Bauer Bot

Tasawar Siddiquy
Electronic Engineering
Hochschule Hamm-Lippstadt
Lippstadt, Germany
tasawar.siddiquy@hshl.stud.de

Md Saidur Rahman
Electronic Engineering
Hochschule Hamm-Lippstadt
Lippstadt, Germany
md-saidur.rahman@stud.hshl.de

Arfat Kamal
Electronic Engineering
Hochschule Hamm-Lippstadt
Lippstadt, Germany
arfat.kamal@stud.hshl.de

Sahat Al Ferdous Fahim
Electronic Engineering
Hochschule Hamm-Lippstadt
Lippstadt, Germany
Sahat-al-ferdous.fahim@stud.hshl.de

***Abstract*—An autonomous line-following vehicle has been developed using Arduino Uno, motors and sensors. with the concept of precision farming. The vehicle have been designed using Solid works and the built with plywood and 3D printed PLA parts. The vehicle is able to drive and navigate on a test grid constructed with colorful lines.**

***Index Terms*—Design, System Engineering, Arduino, Navigation, Autonomous.**

I. Introduction

We were asked to design and develop a prototype of an autonomous precision farming robot using specified hardware components. The motivation behind the prototype is the development of such a robot which can autonomously detect and collect objects from farms and help farmers increase production. A test environment was introduced where the robot has to autonomously drive from one coordinate to another or multiple coordinates by detecting it's position on the grid (Figure 1).

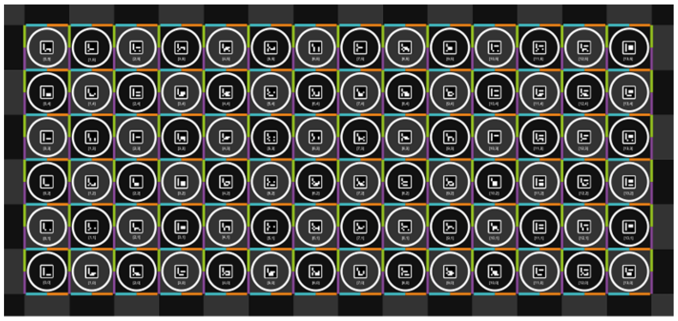


Fig. 1. Grid Test Environment

II. Requirements

The system has both functional and non functional requirement.

A. Functional Requirements

The main functional requirement is to follow the lines and move to different coordinates.in the test environment shown in figure The grid contains two-dimensional coordinates, we will use those coordinates to determine our vehicle's location and target location. The specific tasks the system will have to fulfill are :

- Drive on the line (vertically and horizontally)
- Differentiate between all colors on the lines
- count the lines where the vehicle is (both vertically and horizontally)
- Drive on the line, turn 90 degrees and keep driving on the new line
- Drive on the line for two squares, turn 90 degrees and keep driving on the new line for two squares
- Drive from one point to another (keep track of where the vehicle is on the map). starting point and the endpoint will be specified.
- Drive from point to point

B. Non Functional Requirements

The non-functional requirements are mainly the hardware component constrains that we have to fulfil while building the system. We have to build the system using only the components listed below

- Arduino Uno
- Battery 11.1 V
- 2x RB-35 gearbox Motors (1:30)
- SBC-MotoDriver2 (L298N)
- 3x IR Sensors (ST1140)
- Ultrasonic Sensor(HC SR-04)
- RGB Color Sensor (TCS3200)
- Breadboard
- Jumper wires
- 2x Wheels
- 1x Ball Caster Wheel

We will describe some detailed information of the main modules.

1) Arduino Uno: Arduino Uno (Figure 2) is a microcontroller board based on the ATmega328P (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator (CSTCE16M0V53-R0), a USB connection, a power jack, an ICSP header, and a reset button. We used this module as our main component. All our components are connected and communicate through Arduino.

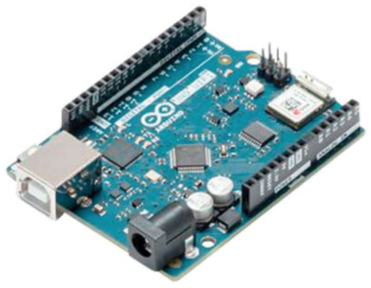


Fig. 2. Arduino Uno

https://www.reichelt.de/arduino-uno-wifi-rev2-atmega-4809-arduino-uno-rev2
-p248661.html?search=Arduino+UNO+WiFi+Rev2

2) Battery 11.1 V 3200 mAh: The Conrad Energy Eco-Line LiPo batteries (Figure 3) are very reliable and stable and are suitable for use in your micro-Helis, motor sailors, artificial flight models, or model cars. The new Conrad Energy battery pack generation is based on the latest lipo-technology which leads to extraordinary performance. The greater energy density could be reduced to minimum dimensions which also on your low weight noticeable. The battery is the main power source of our vehicle. It supplies power through the motor driver to our system.

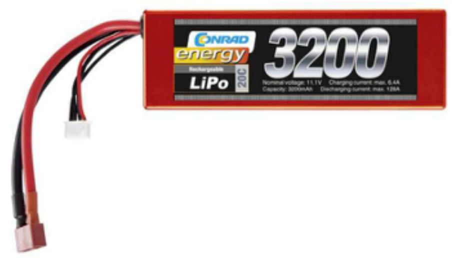


Fig. 3. Battery 11.1 V 3200 mAh

https://www.conrad.de/de/p/conrad-energy-modellbau-akkupack-lipo-11-1-v-3200-mah-zellen-zahl-3-20-c-box-hardcase-t-buchse-1414162.htmlproductTechData

3) Motor (1:30) : RB-35 gearbox integrated motors(Figure 4) are Excellently machined, high resistance, quiet, and small yet still offer high efficiency. The input and output shafts are offset slightly. The two gearbox integrated motors are responsible for the movement of our vehicle. The speed is also can be controlled and configured by giving commands from the code.

Fig. 4. Motor (1:30)

https://www.conrad.de/de/p/modelcraft-rb350030-0a101r-getriebemotor-12-v-1-30-227544.html

4) SBC-MotoDriver 2 (L298N): The MotoDriver 2 (Figure 5) is an expansion board that can be connected fast and easily to your single-board computer. It enables either the controlling as well as the power supply of two direct current motors. Additional power supplies for the motors or a huge amount of cables are not necessary anymore. The expansion board enables the control with a constant voltage between 5 to 35 V. The motor driver enables us to control the motors individually and more efficiently. the direction of wheels and increasing or decreasing the motor's speed individually is possible because of the motor driver.

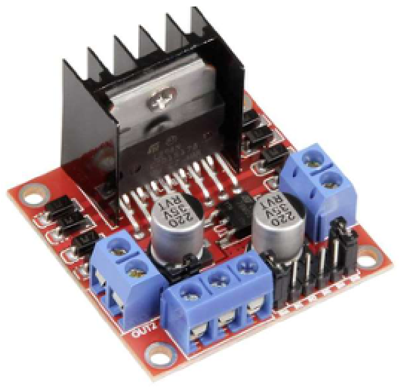


Fig. 5. SBC-MotoDriver2 (L298N)

https://www.conrad.de/de/p/joy-it-motormodul-2-u-4-phasen-6-bis-12v-1573541.html

5) IR Sensor (ST1140) : With this Ir Sensor (Figure 6) any vehicle can walk only along a one-line way. detector move from white to black, it could output TTL signal So if we draw one black line between the two wheels of your car, it will walk along the expecting road. We have used three of them to efficiently keep our vehicle on the grid line.

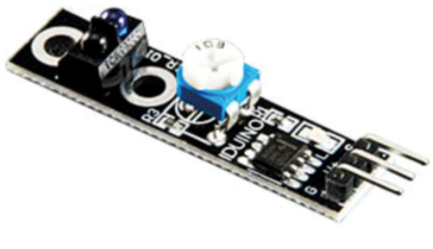


Fig. 6. IR Sensor(ST1140)

https://www.reichelt.de/arduino-ir-linienfinder-ard-line-finder1-p282520.html?trstct=pos$_0nbc = 1$

6) Ultrasonic Sensor(HC SR-04) : The ultrasonic sensor (Figure 7) is mainly used for detecting an object. We have also used this for the same purpose.

Fig. 7. Ultrasonic Sensor(HC SR-04)

https://www.conrad.de/de/p/iduino-st1099-ultraschallsensor-1-st-1616245.html

7) RGB Color Sensor (TCS3200): With this Color Sensor module (Figure 8) it is possible to determine the composition (red, green, and blue components) of colors. The color sensor also called a colorimeter, emits a signal at its outputs, the frequency of which is set to the intensity of the color measured. In addition, this signal can be converted into a color value by microcontrollers (with real-time timers). We have used it to detect the color of the lines on the grid.

For building the chassis of the system we had two options:

- Laser-cut Plywood:Thickness of 6 mm
- 3D printed PLA parts: Maximum Material Consumption of 1 Kilogram.

We also had a dimension constrain of maximum 30*20*20 CM for the overall size of the robot.

III. System Modeling

Modeling the system is very crucial to visualize a system before constructing it. We have used SysML and UML modeling language to provide a clear idea about our system. The user is also able to find some flaws in the system by visualizing

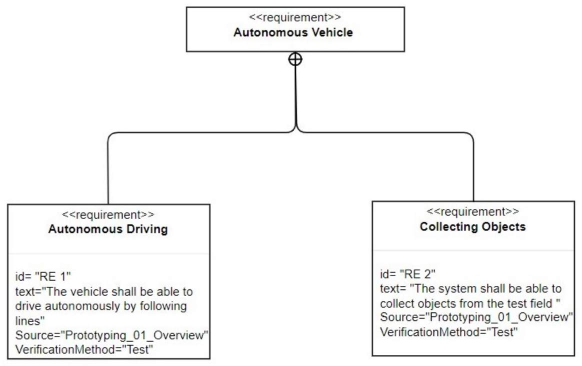
Fig. 8. RGB Color Sensor (TCS3200)

https://www.reichelt.de/arduino-farbsensor-fuer-arduino-tcs3200-ard-color-sensor-p192148.html?search=Arduino+-+Farbsensor+f

the concept. We can save a lot of time and expense by using various modeling languages and also verify our system. We will describe all the diagrams we have used for our modeling system and these are the following:

A. Requirement Diagram

We have developed a requirement diagram by analysing the test grid and the tasks mentioned in section I II.



<<requirement>>
Autonomous Vehicle
<<requirement>>
Autonomous Driving
id= "RE 1"
text="The vehicle shall be able to drive autonomously by following lines"
Source="Prototyping_01_Overview"
VerificationMethod="Test"
<<requirement>>
Collecting Objects
id= "RE 2"
text= "The system shall be able to collect objects from the test field "
Source="Prototyping_01_Overview"
VerificationMethod="Test"

Fig. 9. Requirement Diagram part 1

B. Use Case Diagram

A use case diagram demonstrates graphically how a user may interact with a system. It also shows the interactions between a system and its actors. It describes the high level functions and scope of a system. Normally a use case diagram is made primarily before creating an activity diagram or a sequence diagram, as an analysis based approach in Systems Engineering.

Figure 13 is a Use Case Diagram of our farming robot. It has mainly four use cases for the prototype that we have built. The use cases are driving autonomously, detecting and avoiding obstacles, collecting objects and bringing the objects

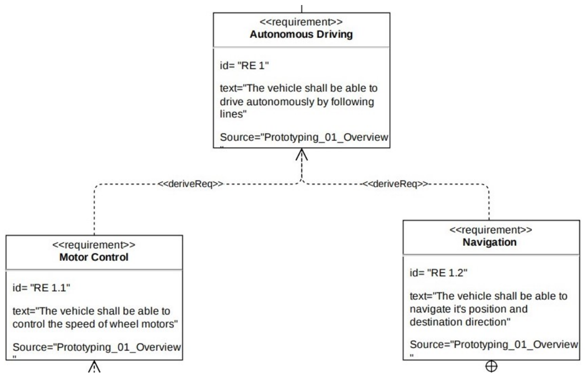

<<requirement>>
Autonomous Driving
id= "RE 1"
text="The vehicle shall be able to drive autonomously by following lines"
Source="Prototyping_01_Overview"
<<deriveReq>>
<<deriveReq>>
<<requirement>>
Motor Control
id= "RE 1.1"
text="The vehicle shall be able to control the speed of wheel motors"
Source="Prototyping_01_Overview"
<<requirement>>
Navigation
id= "RE 1.2"
text="The vehicle shall be able to navigate it's position and destination direction"
Source="Prototyping_01_Overview"

Fig. 10. Requirement Diagram: Autonomous Driving

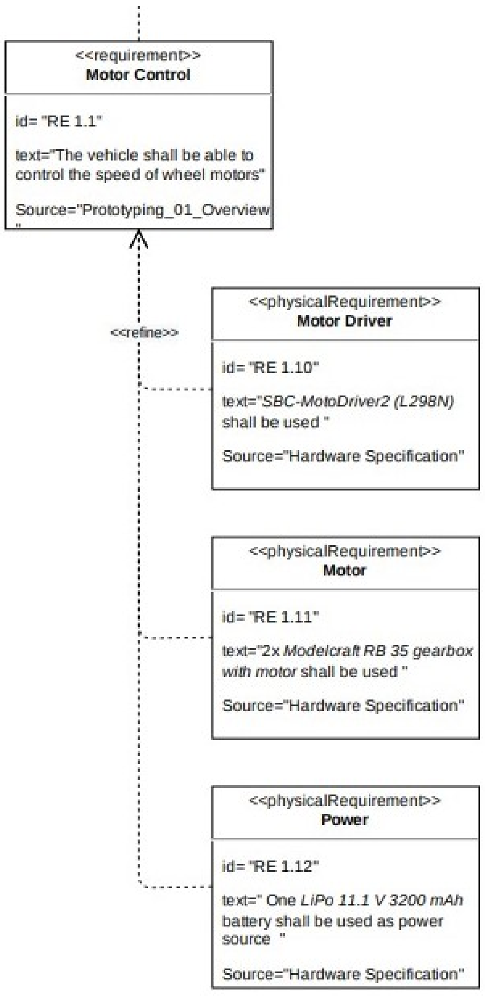

<<requirement>>
Motor Control
id= "RE 1.1"
text="The vehicle shall be able to control the speed of wheel motors"
Source="Prototyping_01_Overview"
<<refine>>
<<physicalRequirement>>
Motor Driver
id= "RE 1.10"
text="SBC-MotoDriver2 (L298N) shall be used "
Source="Hardware Specification"
<<physicalRequirement>>
Motor
id= "RE 1.11"
text="2x Modelcraft RB 35 gearbox with motor shall be used "
Source="Hardware Specification"
<<physicalRequirement>>
Power
id= "RE 1.12"
text=" One LiPo 11.1 V 3200 mAh battery shall be used as power source "
Source="Hardware Specification"

Fig. 11. Requirement Diagram: Motor Control

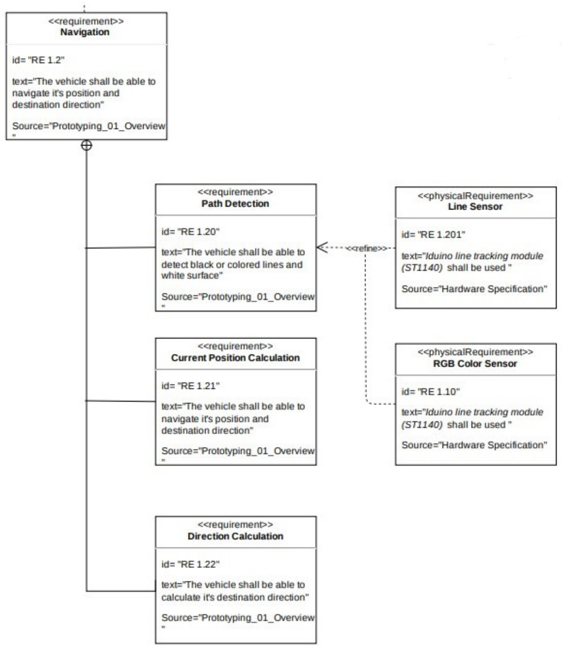

<<requirement>>
Navigation
id= "RE 1.2"
text="The vehicle shall be able to navigate it's position and destination direction"
Source="Prototyping_01_Overview"
<<requirement>>
Path Detection
id= "RE 1.20"
text="The vehicle shall be able to detect black or colored lines and white surface"
Source="Prototyping_01_Overview"
<<refine>>
<<physicalRequirement>>
Line Sensor
id= "RE 1.201"
text="Iduino line tracking module (ST1140) shall be used "
Source="Hardware Specification"
<<requirement>>
Current Position Calculation
id= "RE 1.21"
text="The vehicle shall be able to navigate it's position and destination direction"
Source="Prototyping_01_Overview"
<<physicalRequirement>>
RGB Color Sensor
id= "RE 1.10"
text="Iduino line tracking module (ST1140) shall be used "
Source="Hardware Specification"
<<requirement>>
Direction Calculation
id= "RE 1.22"
text="The vehicle shall be able to calculate it's destination direction"
Source="Prototyping_01_Overview"

Fig. 12. Requirement Diagram: Navigation

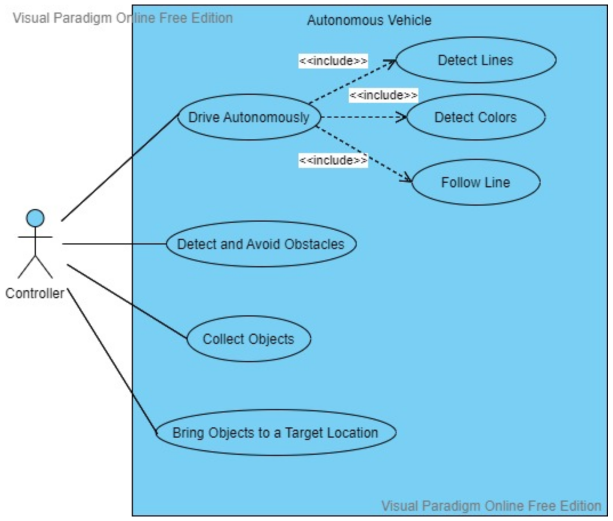

Visual Paradigm Online Free Edition
Autonomous Vehicle
<<include>>
Detect Lines
<<include>>
Drive Autonomously
Detect Colors
<<include>>
Follow Line
Controller
Detect and Avoid Obstacles
Collect Objects
Bring Objects to a Target Location
Visual Paradigm Online Free Edition

Fig. 13. Use Case Diagram

to a target location. Driving autonomously is done mainly by detecting lines with the IR sensor, detecting colors with the RGB color sensor, following those lines using those both sensors and taking turns when needed.

C. Block Diagram

A block Diagram provides a high-level overview of central system components, key process participants, and important functional relationships.

In Figure 9 the main vehicle block has been defined with functions, parts, and some constraints. The battery block has been connected to our main vehicle. The motor driver and sensors are also connected to the main block.

We have used three wheels for our vehicle. In figure 10 we can see that the wheels have been connected to two motors as well as the chassis.

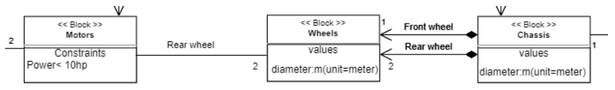


Fig. 14. Block Diagram part 2

D. Constraint Diagram

In Unified Modelling Language, a constraint diagram represents constraints of a system which are extension mechanisms that are used to refine the semantics of a model.

Figure 15 is a Constraint Diagram of our prototype robot. The diagram depicts 4 main Constraints and they are Power Source, Light Source, Speed and Weight Limit.

Power source is a constraint because our robot drives using lithium battery and speed of the motors varies depending on the power level of the battery. This can be an issue while taking turns as we depend on the speed of the motors for a precise turn of the robot.

Speed is considered as a constraint as the robot needs to run on an optimum speed to be on the line all the time. If the speed is too much then it is possible that the robot will go off the track while taking a turn. At the same time, if the speed is too low, the robot does not move due to inertia and weight of the robot.

Light source is also considered as a constraint in terms of using the color sensor. Because the color sensor reads different values if the lighting condition changes. So we need to make sure a stable lighting condition in order to read the right colors at all times.

Weight limit can be a constraint, depending on the weight of the object that needs to be moved or the weight of the robot itself. Because both of these weights can play a role manipulating the speed of the vehicle. Since the example object that we are using for testing our project is not so heavy, this constraint is not so important for the time being. But it should be considered with more emphasis if we plan to experiment with different objects with more weight.

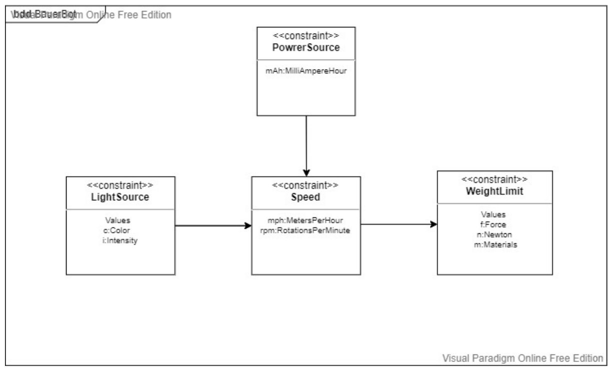


Fig. 15. Constraint Diagram

E. Internal Block Diagram

An Internal Block Diagram explains the internal structure of block elements, like the connection between ports and parts.

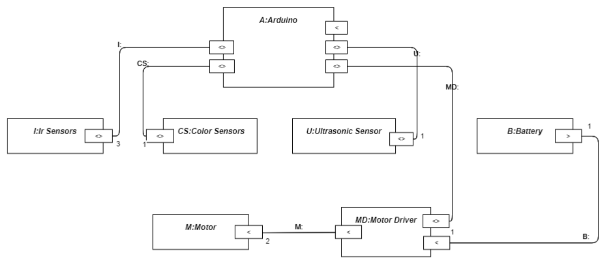
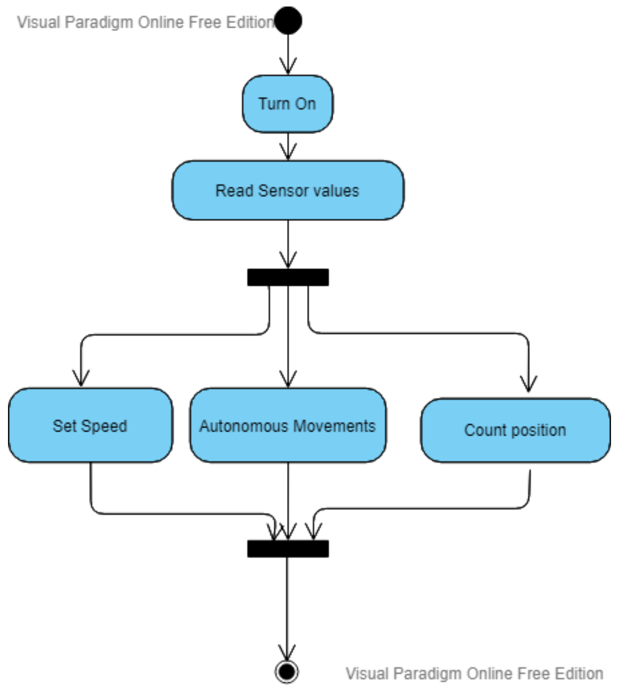


Fig. 16. Internal Block Diagram

In Figure 16 we can observe how the components are connecting and also the power flow and direction of electrical signals. Our battery is mainly connected to the motor driver. The motor driver supplies power to the motors as well as the Arduino. The Arduino supplies power to sensors and gets the reading data from sensors.

F. Activity Diagram

Activity diagrams are visual representations of workflows of step-wise actions with support for choice, iteration, and concurrency. An activity diagram explains a sequence of actions or flow of control in a system.



Visual Paradigm Online Free Edition
Turn On
Read Sensor values
Set Speed
Autonomous Movements
Count position
Visual Paradigm Online Free Edition

Fig. 17. Activity Diagram

We have used an activity diagram to present the main action flows and also concurrent steps in our vehicle. In Figure 17 we can notice that after turning on the system, it will always read all the sensors data. After that our main concurrent functions set speed , autonomous movements and count position with run continuously. The set speed function will keep our vehicle on track by reading Ir sensor data and increasing the individual motor speed. The count position function will calculate the lines on grid .

G. Sequence Diagram

A sequence diagram shows how objects in a system interact with each other. It also represents the time sequence in a system.

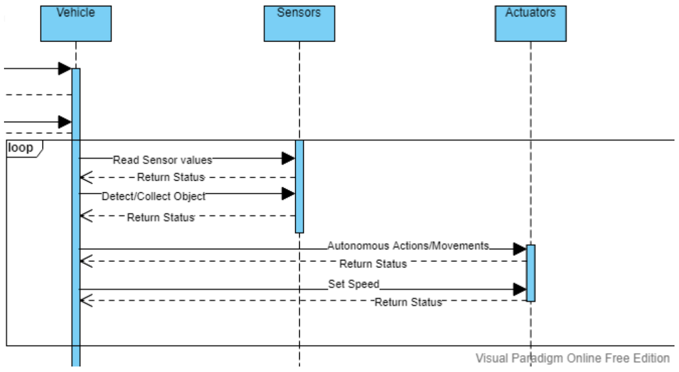


Fig. 18. Sequence Diagram

We can see in Figure 18 how the objects are passing messages between each other. The same functionally as the activity diagrams are operating here in the loop.

IV. Simulation

After creating the system models we simulated our system using Tinker-CAD. We wrote our Arduino code according to our State Machine diagram and connected the components according to the pin connections provided by the instructor. For simulating the test environment we used the IR remote and mapped different IR remote inputs to possible different IR sensor readings. The simulation was successful, which confirmed that our state machine logic was correct.

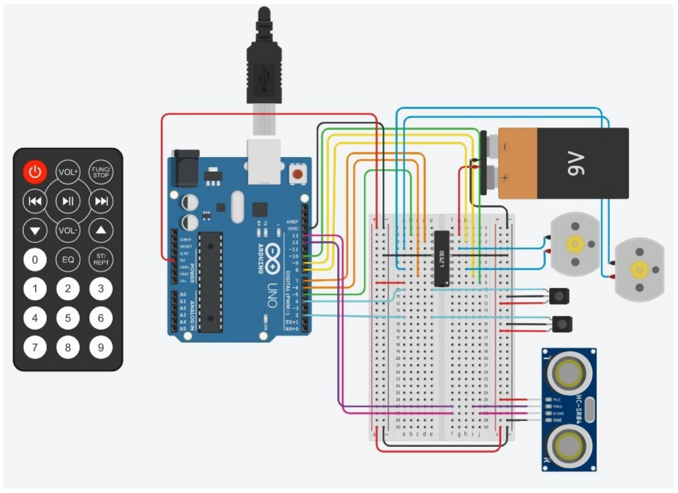


Fig. 19. Tinker-CAD Simulation Circuit

V. Design

First of all We discussed among ourselves about the possible outlook of the prototype. We had to keep in mind how and where to place the specified hardware components. We had to consider their dimensions, their functions and the dimension constrains of the whole robot. We drew some sketches in paper (Figure 20) and after finalizing the layout we used solid works to draw a complete 3D model of the robot.

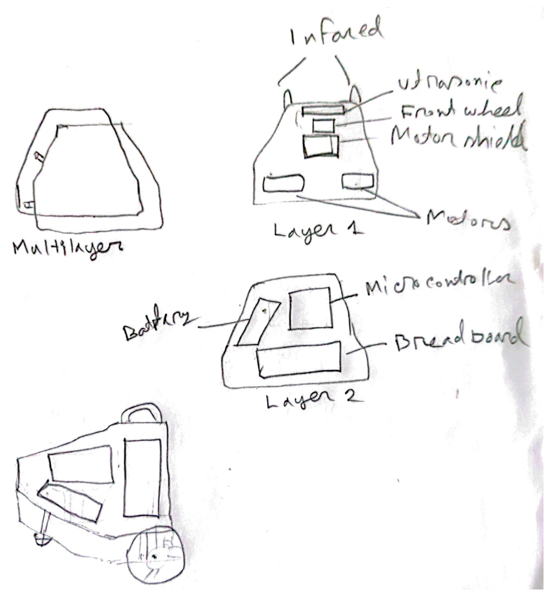


Fig. 20. Hand Sketch

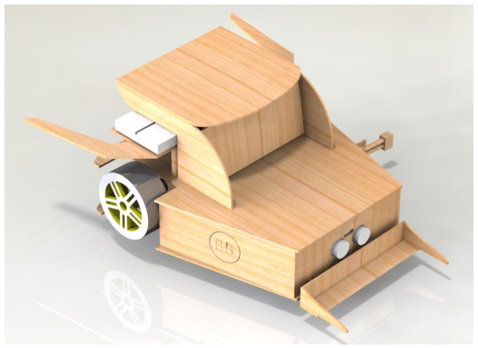


Fig. 21. Final Design

Figure 21, is our final SolidWorks model which we have used for our prototyping part and in the SolidWorks part we have explained the individual part's in detail.

A. Design Motivation

We opted for a Super Hero car themed design because we believe farmers are our greatest Superheroes. Our design have multiple layers for placing the components.Our aim was to have a clean looking design and to hide the components and wires inside the robot. We have discussed our major design decisions below.

- The bottom layer holds the motors, the battery and the motor driver. The RGB color sensor and the IR sensors are attached to the external surface of the bottom layer, facing the floor.The wires from the sensors travel through a hole to the internal cabin and then to the breadboard. This way we have minimized wire visibility to give the robot a clean and nice outlook.
- We have used two brackets to attach the motor to the main chassis. We opted this design so that we can detach the motor if necessary by simply unscrewing the brackets. The motors, motor driver and the battery are placed together to minimize wire travel.
- The second layer acts as the cover for the first layer. It is attached to the first layer with fixed screws on the back end and using magnet on the front end.It has a hinge in the middle. We can detach the magnet and fold the front half of the cover to access and inspect the components on the first layer. The second layer holds the Arduino on it's back end. It also has a hole near the hinge for all the wires from the bottom layer to come out and connect to the Arduino and the breadboard.
- On top of the second layer there is a small stage for placing the breadboard. This stage covers the Arduino and is attached to the second layer using 2 columns. The shape of this stage gives the car it's themed look.
- We have a harvester attached to the front of the bottom layer. Initially the requirements included collecting objects from the grid using a harvester. We designed the harvester symmetrically with the breadboard stage to complete the look. The whole car has a narrow front, wide back shape just for design purpose.

B. Solidworks Parts

As we have described before our design motivation now will more focus on individual parts we made with Solidworks.

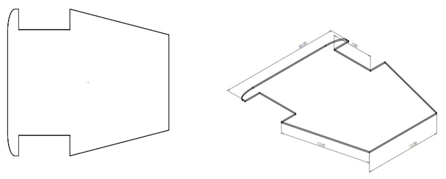


Fig. 22. Lower Chassis

In Figure 22 we can see the bottom layer which is the main base of our vehicle and holds the major components like sensors, motors, and the battery.

We have used a multi-layer design for our vehicle which we can already observe from the hand sketch. In Figure 23, our middle layer act as a cover for the bottom layer. We also have a cut-out to connect the wires between the components.

In Figure 24, we covered both sides with the side body layers. It also shows the logo 'BB' (Bauer Bot) of our vehicle. By closing both sides our internal components also got a safer structure.

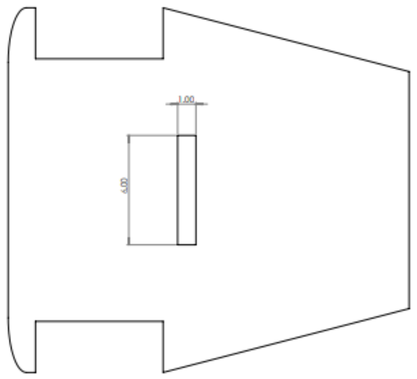


Fig. 23. Second layer

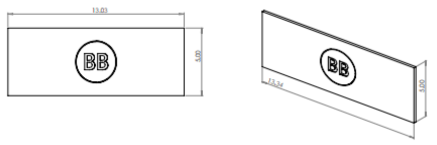


Fig. 24. Side body

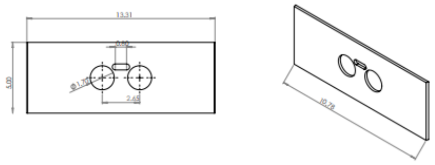


Fig. 25. Ultrasonic holder

Figure 25 is our Ultrasonic sensor holder which is a very crucial part of our vehicle. It holds the sensor very durably.

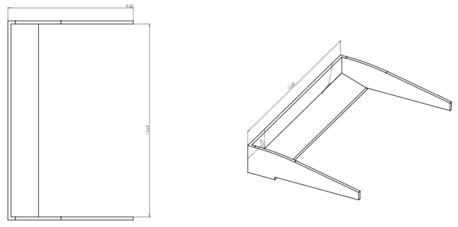


Fig. 26. Object Harvester

In Figure 26, we designed an object pusher or harvester for future farming implementations. It can be attached or detached from the body as well as any kind of other harvester or pusher type also can be used later on.

Figure 27, is our breadboard holder which keeps the breadboard attached to our vehicle. We can also notice the dimensions from the figure.

Figure 28 is our Breadboard shade which keeps the breadboard safe from the physical objects.

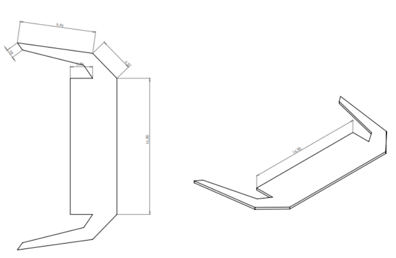


Fig. 27. Breadboard Holder

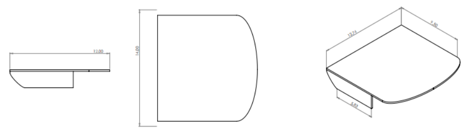


Fig. 28. Breadboard Shade

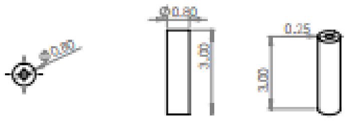


Fig. 29. Support Column for Breadboard Stage

In Figure 29, we used some of these joints to make the breadboard holder and shade stable. these parts are used to join figure 14 and figure 15 in our vehicle.

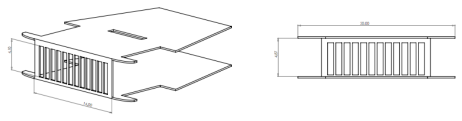


Fig. 30. Rear Components Shield

In Figure 30, the Rear components shield is used to protect the components and also to keep the temperature normal inside our vehicle.

Figure 31, is the front component shield which is used for giving our vehicle a futuristic look and stability.

Figure 32, is one of the most important parts of our vehicle which holds the motors in the bottom layer. It also prevents the motors from causing any vibration while moving the vehicle so it makes our vehicle more durable.

C. Placement of Components

We tried to place all the components and sensors in a way so that the robot can react in shortest possible time. For example,

- we have placed the color sensor and the IR sensors on the front end of the robot and the motors on the back end. This way the motors can have sufficient time to react once a particular spot is detected by the IR and the color sensors.
- We have placed the Arduino on the second layer and oriented it in such a way so that the serial port is easily accessible for uploading the code.
- We placed the power button on the open and easily accessible surface of the second layer so that we can turn the robot off faster in case it is about to hit something.

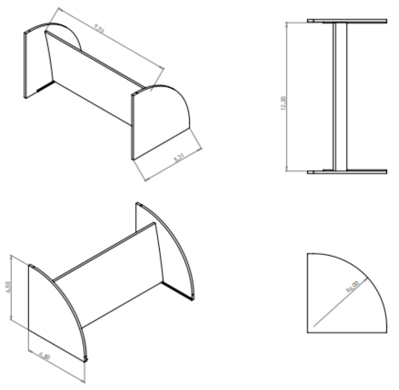


Fig. 31. Components shield

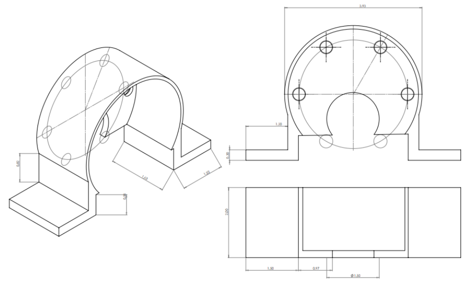


Fig. 32. Motor Bracket

D. Material Selection

For the main body and bigger parts we used laser-cut plywood. The plywood has a thickness of 6 mm, which is strong enough for building the main body. It is also fast and easy to modify and reproduce a part using laser-cut plywood compared to 3D print. The motor bracket and the support pillar for the breadboard was 3D printed as they had geometric shapes impossible to manufacture using plywood.

VI. Circuit Connection

We have used a breadboard to connect different components with the Arduino and the power supply. The connections are explained below

a) Power Supply: The provided 1.1 Volt battery is our only power supply. Arduino Uno can take only 5V as power input. Therefore we connected the battery output to the motor driver. The motor driver has a 5 volt output pin. we connected

Component	Component Pin	Connection
Left IR	S	Arduino A0
	V+	Breadboard 5V
	G	Breadboard Gnd
Right IR	S	Arduino A1
	V+	Breadboard 5V
	G	Breadboard Gnd
RGB Color Sensor	S0	D0
	S1	D1
	S2	D2
	S3	D3
	Out	D4
	GND	Breadboard Gnd
	VCC	Breadboard 5V
	LED	Breadboard Gnd
Motor Driver	IN 1	9
	IN 2	8
	IN 3	7
	IN 4	6
	EN A	10
	EN B	5
	V in	Battery (+)
	GND	Battery(-)
		Breadboard

this 5 volt output pin to breadboard and user that as 5V power source for all other components. We also connected the battery ground to the breadboard to use it as a common ground for all the components.

b) Pin Connections: We used the following pin connections to connect different components with the Arduino Uno:

VII. Programming

We have used one Arduino as the controller unit for the system. We programmed the Arduino micro controller using Arduino IDE and Arduino programming Language(C++). Our major programming decisions are discussed below:

A. Real-time ability

In our use case the system must react as soon as possible for smooth and successful operation.For example, the robot must instantly stop when it detects a turning point. It would not be able to find the line to follow if the turning is not done in the right place. In sequential coding method it takes much longer for the robot to react to a change in sensor inputs. It is because the loop executes all the commands sequentially which delays reading the sensor value and calculating the decision. We therefore used threads to give the program real-time ability. The ATmega328P micro-controller of Arduino Uno is a single core micro-controller. Therefore actual parallel execution is not possible. However, we have used Arduino FreeRTOS library which allowed us to create multiple threads for different tasks and virtually execute them in parallel. This way we have highly reduced the reaction time of the system.

B. Use of Switch Case

We have implemented our state machine using switch case wherever possible. This made the logic implementation and debugging a lot easier. We could easily figure out which state or case we have to debug or improve. We could also figure out in which state the robot is showing unwanted behaviour by simply just printing the state to the serial monitor. By analysing the behaviour we could come to a solution in shorter time.

C. Keeping the code DRY

We have followed DRY coding method to keep the code clean, modifiable and understandable. Our state machine requires repetition of certain commands such as motor control and reading sensor values. Instead of writing these commands again and again, We used functions for these commands and called the functions where necessary.

D. Minimizing Memory Consumption

We have used many local and global variables for assigning the pins and for other computation purpose. We have realized that in all cases the values can be stored only in 8 bit. Therefore instead of using regular integer variables (int), which requires 16 bit for each variable, we have used uint8t. This data type requires only 8 bits to store a variable. This way we have reduced our memory consumption for variables to half of the regular amount.

VIII. Input-Output Logic

A. Line Following

For following the line we have used two Infrared Sensors. Both sensors placed in such a way that they stay over the line edge. The reading for both sensor is 0 when they are on the colored line. The reading is 1 if they detect black surface. If the IR sensor on the right detects black surface this means the vehicle is out of the line towards right and needs to be turned to the left to adjust direction. This is simply similar for the left IR sensor, in which case the vehicle needs to be turned to the right. The turning must be short and precise. Otherwise the vehicle will completely go out of the line. Therefore the turning takes place until both sensor detects the colored line. For turning to a particular direction, we increase the motor speed of the opposite side and decrease the speed of the motor on the turning side.

B. Line counting

We need to count the lines in order to navigate the vehicle's position on the grid. We have used two methods for counting the lines.The first method is using the RGB color sensor and the second one is using one additional IR sensor with two line detecting IR sensors.

a) Counting with RGB Color sensor: The test grid have lines with four different colors: Purple, Orange, Cyan and Green. There are two repeating pattern of colors in the lines. On the X axis Cyan and Orange is repeated. On the X axis Purple and green is repeated.Therefore we used the corresponding colors for each axis to count the lines.

The RGB color sensor gives distinct readings for different color arrays in these four different lines. we differentiated the colors using this distinction. We then followed the logic shown in figure to count the lines using RGB sensor reading. This diagram shows the count logic for X axis but it is similar also

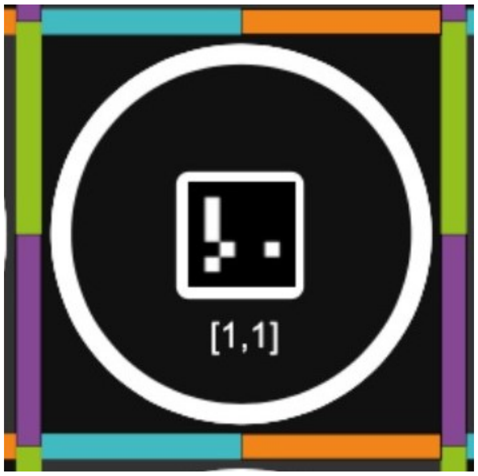


Fig. 33. Color pattern of the grid.

for the Y axis. We incremented our line count variable once it detects Cyan color. We then switch the state to Orange so that the increment looping does not continues as the vehicle will drive over Cyan line for a long period. In this state the vehicle is still over the Cyan line but we do not increment line count. Once the sensor detects Orange line we switch back to Cyan state. Now the vehicle is on the orange line. Once the orange line ends and Cyan is detected we increase line count again and repeat the process.

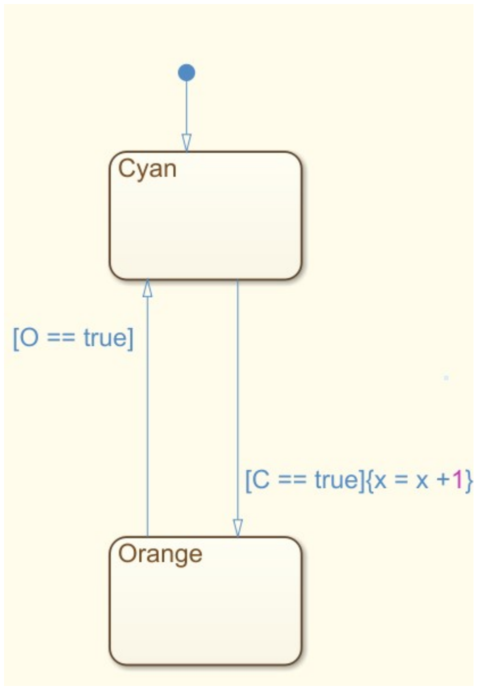


Fig. 34. RGB line count logic

b) Counting with IR Sensors: The RGB color sensor readind was not always consistant. It showed different value range depending on the battery power and lighting condition. Therefore it was not convenient to use the RGB sensor method to implement the functional requirements. We opted a better and more accurate method for line counting using IR sensors.

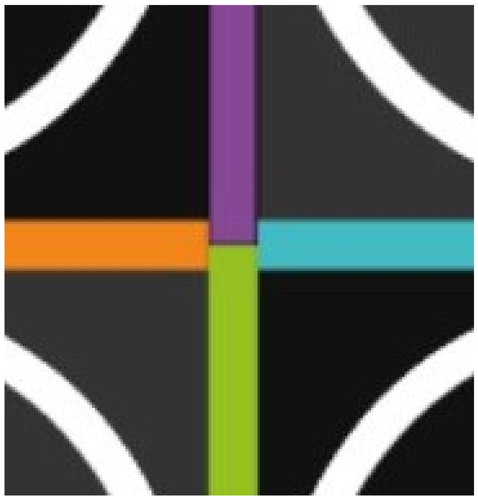


Fig. 35. Grid Junction

We have used one additional IR sensor on the front end of the lower layer of our vehicle. we named it IRCount. The sensor was placed out of the line, over the black surface. The sensor reads 1 on black surface and 0 on colorful lines. As the vehicle drives forward following the line, IRCount detects the colored line over the junctions. By counting the junctions we we counted the lines. We used the same logic as used for RGB method for the increment of counting variable.

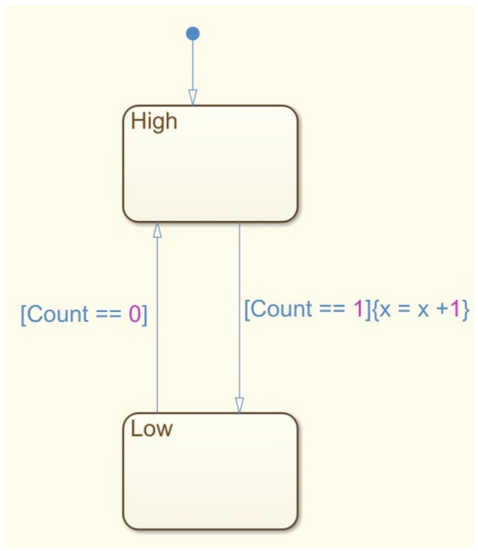


Fig. 36. IR Line Count Logic

IX. Code explanation

As mentioned before that we have implemented our program using threads, we will discuss the threads, main switch cases and most important functions in this section.

A. Navigation

We have used a thread for detecting the vehicle's position on the grid and deciding turning directions.The thread is named as *positionCount*. This thread reads the IR sensor values. Afterwards it enters the switch case.

a) Switch Case: The switch case always start with *X axis count*. This is because we always start our vehicle from the X axis. In *X axis count* the current position is calculated based on the sensor readings. If the *target X* value is greater than *current X* value, we increase *current X* value by 1 each time the vehicle passes through a junction. if the *target X* value is smaller than *current X* value we decrease the *current X* in the same manner. After the increment or decrement we calculate *DestinationX*= (*target X*-*current X*). Once the vehicle reaches it's *target X* coordinate, *DestinationX* value becomes 0. It then enters the *turn* state. This state evaluates weather the vehicle have reached it's target coordinates or not. based on the decision the vehicle executes either the *turning* function and drives on the next axis after successful a turn or it enters in *destination* state and sets the new destination. Figure shows the state flow of this thread.

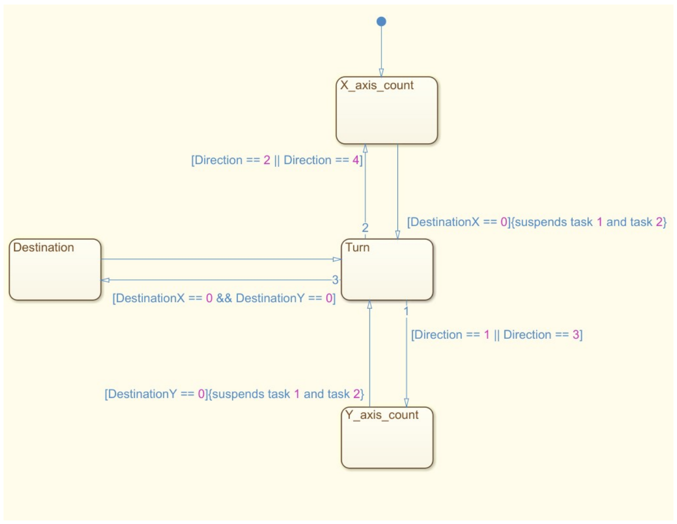

X_axis_count
[Direction == 2 || Direction == 4]
[DestinationX == 0]{suspends task 1 and task 2}
Destination
Turn
[DestinationX == 0 && DestinationY == 0]
[Direction == 1 || Direction == 3]
[DestinationY == 0]{suspends task 1 and task 2}
Y_axis_count

Fig. 37. Position Count State Flow

b) turning: This function executes a switch case. The switch case considers 4 cases for four directions. Based on the vehicle's current direction, current coordinate and target coordinate, the vehicle takes either left or right turn. After a successful turn the direction value is set according to the new direction of the vehicle.

The vehicle drives on the new axis in the same manner until it reaches the destination coordinate. Once the first target is reached the destination coordinates are changed to the second target coordinates and the journey continues. The vehicle stops once it reaches the second target location.

During the turning state we suspend our thread for forward driving and set necessary speed for the motors. Once the turn is complete we resume our forward driving thread.

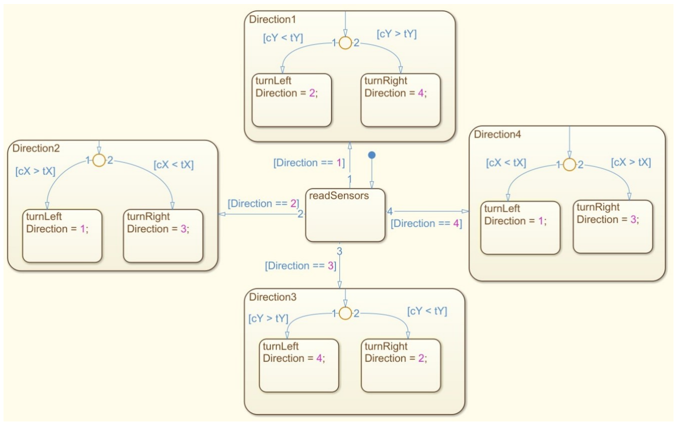


Fig. 38. Turning Decision Logic

B. Line Following

We noticed a difference in RPM of our left and Right motor with same value for PWM. Moreover, Our motors showed different RPM on different battery charge level. Therefore we often had to manually adjust the PWM to achieve the right speed for the motors. As mentioned before, turning for line following had to be very precise in order to avoid loosing the line. Setting the right PWM was therefore mandatory to achieve the right speed.

To solve this problem we used a gradual increment of the motor speed until it achieves sufficient speed and the vehicle moves to the right position. Figure **??** shows the logic for increasing motor speed and tuning.

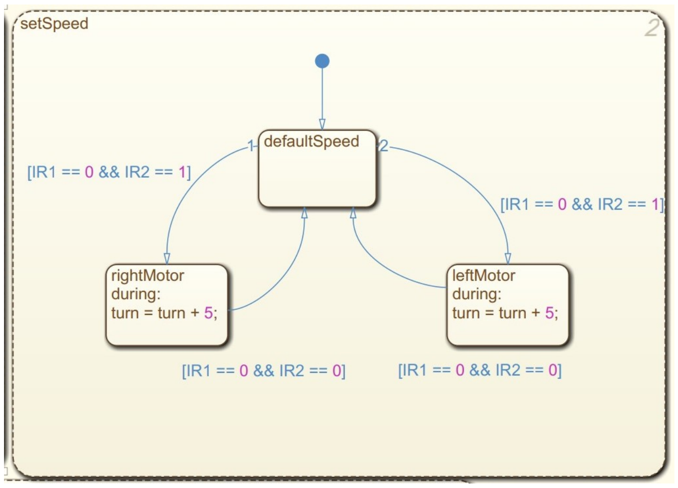


Fig. 39. Motor Speed Logic

We have already discussed input-output logic in section VIII(A). IR1 is placed on the left and IR2 on the right. When the vehicle needs to adjust to the right, we increase the left motor speed and the right motor is in complete stop. Once the vehicle is in the correct position we set default speed for the both motors. Similarly we do the same when we need to adjust our course to the to the left.

X. Troubleshoot

While developing our prototype we faced many difficult challenges. Mainly we had hardware issues which are related to sensors. Firstly we noticed some strange behavior in our motors. We calibrated both motor speeds but despite that, the speed of the motors was changing individually.

Later on, we figured out the problem was inside the motor somehow they got damaged internally So we installed a new pair of motors then our vehicle was operating as we planned. As we explained before we used IR Sensors (ST1140) to follow the grid lines. But in the beginning, we attached the IR sensors outside the line as to when it will detect the line the vehicle will adjust the position automatically. But there is a calibration nob inside the IR sensors which seems to be changing again and again which lead us to have different value every time. There was also some interference of light and color reflection of the grid so to prevent that we attach our IR inside the line which will always read the black value and our vehicle functioned more appropriately.

The same kind of malfunction we faced for our color sensors. The color sensor mainly detects the lighting reflection to detect the color, because of the external light interference our color sensor was giving different values for everything. So we planned to count the line using a color sensor but after having such a problem we used an extra IR sensor to detect our position like when all the value of IR detects black it will count one whole grid So that is how we implemented our scenarios.

Fortunately, we troubleshot all of these difficulties and we were finally able to complete our implementation part.

XI. GitHub link

https://github.com/arfatKamal/Prototyping—Bauer-Bot/releases/tag/V1.1.1

XII. Appendix

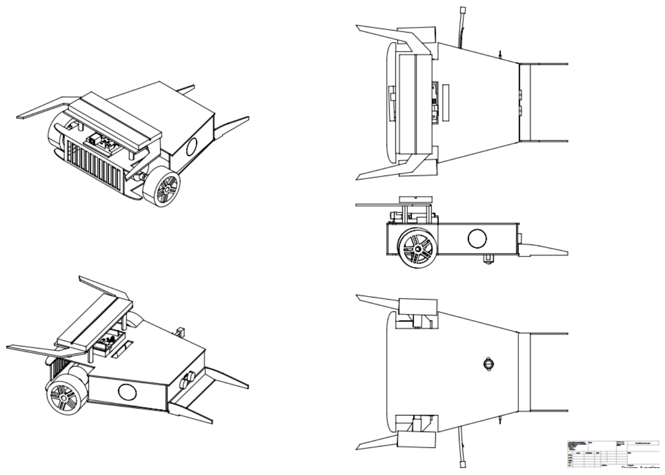


Fig. 40.

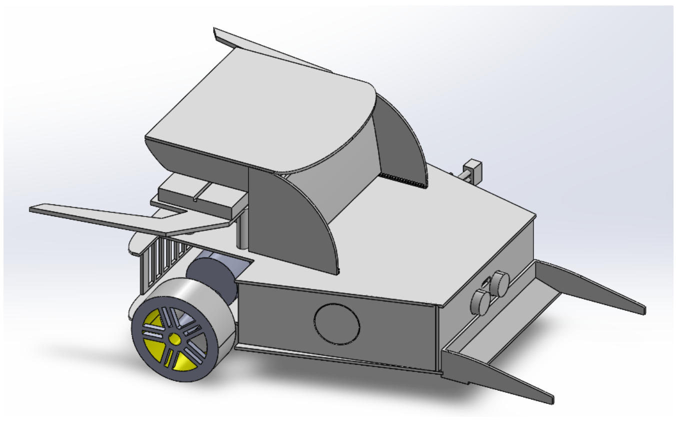


Fig. 41.

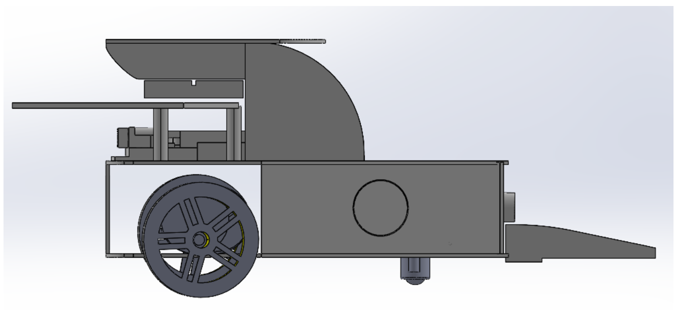


Fig. 42.

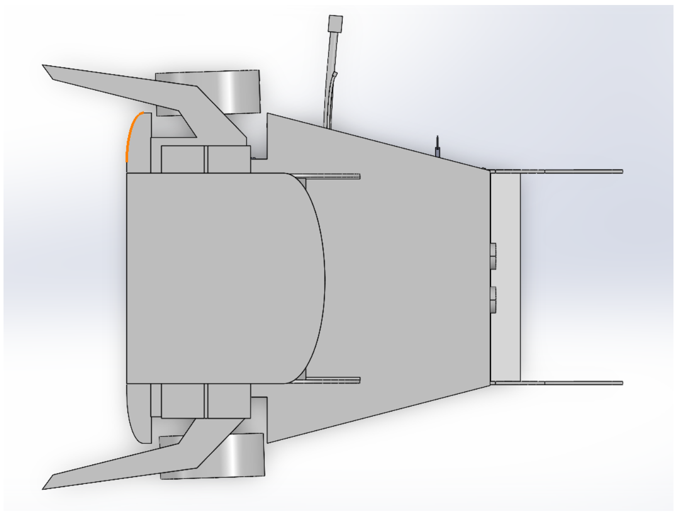


Fig. 43.

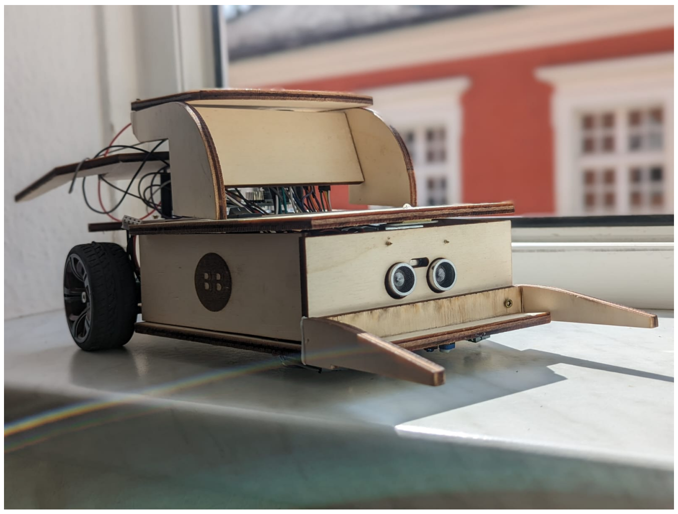


Fig. 44.

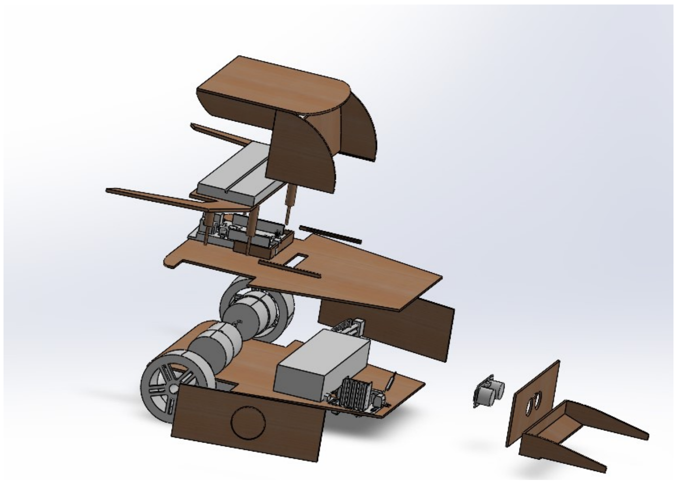


Fig. 45.

Name	System Engineering	Design	Programming	Documentation
Sahat Al Ferdous Fahim	Block Diagram	Lower Stage	positionCount	Section III(C, E)
	Internal Block Diagram	Middle Stage	Setup	Section V(B)
			turnManagerY()	Section IX
		Logo Engrave	turnManagerX()	Section VI
Tasawar Siddiquy	Sequence Diagram	Ultrasonic Bracket	running	Section II(B)
	Activity Diagram	Side Bodies	Pin Assignment	Section V(C,D)
			Ultrasonic Read	Section VII
				Section X
Md Saidur Rahman	Requirement Diagram	Motor Bracket	setSpeed	Section I
		Top Cover	Color Sensor Read	Section II(A)
		Object Harvester	turning	Section VIII
		Rear Shield		Section V (A)
				Section IV
Arfat Kamal	Use Case	Support Column	forward$_{r}ight()$	Section III(B,D)
	Constraint	Breadboard Holder	forward$_{l}eft()$	
			forward()	
			backward()	
			stop()	

Eidesstattliche Erklärung

Hiermit bestätige ich, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen sowie Hilfsmittel genutzt habe. Alle Ausführungen, die anderen Quellen im Wortlaut oder dem Sinn nach entnommen wurden, sind deutlich kenntlich gemacht. Außerdem versichere ich, dass die vorliegende Arbeit in gleicher oder ähnlicher Fassung noch nicht Bestandteil einer Studien- oder Prüfungsleistung war.

Affidavit

I hereby confirm that I have written this paper independently and have not used any sources or aids other than those indicated. All statements taken from other sources in wording or sense are clearly marked. Furthermore, I assure that this paper has not been part of a course or examination in the same or a similar version.

Rahman, Md Saidur	**59494, Soest, 19/06/2022**	[illegible]
Name, Vorname	Ort, Datum	Unterschrift
Last Name, First Name	Location, Date	Signature

Eidesstattliche Erklärung

Hiermit bestätige ich, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen sowie Hilfsmittel genutzt habe. Alle Ausführungen, die anderen Quellen im Wortlaut oder dem Sinn nach entnommen wurden, sind deutlich kenntlich gemacht. Außerdem versichere ich, dass die vorliegende Arbeit in gleicher oder ähnlicher Fassung noch nicht Bestandteil einer Studien- oder Prüfungsleistung war.

Affidavit

I hereby confirm that I have written this paper independently and have not used any sources or aids other than those indicated. All statements taken from other sources in wording or sense are clearly marked. Furthermore, I assure that this paper has not been part of a course or examination in the same or a similar version.

Siddiquy, Tasawar	Hamm,19/06/22	Tasawar
Name, Vorname	Ort, Datum	Unterschrift
Last Name, First Name	Location, Date	Signature

Eidesstattliche Erklärung

Hiermit bestätige ich, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen sowie Hilfsmittel genutzt habe. Alle Ausführungen, die anderen Quellen im Wortlaut oder dem Sinn nach entnommen wurden, sind deutlich kenntlich gemacht. Außerdem versichere ich, dass die vorliegende Arbeit in gleicher oder ähnlicher Fassung noch nicht Bestandteil einer Studien- oder Prüfungsleistung war.

Affidavit

I hereby confirm that I have written this paper independently and have not used any sources or aids other than those indicated. All statements taken from other sources in wording or sense are clearly marked. Furthermore, I assure that this paper has not been part of a course or examination in the same or a similar version.

Fahim, Sahat al Ferdous	Soest, 19/06/2022	[illegible]
Name, Vorname	Ort, Datum	Unterschrift
Last Name, First Name	Location, Date	Signature

Eidesstattliche Erklärung

Hiermit bestätige ich, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen sowie Hilfsmittel genutzt habe. Alle Ausführungen, die anderen Quellen im Wortlaut oder dem Sinn nach entnommen wurden, sind deutlich kenntlich gemacht. Außerdem versichere ich, dass die vorliegende Arbeit in gleicher oder ähnlicher Fassung noch nicht Bestandteil einer Studien- oder Prüfungsleistung war.

Affidavit

I hereby confirm that I have written this paper independently and have not used any sources or aids other than those indicated. All statements taken from other sources in wording or sense are clearly marked. Furthermore, I assure that this paper has not been part of a course or examination in the same or a similar version.

Name, Vorname	Ort, Datum	Unterschrift
Last Name, First Name	Location, Date	Signature
Kamal, Arfat	Lippstadt, 19.06.2022	