Autonomous Vehicle

Prototyping (Group A5)

Shihab Ud Doula
Department of Electronic Engineering
Hochschule Hamm-Lippstadt
Lippstadt, Germany
shihab-ud.doula@stud.hshl.de

Emirkan Sali
Department of Electronic Engineering
Hochschule Hamm-Lippstadt
Lippstadt, Germany
Emirkan.sali@stud.hshl.de

Ukamaka Akumili Anaedu
Department of Electronic Engineering
Hochschule Hamm-Lippstadt
Lippstadt, Germany

Md. Salman Bin Shahid
Department of Electronic Engineering
Hochschule Hamm-Lippstadt
Lippstadt, Germany

md-salman.bin-shahid@stud.hshl.de

Abstract: Technology is evolving every day. In the last few decades tech innovation has given rise to some products are now part of daily human life. Among one of the innovations are autonomous vehicles that has made a new standard of safety and reliability. This paper aims to define a product of such class that we have created by implementing such sophisticated elements which included an Arduino microcontroller, Ultrasonic, RGB and line sensors, a motor, and a motor driver. We will briefly explain the system engineering and programming behind this whole project by defining each component and their functions. We have executed all these elements and made such vehicle which will have a major impact in the Agricultural and farming sector. This vehicle is able life bales of straw according to user demands and offering higher precision is safety and management. This vehicle will be replacing the traditional way of farming and agriculture and will make us go a step further in the world tech and innovation.

Index terms: Autonomous vehicle, microcontroller, Sensors, motor, and motor drivers.

1. MOTIVATION

Technology has changed our lives by increasing the speed of time. We were human. We invented and developed the technologies to change our lives to their best. Now that technology is changing our lives every second. Robots are our new human model and, in the end, only robots control this world. In every sector, the use of Artificial Intelligence, cloud computing, machine learning, predictive analytics are now creating new methods to conduct, operate, and management. According to The Global Report on Food Crises 2022, 193 million people are in food crisis and need urgent assistance across 53 countries and Territories. By using our efficient technological advance, we can make a difference in this sector of food crisis by advancing our agricultural sector to meet people's demand. Our team has worked hard to make such prototyping Vehicle with advanced technical components which requires less human interaction and can do the most difficult job more precisely giving higher safety. This vehicle can accurately map the farming field with sensors and working on its own my user's demand. Its accuracy has been

tested in the lab and the efficiency it provides is being worthy.

Our intention to create such vehicle required a higher precision to develop it from scratch. Each component has studied thoroughly using UML diagrams and software techniques. Then it designed in 3D and laser cutting techniques were used for it. By using advanced programming using a microcontroller we have achieved the accuracy of the vehicle to get the job done. More renditions are required to work it in the real world, but in the next few this we believe our prototype will make a difference in the agricultural sector.

Objectives

Goal: To create an autonomous vehicle that can harvest geometries with box shape.

Problem to be solved: Being able to move and lift bales of straw in crop field automatically by autonomous sensors.

2. REQUIREMENTS

- The vehicle needs to be autonomous and navigate by user specified location
- The system needs to provide a way for the user to interact.
- It should ecological and must recharge and the power source is non burnable
- It should be efficient to complete task and raise economical value to users
- It should minimize errors by optimized code and power.
- The hardware should be well built for potential task, and it should have specified speed frequency.

3. SKETCH OF APPROACH - MODELLING

The initial approach of this project was having a clear idea of system engineering which we learned in the course. This includes the microcontroller components and its related components and using UML diagrams for each section of the system. An appropriate organization of requirements diagram, activity diagram, block diagram, sequence diagram and state machine diagram made the system to be more understanding and preparing for the next following steps. Next, we also did an online simulation of the overall system by implementing the components in a microcontroller and the result were positive. All the diagrams are discussed briefly in the paper to have a clear overview of the prototype.

3.1 Sequence Diagram

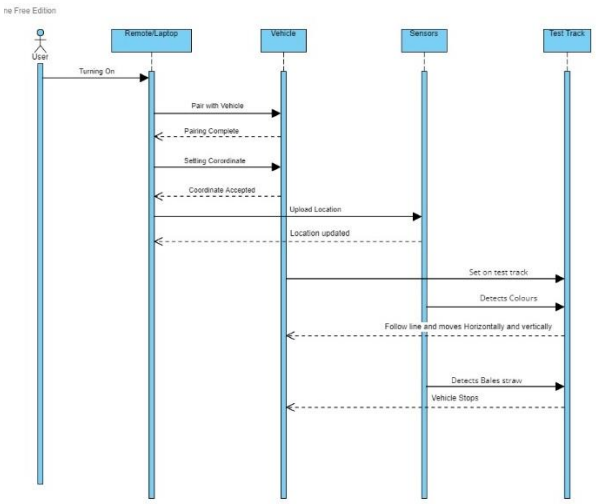

Figure 1: Sequence Diagram

A sequence diagram or system sequence diagram shows process interactions arranged in time sequence in the field of software engineering. It depicts the processes involved and the sequence of messages exchanged between the processes needed to carry out the functionality. In the following diagram, the activates the device from where the coordinates and location are uploaded from. In this case, Arduino from the vehicle is paired with the device via cable. After that the user determines the following coordinates and sets the speed

accordingly through ‘Arduino IDE’. Here the user is given the full priority input any other coordinates or any changes required for the system. The location of the vehicles is given Horizontally or Vertically. Once the code in Arduino is run properly, they vehicle is set to be tested in the track. The ultrasonic sensor the
front detects obstacles or any object to proceed accordingly. For the test track, there are colors which are used for the vehicle to move according. The line and RGB sensor are used to follow the line and detect colors to move forward and taking turns according as programmed. As a challenge from our course coordinators, we must detect the Bales of straw in the field move it aside to clear the track. We have implemented Ultrasonic sensor to detect that, and the aftermath of it is still in progress. We hope to achieve it soon.

3.2 *Block Diagram*

A Block diagram is modular units of system that encapsulates its contents, which include attributes, operations, and constraints. Block diagrams are mostly used for definitions, and its parts are used for applications. Block diagram is used throughout all phases of system specification and design and can also be applied in different systems. It’s also used
in collection of features to describe a system or other element of interest. In this system design (Autonomous vehicle), the Block diagram consist of mostly the 4 main part which are the Navigation, Object detection, Object collection and the constrain constraint Block, as shown in fig 1.

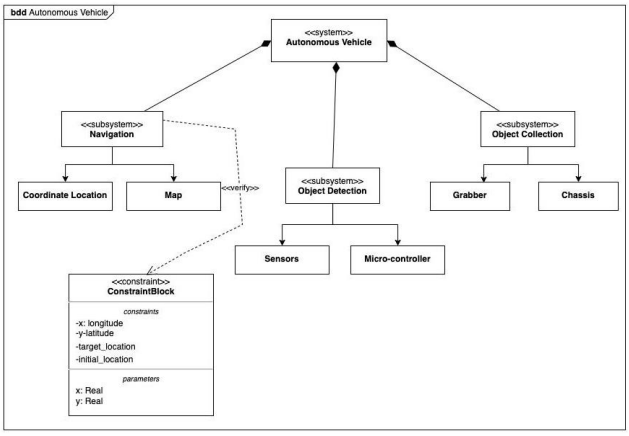

Figure 2: Block Diagram

3:3 Sate Machine Diagram

A state machine diagram models the behavior of a single object, specifying the sequence of events that an object goes through during its lifetime in response to events.

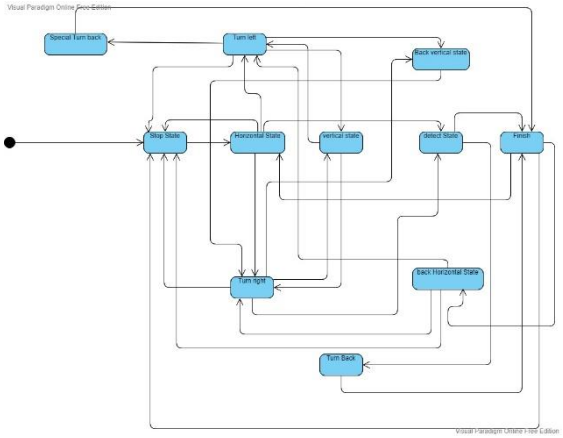

Figure:

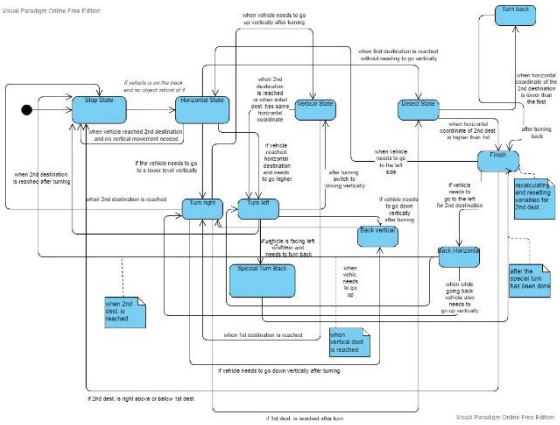

Figure:

4 Design

This part is intended to define the physical prototyping of our project and thus the 3D design, considering the different interactions with the environment and its reactions. Before any real technical inputs were made as regards to the design of the vehicle, adequate level of research was first carried out. Sketches were made for different scenarios of movement. Also, a slide to properly present the idea was also made. During the research, we explored different methods that could enable our vehicle to meet the given requirements, one of which is to follow lines with sensors. Based on the dimensional restriction given for our vehicle, the possibility for a small sized vehicle to pick up objects was not possible hence we had to readjust and eliminate certain functionalities out of our system.

It contains four different parts as follow: Research, Hardware design, and Final model

4.1 Research

By the advanced of modern design techniques, one must do the research behind it. We were given the challenge to create such sophisticated vehicle that is autonomous and work with users demand without human interaction. For such requirements we had to start from scratch by utilizing each component of the prototype and its function. The requirements given to us made us required working with advanced design software like Solidworks and Rhino7. As we are aiming for a Retro Design for our vehicle, we studied some vintage vehicle from 1900's. The design structure ´of such vehicles were easy to understand as well as technology behind them. As we didn't want to make a replica of such vehicle, we analyzed the engineering behind it. More of our research went in search of unique shapes for the design of the vehicle. We made a "Mood Board" which are great for physical or digital collages to range images, materials, text, and other design elements into a format that's representative of the final design's style. This helped us a lot rendering the final design of our retro car. Less effort was given on the internal elements as they were already given from our coordinators. But rendering all the elements required us to see if a simulation works in real world scenario. So, an online simulation on "Tinkercad" helped us realize the real-world simulation of our prototype. One of the most important parts of this projects to know the basics of a 'Microcontroller' which we had an idea from our previous semester. We worked with 'Arduino Uno' which is an excellent micro controller for such case.

4.2 Hardware Design

Goal
Our task was to design and construct an autonomous vehicle which can harvest geometries with a box shape of 50mmx50mmx50mm.

According to the requirements the dimensions of our vehicle were x<30cm, y<20cm,z<20cm. The methods to bring your design vision into the real world are FDM 3d printing with the material PLA and laser cutting with plywood with a thickness of 5mm. As our design approach we were aiming in creating a retro vehicle that shows a sign vintage to it. We did the research of finding unique shapes and design objects that will make our prototype different.

One of the vehicles that inspired us was the **Škoda Hispano Suiza 25/100 PS.** We collected our ideas and thoughts in a **'Mood Board'**. The mood board will give an overall idea of our vehicles design. As we were free to use any 3d/CAD software, we started our initial approach by implementing all internal implements in Tinkercad , we made an initial design simulation inspiring from the above mentioned vehicle. We made the frame above and tried maintaining dimensions to fit all the component just to see an initial look of the design. Initial design

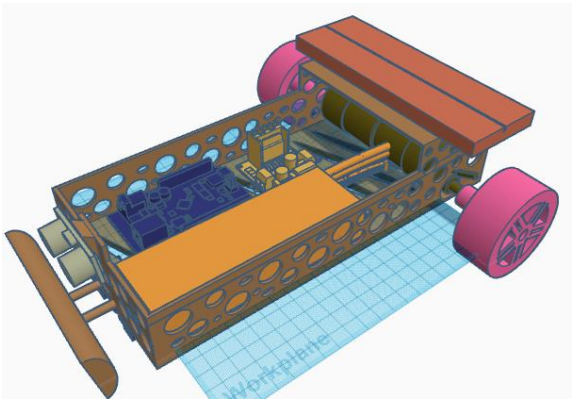

Figure 3: Initial Design

We were given the option to Laser Cut or 3D for our prototype. As we proceeded with our initial design, we faced a challenge in file formats to convert them in .dxf or .stl files while working in Tinker cad. To make precision in our frame our coordinators guided us with ideas to make the vehicle lightweight and to make a more improvised version of our prototype in Rhinoceros.

4.3 Final Model

For the final design application, our vehicle dimension was,

- x<30cm, y<20cm, z<20cm

- Plywood thickness 5mm for laser cut

-Thickness <2mm for 3D

By analyzing overall dimension of the vehicle, we created a model in Rhinoceros 3D which was ready for laser cutting and 3D printing.

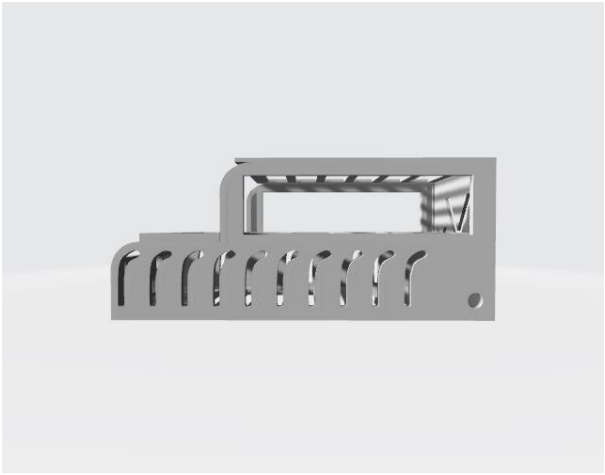

Figure 4: Final Design I

Figure 5: Final Design II

The frame of the vehicle was design to be lightweight by making shapes on the frame by not sacrificing the overall design. It remained rigid and could hold the overall weight of the internal components. This idea is based on modern architecture that utilize both quantity and quality. Our design was ready for production and as our design was a sophisticated, we needed both laser cutting and 3D. Our production was ready in 48 hours. After our production we attached each part of laser cut and 3d design object by screws and wires. Our prototype Finally got a rigid shape as we wanted. Our next step was to implements the mechanical parts in place for the whole vehicle to function.

To power the whole system and to obtain our goal certain elements were used in the prototype which are:

- Arduino Uno Microcontroller
- Motors
- Motor Driver
- Bread board
- Wires
- Ultrasonic sensor
- Line Sensor
- RGB Sensor
- Wheels

Arduino Uno Microcontroller: Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online.

Motor: A DC motor (Direct Current motor) is the most common type of motor. DC motors normally have just two leads, one positive and one negative. If you connect these two leads directly to a battery, the motor will rotate. If you switch the leads, the motor will rotate in the opposite direction.

Motor Driver: The Motor Driver is a module for motors that allows you to control the working speed and direction of two motors simultaneously. This Motor Driver is designed and developed based on L293D IC. L293D is a 16 Pin Motor Driver IC. This is designed to provide bidirectional drive currents at voltages from 5 V to 36 V

Breadboard: A breadboard is a solderless construction base used for developing an electronic circuit and wiring for projects with microcontroller boards like Arduino. As common as it seems, it may be daunting when first getting started with using on

Wires: The 24-gauge wire that is in CAT-5 cable kind of works but is thin enough that it will often come out when you are just moving the board around. 20-22 gauge is just about right and can handle any generic Arduino load that you put on it, as well as things like small DC motors, etc. that draw < 1.5A

Ultrasonic Sensor: It works by sending sound waves from the transmitter, which then bounce off of an object and then return to the receiver. You can determine how far away something is by the time it takes for the sound waves to get back to the sensor

Line sensor: Line sensors detect the presence of a black line by emitting infrared (IR) light and detecting the light levels that return to the sensor. They do this using two components: an emitter and a light sensor (receiver).

RGB Sensor: The light sensor works by shining a white light at an object and then recording the reflected color. It can also record the intensity of the reflection (brightness). Through red, green, and blue color filters the photodiode converts the amount of light to current.

To maintain the precision of each element in its place in the vehicle we did the measurements and after the production it was very much to assemble.

Figure 6: Final Design III

As shown in the figure III, in the front we kept two holes for ultrasonic sensor to detect objects by carefully utilizing the measurements. The upper housing was kept keeping the breadboard and Arduino to have easy access to the user while rendering code and coordinates. Inside of the vehicle have the remaining elements which includes two Motors on the back in each side. A battery to power the whole vehicle and the motor driver. We are using a base wheel on the front of the vehicle which is on base of the vehicle. And on the side of the base attached two-line sensors which will detect the line on the test track. There is RGB sensor which is on the middle base to detect colors on the test track.

5. IMPLEMENTATION

This section is related to the software part of our vehicle, referring to the implementation i.e., coding in the C programming language of all the requirements stated before. we decided to separate the degree of complexity of our code and move step by step i.e.:

- To Follow the line first.
- To understand to take turns in the test track
- Able to detect object on the front.

Our first milestone was to program an Arduino through an online simulation in Tinkercad. We made a online simulation of the system with its component and simulated it online. After that we programmed it in a ready-made prototype for our first milestone. Our circuit had to be programmed in such a way that it would follow a black line on the track with certain speed and completes a lap. An online simulation of our first mile's stone is given below:

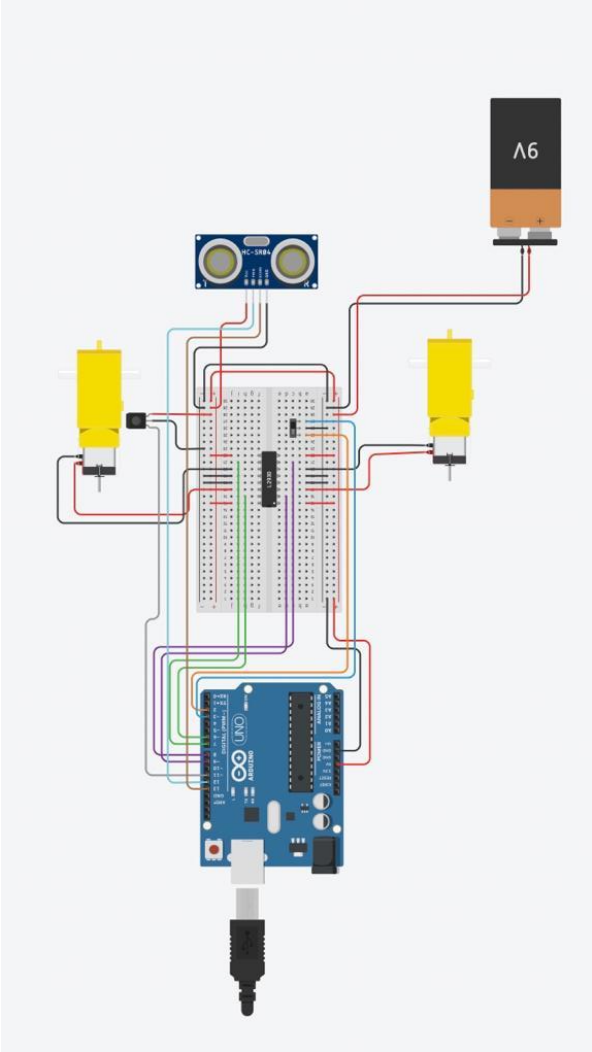

Figure 7: Tinkercad Simulation (without RGB)

The next task was to create a the the our own prototype with new design and in a challenging light situation we need our vehicle to follow the line by detecting various colors with the help of RGB sensor and changed direction according to it.

The code implementation are as follows:

6. CODES

6.1 Final Vehicle Code: Before set up

Firstly, in the code, all the pins that are needed for the components are established and named.
Right after and before the set-up code, various variables are set up for the vehicle to work:

starting with the distance and duration variables for the ultrasonic sensor, then “Red”, “Blue”, “Green” and several “on”-specific color variables for the RGB sensor. 2 important variables “vertical” and “horizontal” are set up and are to be changed when the starting position of the vehicle on the track is changed.
2 “turnback” variables are set up after, which are simply counted up to 2 in the turn back state to indicate for the vehicle that the “turning back” is completed and it should move on to the next state.
It continues with setting up a “way” variable that indicates the vehicle being either being on its way to the 1st destination or on its way to the 2nd destination. Right after come the destination variables, where user input is needed to set up the coordinates for each location the vehicle should go on the track. For both destination there are 2 sets of coordinates for horizontal and vertical location, so 4 variables for every destination. The first two variables are used in the code and might change over the time, while the other two variables are set and only used to refer back to the original goal coordinates for the vehicle.
The variable “facingfrontback” simply changes to 1 whenever the vehicle faces the left or bottom site of the track and changes back when its turned around again. It basically just indicates if the vehicle is going backwards or straight through the coordinates.
A special case variable “startposHorizon” is used when the 1st destination of the vehicle is on the same horizontal axis as its starting point. Another special case set of variables are used when the 2nd destination coordinates are right above or below the 1st destination coordinates.
At last the final variable simply changes the starting state when changed in the code and it should be always 0, unless for testing.

6.2 Set up

After all the components have been set up to work as intended in the set up code, the needed way to the coordinates is calculated by simply subtracting the current position from the goal position.
For the special case of the 1st goal position being on the same horizontal coordinate some variables are changed to cause the needed behavior of the vehicle in the end.

6.3 Functions

void UltrasonicSensor:
sets up the ultrasonic sensor of the vehicle to detect objects in front of it. The Variable “distance” can be used to work with the distance of the object in.

void displayColors:
displays the red, green and blue values that the color sensor gives. Used for calibrating.

void GetColors:
Sets up the color sensor to work under certain

lighting conditions. Displays one of the colors that the RGB sensor recognizes. The sensor can give out Black, White, Green, Magenta, Blue and Orange. Also sets the color variables when on every color.

void drive:
the vehicle drives forward and follows the line automatically when this function is called.

void navigation:
not used in the prototype. It keeps track of the vehicles position on the track when running, using the RGB sensor and the "vertical" and "horizontal" variables.

void Stop:
simply stops the vehicle.

void *turnRight:*
the function for the motor of the vehicle to deactivate the right motor, to start turning right.

void turnLeft:
the function for the motor of the vehicle to deactivate the left motor, to start turning left.

void driveStraight:
the vehicle starts driving straight forward without following the line. Used for turning correctly.

6.4 Main Loop

the main loop uses switch case with eleven states in total.

Stopstate:

The starting and final state of the vehicle, where it simply stops. If placed on the track with set goal, the vehicle will switch into the horizontal state "HoriState".

HoriState:

In the horizontal state, the vehicles drives forward until it reaches the desired horizontal coordinate of the 1st goal. It does so by decrementing the "horizonGoal" variable every time it passes a junction on the track until said variable is valued zero. The vehicle will switch into the "TurnRight" state, when the desired vertical coordinate is less than its actual position, to the "TurnLeft" state when the desired vertical coordinate is above the vehicle. If the vehicles 2nd goal is reached while in the Horizontal state without a need to switch vertical lines on the track, the vehicle returns into the "StopState". When the vehicles 1st goal is reached instead, it will switch to the "DetectState".

VertiState:

In the vertical state, the vehicles drives upwards until it reaches the desired vertical coordinate of the 1st goal. It does so by decrementing the "verticalGoal" variable every time it passes a junction on the track until said variable is valued zero. After reaching its desired vertical coordinate, the vehicle will either switch into the TurnRight state after which it will reach its 1st goal or into the Turnleft state when its about to reach its 2nd goal.

Turnleft:

the vehicle turns left on the track using delays. Depending on battery power, the delays might need adjustment to make the vehicle turn perfectly. The vehicle will switch into the Vertical State "VertiState" or into the back vertical state "BackVerti" if it needs to go up or down on the track after turning. It will switch into the Stop State when its final goal is reached. In a special case where the 1st goal of the vehicle has the same horizontal coordinate as its starting coordinate, the vehicle will switch into a special turning back state when reaching its first goal after turning.

Turnright:

the vehicle turns right on the track using delays. Depending on battery power, the delays might need adjustment to make the vehicle turn perfectly. The vehicle will switch into the vertical or back vertical state depending on the goal coordinates. If it reached its 1st goal after turning right, it will switch into the detect state or if it reaches its 2nd goal after turning, it will switch into the Stop state.

DetectState

the detect state is initially implemented to make use of the ultrasonic sensor to move an object on the track. Our prototype will not use that feature but this state remains for now as a kind of junction state. If the 2nd goal´s horizontal coordinate are greater than the 1st goal´s horizontal coordinate, the vehicle will switch into the finish state and start navigating to its 2nd goal. If the horizontal coordinate of the 2nd goal are lower or equal to the 1st, the vehicle will switch into the TurnBack state.

TurnBack

the vehicle simply turns around on the track taking two left turns or two right turns depending on the desired vertical coordinate of the 2nd goal. Vertical position of the vehicle will either be decremented

or incremented. After the turn, it will switch to the Finish state.

Finish:

the finish state is an in between state where the vehicle stops and recalculates every variable needed for its way to the 2nd goal on the track. If after the turnBack state, the vehicle is already on its 2nd goal (if the 2nd goal is right above or below the 1st), it will switch to the Stop state. Depending on if the vehicle needed to turn back or not, it will enter the HoriState or the BackHori state.

BackHori:

Same as the horizontal state, but driving the other way, the vehicle facing the left side of the track. Instead of decrementing until the desired coordinate is reached, it will increment until the variable is zero again. If no other change in the vertical coordinate is needed for the vehicle to reach its 2nd goal, it will switch into the Stop state. Otherwise it will switch to TurnLeft or TurnRight depending on the desired vertical coordinate being below or above the current position

BackVerti

Same as the vertical state, but driving the other way, the vehicle facing the bottom side of the track. Instead of decrementing until the desired coordinate is reached, it will increment until the variable is zero again. After reaching its desired vertical coordinate, the vehicle will switch into the TurnRight state to turn to its 2nd goal.

SpecialTurnBack

a special state that is only entered in a case where the horizontal coordinate of the first goal is the same as the starting horizontal coordinate of the vehicle. After the vehicle reached its first goal, it will turn back the same way as in the TurnBack state, but the other way around, so it will face the right side of the track after turning back. Depending on the 2nd vertical goal it will take two right turns or two left turns and increment or decrement its vertical position. Lastly it will switch into the Finish state

With this code running on the vehicle, it will be able to go up to two locations on the track and stop there on its horizontal line.

7. SUMMARY

Thinking about the food uncertainty in the future and utilizing technology in modern agriculture we have tried to create such vehicle that can sustain for a long time. A long-term promise can be made with the farmers, by offering such means of Agricultural suppliance. We tried to maintain the design elements by focusing the farmers and their easy adaptability with the product.

Our product is an example of technological excellence showing how tech and real world evolves. We had an excellent experience with our prototype, and we believe this prototype will someday be a vehicle that will excel for our future thinking.

8. REFERANCES

- Clairre. (2021, June 11). *Has Technology Really Changed Our Lives For the Better?* PITAKA. https://www.ipitaka.com/blogs/news/has-technology-really-changed-our-lives-for-the-better
- Sharma, V. K. (2022, June 16). *Top 12 Examples, How Technology Has Changed Our Lives.* KLIENT SOLUTECH. https://www.klientsolutech.com/examples-of-how-technology-has-changed-our-lives/
- *HTTP Status 429 â€ " Too Many Requests*. (n.d.). World Bank Group. https://openknowledge.worldbank.org/handle/10986/34510
- GeeksforGeeks. (2022, May 2). *Unified Modeling Language (UML) | Activity Diagrams*. https://www.geeksforgeeks.org/unified-modeling-language-uml-activity-diagrams/#:%7E:text=An%20activity%20diagram%20is%20a,the%20activity%20is%20being%20executed.
- ŠKODA. (2021, May 10). *Der ŠKODA Hispano-Suiza* [Der ŠKODA Hispano-Suiza]. Der ŠKODA Hispano-Suiza. https://www.skoda-storyboard.com/de/210510_in-the-service-of-president-tgm-4/
- Tim Weilkiens, “Systems Engineering with SysML/UML”https://learning.oreilly.com/library/view/systems-engineering-with/9780123742742/

- -Kocic J, Jevicie N, Drndarevic V. An End-to-End Deep Neural Network for Autonomous Driving Designed for Embedded Automotive Platforms. Sensors, 2019 Jan: 19(9):2064. https://www.researchgate.net

CONTRIBUTION OF THE TEAM MEMBERS IN THE PAPER AND PROTOTYPE

- Abstract: Md Salman Bin Shahid
- Motivation: Shihab Ud Doula
- Requirements: Shihab Ud Doula & Emirkan Sali
- Sequence Diagram: Shihab Ud Doula
- Block Diagram: Ukamaka Akumeli
- State Chart Diagarm: Emirkan Sali
- Design: Shihab Ud Doula & Md. Salman Bin Shahid
- Final Model : Md Salamn Bin Shahid
- Implementation: Emirkan Sali
- Code: Emirkan Sali
- Summary: Shihab Ud Doula

Screent shot for code

```
// Code written by Emirkan Sali

#define s0 A0        //pins for RGB Sensor
#define s1 A1
#define s2 A2
#define s3 A3
#define out A4

#define echo 13    //pins for Ultrasonic Sensor
#define trig 12

#define motor1pin1 9   //pins for Motor driver. Enable Pins are always on
#define motor1pin2 8
#define motor2pin1 7
#define motor2pin2 6

#define sensor1 2   //pins for IR sensors
#define sensor2 4

long duration;  //variables for Ultrasonic Sensor
int distance;

int Red=0, Blue=0, Green=0;  //RGB values
int onGreen=0, onBlue=0, onMagenta=0, onOrange=0, onBlack=0, onWhite=0; //Values for if the veh


int vertical=1, horizontal=1; //set initial position of Vehicle

int back=0, forward=0; // variable for turning back. unused

int turnBackminus= 0, turnBackplus = 0;

int process = 0; //unused

int way = 1; //change to 2 when the vehicle should only go to one location

// Please put in the location of the object that will change
int horizonGoal = 4; // possible inputs: 0-13
int verticalGoal = 3; // possible inputs: 0-5
// Please put in the location of the object that will change

// Please put in the location of the object that is set permanently
int horizonGoalset = 4; // possible inputs: 0-13
int verticalGoalset = 3; // possible inputs: 0-5
// Please put in the location of the object that is set permanently

int sechoriGoalset = 1, secvertiGoalset = 5;

int sechoriGoal = 1, secvertiGoal = 5;

int facingfrontback = 0; // variable to indicate which direction the vehicle is facing.

void loop()
{
  case StopState:

  Stop();
  Serial.println("Stop State");
  getColors();
  displayColors();
  UltrasonicSensor();

if ((horizon == 2 || onAgenta == 2 || onOne == 2 || onOrange == 2) && !digitalRead(sensor1)== HIGH && !digitalRead(sensor2) == HIGH && distance > 25) && horizonval < 6 || verticalGoal > 0)
{
  Serial.println("Creating navigation");

  state = nextState;

}
break;

  case HoriState:

  Serial.println("Horizontal State");
  Serial.println("HorizonGoal = ");
  Serial.print(horizonGoal);
  GetColors();

  UltrasonicSensor();
  drive();
  if((digitalRead(sensor1)== LOW) && (digitalRead(sensor2) == LOW))
  {
    if (horizonGoal > 1 )
    {
      delay(400);
    }
      horizonGoal = --horizonGoal;
      if (horizonGoal < 0)
      {
        horizonGoal == 0;
      }
    }

  Serial.println("verticalGoal = ");
  Serial.print(verticalGoal);


   if( (horizonGoal == 0 && verticalGoal < 0) && (digitalRead(sensor1)== LOW) && (digitalRead(sensor2) == LOW))
   {
    Stop();
    state = Turnright;
   }
else if( way == 2 && horizonGoal == 0 && verticalGoal == 0)
{
  drive();
  delay(300);
  Stop();
  state = StopState;
}
  else if(horizonGoal == 0 && verticalGoalset == 1)
  {
  drive();
  delay(300);
  state = DetectState;
  }
  else if(horizonGoal == 0 && (digitalRead(sensor1)== LOW) && (digitalRead(sensor2) == LOW))
    {
      Stop();
      state = Turnleft;

    }
```

Screent shot for code

```
break;

case Turnleft:

Serial.println("turn left State");

delay(1000);
GetColors();
driveStraight();
delay(500);

turnLeft();
Stop();
delay(1000);


 if(startposHorizon == 1 && verticalGoal == 0)
 {
 facingfrontback = 1;
 state = specialTurnBack;
 }
else if((horizonGoal == 0 && verticalGoal == 0) && way == 2)
 {
  drive();
  delay(300);
  Stop();
  state = StopState;
 }

else if(horizonGoal == 0 && (sechoriGoal != 0) )
 {
  state = VertiState;
 }
else if(verticalGoal < 0)
 {
  state = BackVerti;
 }


break;

case VertiState:

Serial.println("Vertical State");
GetColors();
UltrasonicSensor();
drive();

 if((digitalRead(sensor1)== LOW) && (digitalRead(sensor2) == LOW) && onMagenta < 2)
{
 if (verticalGoal > 1 )
{
 delay(400);
}
 verticalGoal = --verticalGoal;
 if (verticalGoal < 0)
 {
   verticalGoal == 0;
 }
}

Serial.println(verticalGoal);

if((verticalGoal == 0 && sechoriGoal == 0) || (verticalGoal == 0 && startposHorizon == 1))
{
 Stop();
 state = Turnleft;
}
else if(verticalGoal == 0)
{
 Stop();
 state = Turnright;
}


break;

int startposHorizon = 0;

int earlyFinishverti = verticalGoalset - secvertiGoalset; //early finish when 2nd destination is right above or below 1st destination
int earlyFinishhori = horizonGoalset - sechoriGoalset;

int state = 1 ; //start state

void setup() {
 Serial.begin(9600);   //initialize the serial monitor baud rate

 pinMode(9, OUTPUT);
 pinMode(8, OUTPUT);
 pinMode(7, OUTPUT);
 pinMode(6, OUTPUT);

  pinMode(trig, OUTPUT);
  pinMode(echo, INPUT);
 Serial.println("Ultrasonic Sensor Distance");

  pinMode(4, INPUT);
  pinMode(2, INPUT);

  pinMode(s0,OUTPUT);    //pin modes
  pinMode(s1,OUTPUT);
  pinMode(s2,OUTPUT);
  pinMode(s3,OUTPUT);
  pinMode(out,INPUT);

  pinMode(5, OUTPUT);
  pinMode(10, OUTPUT);

  digitalWrite(s0,HIGH); //Putting S0/S1 on HIGH/HIGH levels means the output frequency scalling is at 100% (recommended)
  digitalWrite(s1,LOW); //LOW/LOW is off HIGH/LOW is 20% and LOW/HIGH is  2%

  horizonGoal = horizonGoal - horizontal;
  verticalGoal = verticalGoal - vertical;

  if(horizonGoal == 0)
  {
   horizonGoal = 1;
   startposHorizon = 1;
  }

  Serial.println("early finish =");
  Serial.print(earlyFinishverti);
}

enum state {StopState = 0,HoriState = 1, VertiState = 2, Turnleft = 3, Turnright = 4, DetectState = 5, TurnBack = 6, BackHori = 7, BackVerti = 8, Finish = 9, specialTurnBack = 10 };
void loop() {
switch (state)
```

```
case DetectState:

Serial.println("Detect State");
GetColors();
UltrasonicSensor();
Stop();

if(sechoriGoalset <= horizonGoalset)  {
  state = TurnBack;
}
else state = Finish;

break;

case TurnBack:

Serial.println("turn back state");

drive();

if((digitalRead(sensor1)== LOW) && (digitalRead(sensor2) == LOW) && (secvertiGoalset != verticalGoalset) && (turnBackminus < 2)) && verticalGoalset != 5) // turn right side back
{
 Stop();
 delay(1000);
 GetColors();
 driveStraight();
 delay(500);

 turnRight();
 Stop();
 delay(1000);

 turnBackminus = ++turnBackminus;
 Serial.println("turn back state");
}

if((digitalRead(sensor1)== LOW) && (digitalRead(sensor2) == LOW) && (secvertiGoalset >= verticalGoalset) && turnBackplus < 2) && verticalGoalset != 5)  // turn left side back
{
 Stop();
delay(1000);
GetColors();
driveStraight();
delay(500);

turnLeft();
Stop();
delay(1000);
turnBackplus = ++turnBackplus;
Serial.println("turn back state");
}

if(turnBackminus == 2)
{
  if(facingfrontback == 0)
  {
    facingfrontback = 1;
  }
  else if(facingfrontback == 1)
  {
    facingfrontback = 0;
  }
  vertical = verticalGoalset;
  vertical = --vertical;
  horizontal = horizonGoalset;
  verticalGoalset= --verticalGoalset;
  state = Finish;
}

if(turnBackplus == 2)
{
  if(facingfrontback == 0)
  {
    facingfrontback = 1;
  }
  else if(facingfrontback == 1)
  {
    facingfrontback = 0;
  }
  vertical = verticalGoalset;
  vertical = ++vertical;
  horizontal = horizonGoalset;
  verticalGoalset= ++verticalGoalset;
  state = Finish;
}

break;
```

```
case Turnright:

Serial.println("turn right State");

delay(1000);
GetColors();
driveStraight();
delay(500);

turnRight();
Stop();
delay(1000);

Serial.println("verticalGoal = ");
Serial.print(verticalGoal);
Serial.println("secvertiGoal = ");
Serial.print(secvertiGoal);
Serial.println("sechoriGoal = ");
Serial.print(sechoriGoal);

if(((secvertiGoal == 0 && sechoriGoal == 0) || (horizonGoal == 0 && verticalGoal == 0)) && way == 2)
{
  drive();
  delay(300);
  Stop();
  state = StopState;
}
else if (verticalGoal < 0 && horizonGoal == 0)
{
  state = BackVerti;
}
else if (verticalGoal > 0)
{
  state = VertiState;
}
else if(verticalGoal == 0)
{
  state = DetectState;
}
```

```
if(startposHorizon == 1)
{
  horizonGoal = --horizonGoal;
  startposHorizon = 0;
}

Serial.println(sechoriGoal);
Serial.println("horizongoal");
Serial.print(horizonGoal);
if(sechoriGoal < 0)
{
  state = BackHori;
}
else
{
  state = HoriState;
  horizonGoal = ++horizonGoal;
  facingfrontback = 1;
}

break;
```

```
case BackHori:

Serial.println("Back Horizontal State");
GetColors();

UltrasonicSensor();
drive();
if((digitalRead(sensor1)== LOW) && (digitalRead(sensor2) == LOW))
{

  if (horizonGoal < -1  )
{
  delay(500);
}
  horizonGoal = ++horizonGoal;
  Serial.println("minus 1 horizontal");
}

Serial.println(horizonGoal);

if(horizonGoal >= 0)
{
  Stop();
  sechoriGoal = 0;
  if(secvertiGoalset > verticalGoalset)
  {
  state = Turnright;
  }
  else if(secvertiGoalset < verticalGoalset)
  {
    state = Turnleft;
  }
  else
  {
    state = StopState;
  }
}

break;
```

```
case Finish:

Serial.println("Finish State");

Stop();

Serial.println(earlyFinishverti);
Serial.println(earlyFinishhori);

while(earlyFinishverti == 1 && earlyFinishhori == 0 )
{
  Serial.println("Earlyfinish");
  Stop();
  state = StopState;
}

while(earlyFinishverti == -1 && earlyFinishhori == 0)
{
  Serial.println("Earlyfinish");
  Stop();
  state = StopState;
}


horizontal = horizonGoalset;
vertical = verticalGoalset;

sechoriGoal = sechoriGoal-horizonGoalset;
secvertiGoal = secvertiGoal-verticalGoalset;

horizonGoal = sechoriGoal;
verticalGoal = secvertiGoal;

way = ++way;

if(way > 2)
{
  way = 2;
}
```

```
case specialTurnBack: // only if the starting horizontal position is the same as the destination or the horizontal coordinate

Serial.println("special turn back state");

drive();

if((digitalRead(sensor1)== LOW) && (digitalRead(sensor2) == LOW) && (secvertiGoalset == verticalGoalset) && (turnBackvalue < 2)) // turn right side back
{
  Stop();
  delay(1000);
  GetColors();
  driveStraight();
  delay(500);

  turnLeft();
  Stop();
  delay(1000);

  turnBackvalue = ++turnBackvalue;
  Serial.println("turn back state");
}

if((digitalRead(sensor1)== LOW) && (digitalRead(sensor2) == LOW) && (secvertiGoalset > verticalGoalset) && turnBackvalue < 2)  // turn left side back
{
  Stop();
delay(1000);
GetColors();
driveStraight();
delay(500);

turnRight();
Stop();
delay(1000);
turnBackvalue = --turnBackvalue;
Serial.println("turn back state");
}
```

```
 if(turnBackminus == 2)
 {
   facingfrontback = 0;
   vertical = verticalGoalset;
   vertical = --vertical;
   horizontal = horizonGoalset;
   verticalGoalset= --verticalGoalset;
   state = Finish;
 }

 if(turnBackplus == 2)
 {
   facingfrontback = 0;
   vertical = verticalGoalset;
   vertical = ++vertical;
   horizontal = horizonGoalset;
   verticalGoalset= ++verticalGoalset;
   state = Finish;
 }

  break;

case BackVerti:

Serial.println("Back Vertical State");
Serial.println("verticalGoal=");
Serial.print(verticalGoal);

GetColors();
UltrasonicSensor();
drive();

 if((digitalRead(sensor1)== LOW) && (digitalRead(sensor2) == LOW) && onMagenta < 2)
{
  if (verticalGoal < -1 )
{
  delay(400);
}
  verticalGoal = ++verticalGoal;
  if (verticalGoal > 0)
  {
    verticalGoal == 0;
  }
  Serial.println("minus 1 verticalGoal");
  Serial.println(verticalGoal);
}

if(verticalGoal == 0)
{
  Stop();
  secvertiGoal = 0;
  sechoriGoal = 0;
  state = Turnright;
}

break;
```

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.

Sali, Emirkan	19/06/2022	
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.

Bin Shahid, Md Salman	Hamm, 18.06.2022	
Name, Vorname Last Name, First Name	Ort, Datum Location, Date	Unterschrift 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.

Doula,Shihab Ud	Lippstadt, 19.06.22	Shihab Ud Dowla
Name, Vorname Last Name, First Name	Ort, Datum Location, Date	Unterschrift 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.

Anaedu Ukamaka A	Nigeria 16/06/2022	[signature]
Name, Vorname Last Name, First Name	Ort, Datum Location, Date	Unterschrift Signature