

Development of Autonomous Robot Guide

Gustavo Moran Diaz – Electrical Engineering
Hans Sasuman – Industrial and Systems Engineering
Aaron Shamouil – Biomedical Engineering

Engr 122 S - Group 5

"I pledge my honor that I have abided by the Stevens Honor System"

I. **Executive Summary:**¹ The project requirements encompass time limitations, environmental factors, manufacturing limitations, and part specifications. In terms of time limitations, for the final project, a total limit of 4 cumulative hours is allowed for the printing of all parts. For the environmental factors, the design requirement calls for an acknowledgment of the arena's size of 96" x 36" and sidewall height of 3.5". Manufacturing limitations regard the tools and parts available at hand to use in the final design. In essence, the design should not necessitate the use of any parts or tools not already available to the designer. The last design requirement, part specifications, regards several miscellaneous design part parameters. Firstly, the design is required to use and store at least 3 sensors, a WeMos board, an OLED, a power switch, a motor controller, and a GPS float. Secondly, the design must be able to integrate a method of wiring besides the use of a breadboard. Lastly, the wiring of the part itself must be color coordinated.

Our final design consists of 4 separately printed parts. The total print time for the parts came out to approximately 2 hours and around 10 minutes, satisfying the time limitations. The four parts include a bottom level, a top level, a GPS float attachment, and a light switch attachment. The bottom level will hold the WeMos board, a single front-facing sensor, and the soldered on shield. The top-level will hold two more sensors in the left and right direction, the motor controller, the OLED, and a mount for the GPS float, satisfying the part specifications. All printed parts, as well as all the part specifications (OLED, sensors, etc.), were all fastened to the chassis and to each other through the use of the screws and spacers, fulfilling the manufacturing limitations. Lastly, environmental limitations are more

¹ Written by Hans Sasuman

significantly addressed in the software area of this project, however, the overall design aims for a more significant vertical component rather than a horizontal component for the purposes of easier movement throughout the arena.

- II. **Introduction and Project Objectives:**² Very often, college tours here on Stevens, while an exciting and intriguing experience for some, can be tiresome and long for others. In an attempt to rectify such an experience, the possible solution of an automated and self-guided tour, done from an autonomous vehicle in which visitors may sit and relax as it tours them through campus has been devised. Such a solution is not a simple solution, however, as it requires a very adept and dynamic program and design that can safely and efficiently guide the robot through the campus all while avoiding a number of possible obstructions. Part of this solution includes small-scale testing of such a machine. It is the goal of this project to do exactly that as it is an automated robot that may maneuver through an arena that will be designed, built, and programmed by the students.

Here is where our project comes into the frame. In order to model a functional self-guided tour, our project and its design have been developed to address several key objectives a real-world or small-scale version of the automated tour would need to acknowledge. Firstly, the design must not at any time come into contact with any structures or obstructions it comes across. In other words, our project must be able to sense and adjust accordingly to any and all blocks in its way. Secondly, the design must have some way to understand its position relative to its destination no matter where on the arena it or its destination is. Thirdly, the design must be able to work efficiently and timely through a controlled and safe variation of its velocity according to whatever situation it may be in. Lastly, the design must be able to understand its orientation in reference to its next target location.

² Written by Hans Sasuman

III. **Design Requirements:**³ Considering the objectives that we wish to have our robot exhibit; we also must remember and consider the design constraints or requirements our robot is limited to. In thinking about such design constraints, it's important to realize and understand that such constraints can be thought of as a limit or restriction on the features or behaviors of the design. In reality, such limitations are important as resource management, design practicality, and design functionality can all somehow put limitations on real-world design solutions. While the design constraints on our small-scale design are not as dramatic as they could be designing a full-size transportation vehicle, the limitations will still majorly impact the design process and elegance of our robot.

To begin, there were six main categories into which the various design restraints can be organized. These categories are as follows: cost (for design & manufacturing), time, available parts, part dimension & specification, available manufacturing tools, and environmental factors.

i. Cost

- For this small-scale design, there is no monetary budget allotted to us. With this said, our design is limited to certain cost constraints, particularly when it comes to how many replacement elements, we are allotted in the case that we break the originals. Similarly, while overlapping with the time requirement, there is ultimately a limit to how much 3D printing material we can use, but rather than measured by length, it is measured by time.

ii. Time

- As mentioned previously, there is a time limit on how much time it takes to print our main design and all its parts. This time limit is 2 hours and 30 minutes. Considering this,

³ Written by Hans Sasuman

it is vital that the total print time for all 3D printed parts of the design is under or equal to 2 hours and 30 minutes.

iii. Available Parts

- In considering a design, it is important to consider the parts and bits available and not available for use. Presently, different sizes of screws, bolts, nuts, and spacers are available for use. Whether it comes to fastening the design to the chassis, or the required elements to the design, the specifications of all available parts should be considered and utilized to the best extent.

iv. Part Dimension and Specification

- Part Dimension and Specification is one of the constant variables in this project as the spacings and dimension of all the required elements such as the WeMos board, chassis, and control board does not change. Considering this, when creating a design, the specifications and dimensions of all the required parts must be considered in order to best create a functional and elegant final robot.

v. Available Manufacturing Tools

- The practicality of the build of the design and the final robot is an important factor to consider. Whether or not the design requires certain tools to adjust or edit the design, and what tools will be required to create a design to begin with is significant. Presently, a 3D printer is available for printing designs, soldering stations are available for sauntering, sandpaper is available for sanding, and different types of screwdrivers are available for general use in fastening and such actions.

vi. Environmental Factors

- Environmental factors encompass the physical testing of the project and whether the design accounts for such things or not. Considering design width because of the

distances between model buildings is one example of an environmental factor influencing the design. The height of the sensor holders on the design is also important to consider as if the height is above the height of the model buildings, trouble sensing the building may arise.

IV. Concept Design:⁴

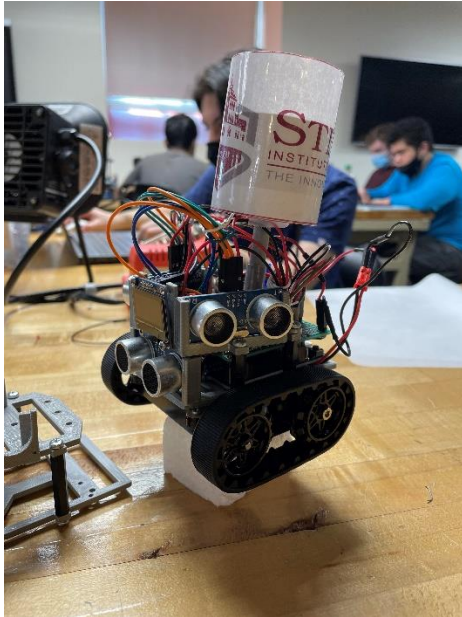


Figure 1 is our final design

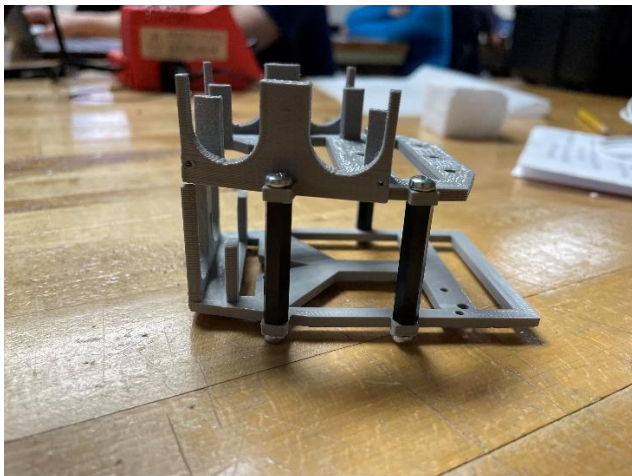


Figure 2 was our original print

⁴ Written by Hans Sasuman

There were other unprinted designs we had considered. Such designs differed in height and width, particularly regarding the distinction between 1 and 2 levels on the robot. There were also major differences in element capacity in each of the designs, or in other words, what each design could/was meant to hold.

V. Design Concept Evaluation:⁵ The first major decision that swayed our ultimate design template was considering either a one level or two-level design. The benefit of the one level design is the conciseness and speediness of the design and print time. Not only would it benefit from a shorter print time, but because it'd be one level and one piece, there is no need for any complicated fastening or element integration. The benefit of the two-level design was a larger capacity for required elements, a smaller and more concise width. The concise width is significant because there would be no overhang on the chassis, maximizing stability, and addressing certain environmental constraints in the obstacle course. Ultimately, we decided upon a two-level design as the concise width was an integral feature that we wanted to integrate to our design. Points regarding the design elegance, wiring organization, storage capacity, and robot stability also influenced our decision heavily. Also worth mentioning is the overall similarly short print time our two-level design had in comparison to a level design. In terms of fastening and integration of the two-level design, all the parts available to us were more than enough to securely and elegantly fasten the design to the chassis. In all, the two-level design not only had many of its own unique benefits, but its disadvantages compared to other potential designs proved not to be significant, influencing our decision to go with the two-level design.

VI. Final Design Description:⁶

⁵ Written by Hans Sasuman

⁶ Written by Gustavo Moran Diaz

Item Number	Part	Quantity
1	Zumo Chassis	1
2	Bottom Half of Bracket	1
3	B18.6.7M – M3 x 0.5 x 6 Type I Cross Recessed PHMS –6N	9
4	B18.6.7M – M2 x 0.4 x 20 Type I Cross Recessed PHMS –10N	4
5	B18.6.7M – M2 x 0.4 x 13 Type I Cross PHMS --13N	4
6	B18.6.7M 0 M2 x 0.4 x 6 Type I Cross Recessed PHMS –6N	2
7	B18.2.4.1M – Hex nut, Style 1 M2 x).4 –D-N	8
8	WeMos Board	1
9	Top Half of Bracket	1
10	Bracket Standoffs	4
11	Motor Controller	1
12	OLED Display	1
13	Ultrasonic Sensor	3
14	Pin Board	1
15	Bracket for Switch	1

i. System Architecture:

Overall, the system was made with two major design goals in mind; these design goals were stability and efficiency.

Efficiency: The printed parts, required elements, and wiring design were all meticulously made and put together to maximize the efficiency of the robot. To begin, one goal in efficiency we hoped to achieve with our design is a robot that is easy and intuitive to disassemble and reassemble to better ease any edits we might have needed to make throughout the course of designing it. With the two-level design, attached through screws, we feel we've achieved this goal as we have easy and open access to the whole of the robot through simply unscrewing the screws.

Stability: Considering the objective of the project and remembering the environmental constraints our robot must be able to navigate, a system architecture that prioritized stability over the

chassis was paramount. With the two-level design, all the printed parts stay directly on top of the chassis, minimizing the total space taken up by the robot in part to address the environmental constraints and to maximize the stability of the robot.

ii. Mechanical Design:

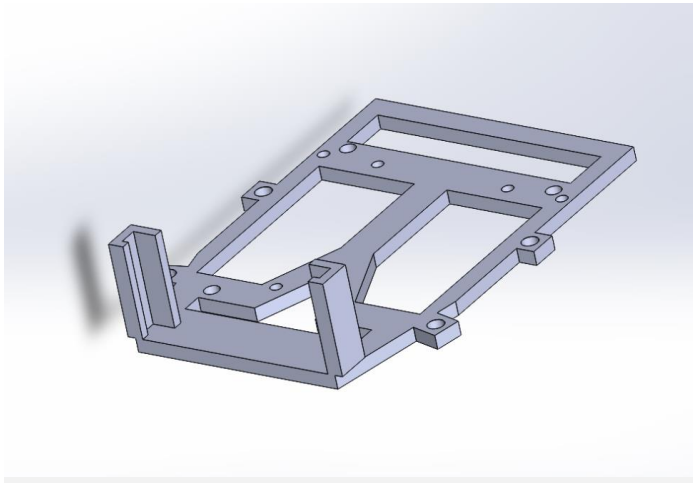


Figure 1.

Part Designation: Bottom Half of the Bracket

Part Designer: Gustavo Moran Diaz

Print Time: 42 Minutes

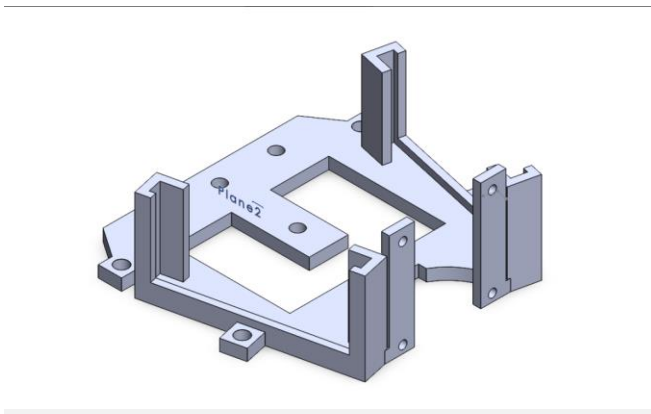


Figure 2.

Part Designation: Top half of the bracket

Part Designer: Gustavo Moran Diaz

Print Time: 50 Minuets

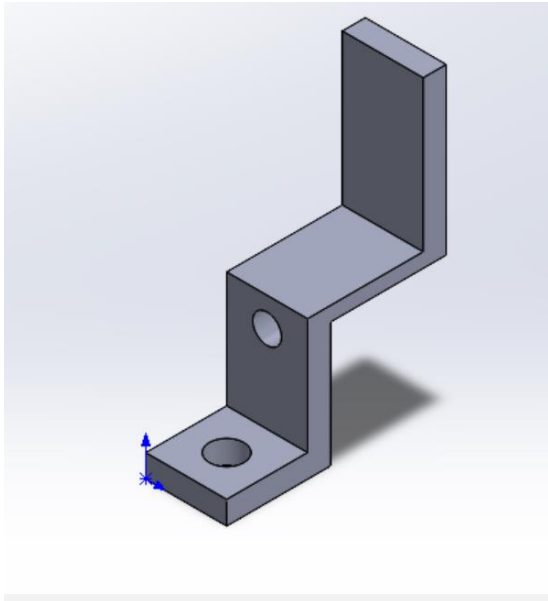


Figure 3.

Part Designation: Bracket for the power switch

Part Designer: Gustavo Moran Diaz

Print Time: 3 Minutes

iii. Electrical and Wiring Design:

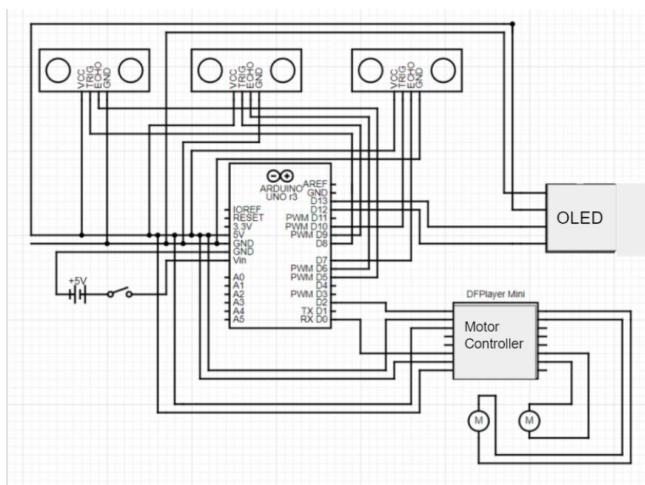
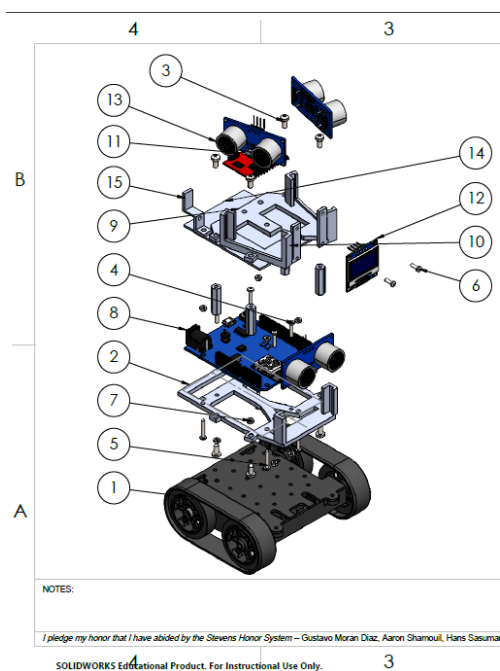


Figure 4.

This wiring design is optimized to the fullest extent for an elegant and efficient robot. Many groups of wires were tied together with a tool available to all teams, zip ties. The wires are also color coordinated depending on what the wire was connected to, as well as the particular use of the wire itself (i.g. red wires for 5v and black wires for grounds).⁷

VII. Assembly Instructions:⁸



ITEM NO.	PART NUMBER	DESCRIPTION	QTY.
1		Zumo Chassis	1
2		Bottom Half of Bracket	1
3	B18.6.7M - M3 x 0.5 x 6 Type I Cross Recessed PHMS --6N		9
4	B18.6.7M - M2 x 0.4 x 10 Type I Cross Recessed PHMS --10N		4
5	B18.6.7M - M2 x 0.4 x 13 Type I Cross Recessed PHMS --13N		4
6	B18.6.7M - M2 x 0.4 x 6 Type I Cross Recessed PHMS --6N		2
7	B18.2.4.1M - Hex nut, Style 1, M2 x 0.4 --D-N		8
8		WeMos Board	1
9		Top Half of Bracket	1
10		bracket standoffs	4
11		Motor Controller	1
12		OLED Display	1
13		Ultrasonic Sensor	3
14		Pin Board	1
15		Bracket for Switch	1

NOTES:

I pledge my honor that I have abided by the Stevens Honor System -- Gustavo Moran Diaz, Aaron Shamouil, Hans Sasuman

Last updated: Tuesday, April 18, 2022 5:14:40 PM

UNITS: MM SCALE: 1:4 REV: SHEET 1 OF 1

SOLIDWORKS Educational Product. For Instructional Use Only.

NAME: Gustavo Moran Diaz, Aaron Shamouil, Hans Sasuman
 CLASS: SIT ENGR122 S
 PROJECT: FinalAssembly
 MATERIAL: MASS: SIZE DWG. #
 B 1

Step 1) Open battery compartment of Zumo chassis and place nuts in the cavities that align with the smaller holes on the bottom half of the main bracket.

Step 2) Place M3 screws through the “larger” holes on the bottom of the bottom half of the main bracket and place larger M3 screws through the 4 outermost (and largest) holes, carefully, while holding those screws in place, place the bottom half of the main bracket on the Zumo chassis, make sure that the 4 outermost screws have the threads facing up and the inner holes have the screwheads facing up.

⁷ Figure 4 designed by Gustavo Moran Diaz; section written by Hans Sasuman

⁸ Written by Gustavo Moran Diaz

Step 3) While the bottom half of the main bracket is still not fully secured, slip the motor and battery cables through the respective holes on the bracket to make proceeding steps easier

Step 4) Align the smaller holes on the main bracket with the holes that you have placed the nuts into in step 1 and place M2 screws through and tighten until secure

Step 5) Next place the WeMos board onto the bracket making sure that the screws placed during step 2 are going through the mounting holes on the WeMos and place the shield pcb onto the WeMos board at this time as well.

Step 6) Secure the Wemos using M2 nuts on each of the bolts while making sure that the battery and motor cables are still easily accessible and not restrained by over tightening of the nuts

Step 7) Place the front ultrasonic sensor into its bracket since this will be more difficult to do later.

Step 8) Screw the spacers onto the 4 M3 bolts that were placed below the main bracket in step 2, then place 4 more M3 bolts through the respective holes on the top half of the bracket and screw them into the spacers.

Step 9) Place the motor controller onto the top half of the bracket and secure it using m3 bolts and nuts

Step 10) Route the two motor cables through the top half of the bracket and plug them into the motor controller

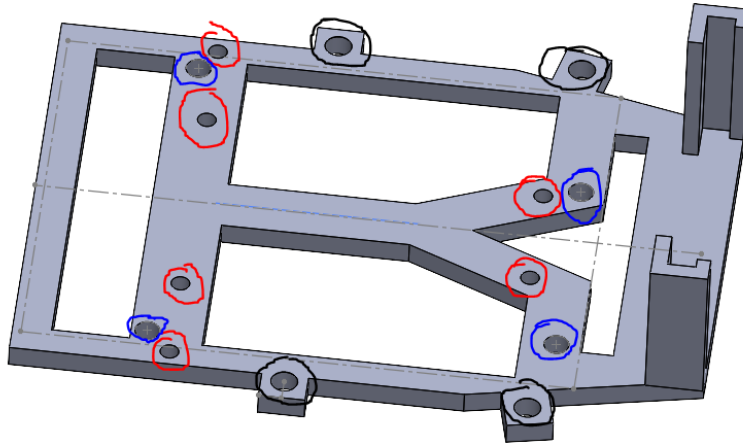
Step 11) Slot the two remaining ultra-sonic sensors into their slots in the top half bracket

Step 12) Secure the OLED module onto the front of the top half bracket using m2 bolts, if friction is not enough to hold the bolts into place use nuts on the backs of each to securely mount.

Step 13) Secure the switch bracket onto the shield PCB using an M3 bolt and nut

Mechanical assembly complete

VIII. Prototyping Specifications:⁹



(Clarification for step 2 and 4) Bottom Half of bracket depicted here. M3 screws with the threads facing upwards should be put in the holes circled in black, M3 screws facing downwards should be inserted into the holes circled in blue and M3 screws should be inserted into the small holes circled in red with the threads facing downward.

⁹ Written by Gustavo Moran Diaz

IX. Software:¹⁰

Program Logic:

Using Arduino, a coding platform that incorporates the C/C++ language, our group was able to program the robot to navigate through four targets that represented different parts of Stevens Institute of Technology's campus. Additionally, the program utilized the coordinates that were recorded by the LiDAR system to determine the path required for the robot to move towards the current target. During previous weeks our group completed the Left-Hand Loop challenge, which helped with calibrating the motors of the robot and navigating throughout the obstacle course. The design of our program for the final challenge incorporated four main logic sections including pathfinding, turning, obstacle avoidance, and target finding. Lastly, a baseline program was given to work with and edit, which included the necessary libraries and functions for the robot to operate properly.

Pathfinding:

To determine the path required for the robot to navigate towards a target, Mr. Shamouil incorporated vector calculations to track the direction and expected path of the robot. For example, there were three separate coordinates that were updated throughout the course of our program including: $(previous_x, previous_y)$, $(current_x, current_y)$, $(target_x, target_y)$. The three coordinates were used to determine two vectors that would calculate first the magnitude and direction of the previous and current coordinates and second the magnitude and direction of the current and target coordinates. Creating the two vectors allowed the program to use the dot product, which is $\cos(\theta) = \frac{(a \cdot b)}{|a| \cdot |b|}$. In the program, values of a and b in the dot product equation were substituted with the corresponding components of the two vectors. Additionally, the `<math.h>` library was applied in the

¹⁰ Written by Aaron Shamouil

program to perform math calculations needed for determining the components of both vectors and the turn angle. The main purpose of the dot product was to determine the turn angle of the robot, when positioning the direction of the robot towards the target. Another formula that our program utilized was the vector product, which is $(current_x - previous_x) * (target_y - current_y) - (current_y - previous_y) * (target_x - current_x)$. The values in the formula were replaced with corresponding outputs from the LiDAR system to determine the direction of the turn angle that would allow the robot to reach the current target. Finally, the Pythagorean theorem

$(\text{sqrt} \sqrt{(target_y - current_y)^2 + ((target_x - current_x)^2)})$ was used in the program to

measure the distance from the current coordinates to the target coordinates. The Pythagorean theorem allowed the robot to stop at a target if the distance was less than 100 mm, which would update the new coordinate values for the next target location.

Turning:

There were two different counter variables utilized in the program to identify when the program would turn towards a target. One of the counter variables (post_obstacle_avoidance) would run a function in the for loop to determine a new turn angle once the robot exited obstacle avoidance. The second counter variable would run the function of turning the robot towards the direction of the new target once the current target was reached by the robot. Additionally, in both functions there were two if statements that would determine the turn direction of the robot based on the vector product and the angle calculated from the dot product. For the vector product, if the value was positive then the robot would turn left and if the vector product was negative, then the robot would turn right. The angle calculated from the dot product was integrated into these if statements by multiplying the angle by an experimentally determined factor to determine the delay time for the robot to spin. Based on individual experiments with the motor speeds of the robot, the delay time for turning the robot was determined

to be an experimental factor of 3.7, which was multiplied by the angle value converted to degrees. After the turn was complete, the previous x and previous y values of the robot were changed to the current x and current y values of the robot to reflect the robot moving forward in that direction, until the target was reached, or another obstacle was detected.

Obstacle Avoidance:

The robot avoided obstacles along its path by reading the distance values between the robot and the obstacle in millimeters. In our program, a counter variable called `obstacle_counter` would track when the robot was in obstacle avoidance to allow the obstacle avoidance if statements to run. The obstacle avoidance statements were conditioned based on the sensor readings of the front, left, and right sensor. Additionally, the motor speeds were tested in this section of our code to meet the error correction adjustments needed to avoid obstacles and adjust the robot's path. After the robot exited obstacle avoidance, there was a check incorporated that would allow the robot to navigate a new angle or move forward depending on which if statements were run. Furthermore, counter variables such as `obstacle_counter` and `post_obstacle` avoidance helped us differentiate between circumstances our robot was causing issues in our code, since we could keep track of special cases regarding the robot's path. Lastly, to account for the blind spot in the design, the program had an additional delay in the left and right turn functions to account for the area that was causing issues with obstacle avoidance.

Target Finding:

The target finding section of the program focused on determining how close the robot was to the was to a target and at what angle the robot would need to turn to navigate towards the next target. Additionally, the Pythagorean Theorem was utilized to determine how far the robot was from the reaching the target. Each target had a bubble radius of one hundred millimeters. Therefore, to determine where the robot would stop, the if statement was conditioned to recognize coordinates

located in the bubble created around each target. Furthermore, the target_counter counter variable allowed for the robot to update the target coordinates and the robot to navigate towards the next target. Once the robot reached a certain target the robot would stop for two seconds at that target and then find the new angle needed to redirect the robot's path towards the next target. Once the target_counter variable had a value of 4 then, the robot would know that the robot reached all targets and would therefore stop at that position.

Program Segments:

Pathfinding:

```

173 float angle_equation(int x1,int xprev,int y1, int yprev,int xtarget,int ytarget)
174 {
175     int x_previousv = x1-xprev; // Vector 1 x-component
176     int y_previousv = y1-yprev; // Vector 1 y-component
177     int x_targv = xtarget-x1; // Vector 2 x-component
178     int y_targv = ytarget-y1; // Vector 2 y-component
179
180     float dot= ((x_previousv*x_targv)+(y_previousv*y_targv)); // Dot Product of Vector 1 and Vector 2
181
182     float magnitude_1 = sqrt((pow(x_previousv,2)+pow(y_previousv,2))); // Magnitude of Vector 1
183     float magnitude_2 = sqrt((pow(x_targv,2)+pow(y_targv,2))); // Magnitude of Vector 2
184     float magnitude_result = (magnitude_1*magnitude_2); // Magnitude of Vector 1* Vector 2
185
186     float result = dot/magnitude_result;
187     float angle = acos(result); // Turn Angle(Radians)
188     float angle1 = (angle*180)/(M_PI); // Turn Angle(Degrees)
189     return angle1;
190 }
191
192 float vector_product(int x1,int xprev,int y1, int yprev,int xtarget,int ytarget)
193 {
194     float vector_prod = (x1-xprev)*(ytarget-y1)-(y1-yprev)*(xtarget-x1); // Vector Product to determine left(vecprod > 0) and right(vecprod < 0)
195     return vector_prod;
196 }
197
198 float distance_target_func(int x1, int y1, int xtarget, int ytarget) // Pythagorean Theorem (Distance to Target)
199 {
200     float distance_to_target = sqrt(pow((ytarget-y1),2)+pow(xtarget-x1,2));
201     return distance_to_target;
202 }

```

Turn Angle:

```

373 if(spin_target == 0 && obstacle_counter == 1 )
374 {
375     if( vector_prod > 0 ) //Turn left
376     {
377         spin_left((angle_output*(3.7)));
378         obstacle_counter = 1;
379         post_obstacle_avoidance = 1;
380         spin_target = -1;
381
382         previous_x = x;
383         previous_y = y;
384     }
385 }
386
387 if ( vector_prod < 0 ) //Turn right
388 {
389     spin_right((angle_output*(3.7)));
390     obstacle_counter = 1;
391     post_obstacle_avoidance = 1;
392     spin_target = -1;
393
394     previous_x = x;
395     previous_y = y;
396 }
397 }
398 }

```

```

400 if(post_obstacle_avoidance == 0)
401 {
402     if(vector_prod > 0 ) //Turn left
403     {
404         spin_left((angle_output*(3.7)));
405         obstacle_counter = 1;
406         post_obstacle_avoidance = 1;
407
408         previous_x = x;
409         previous_y = y;
410     }
411     if (vector_prod < 0 ) //Turn right
412     {
413         spin_right((angle_output*(3.7)));
414         obstacle_counter = 1;
415         post_obstacle_avoidance = 1;
416
417         previous_x = x;
418         previous_y = y;
419     }
420 }

```

Obstacle Avoidance:

```

551 // Obstacle Avoidance When Distance Front is Less than 50 mm //
552
553 if(distance_front < 50 && distance_left > 50 && distance_right > 50)
554 {
555     backward();
556     delay(200);
557
558     previous_x = x;
559     previous_y = y;
560
561     obstacle_counter = 0;
562 }
563 if(distance_front < 50 && distance_left < 50 && distance_right > 50)
564 {
565     motor1.write(61); // Back and to the right
566     motor2.write(75);
567     delay(200);
568
569     previous_x = x;
570     previous_y = y;
571
572     obstacle_counter = 0;
573 }
574
575 if(distance_front < 50 && distance_left > 50 && distance_right < 50)
576 {
577     motor1.write(75); // Back and to the left
578     motor2.write(61);
579     delay(200);
580
581     previous_x = x;
582     previous_y = y;
583
584     obstacle_counter = 0;
585 }
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

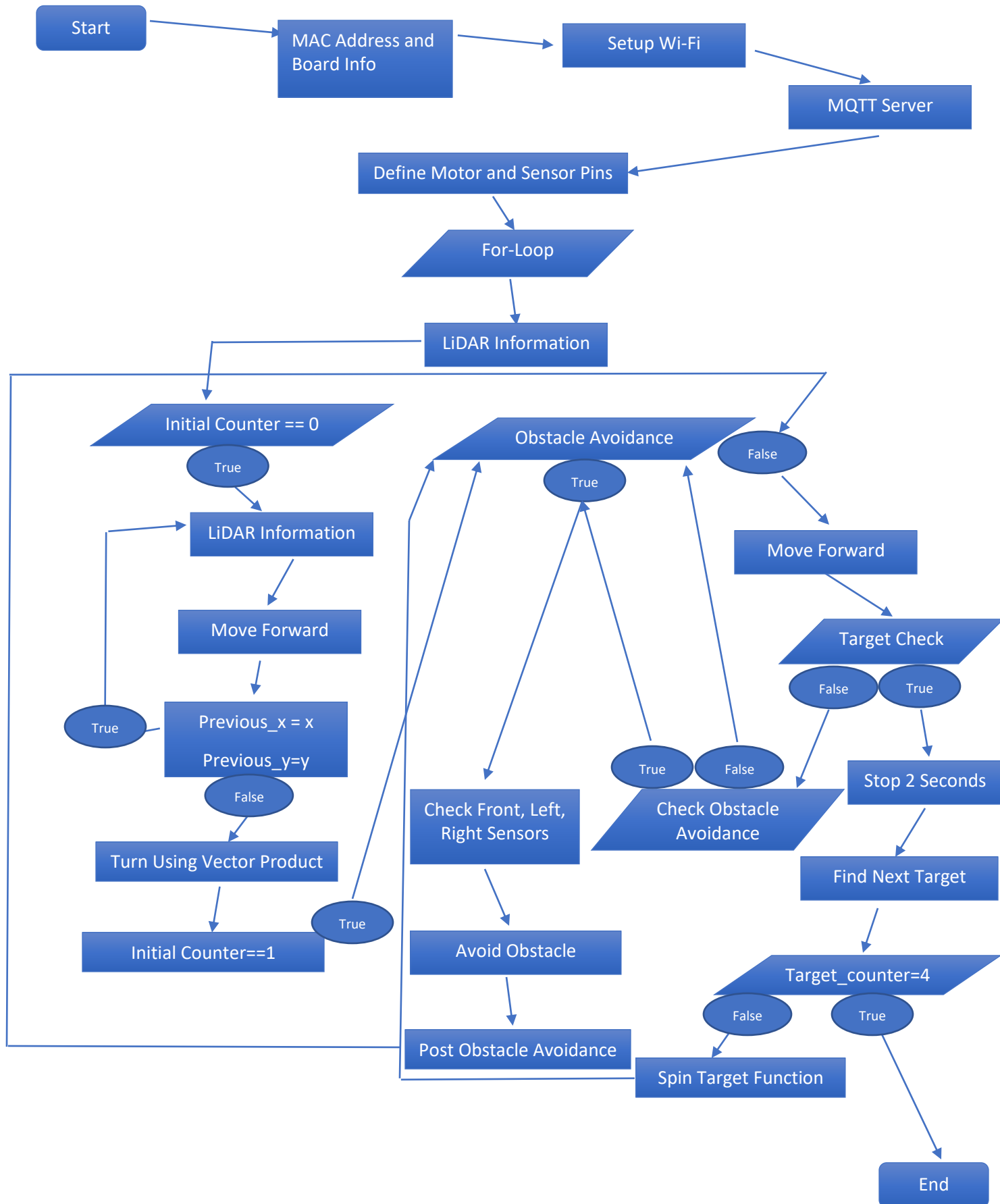
```

Target Finding:

```
if (distance_to_target <= 100) // Distance between robot and target
{
    motor1.write(90);
    motor2.write(90);
    delay(2000);
    target_counter = target_counter + 1;
    previous_x = x;
    previous_y = y;
    spin_target = 0;
    post_obstacle_avoidance = 1;
}

if (target_counter = 4) // Stop Robot when final target reached
{
    motor1.write(90);
    motor2.write(90);
}
}
```

Software Flow Chart



X. Performance Report:¹¹

The robot was able to achieve both the first and second target when navigating throughout the obstacle course. The positives of the run were that the robot was able to reach the second target without any crashes, while achieving a time of 1.05 minutes. The issues that were faced without performances were the inconsistencies of the LiDAR system, which had a delay time of when the coordinates of the current robot location were updated.

Additionally, when navigating from target two three, Mr. Shamouil noticed that the robot would get stuck on an obstacle between the two targets. During the later stages of testing, the motors of the robot had a mechanical issue in which the motors were replaced and re-calibrated by Mr. Moran-Diaz, which took time away from testing the initial program.

Therefore, the plan for future runs is to add intermediate points between the targets to avoid the obstacle that was causing issues in our test runs. Additionally, to account for the functionality of the design, there were no issues noted regarding the sensor readings, wiring, or the motor speeds. Furthermore, as a group more testing will be performed to account for other changes in the program as necessary to navigate to at least target 3, but with a goal of achieving 4 targets.

¹¹ Written by Aaron Shamouil

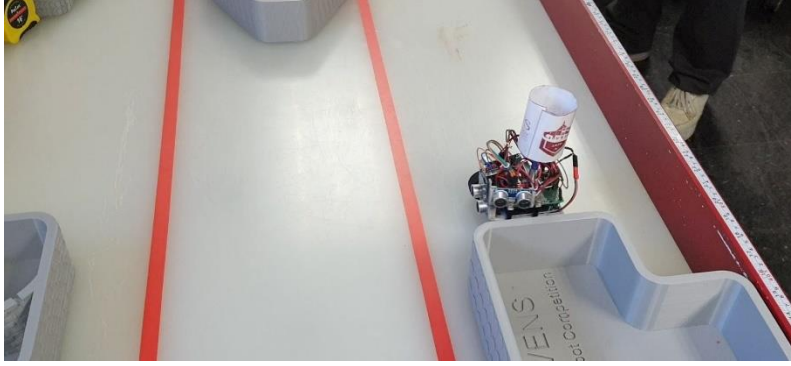


Figure V: Picture of robot reaching Target 2

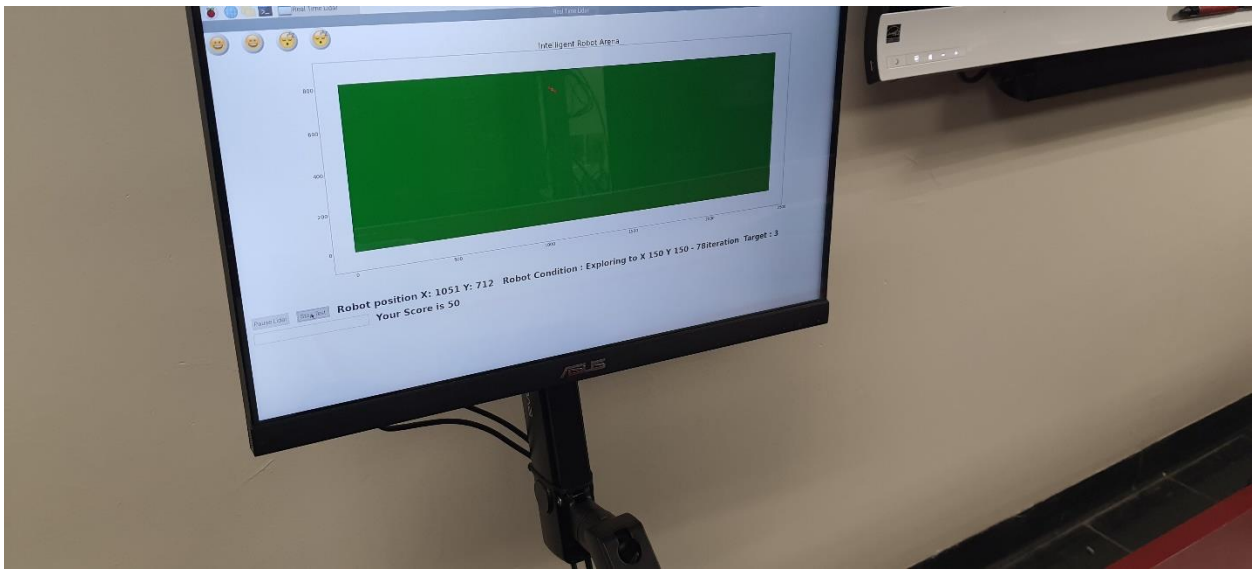


Figure VI: LiDAR Reading of Target reached

XI. Concluding Remarks:¹²

In closing, there are several key areas where we feel that as a team, we could have spent more time and effort, particularly early on, in addressing. Firstly, something we wished we had acknowledged sooner to have more time to address regarded the LiDAR system and its extreme inconsistency. Secondly, better understanding of “previous_x and previous_y” would have been more conducive to a more expedited and efficient coding experience.

Thirdly, counter variable and error checking was another place where our team had much trouble, pushing our progress further and further back. Lastly, speaking to a problem that involves both the physical and software components of our projects, dealing with the blind spots and accounting for them in the code was a place where we also ran into much difficulty. Overall, the places in which we had a lot of trouble, and we wish we had paid more attention to earlier are very indicative overall of where the majority of the group's efforts were focused, especially towards the end of the semester. In concluding, however, through efficient communication, and real dedication to completing this project, our team was able to beat all of the previous battles.