

RESEARCH ON DEVELOPING AND BUILDING AN AUTONOMOUS ROBOT FOR SURFACE CLEANING

CRIȘU Dragoș-Constantin, BĂLAN Andrei-Teodor, COJOCARU Michael, LUPU Adrian-Cătălin, MARINESCU Marinela Nicoleta, BUȚU Larisa

Faculty: Faculty of Industrial Engineering and Robotics, Specialization: Robotics, Year of Study: I,
e-mail: dragos.crisu@gmail.com

The study "Research on developing and building of an autonomous robot for cleaning surfaces" aims to develop an autonomous robot capable of industrial cleaning large surfaces after various processes, such as the machining process. The robot will be equipped with a sensor system and navigation algorithms that will allow it to avoid obstacles and move autonomously in the workspace. Additionally, it will be equipped with an efficient cleaning system that can be adapted to different types of surfaces. A prototype of the robot has been built on a limited budget, and the cleaning brushes will be interchangeable to meet different cleaning needs. Developing such an autonomous robot could bring multiple benefits to the metal processing industry, such as reducing cleaning time and cost, as well as improving working conditions for employees.

KEYWORDS: *autonomous robot, sensor system and navigation algorithms, prototype of the robot, benefits to the metal processing industry*

1. Introduction

Certainly, robotics is one of the most interesting and promising branches of modern engineering. In recent years, autonomous robots have started to penetrate various fields, from the automotive and aerospace industries to medicine and space exploration. In this context, research on the design and development of an autonomous surface cleaning robot is particularly important in the metal processing industry, where surface cleaning is a recurring and necessary task.

Our objective was to develop an autonomous robot capable of performing industrial cleaning on large surfaces after various processes, such as the machining process. For this purpose, we used a scaled-down prototype constructed with different materials and equipped it with Arduino as the microcontroller and front sensors. We programmed the Arduino to enable the robot to operate using its onboard sensors and perform surface cleaning efficiently and autonomously.



Fig. 1. Arduino Microcontroller

2. Current stage

The current stage of research regarding the design and development of an autonomous surface cleaning robot has led to the development of a prototype robot capable of performing industrial cleaning operations. This robot is constructed using various components such as Arduino, ultrasonic sensors, and an efficient cleaning system [3].

Over the past few years, research in the field of robotics has progressed rapidly, enabling the development of increasingly advanced robots. Regarding cleaning robots, the market is already saturated with models that are manually or semi-automatically controlled but lack the ability to autonomously navigate and perform cleaning operations on large surfaces.

The prototype of the autonomous surface cleaning robot developed within this research represents an innovative solution to this problem. By utilizing a system of sensors and navigation algorithms, this robot can avoid obstacles and autonomously navigate in the working space.

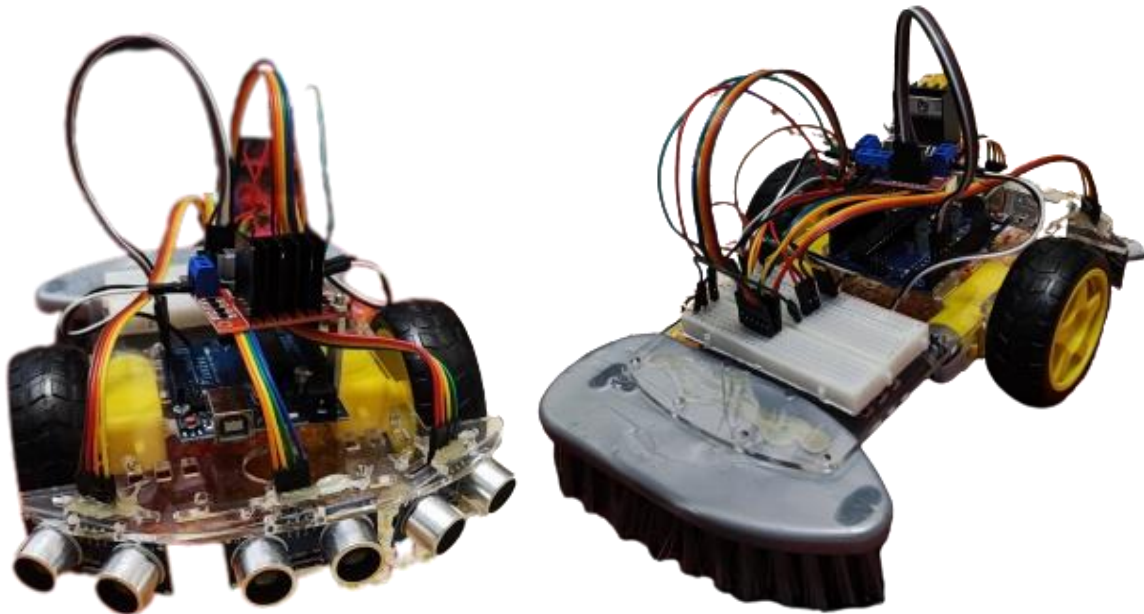


Fig. 2. Prototipul de robot autonom

In the robot's code, we have utilized a multitude of functions to achieve the current efficiency of the autonomous prototype robot. An important equation used in the prototype's code is equation (1), where d represents the distance at which the sensor is from the obstacle, expressed in centimeters, Δt is the time in which the sensor receives the reflected sound from the obstacle. We multiply it by 0.034 to convert the speed of sound to centimeters and divide by 2 for one-way duration since the sound needs to travel back to the sensor to provide information, similar to echolocation. Additionally, lines of code (2,3) are crucial to the functioning of the prototype as they implement a way to avoid obstacles. These lines of code check if the distance detected by the front sensor is smaller than the maximum allowed distance and if the distances detected by the side sensors are also smaller than the maximum allowed distance. If these conditions are met, the prototype is programmed to avoid the obstacles by rotating in the opposite direction of the obstacle and then moving forward. This approach enables the prototype to detect obstacles in a timely manner and efficiently avoid them, preventing potential collisions and damage to the robot.

As improvements, we intend to incorporate infrared sensors in the next version of the prototype as they offer higher performance and lower margin of error compared to the current ultrasonic sensors. We also plan to replace the brush at the back of the robot with a rotating brush to further enhance the cleaning process. Optimizing the code, replacing Arduino with a more powerful microcontroller or even a mini-computer are also part of our plans. Additionally, to accommodate these enhancements, we will install a more powerful motor to compensate for the added weight.

$$D = \frac{\Delta t \cdot 0.034}{2} \quad (1)$$

$$\text{distancef} < \text{distance_max} \ \&\& \ \text{distanced} < \text{distance_max} \quad (2)$$

$$\text{distancef} < \text{distance_max} \ \&\& \ \text{distances} < \text{distance_max} \quad (3)$$

3. Robot Production Technology

The prototype of the autonomous surface cleaning robot was developed using a series of key components to ensure proper functionality and performance. Among the components used are an Arduino board, three HC-SR04 ultrasonic sensors, an L298N motor driver, and two motors. These components were carefully selected to meet the specific requirements of the robot and enable precise control of movement and obstacle detection in the working environment.

An important aspect of the prototype is its power system. It utilizes two 1850mAh batteries, which provide sufficient energy for the robot to operate for an extended period of time. The choice of suitable batteries was crucial to ensure the autonomy and mobility of the robot during the surface cleaning process.

Regarding the cleaning brush, it should be mentioned that the prototype was equipped with a standard brush, which is commonly available in the market. Due to cost constraints during the prototype development stage, we did not have the opportunity to order or design a specialized brush tailored to the specific needs of the robot. However, even with a standard brush, the prototype has demonstrated satisfactory capabilities in terms of surface cleaning.

4. Construction stages

I. Hardware Development

We initiated the construction process of the autonomous surface cleaning robot by starting with a robot chassis kit [2], which we purchased at a budget-friendly price from a specialized robot parts website. This kit included two electric motors and the mounting system for the chassis, providing us with a sturdy foundation for building the robot. Additionally, the kit included a small wheel, similar to a caster wheel, which served the purpose of supporting and balancing the robot. However, we decided to replace the wheel with a brush, considering its dual role in our prototype: both cleaning and supporting the robot.

The first step in constructing this robot was to mount the Arduino board to facilitate any potential replacements in case of short circuits or malfunctions. The Arduino board was securely attached to the chassis using two screws, ensuring stability and accessibility. The next step involved mounting a breadboard, a board that facilitates connections and allows for modifications within the prototype. To ensure power supply, we proceeded with mounting the power source. Due to space constraints, we opted to mount the power source on the lower part of the chassis, with a switch directly connected to it.

Another crucial step was the installation of the motor driver. Initially, we considered using an L298D motor driver shield, which would have allowed us to connect multiple motors for future enhancements. However, we encountered an issue with the shield blocking all digital pins. Our solution was to use a standard motor driver, L298N. This alternative allowed us to bypass the restrictions imposed by the previously mentioned shield. Regarding the installation of the motor driver, we faced the same challenge as with the power source, but we found a solution by mounting it above the Arduino board using a pillar to ensure stable fixation.

The last hardware components mounted on the chassis were the three HC-SR04 ultrasonic sensors, placed underneath the chassis for better visibility. One sensor was oriented directly in the direction of the robot's movement, while the other two were mounted laterally to cover a wider angle of visibility. After mounting all the components on the chassis, we proceeded to establish the connections between them (Fig. 3.). This process involved connecting the Arduino board to the motor driver, ultrasonic sensors, and power source, ensuring correct interaction and efficient operation of all components.

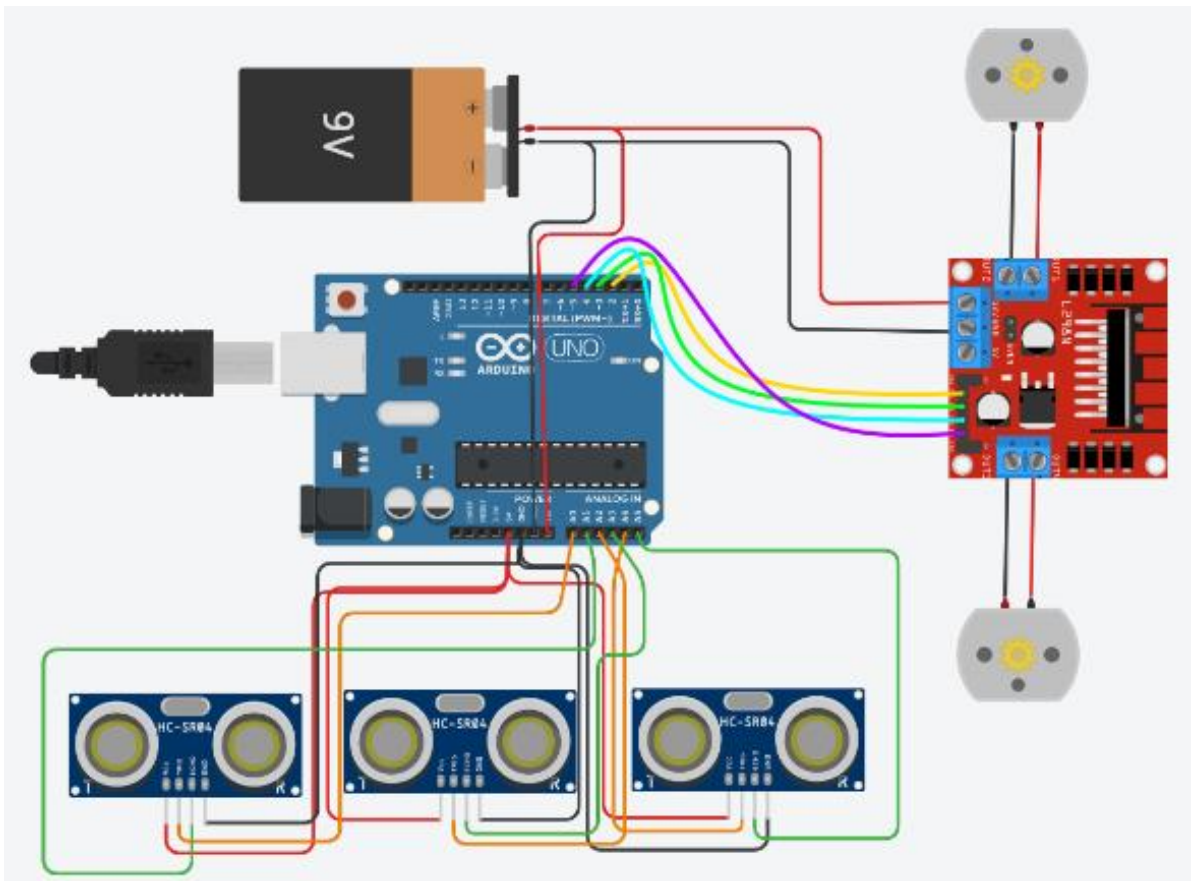


Fig. 3. Wiring diagram

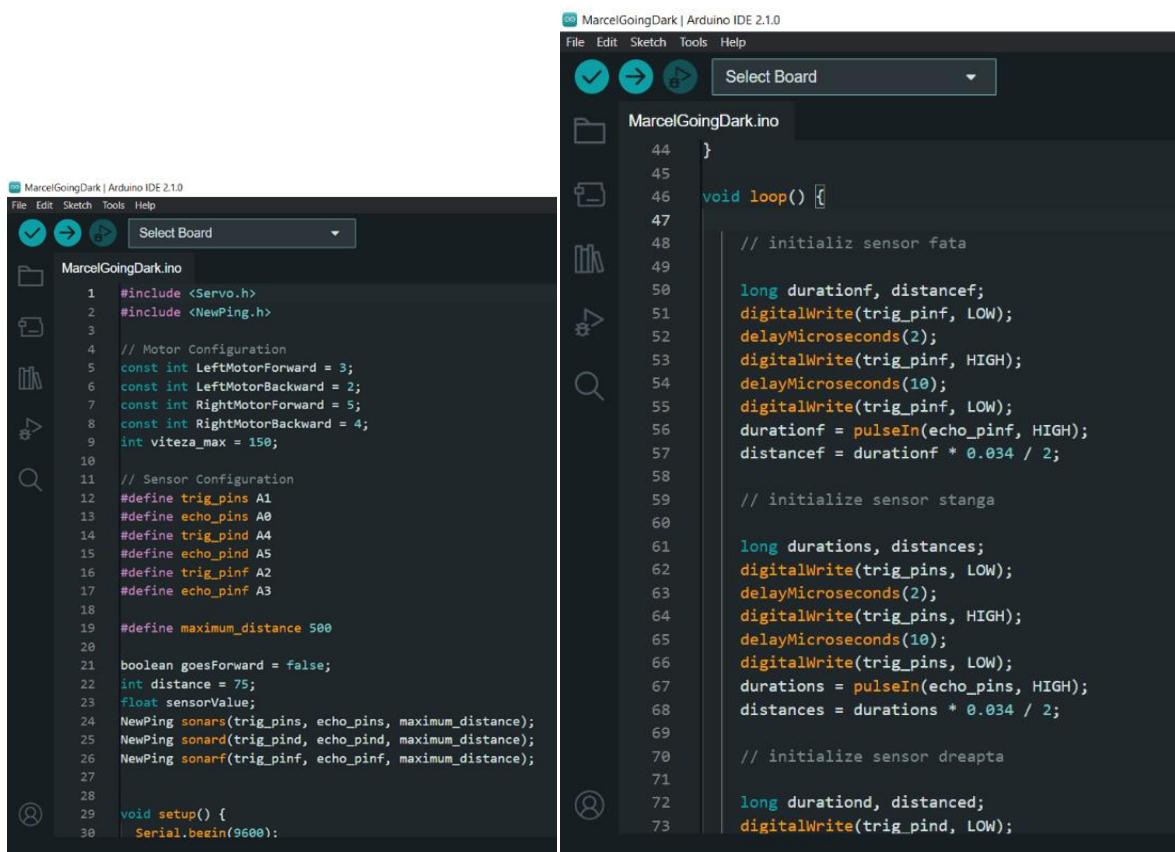
II. Software Development

Regarding the software part, we used two essential libraries for controlling the motors and ultrasonic sensors, namely the "Servo.h" and "NewPing.h" libraries [2]. These libraries provided us with the necessary functionalities to interact with these key components of our autonomous robot.

To control the movement of the motors, we initialized each motor using a specific pin and set a maximum speed for the robot. We also defined and properly initialized the ultrasonic sensors in the code. Using the "Servo.h" library, we assigned a specific role to each pin of the Arduino board, whether it was an input or output, to ensure the correct functioning of the components.

In the next stage, we implemented a function that continuously repeated during the Arduino board's power supply. This function started by reading the values provided by the sensors and calculating the distance in centimeters. Then, we had a series of comparisons between the read distance and the maximum allowed distance, and based on the obtained results, the corresponding movements were initiated to avoid obstacles encountered by the robot.

Throughout the software development process, we encountered a few specific issues. The first problem was related to correctly reading the values from the sensors, and to solve it, we had to repeatedly rewrite the code responsible for reading them. The second issue we faced was accurately instructing the robot and making precise comparisons between the read distance values. This problem was also addressed by repeatedly rewriting the code to achieve the desired results and ensure the optimal functioning of the prototype.



```
1 #include <Servo.h>
2 #include <NewPing.h>
3
4 // Motor Configuration
5 const int LeftMotorForward = 3;
6 const int LeftMotorBackward = 2;
7 const int RightMotorForward = 5;
8 const int RightMotorBackward = 4;
9 int vitez_max = 150;
10
11 // Sensor Configuration
12 #define trig_pins A1
13 #define echo_pins A0
14 #define trig_pind A4
15 #define echo_pind A5
16 #define trig_pinf A2
17 #define echo_pinf A3
18
19 #define maximum_distance 500
20
21 boolean goesForward = false;
22 int distance = 75;
23 float sensorValue;
24 NewPing sonars(trig_pins, echo_pins, maximum_distance);
25 NewPing sonard(trig_pind, echo_pind, maximum_distance);
26 NewPing sonarf(trig_pinf, echo_pinf, maximum_distance);
27
28
29 void setup() {
30   Serial.begin(9600);
31
32   // Motor Configuration
33   pinMode(LeftMotorForward, OUTPUT);
34   pinMode(LeftMotorBackward, OUTPUT);
35   pinMode(RightMotorForward, OUTPUT);
36   pinMode(RightMotorBackward, OUTPUT);
37
38   // Sensor Configuration
39   pinMode(trig_pins, OUTPUT);
40   pinMode(echo_pins, INPUT);
41   pinMode(trig_pind, OUTPUT);
42   pinMode(echo_pind, INPUT);
43   pinMode(trig_pinf, OUTPUT);
44   pinMode(echo_pinf, INPUT);
45
46   // initialize sensor stanga
47   long durationf, distancef;
48   digitalWrite(trig_pinf, LOW);
49   delayMicroseconds(2);
50   digitalWrite(trig_pinf, HIGH);
51   delayMicroseconds(10);
52   digitalWrite(trig_pinf, LOW);
53   durationf = pulseIn(echo_pinf, HIGH);
54   distancef = durationf * 0.034 / 2;
55
56   // initialize sensor dreapta
57   long durations, distances;
58   digitalWrite(trig_pins, LOW);
59   delayMicroseconds(2);
60   digitalWrite(trig_pins, HIGH);
61   delayMicroseconds(10);
62   digitalWrite(trig_pins, LOW);
63   durations = pulseIn(echo_pins, HIGH);
64   distances = durations * 0.034 / 2;
65
66   // initialize sensor stanga
67   digitalWrite(trig_pind, LOW);
68   delayMicroseconds(2);
69   digitalWrite(trig_pind, HIGH);
70   delayMicroseconds(10);
71   digitalWrite(trig_pind, LOW);
72   durationd = pulseIn(echo_pind, HIGH);
73   distanced = durationd * 0.034 / 2;
74
75   // initialize sensor dreapta
76   digitalWrite(trig_pins, LOW);
77   delayMicroseconds(2);
78   digitalWrite(trig_pins, HIGH);
79   delayMicroseconds(10);
80   digitalWrite(trig_pins, LOW);
81   durationr = pulseIn(echo_pins, HIGH);
82   distancer = durationr * 0.034 / 2;
83
84   // initialize motor
85   digitalWrite(LeftMotorForward, LOW);
86   digitalWrite(LeftMotorBackward, LOW);
87   digitalWrite(RightMotorForward, LOW);
88   digitalWrite(RightMotorBackward, LOW);
89
90   // initialize servo
91   servo.attach(1);
92   servo.write(90);
93
94   // initialize buzzer
95   pinMode(buzzerPin, OUTPUT);
96   digitalWrite(buzzerPin, LOW);
97
98   // initialize led
99   pinMode(ledPin, OUTPUT);
100  digitalWrite(ledPin, LOW);
101
102  // initialize display
103  display.begin(16, 2);
104  display.clear();
105  display.setCursor(0, 0);
106  display.print(" ");
107
108  // initialize variables
109  goesForward = false;
110  distance = 75;
111  sensorValue = 0;
112
113  // initialize variables
114  durationf = 0;
115  distancef = 0;
116  durations = 0;
117  distances = 0;
118  durationd = 0;
119  distanced = 0;
120  durationr = 0;
121  distancer = 0;
122
123  // initialize variables
124  digitalWrite(LeftMotorForward, LOW);
125  digitalWrite(LeftMotorBackward, LOW);
126  digitalWrite(RightMotorForward, LOW);
127  digitalWrite(RightMotorBackward, LOW);
128
129  // initialize variables
130  digitalWrite(trig_pins, LOW);
131  digitalWrite(trig_pind, LOW);
132  digitalWrite(trig_pinf, LOW);
133
134  // initialize variables
135  pinMode(echo_pins, INPUT);
136  pinMode(echo_pind, INPUT);
137  pinMode(echo_pinf, INPUT);
138
139  // initialize variables
140  pinMode(buzzerPin, OUTPUT);
141  pinMode(ledPin, OUTPUT);
142
143  // initialize variables
144  display.begin(16, 2);
145  display.clear();
146  display.setCursor(0, 0);
147  display.print(" ");
148
149  // initialize variables
150  goesForward = false;
151  distance = 75;
152  sensorValue = 0;
153
154  // initialize variables
155  durationf = 0;
156  distancef = 0;
157  durations = 0;
158  distances = 0;
159  durationd = 0;
160  distanced = 0;
161  durationr = 0;
162  distancer = 0;
163
164  // initialize variables
165  digitalWrite(LeftMotorForward, LOW);
166  digitalWrite(LeftMotorBackward, LOW);
167  digitalWrite(RightMotorForward, LOW);
168  digitalWrite(RightMotorBackward, LOW);
169
170  // initialize variables
171  digitalWrite(trig_pins, LOW);
172  digitalWrite(trig_pind, LOW);
173  digitalWrite(trig_pinf, LOW);
174
175  // initialize variables
176  pinMode(echo_pins, INPUT);
177  pinMode(echo_pind, INPUT);
178  pinMode(echo_pinf, INPUT);
179
180  // initialize variables
181  pinMode(buzzerPin, OUTPUT);
182  pinMode(ledPin, OUTPUT);
183
184  // initialize variables
185  display.begin(16, 2);
186  display.clear();
187  display.setCursor(0, 0);
188  display.print(" ");
189
190  // initialize variables
191  goesForward = false;
192  distance = 75;
193  sensorValue = 0;
194
195  // initialize variables
196  durationf = 0;
197  distancef = 0;
198  durations = 0;
199  distances = 0;
200  durationd = 0;
201  distanced = 0;
202  durationr = 0;
203  distancer = 0;
204
205  // initialize variables
206  digitalWrite(LeftMotorForward, LOW);
207  digitalWrite(LeftMotorBackward, LOW);
208  digitalWrite(RightMotorForward, LOW);
209  digitalWrite(RightMotorBackward, LOW);
210
211  // initialize variables
212  digitalWrite(trig_pins, LOW);
213  digitalWrite(trig_pind, LOW);
214  digitalWrite(trig_pinf, LOW);
215
216  // initialize variables
217  pinMode(echo_pins, INPUT);
218  pinMode(echo_pind, INPUT);
219  pinMode(echo_pinf, INPUT);
220
221  // initialize variables
222  pinMode(buzzerPin, OUTPUT);
223  pinMode(ledPin, OUTPUT);
224
225  // initialize variables
226  display.begin(16, 2);
227  display.clear();
228  display.setCursor(0, 0);
229  display.print(" ");
230
231  // initialize variables
232  goesForward = false;
233  distance = 75;
234  sensorValue = 0;
235
236  // initialize variables
237  durationf = 0;
238  distancef = 0;
239  durations = 0;
240  distances = 0;
241  durationd = 0;
242  distanced = 0;
243  durationr = 0;
244  distancer = 0;
245
246  // initialize variables
247  digitalWrite(LeftMotorForward, LOW);
248  digitalWrite(LeftMotorBackward, LOW);
249  digitalWrite(RightMotorForward, LOW);
250  digitalWrite(RightMotorBackward, LOW);
251
252  // initialize variables
253  digitalWrite(trig_pins, LOW);
254  digitalWrite(trig_pind, LOW);
255  digitalWrite(trig_pinf, LOW);
256
257  // initialize variables
258  pinMode(echo_pins, INPUT);
259  pinMode(echo_pind, INPUT);
260  pinMode(echo_pinf, INPUT);
261
262  // initialize variables
263  pinMode(buzzerPin, OUTPUT);
264  pinMode(ledPin, OUTPUT);
265
266  // initialize variables
267  display.begin(16, 2);
268  display.clear();
269  display.setCursor(0, 0);
270  display.print(" ");
271
272  // initialize variables
273  goesForward = false;
274  distance = 75;
275  sensorValue = 0;
276
277  // initialize variables
278  durationf = 0;
279  distancef = 0;
280  durations = 0;
281  distances = 0;
282  durationd = 0;
283  distanced = 0;
284  durationr = 0;
285  distancer = 0;
286
287  // initialize variables
288  digitalWrite(LeftMotorForward, LOW);
289  digitalWrite(LeftMotorBackward, LOW);
290  digitalWrite(RightMotorForward, LOW);
291  digitalWrite(RightMotorBackward, LOW);
292
293  // initialize variables
294  digitalWrite(trig_pins, LOW);
295  digitalWrite(trig_pind, LOW);
296  digitalWrite(trig_pinf, LOW);
297
298  // initialize variables
299  pinMode(echo_pins, INPUT);
300  pinMode(echo_pind, INPUT);
301  pinMode(echo_pinf, INPUT);
302
303  // initialize variables
304  pinMode(buzzerPin, OUTPUT);
305  pinMode(ledPin, OUTPUT);
306
307  // initialize variables
308  display.begin(16, 2);
309  display.clear();
310  display.setCursor(0, 0);
311  display.print(" ");
312
313  // initialize variables
314  goesForward = false;
315  distance = 75;
316  sensorValue = 0;
317
318  // initialize variables
319  durationf = 0;
320  distancef = 0;
321  durations = 0;
322  distances = 0;
323  durationd = 0;
324  distanced = 0;
325  durationr = 0;
326  distancer = 0;
327
328  // initialize variables
329  digitalWrite(LeftMotorForward, LOW);
330  digitalWrite(LeftMotorBackward, LOW);
331  digitalWrite(RightMotorForward, LOW);
332  digitalWrite(RightMotorBackward, LOW);
333
334  // initialize variables
335  digitalWrite(trig_pins, LOW);
336  digitalWrite(trig_pind, LOW);
337  digitalWrite(trig_pinf, LOW);
338
339  // initialize variables
340  pinMode(echo_pins, INPUT);
341  pinMode(echo_pind, INPUT);
342  pinMode(echo_pinf, INPUT);
343
344  // initialize variables
345  pinMode(buzzerPin, OUTPUT);
346  pinMode(ledPin, OUTPUT);
347
348  // initialize variables
349  display.begin(16, 2);
350  display.clear();
351  display.setCursor(0, 0);
352  display.print(" ");
353
354  // initialize variables
355  goesForward = false;
356  distance = 75;
357  sensorValue = 0;
358
359  // initialize variables
360  durationf = 0;
361  distancef = 0;
362  durations = 0;
363  distances = 0;
364  durationd = 0;
365  distanced = 0;
366  durationr = 0;
367  distancer = 0;
368
369  // initialize variables
370  digitalWrite(LeftMotorForward, LOW);
371  digitalWrite(LeftMotorBackward, LOW);
372  digitalWrite(RightMotorForward, LOW);
373  digitalWrite(RightMotorBackward, LOW);
374
375  // initialize variables
376  digitalWrite(trig_pins, LOW);
377  digitalWrite(trig_pind, LOW);
378  digitalWrite(trig_pinf, LOW);
379
380  // initialize variables
381  pinMode(echo_pins, INPUT);
382  pinMode(echo_pind, INPUT);
383  pinMode(echo_pinf, INPUT);
384
385  // initialize variables
386  pinMode(buzzerPin, OUTPUT);
387  pinMode(ledPin, OUTPUT);
388
389  // initialize variables
390  display.begin(16, 2);
391  display.clear();
392  display.setCursor(0, 0);
393  display.print(" ");
394
395  // initialize variables
396  goesForward = false;
397  distance = 75;
398  sensorValue = 0;
399
400  // initialize variables
401  durationf = 0;
402  distancef = 0;
403  durations = 0;
404  distances = 0;
405  durationd = 0;
406  distanced = 0;
407  durationr = 0;
408  distancer = 0;
409
410  // initialize variables
411  digitalWrite(LeftMotorForward, LOW);
412  digitalWrite(LeftMotorBackward, LOW);
413  digitalWrite(RightMotorForward, LOW);
414  digitalWrite(RightMotorBackward, LOW);
415
416  // initialize variables
417  digitalWrite(trig_pins, LOW);
418  digitalWrite(trig_pind, LOW);
419  digitalWrite(trig_pinf, LOW);
420
421  // initialize variables
422  pinMode(echo_pins, INPUT);
423  pinMode(echo_pind, INPUT);
424  pinMode(echo_pinf, INPUT);
425
426  // initialize variables
427  pinMode(buzzerPin, OUTPUT);
428  pinMode(ledPin, OUTPUT);
429
430  // initialize variables
431  display.begin(16, 2);
432  display.clear();
433  display.setCursor(0, 0);
434  display.print(" ");
435
436  // initialize variables
437  goesForward = false;
438  distance = 75;
439  sensorValue = 0;
440
441  // initialize variables
442  durationf = 0;
443  distancef = 0;
444  durations = 0;
445  distances = 0;
446  durationd = 0;
447  distanced = 0;
448  durationr = 0;
449  distancer = 0;
450
451  // initialize variables
452  digitalWrite(LeftMotorForward, LOW);
453  digitalWrite(LeftMotorBackward, LOW);
454  digitalWrite(RightMotorForward, LOW);
455  digitalWrite(RightMotorBackward, LOW);
456
457  // initialize variables
458  digitalWrite(trig_pins, LOW);
459  digitalWrite(trig_pind, LOW);
460  digitalWrite(trig_pinf, LOW);
461
462  // initialize variables
463  pinMode(echo_pins, INPUT);
464  pinMode(echo_pind, INPUT);
465  pinMode(echo_pinf, INPUT);
466
467  // initialize variables
468  pinMode(buzzerPin, OUTPUT);
469  pinMode(ledPin, OUTPUT);
470
471  // initialize variables
472  display.begin(16, 2);
473  display.clear();
474  display.setCursor(0, 0);
475  display.print(" ");
476
477  // initialize variables
478  goesForward = false;
479  distance = 75;
480  sensorValue = 0;
481
482  // initialize variables
483  durationf = 0;
484  distancef = 0;
485  durations = 0;
486  distances = 0;
487  durationd = 0;
488  distanced = 0;
489  durationr = 0;
490  distancer = 0;
491
492  // initialize variables
493  digitalWrite(LeftMotorForward, LOW);
494  digitalWrite(LeftMotorBackward, LOW);
495  digitalWrite(RightMotorForward, LOW);
496  digitalWrite(RightMotorBackward, LOW);
497
498  // initialize variables
499  digitalWrite(trig_pins, LOW);
500  digitalWrite(trig_pind, LOW);
501  digitalWrite(trig_pinf, LOW);
502
503  // initialize variables
504  pinMode(echo_pins, INPUT);
505  pinMode(echo_pind, INPUT);
506  pinMode(echo_pinf, INPUT);
507
508  // initialize variables
509  pinMode(buzzerPin, OUTPUT);
510  pinMode(ledPin, OUTPUT);
511
512  // initialize variables
513  display.begin(16, 2);
514  display.clear();
515  display.setCursor(0, 0);
516  display.print(" ");
517
518  // initialize variables
519  goesForward = false;
520  distance = 75;
521  sensorValue = 0;
522
523  // initialize variables
524  durationf = 0;
525  distancef = 0;
526  durations = 0;
527  distances = 0;
528  durationd = 0;
529  distanced = 0;
530  durationr = 0;
531  distancer = 0;
532
533  // initialize variables
534  digitalWrite(LeftMotorForward, LOW);
535  digitalWrite(LeftMotorBackward, LOW);
536  digitalWrite(RightMotorForward, LOW);
537  digitalWrite(RightMotorBackward, LOW);
538
539  // initialize variables
540  digitalWrite(trig_pins, LOW);
541  digitalWrite(trig_pind, LOW);
542  digitalWrite(trig_pinf, LOW);
543
544  // initialize variables
545  pinMode(echo_pins, INPUT);
546  pinMode(echo_pind, INPUT);
547  pinMode(echo_pinf, INPUT);
548
549  // initialize variables
550  pinMode(buzzerPin, OUTPUT);
551  pinMode(ledPin, OUTPUT);
552
553  // initialize variables
554  display.begin(16, 2);
555  display.clear();
556  display.setCursor(0, 0);
557  display.print(" ");
558
559  // initialize variables
560  goesForward = false;
561  distance = 75;
562  sensorValue = 0;
563
564  // initialize variables
565  durationf = 0;
566  distancef = 0;
567  durations = 0;
568  distances = 0;
569  durationd = 0;
570  distanced = 0;
571  durationr = 0;
572  distancer = 0;
573
574  // initialize variables
575  digitalWrite(LeftMotorForward, LOW);
576  digitalWrite(LeftMotorBackward, LOW);
577  digitalWrite(RightMotorForward, LOW);
578  digitalWrite(RightMotorBackward, LOW);
579
580  // initialize variables
581  digitalWrite(trig_pins, LOW);
582  digitalWrite(trig_pind, LOW);
583  digitalWrite(trig_pinf, LOW);
584
585  // initialize variables
586  pinMode(echo_pins, INPUT);
587  pinMode(echo_pind, INPUT);
588  pinMode(echo_pinf, INPUT);
589
590  // initialize variables
591  pinMode(buzzerPin, OUTPUT);
592  pinMode(ledPin, OUTPUT);
593
594  // initialize variables
595  display.begin(16, 2);
596  display.clear();
597  display.setCursor(0, 0);
598  display.print(" ");
599
600  // initialize variables
601  goesForward = false;
602  distance = 75;
603  sensorValue = 0;
604
605  // initialize variables
606  durationf = 0;
607  distancef = 0;
608  durations = 0;
609  distances = 0;
610  durationd = 0;
611  distanced = 0;
612  durationr = 0;
613  distancer = 0;
614
615  // initialize variables
616  digitalWrite(LeftMotorForward, LOW);
617  digitalWrite(LeftMotorBackward, LOW);
618  digitalWrite(RightMotorForward, LOW);
619  digitalWrite(RightMotorBackward, LOW);
620
621  // initialize variables
622  digitalWrite(trig_pins, LOW);
623  digitalWrite(trig_pind, LOW);
624  digitalWrite(trig_pinf, LOW);
625
626  // initialize variables
627  pinMode(echo_pins, INPUT);
628  pinMode(echo_pind, INPUT);
629  pinMode(echo_pinf, INPUT);
630
631  // initialize variables
632  pinMode(buzzerPin, OUTPUT);
633  pinMode(ledPin, OUTPUT);
634
635  // initialize variables
636  display.begin(16, 2);
637  display.clear();
638  display.setCursor(0, 0);
639  display.print(" ");
640
641  // initialize variables
642  goesForward = false;
643  distance = 75;
644  sensorValue = 0;
645
646  // initialize variables
647  durationf = 0;
648  distancef = 0;
649  durations = 0;
650  distances = 0;
651  durationd = 0;
652  distanced = 0;
653  durationr = 0;
654  distancer = 0;
655
656  // initialize variables
657  digitalWrite(LeftMotorForward, LOW);
658  digitalWrite(LeftMotorBackward, LOW);
659  digitalWrite(RightMotorForward, LOW);
660  digitalWrite(RightMotorBackward, LOW);
661
662  // initialize variables
663  digitalWrite(trig_pins, LOW);
664  digitalWrite(trig_pind, LOW);
665  digitalWrite(trig_pinf, LOW);
666
667  // initialize variables
668  pinMode(echo_pins, INPUT);
669  pinMode(echo_pind, INPUT);
670  pinMode(echo_pinf, INPUT);
671
672  // initialize variables
673  pinMode(buzzerPin, OUTPUT);
674  pinMode(ledPin, OUTPUT);
675
676  // initialize variables
677  display.begin(16, 2);
678  display.clear();
679  display.setCursor(0, 0);
680  display.print(" ");
681
682  // initialize variables
683  goesForward = false;
684  distance = 75;
685  sensorValue = 0;
686
687  // initialize variables
688  durationf = 0;
689  distancef = 0;
690  durations = 0;
691  distances = 0;
692  durationd = 0;
693  distanced = 0;
694  durationr = 0;
695  distancer = 0;
696
697  // initialize variables
698  digitalWrite(LeftMotorForward, LOW);
699  digitalWrite(LeftMotorBackward, LOW);
700  digitalWrite(RightMotorForward, LOW);
701  digitalWrite(RightMotorBackward, LOW);
702
703  // initialize variables
704  digitalWrite(trig_pins, LOW);
705  digitalWrite(trig_pind, LOW);
706  digitalWrite(trig_pinf, LOW);
707
708  // initialize variables
709  pinMode(echo_pins, INPUT);
710  pinMode(echo_pind, INPUT);
711  pinMode(echo_pinf, INPUT);
712
713  // initialize variables
714  pinMode(buzzerPin, OUTPUT);
715  pinMode(ledPin, OUTPUT);
716
717  // initialize variables
718  display.begin(16, 2);
719  display.clear();
720  display.setCursor(0, 0);
721  display.print(" ");
722
723  // initialize variables
724  goesForward = false;
725  distance = 75;
726  sensorValue = 0;
727
728  // initialize variables
729  durationf = 0;
730  distancef = 0;
731  durations = 0;
732  distances = 0;
733  durationd = 0;
734  distanced = 0;
735  durationr = 0;
736  distancer = 0;
737
738  // initialize variables
739  digitalWrite(LeftMotorForward, LOW);
740  digitalWrite(LeftMotorBackward, LOW);
741  digitalWrite(RightMotorForward, LOW);
742  digitalWrite(RightMotorBackward, LOW);
743
744  // initialize variables
745  digitalWrite(trig_pins, LOW);
746  digitalWrite(trig_pind, LOW);
747  digitalWrite(trig_pinf, LOW);
748
749  // initialize variables
750  pinMode(echo_pins, INPUT);
751  pinMode(echo_pind, INPUT);
752  pinMode(echo_pinf, INPUT);
753
754  // initialize variables
755  pinMode(buzzerPin, OUTPUT);
756  pinMode(ledPin, OUTPUT);
757
758  // initialize variables
759  display.begin(16, 2);
760  display.clear();
761  display.setCursor(0, 0);
762  display.print(" ");
763
764  // initialize variables
765  goesForward = false;
766  distance = 75;
767  sensorValue = 0;
768
769  // initialize variables
770  durationf = 0;
771  distancef = 0;
772  durations = 0;
773  distances = 0;
774  durationd = 0;
775  distanced = 0;
776  durationr = 0;
777  distancer = 0;
778
779  // initialize variables
780  digitalWrite(LeftMotorForward, LOW);
781  digitalWrite(LeftMotorBackward, LOW);
782  digitalWrite(RightMotorForward, LOW);
783  digitalWrite(RightMotorBackward, LOW);
784
785  // initialize variables
786  digitalWrite(trig_pins, LOW);
787  digitalWrite(trig_pind, LOW);
788  digitalWrite(trig_pinf, LOW);
789
790  // initialize variables
791  pinMode(echo_pins, INPUT);
792  pinMode(echo_pind, INPUT);
793  pinMode(echo_pinf, INPUT);
794
795  // initialize variables
796  pinMode(buzzerPin, OUTPUT);
797  pinMode(ledPin, OUTPUT);
798
799  // initialize variables
800  display.begin(16, 2);
801  display.clear();
802  display.setCursor(0, 0);
803  display.print(" ");
804
805  // initialize variables
806  goesForward = false;
807  distance = 75;
808  sensorValue = 0;
809
810  // initialize variables
811  durationf = 0;
812  distancef = 0;
813  durations = 0;
814  distances = 0;
815  durationd = 0;
816  distanced = 0;
817  durationr = 0;
818  distancer = 0;
819
820  // initialize variables
821  digitalWrite(LeftMotorForward, LOW);
822  digitalWrite(LeftMotorBackward, LOW);
823  digitalWrite(RightMotorForward, LOW);
824  digitalWrite(RightMotorBackward, LOW);
825
826  // initialize variables
827  digitalWrite(trig_pins, LOW);
828  digitalWrite(trig_pind, LOW);
829  digitalWrite(trig_pinf, LOW);
830
831  // initialize variables
832  pinMode(echo_pins, INPUT);
833  pinMode(echo_pind, INPUT);
834  pinMode(echo_pinf, INPUT);
835
836  // initialize variables
837  pinMode(buzzerPin, OUTPUT);
838  pinMode(ledPin, OUTPUT);
839
840  // initialize variables
841  display.begin(16, 2);
842  display.clear();
843  display.setCursor(0, 0);
844  display.print(" ");
845
846  // initialize variables
847  goesForward = false;
848  distance = 75;
849  sensorValue = 0;
850
851  // initialize variables
852  durationf = 0;
853  distancef = 0;
854  durations = 0;
855  distances = 0;
856  durationd = 0;
857  distanced = 0;
858  durationr = 0;
859  distancer = 0;
860
861  // initialize variables
862  digitalWrite(LeftMotorForward, LOW);
863  digitalWrite(LeftMotorBackward, LOW);
864  digitalWrite(RightMotorForward, LOW);
865  digitalWrite(RightMotorBackward, LOW);
866
867  // initialize variables
868  digitalWrite(trig_pins, LOW);
869  digitalWrite(trig_pind, LOW);
870  digitalWrite(trig_pinf, LOW);
871
872  // initialize variables
873  pinMode(echo_pins, INPUT);
874  pinMode(echo_pind, INPUT);
875  pinMode(echo_pinf, INPUT);
876
877  // initialize variables
878  pinMode(buzzerPin, OUTPUT);
879  pinMode(ledPin, OUTPUT);
880
881  // initialize variables
882  display.begin(16, 2);
883  display.clear();
884  display.setCursor(0, 0);
885  display.print(" ");
886
887  // initialize variables
888  goesForward = false;
889  distance = 75;
890  sensorValue = 0;
891
892  // initialize variables
893  durationf = 0;
894  distancef = 0;
895  durations = 0;
896  distances = 0;
897  durationd = 0;
898  distanced = 0;
899  durationr = 0;
900  distancer = 0;
901
902  // initialize variables
903  digitalWrite(LeftMotorForward, LOW);
904  digitalWrite(LeftMotorBackward, LOW);
905  digitalWrite(RightMotorForward, LOW);
906  digitalWrite(RightMotorBackward, LOW);
907
908  // initialize variables
909  digitalWrite(trig_pins, LOW);
910  digitalWrite(trig_pind, LOW);
911  digitalWrite(trig_pinf, LOW);
912
913  // initialize variables
914  pinMode(echo_pins, INPUT);
915  pinMode(echo_pind, INPUT);
916  pinMode(echo_pinf, INPUT);
917
918  // initialize variables
919  pinMode(buzzerPin, OUTPUT);
920  pinMode(ledPin, OUTPUT);
921
922  // initialize variables
923  display.begin(16, 2);
924  display.clear();
925  display.setCursor(0, 0);
926  display.print(" ");
927
928  // initialize variables
929  goesForward = false;
930  distance = 75;
931  sensorValue = 0;
932
933  // initialize variables
934  durationf = 0;
935  distancef = 0;
936  durations = 0;
937  distances = 0;
938  durationd = 0;
939  distanced = 0;
940  durationr = 0;
941  distancer = 0;
942
943  // initialize variables
944  digitalWrite(LeftMotorForward, LOW);
945  digitalWrite(LeftMotorBackward, LOW);
946  digitalWrite(RightMotorForward, LOW);
947  digitalWrite(RightMotorBackward, LOW);
948
949  // initialize variables
950  digitalWrite(trig_pins, LOW);
951  digitalWrite(trig_pind, LOW);
952  digitalWrite(trig_pinf, LOW);
953
954  // initialize variables
955  pinMode(echo_pins, INPUT);
956  pinMode(echo_pind, INPUT);
957  pinMode(echo_pinf, INPUT);
958
959  // initialize variables
960  pinMode(buzzerPin, OUTPUT);
961  pinMode(ledPin, OUTPUT);
962
963  // initialize variables
964  display.begin(16, 2);
965  display.clear();
966  display.setCursor(0, 0);
967  display.print(" ");
968
969  // initialize variables
970  goesForward = false;
971  distance = 75;
972  sensorValue = 0;
973
974  // initialize variables
975  durationf = 0;
976  distancef = 0;
977  durations = 0;
978  distances = 0;
979  durationd = 0;
980  distanced = 0;
981  durationr = 0;
982  distancer = 0;
983
984  // initialize variables
985  digitalWrite(LeftMotorForward, LOW);
986  digitalWrite(LeftMotorBackward, LOW);
987  digitalWrite(RightMotorForward, LOW);
988  digitalWrite(RightMotorBackward, LOW);
989
990  // initialize variables
991  digitalWrite(trig_pins, LOW);
992  digitalWrite(trig_pind, LOW);
993  digitalWrite(trig_pinf, LOW);
994
995  // initialize variables
996  pinMode(echo_pins, INPUT);
997  pinMode(echo_pind, INPUT);
998  pinMode(echo_pinf, INPUT);
999
1000 // initialize variables
1001 pinMode(buzzerPin, OUTPUT);
1002 pinMode(ledPin, OUTPUT);
1003
1004 // initialize variables
1005 display.begin(16, 2);
1006 display.clear();
1007 display.setCursor(0, 0);
1008 display.print(" ");
1009
1010 // initialize variables
1011 goesForward = false;
1012 distance = 75;
1013 sensorValue = 0;
1014
1015 // initialize variables
1016 durationf = 0;
1017 distancef = 0;
1018 durations = 0;
1019 distances = 0;
1020 durationd = 0;
1021 distanced = 0;
1022 durationr = 0;
1023 distancer = 0;
1024
1025 // initialize variables
1026 digitalWrite(LeftMotorForward, LOW);
1027 digitalWrite(LeftMotorBackward, LOW);
1028 digitalWrite(RightMotorForward, LOW);
1029 digitalWrite(RightMotorBackward, LOW);
1030
1031 // initialize variables
1032 digitalWrite(trig_pins, LOW);
1033 digitalWrite(trig_pind, LOW);
1034 digitalWrite(trig_pinf, LOW);
1035
1036 // initialize variables
1037 pinMode(echo_pins, INPUT);
1038 pinMode(echo_pind, INPUT);
1039 pinMode(echo_pinf, INPUT);
1040
1041 // initialize variables
1042 pinMode(buzzerPin, OUTPUT);
1043 pinMode(ledPin, OUTPUT);
1044
1045 // initialize variables
1046 display.begin(16, 2);
1047 display.clear();
1048 display.setCursor(0, 0);
1049 display.print(" ");
1050
1051 // initialize variables
1052 goesForward = false;
1053 distance = 75;
1054 sensorValue = 0;
1055
1056 // initialize variables
1057 durationf = 0;
1058 distancef = 0;
1059 durations = 0;
1060 distances = 0;
1061 durationd = 0;
1062 distanced = 0;
1063 durationr = 0;
1064 distancer = 0;
1065
1066 // initialize variables
1067 digitalWrite(LeftMotorForward, LOW);
1068 digitalWrite(LeftMotorBackward, LOW);
1069 digitalWrite(RightMotorForward, LOW);
1070 digitalWrite(RightMotorBackward, LOW);
1071
1072 // initialize variables
1073 digitalWrite(trig_pins, LOW);
1074 digitalWrite(trig_pind, LOW);
1075 digitalWrite(trig_pinf, LOW);
1076
1077 // initialize variables
1078 pinMode(echo_pins, INPUT);
1079 pinMode(echo_pind, INPUT);
1080 pinMode(echo_pinf, INPUT);
1081
1082 // initialize variables
1083 pinMode(buzzerPin, OUTPUT);
1084 pinMode(ledPin, OUTPUT);
1085
1086 // initialize variables
1087 display.begin(16, 2);
1088 display.clear();
1089 display.setCursor(0, 0);
1090 display.print(" ");
1091
1092 // initialize variables
1093 goesForward = false;
1094 distance = 75;
1095 sensorValue = 0;
1096
1097 // initialize variables
1098 durationf = 0;
1099 distancef = 0;
1100 durations = 0;
1101 distances = 0;
1102 durationd = 0;
1103 distanced = 0;
1104 durationr = 0;
1105 distancer = 0;
1106
1107 // initialize variables
1108 digitalWrite(LeftMotorForward, LOW);
1109 digitalWrite(LeftMotorBackward, LOW);
1110 digitalWrite(RightMotorForward, LOW);
1111 digitalWrite(RightMotorBackward, LOW);
1112
1113 // initialize variables
1114 digitalWrite(trig_pins, LOW);
1115 digitalWrite(trig_pind, LOW);
1116 digitalWrite(trig_pinf, LOW);
1117
1118 // initialize variables
1119 pinMode(echo_pins, INPUT);
1120 pinMode(echo_pind, INPUT);
1121 pinMode(echo_pinf, INPUT);
1122
1123 // initialize variables
1124 pinMode(buzzerPin, OUTPUT);
1125 pinMode(ledPin, OUTPUT);
1126
1127 // initialize variables
1128 display.begin(16, 2);
1129 display.clear();
1130 display.setCursor(0, 0);
1131 display.print(" ");
1132
1133 // initialize variables
1134 goesForward = false;
1135 distance = 75;
1136 sensorValue = 0;
1137
1138 // initialize variables
1139 durationf = 0;
1140 distancef = 0;
1141 durations = 0;
1142 distances = 0;
1143 durationd = 0;
1144 distanced = 0;
1145 durationr = 0;
1146 distancer = 0;
1147
1148 // initialize variables
1149 digitalWrite(LeftMotorForward, LOW);
1150 digitalWrite(LeftMotorBackward, LOW);
1151 digitalWrite(RightMotorForward, LOW);
1152 digitalWrite(RightMotorBackward, LOW);
1153
1154 // initialize variables
1155 digitalWrite(trig_pins, LOW);
1156 digitalWrite(trig_pind, LOW);
1157 digitalWrite(trig_pinf, LOW);
1158
1159 // initialize variables
1160 pinMode(echo_pins, INPUT);
1161 pinMode(echo_pind, INPUT);
1162 pinMode(echo_pinf, INPUT);
1163
1164 // initialize variables
1165 pinMode(buzzerPin, OUTPUT);
1166 pinMode(ledPin, OUTPUT);
1167
1168 // initialize variables
1169 display.begin(16, 2);
1170 display.clear();
1171 display.setCursor(0, 0);
1172 display.print(" ");
1173
1174 // initialize variables
1175 goesForward = false;
1176 distance = 75;
1177 sensorValue = 0;
1178
1179 // initialize variables
1180 durationf = 0;
1181 distancef = 0;
1182 durations = 0;
1183 distances = 0;
1184 durationd = 0;
1185 distanced = 0;
1186 durationr = 0;
1187 distancer = 0;
1188
1189 // initialize variables
1190 digitalWrite(LeftMotorForward, LOW);
1191 digitalWrite(LeftMotorBackward, LOW);
1192 digitalWrite(RightMotorForward, LOW);
1193 digitalWrite(RightMotorBackward, LOW);
1194
1195 // initialize variables
1196 digitalWrite(trig_pins, LOW);
1197 digitalWrite(trig_pind, LOW);
1198 digitalWrite(trig_pinf, LOW);
1199
1200 // initialize variables
1201 pinMode(echo_pins, INPUT);
1202 pinMode(echo_pind, INPUT);
1203 pinMode(echo_pinf, INPUT);
1204
1205 // initialize variables
1206 pinMode(buzzerPin, OUTPUT);
1207 pinMode(ledPin, OUTPUT);
1208
1209 // initialize variables
1210 display.begin(16, 2);
1211 display.clear();
1212 display.setCursor(0, 0);
1213 display.print(" ");
1214
1
```


5. Conclusions

The design and development of an autonomous robot for surface cleaning have been a complex and challenging research stage, where various issues and obstacles were encountered. However, the developed prototype has far exceeded our initial expectations in terms of performance and results achieved.

Nevertheless, it should be noted that the prototype is not without imperfections. A significant aspect is that due to the limited performance of the used sensors, the robot stops at a shorter distance than initially programmed. To address this issue, adjustments were made to the control code parameters. It is important to mention that ultrasonic sensors face difficulties in detecting objects that cannot adequately reflect sound waves. These limitations can be overcome by replacing ultrasonic sensors with infrared sensors, which offer superior capability in detecting and avoiding obstacles.

Thus, the identification and resolution of these issues represent a future research direction to enhance the functionality and performance of the autonomous robot for surface cleaning. By implementing new technological solutions and optimizing control algorithms, we can achieve higher precision and efficiency in the cleaning process, thereby improving the prototype's performance. We have ambitious plans for the future of the project, including a modern navigation system for autonomous and efficient movement, an adaptable cleaning system for industrial applications, improving energy consumption and increasing motor power, as well as implementing a safety system to prevent damage in case of an accident. Additionally, we aim to upscale the product, provide easy replacement of wearable parts and their separate acquisition, and integrate an AI system for automatic learning of new cleaning spaces.

6. Bibliography

- [1].<https://www.tinkercad.com/dashboard>
- [2].<https://docs.arduino.cc/>
- [3].<https://www.optimusdigital.ro>