





# Wall-Aye

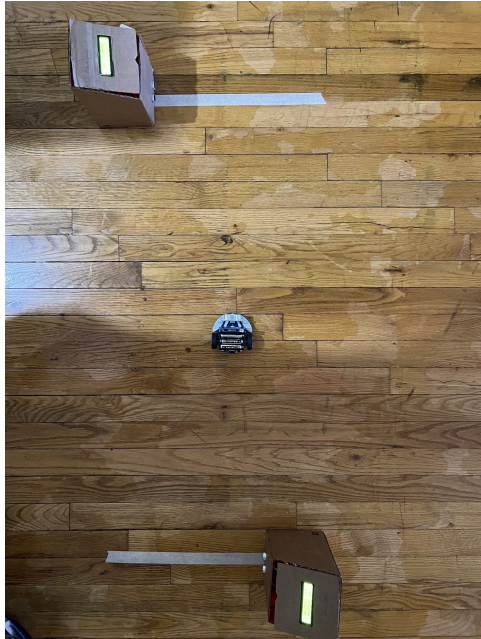
By Lea Mae Cruzat, Jacob Fernandez, Rilly Liu  
CET 4811 - Final Project  
Professor Zia



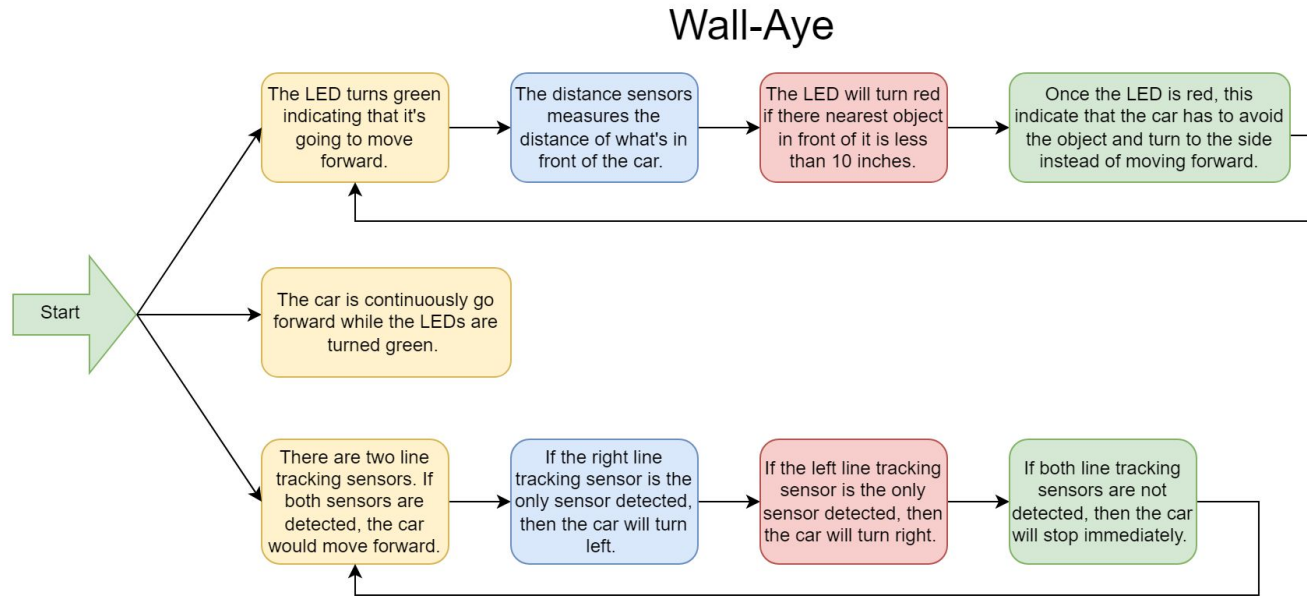
# Abstract/Summary

There are different modes to this project. The first mode of the car would be the like an avoiding function, whereas the car would move forward on a straight path, all the while the distance sensor measures if there is anything in front of it. If the object in front of the sensor is less than 10 inches, then the robot will turn to avoid the wall or any object in front of it. The second mode is just the car going autonomously, like it was in a race. The third mode is to follow a black line track using the two line tracking sensors. There would be a left and right line tracking sensors, and depending on which sensor is detected, the car would move with it. As the second mode is happening, there would be a time tracker to track the time the car leaves a starting line and the finish line. Instead of printing it in seconds, it will print the real time clock of when it left the starting line and arrives on the finish line.

# Finished Project - Image

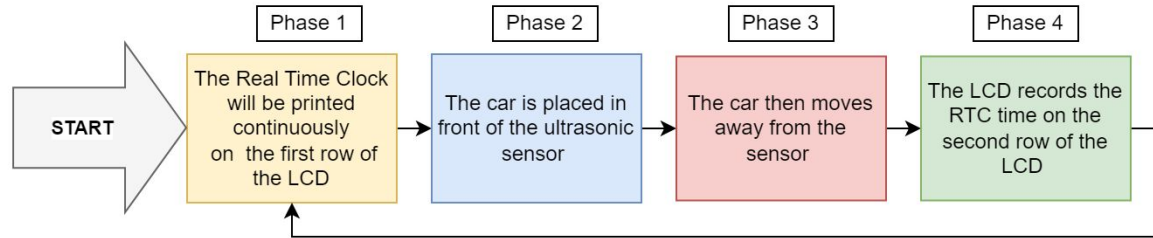


# Block Diagram

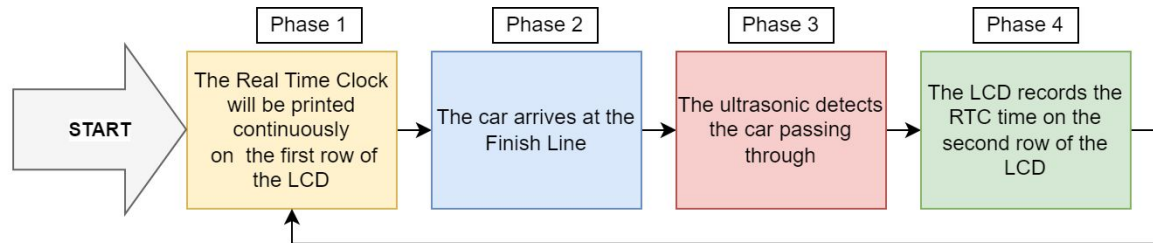


# Block Diagram

## Wall-Aye Start Line



## Wall-Aye Finish Line

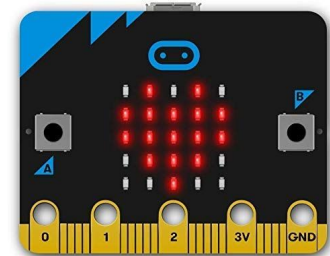
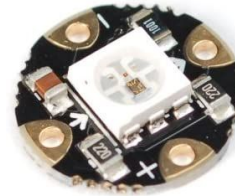


# Components



## Wall-Aye

- Line Tracking Sensors
- Ultrasonic Distance Sensor
- DC micro gear deceleration motors
- RGB LEDs
- Neopixel LEDs
- Micro:Bit



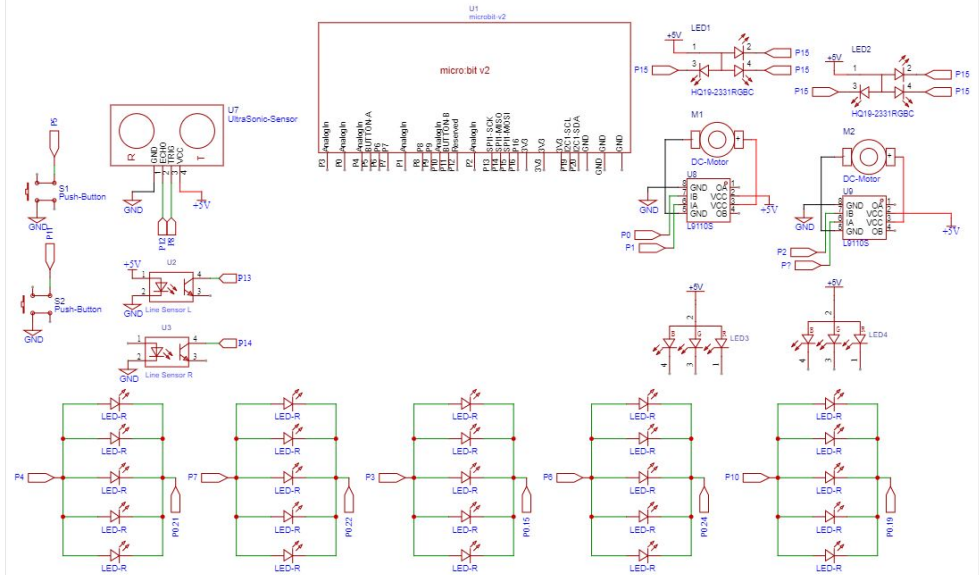
## Time Tracker:

- RTC
- Arduino Uno
- Ultrasonic Distance Sensor
- LCD



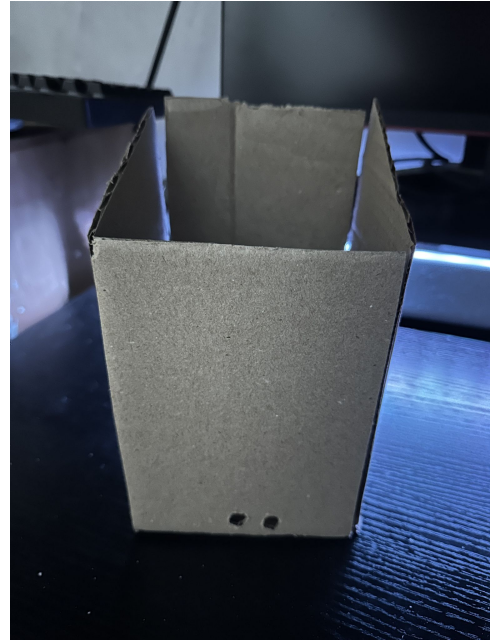
The diagram shows an Arduino-UNO microcontroller board connected to three external components:

- Ultrasonic HC SR04V:** A non-contact distance sensor. It is connected to the Arduino's 5V pin, GND pin, and TRIG pin. The sensor's output is connected to the Arduino's ECHO pin.
- RTC (Real Time Clock):** A DS1307 module. It is connected to the Arduino's 5V pin, GND pin, SCL pin, and SDA pin.
- LCD (Liquid Crystal Display):** A 16x2 character LCD. It is connected to the Arduino's 5V pin, GND pin, SCL pin, SDA pin, and RS pin. The LCD displays the text "HELLO WORLD!"



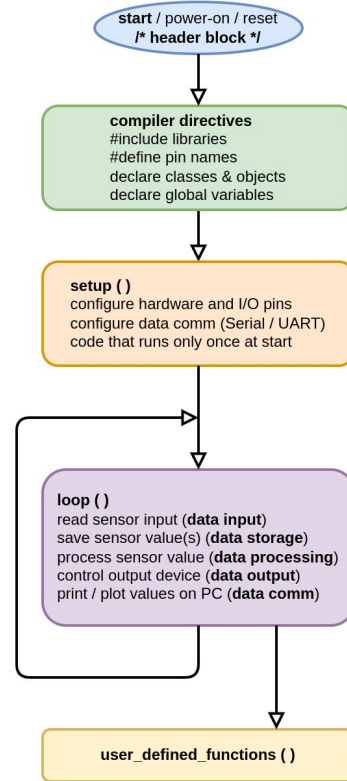
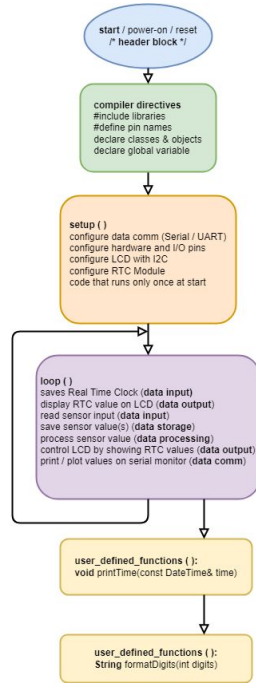


# Construction





# Flow Chart



# Software - Wall-Aye

```
let distance = 0
basic.showIcon(IconNames.EighthNote)
cuteBot.colorLight(cuteBot.RGBLights.ALL, 0xffffffff)
cuteBot.stopcar()
let strip = neopixel.create(DigitalPin.P15, 2, NeoPixelMode.RGB)
```

```
basic.forever(function () {
  distance = cuteBot.ultrasonic(cuteBot.SonarUnit.Inches)
  serial.writeValue("distance in inches", distance)
  // wait for 100 milliseconds before taking the next reading
  basic.pause(100)
  if (cuteBot.tracking(cuteBot.TrackingState.L_R_line)) {
    serial.writeString("Both probe is tracked, go straight!")
  } else if (cuteBot.tracking(cuteBot.TrackingState.L_unline_R_line)) {
    serial.writeString("Right probe is not detected, go left!")
  } else if (cuteBot.tracking(cuteBot.TrackingState.L_line_R_unline)) {
    serial.writeString("Left probe is not detected, go right!")
  } else {
    serial.writeString("Both probe is not detected, stop car!")
  }
})
```

# Software - Wall-Aye

```
cuteBot.colorLight(cuteBot.RGBLights.ALL, 0xffffffff)
strip.showColor(neopixel.colors(NeoPixelColors.Green))
basic.showIcon(IconNames.Cow)
if (cuteBot.ultrasonic(cuteBot.SonarUnit.Inches) <= 10) {
  cuteBot.colorLight(cuteBot.RGBLights.ALL, 0xff0000)
  cuteBot.moveTime(cuteBot.Direction.right, 60, 1)
} else {
  cuteBot.motors(98, 100)
  cuteBot.colorLight(cuteBot.RGBLights.ALL, 0x00ff00)
}
})
```

```
cuteBot.colorLight(cuteBot.RGBLights.ALL, 0xffffffff)
strip.showColor(neopixel.colors(NeoPixelColors.Green))
basic.showLeds(`
  . . # . .
  . # # # .
  # . . # #
  . . # . .
  . . # . .
  `)
cuteBot.motors(98, 100)
```

```
cuteBot.colorLight(cuteBot.RGBLights.ALL, 0xffffffff)
strip.showColor(neopixel.colors(NeoPixelColors.Green))
basic.showIcon(IconNames.Ghost)
cuteBot.colorLight(cuteBot.RGBLights.ALL, 0xffffffff)
if (cuteBot.tracking(cuteBot.TrackingState.L_R_line)) {
  serial.writeString("Both probe is tracked, go straight!")
  cuteBot.motors(40, 40)
} else if (cuteBot.tracking(cuteBot.TrackingState.L_unline_R_line)) {
  serial.writeString("Right probe is not detected, go left!")
  cuteBot.motors(40, 20)
} else if (cuteBot.tracking(cuteBot.TrackingState.L_line_R_unline)) {
  serial.writeString("Left probe is not detected, go right!")
  cuteBot.motors(20, 40)
} else {
  serial.writeString("Both probe is not detected, stop car!")
  cuteBot.stopcar()
}
```

# Software - Starting Line

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <RTCLib.h>

RTC_DS3231 rtc; // Create an RTC object

LiquidCrystal_I2C lcd(0x3F, 16, 2);

const int trigPin = 7; // Ultrasound Trigger Pin to Pin 7
const int echoPin = 6; // Ultrasound Echo Pin to Pin 6

bool objectDetected = false; // Flag to track object detection
unsigned long objectStart = 0; // Timestamp when object was detected
```

```
void setup() {
  Serial.begin(9600); // Begin serial communication
  pinMode(trigPin, OUTPUT); // Set trigger pin to output
  pinMode(echoPin, INPUT); // Set echo pin to input
  lcd.init(); // Initialize LCD
  lcd.backlight(); // Turn on LCD backlight
  lcd.setCursor(0, 0); // Set LCD cursor to the first line
  lcd.print("Time: "); // Print label for real-time clock
  Wire.begin(); // Begin I2C communication
  rtc.begin(); // Begin RTC module time count
}

if (!rtc.begin()) {
  Serial.println("Couldn't find RTC"); //if rtc module is not wired correctly, indicate as such
  while (1);
}

if (rtc.lostPower()) {
  Serial.println("RTC lost power, setting the time..."); //if rtc loses power or battery runs out, allow for adjusting time manually
  rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
}
```

# Software - Time Tracker (Starting Line)

```
DateTime now = rtc.now(); // Get current time from RTC module
lcd.setCursor(6, 0); // Set cursor position to print real-time clock
printTime(now); // Print real-time clock on LCD and serial monitor

// Trigger ultrasonic sensor and check for object detection
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
long duration = pulseIn(echoPin, HIGH);
float distance = duration * 0.034 / 2;
```

```
// If an object is detected, store the timestamp
if (distance > 0 && distance < 400) { // Modify measurement as needed
    if (!objectDetected) {
        objectStart = now.unixtime();
        objectDetected = true;
    }
} else {
    // If no object is detected and object was previously detected, print the time when the object was removed
    if (objectDetected) {
        objectDetected = false;
        lcd.setCursor(0, 1); // Set cursor to the second line of the LCD
        lcd.print("Dash: "); // Print label for object removal time
        printTime(DateTime(objectStart)); // Print time when the object was removed
    }
}
```

# Software - Time Tracker (Starting Line)

```
void printTime(const DateTime& time) {  
    lcd.print(formatDigits(time.hour())); // Print hours  
    lcd.print(':');  
    lcd.print(formatDigits(time.minute())); // Print minutes  
    lcd.print(':');  
    lcd.print(formatDigits(time.second())); // Print seconds  
}  
  
String formatDigits(int digits) {  
    String formattedDigits = String(digits);  
    if (digits < 10) {  
        formattedDigits = '0' + formattedDigits; // Add leading zero if the value is less than 10  
    }  
    return formattedDigits;  
}
```

# Software - Time Tracker (Finish Line)

```
// If an object is removed, store the timestamp
if (distance > 10) { // Modify measurement as needed
    if (objectDetected) {
        objectDetected = false;
        lcd.setCursor(0, 1); // Set cursor to the second line of the LCD
        lcd.print("                "); // Clear the line
    }
} else {
    if (!objectDetected) {
        objectStart = now.unixtime();
        objectDetected = true;
        lcd.setCursor(0, 1); // Set cursor to the second line of the LCD
        lcd.print("Finish: "); // Print label for object detection time
        printTime(DateTime(objectStart)); // Print time when the object was detected
    }
}
```



# Troubleshooting

- Error with showing Real Time Clock on the Liquid Crystal Display
  - Fixed the format by showing the “0” in front if the time hit one digit numbers
- The Ultrasonic sensor on the Finish Line since it was not detecting an object
  - Ran tests to see the communication between the sensor and serial monitor
  - Replaced the ultrasonic sensor
- Liquid Crystal Display did not display values
  - By using a Phillip screw, we can turn the potentiometer on the I2C to control the brightness of the LCD

# Conclusion

This project could be useful in real life by implementing the same kind of system with autonomous cars, which could potentially help those who cannot drive. With this project, micro:bit could be better understood, which would help future projects that could be sufficiently and efficiently done with micro:bit. Many have tried to make an autonomous car using micro:bit and Arduino. It is accomplished with various components such as distance sensors and light sensors, and tracking sensors, much in the same fashion as we are attempting to do. In terms of making the car autonomous, people have tried to solve it the same way, by using distance sensors and the other sensors mentioned earlier. There are also some people who use flex sensors. There are many ways of building the project we are doing, and micro:bit is proving to be an efficient microcontroller for such a project.

The utilization of an ultrasonic sensor and the Real Time Clock (RTC) module presents various applications to accurately measure the time taken by an object from traveling from one starting point to another. A practical example of this would be in the field of race cars: race car timing systems. By implementing the use of the ultrasonic sensor and the RTC, the possibility of precisely tracking race cars as they pass the starting and finish line increases. This combination allows for an efficient and reliable way of recording the time it takes to arrive from the starting line and pass the finish line. We can expand upon this and change certain components, but keep the core idea and could implement this project into other devices that record time by detecting an object's movement.

# Reference

- [1] E. Team, "ElecFreaks Wiki," 1. Introduction to Cutebot - ELECFREAKS WIKI, [https://www.electfreaks.com/learn-en/microbitKit/smart\\_cutebot/cutebot\\_car.html](https://www.electfreaks.com/learn-en/microbitKit/smart_cutebot/cutebot_car.html) (accessed May 17, 2023).
- [2] J. Mallari, "How to use a real-time clock module with the Arduino," Circuit Basics, <https://www.circuitbasics.com/how-to-use-a-real-time-clock-module-with-the-arduino/> (accessed May 17, 2023).
- [3] Microsoft, "Micro:bit V2," Microsoft MakeCode, <https://makecode.microbit.org/device/v2> (accessed May 17, 2023).
- [4] SFUPTOWNMAKER, "I2C," I2C - SparkFun Learn, <https://learn.sparkfun.com/tutorials/i2c/all> (accessed May 17, 2023).

# Video Presentation

