

**MICROCONTROLLERS FOR MECHATRONICS – MECA442**

Experiment 3: Speed measurement using microcontrollers

**Jana Kabrit1, Oussama Al Ahmad Matar1**

**1** Rafik Hariri University, College of Engineering, Department of Mechanical and Mechatronics

*Abstract*

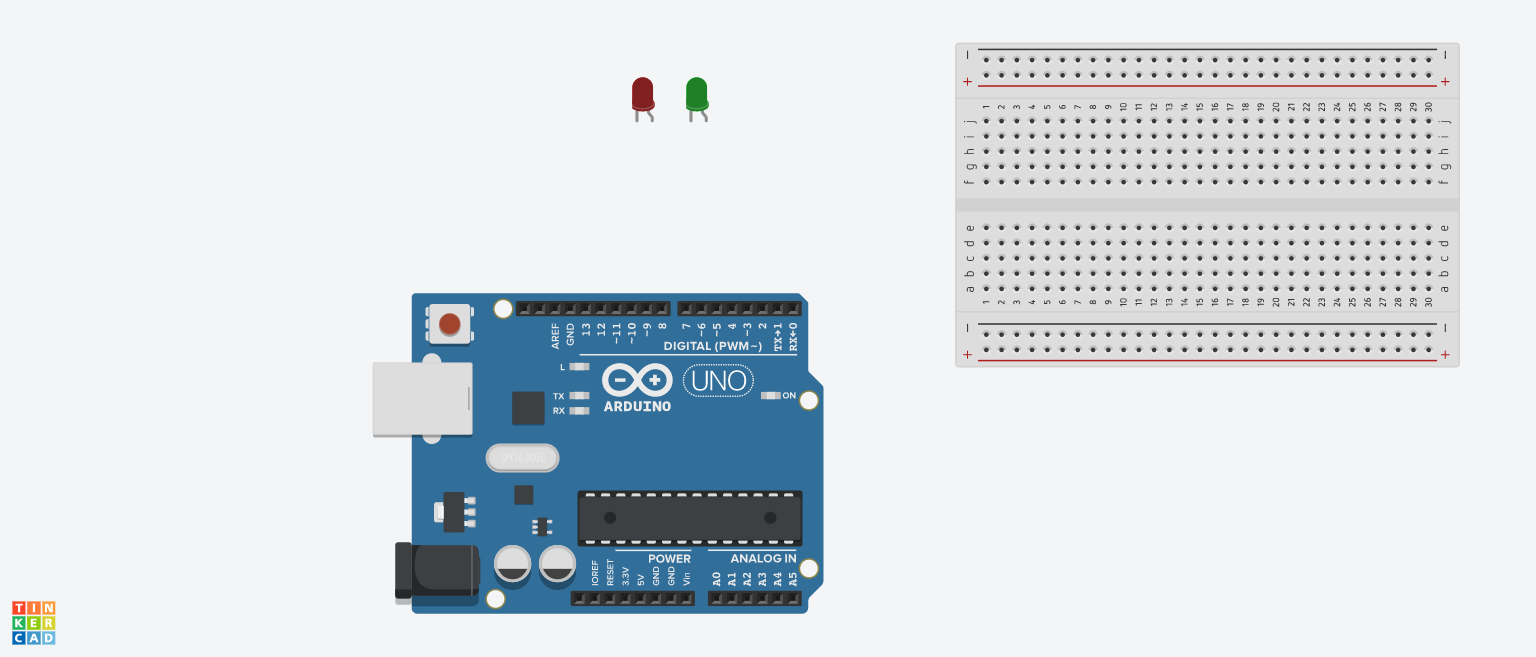
*The aim of this experiment is to design and simulate a circuit that will measure the distance between a robot an obstacle, record the difference in time, and display the speed. This will be done through connecting an Arduino Uno to an ultrasonic sensor for measuring the distance. The difference in time can be measured through the function Millis in the Arduino Uno code. Finally, the speed will be calculated in the code of the Arduino using the basic formula (speed = distance / time) and displayed on an LCD screen that is connected to the Arduino. The simulation will take place on Tinker CAD.*

**Keywords:** Arduino Uno, Ultrasonic Sensor, Millis, LCD Screen, Speed, Tinker CAD.

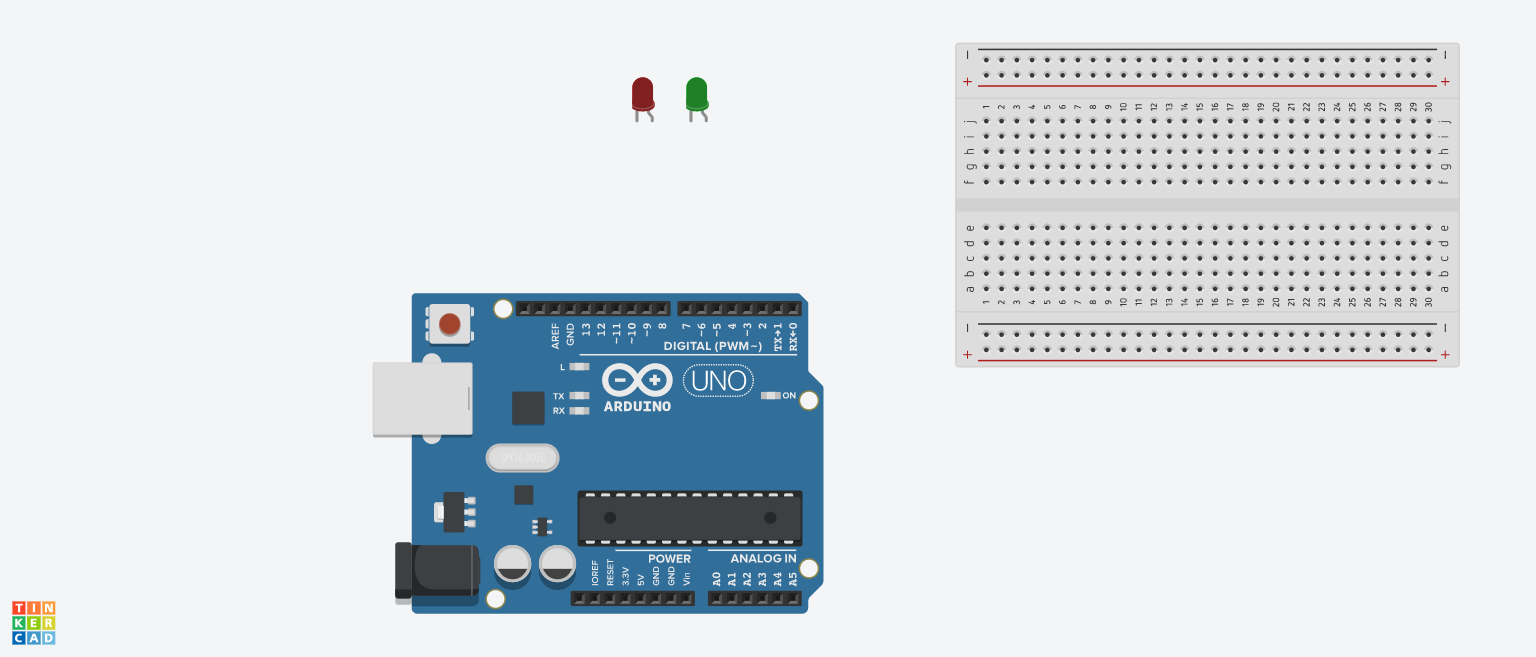
1. INTRODUCTION

In this experiment, we will display the speed of a moving robot by recording the difference in distance it is moving every 0.1 seconds and calculating the speed using the formula (speed = Δ distance / Δ time). We will be using an ultrasonic sensor to measure the distance it is moving by measuring how close the robot is getting to an obstacle. In the code, the Arduino will measure the new distance every 0.1 seconds using the Millis function. Then, we will subtract the new distance measured from the previous one- obtaining the difference in distance. The speed is then displayed on an LCD screen that is connected to the Arduino directly and through a breadboard.

1. **MATERIALS AND METHODS**
   1. **Materials**
      1. **Simulated Electronics Components**
2. Arduino Uno

The Arduino Uno (Figure 1) is a microcontroller board based on the ATmega328. It has 20 digital input/output pins (of which 6 can be used as PWM outputs and 6 can be used as analog inputs), a 16 MHz resonator, a USB connection, a power jack, an in-circuit system programming (ICSP) header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.[1] In this experiment, we need the Arduino so that we can program it to record the difference of distance every 0.1 seconds so we can calculate the speed.

**FIGURE 1:** ARDUINO UNO BOARD



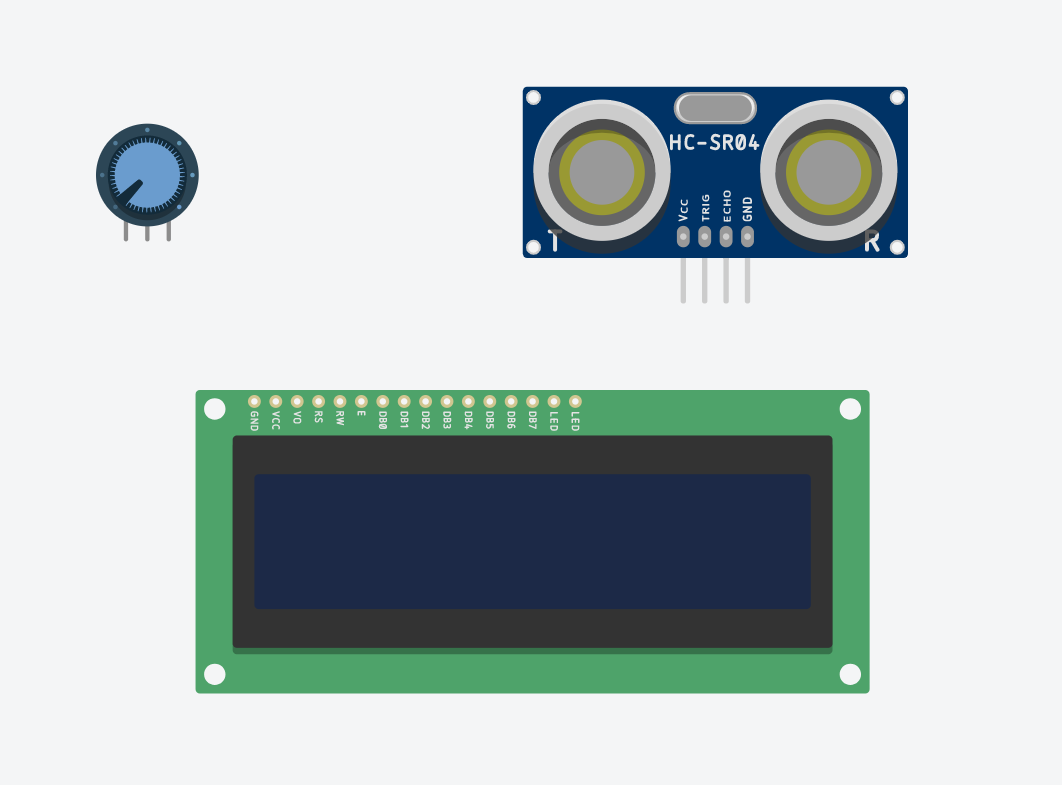
1. Breadboard:

A breadboard (Figure 2) is a rectangular plastic board with a bunch of tiny holes in it. These holes let you easily insert electronic components to prototype (meaning to build and test an early version of) an electronic circuit.[2] In this experiment, we will use the breadboard to connect the LCD display to the Arduino through a potentiometer.

**FIGURE 2:** BREADBOARD

1. Ultrasonic Sensor:

The HC-SR04 ultrasonic sensor (Figure 3) uses SONAR to determine the distance of an object just like the bats do. It offers excellent non-contact range detection with high accuracy and stable readings in an easy-to-use package from 2 cm to 400 cm or 1” to 13 feet.



The operation is not affected by sunlight or black material, although acoustically, soft materials like cloth can be difficult to detect. It comes complete with ultrasonic transmitter and receiver module.[3] In this experiment, we will use the ultrasonic sensor to calculate the difference in distance of the robot movement to later calculate the speed and display it.

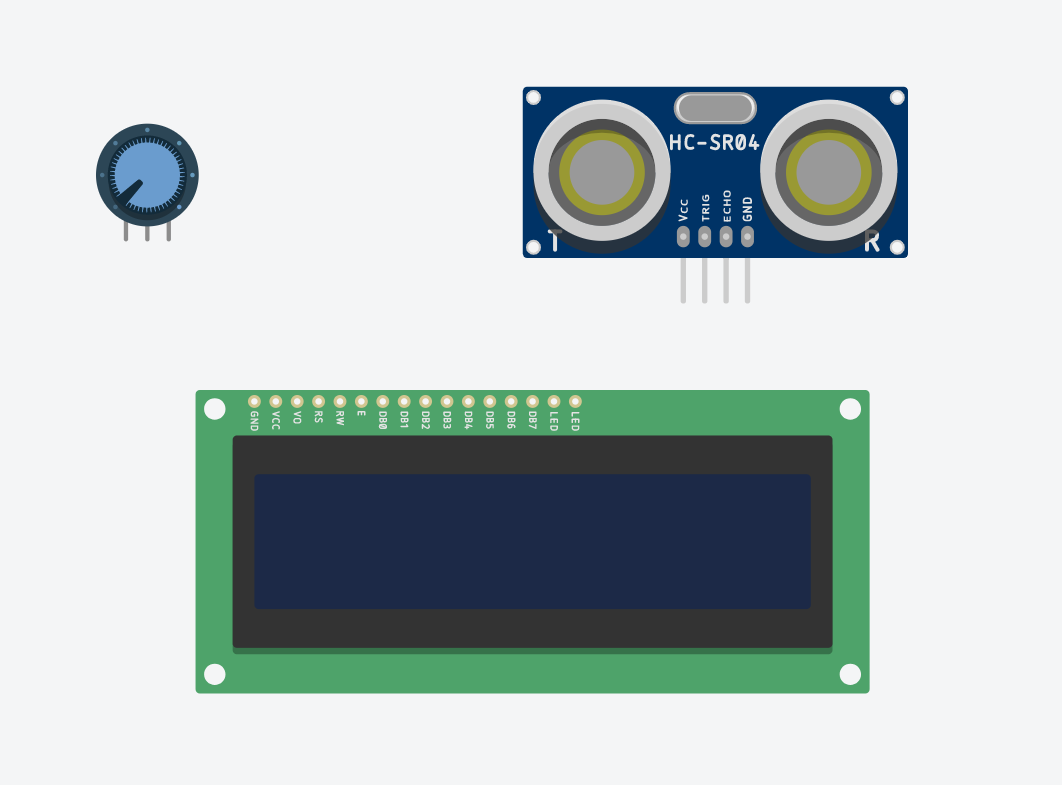
**FIGURE 3:** ULTRASONIC SENSOR

1. LCD Screen:

 These LCDs are available in many different sizes (16×2 1602, 20×4 2004, 16×1 etc.), but they all use the same HD44780 parallel interface LCD controller chip from Hitachi. This means you can easily swap them. The LCD has 16 connection pins, numbered 1-16 from left to right. If the display does not include a resistor, you will need to add one between 5 V and pin 15. It should be safe to use a 220Ω resistor, but this value might make your display a bit dim. In this experiment, we used a potentiometer to get the best brightness. The maximum current rating of the backlight can be checked from the and used this to select an appropriate resistor value.[4]

**FIGURE 4:** LIQUID CRYSTAL DISPLAY

1. Potentiometer:

 The potentiometer, commonly referred to as a “pot”, is a three-terminal mechanically operated rotary analogue device which can be found and used in a large variety of electrical and electronic circuits. They are passive devices, meaning they do not require a power supply or additional circuitry in order to perform their basic linear or rotary position function.[5] In this experiment, we will use the potentiometer to get the best brightness of the LCD screen available.

**FIGURE 5:** POTENTIOMETER

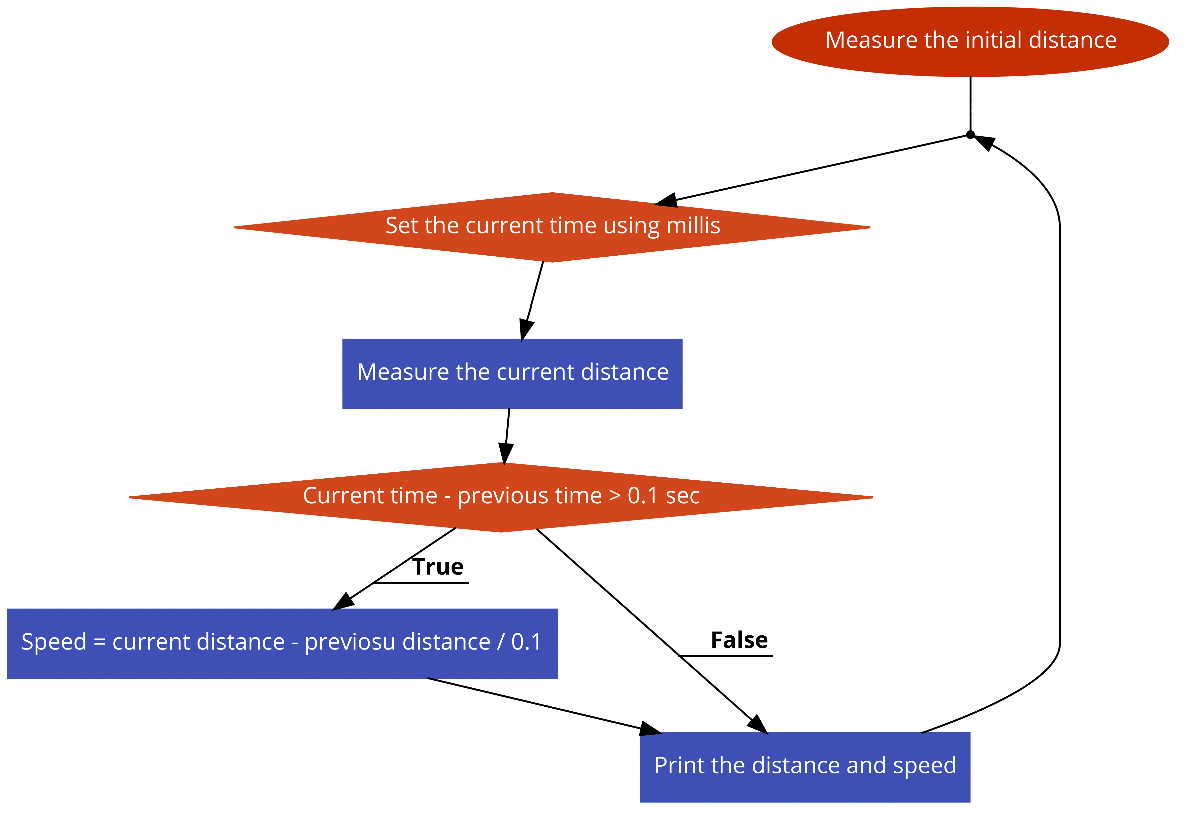
* + 1. **Code Components**

1. const: A variable qualifier that modifies the behavior of the variable, making a variable "read-only" and constant.
2. long int: Long variables are extended size variables for number storage, and store 32 bits (4 bytes) instead of the usual 16 bits (2 bytes).
3. unsigned int: Same as int in that it stores a 2-byte value. Instead of storing negative numbers however they only store positive values, yielding a useful range of 0 to 65,535.
4. void setup: The function is called whenever the program starts. It is used to initialize variables, pin modes, etc. It will only run once after each powerup or reset of the Arduino board.
5. pinMode(): A function used to configure a specific pin to behave either as an input or an output, usually in the void setup.
6. void loop: The function that holds the code inside and runs over and over as long as the board is turned on.
7. millis(): A command that returns the number of milliseconds passed since the Arduino board began running the current program. This number will overflow (go back to zero), after approximately 50 days.
8. LiquidCrystal lcd: Provides a set of endpoints to manage Arduino IoT Cloud Devices, Things, Properties and Timeseries. This API can be called just with any HTTP Client, or using one of these clients
9. lcd.setCursor(): Initializes the interface to the LCD screen, and specifies the dimensions (width and height) of the display
10. lcd.print(): Displays what is in the argument on the LCD Screen
11. pulseIn: Reads the distance between the robot and the obstacle from the Ultrasonic Sensor.
    1. **Methods**
       1. **Code Description**

Once the Arduino Uno turns on, it sets up the pins for the LCD display (in our case pins 1, 2, 4, 5, 6, and 7 as the RS, enable, D4, D5, D6, and D7 respectively) to be able to print the results on it, and the pins for the ultrasonic sensor for trigger and echo at pins 10 and 9 in the void setup. It also sets the time interval as 100 milliseconds (or 0.1 seconds) and measures the initial distance once using the ultrasonic sensor function which returns the distance in centimeters.

In the void loop, the current time is set using the millis() function and the current distance is measured using the ultrasonic sensor function, such that if the difference between current and previous time is greater than the 100ms interval (which means 0.1 seconds have passed), the speed will be calculated by dividing the absolute value of difference of the current distance with the previous distance by 0.1 seconds. And before exiting the if statement, sets a new value for the previous time and previous distance as current time and current distance so it remains updated.

Finally, the distance that is constantly being measured in every loop is printed on the LCD display in cm on the first line, thus the cursor is set to (0,0), then the most recent speed which is calculated every 0.1 seconds is printed in cm/s on the second line of the LCD display, thus the cursor is set to (0,1). Lastly, the ultrasonic sensor function is basically measuring the duration taken by the echo to read from the trigger pin, then multiplies it by 0.034 / 2 to return the distance in centimeters. A flowchart is shown below describing the code process in Figure 6.

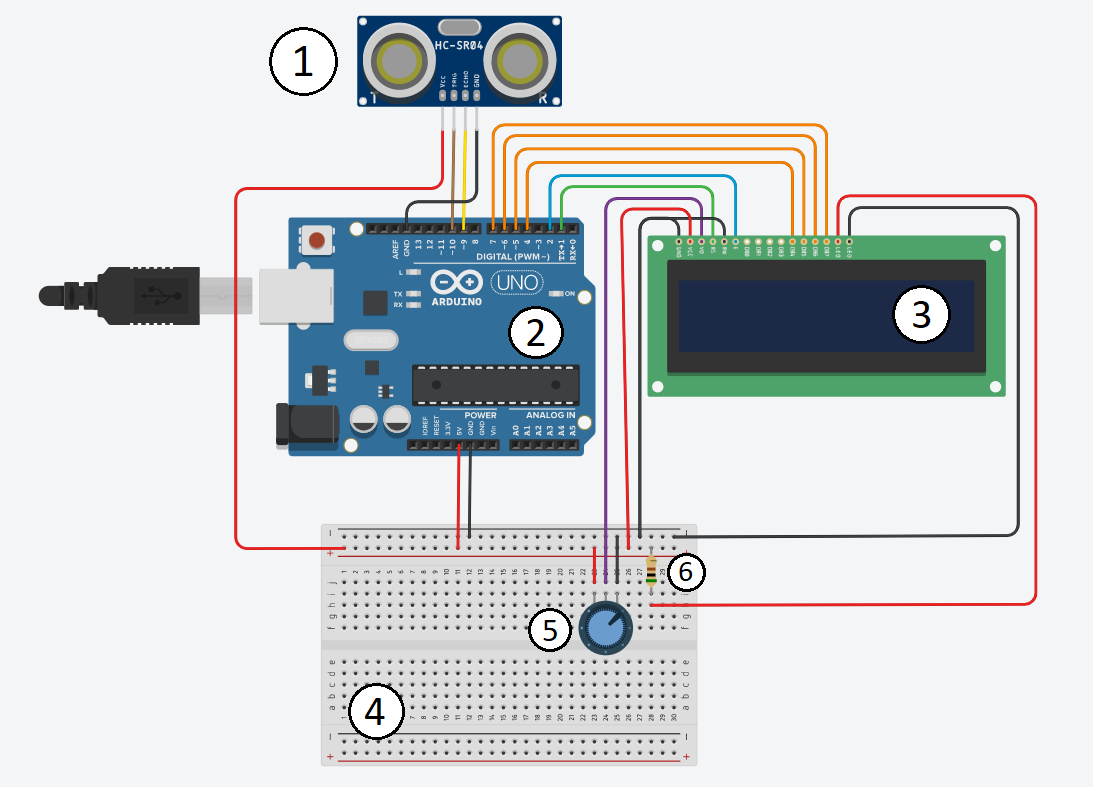


**FIGURE 6:** FLOWCHART

* + 1. **Connections on Tinker CAD**

1. First, we connected the 5V and ground pins of the Arduino Uno to the breadboard power and ground (red and black wires), and the 5V and ground pins of the Ultrasonic Sensor to the 5V and ground of the Arduino.
2. We connected the trigger pin of the ultrasonic sensor to pin 10 (brown wire), and the echo pin to pin 9 of the Arduino (yellow wire).
3. For the LCD, we connected the data pins D4, D5, D6, and D7 to the Arduino at pins 4, 5, 6, and 7 respectively to be able to print on the display (orange wires). The ground, R/W, and cathode pins are connected to the ground pin of the breadboard, and both the Vcc and anode of the LCD are connected to the 5V of the breadboard, however, the anode is first connected to a 500-ohm resistor to avoid damaging the screen.
4. The RS pin of the LCD is connected to pin 1 (green wire), and the Enable to pin 2 (blue wire), and finally V0 is connected to a potentiometer (purple wire) which is connected to the 5V and ground of the breadboard.
5. **RESULTS AND DISCUSSION**
   1. **Simulation**

We implemented the circuit on Tinker CAD, adding the code to the Arduino Simulation, obtaining the circuit shown in Figure 7.



**FIGURE 7:** CIRCUIT SCHEMATIC ON TINKER CAD

**TABLE 1:** LEGEND

* 1. **Analysis**

After running the simulation, the circuit showed the LCD displaying a speed of 0. After moving the object indicator in the ultrasonic sensor simulation, the LCD screen displayed a speed that is more than zero. We displayed the distance between the robot and the obstacle and the speed on the screen. We can also turn the knob of the potentiometer and get a variance in the LCD screen brightness. The video of the running simulation can be seen through [this link.](https://drive.google.com/file/d/1pUtAtAlRWDO_mAx8fhX30Z1lSPldk1O-/view?usp=sharing)

1. **CONCLUSION**

The objective of this experiment is to display the speed of a robot using an Ultrasonic sensor on an Arduino Uno and calculating the speed in the code. We used the Millis function in the Arduino Uno to get the difference in the distance every 0.1 seconds. The speed was calculated in the code using the basic formula (speed = Δ distance / Δ time) and displayed on the LCD screen. The experiment was successful, updating the speed every 0.1 seconds according to the readings.

**REFERENCES**

[1] Pololu Robotics & Electronics. *Arduino Uno*, 2020

https://www.pololu.com/product/2191#:~:text=Overview,header%2C%20and%20a%20reset%20button.

[2] Science Buddies. *How to Use a BreadBoard*, 2020

https://www.sciencebuddies.org/science-fair-projects/references/how-to-use-a-breadboard

[3] Tutorials Point, *Arduino – Ultrasonic Sensor*, 2020

https://www.tutorialspoint.com/arduino/arduino\_ultrasonic\_sensor.html

[4] Maker Guides, *How to use a 16×2 character LCD with Arduino*, 2020

https://www.makerguides.com/character-lcd-arduino-tutorial/

[5] Electronics Tutorials, *Potentiometers*, 2020

https://www.electronics-tutorials.ws/resistor/potentiometer.html

[6] Arduino Official Website  
 https://www.arduino.cc

**Appendix**

* + - 1. Arduino Code:

#include <LiquidCrystal.h>

LiquidCrystal lcd(1, 2, 4, 5, 6, 7); // Creates an LCD object. Parameters: (rs, enable, d4, d5, d6, d7)

const int trigPin = 10;

const int echoPin = 9;

const int interval = 100;

float previousdistance, speed = 0;

unsigned long previousmillis = 0;

void setup() {

lcd.begin(16,2); // Initializes the interface to the LCD screen, and specifies the dimensions (width and height) of the display

pinMode(trigPin, OUTPUT);

pinMode(echoPin, INPUT);

previousdistance = ultrasonic();

}

void loop() {

unsigned long currentmillis = millis();

float currentdistance = ultrasonic();

if(currentmillis - previousmillis >= interval){

speed = abs(currentdistance - previousdistance) / 0.1 ;

previousmillis = currentmillis;

previousdistance = currentdistance;

}

lcd.setCursor(0,0); // Sets the location at which subsequent text written to the LCD will be displayed

lcd.print("Dist: "); // Prints string "Distance" on the LCD

lcd.print(currentdistance); // Prints the distance value from the sensor

lcd.print(" cm ");

lcd.setCursor(0,1);

lcd.print("Speed: ");

lcd.print(speed);

lcd.print(" cm/s ");

}

float ultrasonic(){

long duration;

float distanceCm;

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

duration = pulseIn(echoPin, HIGH);

distanceCm = duration \* 0.034 / 2;

return distanceCm;

}