cars need time to stop for sometime!
- So, you use the delay() function for the red light to stay ON for few seconds before it turns green.

Red Light ON

delay of 30 sec

Green Light turns ON

FUN FACT!

YEARS IN WHICH DIFFERENT COLOR LEDS WERE INVENTED

8 year later →

23 year later →

Red: 1962

Green: 1970

Blue: 1993

CHAPTER 08

USE OF SERIAL MONITOR

SERIAL COMMUNICATION

TO INTRODUCE SERIAL COMMUNICATION...

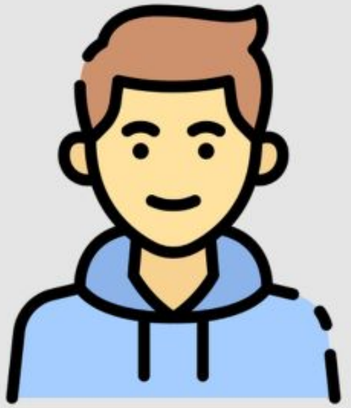
Serial communication is basically **sending and receiving information, one piece at a time**, in a systematic and orderly way.

Let us understand this with an example - It's like having a conversation with your friend...

- It's your turn to speak and you speak one word at a time
- Your friend carefully listens to each word you say
- Now, it's your friend's turn to respond
- Your friend speaks one word at a time and you carefully listen to each word
- That's how serial communication works, sending and receiving messages step by step

Hey, did you checkout the latest AI model launch by Google® ?

Yes, Gemini correct? It was lit!

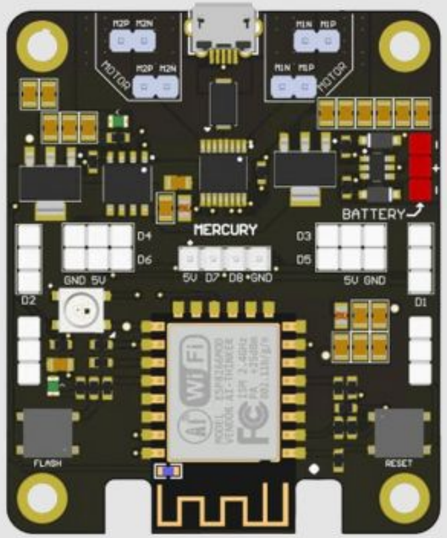


SERIAL COMMUNICATION

SERIAL COMMUNICATION WITH MERCURY

Here the serial communication will happen between Mercury and your laptop via USB.

FYI... USB stand for **Universal “Serial” Bus**

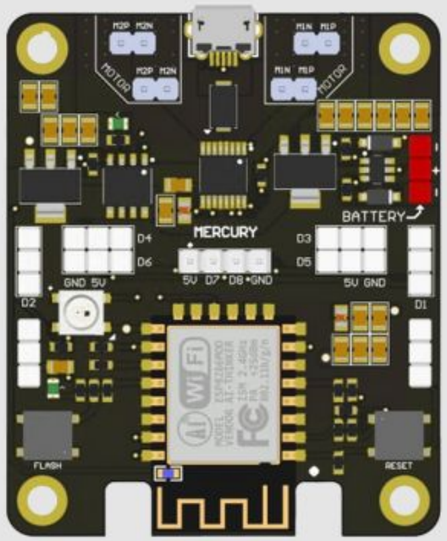


2-way communication using USB cable

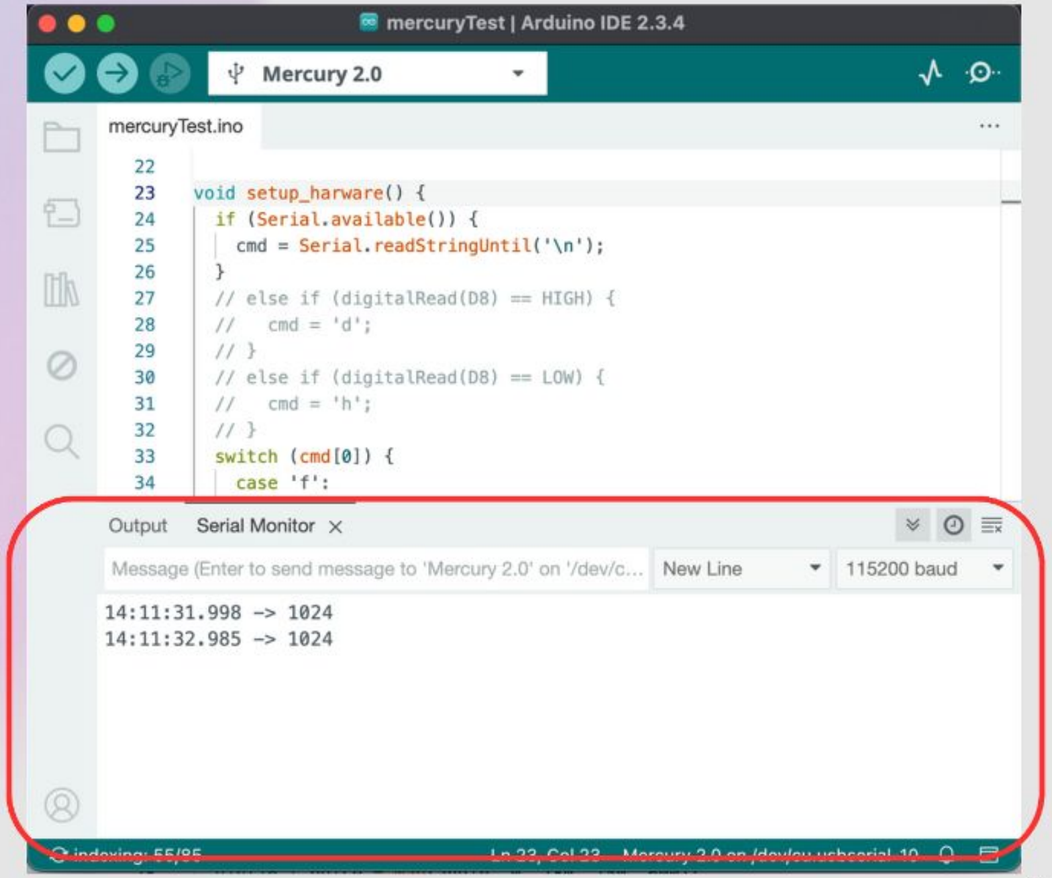
ARDUINO SERIAL COMMUNICATION

SERIAL MONITOR

Serial Monitor is a medium used in Arduino IDE to send and receive data when using serial communication.



2-way communication using USB cable

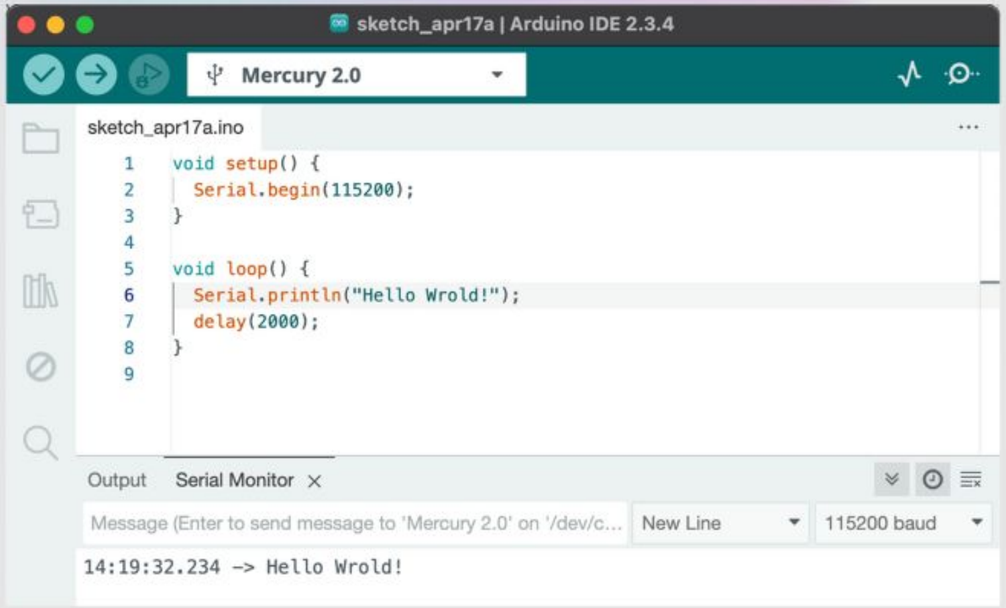


SERIAL COMMUNICATION

SERIAL BEGIN

Serial.begin(115200);

- Serial.begin(...) starts the serial communication.
- 115200 → **baud rate**: it's the speed of the conversation.
- In the code below, Mercury will be sending the text "**Hello World!**" at 115200 baud rate and the Laptop (Serial Monitor) will be receiving this text at 115200 baud rate, to print it on the serial monitor.



SERIAL COMMUNICATION

BAUD RATE

Baud rate: it's the speed of the conversation.

- It's important for both communicating devices to talk at the same speed
- If they don't, the message might sound like gibberish to the listener.

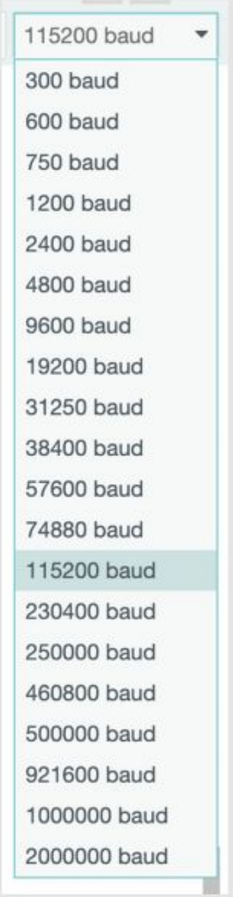
COMMONLY USED BAUD RATES

While there are several baud rates starting from 300, all the way up to 2000000.
The most common ones are 9600 and 115200.

SO WHAT IS 115200?

115200 baud rate implies a speed of **115,200 bits of data per second**

- This implies, Mercury Board can send and receive 115200 bits of data per second.
- 115200/1000000 → 0.1 Mbps
- Do you know what is your WiFi speed, well its generally 100Mbps
- 1000 times the speed of this serial communication



SERIAL PRINT

SERIAL PRINT()

Serial.print(“Hello World!”);
It is a command that tells the microcontroller to send data to your computer so you can see it on the Serial Monitor (the built-in console in the Arduino IDE).

SERIAL PRINTLN()

It’s like hitting an “enter” to start on a new line.

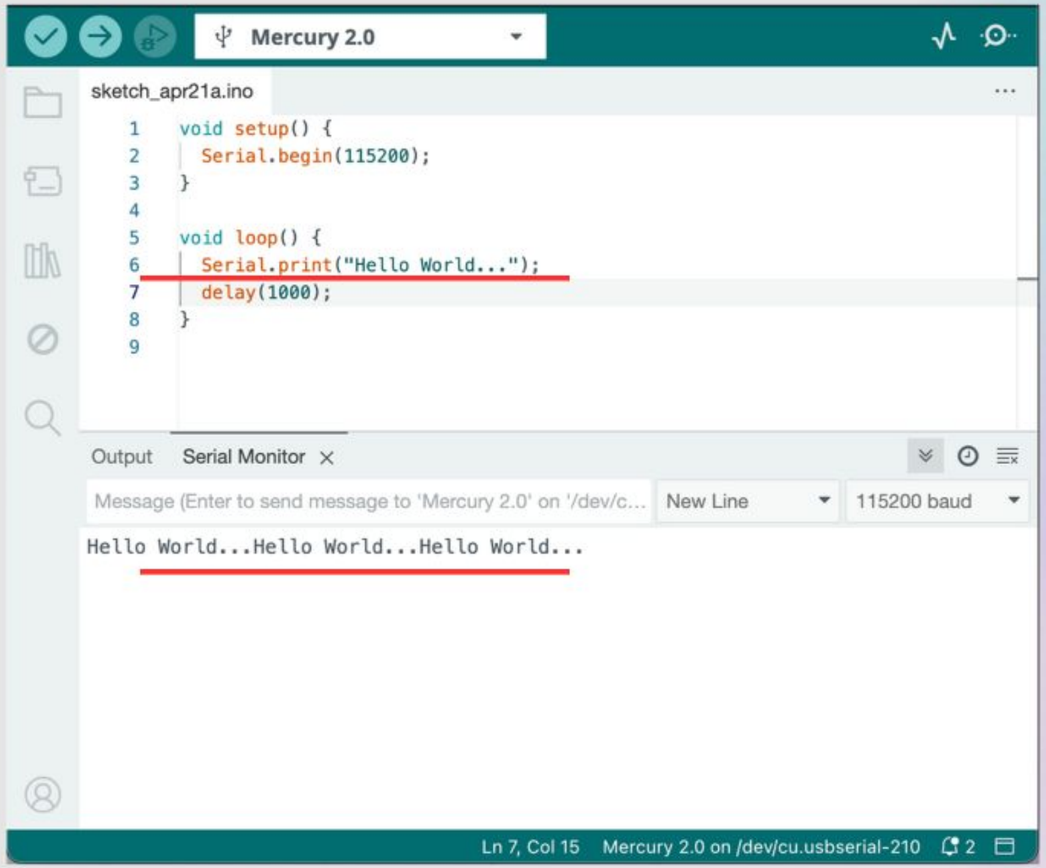
- Serial.print() function prints text sequentially one-after-another
- You will explicitly have to define a tab or white-space for serial function to display that “space”
- Serial.println() places an “\n” or a new-line character in the end of the text
- Thus when the serial printing function is called next time, it starts printing on a new line

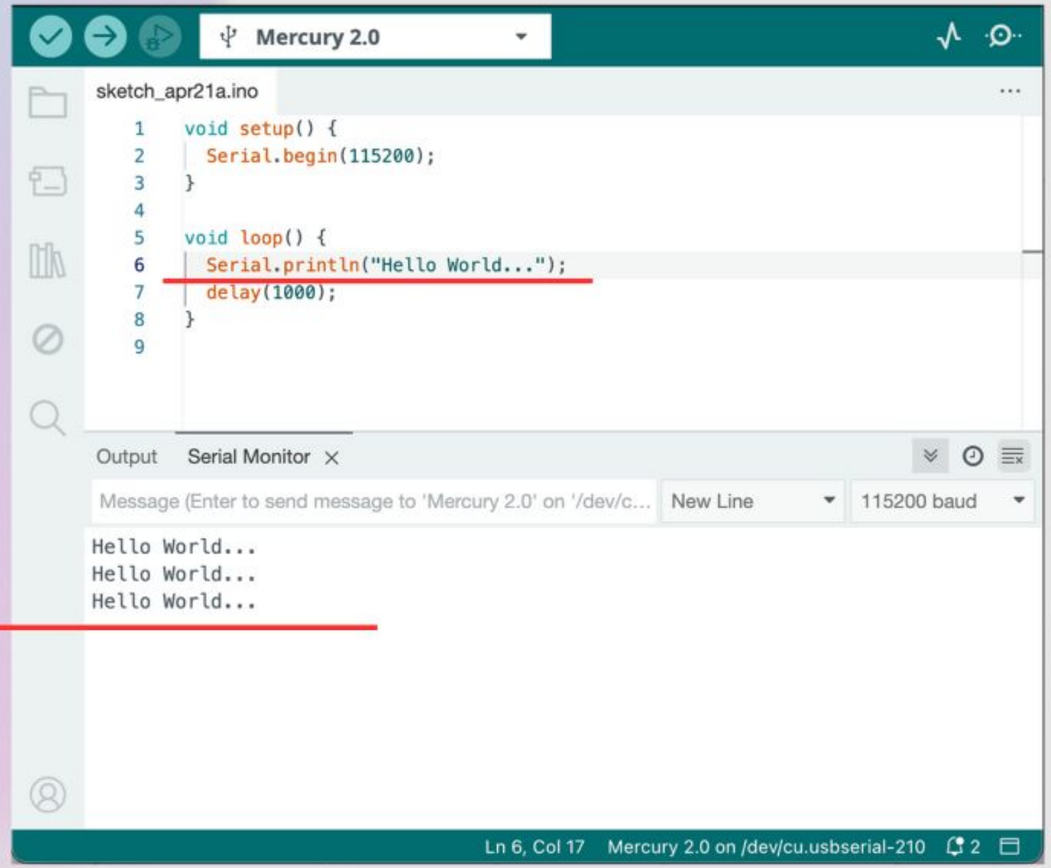
Another way to look at this is to consider your laptop cursor, the location where cursor is displayed , is where you type.

Serial.println() function moves the cursor to a new line, thus the next character that gets printed, appears on a new line.

SERIAL PRINT

SERIAL PRINT() VS SERIAL PRINTLN()





CHAPTER 09

VARIABLES AND DATA TYPES

VARIABLES

WHAT IS A VARIABLE IN GENERAL?

Variable, as the name suggests, is something that can change over time. In programming, variable is like a **box that holds information**, information that can change.

WHY USE A VARIABLE?

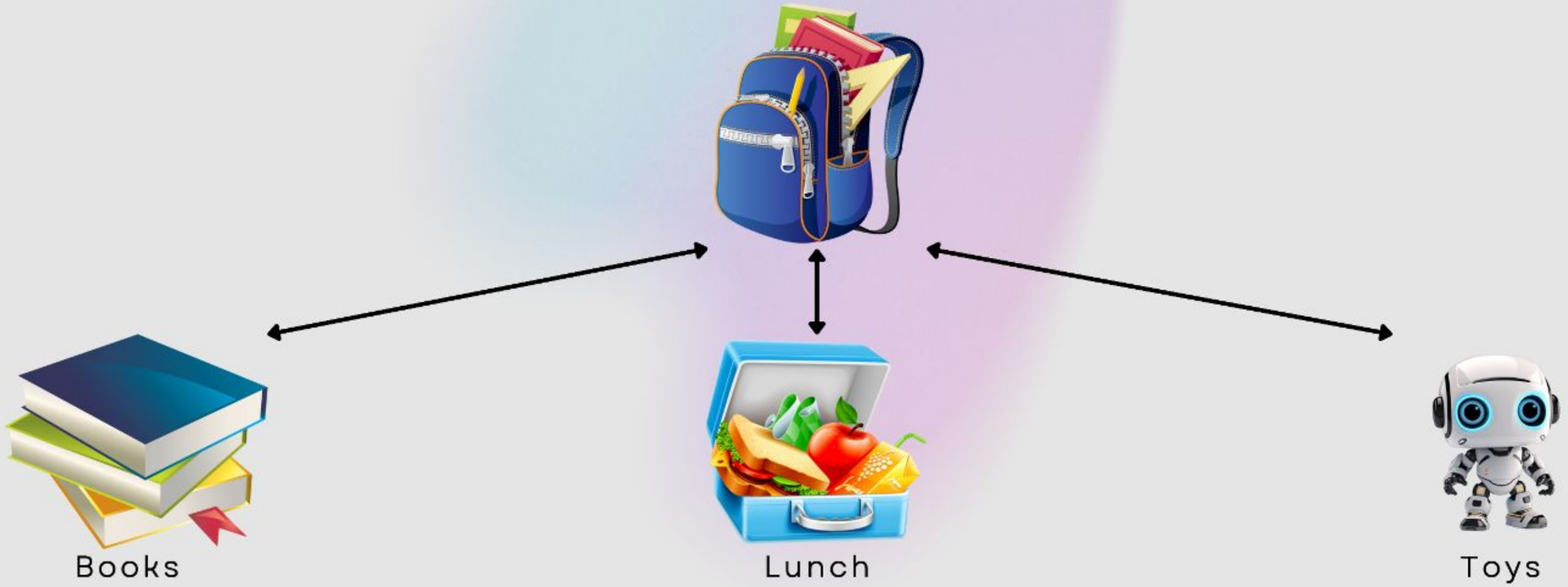
Variables allow us to store information that we can be used and modified while our program runs.

- Help store information
- Make program flexible
- Make program easy to understand

VARIABLES

SCHOOL BACKPACK

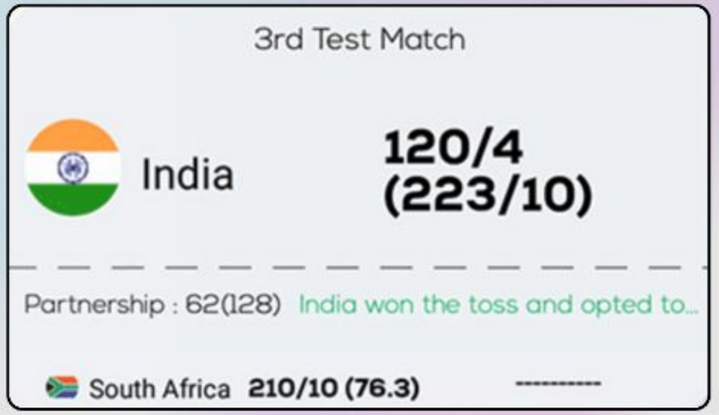
Imagine you have a **backpack**. Sometimes you put a **book** in it, sometimes your **lunch**, and sometimes your **toys**. The backpack is like a variable that holds different items at different times.



VARIABLES

POINT TABLE

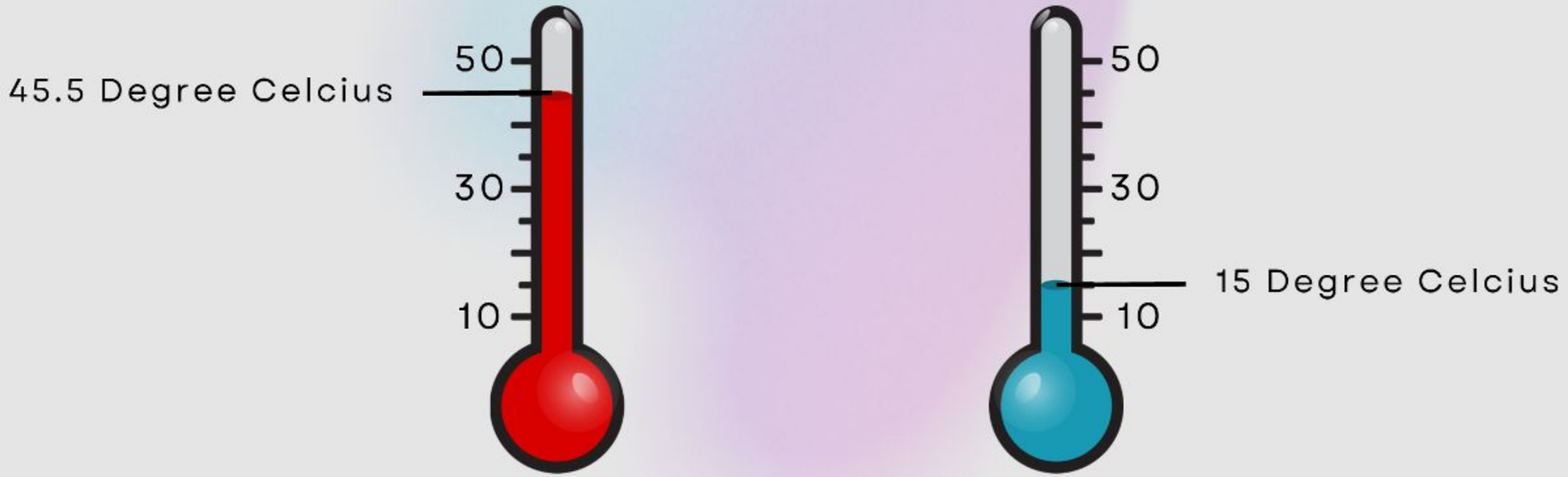
When you play a game, you keep track of your **score**. The score is a variable because it changes as you play the game. It starts at **zero and increases as you earn points**.



VARIABLES

TEMPERATURE TRACKING

Think of a thermometer that shows the temperature. The **temperature reading** is a variable because it changes throughout the day.



LED BRIGHTNESS

LED BRIGHTNESS CHANGE USING VARIABLE

- **“brightness”** is defined as a variable with an initial value of **zero**.
- As the program begins, the brightness level is set to **100.**
- Later, the variable values is updated to level **200**.

At level 100, the LED glows dim and at level 200, the LED glows bright.

“brightness” set as a variable

```
1  #include <Arduino.h>
2
3  void setup() {
4    pinMode(BUILTIN_LED, OUTPUT);
5
6  }
7
8  int brightness = 0;
9  void loop() {
10   brightness = 100;
11   analogWrite(BUILTIN_LED, brightness);
12   delay(1000);
13   brightness = 200;
14   analogWrite(BUILTIN_LED, brightness);
15   delay(1000);
16 }
```

If someone did not tell you that this LED brightness, would you know what this 100 and 200 is doing?

```
void loop() {
  analogWrite(BUILTIN_LED, 100);
  delay(1000);
  analogWrite(BUILTIN_LED, 200);
  delay(1000);
}
```

DATA TYPE

WHAT ARE DATA TYPES?

In programming, data types tell the computer **what kind of information** a variable is storing. It's like knowing what kind of things you're putting in your backpack–whether it's a book, a lunchbox, or a toy.

COMMON DATA TYPES:

- **int** → Integer
  - Used to store whole numbers (-5, 0, 1)
  - Example: Keeping track of number of books in the backpack
- **float** → Floating point (fraction/decimal)
  - Used to store number with decimal points (-5.5, 0.2, 1.9)
  - Example: Temperature measured by thermometer => 45.5 degree C
- **char** → Character
  - Used to store a single character (S, g, @,$)
  - Example: First letter of your name (Ralio Tech -> ‘R’)
- **bool** → Boolean
  - Used to store binary states ‘0’ or ‘1’, ‘TRUE’ or ‘FALSE’, ‘HIGH’ or ‘LOW’

DATA TYPE

int number_of_pizza = 2

float number_of_pizza = 1.5

char first_letter = ‘p’

VARIABLE DECLARATION

What is variable declaration?

Declaring a variable is like putting a “**label**” to it. This helps decide what type of information will be stored in that variable.

int number_of_pizza = 2

number_of_pizza is **declared** as an “**integer**” type variable as it will store the information relating to quantity of pizza

VARIABLE DECLARATION

LED BRIGHTNESS

“brightness” is defined as a variable and **declared as an integer type**.

“brightness” declared as an **integer** variable

```
1   #include <Arduino.h>
2
3   void setup() {
4     pinMode(BUILTIN_LED, OUTPUT);
5
6   }
7
8   int brightness = 0;
9   void loop() {
10    brightness = 100;
11    analogWrite(BUILTIN_LED, brightness);
12    delay(1000);
13    brightness = 200;
14    analogWrite(BUILTIN_LED, brightness);
15    delay(1000);
16  }
```

CHAPTER 10

CONTROL STRUCTURE

CONTROL STRUCTURES

WHAT ARE CONTROL STRUCTURES?

Control structures in programming are like the rules or instructions that tell the program what to do next. They help the program make decisions and repeat actions based on given conditions.

COMMON FORMS FOR CONTROL STRUCTURES:

- **If - Else condition**
  - Action 1 is executed “if” a condition is satisfied “else” action 2 is executed.
- **For Loop**
  - Repeat actions for “n” number of times
- **While Loop**
  - Keep repeating an action till the condition is valid

IF - ELSE

```
if (color of apple “is” red)
   {
      say (YES);
   }
```

condition

action

```
if (number of pizza “is” 1)
   {
      say (YES);
   }
else:
   {
      say (NO);
   }
```

IF - ELSE

WORKING OF IF-ELSE STRUCTURE:

If condition_1 is satisfied, action_1 should get executed **else** (that is if condition_1 is not satisfied) then action_2 should get executed.

SYNTAX

- The actions statements are placed inside the curly brackets { }

```
if (color_of_apple == red)
   {
      print (“YES”);
   }
else
   {
      print (“NO”);
   }
```

FOR LOOP

WORKING OF FOR LOOP

Control structures in programming are like the rules or instructions that tell the program what to do next. They help the program make decisions and repeat actions based on given conditions.

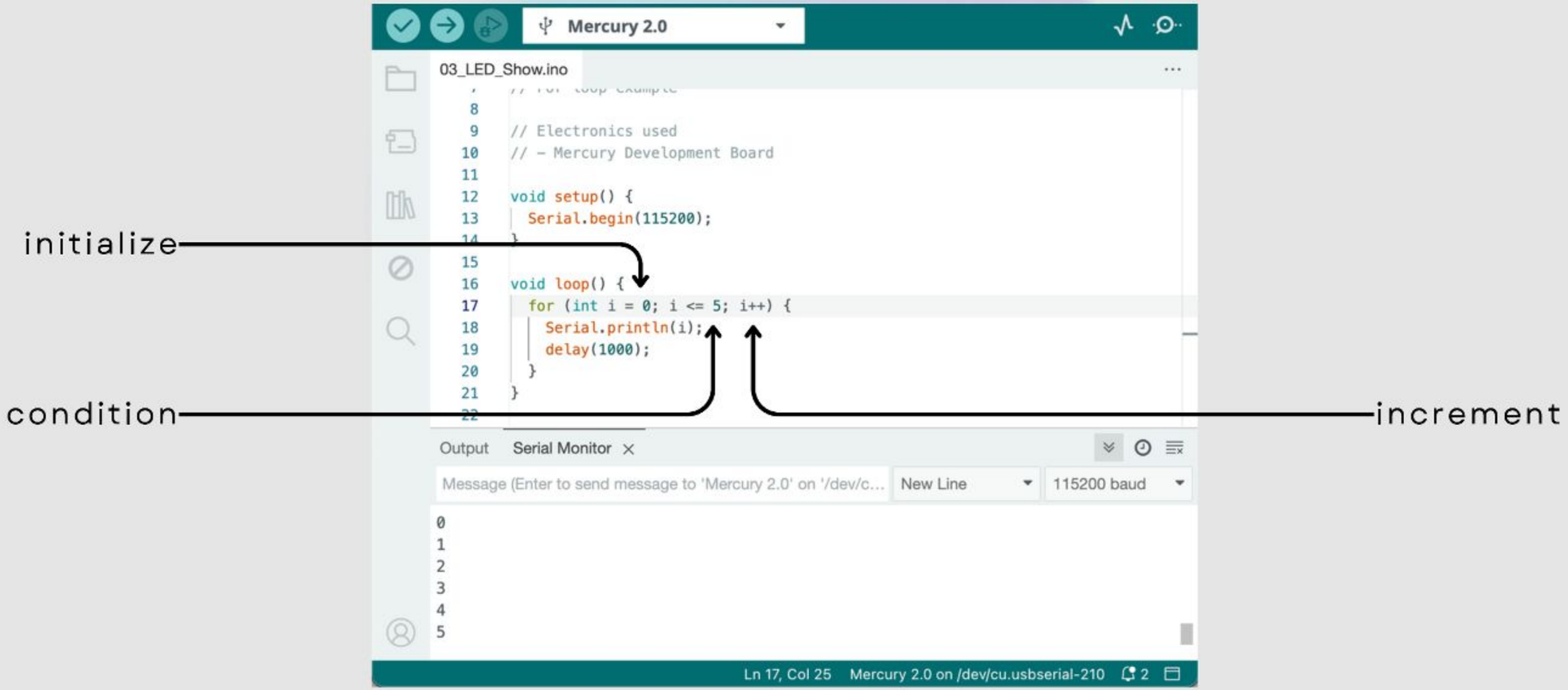
SYNTAX

- “;” is placed after each statement in initiating a for loop
- The actions statements are placed inside the curly brackets { }

FOR LOOP

WORKING OF FOR LOOP

A variable “i” is initialized as “0” and the task of the for loop is to keep printing value of “i” till it exceeds the value “5”.



WHILE LOOP

WORKING OF WHILE LOOP

A 15 minute timer is set and three friends have to eat as many pizzas as possible.

- You can only eat the pizza, **While** the **timer is running.**
- But as the **timer stopped**, they had to **stop eating** the pizza.

WHILE LOOP

WORKING OF WHILE LOOP

Similar to the FOR LOOP example

```
1   int i = 0;
2
3   void setup() {
4     Serial.begin(115200);
5   }
6
7   void loop() {
8     while (i <= 5) {
9       Serial.println(i);
10      i++;
11    }
12    delay(1000);
13  }
14
```

CHAPTER 11

OPERATORS

OPERATORS

WHAT ARE OPERATORS?

An operator is a symbol that tells the computer to perform a **specific action on values or variables.**

ACTION OPERATORS

- Equals (==)
- Not equal to (!=)
- Greater than (>)
- Less than (<)

EQUALS

IS A == B?

Check if two things are same.

color_of_apple == red ?

TRUE

number_of_pizza == 1 ?

FALSE

2 + 4 == 6?

TRUE

NOT EQUAL TO

IS A != B?

Check if two things are **NOT** same.

color_of_apple != red ?

FALSE

number_of_pizza **!=** 1 ?

TRUE

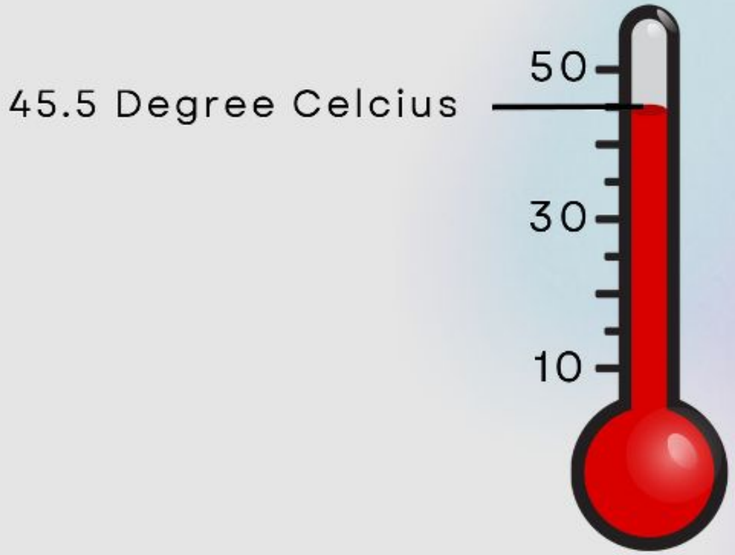
2 + 4 != 6?

FALSE

COMPARISON

IS A > B?

Check if one item is greater than another.

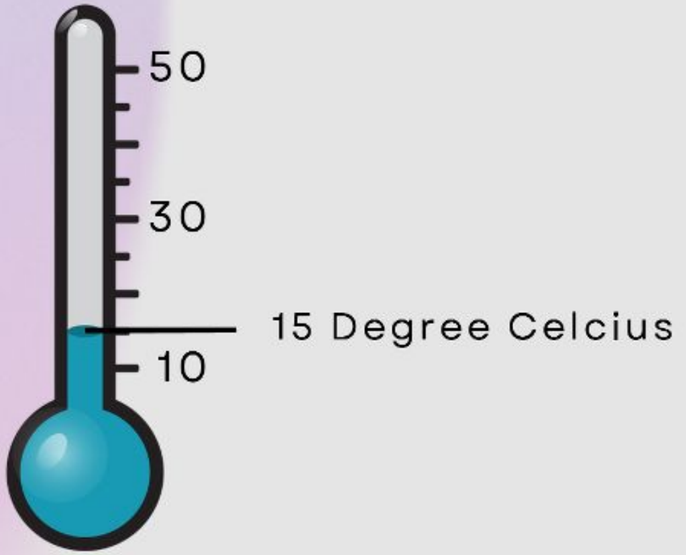


temperature **>** 50 C ?

FALSE

IS A < B?

Check if one item is lesses than another.



temperature **<** 50 C ?

TRUE

CHAPTER 12

RGB LED

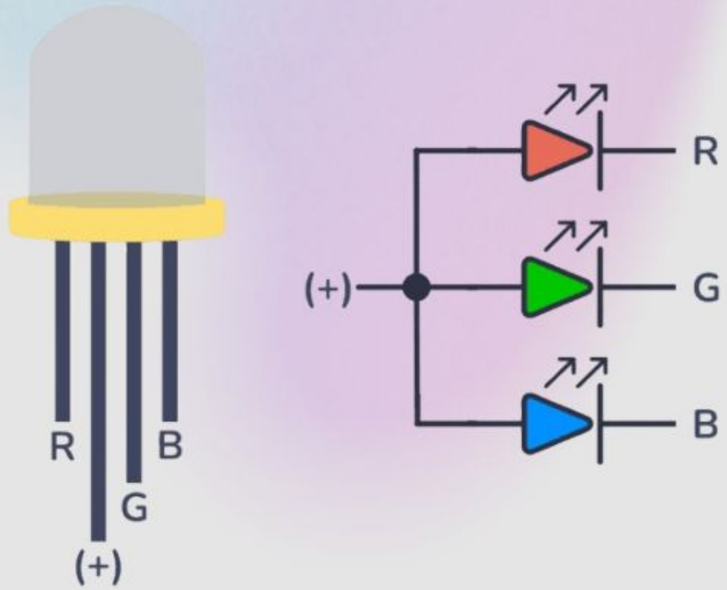
RGB LED STRUCTURE

COMMONLY AVAILABLE RGB LED

In this example, the RGB LED comes with four pins, one common pin and three individual pins to control the three intensity of three primary color.

- **RED**
- **GREEN**
- **BLUE**

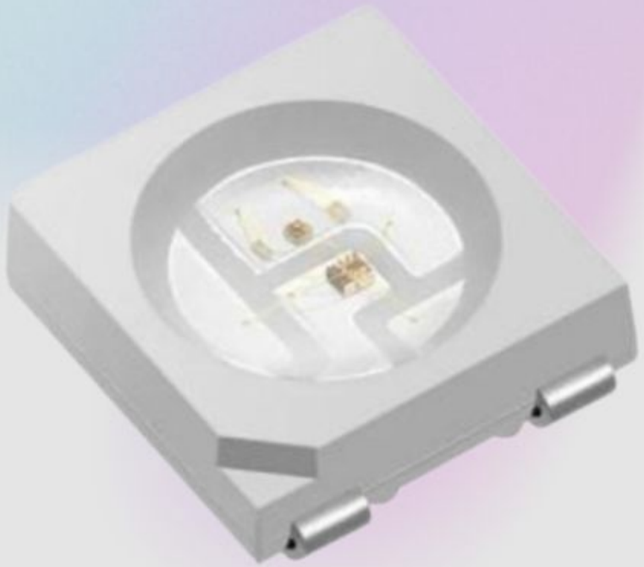
You need three micro-controller pins to operate this RGB LED (what a waste of pins!)



WS2818B

WHAT RGB LED DOES MERCURY HAVE?

- It’s a smart RGB LED!
- Built-in tiny chip that lets us control the color of each LED individually **using just one pin** from the microcontroller.
- One pin - millions of colors 😄



RGB LED OPERATION

Before we start playing with this RGB LED,
we will have to learn about **LIBRARY**...

CHAPTER 13

ARDUINO® LIBRARIES

LIBRARY

WHAT ARE LIBRARIES?

It's a key concept in Arduino where a Library is like a **ready-made toolbox** full of useful tools (code) that **helps you talk to different electronic components – like sensors, motors**, etc. – **without writing all the complex code yourself**.

Imagine trying to build a robot from scratch – not just the body, but every single nut and bolt! 😲

Libraries save you from all that hard work such that, instead of writing 100 lines of code to control an RGB LED, you might just need 2 or 3!

ADAFRUIT_NEOPIXEL

Let us jump into installing this Library

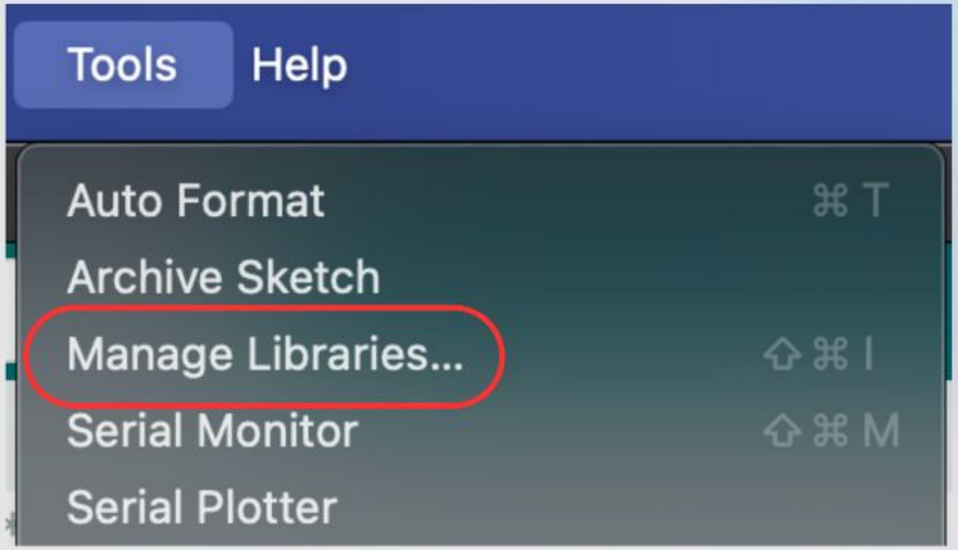
CHAPTER 14

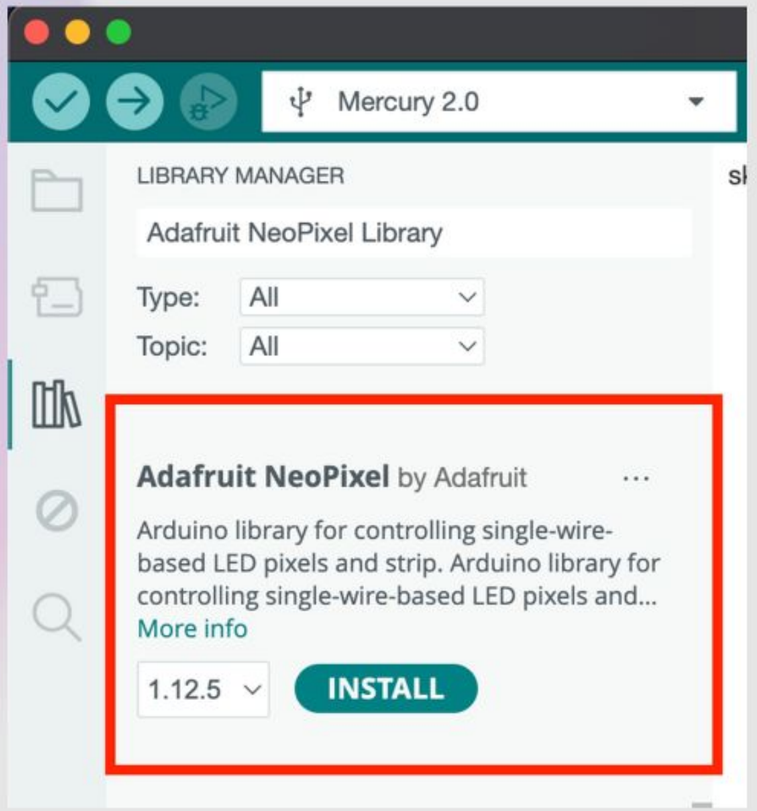
ARDUINO® LIBRARIES

ADAFRUIT_NEOPIXEL LIBRARY

ADAFRUIT_NEOPIXEL

HOW TO INSTALL A LIBRARY?





CHAPTER 15

PLAYING WITH RGB LED

RGB PULSE

```
// *************************
//
// Robotics AI Lab
//
// *************************

// RGB Pulse
// This code displays a RGB Pulse with onboard WS2812 RGB LED.
// Using the Adafruit NeoPixel Library, different LED colors are produced.

// Electronics used
// - Mercury Development Board

#include <Adafruit_NeoPixel.h>

#define LED_PIN RGB
#define NUM_PIXELS 1  // Number of LEDs

Adafruit_NeoPixel pixels(NUM_PIXELS, LED_PIN, NEO_GRB + NEO_KHZ800);

void setup() {
  pixels.begin();  // Initialize the LED
}

void loop() {
  pixels.setPixelColor(0, pixels.Color(255, 0, 0, 0));  // Red
  pixels.show();
  delay(1000);

  pixels.setPixelColor(0, pixels.Color(0, 255, 0, 0));  // Green
  pixels.show();
  delay(1000);

  pixels.setPixelColor(0, pixels.Color(0, 0, 255, 0));  // Blue
  pixels.show();
  delay(1000);
}
```

RGB PULSE

DEEP DIVE INTO THE CODE

#include <Adafruit_NeoPixel.h>
This line imports the Adafruit NeoPixel library, which contains all the functions we need to control the RGB LED on Mercury Board.

#define LED_PIN RGB
LED_PIN is defined as RGB_LED which is a predefined constant in the hardware setup of the Mercury Board, referring to the onboard RGB LED pin.

#define NUM_PIXELS 1
This means you're controlling a single NeoPixel LED. If you had a strip of, say, 32 LEDs, this would be 32.

RGB PULSE

DEEP DIVE INTO THE CODE

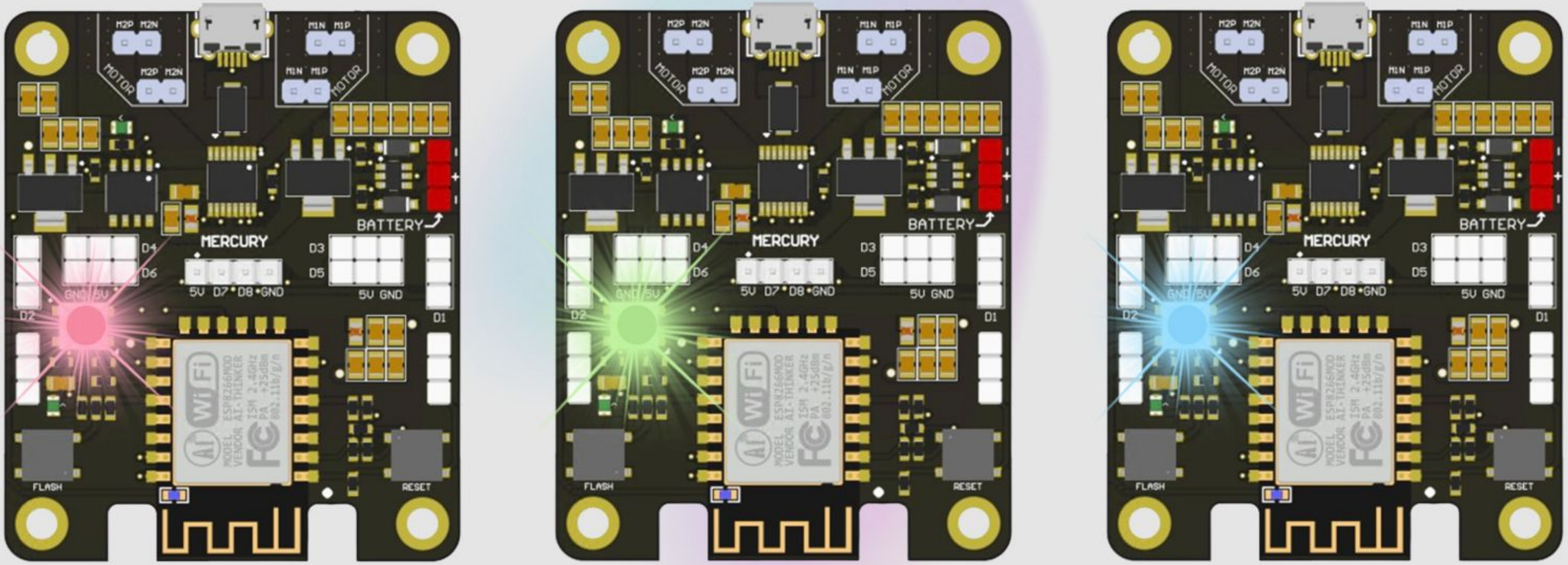
Adafruit_NeoPixel pixels(NUM_PIXELS, LED_PIN, NEO_GRB + NEO_KHZ800);
Here we are creating a "NeoPixel Object". This object, named pixels, represents our LED setup.
* `**NUM_PIXELS**`: The number of LEDs in your strip or ring (1 in this case).
* `**LED_PIN**`: The Arduino pin connected to the data line of the NeoPixel LEDs.
* `**NEO_GRB + NEO_KHZ800**`: This specifies the color order (Green, Red, Blue) and clock speed

pixels.begin();
Initializes the NeoPixel library and gets the LED ready to receive color data.

pixels.Color(255, 0, 0)
Color scheme (R, G, B)
- values for R, G, B can vary between [0, 255]. Changing these values will generate different colors.
- Red light → pixels.Color(255, 0, 0)
- Green light → pixels.Color(0, 255, 0)
- Blue light → pixels.Color(0, 0, 255)

pixels.show();
This function will actually glow the LED.

RGB PULSE


M2P M2N
M1N M1P
MOTOR
BATTERY
MERCURY
D4
D6
D3
D5
5V D7 D8 GND
5V GND
GND 5V
D2
D1
FLASH
RESET

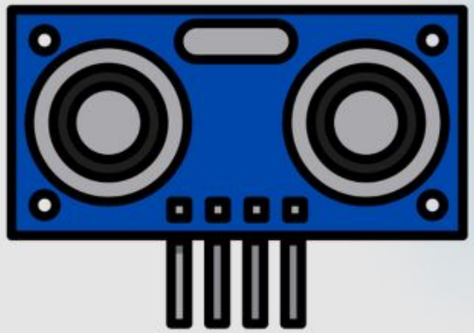
CHAPTER 16

INTRODUCTION TO SENSORS - INPUT DEVICES

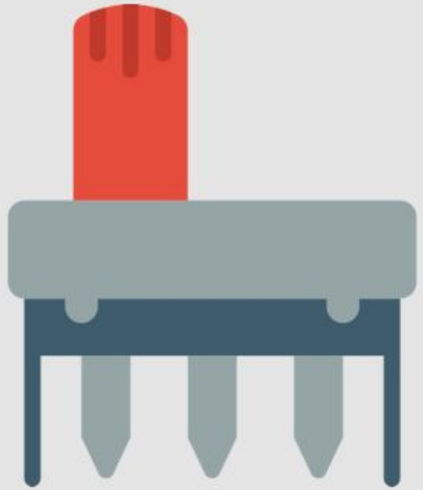
INPUT DEVICES

WHAT ARE INPUT DEVICES?

- Components or devices that send information or signal to the microcontroller are considered to be input devices



Sensor
(Ultrasonic)



Switch
(Toggle)

HUMAN SENSES

WHAT IS A SENSE?

A sense is a system used by organisms/machines to gather information about the surroundings and send it to the brain/microcontroller.

EXAMPLES OF HUMAN SENSES

5 main human senses:

- VISION → Eyes act as the sensors
- HEARING → Ears act as sensors
- SMELLING → Nose acts as sensor
- TASTING → Tongue acts as sensor
- TOUCHING → Skin acts as sensor

NOW IMAGINE...

- Your eyes help you see- A robot uses Camera to see
- Your ear helps you hear- A robot uses Microphone to hear
- Your hands help you feel- A robot uses a Touch/Force sensor

Sensors are like **superpowers for the Robot** to understand what's happening around them.

SENSORS

WHAT DO SENSORS DO?

To summarize, sensors take informations from the world around them like - light, sound, distance, movement, etc. - and send it to the brain of the robot (called the Microcontroller).

EXAMPLES OF SENSORS

Thermometer
temperature - sensor

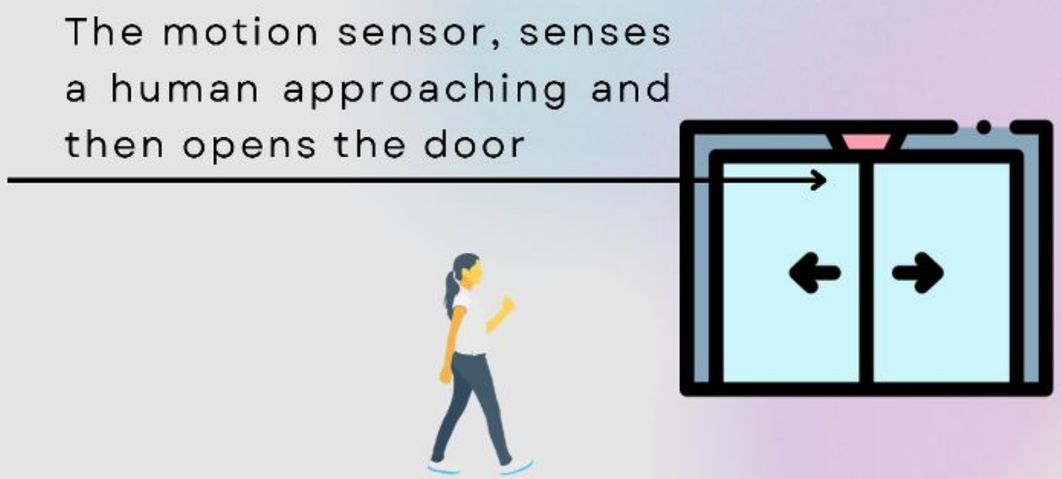
Microphone
sound - sensor

SENSORS WORKING IN REAL LIFE

AUTOMATIC DOOR

Ever wondered how the doors of malls and stores automatically open when you approach them without any human interference.

Its because these doors have motion and distance sensors installed in them



SENSORS WORKING IN REAL LIFE

TOUCH SCREEN - ALONG WITH FINGER SCANNING

Our Smartphones, Tablets, ATM machines, Station ticket machines are all installed with interactive touch screen that take our response.

Modern touch sensors
are capacitive in nature

CHAPTER 17

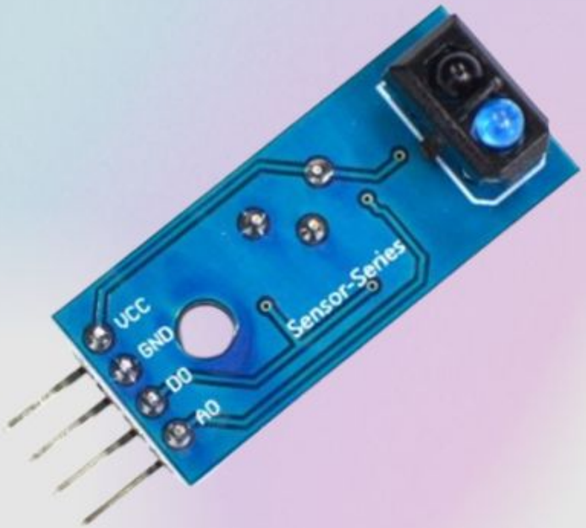
INTRODUCTION TO IR SENSORS

IR SENSORS

INTRODUCTION

Imagine if your robot had secret spy eyes👀

An IR (Infrared) sensor is an electronic device that detects objects or obstacles using infrared light. It's like a pair of invisible eyes for your robot!

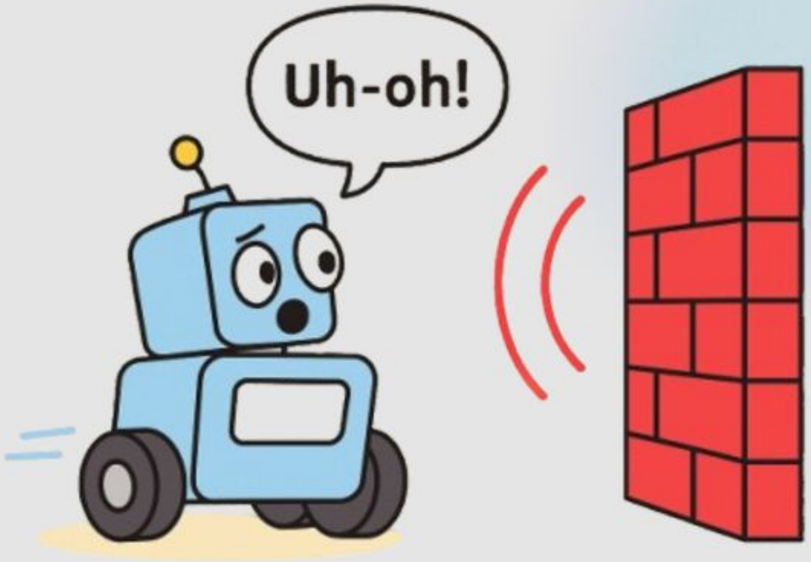


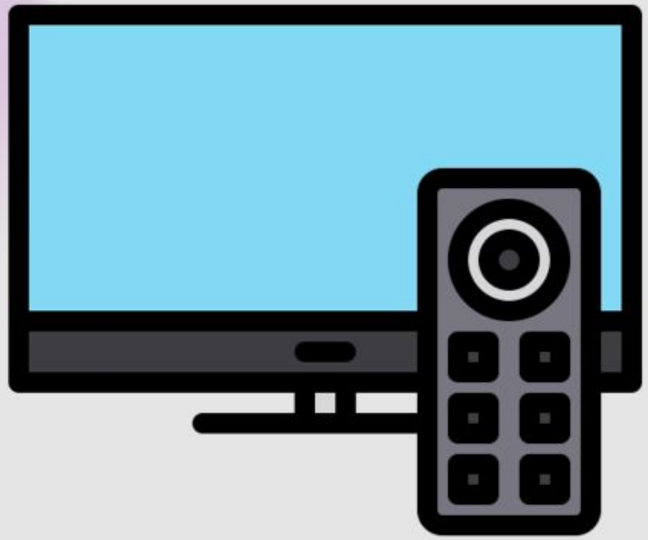
INVISIBLE SUPERPOWERS

MEET THE IR SENSOR

A robot toy that never crashes into walls...It just stops when something is in the way.
Or have you noticed how your TV changes channels when you press the remote?

That's **IR technology** in action!



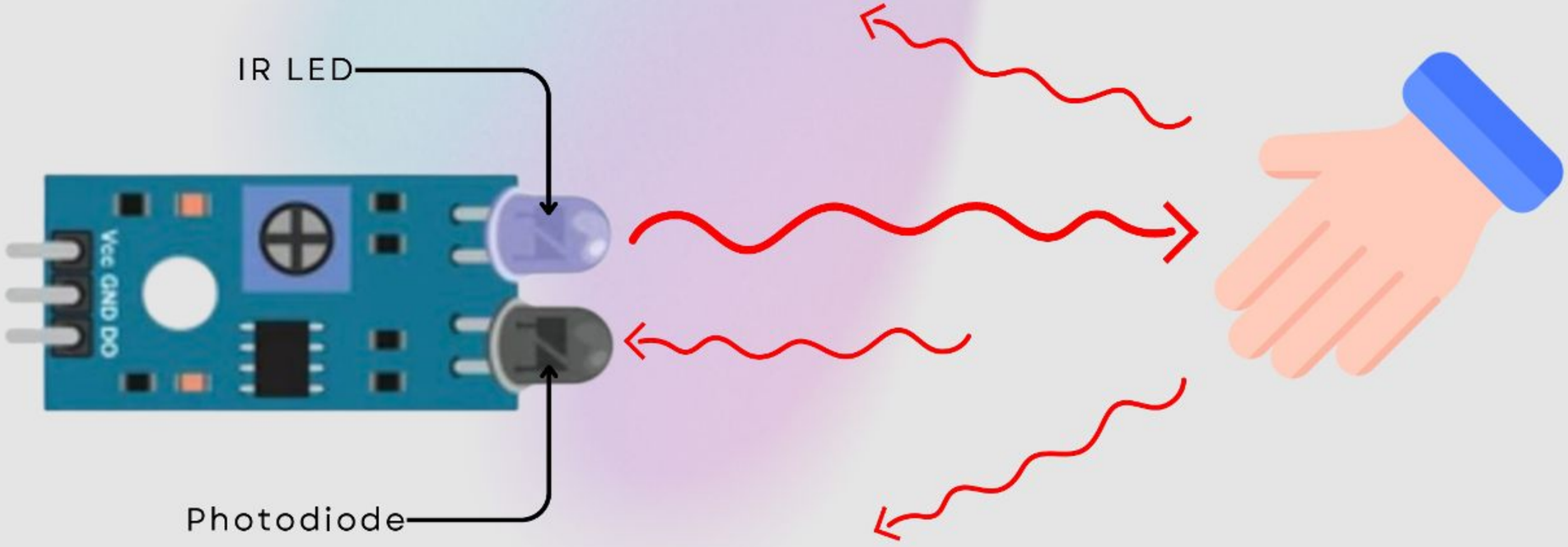


HOW DOES AN IR SENSOR WORK?

PEEK INTO THE MAGIC OF IR TECHNOLOGY

- IR LED emits invisible infrared (IR) light waves
- These IR waves bounce back after hitting a nearby object
- The photodiode receives some of these reflected waves

And the robot feels, "Aha! Something's there."



THE INVISIBLE LIGHT TRICK

WANT TO TRY THE INVISIBLE TRICK RIGHT NOW?

- Take remote control, TV, AC, etc.
- Turn ON your phone camera and face it towards the remote
- Press any button on the remote
- BOOM 💥 you will see a reddish light flash on your mobile screen

Here your remote is the emitter and your camera acts as the receiver

CHAPTER 18

ARDUINO® CODING - DIGITAL READ

DIGITAL READ

WHAT IS A DIGITALREAD()?

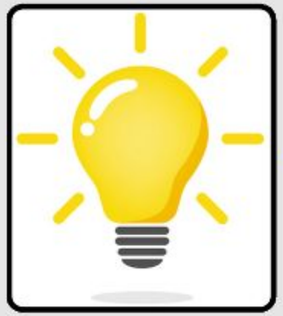
digitalRead(pin);
DigitalRead is a function that reads the value from a specified digital pin.

Think of digitalRead as the **process of observing** if the light in your room is ON or OFF.

If the pin is set as **INPUT with pinMode**, then reading the value of pin is like asking the microcontroller board if the light is ON or OFF?

If the light is ON, the microcontroller will return a value **HIGH or '1'** and similarly if the light is OFF, the microcontroller will return a value **LOW or '0'.**

DIGITAL READ

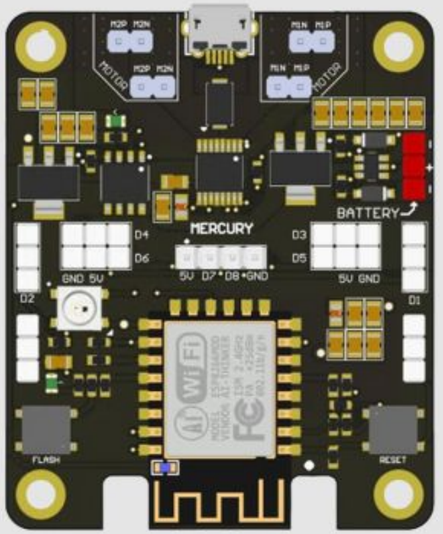


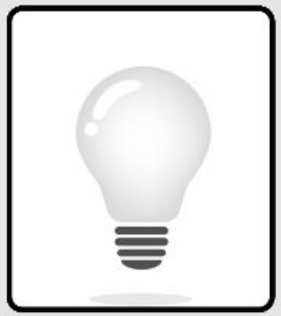
Light ON

LIGHT SENSOR

digitalRead(LIGHT_SENSOR_PIN)

= HIGH (ON)



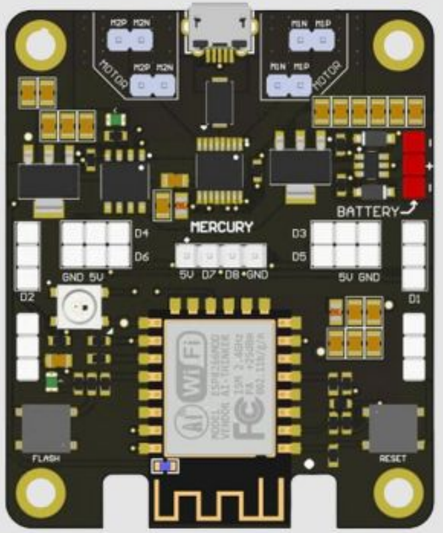


Light OFF

LIGHT SENSOR

digitalRead(LIGHT_SENSOR_PIN)

= LOW (OFF)



CHAPTER 19

SENSITIVITY VS RESOLUTION OF SENSOR

SENSITIVITY

HOW TO UNDERSTAND SENSOR SENSITIVITY?

Think of a sensor's sensitivity like a nose smelling cookies.

A very sensitive nose can smell cookies baking almost immediately, while a less sensitive nose might take much longer to notice the smell.

In the same way, sensitivity means how quickly and easily a sensor can detect what it is supposed to sense.

Person A
Less Sensitive Nose

Smells Nothing

Person B
More Sensitive Nose

Smells the Cookie
His mouth water🤤

RESOLUTION

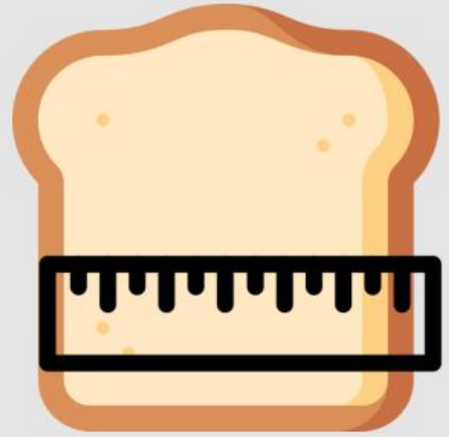
HOW TO UNDERSTAND SENSOR RESOLUTION?

Resolution is how detailed the sensor's measurement is.

You have a piece of bread that is 98mm in width

- Say, you have a **Scale A** with least measurement unit of **1 cm**
- You have **Scale B** with least measurement unit of **1 mm**
- We know **1cm = 10mm**
- If you measure the bread with **Scale A**, we will say the length of the bread is **between 9cm and 10cm.** But we will not be able to tell precisely how much.
- If you measure the bread with **Scale B**, you will be able to tell that the bread is **exactly 98mm**.

Scale A: Low Resolution



Scale B: High Resolution

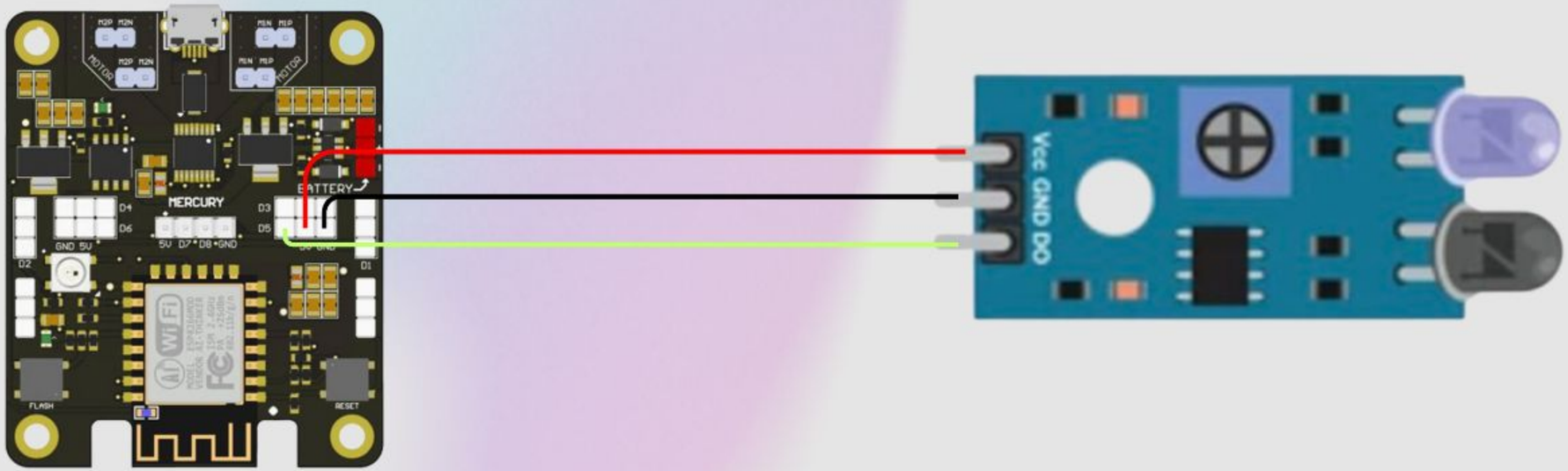
CHAPTER 20

INTERFACING IR SENSOR WITH MERCURY BOARD

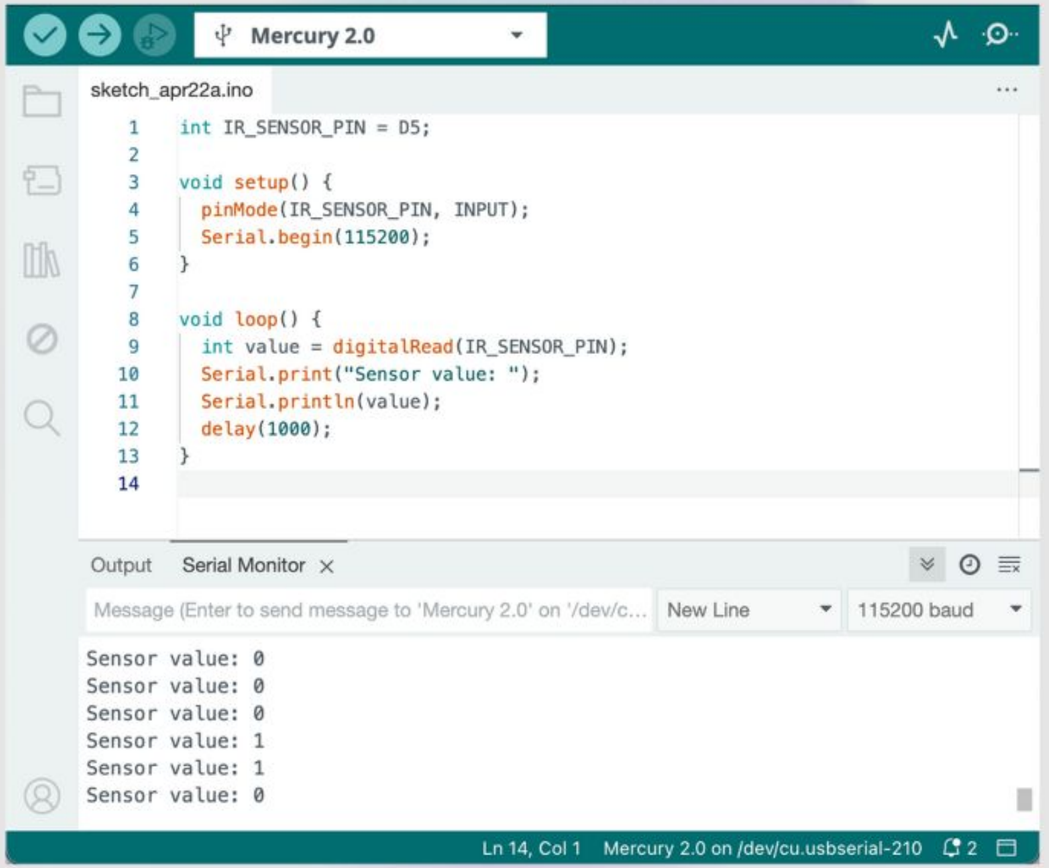
IR SENSOR TO MERCURY BOARD CONNECTION

IR CONNECTIONS

- VCC → 5V on the board
- GND → GND on the board
- D0 → D5 pin on the board



LET'S GET THE IR SENSOR WORKING



WHAT'S THAT CODE DOING?

DEEP DIVE INTO THE CODE

int IR_SENSOR_PIN = D5
This indicated that we will connecting the signal pin or IR sensor to D5 pin

pinMode(IR_SENSOR_PIN, INPUT)
This line tells the mercury board that it should listen to or "keep an eye" at this signal pin

digitalRead(IR_SENSOR_PIN)
This will read the status on the signal pin of IR sensor.

Object detected → Signal pin value = 1;
Object not detected → Signal pin value = 0;

SENSOR SENSITIVITY

This is the moment you set the sensor sensitivity properly such that it detect an object at a set distance and when the object is removed, the sensor doesn't detect it anymore.

CHAPTER 21

INTRODUCTION TO BUZZER

MEET THE BUZZER

LET'S MAKE SOME NOISE!

Have you ever heard a toy beep? Or an alarm go off? Well, that's the buzzer doing its job.

A buzzer is an electronic part that makes sound - usually a beep, buzz, or alarm tone. Some buzzers can play different tones, others just ON/OFF beeps.

THEN WHAT IS A SPEAKER?

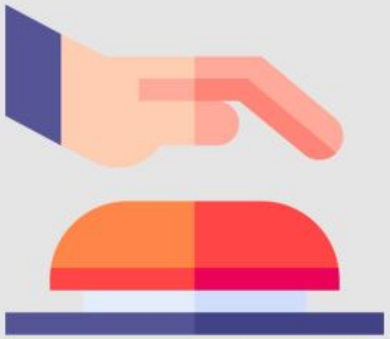
- A buzzer turns electricity into vibration to make one fixed sound.
- A speaker also uses vibration but can change how fast it vibrates, so it can play different pitches and tones – even music!

BUZZERS APPLICATION

WHERE DO WE USE BUZZERS?

Super sounds in everyday Life, buzzers are all around us!

- Alarm clocks
- Fire alarms
- Washing machines
- Reverse car sensors
- Quiz buzzers



SCHOOL BELL

WHY USE A BUZZERS?

Just like how a school bell rings to tell students it's time for lunch or class. A buzzer is used to inform the user about an event that is happening.

- Robot's way of saying: "Time to act!"
- Alerts when a task is done, a problem is detected, or an event starts.

"Think of it like a mini school bell inside your project!"

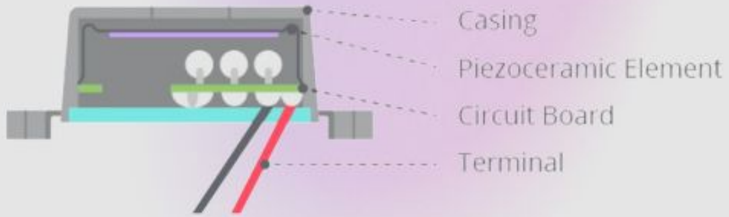
BUZZER OPERATION?

HOW DOES A TRADITIONAL BUZZER WORK?

- A buzzer contains a small metal disc
- Below which, there's a tiny electromagnet
- On energizing the electromagnet, it pulls the metal disc back and forth very rapidly
- This rapid movement causes the disc to vibrate
- Those vibrations are what produce sound

HOW DOES A PIEZO BUZZER WORK?

A piezo buzzer uses a special material called a piezoelectric crystal. When electricity is applied to this crystal, it changes shape very slightly – not enough for us to see, but enough to cause vibrations. This is an alternate to the electromagnet in the example above.



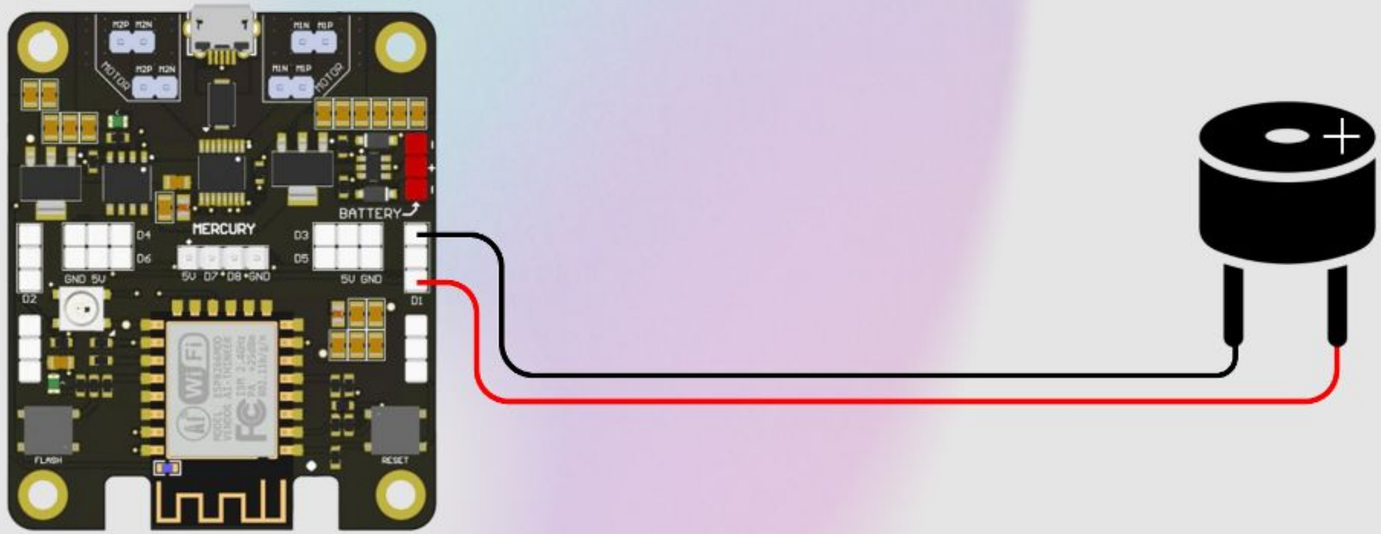
CHAPTER 22

INTERFACING BUZZER WITH MERCURY BOARD

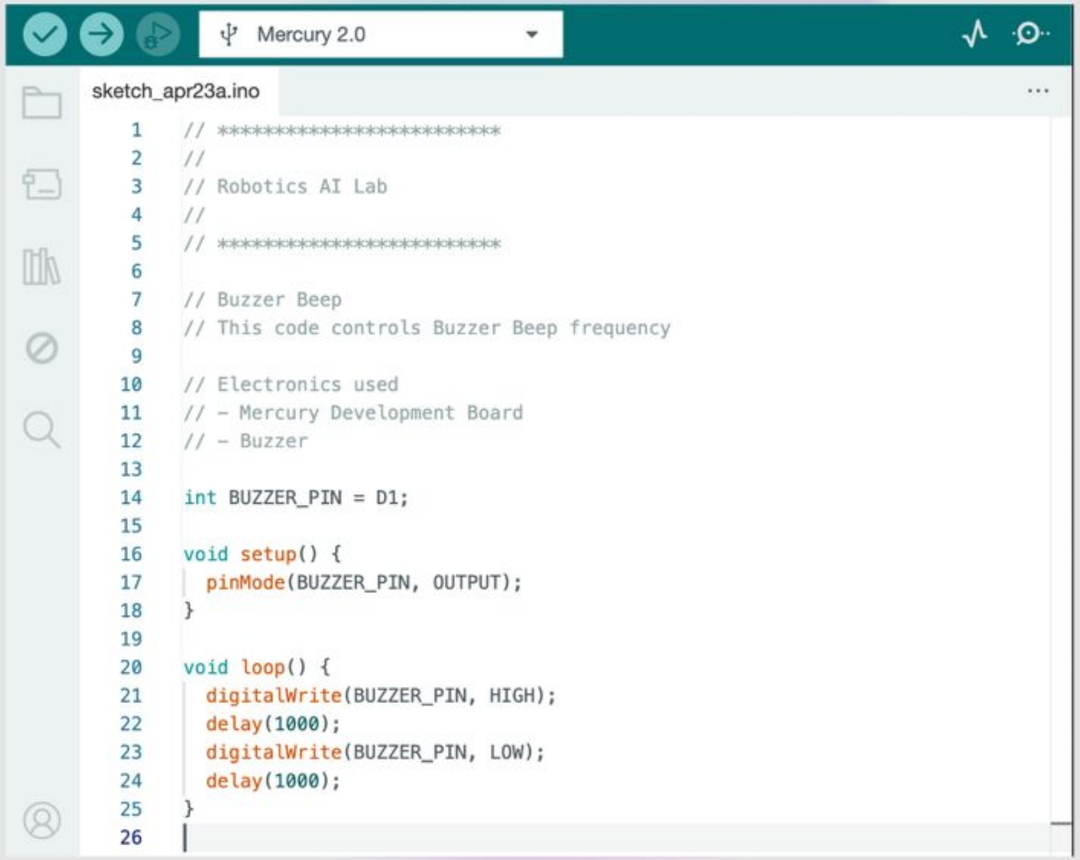
BUZZER TO MERCURY BOARD CONNECTION

BUZZER CONNECTIONS

- (+) 🔴 Positive terminal → D1 pin on the board
- (-) ⚫ Negative terminal → GND on the board



HELLO WORLD – BUZZER STYLE!



WHAT'S THAT CODE DOING?

DEEP DIVE INTO THE CODE

int BUZZER_PIN = D1
This indicated that we will connecting the buzzer (+) positive pin to D1 pin

pinMode(BUZZER_PIN, OUTPUT)
This line tells the mercury board that it should control this pin HIGH or LOW as per code.

digitalWrite(BUZZER_PIN, value)
This will provide 3.3V or 0V to this pin.

3.3V supplied → Buzzer beeps;
0V supplied → Buzzer remains quiet;

CHAPTER 23

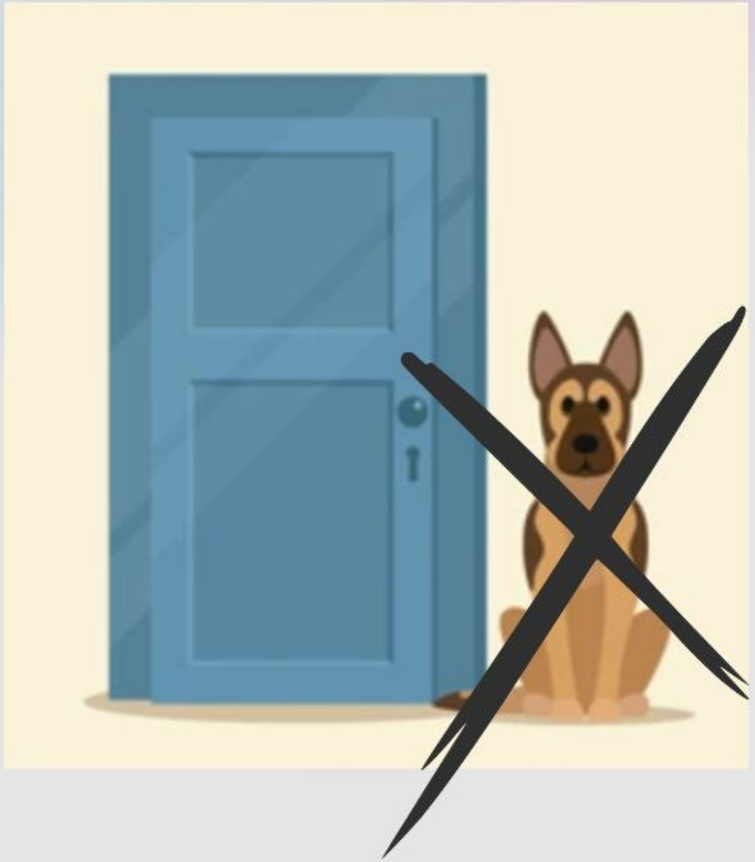
BUILDING A BURGLAR ALARM SYSTEM

BUILD A BURGLAR ALARM!

WHAT IS A BURGLAR ALARM?

A burglar alarm can simply be defined as a security system that makes a loud sound or sends an alert when someone tries to enter a place without permission.

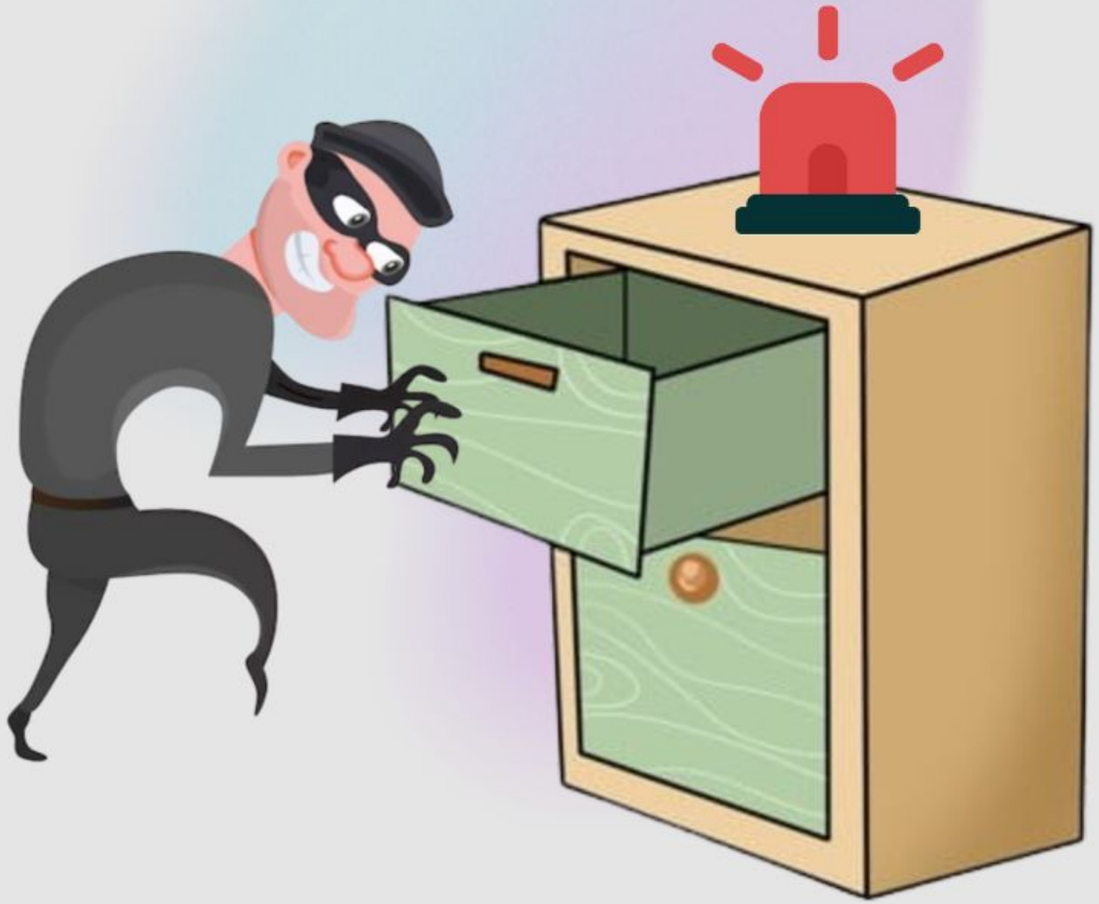
Keep the intruders away... But with tech!



REAL-WORLD HOOK

LET'S DRAW A REAL WORLD SCENARIO?

Someone sneaking into your room and opens your drawer. The intruder was unaware of the fact, that a safety switch needed to be pressed before opening the drawer, and the ALARM goes off!



HOW WILL IT WORK?

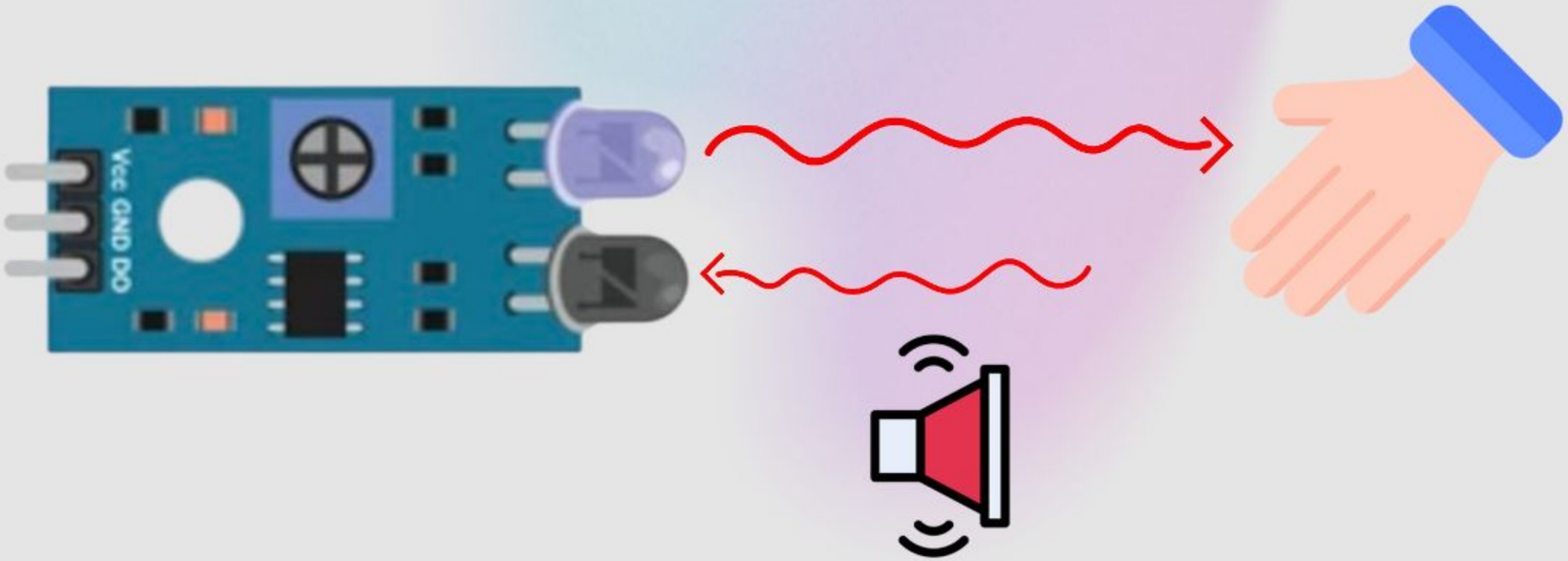
LET'S BUILD A SIMPLE PROTOTYPE

Problem statement

Detect an object and beep the buzzer.

Components

- IR sensor
- Buzzer
- Mercury Board



MAKING THE CIRCUIT

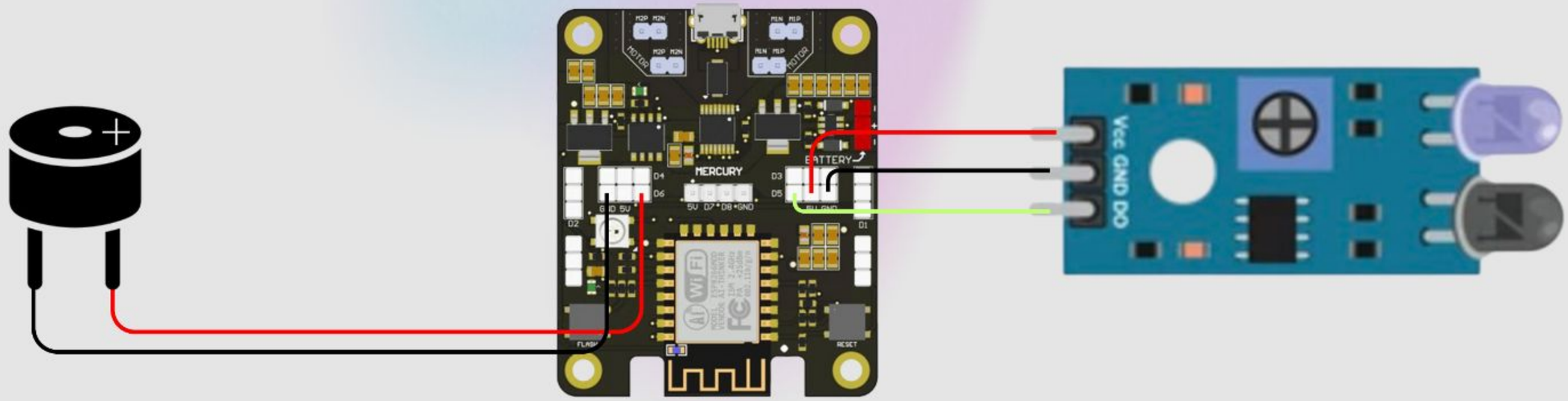
CONNECTIONS

BUZZER CONNECTIONS

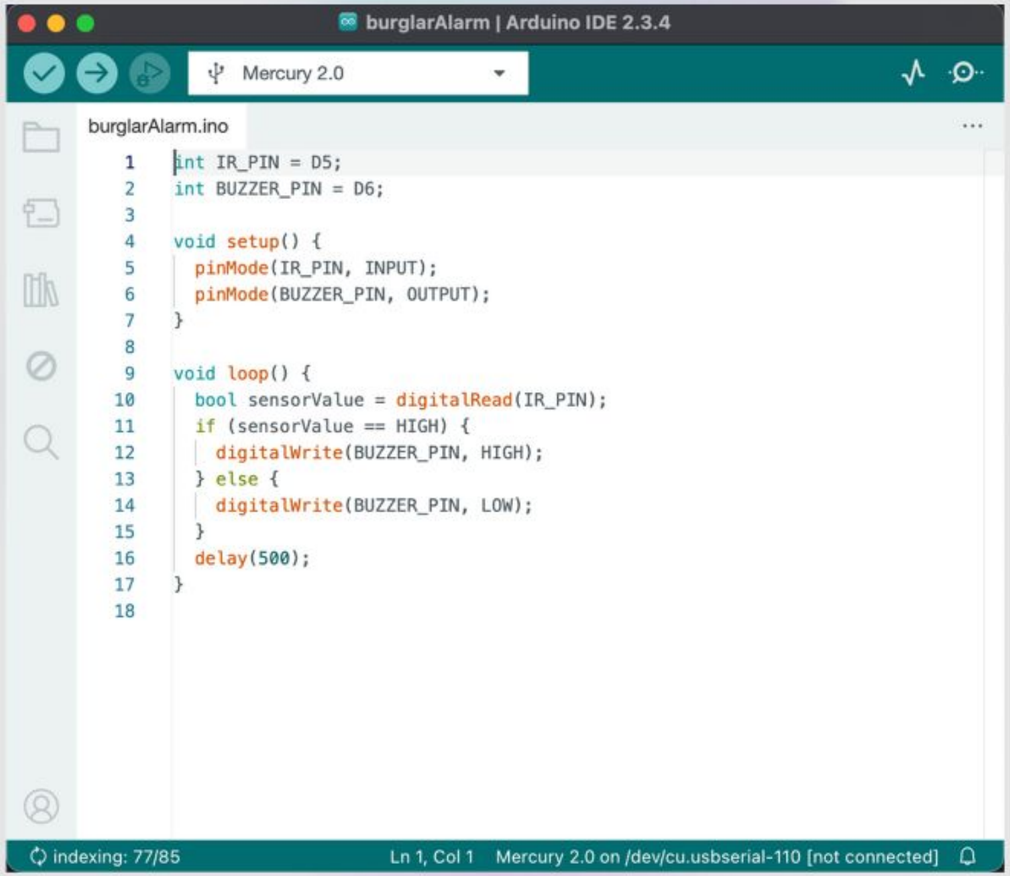
- (+) 🔴 Positive terminal → D6 pin on the board
- (-) ⚫ Negative terminal → GND on the board

IR connections

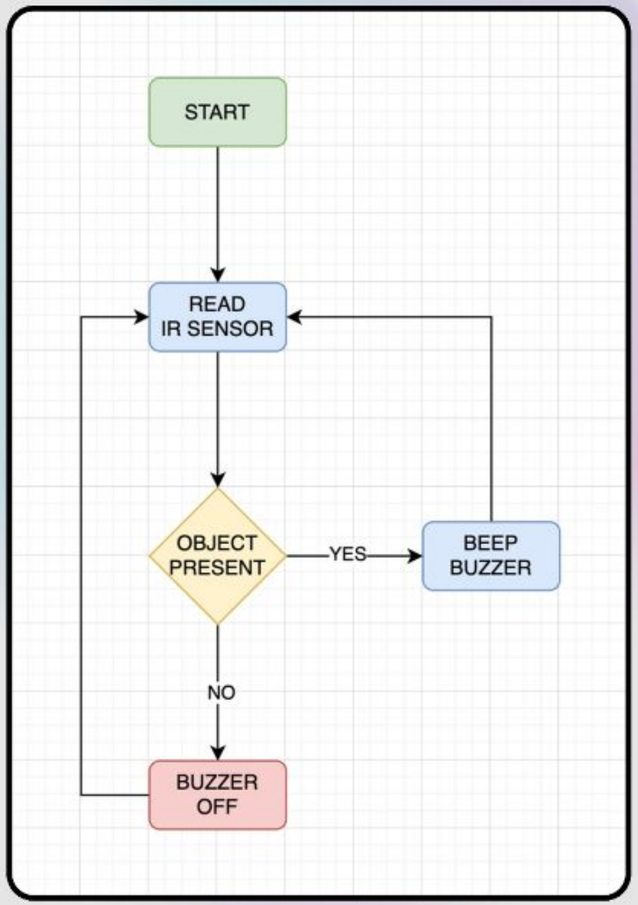
- (+) 🔴 VCC pin → 5V pin on the board
- (-) ⚫ GND pin → GND on the board
- (D0) 🟢 Signal pin → D5 pin on the board



LET'S CODE IT!


burglarAlarm | Arduino IDE 2.3.4
Mercury 2.0
burglarAlarm.ino
int IR_PIN = D5;
int BUZZER_PIN = D6;
void setup() {
pinMode(IR_PIN, INPUT);
pinMode(BUZZER_PIN, OUTPUT);
}
void loop() {
bool sensorValue = digitalRead(IR_PIN);
if (sensorValue == HIGH) {
digitalWrite(BUZZER_PIN, HIGH);
} else {
digitalWrite(BUZZER_PIN, LOW);
}
delay(500);
}
indexing: 77/85
Ln 1, Col 1
Mercury 2.0 on /dev/cu.usbserial-110 [not connected]

CODE FLOW STRUCTURE



WHAT'S THAT CODE DOING?

DEEP DIVE INTO THE CODE - IF <-> ELSE

IF - ELSE CONDITION

- Read the IR Sensor value
- If the object is detected → beep the buzzer
- If the object is NOT detected → the buzzer remains OFF or is turned OFF if it was ON
- as there is a 500 milli-seconds delay, the condition is checked 2 times every second

```
9   void loop() {
10    bool sensorValue = digitalRead(IR_PIN);
11    if (sensorValue == HIGH) {
12      digitalWrite(BUZZER_PIN, HIGH);
13    } else {
14      digitalWrite(BUZZER_PIN, LOW);
15    }
16    delay(500);
17  }
```

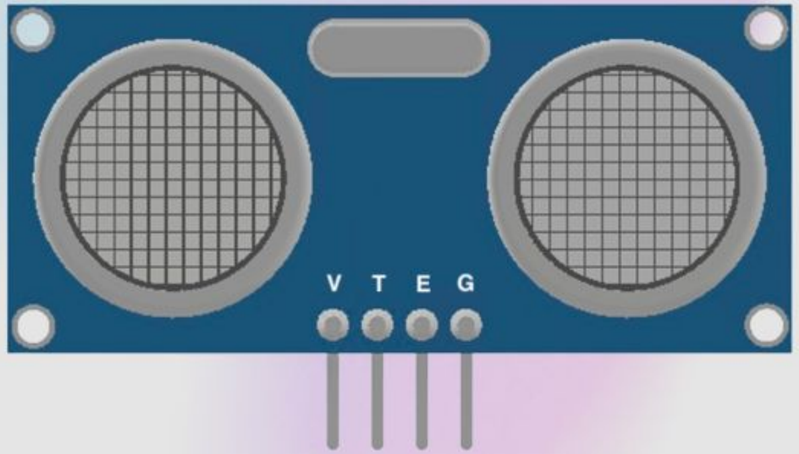
CHAPTER 24

ULTRASONIC SENSOR

ULTRASONIC SENSOR

WHAT IS AN ULTRASONIC SENSOR?

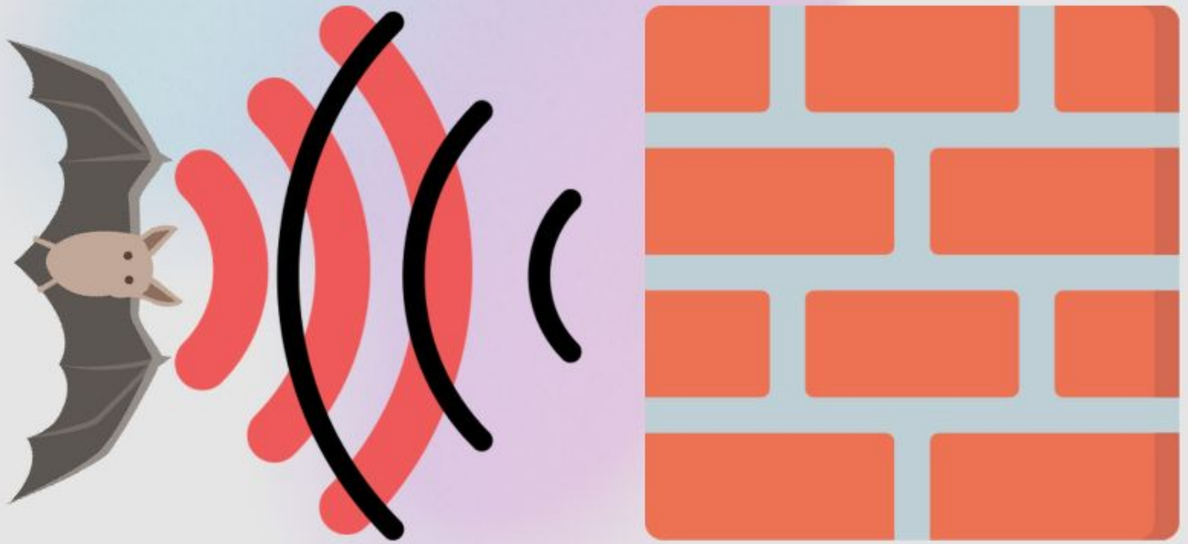
An ultrasonic sensor is a device that **measures distance using sound waves**. But not just any sound – it uses ultrasonic waves, which are sound waves that are too high-pitched for humans to hear.



ULTRASONIC WAVE

WHAT DOES A BAT DO TO FLY SAFELY?

Bats use a natural version of this technology called **echolocation**. They make high-frequency sounds and listen to the echoes to “see” in the dark. This helps them fly safely and catch insects without bumping into things.



ULTRASONIC SENSOR

HOW DOES ULTRASONIC SENSOR FUNCTION?

The sensor has two main parts:

- A transmitter - that sends out an ultrasonic sound wave (like a tiny shout!).
- A receiver - that listens for the echo when the sound bounces off an object.

By measuring the time it takes for the sound to go out and come back, the sensor can calculate how far away the object is – just like how we use echoes to judge distance in a cave.

ULTRASONIC SENSOR

APPLICATION

These are some of the areas in which ultrasonic sensors are widely used:

- 🚗 Obstacle detection in robots and self-driving cars
- 🏠 Smart water tanks to check water levels
- 📏 Measuring distances without touching the object
- 🔐 Security systems to detect movement
- 🛠️ Industrial automation for detecting object presence

Ultrasonic sensors are safe, reliable, and work well even in low light or dark areas, making them ideal for robotics and automation projects.

CHAPTER 25

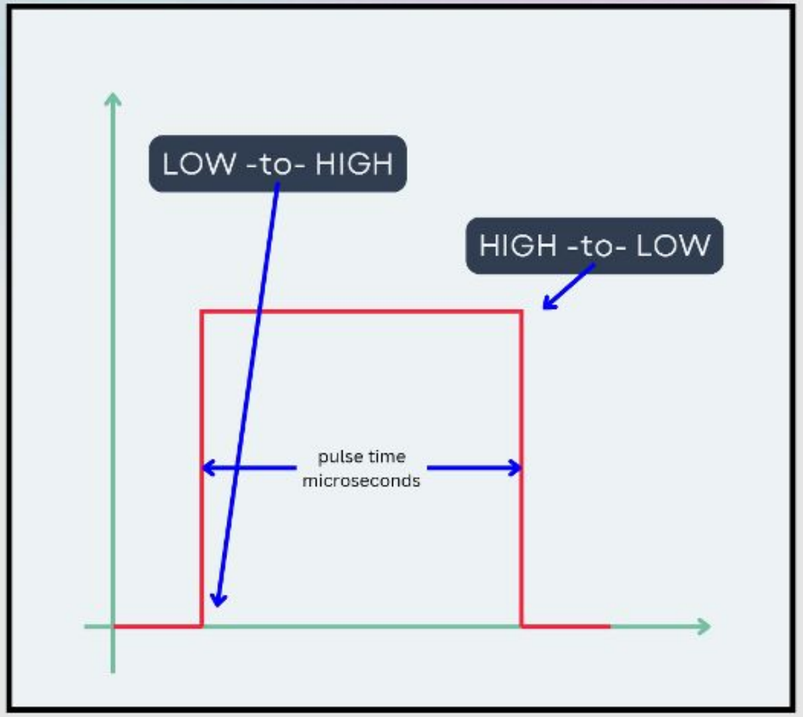
ARDUINO® CODING - PULSE IN

PULSE IN

WHAT IS DOES PULSEIN()?

Defined as: **pulseIn(pin, value)**

Reads a pulse (either HIGH or LOW) on a pin. For example, if value is HIGH, pulseIn() waits for the pin to go from LOW to HIGH, starts timing, then waits for the pin to go LOW and stops timing. Returns the length of the pulse in microseconds.



CHAPTER 26

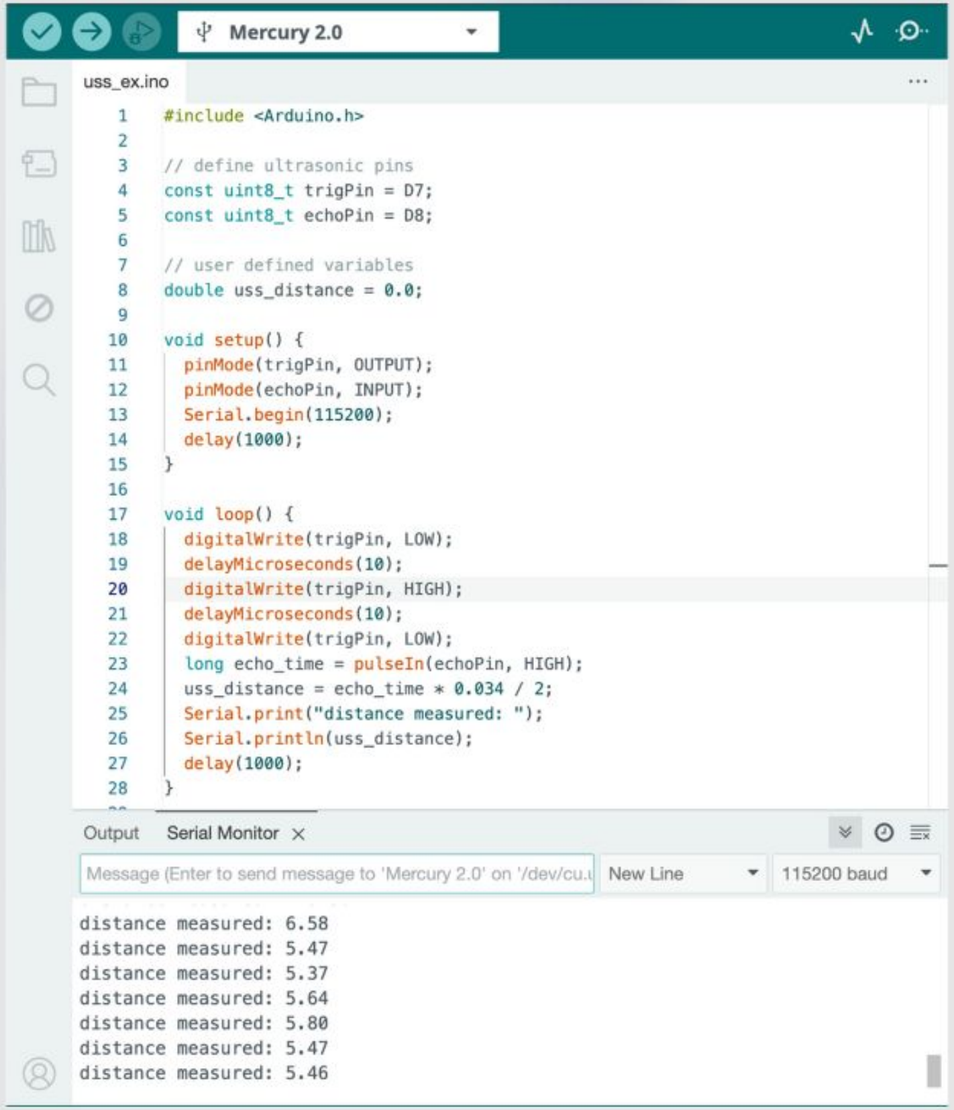
INTERFACING ULTRASONIC SENSOR WITH MERCURY BOARD

USS TO MERCURY BOARD CONNECTION

USS CONNECTIONS

- (+) 🔴 VCC → 5V pin on the board
- (T) 🟢 Trig → D7 pin on the board
- (E) 🔵 Echo → D8 pin on the board
- (-) ⚫ Gnd → GND on the board

DISTANCE MEASUREMENT CODE



WHAT'S THAT CODE DOING?

DEEP DIVE INTO THE CODE

const uint8_t trigPin = D7;
const uint8_t echoPin = D8;
These lines tell the Mercury Board that pin D7 will be utilized as trigger pin and pin D8 as echo.

pinMode(trigPin, OUTPUT);
pinMode(echoPin, INPUT);
These line tells the Mercury Board that trigPin is set as output pin and echoPin as input.

delayMicroseconds(10);
Similar to delay() function, which halts the code for given milliseconds, delayMicroseconds() halts the code for given microseconds.

1 microsecond = 1second / 1,000,000

WHAT'S THAT CODE DOING?

DEEP DIVE INTO THE CODE

TRIGGER is used to generate and emit the ultrasonic waves and **ECHO** is used to capture the bounced waves.

We start by setting the Trig pin **LOW** for 10 microseconds to make sure there's no accidental signal being sent.

Next, a short 10-microsecond **HIGH** pulse from the Trig pin – like giving the sensor a quick "shout" to emit a sound wave.

Now we wait for the echo to return.

- pulseIn() listens on the Echo pin.
- It measures how long the pin stays HIGH (i.e., how long it took for the echo to come back).

This value (**echo_time**) is the **duration of the round trip** (out and back)

```
18    digitalWrite(trigPin, LOW);
19    delayMicroseconds(10);
20    digitalWrite(trigPin, HIGH);
21    delayMicroseconds(10);
22    digitalWrite(trigPin, LOW);
23    long echo_time = pulseIn(echoPin, HIGH);
```

WHAT'S THAT CODE DOING?

DEEP DIVE INTO THE CODE

uss_distance = echo_time * 0.034 / 2;

As we know speed = distance/time => **distance = speed x time**

distance measured = [speed of sound] x echo_time

Generally, under normal room conditions, the speed of sound in air is approximately **340m/s**

1 second = 1,000,000 microseconds
1 meter = 100 centimeter

340 m/s => 340 meter / 1 seconds
340 meters/second = 340 x 100 centimeter / 1,000,000 microseconds
340 meters/second = 34~~0~~ x ~~100~~ centimeter / 1,000,~~000~~ microseconds
340 meters/second = 0.034 centimeter / microseconds

Now we know the value of **speed of sound** is 0.034, and we get **echo_time** value from the pulseIn().

WHAT'S THAT CODE DOING?

DEEP DIVE INTO THE CODE

uss_distance = echo_time * 0.034 / 2; => Why divide by 2?

If we observe carefully, we start the timer when the wave leaves TRIGGER module and travels some distance until it reaches the wall.

Now, it travels the same distance back to the ECHO module before we stop the timer. Thus the TOTAL-TIME captured by the microcontroller is for **two-times the distance travelled**.

So we are simply dividing the echo_time by two to get distance travelled in one direction.

CHAPTER 27

MOTORS (ACTUATORS)

WHAT IS A MOTOR?

WHAT IS A MOTOR?

A motor is a machine that spins when it gets electricity! The magic lies in the fact that this device has the capability to convert electrical energy to mechanical energy (kinetic or motion).

- Think of electricity as food 🥪
- A motor "eats" the electricity and gets the energy to move

WHERE WOULD YOU GENERALLY SEE A MOTOR?

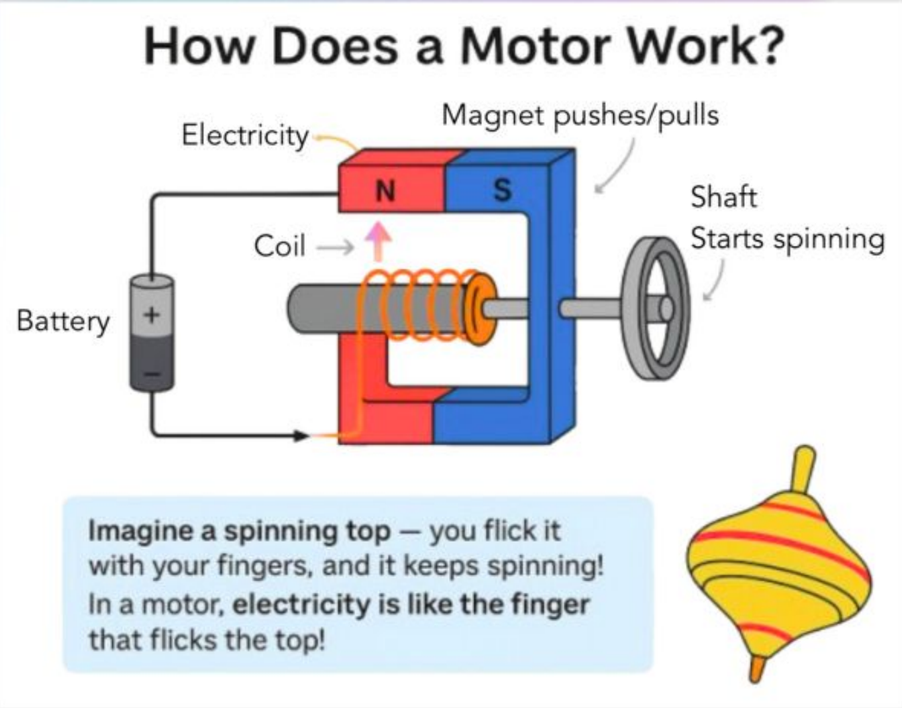
- 🌬️ Fans spinning to keep you cool
- 🚗 Toy cars
- 🎮 Toys moving around
- 🤖 Robots doing cool tasks
- 🛴 Scooters zipping down the street
- Vacuum cleaner

WORKING OF A MOTOR

A PEEK INDISE THIS DEVICE?

It's worth highlighting three major components of a motor

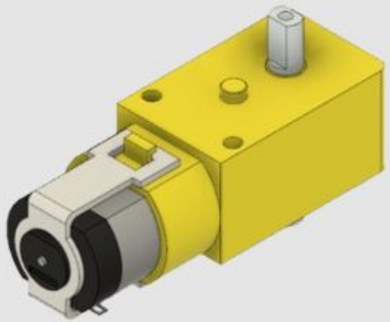
- Permanent Magnets
- Coil - when energized, creates magnetic field - which in turn gets repelled by the permanent magnets
- Shaft - rotates due to this repulsion force and delivers the required mechanical energy

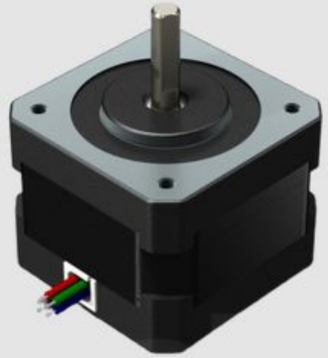


TYPES OF MOTORS

MOTORS RELEVANT FOR ROBOTICS

- DC Motor - **continuous rotation**
- Servo Motor - great for achieving **precise angles** (limited motor)
- Stepper Motor - **continuous rotation** and capability to rotate tiny steps to achieve **precise angles**

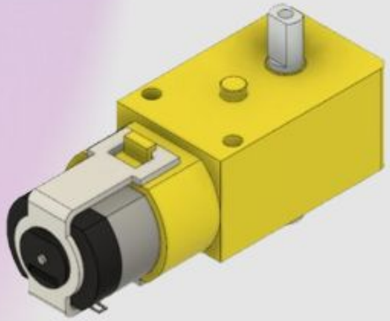




DC MOTOR

OUR YELLOW MOTOR

- Simple DC Motor but with a gear system in the yellow housing
- What does the gear do?
  - Well, it reduces the rotation speed and gets more torque out of it



SERVO MOTOR

WHAT IS SO SPECIAL ABOUT A SERVO MOTOR?

A Servo Motor rotates to a specific angle – not just round and round like a fan.

- High quality Servo motors can move precisely to the angle you tell it – like 21°, 90°, or 180°!
- This being said, servo motor has limited angular motion unlike normal DC motor, generally between 0° to 180°
- A servo motor is like your elbow or knee joint - you can bend it exactly to the angle you need!

WHERE DO WE USE SERVO MOTORS IN ROBOTICS?

- Robotic Arms (to pick up objects)
- Camera Gimbals (to keep cameras steady)
- Mini Airplanes (to move wings and rudders)

Servos listen very carefully to the commands it receives to only move as much as needed – just like a robot doing yoga!

STEPPER MOTOR

NOW LET'S LOOK AT STEPPER MOTOR

A Stepper motors move in tiny steps instead of spinning continuously – 1°or even 0.5°! Unlike servo motor, Stepper motor has no limitations in as to how much it can rotate.

- They move a little… and stop.
- Then move a little more… and stop again.
- They can keep doing this until desired.

Stepper motor is like walking on stepping stones – one small step at a time to reach your goal, without rushing!

Stepper motor gives best of both world, unlimited rotation property of DC motor and achieving precise angle property of servo motor.

STEPPER MOTOR

WHERE DO WE USE STEPPER MOTORS?

- 3D Printers (to create perfect shapes layer by layer)
- Paper Printers (to move the print head precisely)
- CNC Machines (for cutting and carving)
- Camera Sliders (for smooth, steady movement)

Any place that needs precise and highly accurate movement, an engineer would prefer to use a stepper motor.

WHY DO ROBOTS LOVE MOTORS?

WHY USE MOTORS?

Without motors, robots would be like statues! Motors let the robots do following actions:

- Move wheels 🤖
- Wave arms 👋
- Rotate sensors 🔄

By learning about motors, you can build:

- 🤖 Robots
- 🚗 Smart Cars
- 🪄 Magic Machines

It all starts with understanding how to make things move!

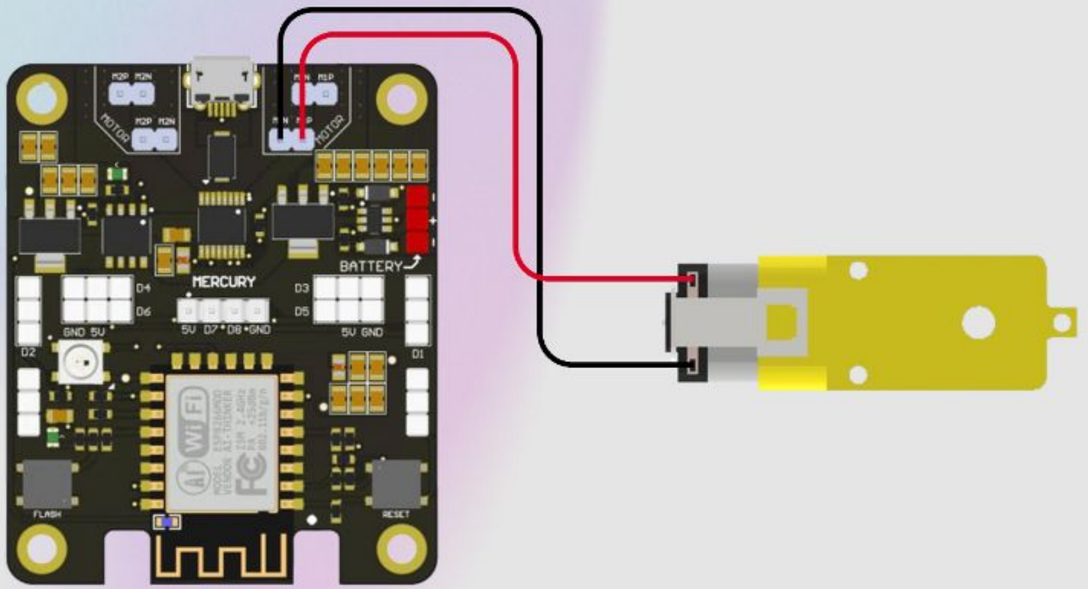
CHAPTER 28

INTERFACING DC MOTOR WITH MERCURY BOARD

MOTOR TO MERCURY BOARD CONNECTION

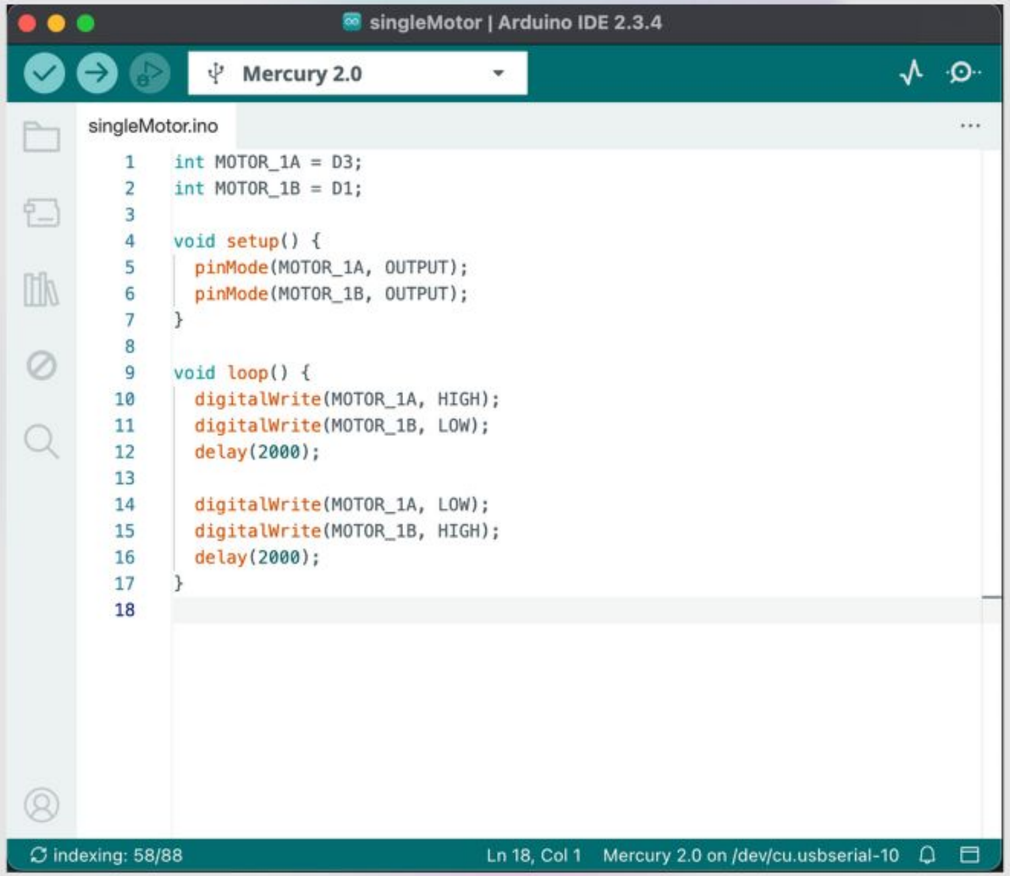
MOTOR CONNECTIONS

- (+) 🔴 Vcc → M1P pin on the board
- (-) ⚫ Gnd → M1N on the board



DRIVING A MOTOR

CODE OVERVIEW



WHAT'S THAT CODE DOING?

DEEP DIVE INTO THE CODE

int MOTOR_1A = D3;
int MOTOR_1B = D1;
These lines tell the Mercury Board that pin D3 and pin D1 will be utilized drive the motor and control its direction.

Special Note: Pin D3 and Pin D1 are not directly connected the motor as we have seen in the LED and other examples. Rather, these pins are talking to the motor driver, which in-turn drives the motor.

In theory you can directly drive the motor using these pin, but that will burn/damage the microcontroller as it will not be able to supply the power required to drive a motor.

pinMode(MOTOR_1A, OUTPUT);
pinMode(MOTOR_1B, OUTPUT);
These line tells the Mercury Board that MOTOR_1A and MOTOR_1B are set as output pins.

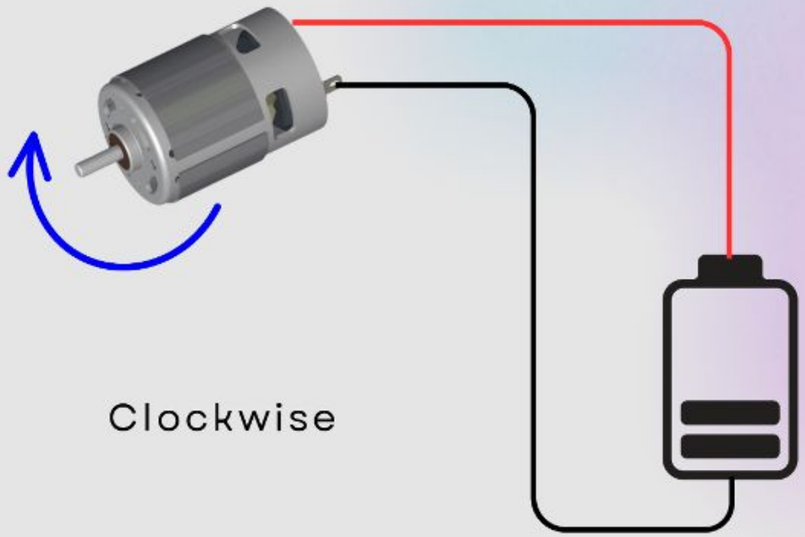
DC MOTOR

INTRODUCTION

- DC Motor simply keeps rotating when given power
- Want to spin it the other way, simply swap the wires

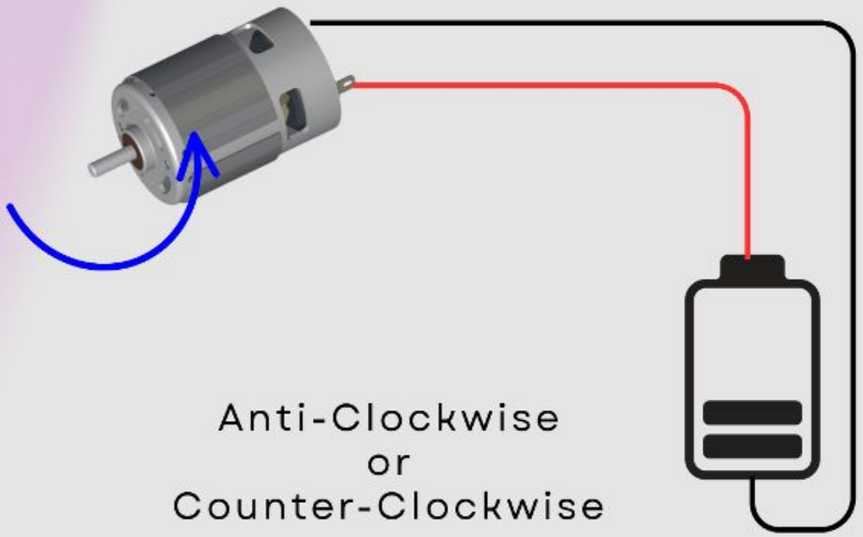
The direction of current flowing through the coil will determine the rotation direction

(+) 🔴 Vcc → M1P pin
(-) ⚫ Gnd → M1N pin



Clockwise

(+) 🔴 Vcc → M1N pin
(-) ⚫ Gnd → M1P pin



Anti-Clockwise
or
Counter-Clockwise

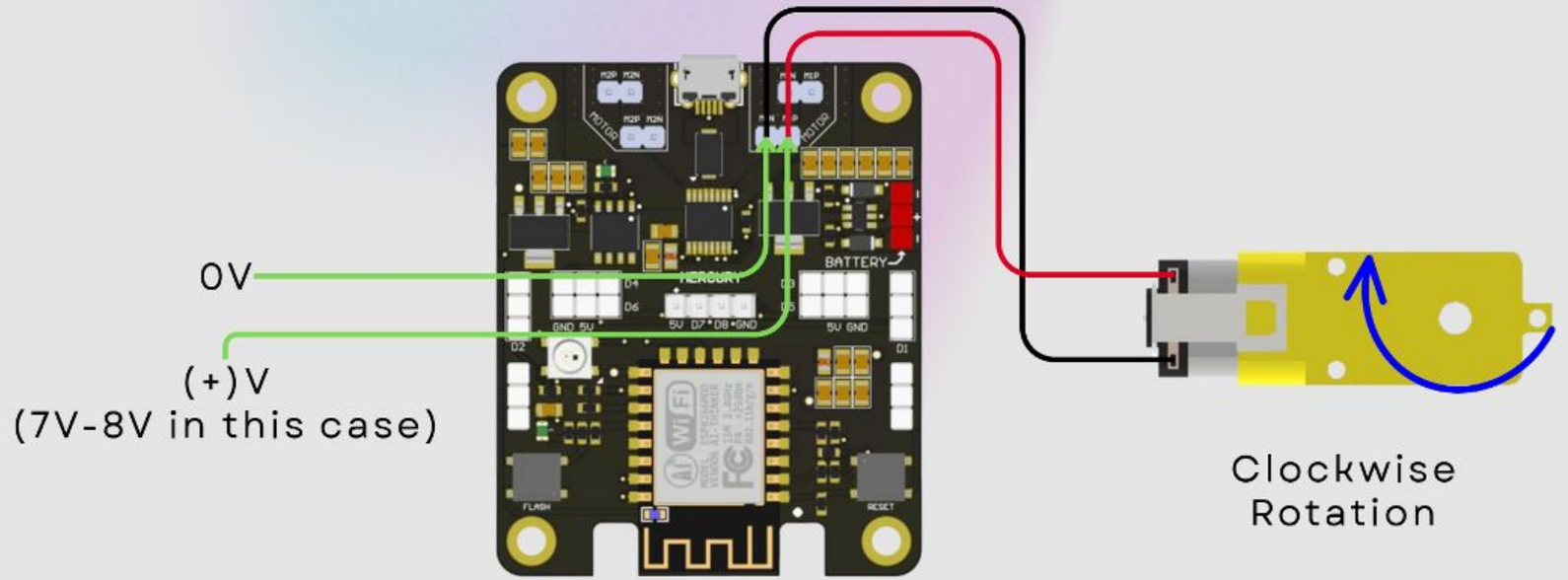
WHAT'S THAT CODE DOING?

DEEP DIVE INTO THE CODE

digitalWrite(MOTOR_1A, HIGH);
digitalWrite(MOTOR_1B, LOW);
Here with **MOTOR_1A → HIGH and MOTOR_1B → LOW:**

- the (+) positive terminal of the motor is getting (+) positive voltage
- the (-) terminal is getting 0V or says grounded.

Due to which the motor rotates is clock wise direction.



WHAT'S THAT CODE DOING?

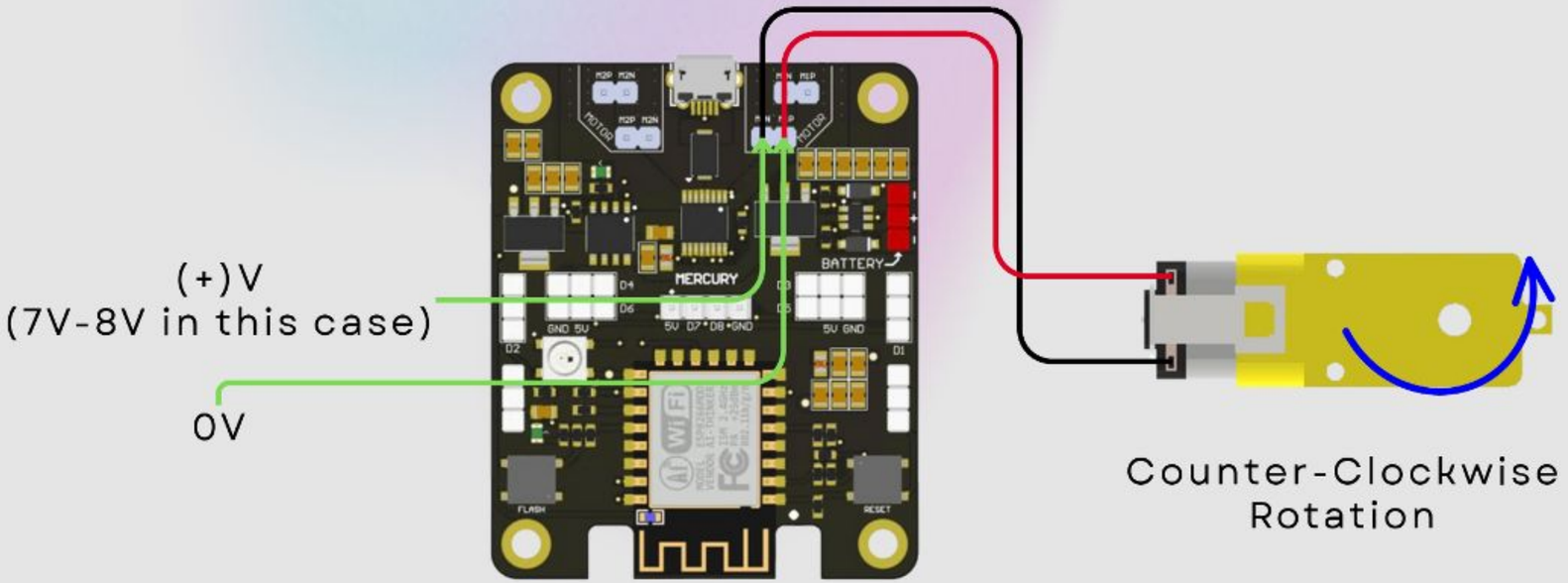
DEEP DIVE INTO THE CODE

digitalWrite(MOTOR_1A, LOW);
digitalWrite(MOTOR_1B, HIGH);

In contrast to the previous situation, here with **MOTOR_1A → LOW and MOTOR_1B → HIGH:**

- the (+) positive terminal of the motor is getting 0V or says grounded whereas
- the (-) terminal is getting (+) positive voltage

Due to which the motor rotates is counter clock wise direction.



CHAPTER 29

FUNCTIONS

FUNCTIONS

WANT TO EAT MAGGI?

When you say, "***Mummy, please make Maggi for me!***" your mom already knows all the steps – boil water, add noodles, put masala, stir, and serve.

You don't tell her every single step correct? You just say "**please make Maggi**", and she does all the work behind the scenes.

In programming, a function is just like that. You give it a name (like **makeMaggi()**) and when you call it, the computer runs all the steps written inside.

FUNCTIONS

DEFINE MAKEMAGGI()

So whenever you want Maggi again, you don't repeat all the steps. You just call the makeMaggi() function!

```
void makeMaggi() {
  boilWater();
  addNoodles();
  addMasala();
  stirAndCook();
  serve();
}
```

FUNCTIONS

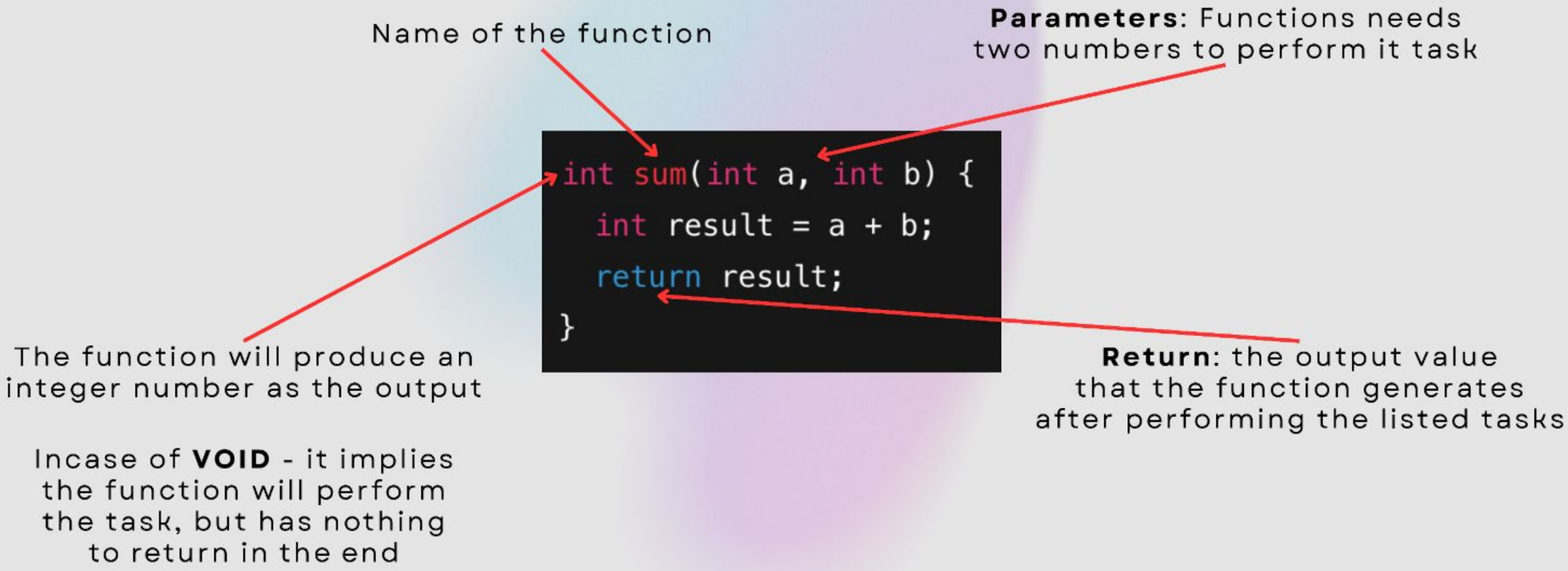
OTHER EXAMPLES ON FUNCTIONS

```
void washHands() {
  openTap();
  applySoap();
  scrubFor20Seconds();
  rinseHands();
  closeTap();
  dryHands();
}
```

```
void bedtimeRoutine() {
  brushTeeth();
  wearPajamas();
  sayGoodNight();
  goToSleep();
}
```

FUNCTIONS

UNDERSTANDING FUNCTION STRUCTURE



FUNCTIONS

LET'S TRY A SUM FUNCTION IN ACTION

The code is expecting two numbers as user input and then calculate and return the sum of them.

```
13  int num1, num2;
14
15  void setup() {
16    Serial.begin(115200);
17    delay(2000);
18    Serial.print("Enter first number: ");
19  }
20
21  void loop() {
22    if (Serial.available() > 0) {
23      num1 = Serial.parseInt();
24      Serial.println(num1);
25      Serial.print("Enter second number to add: ");
26      // if (Serial.read() == '\n') {}
27      while (!Serial.available()) {}
28      num2 = Serial.parseInt();
29      Serial.println(num2);
30      int total = sum(num1, num2);
31      Serial.print("Sum of given numbers is: ");
32      Serial.println(total);
33      Serial.println("");
34      Serial.print("Enter first number: ");
35      // if (Serial.read() == '\n') {}
36    }
37  }
38
39  int sum(int a, int b) {
40    return a + b;
41  }
42
```

Output Serial Monitor ×

Message (Enter to send message to 'Mercury 3.0' on '/dev/c... No Line Ending 115200 baud

```
Enter first number: 6
Enter second number to add: 9
Sum of given numbers is: 15

Enter first number:
```

Ln 43, Col 1 Mercury 3.0 on /dev/cu.usbserial-110 2

Select
"No Line Ending"

CHAPTER 30

BUILDING THE SCOUT ROBOT

CAR ASSEMBLY

WHY DID WE NAME IS “SCOUT”?

A **scout** or a **guide** is someone who goes ahead of the group to explore unfamiliar territory, ensuring the path is safe and suitable before the rest of the team follows and that’s what this smart car is intended to do.

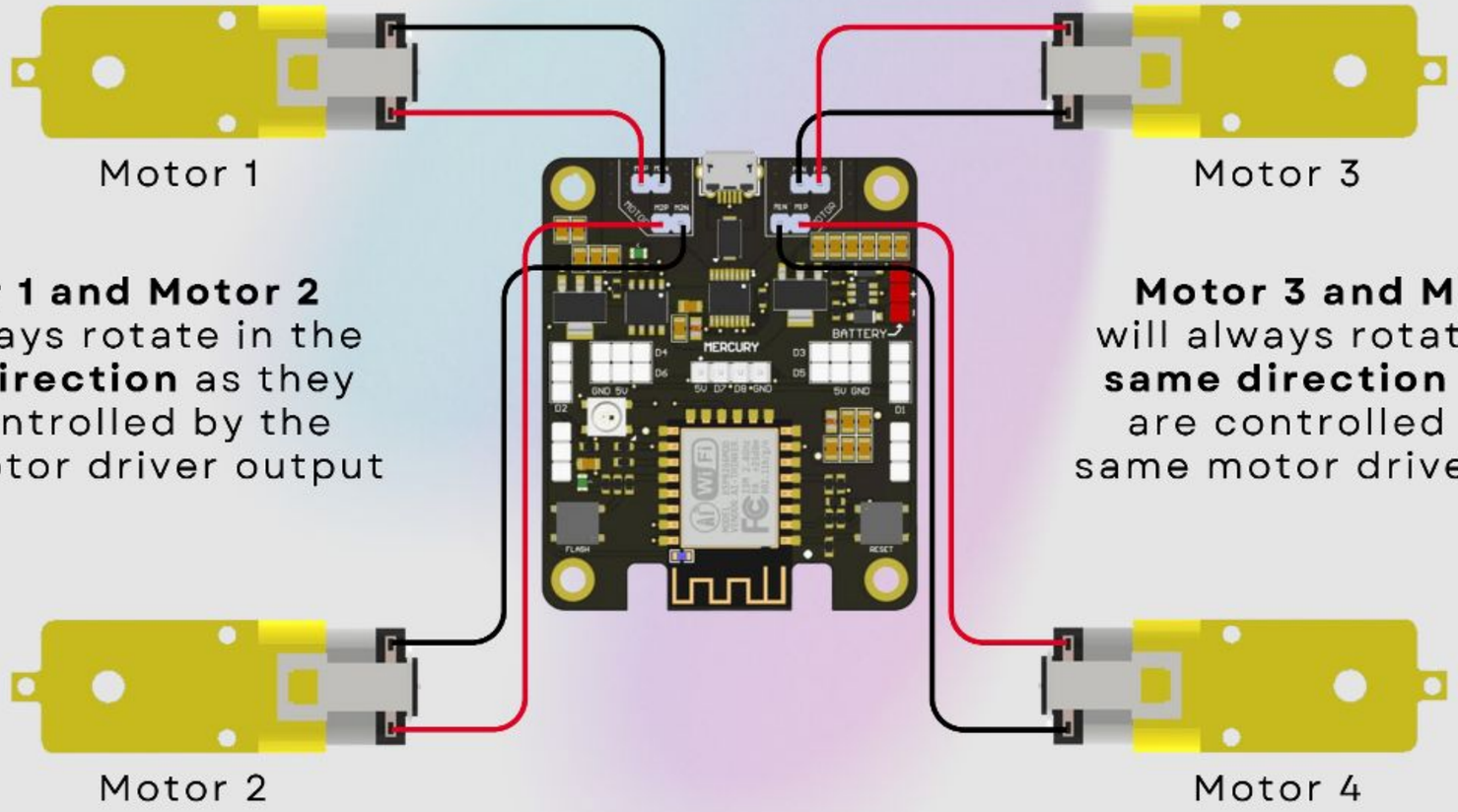
CAR ASSEMBLY

ASSEMBLY LINK

https://youtu.be/oy9Aq4PUm5M

CAR ASSEMBLY

MOTOR CONNECTIONS



Motor 1 and Motor 2 will always rotate in the **same direction** as they are controlled by the same motor driver output

Motor 3 and Motor 4 will always rotate in the **same direction** as they are controlled by the same motor driver output

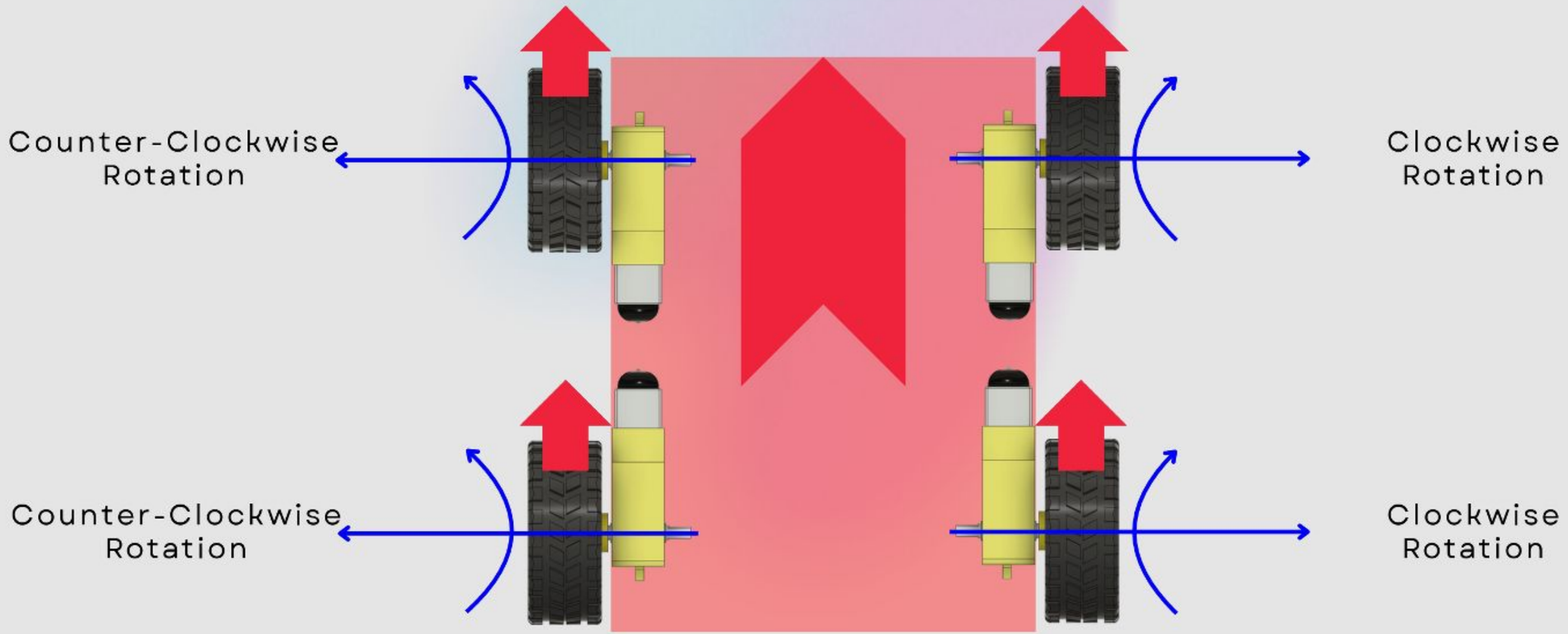
CHAPTER 31

CAR STEERING

FORWARD

MOTOR ROTATION FOR FORWARD MOTION

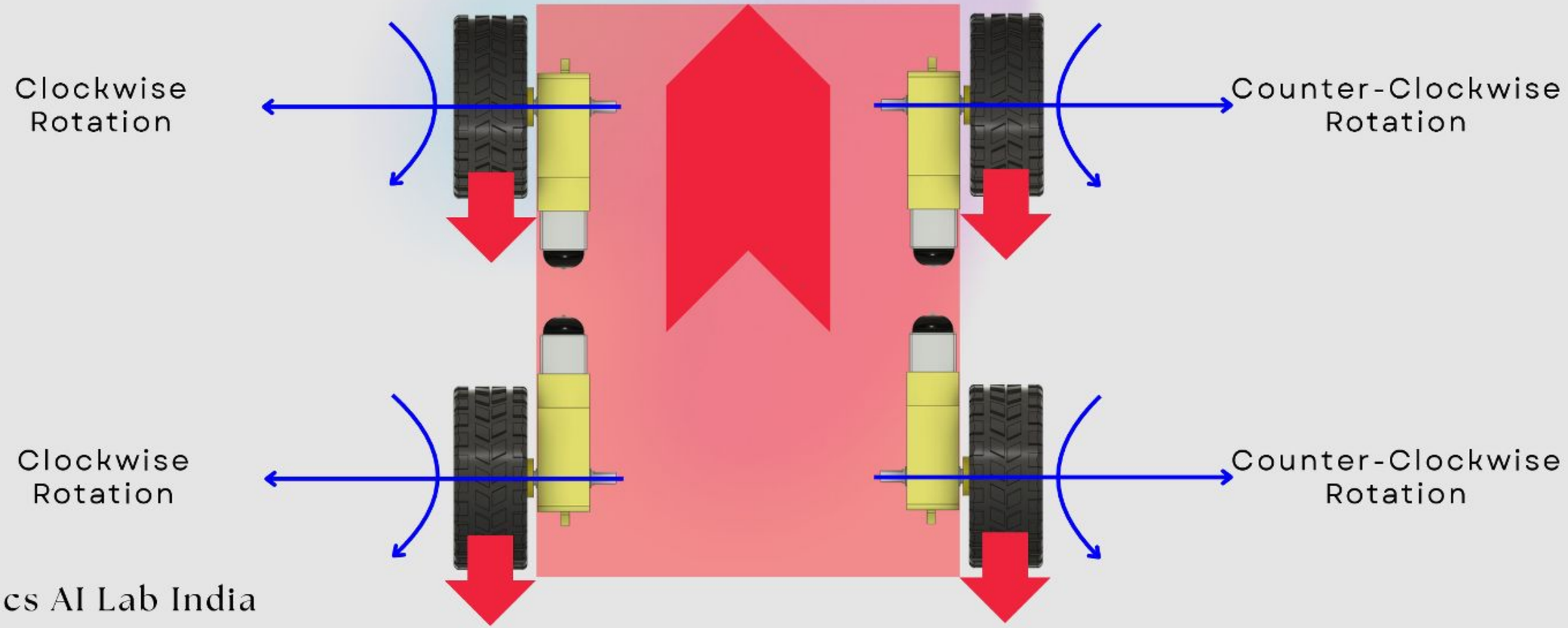
Let's look at individual motor rotation direction: -



REVERSE (BACKWARD)

MOTOR ROTATION FOR REVERSE MOTION

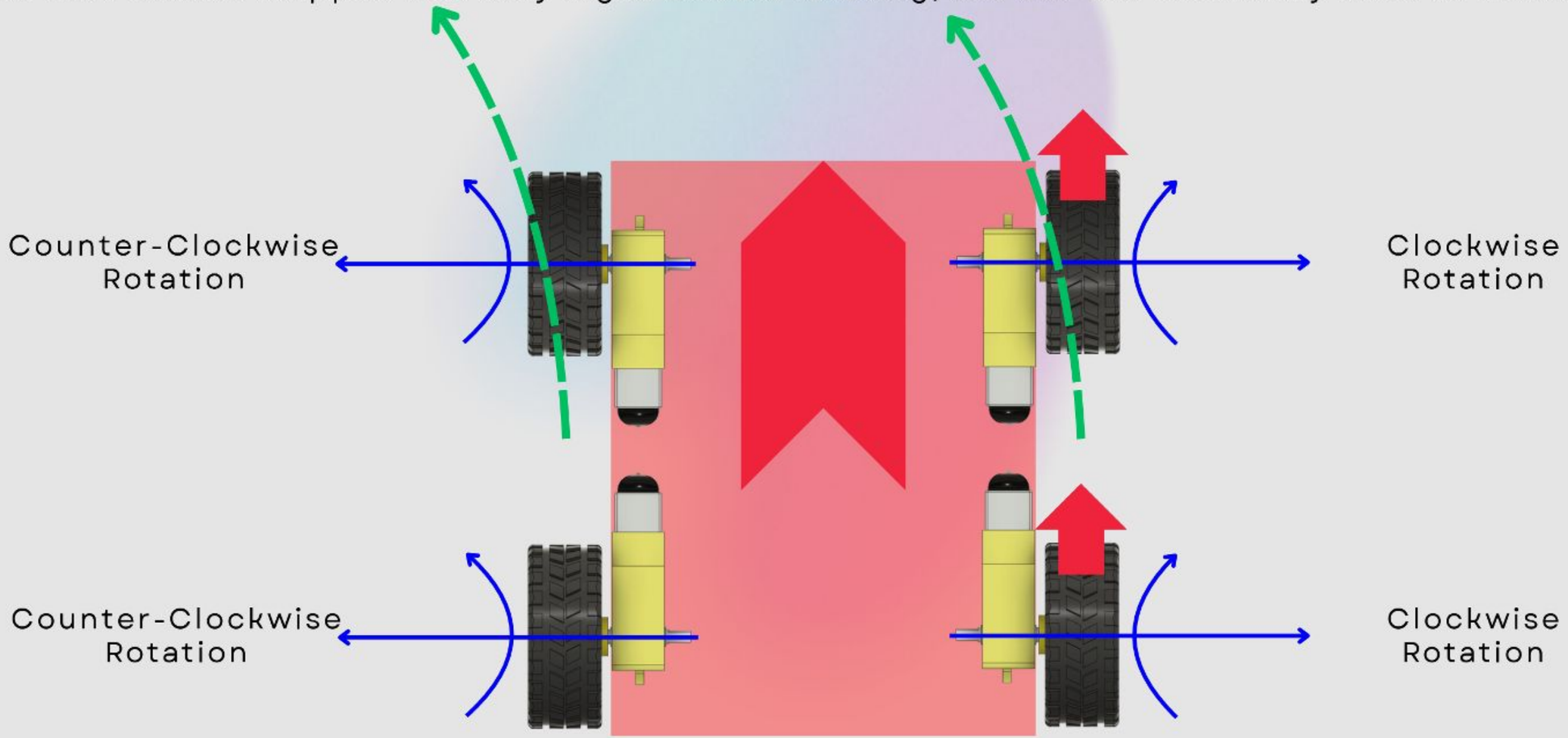
Let's look at individual motor rotation direction: -



LEFT - TURN

MOTOR ROTATION FOR LEFT MOTION

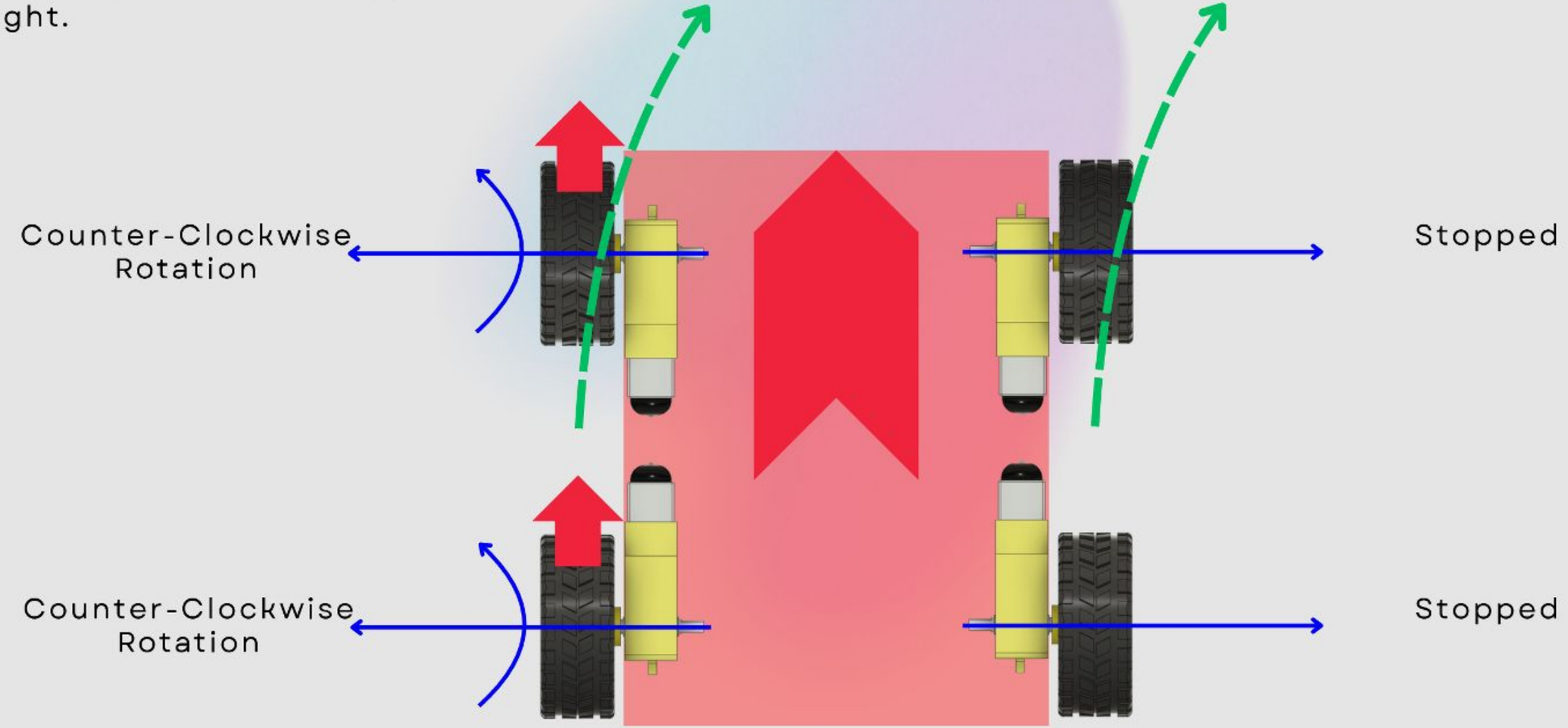
With Left wheels stopped and only Right wheels rotating, the car will inherently start to rotate left.



RIGHT - TURN

MOTOR ROTATION FOR RIGHT MOTION

With Right wheels stopped and only Left wheels rotating, the car will inherently start to rotate Right.

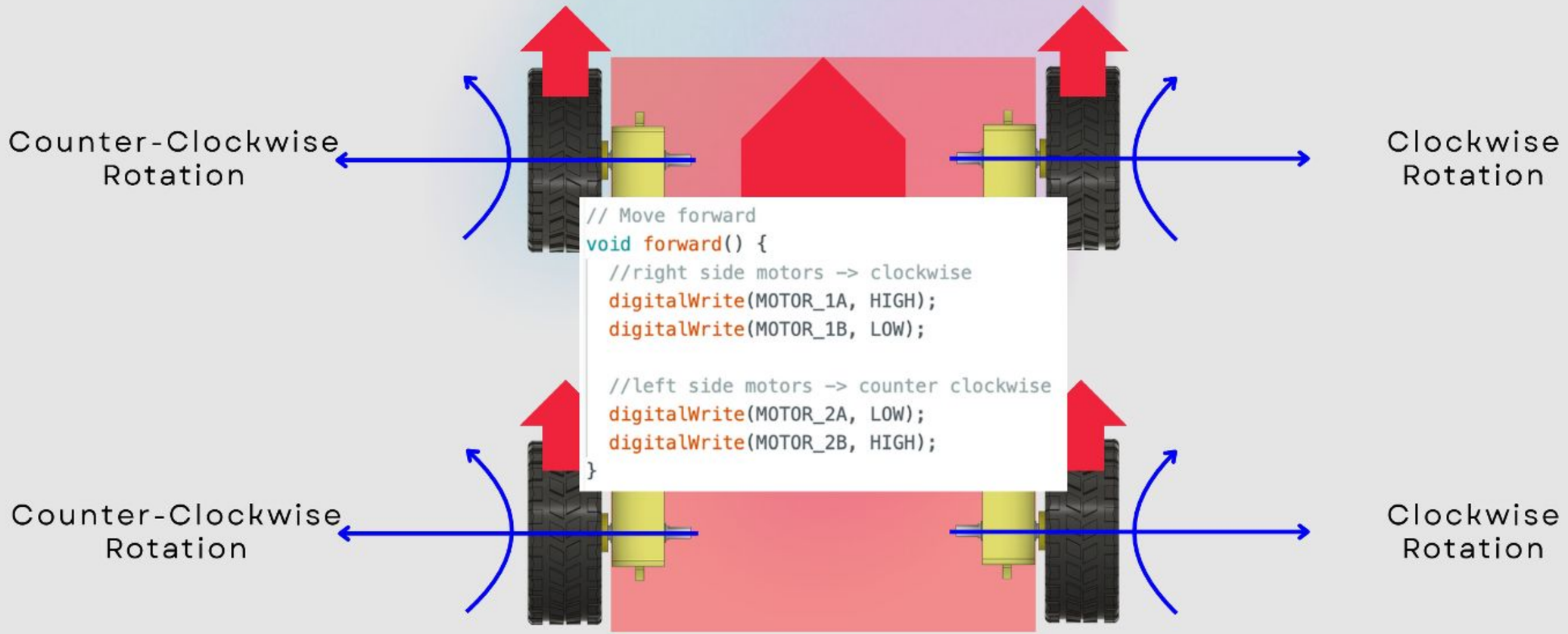


CHAPTER 32

STEERING FUNCTIONS

FORWARD

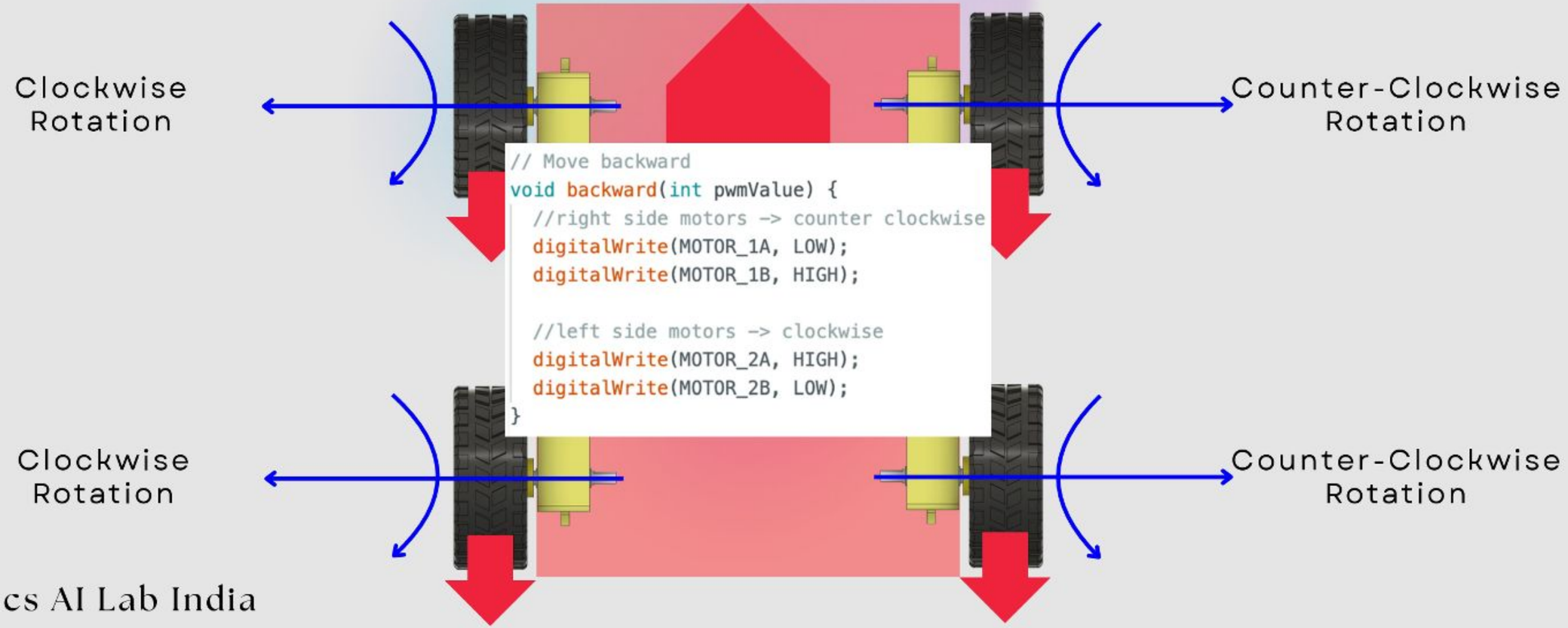
CAR FORWARD FUNCTION



REVERSE (BACKWARD)

MOTOR ROTATION FOR REVERSE MOTION

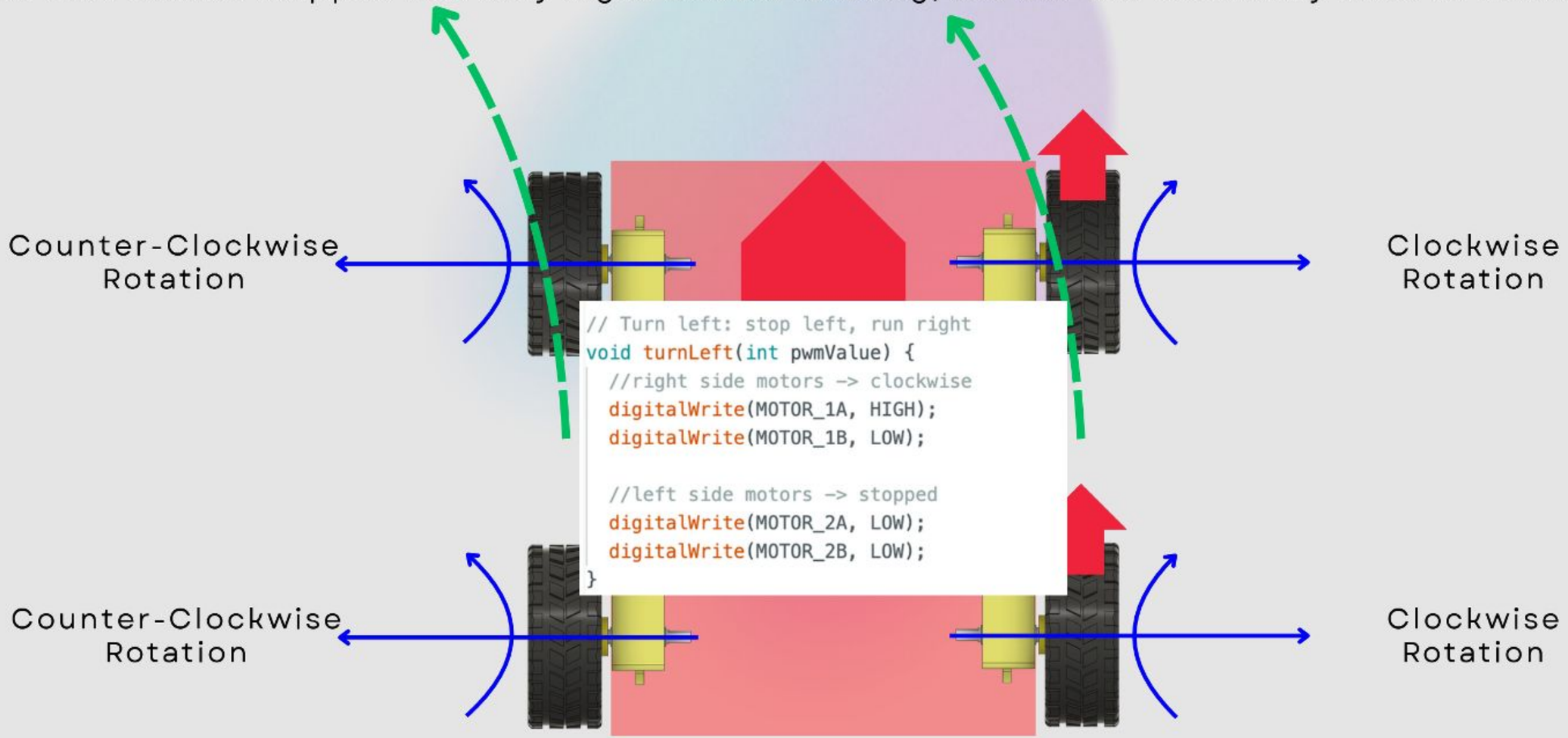
Let's look at individual motor rotation direction: -



LEFT - TURN

MOTOR ROTATION FOR LEFT MOTION

With Left wheels stopped and only Right wheels rotating, the car will inherently start to rotate left.



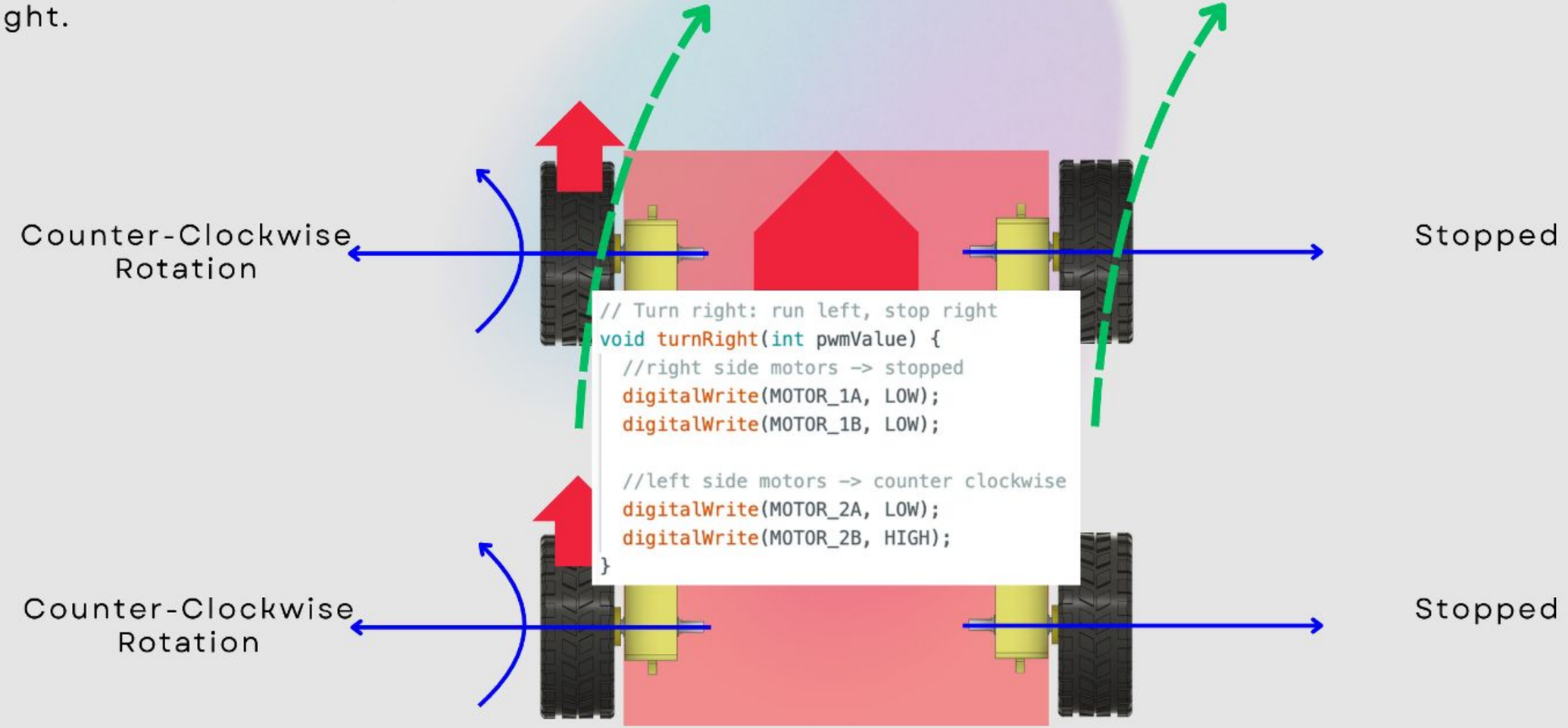

```
// Turn left: stop left, run right
void turnLeft(int pwmValue) {
  //right side motors -> clockwise
  digitalWrite(MOTOR_1A, HIGH);
  digitalWrite(MOTOR_1B, LOW);

  //left side motors -> stopped
  digitalWrite(MOTOR_2A, LOW);
  digitalWrite(MOTOR_2B, LOW);
}
```

RIGHT - TURN

MOTOR ROTATION FOR RIGHT MOTION

With Right wheels stopped and only Left wheels rotating, the car will inherently start to rotate Right.




```
// Turn right: run left, stop right
void turnRight(int pwmValue) {
  //right side motors -> stopped
  digitalWrite(MOTOR_1A, LOW);
  digitalWrite(MOTOR_1B, LOW);

  //left side motors -> counter clockwise
  digitalWrite(MOTOR_2A, LOW);
  digitalWrite(MOTOR_2B, HIGH);
}
```

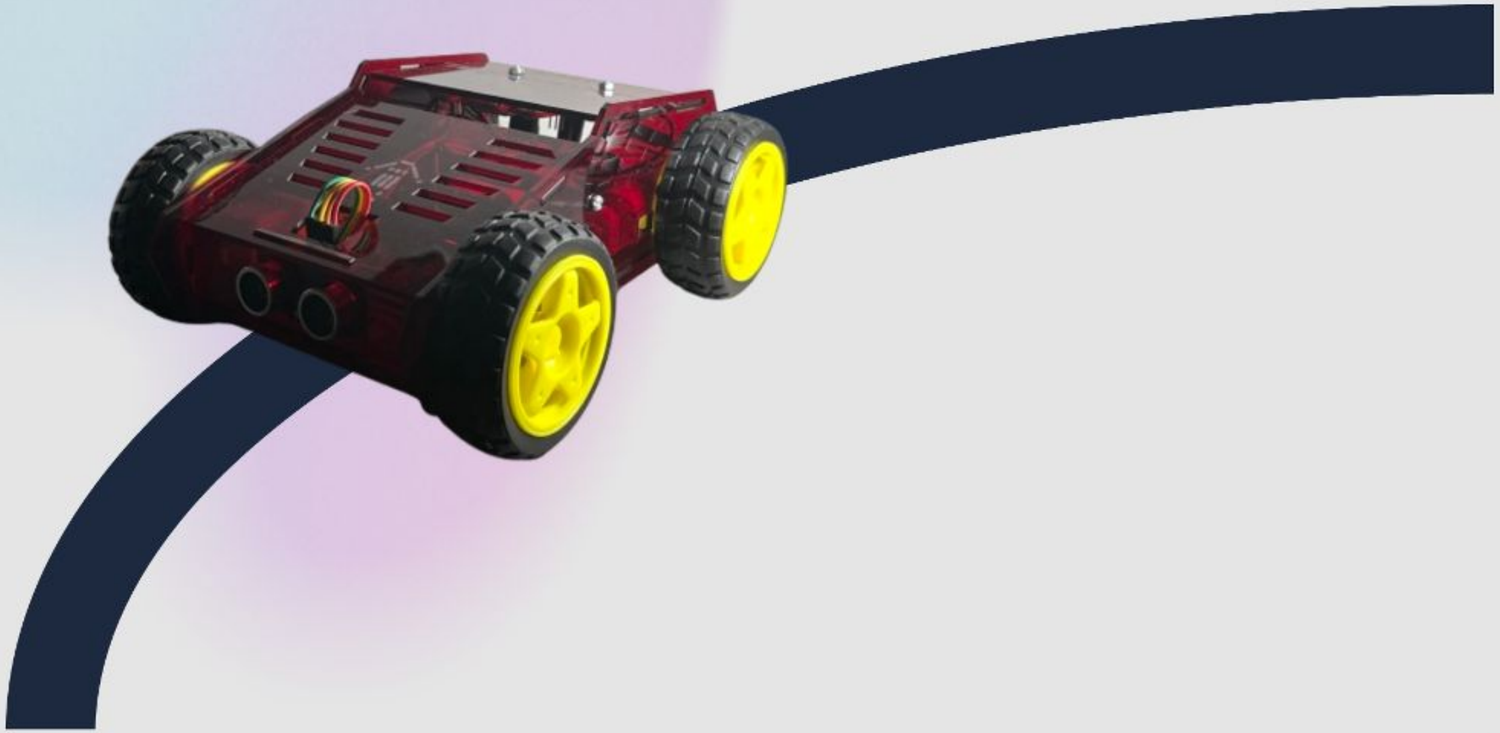
CHAPTER 33

THE LINE FOLLOWING ROBOT

THE LINE FOLLOWER ROBOT

LET'S DEFINE A "LINE FOLLOWING ROBOT"?

A Line Following Robot is a smart, autonomous robot that can sense and follow a line drawn on the ground.



LINE FOLLOWING ROBOT - APPLICATION

ACTION IN REAL LIFE

Line-following robots aren't just for fun—they're used in the real world to help people and industries!

- Big factories use line-following robots to carry parts, tools, and products from one area to another in a guided manner.
- Hospitals use these robots to deliver medicines, reports, or even food to patients.
- In big offices, hotels, or warehouses, line-following robots deliver packages or files across the building.

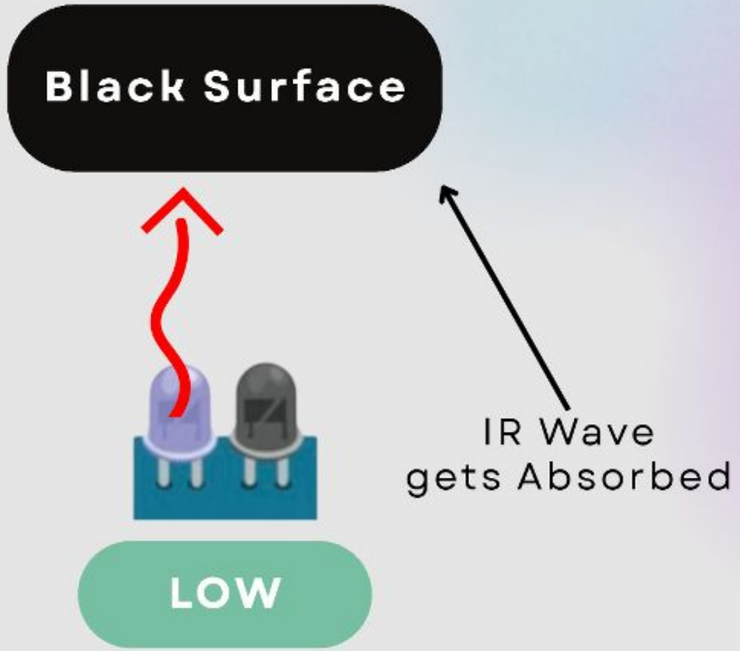
HOW TO SENSE THE LINE?

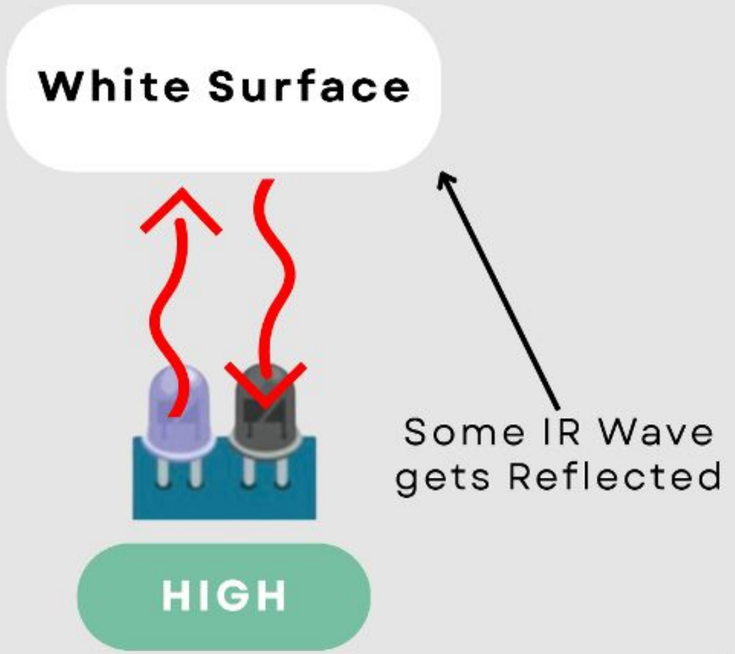
LET'S PUT THE IR SENSORS TO WORK

Line-following robots use Infrared (IR) sensors to "see" the line on the ground.

- IR Sensors = Robot's Eyes
- IR Sensors help differentiate between the black path and the white background

Imagine you shine a torch at a white sheet–it glows bright and bounces back. But if you shine it at black paper, the light disappears!



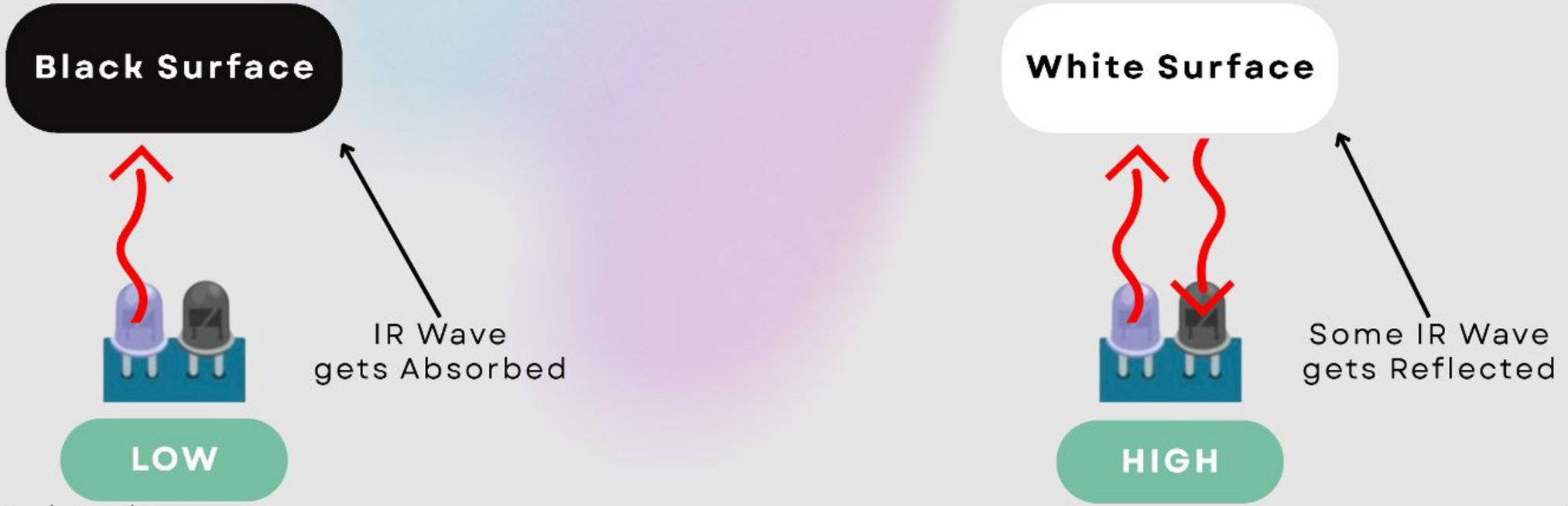


HOW TO SENSE THE LINE?

LET'S PUT THE IR SENSORS TO WORK

The science behind the IR sensor working

- White surface reflects light → Receiver in the Sensor see it → Reads as HIGH
- ⚫ Black surface absorbs light → Receiver in the Sensor sees NOTHING → Reads as LOW



HOW TO SENSE THE LINE?

TO SUMMARISE:

You have a black line and the car will follow it with the help of IR sensors.

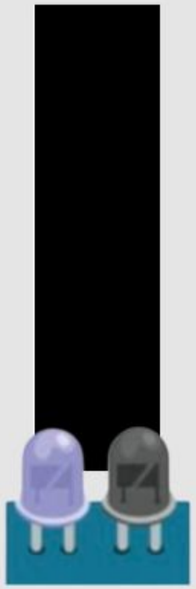
LET'S TALK ABOUT THE SENSOR SETUP

USING ONLY ONE SENSOR

Do you think one sensor is sufficient?

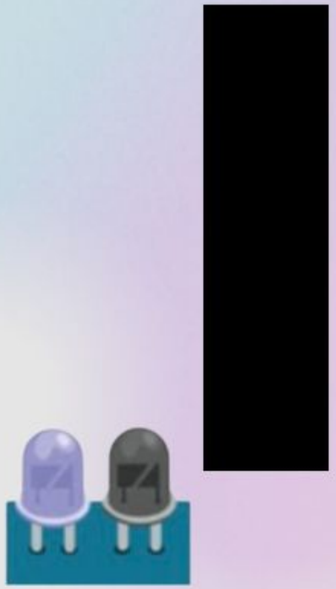
In both cases when the car moves LEFT or RIGHT, the sensor reads HIGH, thus there no unique way to say if the car moved LEFT or RIGHT.

Car is on the line



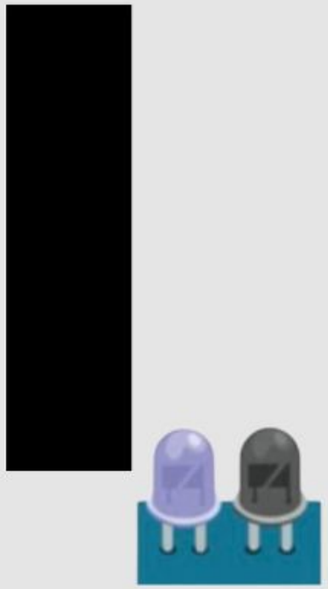
Sensor reading
= LOW

Car moves left



Sensor reading
= HIGH

Car moves right



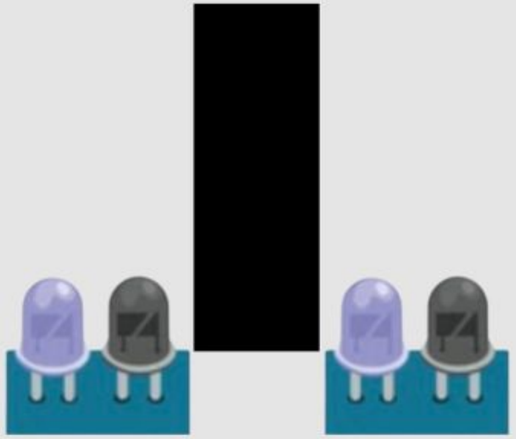
Sensor reading
= HIGH

LET’S TALK ABOUT THE SENSOR SETUP

LET’S TRY TWO SENSORS THIS TIME

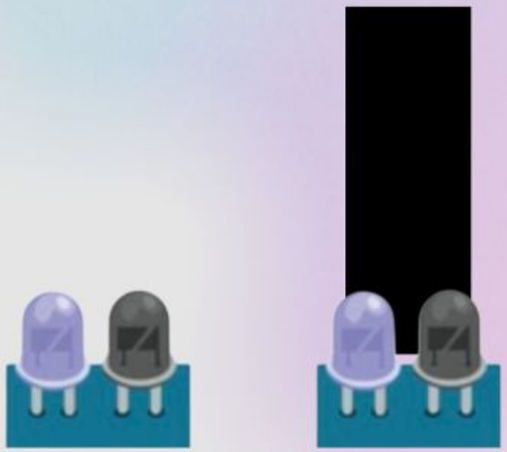
By using two sensors, we are able to get 3 unique sensor value combinations for three different situations.

Car is on the line and centered



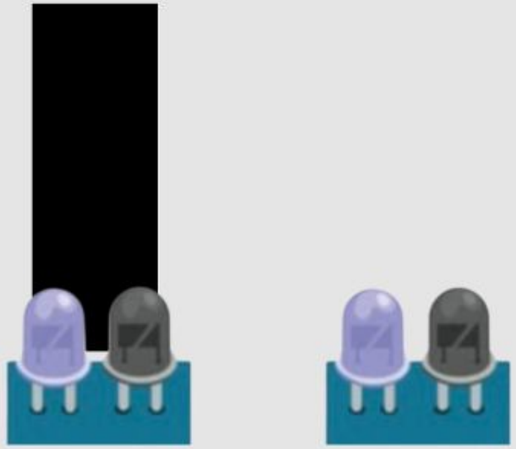
Sensor reading
Sensor-L = HIGH
Sensor-R = HIGH

Car moves left



Sensor reading
Sensor-L = HIGH
Sensor-R = LOW

Car moves right



Sensor reading
Sensor-L = LOW
Sensor-R = HIGH

LET'S TALK ABOUT THE SENSOR SETUP

TRUTH TABLE - REVIEW

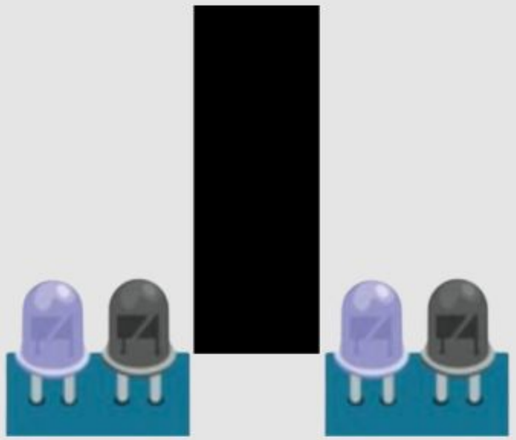
Car Postion	Value of Left Sensor	Value of Right Sensor
CENTER	HIGH	HIGH
LEFT	HIGH	LOW
RIGHT	LOW	HIGH

LET'S TALK ABOUT THE SENSOR SETUP

WHAT ACTIONS WILL THE ROBOT MAKE

Based on the sensor outcome, what should the robot do?

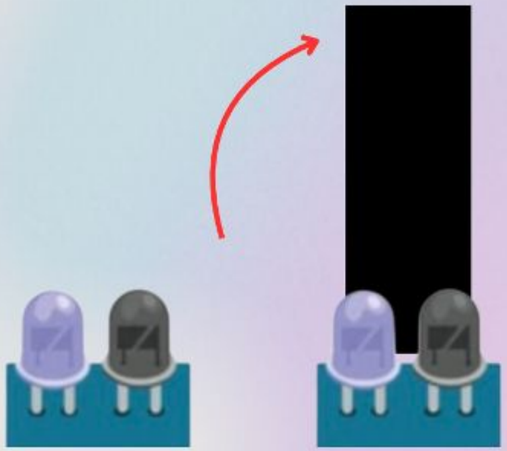
Car is on the line and centered



Sensor reading
Sensor-L = HIGH
Sensor-R = HIGH

Continue to move straight

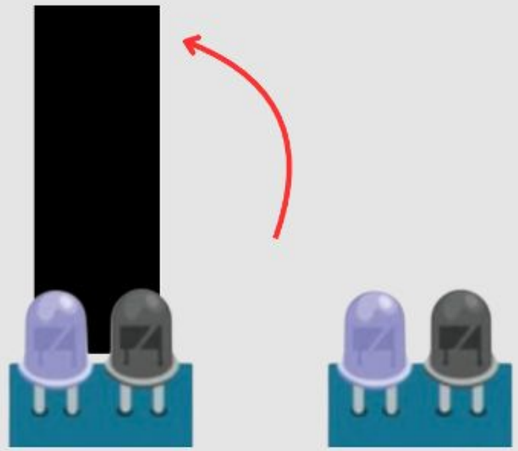
Car has moved to the left



Sensor reading
Sensor-L = HIGH
Sensor-R = LOW

Rotate Right to get back on the line

Car has moved to the right



Sensor reading
Sensor-L = LOW
Sensor-R = HIGH

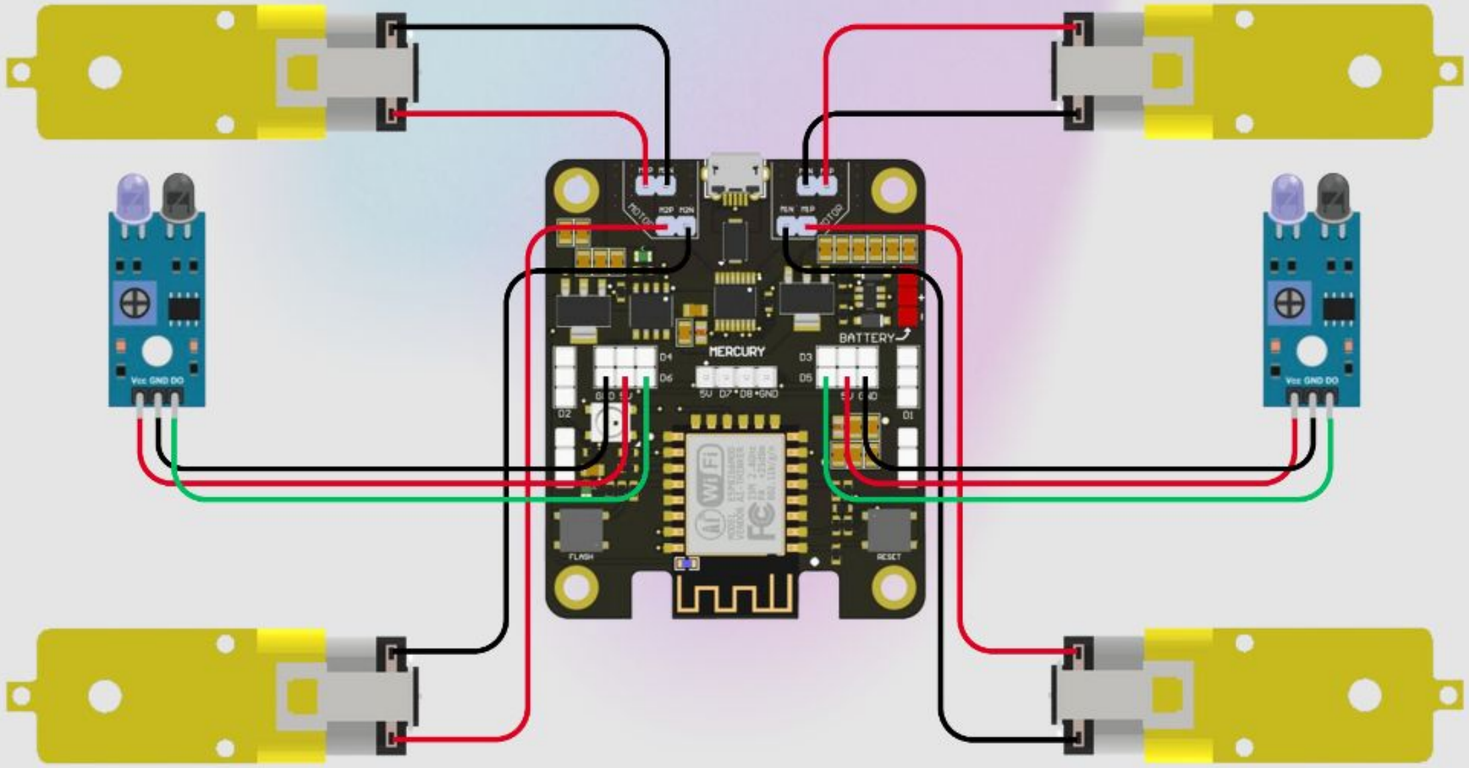
Rotate Left to get back on the line

MAKING THE CIRCUIT

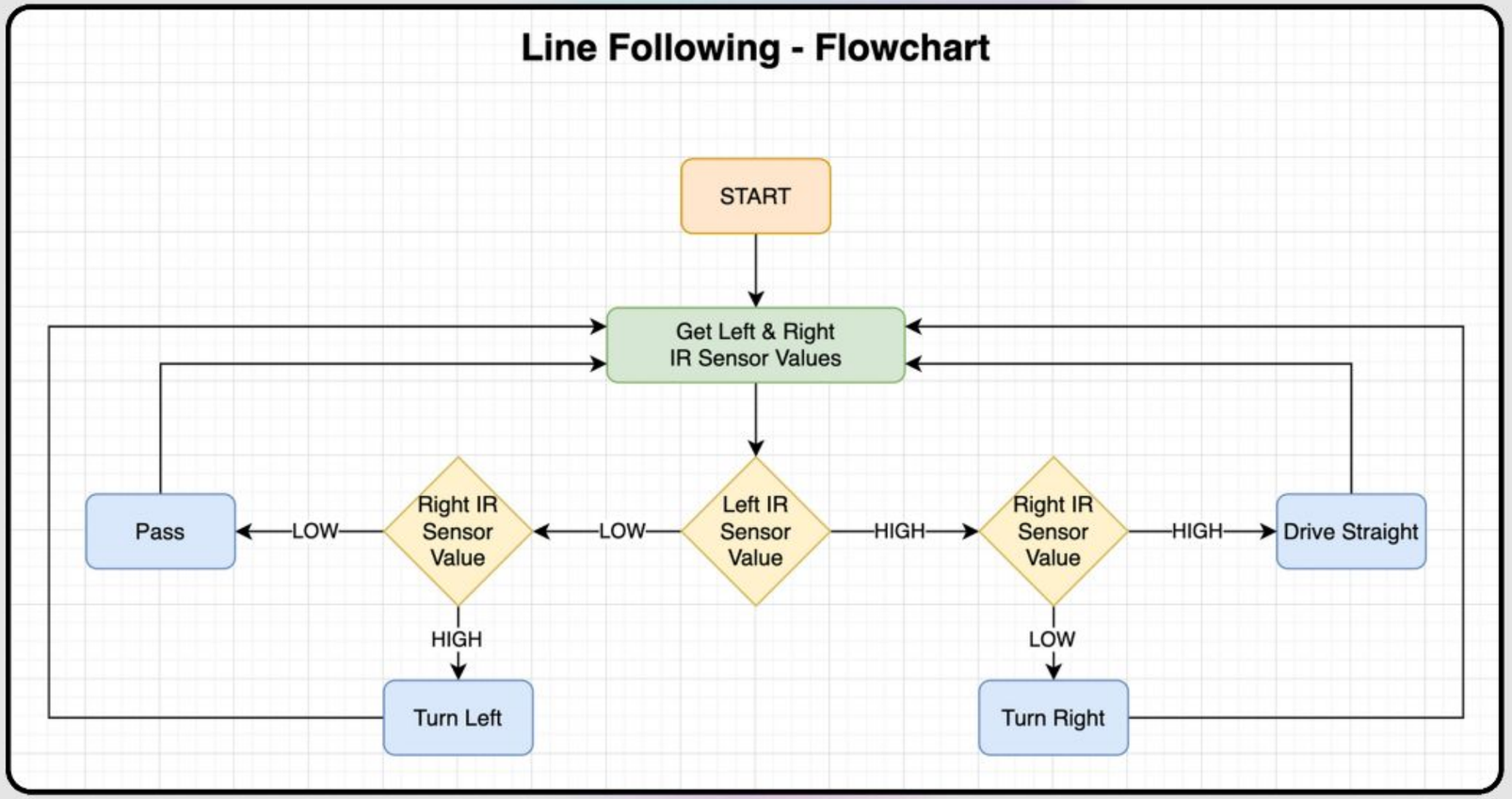
CONNECTING THE ELECTRONICS

- Left Sensor
- (+) 🔴 Vcc → 5V pin on the board
- (-) ⚫ Gnd → Gnd pin on the board
- (D) 🟢 D0 → D6 pin on board

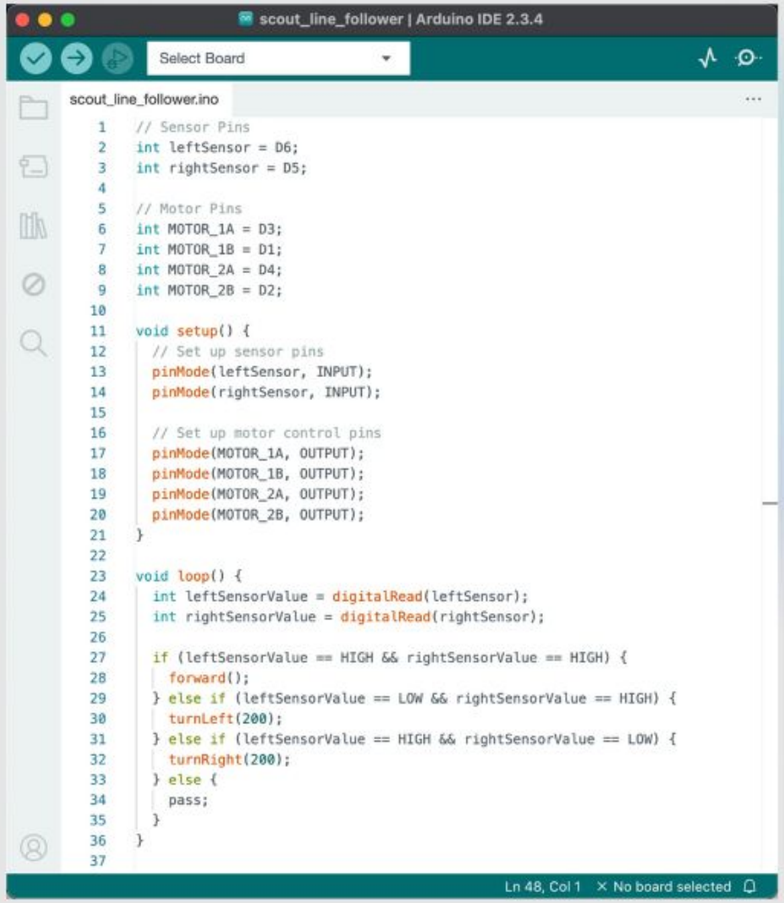
- Right Sensor
- (+) 🔴 Vcc → 5V pin on the board
- (-) ⚫ Gnd → Gnd pin on the board
- (D) 🟢 D0 → D5 pin on board



LOGIC BEHIND THE ROBOT'S BRAIN



CODE TIME!


scout_line_follower | Arduino IDE 2.3.4
Select Board
scout_line_follower.ino
// Sensor Pins
int leftSensor = D6;
int rightSensor = D5;
// Motor Pins
int MOTOR_1A = D3;
int MOTOR_1B = D1;
int MOTOR_2A = D4;
int MOTOR_2B = D2;
void setup() {
// Set up sensor pins
pinMode(leftSensor, INPUT);
pinMode(rightSensor, INPUT);
// Set up motor control pins
pinMode(MOTOR_1A, OUTPUT);
pinMode(MOTOR_1B, OUTPUT);
pinMode(MOTOR_2A, OUTPUT);
pinMode(MOTOR_2B, OUTPUT);
}
void loop() {
int leftSensorValue = digitalRead(leftSensor);
int rightSensorValue = digitalRead(rightSensor);
if (leftSensorValue == HIGH && rightSensorValue == HIGH) {
forward();
} else if (leftSensorValue == LOW && rightSensorValue == HIGH) {
turnLeft(200);
} else if (leftSensorValue == HIGH && rightSensorValue == LOW) {
turnRight(200);
} else {
pass;
}
}
Ln 48, Col 1
No board selected


scout_line_follower | Arduino IDE 2.3.4
Select Board
scout_line_follower.ino
// Move forward
void forward() {
//right side motors -> clockwise
digitalWrite(MOTOR_1A, HIGH);
digitalWrite(MOTOR_1B, LOW);
//left side motors -> counter clockwise
digitalWrite(MOTOR_2A, LOW);
digitalWrite(MOTOR_2B, HIGH);
}
// Move backward
void backward(int pwmValue) {
//right side motors -> counter clockwise
digitalWrite(MOTOR_1A, LOW);
digitalWrite(MOTOR_1B, HIGH);
//left side motors -> clockwise
digitalWrite(MOTOR_2A, HIGH);
digitalWrite(MOTOR_2B, LOW);
}
// Turn left: stop left, run right
void turnLeft(int pwmValue) {
//right side motors -> clockwise
digitalWrite(MOTOR_1A, HIGH);
digitalWrite(MOTOR_1B, LOW);
//left side motors -> stopped
digitalWrite(MOTOR_2A, LOW);
digitalWrite(MOTOR_2B, LOW);
}
// Turn right: run left, stop right
void turnRight(int pwmValue) {
//right side motors -> stopped
digitalWrite(MOTOR_1A, LOW);
digitalWrite(MOTOR_1B, LOW);
//left side motors -> counter clockwise
digitalWrite(MOTOR_2A, LOW);
digitalWrite(MOTOR_2B, HIGH);
}
// Stop both motors
void stopMotors() {
//right side motors -> stopped
digitalWrite(MOTOR_1A, LOW);
digitalWrite(MOTOR_1B, LOW);
//left side motors -> stopped
digitalWrite(MOTOR_2A, LOW);
digitalWrite(MOTOR_2B, LOW);
}
Ln 92, Col 1
No board selected

DEEP DIVE INTO THE CODE

Now that you have become and expert, lets straight away jump to void loop part of the code

HOW DOES THIS CODE WORK?

DEEP DIVE INTO THE CODE

As discussed in the flow chart,

- **IF** both **left and right sensors read HIGH value** → the car is properly aligned with the line, so move **FORWARD**
- **ELSE IF left sensor reads LOW and right sensor read HIGH** → the car has drifted right, so **TURN LEFT**
- **ELSE IF left sensor reads HIGH and right sensor read LOW** → the car has drifted left, so **TURN RIGHT**

```
22
23  void loop() {
24    int leftSensorValue = digitalRead(leftSensor);
25    int rightSensorValue = digitalRead(rightSensor);
26
27    if (leftSensorValue == HIGH && rightSensorValue == HIGH) {
28      forward();
29    } else if (leftSensorValue == LOW && rightSensorValue == HIGH) {
30      turnLeft(200);
31    } else if (leftSensorValue == HIGH && rightSensorValue == LOW) {
32      turnRight(200);
33    } else {
34      pass;
35    }
36  }
37
```

CHAPTER 34

OBSTACLE AVOIDING ROBOT

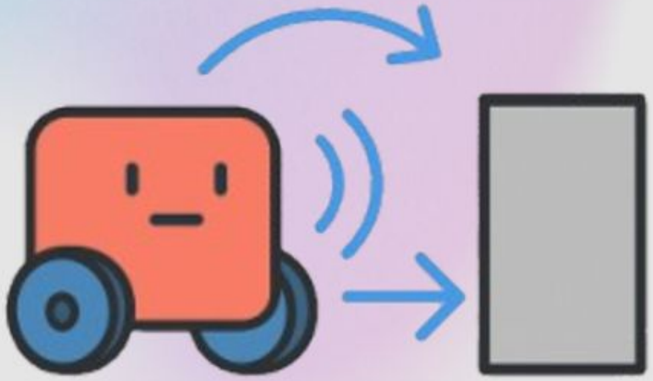
AN OBSTACLE AVOIDING ROBOT!

LET'S DEFINE AN "OBSTACLE AVOIDING ROBOT"?

An obstacle avoiding robot is a smart robot that moves around and dodges obstacles all by itself!

- When something is in the way, the robot turns or stops to avoid crashing into the obstacle.
- It's like a person walking with their hands out in the dark – when you feel something, you stop or move away!

In this project, when an obstacle is detected, the bot will turn right.



OBSTACLE AVOIDING ROBOT - APPLICATION

ACTION IN REAL LIFE

- Vacuum Robots (like Roomba!)
  - These robots move around the house, cleaning floors without bumping into furniture.
  - They use sensors to detect walls, sofas, and pets, and smartly change direction!
- Self-Driving Cars (like Tesla Model Y)
  - Autonomous cars have advanced obstacle sensors to avoid other cars, people, and obstacles on the road.
- Factory Robots
  - Robots in factories move parts and products without crashing into workers or machines.

SENSING OBSTACLES

HOW DOES THE ROBOT SENSE THE OBSTACLE?

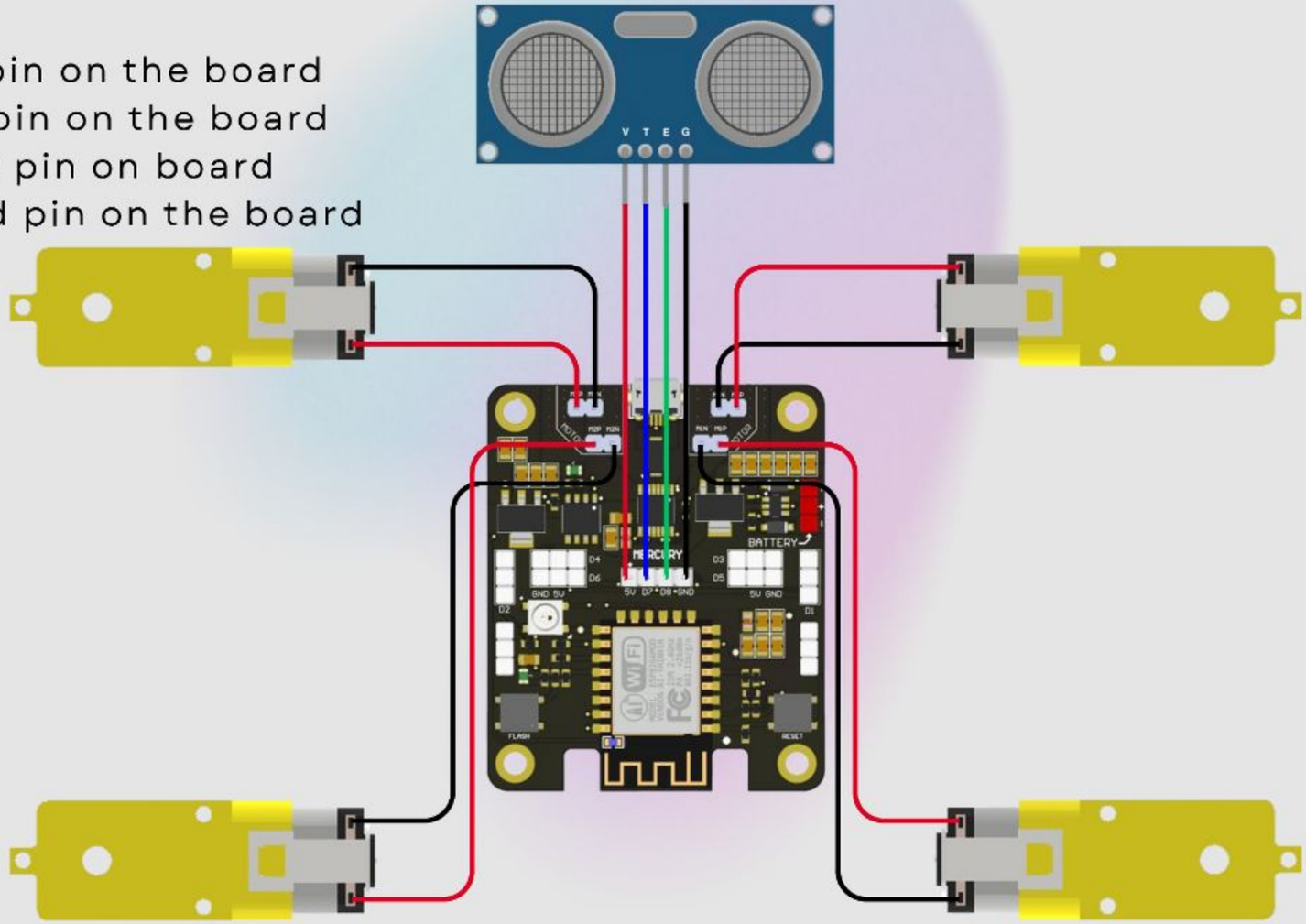
Now that the understand the working of an ultrasonic sensor, we can implement its working principle in our obstacle avoiding robot.

- Use the distance measuring technique
- Set a threshold
- The robot takes an action When the distance from obstacle is below the threshold.

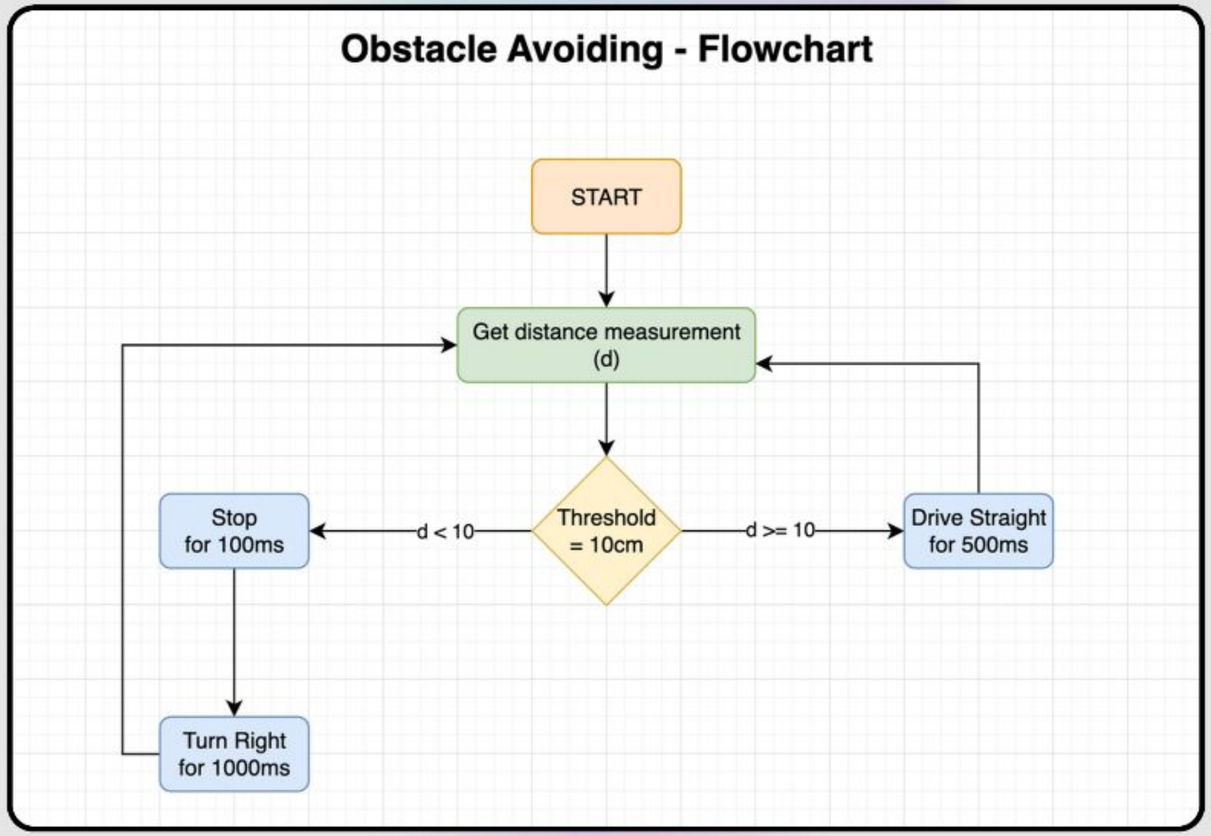
MAKING THE CIRCUIT

CONNECTING THE ELECTRONICS

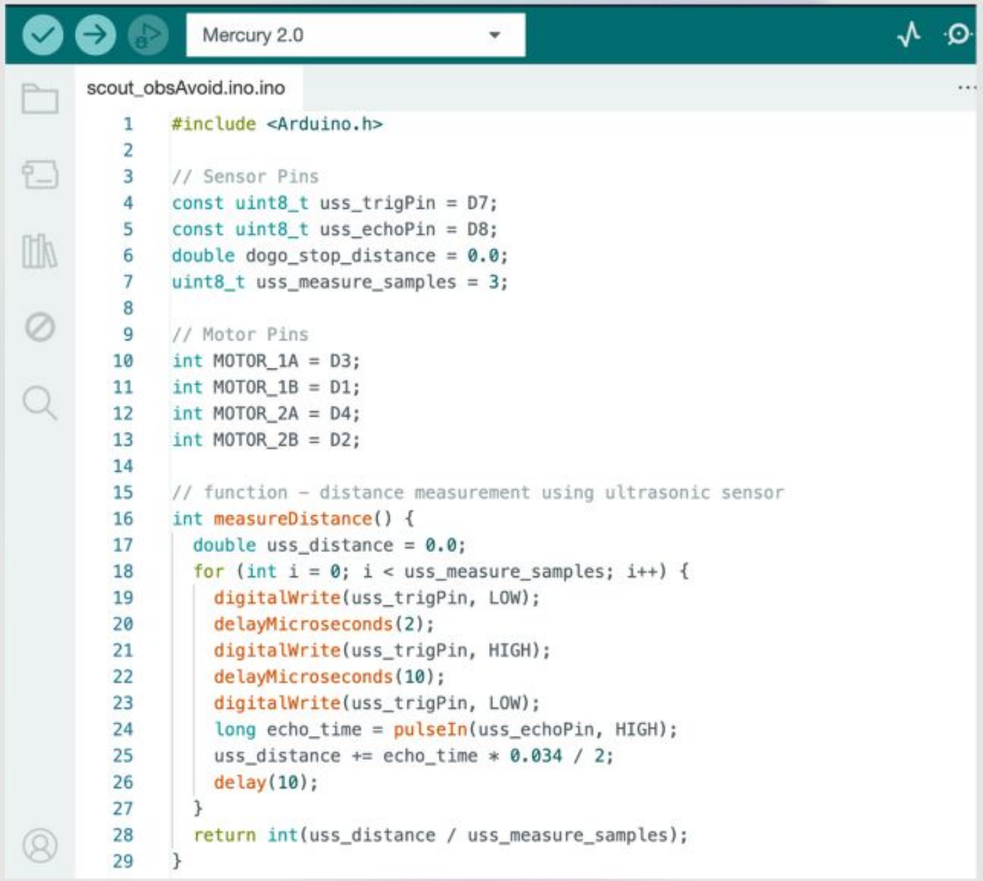
- Left Sensor
- (+) 🔴 Vcc → 5V pin on the board
- (T) 🔵 Trig → D7 pin on the board
- (E) 🟢 Echo → D8 pin on board
- (-) ⚫ Gnd → Gnd pin on the board



CODE LOGIC


Obstacle Avoiding - Flowchart
START
Get distance measurement (d)
Threshold = 10cm
d < 10
d >= 10
Stop for 100ms
Drive Straight for 500ms
Turn Right for 1000ms

CODING THE ROBOT




```ino
#include <Arduino.h>

// Sensor Pins
const uint8_t uss_trigPin = D7;
const uint8_t uss_echoPin = D8;
double dogo_stop_distance = 0.0;
uint8_t uss_measure_samples = 3;

// Motor Pins
int MOTOR_1A = D3;
int MOTOR_1B = D1;
int MOTOR_2A = D4;
int MOTOR_2B = D2;

// function - distance measurement using ultrasonic sensor
int measureDistance() {
  double uss_distance = 0.0;
  for (int i = 0; i < uss_measure_samples; i++) {
    digitalWrite(uss_trigPin, LOW);
    delayMicroseconds(2);
    digitalWrite(uss_trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(uss_trigPin, LOW);
    long echo_time = pulseIn(uss_echoPin, HIGH);
    uss_distance += echo_time * 0.034 / 2;
    delay(10);
  }
  return int(uss_distance / uss_measure_samples);
}
```

DISTANCE MEASUREMENT

APPLY FILTERING TO DISTANCE MEASUREMENT

Previously we have seen distance measurement function, this time lets apply filtering to it.

- We will simply apply “AVERAGING” to filter Noise
- The reason to filter, is to avoid unwarranted sensor reading

Example, in this case if the sensor is supposed to read-out approximately 15cm, but due to some external reason, it randomly read-out 20cm, then this becomes “**noise**” that needs to be suppressed from the actual reading.

Paying attention to line 18: -

- **uss_measure_samples = 3;** => we are collecting 3 samples to average and generate final distance measurement reading.

```
digitalWrite(trigPin, LOW);
delayMicroseconds(10);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
long echo_time = pulseIn(echoPin, HIGH);
```

actual distance measurement

```
15  // function - distance measurement using ultrasonic sensor
16  int measureDistance() {
17    double uss_distance = 0.0;
18    for (int i = 0; i < uss_measure_samples; i++) {
19      digitalWrite(uss_trigPin, LOW);
20      delayMicroseconds(2);
21      digitalWrite(uss_trigPin, HIGH);
22      delayMicroseconds(10);
23      digitalWrite(uss_trigPin, LOW);
24      long echo_time = pulseIn(uss_echoPin, HIGH);
25      uss_distance += echo_time * 0.034 / 2;
26      delay(10);
27    }
28    return int(uss_distance / uss_measure_samples);
29  }
```

filtered measurement

HOW DOES THIS CODE WORK? PART I

DEEP DIVE INTO THE CODE

As discussed in the flow chart,

- Measure the distance between the robot and the obstacle
- **IF** distance between the robot and obstacle is less than 10cm → the car shall **stop for 500ms**, then **keep turning right for 1000ms** or 1s.
  - Turning right **timing** can be tuned for each robot to achieve the correct turning angle.
- **ELSE IF** the distance measured is greater than or equal to 10cm → keep **driving straight for 500ms**
- Back to **measuring the distance again and the loop continues**

```
101  void loop() {
102    dogo_stop_distance = measureDistance();
103    if (dogo_stop_distance < 10) {
104      stopMotors();
105      delay(500);
106      turnRight();
107      delay(1000);
108    } else {
109      forward();
110      delay(500);
111    }
112  }
```

HOW DOES THIS CODE WORK? PART II

MOTOR CONTROL → SAME AS LINE FOLLOWING ROBOT

```
// Move forward
void forward() {
  //right side motors -> clockwise
  digitalWrite(MOTOR_1A, HIGH);
  digitalWrite(MOTOR_1B, LOW);

  //left side motors -> counter clockwise
  digitalWrite(MOTOR_2A, LOW);
  digitalWrite(MOTOR_2B, HIGH);
}

// Move backward
void backward() {
  //right side motors -> counter clockwise
  digitalWrite(MOTOR_1A, LOW);
  digitalWrite(MOTOR_1B, HIGH);

  //left side motors -> clockwise
  digitalWrite(MOTOR_2A, HIGH);
  digitalWrite(MOTOR_2B, LOW);
}

// Turn left: stop left, run right
void turnLeft() {
  //right side motors -> clockwise
  digitalWrite(MOTOR_1A, HIGH);
  digitalWrite(MOTOR_1B, LOW);

  //left side motors -> stopped
  digitalWrite(MOTOR_2A, LOW);
  digitalWrite(MOTOR_2B, LOW);
}

// Turn right: run left, stop right
void turnRight() {
  //right side motors -> stopped
  digitalWrite(MOTOR_1A, LOW);
  digitalWrite(MOTOR_1B, LOW);

  //left side motors -> counter clockwise
  digitalWrite(MOTOR_2A, LOW);
  digitalWrite(MOTOR_2B, HIGH);
}

// Stop both motors
void stopMotors() {
  //right side motors -> stopped
  digitalWrite(MOTOR_1A, LOW);
  digitalWrite(MOTOR_1B, LOW);

  //left side motors -> stopped
  digitalWrite(MOTOR_2A, LOW);
  digitalWrite(MOTOR_2B, LOW);
}
```

THE END

HOPE YOU **LEARNED SOMETHING** NEW
WHILE **ENJOYING** THIS
ROBOTICS JOURNEY