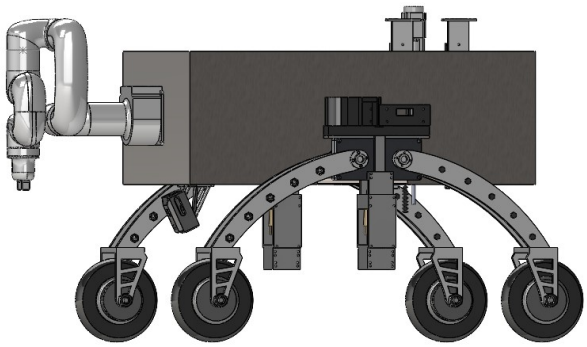
Implementation of a harvesting system in an autonomous agricultural vehicle.

Design and implementation of mechatronic systems

Alfredo Ramirez Morales A01367820@tec.mx
Diego Mondragón Wong A01368238@tec.mx
Arturo Martínez Zambrano A01367486@tec.mx
Johann A. Sandoval López A01025619@tec.mx
José Luis Álvarez Marín A01367514@tec.mx
José Guillermo Cruz García A01366770@tec.mx



ABSTRACT

Summary. The project focuses on the development of an autonomous system for harvesting radishes, combining a robotic arm and a computer vision system. The system identifies and locates ripe radishes using cameras and vision algorithms, determining their exact position for UFactory's Lite6 robotic arm to pick them without damaging them. The arm is mounted on the front of an autonomous vehicle (AGV) chassis, while the cameras and Raspberry Pi are installed in a protective case tilted at 45°, optimizing visual coverage. The system has proven to be technically feasible and efficient, enabling accurate and autonomous radish harvesting, improving agricultural productivity and reducing human intervention.

1 KEY WORDS

Autonomous Harvesting - Robotic Arm - Robotic Arm Computer Vision - Raspberry Pi - Automated Agriculture -

2 RELATED WORK

This project sits at the intersection of robotics and precision agriculture, areas that have seen significant advances in recent years. Similar projects using computer vision for crop identification and harvesting are found in the literature, such as the tomato harvesting system developed by Bartoli et al. (2023) [1] and the innovative agent-modeled robot Garcia et al. (2024) [3].

However, most of these projects face challenges related to accuracy in crop identification and manipulation, taking reference from the [2] strawberry planter and sorter. Our project addresses these challenges by integrating advanced cameras and optimized vision algorithms, as well as implementing a robotic arm with high repeatability accuracy. In addition, the installation of the components on an AGV provides mobility and autonomy, which is crucial for operations in open fields, summarizing in an innovative proposal that maximizes visual coverage and accuracy in radish harvesting, representing a significant advance in agricultural automation.

IMT '24, April 2024-June 2024, Toluca, Mexico
2024.

3 INTRODUCTION

Objective. The project addresses the need to automate radish harvesting, a task currently performed manually, which is error-prone and labor-intensive, affecting crop quality and demanding a lot of manpower. To overcome this challenge, an autonomous system will be developed that integrates a robotic arm with computer vision technology. This system will employ advanced cameras and algorithms to identify ripe radishes, determining their exact position for precise harvesting. The robotic arm, mounted on an autonomous vehicle (AGV) and optimized for its center of mass, will move precisely to harvest radishes without damaging them. Cameras and a Raspberry Pi will be installed in a joint case on the AGV chassis at a 45-degree angle, maximizing visual coverage. The system will be programmed to coordinate the identification, location and harvesting of the radishes and will be field tested to ensure its efficiency and accuracy.

This development will enable efficient and accurate farming operations while minimizing human intervention. The impact on the agricultural industry will be significant, as it will increase productivity, improve harvesting accuracy, foster technological innovation in agriculture, reduce long-term operating costs, and promote more sustainable and efficient farming practices. In short, this autonomous radish harvesting system will not only optimize efficiency and accuracy in agriculture, but will also contribute to the modernization and sustainability of the agricultural sector.

4 METHODOLOGY: VISUAL ROBOTIC SYSTEM

General System Description. The objective of the project is to develop an autonomous system for harvesting radishes, integrating a robotic arm and a computer vision system. The system identifies and locates ripe radishes using cameras and vision algorithms, determining their exact position so that the robotic arm can pick them without damaging them. UFactory's Lite6 robotic arm is mounted

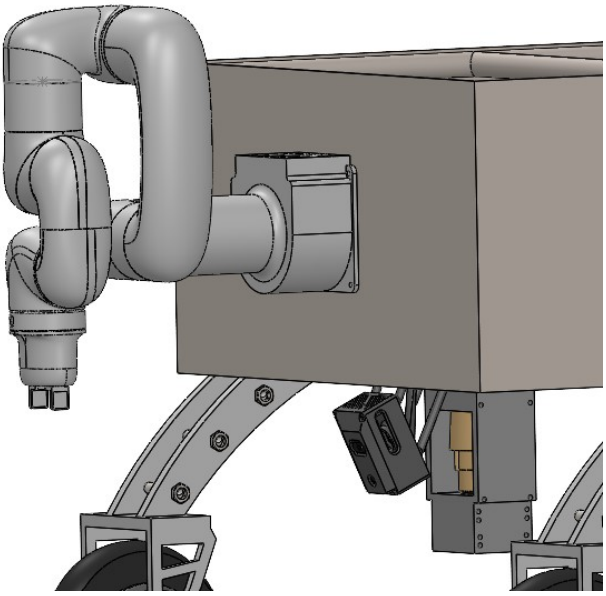


Figure 1: Subsystems Installation

horizontally on the front of an autonomous vehicle (AGV) chassis, while the cameras and Raspberry Pi are installed in a protective case tilted at 45°, maximizing visual coverage.

4.1 Simulations and Proof of Concept

The simulations support us to have a proof of concept about the operation of such a system.

5 INSTALLATION AND SCHEMATIC

Subsystem Installation. The robotic arm is installed on the front of the AGV chassis, adjusting the center of mass for precise movements. The vision system is integrated into a protective case mounted on the bottom-front of the chassis, tilted at 45°, allowing wide visual coverage for object recognition and tracking. This design ensures optimal performance in handling and navigation tasks.1.

3D Visualization and Specifications. The system consists of the Lite6 robotic arm and the 8GB Raspberry Pi 5 with interconnected cameras. The 3D visualization shows the base of the robotic arm horizontally and the cameras pointing 45° downward. This configuration ensures an optimal view of the operating area, facilitating image capture and real-time processing for arm navigation and manipulation.2.

5.1 Vision System

The vision of the system is a fundamental component for the operation of the radish collection, it was mentioned earlier that this was thanks to a Raspberry Pi 5 with two camera modules, being a conventional RGB camera and another camera with a time-of-flight sensor that is used to determine the distance. Both cameras work sequentially for the detection and location of the designated objects, as making them work together proves to be very demanding on the available resources that the Raspberry has.

Cameras. The RGB camera used for the vision system was an 11.9 MP PiCamera Module 3 with a Sony IMX708 sensor, being a sensor used for cell phones and resulting in excellent quality images with features such as autofocus and autoexposure so that the sensor adapts according to the ambient light conditions. This is connected through

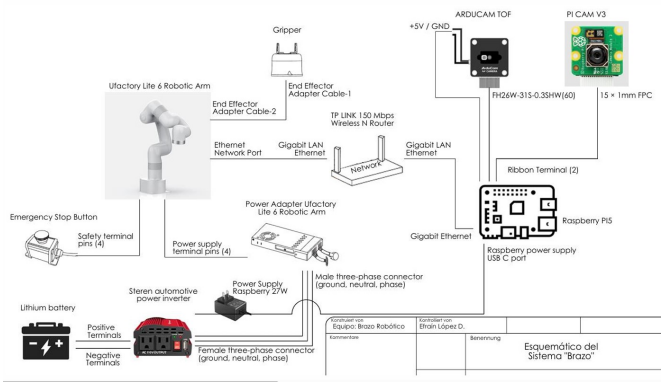


Figure 2: Robotic arm system schematic diagram

a CSI interface to the camera/display connectors that are integrated directly on the Raspberry motherboard, and this requires an adapter that Raspberry's own distributors sell.

On the other hand, the time-of-flight camera used in the system was the Arducam ToF Camera with a 0.43 MP sensor (240*180), coming from Arducam which is a company that makes accessories for the Raspberry and that kind of computers (Nvidia Jetson, Rock Pi, among others) that use this same CSI interface. The camera has a maximum distance range of 4 meters, although for the application we are using, we prefer to set it to 2 meters to obtain greater accuracy in the measurements provided by the lasers, which anyway have some error due to surfaces, angles and the Python API of the camera itself. It uses the same interface as the RGB camera and was one of the main reasons why we chose the Pi 5, since this device has two connectors for cameras and the use of both cameras implied using two connectors for proper operation of the entire vision module.

Functionality. In the vision system, the RGB camera is used for object and color detection by using the MediaPipe library, OpenCV as well as the Picamera2 library which is the one used for the official Raspberry cameras both to initialize and set the desired configuration. Because we have differences in aspect ratios with the PiCamera capturing at 16:9 while the time-of-flight camera records at 4:3, images of the same aspect ratios must be used because otherwise the centroid could be found outside the perspective of the time-of-flight camera and possibly give errors in the code since it is looking for a point that is not part of the matrix set by this camera.

This is of utmost importance because the point obtained in the camera must coincide with the point set in the time-of-flight camera because this is only to determine the distance at the point specified by the centroid found by the color detection by the figure found. The average distance is determined and with this is that the whole question of distance at the points x, z which are where our robot moves is performed. This is responsible for determining the distance that the robot should move and to which point it should be directed to achieve the optimal route.

6 ARTIFICIAL INTELLIGENCE - SSD MOBILENET MULTI-HW

6.1 Neural network architecture

Neural network. MobileNetV2 MultiHW is a convolutional neural network architecture optimized for use on a variety of devices with different hardware capabilities. The MobileNetV2 architecture is composed of inverted residual blocks with point and depth convolutional layers, known as inverted residuals and depthwise separable convolutions. The inverted residual blocks facilitate the learning of

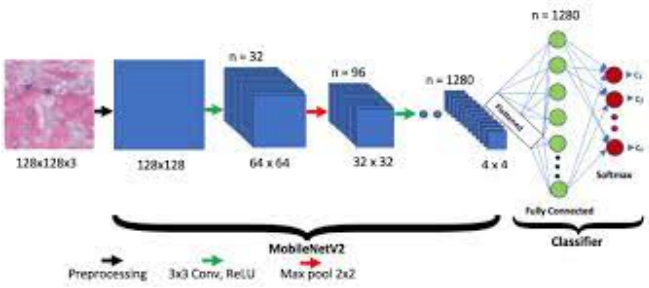


Figure 3: Neuronal network - SSD MobileNet Multi-HW

Figure 4: Radish Identification

complex spatial features while maintaining computational efficiency. In addition, shortcut connections within these blocks allow for better gradient propagation during the training process, which improves model performance.

In our project, we have implemented MobileNetV2 MultiHW for the specific task of radish recognition. MobileNetV2's ability to run highly accurate computer vision models in real time is crucial for automated applications, where fast and accurate image processing is essential. We used MediaPipe to optimize image processing and improve model efficiency on the Raspberry Pi 5 we used, the platform on which we ran the application. MediaPipe provided efficient tools and frameworks for data processing, which enabled a smoother and faster integration with MobileNetV2. 3.

The network was trained to identify and locate radishes in images, using a specific dataset that includes various lighting conditions and capture angles. This implementation has proven to be effective in automated radish identification and harvesting, contributing to the efficiency and productivity of farming operations. MobileNetV2 MultiHW's optimized architecture, coupled with MediaPipe's capabilities, ensures that the model can be deployed on resource-constrained hardware, such as the Raspberry Pi 5. 4.

Figure 5: Model Comparison: Mojalvin vs. Printed Sheet

6.2 Traning the model

Training Process. Two models were trained using the MediaPipe Model Maker library created by Google following all the aforementioned documentation and making small modifications to the hyperparameters of the model to make it more accurate without overtraining it. We used a learning rate of 0.45, a batch size of 8 (referring to the number of images it reviews at a time) and 75 or 90 epochs, which is the number of steps the model will have and this is not necessarily better, because it can over-train and this model may not work well with other images that are not the ones used for training. These two models were trained for the detection of radishes (being this our real use) and for the detection of a small box as "proof of concept" where it is given the name of "Mojalvin", the reason for this object is because we were looking for something that the robot could easily embrace with the limited strength we had. 5.

Model Comparison. The two models were trained to detect completely different objects, but both achieve an object accuracy above 70 percent even with blurred or out-of-focus images and even by changing the video recording hardware, using a webcam with lower quality than the Pi Camera offers.

Training the model took approximately 40 minutes for "Mojalvin" and about an hour for radish detection due to the difference in epochs, and was performed under this time with the following specifications: CPU: AMD Ryzen 9 5950X - 16 cores and 32 threads. RAM: 32 GB DDR4. GPU: Nvidia GeForce RTX 3090 with 24 GB of GDDR6X graphics memory modified to achieve 1 TB/s memory bandwidth.

The estimated power consumption was 400W during the training of both models over time. Similarly, the Windows Subsystem for Linux (WSL2) was used because the version of Tensorflow with Keras (2.15.1) used does not support hardware acceleration with Nvidia's CUDA with the Windows OS, and this allows considerably accelerating the training speed as opposed to doing it only with the system's CPU.

6.3 Performance of the model deployed on low resources

Efficiency comparison. We mentioned earlier that we use the MobileNet Multi-HW architecture, being because this model has an additional layer during training that is able to determine the type of hardware on which it is running and thus adapt the types of calculations for this to work in the most optimal way possible within the system that is found, in this case as we use a Raspberry, we had a performance problem due to the fact that the hardware used inside these small computers is very low power consumption, obtaining object inference speeds of approximately 150 ms (approximately 6 frames per second) and due to this is that the box generated over the image has quite a delay when following it, because the camera frame rate are decoupled to the frame rate of the model inference.

In an attempt to mitigate this problem, we made some quantization adjustments to reduce the size of the models from 32-bit floating point operations to 16-bit operations or 8-bit integer operations for the best performance but worst model accuracy. This is why it was decided to use the 16-bit model since it offered a balance between performance and model accuracy.

7 ROBOTIC SYSTEM

Robotic System Criteria. The movement of the arm was elaborated by using the Python programming language using the U-Factory development kit for the Lite6 robotic arm, based on the kinematic development implemented by Yanza. [4]. This kit allows the movement of the robotic arm using absolute coordinates as well as the movement of each link, for the problem presented it was decided to solve by means of absolute coordinates. Unfortunately the capabilities of the arm are limited due to the solutions created by the kit control creating singularities or collisions in situations where an arm with better programming can move freely.

Coordinate calculation.

$$ObjectAngle = (41(2)(1080))(ObjectPosition) - 412 + 25 \quad (1)$$

The camera angle is 41°, this equation is based on the formula of a slope y=mx+b, where m is the angle of the object with respect to the camera, x is the position (in pixels) in which the object is located and b is the displacement, which is half of the camera viewing angle added with the angle in which the camera is located, in this situation it is at 25°. With these values the displacement of the arm can be obtained by means of the following formulas:

$$ZMovement = (ObjectDistance)(sin(90 - ObjectAngle)) \quad (2)$$

$$XMovement = (ObjectDistance)(sin(ObjectAngle)) \quad (3)$$

Having the values of the 90° angle, the angle of the object and its distance, it is possible to calculate the required displacement of the arm to approach the object. It is also necessary to consider that the camera position is at a fixed distance from the arm, so the offset distances must be compensated.

Arm Movements. Due to the limited access to the robotic arm development library, it is not possible to move the arm to the desired coordinates with only one instruction, so it is recommended to initially move the Z axis and then move the X axis, avoiding collisions with the arm itself and singularities that block the movement of the arm.

It is of utmost importance to consider the position of the end effector and the velocity of the arm. In the presented solution a speed of 30 percent has been estimated avoiding chassis unbalance. For the end effector it has been instructed to keep the end effector pointing in the direction of the floor avoiding unnecessary movements as well as collisions (either the floor or the arm itself).

It is of utmost importance to consider the position of the end effector and the velocity of the arm. In the presented solution a speed of 30 percent has been estimated to avoid chassis unbalance. For the end effector it has been instructed to keep the end effector pointing in the direction of the floor avoiding unnecessary movements as well as collisions (either the floor or the arm itself).

8 SYSTEM SIMULATIONS

General information about the simulations. In the section of the simulations it was decided to perform two, the first was done in RoboDK software and shows mainly the operation of the arm subsystem, more details will be given below, the second simulation was done in Matlab Simulink starting from the CAD design in Solidworks, which shows the operation of the entire complete system, from the movement thanks to the navigation subsystem, as well as the process of planting new radish seeds using the actuator and finally the collection of these same with the use of the robotic arm.

Figure 6: CAD Design Integrated Systems

8.1 Simulation of the Robotic Arm System

As already mentioned, the simulation was carried out with the use of the robodk application by means of which it is sought to exemplify as a digital twin the operation of the process of recognition and location of the "radish" through image processing by computer vision, in which it can be identified how the code used isolates the location of the object, and by means of trigonometric functions the Cartesian coordinates of the object are determined. This is done by taking as reference or origin in the Cartesian plane the lens of the camera used, from which a conversion of the object coordinates is generated, with reference to the central point of the effector (tcp).

Subsequently activate the robotic arm and simulate a routine where it collects the object and deposits it in a container. It is worth mentioning that the arm shown in the simulation does not represent the exact model used for the physical prototype, this due to limitations beyond the team's control regarding the proprietary library of the arm used, instead an arm with the same number of degrees of freedom and a structure of the links similar to that found in the physical model is shown, finally in the following image you can visually see how the simulation looks inside the application.

The corresponding simulation is shared in the following Youtube link, where the sequence of operation of the robotic arm system is shown.

LINK: Robotic Arm Simulation.

8.2 Lunar Rover Simulation with Systems Integration

Lunar Rover Simulation. For the simulation of the entire Lunar Rover complex, the complete assembly of all the subsystems that make up the project; robotic arm, seeding actuator, locomotion and suspension, was performed first. Within the solidworks software the CAD simplification of each subsystem was performed in order to keep active only the position relations necessary for the simulation of the prototype, being these the revolves of the robotic arm, the rotational joints of the suspension, as well as the tires, and the prismatic joint of the actuator. 6.

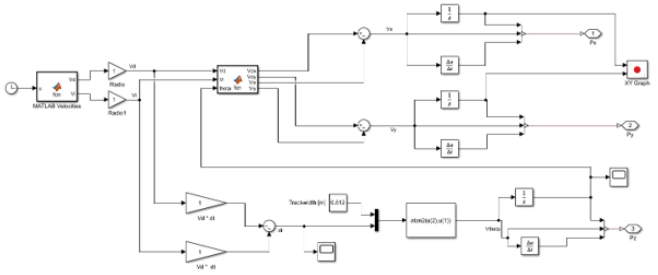


Figure 7: Simulink Simscape - Speed Control

Simulink-Simscape. Having the subsystems designed and the links ready for the simulation, we proceeded to use the "Simscape Multibody Link Plugin in SolidWorks" tool of the Matlab software. With this Add-On the translation of CAD files - Solidworks, to Block Diagram - Simscape/Simulink Matlab was performed.

For the generation of the simulation a speed control of a dynamic model for the navigation of the whole system was performed, using as the moving reference frame the coordinate system of the chassis. This reference system is based on a closed-loop control where the navigation angle of the vehicle is fed back in each iteration, starting from the velocities at each instant of time of the right and left side wheels. 7.

This control was used to perform a route proposed by navigation in order to demonstrate all the capabilities of the prototype, being these the linear forward and backward movement, rotation with variable turning radius and rotation on the same axis. Adding to this behavior the lunar rover performs the lifting and lowering movements of the suspension, lifting and lowering of the seeding actuator system and the movements of each link of the robotic arm.

The complete simulation is shared in the following Youtube link, where the integration of each subsystem of the Lunar Rover is shown.

LINK: Lunar Rover farmer simulation.

8.3 Static simulation for critical cases

Static Simulation. The analysis focused on validating the decision to place the robotic arm horizontally at the front of the lunar rover, evaluating the circumstances under which this configuration could generate complications in the system. A beam analysis was conducted considering the critical scenario of the arm fully extended while the vehicle is static. A beam with the chassis length supported at its midpoint was used, with the objective of minimizing the moment generated in the chassis axis due to the static loads of the installed components, where a non-significant torque is obtained during the static operation of the vehicle.8.

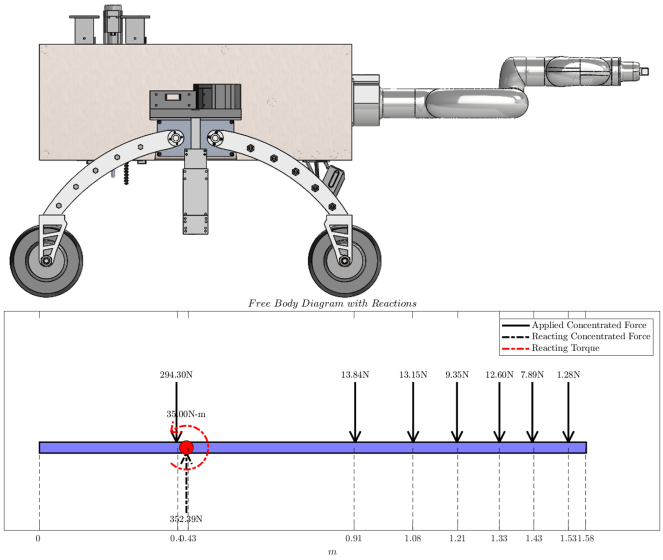


Figure 8: Beam Analysis - Matlab

CODE AND SYSTEM LOGIC

(1) Artificial Intelligence Vision

1.-. The "MediaPipe" library with the "MobileNet Multi-HW" architecture has been used for radish recognition. Within the code it searches for an object (either box or radish) with a hit percentage greater than 60 percent. This creates a green box showing the position of the object as well as the hit percentage. Once the object is found, the detection value is saved. If more than 15 values are saved, it ensures that it is the correct object and not a vision error. If nothing is found for 30 seconds the code is interrupted and a signal is given to the communication subsystem of the detection status.

(2) Color detection by OpenCV

2.-. Once the AI has confirmed the detection of a radish the OpenCV color detection process is initiated. In this a range of colors is identified, in the case of looking for radishes it is recommended to use the HSV range of 168, 148, 99 to 173, 209, 153. For practicality a wide range of reds was used for detection of the box, or in this case what would be a visually similar color to the radishes. It is possible that more than one red object is detected, these small objects are called "blobs" within the code has been implemented a restriction in which identifies the blob of greater size ignoring the rest.

(3) Position detection by OpenCV

3.-. Following this, a frame of the largest identified color object is created and the estimation of the center of the frame is obtained by saving the value, then 5 values of the center coordinate are saved to avoid detection errors due to light conditions and averaged.

(4) Distance approximation by TOF

4.-. The time-of-flight camera is approximately 3 cm away from the color camera, so it is required to implement a displacement of the center found. This was done experimentally since in addition to the 3 cm offset one can have angles not considered in the assembly.

The resolutions of the cameras are different, while the PiCam has a resolution of 1440 x 1080 and the TOF of 240 x 180, so it is necessary to perform a conversion of the center by simply dividing by 6. The time-of-flight camera identifies the distance of the indicated point for a period of time saving the values and performing an average.

(5) Arm movement

5.-. Once the distance is obtained, calculations are made for the displacement of the arm to approach the identified object, hold it and return to the assigned "home" position, saving the values of the position to which it has been directed for later sending it to the communication system.

(6) Communication

6.-. The communication system has provided us with a code where it is only necessary to implement the code of a system and the variables that are sent and received.

The arm system has only 1 variable received, the navigation state (In motion or stopped) and 2 variables are sent, the permissive (In motion or stopped) and the position of the arm. Once the navigation state data is received, if this system is stopped the arm system sends the permissive and keeps navigation stopped, our code is executed and in case of finding the object it continues sending the permissive until it collects it, finally the "stopped" state is sent and the position to which the arm has been directed. The communication code works through 3 threads, the raspberry is able to send, receive and run the code of the collecting arm simultaneously, so it does not have unwanted delays such as arduino or ESP cases.

9 PROJECT RESULTS

The project has successfully made a mobile robot detect and pick radishes using computer vision. Using color detection, the system can identify radishes and mark them with a green square. Subsequently, an AI confirms the detection and activates a depth camera to identify the radish's centroid and calculate its exact coordinates. This process allows the robotic arm to accurately approach and pick up the radish.

9.1 Technical Feasibility

The developed system has proven to be technically feasible. The integration of cameras and a Raspberry Pi, together with advanced computer vision algorithms and a robotic arm optimized for the AGV's center of mass, has enabled accurate and efficient operation. The results confirm that the robot can identify and harvest radishes autonomously, validating the technical feasibility of the project.

9.2 Coverage of the Specified Scope

The results obtained fully cover the specified scope. The autonomous system can accurately identify, locate and harvest ripe radishes, minimizing human intervention and improving agricultural productivity, meeting the project objectives.

9.3 Project Innovation

This innovative project integrates advanced robotics and computer vision technologies to automate radish harvesting, optimizing the process and advancing agricultural automation. Technical feasibility has been demonstrated in tests, using robust technologies. Economically, it reduces long-term labor costs. Operationally, it can operate autonomously in the field with minimal human intervention, facilitating its implementation and continuous operation.

The impact is positive, increasing harvesting efficiency and accuracy, reducing waste and operating costs, and promoting more sustainable and modern agriculture through the use of advanced technologies.

10 CONCLUSION

The project represents a significant challenge that has required effective communication and proper distribution of work. The experience has improved time management skills and demonstrated the importance of team collaboration for the success of complex projects. The integration of different mechanical, electrical and control systems has provided a deeper understanding of their interdependencies and the need for detailed planning.

11 BIBLIOGRAPHY

REFERENCES

[1] N. Bartoli and F. A. Conde. *Diseño de Brazo robótico articulado con capacidad de translación, para la recolección de tomate en la zona de "Coyunco".* PhD thesis, Universidad Nacional de Mar del Plata. Facultad de Ingeniería, Argentina, 2023.

[2] M. A. L. Bedón and C. C. P. Ramírez. Diseño y construcción de un sistema mecatrónico clasificador de fresas mediante visión por computador. *Nombre de la revista (si aplica)*, 2018.

[3] L. García Perez. Navegación autónoma de robots en agricultura: un modelo de agentes. *Nombre de la revista (si aplica)*, 2004.

[4] D. Yanza. *Departamento de Ciencias de la Energía y Mecánica Carrera de Mecatrónica Trabajo de titulación, previo a la obtención del título de Ingeniero en Mecatrónica.* PhD thesis, Universidad de las Fuerzas Armadas, 2023.

12 APPENDIX

LINK: Appendix for Documents.