



Universidad de Valladolid



**ESCUELA DE INGENIERÍAS
INDUSTRIALES**

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERIAS INDUSTRIALES

Grado en Ingeniería de tecnologías industriales ITI

**-GPS GUIDANCE SYSTEM FOR AUTOMATIC
DRIVING-**

Autor:

Arribas Molina, Jorge

Responsable de Intercambio en la Uva:

Eusebio de la Fuente

UCLL, University College Leuven-Limburg

Valladolid, Febrero 2023.

TFG REALIZADO EN PROGRAMA DE INTERCAMBIO

TÍTULO: GPS guidance system for automatic driving.
ALUMNO: Jorge Arribas Molina.
FECHA: 27 de enero de 2023.
CENTRO: Faculty of technology Campus Diepenbeek.
UNIVERSIDAD: UCLL, University College Leuven-Limburg.
TUTOR: Roel Conings.

ACKNOWLEDGEMENTS

I would like to convey my most sincere gratitude to all those who have helped me throughout this stage and have collaborated in this investigation.

First of all, to the UCLL and the University of Valladolid for making the Erasmus experience possible. To my project tutor, Roel Conings, for his help and guidance in this Final Degree Project. To César Castañón Ares and Geert for their help and support with the programming of the code. To Beatriz and Luis Arribas for their support during my stay in Belgium. And finally, to my parents and brother for everything.

INDEX

1.	ABSTRACT	1
2.	INTRODUCTION	2
2.1.	PROJECT GEPLAND FRUIT VEHICLE'S CONTEX.....	2
2.2.	ENERGY SYSTEM.....	2
2.2.1.	POWER GENERATORS.....	2
2.2.1.1.	GASOLINE ENGINE.....	2
2.2.1.2.	ORBITAL HYDRAULIC MOTOR.....	3
2.2.1.3.	BIO-GAS CONVERSION.....	4
2.2.1.4.	SOLAR PANEL.....	4
2.2.2.	ENERGY STORAGE SYSTEM.....	5
2.3.	KEY COMPONENTS.....	7
2.3.1.	GPS DEVICE.....	7
2.3.2.	CONTROL SYSTEM.....	8
2.3.3.	ANGLE SENSOR.....	9
2.3.4.	HYDRAULIC SYSTEM.....	11
2.3.4.1.	HYDRAULIC POWER PACK.....	11
2.3.4.2.	DOUBLE ACTING HYDRAULIC CYLINDER.....	12
3.	OBJECTIVE.....	12
4.	SOFTWARE AND METHODOLOGY	13
4.1.	SOFTWARE.....	13
4.1.1.	AgOpenGPS.....	13
4.1.1.1.	GPS-NTRIP CONNECTION.....	15
4.1.1.2.	VEHICLE CONFIGURATION.....	18
4.1.1.3.	RECORDED PATH AND AUTOSTEER CONFIGURATION.....	24
4.1.2.	TIA-Portal.....	27
4.2.	METHODOLOGY, EXPERIMENTAL PROCEDURE.....	28
4.2.1.	IMPLEMENTED AND MODIFIED CODE.....	28
4.2.1.1.	SHARP7.....	28
4.2.1.2.	CODE IMPLEMENTED IN AgOpenGPS.....	29
4.2.1.3.	CODE PROGRAMMED IN TIA PORTAL.....	36
4.2.2.	EXPERIMENTAL PROCEDURE.....	39

5.	CALCULATIONS AND MEASUREMENTS	43
5.1.	ROTATION ANGLE RANGE OF THE VEHICLE AXLE.....	43
5.2.	STEER ANGLE DETECTION.....	45
5.3.	TURNING RADIUS.....	50
6.	RESULTS.....	51
7.	FURTHER IMPROVEMENTS	52
8.	CONCLUSIONS	53
9.	REFERENCES	54

LIST OF FIGURES

Figure 1. "PTM760PRO" gasoline engine	2
Figure 2. Orbit hydraulic motor.	3
Figure 3. 115W solar panels.....	4
Figure 4. MPPT system.	5
Figure 5. Energy storage system diagram.....	6
Figure 6. Smart Battery Management System.	7
Figure 7. SimpleRTK2B V3 NTRIP- GPS device.	7
Figure 8. SIEMENS PLC.	8
Figure 9. KTC-300-P - Linear motion position sensor.	9
Figure 10. Hydraulic power pack.....	11
Figure 11. Double acting hydraulic cylinder.....	12
Figure 12. Project's system block diagram.....	13
Figure 13. Original AgOpenGPS project's components diagram.	14
Figure 14. GPS ports communication menu.....	15
Figure 15. Satellite-Antenna-GPS device communication diagram.	16
Figure 16. FLEPOS network on flanders.....	16
Figure 17. GPS error demonstration.	17
Figure 18. GPS Conection problems examples.	17
Figure 19. FLEPOS network's connection menu.....	18
Figure 20. Articulated tractor.	18
Figure 21. AgOpenGPS vehicle selection menu.	19
Figure 22. AGV.....	20
Figure 23. AGV measures.	21
Figure 24. Antenna position relative to the back axis.	22
Figure 25. Different GPS positioning.	22
Figure 26. GPS signal in three different cases.	23
Figure 27. Antenna configuration menu.	23
Figure 28. Double antenna.	24
Figure 29. Path selection menu.....	24
Figure 30. Pre-Recorded path arround UCLL technologie building.....	25
Figure 31. Dubins path to the beginng of the recorded path.	25
Figure 32. KML file from GoogleEarth to path.	26
Figure 33. Auto steer configuration menu.	26
Figure 34. TIA Portal interface.....	27
Figure 35. Sharp7 logo.	28

Figure 36. Writing the variable methods.....	29
Figure 37. Angle calculation function.....	30
Figure 38. (a) Connection to the PLC method.....	30
Figure 39. Calling of the connection method.....	31
Figure 40. (b) Sending the variable to the PLC.....	31
Figure 41. EscribirDatosPLC call inside PurePursuitRecPath().	32
Figure 42. EscribirDatosPLC call inside PurePursuitRecDubins().	33
Figure 43. Dubins path calculation.....	33
Figure 44. Connection to read form PLC.....	34
Figure 45. Disconnection after reading from PLC.....	34
Figure 46. Reading the angle's sensor data from the PLC method.....	35
Figure 47. Start driving on GPS secuencia code.....	36
Figure 48. Flip-Flop.....	37
Figure 49. Flip-Flop square signal.....	37
Figure 50. Comparison between the calculated angle and the real one.....	38
Figure 51. Steering right or left code.....	38
Figure 52. Driving at constant speed.....	40
Figure 53. Steering program's secuencia without any feedback.....	40
Figure 54. Steering program without feedback.....	41
Figure 55. Steering program's secuencia without the sensor's feedback.....	42
Figure 56. Steering range of the vehicle.....	43
Figure 57. Wooden planks aligned with the axi.....	43
Figure 58. steering angle's range measurement.....	44
Figure 59. Steering angle's reference system.....	44
Figure 60. Wiring diagram of the angle sensor.....	45
Figure 61. Angle range condition.....	47
Figure 62. Angle conversion's secuencia for the left interval.....	48
Figure 63. Angle conversion's secuencia for the right interval.....	49
Figure 64. Sending the actual angle's value to a DataBlock.....	49
Figure 65. Turnig radius of a vehicle.....	50
Figure 66. Guidance problem.....	51
Figure 67. IMU sensor and Euler's angles.....	52
Figure 68. RTK antena.....	52

LIST OF TABLES

Table 1. Technical properties of the AGV's gasoline engine.....	3
Table 2. Principal features of the angle sensor.	10
Table 3. Technical specifications of the angle sensor.	11
Table 4. Vehicle measures.	21
Table 5. Value range conversion of the actual angle.	45

1. ABSTRACT

1.1. ESPAÑOL

Desde hace un tiempo, la automatización se ha convertido en una de las estrategias más importantes a implementar en la industria y en el día a día de las personas debido a sus beneficios, como una mayor seguridad o mayores tasas de producción. Por todo ello, la UCLL trabaja desde hace casi cuatro años en el proyecto *Gepland Fruit* que consiste en un vehículo autónomo que facilita la tarea de recogida de cajas en los campos de fruta. El objetivo principal de este proyecto es conseguir una conducción autónoma mediante conexión GPS.

Se ha utilizado un programa de código abierto llamado AgOpenGPS que proporciona conexión GPS y cálculos de ángulo de dirección. En este proyecto, el código fuente ha sido modificado para que se conecte al sistema de control PLC que acciona el vehículo AGV autónomo.

El propósito global sería lograr la conducción automática a través de un campo de frutas mientras recogiendo cajas que estaban al lado del camino.

Palabras clave: GPS, Autoguiado, PLC, vehículo AGV, AgOpenGPS, Angulo de giro.

1.2. ENGLISH

For some time now, automation has become one of the most important strategies to implement in the industry and in people's daily lives due to its benefits such as an improved safety or higher production rates. For all this, the UCLL has been working for almost four years in the *Gepland Fruit* project that consists of an autonomous vehicle that facilitates the task of box collection on fruit fields. The main goal of this project is to achieve autonomous driving using GPS connection.

An opensource program called AgOpenGPS has been used thanks to its features, providing GPS connection and steering angle calculations. In this project, the source code has been modified so that it connects to the PLC control system that drives the autonomous vehicle.

The final goal would be to achieve automatic driving across a fruit field while picking up boxes that lay next to the path.

Keywords: GPS, Autoguidance, PLC, AGV vehicle, AgOpenGPS, steering angle.

1. INTRODUCTION

1.1. PROJECT GEPLAND FRUIT VEHICLE'S CONTEX.

Due to loss of strength in the fruit sector as a consequence of competition and labor shortages. (For example, the production of apples was set to fall by 63%) and increased competitiveness of the Belgian fruit sector.

In this context the Gepland Fruit project started on 1 sept 2019, oriented on:

- Logistical support during collection of pallet and transportation in farm fields.
- Use of green/clean energy
- Automation systems to improve effectiveness and efforts. Such as box detection by camera AND autosteer system.

So, in addition to all the functions that the AGV vehicle already has implemented, this part of the project seeks to update it and include an autonomous GPS guidance system.

1.2. ENERGY SYSTEM.

1.2.1. POWER GENERATORS.

2.2.1.1. GASOLINE ENGINE.

PTM hydraulic power pack powered by a "PTM760PRO" gasoline engine of max. power of 26 HP. It can be purchased fully assembled on a galvanized frame. The hydraulic power pack includes a gasoline engine, gear pump, hoses, 4/3 direction valve, pressure relief valve, hydraulic tank, pressure gauge and all necessary (quick) fittings and couplings. The 18 l gasoline tank, the oil cooler and the 40 A alternator are options at the customer's choice, which are included for our AGV project.



Figure 1. "PTM760PRO" gasoline engine

Table 1. Technical properties of the AGV's gasoline engine.

TECHNICAL PROPERTIES	
Model	PTM200-50PRO
Engine type	22 HP, Oil-cooled, V-twin, Gasoline engine
Max. Assets	26 horsepower (19kW)
Cylinder content	760 cc
Starting system	Electric start (incl. 12v charging coil)
Hydraulic pump	Group 2 gear pump (50/60 liters per minute)
Max Continuously press	240 bars
Oil flow adjustment range	50-60 liters
Slider	4/3 direction slide (center position is rest position)
Hydraulic tank	30 liters
Weight without oil	79kg
sound	80dB
Battery	35Ah 12V

2.2.1.2. ORBITAL HYDRAULIC MOTOR.

Orbit hydraulic motors are widely used for their better control, smaller weight per unit of power and compactness. This kind of motor possesses excellent low-speed high-torque (LSHT) characteristics due to its high output torque-inertia ratio and simplicity in construction.

These hydraulic machines are widely used in mechatronic systems with a gearless hydraulic drive of active working bodies and running systems of construction, railway, agricultural, drilling, municipal, logging and other self-propelled equipment. A distinctive feature of these hydraulic motors is the presence of external and internal rotors with a special hypocycloidal tooth profile.

In this AGV project, we use two of them with a capacity of 20 l/min each.



Figure 2. Orbit hydraulic motor.

2.2.1.3. BIO-GAS CONVERSION

Info: one bottle of 6kg = 47\$, bottles are refillable

Petrol hydraulic power pack → Conversion to gas engine on Bio-gas

Because we want to work with a compressible gas, we need a *Pressure regulator* for the gas bottles

-Consequences: Power decreases a number of horsepower

The company PRIMAGAZ is a Manufacturer based in Tessengerlo (Belgium), which carries out its activity in the Production and distribution of gaseous fuels sector.

Primagaz easy blue is a liquefied petroleum gas (LPG) system usually used for home and camping that uses natural gas instead of propane or butane gas. The system consists of an LPG cylinder, a regulator and a hose that connects the cylinder to the appliances.

It is easy to install and can be used to heat water or cook and, in this case, to power the engine of a vehicle. It is safe and complies with all safety regulations and standards. In addition, it is an environmentally friendly option as LPG is a clean and efficient energy source.

PRIMAGAZ EASY BLUE is available in several cylinder sizes, and they can be swapped and refilled at service stations or authorized distribution points.

2.2.1.4. SOLAR PANEL.

X2 solar panel power 115W=230W

The Photovoltaic system uses a **MPPT** system (Maximum Power Point Tracking) Figure 4. It is a technique used in photovoltaic systems to maximize the power output of a PV panel by continually adjusting the operating point of the panel to match the maximum power point of the panel.

The maximum power point of a PV panel is the point at which the panel is able to produce the most power. This point is determined by many things such as the temperature, intensity of sunlight, the inclination of the panel in relation to the sun, and load resistance of the panel. The operating point of the panel is the point at which it is currently producing power.



Figure 3. 115W solar panels.



In conclusion, an MPPT system monitors the operating point of the PV panel and adjusts it to match the maximum power point. This allows the panel to produce the most power possible to increase the efficiency.

Figure 4. MPPT system.

1.2.2. ENERGY STORAGE SYSTEM.

- Electric system 12-24 V_{cc}: some components work with 12V and some with 24V. We use a 12/24V 20A 480W converter.
- With 12V coming directly from the solar panels we charge a 12,8V 200Ah 2,56kWh **Li-ion battery** which is a rechargeable battery that uses lithium ions as the main charge carrier. They have a high energy density, which means they can store a large amount of energy in a small package. This makes them ideal for use in vehicles. They do NOT suffer from the "memory effect," which is a phenomenon where their capacity is reduced if they are not fully discharged before recharging.
- The solar panels also power a Start Pb Battery that generates the initial spark of the generator, using a chemical reaction between lead and sulfuric acid to produce electricity. They are commonly used as starting batteries in vehicles. These batteries are designed specifically to provide a large burst of power to start an engine. They are typically larger and their crank amp rating is higher than other types of lead-acid batteries, which allows them to deliver a high current to the starter motor.

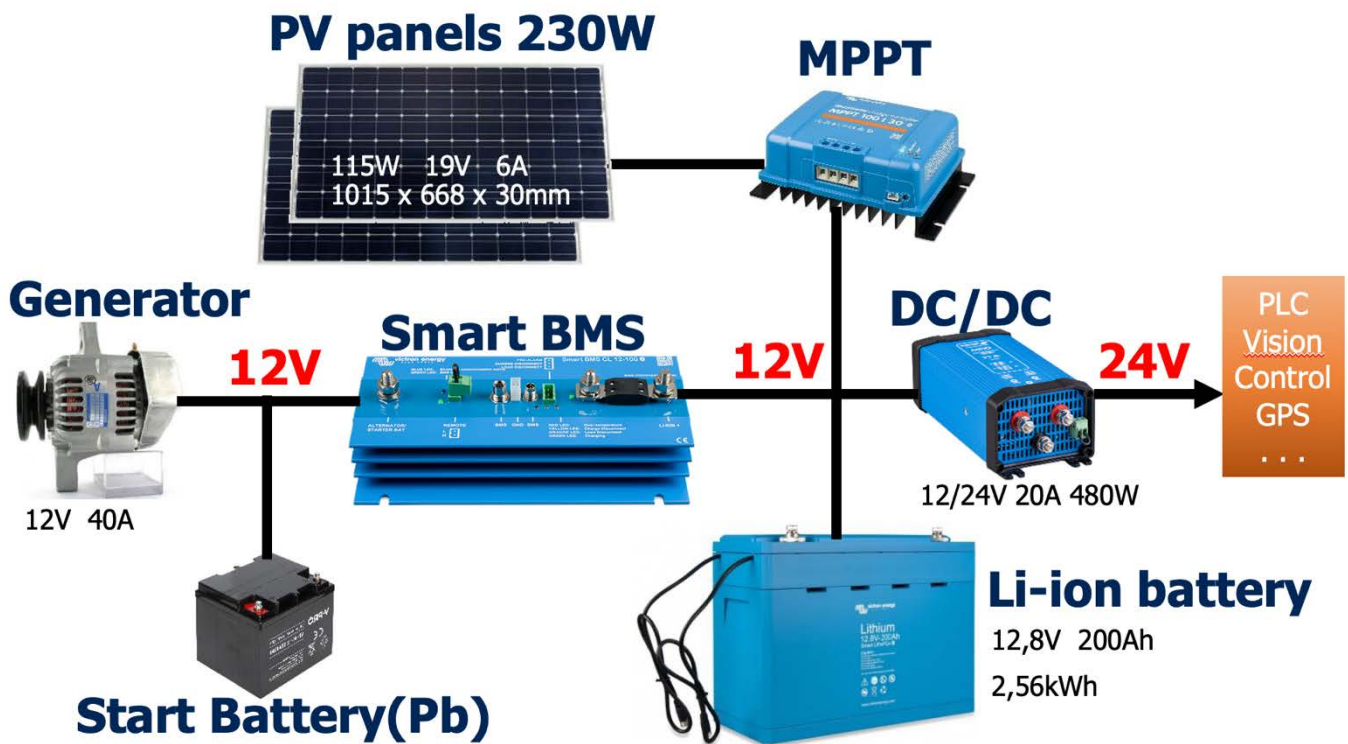


Figure 5. Energy storage system diagram.

The storage system uses a **Smart BMS** (Smart Battery Management System) Figure 6 to monitor and manage the performance and health of the batteries. It consists of hardware and software that monitor and control the charging and discharging of the battery, as well as provide information about the battery performance and durability.

A smart BMS can provide a variety of benefits, including:

- Improved battery performance and longevity: by monitoring and controlling the charging and discharging of the battery, a smart BMS can help extend the life of the battery and improve its performance.
- Increased safety: a smart BMS can help prevent overcharging, over discharging and other issues that can damage a battery or cause it to fail.
- Enhanced system performance: by optimizing the performance of the battery, a smart BMS can help improve the overall performance of the system PROCESS it is used in.

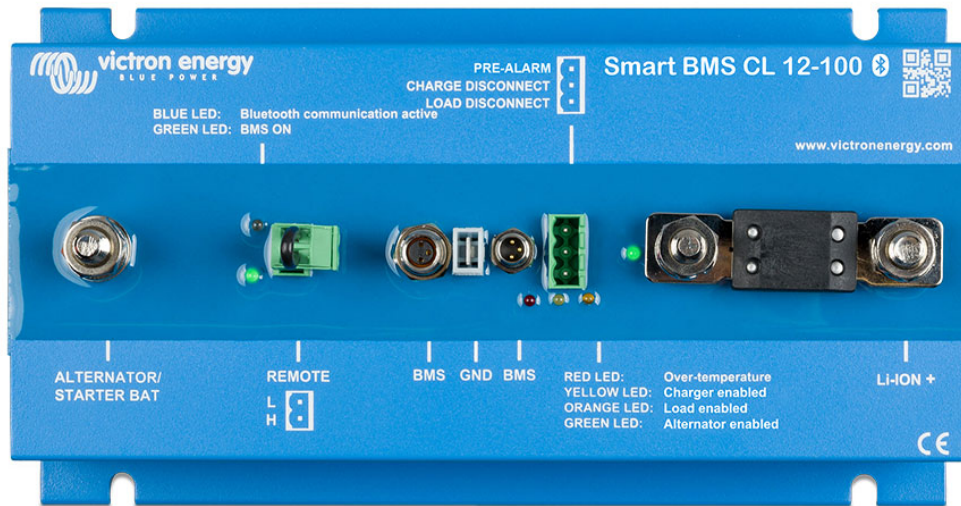


Figure 6. Smart Battery Management System.

1.3. KEY COMPONENTS.

1.3.1. GPS DEVICE

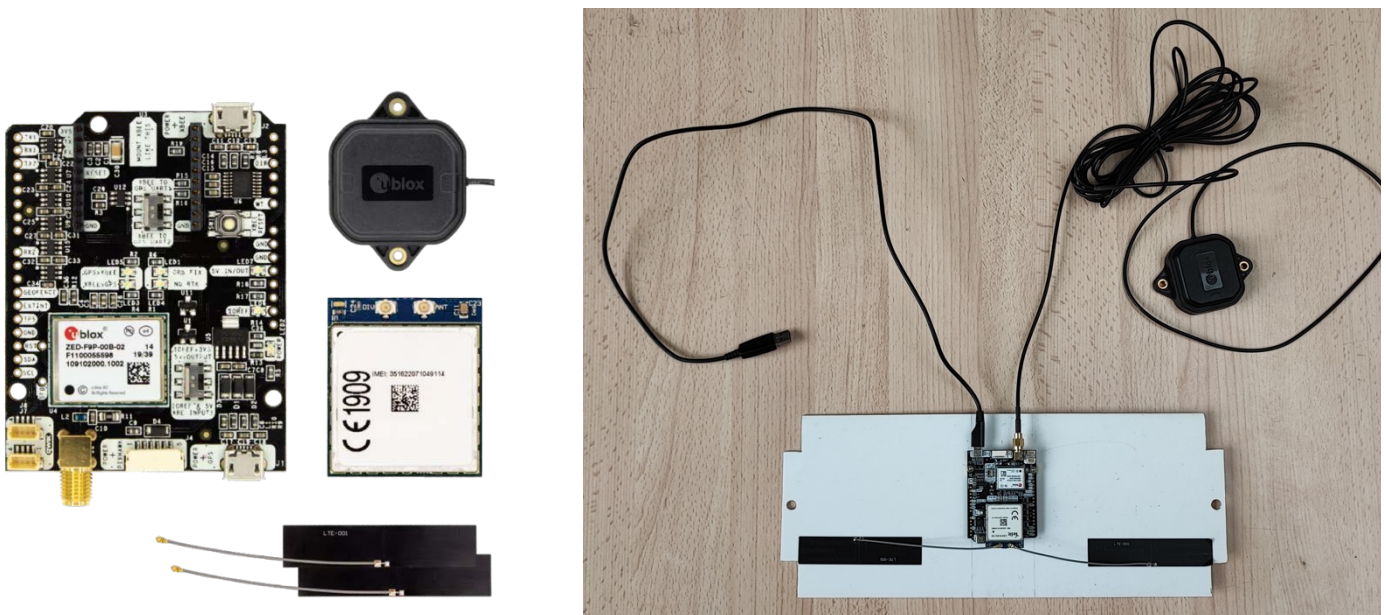


Figure 7. SimpleRTK2B V3 NTRIP- GPS device.

The GPS model we use, the *SimpleRTK2B V3 NTRIP* is a GPS device that is used for real-time kinematic (RTK) positioning. RTK is a high-precision positioning technique that uses measurements from a base station and a rover (mobile) device to calculate the position of the rover with very high accuracy. In our case, the SimpleRTK2B V3 NTRIP

is designed to be used as a rover device in an RTK system. It can receive RTK corrections over the internet using the Networked Transport of RTCM via Internet Protocol (NTRIP) protocol, which allows it to achieve sub-decimeter accuracy in real-time, in this case +/-1 cm accuracy.

The SimpleRTK2B V3 NTRIP is manufactured by ArduSimple, a company that specializes in developing RTK systems and related technology. The device is also equipped with a built-in inertial measurement unit (IMU) and a barometer, which can be used to improve the accuracy of the position solution.

1.3.2. CONTROL SYSTEM.

PLC SIEMENS SIMATIC S7-300

Our AGV vehicle is controlled by a PLC, specifically the SIEMENS SIMATIC S7-300. The Siemens S7-300 is a programmable logic controller (PLC) that is used to control industrial automation systems. PLCs are computer-based systems that are used to control and monitor industrial processes and machinery. In our case, all the movements and programmable functions the AGV can do.

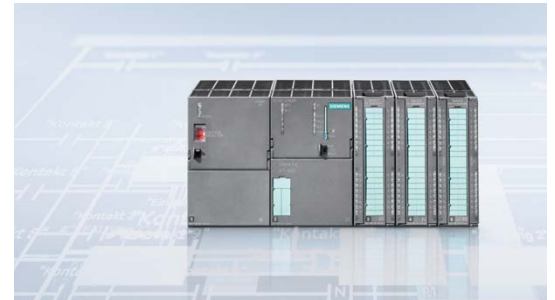


Figure 8. SIEMENS PLC.

The SIMATIC S7-300 has a large variety of advantages that make it one of the best controller devices on the market. Some of its best qualities, according to the official SIEMENS webpage, are that it is used in many applications worldwide and has been proven successful millions of times. The SIMATIC S7-300 universal Controllers saves on installation space and features a modular design. A wide range of modules can be used to expand the system centrally or to create decentralized structures according to the task at hand and facilitates a cost-effective stock of spare parts. SIMATIC is known for continuity and quality. It's equipped with a powerful CPU and a range of input/output (I/O) modules that can be used to interface with a variety of sensors, actuators, and other field devices. It also has a built-in Ethernet interface that can be used to connect the PLC to other devices and systems on a network.

Technical information about the SIMATIC S7-300:

For the overview, it is a modular mini PLC system for the low and mid-performance ranges. With comprehensive range of modules for optimum adaptation to the automation task. Flexible use through simple implementation of distributed structures and versatile networking, it has a fan-free design and can be expanded easily.

For the expansion system, it can get expanded at most with 32 modules on CC and 3 EUs: in total, a maximum of 3 expansion units (EUs) can be connected to the central controller (CC). Eight modules can be connected to each CC/EU.

Regarding the communication, the S7-300 has different communication interfaces such as communications processors for connecting to the bus systems AS-Interface, PROFIBUS and PROFINET/Industrial Ethernet that is the case for our AGV vehicle, and communication modules for point-to-point connections and multipoint interface (MPI), integrated into the CPU, which allows it to be easily integrated into a variety of automation procedures.

1.3.3. ANGLE SENSOR.

Named KTC-300-P - Linear motion position sensor 0.05 % 5 kOhm Voltage divider, Variomh EuroSensor.



Figure 9. KTC-300-P - Linear motion position sensor.

In general, the KTC standard linear transducers are designed for direct absolute measurement and are available in different stroke lengths up to 1250mm.

In

The one installed in the AGV has a $\square 33$ Section, and standard dimensions of 50-1250mm.

The sensors give exceptional resolution with repeatability of $\pm 0.01\text{mm}$ and outstanding linearity of $\pm 0.05\%$ maximum.

The sensors can be directly mounted into the mechanical piston using a screws-and-nuts system eliminating the use of racks and pinions or similar devices.

The KTC series has a solid stainless-steel shaft with long bearings in the housing for a robust and smooth operation with long life. The slider has a ball coupling which reduces the effects of misalignment with the actuating part.

An improved technique for making connection to resistance track (Double Trimming Technique) ensures the higher degrees of reliability and linearity, while multi-fingers wipers stabilize output signals, even in the most adverse working conditions.

The fixing feet are adjustable to the desired positions.

Table 2. Principal features of the angle sensor.

PRINCIPAL FEATURES	
Durable bearing and slider	
Anodized Aluminum housing	
Smooth low noise output from conductive plastic track	
DIN 43650 ISO 4400 Connector	
Very Long Life	>100 x 10 ⁶ Cycles
Stroke	50-1250mm
Outstanding Linearity	$\pm 0.05\%$
High Resolution	Infinite
Excellent Repeatability	$\pm 0.01\text{mm}$
Max operating speed	10 m/s max.
Operating temperature	-30 ~ 100 °C
Storage Temperature	-50 ~ 120°C

Table 3. Technical specifications of the angle sensor.

Technical Specifications	
Sealing - KTC	IP60
Sealing - KTC-P	IP65 I
Current Resistance	≤ 10mA
Current Wiper	≤ 1mA
Operating Force	≤ 2N (KTC)
	≤ 10N (KTC-P)
Power Consumption	3W-10W
Output Smoothness	< ± 0.1% against input voltage
Input Voltage	60 V Max
Insulation Voltage	500V-1 min Residue < 5 μA
Vibration	IEC 68-2-6:1982 10g
Shock	IEC 68-2-29:1968 40g

1.3.4. HYDRAULIC SYSTEM.

2.3.4.1. HYDRAULIC POWER PACK.

The hydraulic power pack of which this vehicle is composed consists of two hydraulic cylinders that control 1. the steering (left-right) and 2. the lifting pallet (up-down).

In addition, there are two other valves that control the engine for driving (forward-backwards...). Lastly, there is four electrically operated prop pilot valves.



Figure 10. Hydraulic power pack

All this is controlled from the PLC for remote driving and four levers for the direct driving.

2.3.4.2. DOUBLE ACTING HYDRAULIC CYLINDER.

The AGV uses two double acting hydraulic cylinder with a capacity of 5 l/min each.

A double-acting hydraulic cylinder is a type of hydraulic cylinder that uses pressurized fluid to extend and retract the cylinder rod. This is in contrast to a single-acting cylinder, which only uses pressurized fluid to extend the cylinder rod. The cylinder has ports at

both the rod and base end, which allow fluid to enter and leave the cylinder to extend or retract the rod. The double-acting cylinder is more versatile than a single-acting cylinder because it can be used for both pushing and pulling applications.



Figure 11. Double acting hydraulic cylinder.

2. OBJECTIVE.

The general objective of this project is to create a new system to make the previously described vehicle (AGV), achieve autonomous driving on a pre-recorded path.

Through research and the assistance of the University of Kortrijk, the approach chosen to meet these goals was to use the existing program AgOpenGPS and modify its source code, taking advantage of the fact that it is an open-source program (for its free management), and thus achieve the communication with the PLC TIA-Portal software (the most complicated part of the project) and being able to manage and program the movements of the vehicle from the PLC. All this taking full advantage of the features and options that the AgOpenGPS program already provides, like the connection with GPS, the option to pre-record a path to later be able to follow it, and all the computational calculations that this implies.

Going into more detail, an external library called *Sharp7* will be used to send and receive key parameters to the TIA Portal software that controls the AGV's PLC and with these parameters try to control the hydraulic system and therefore, the movement of the vehicle. In Figure 12, it is shown a block diagram describing the system.

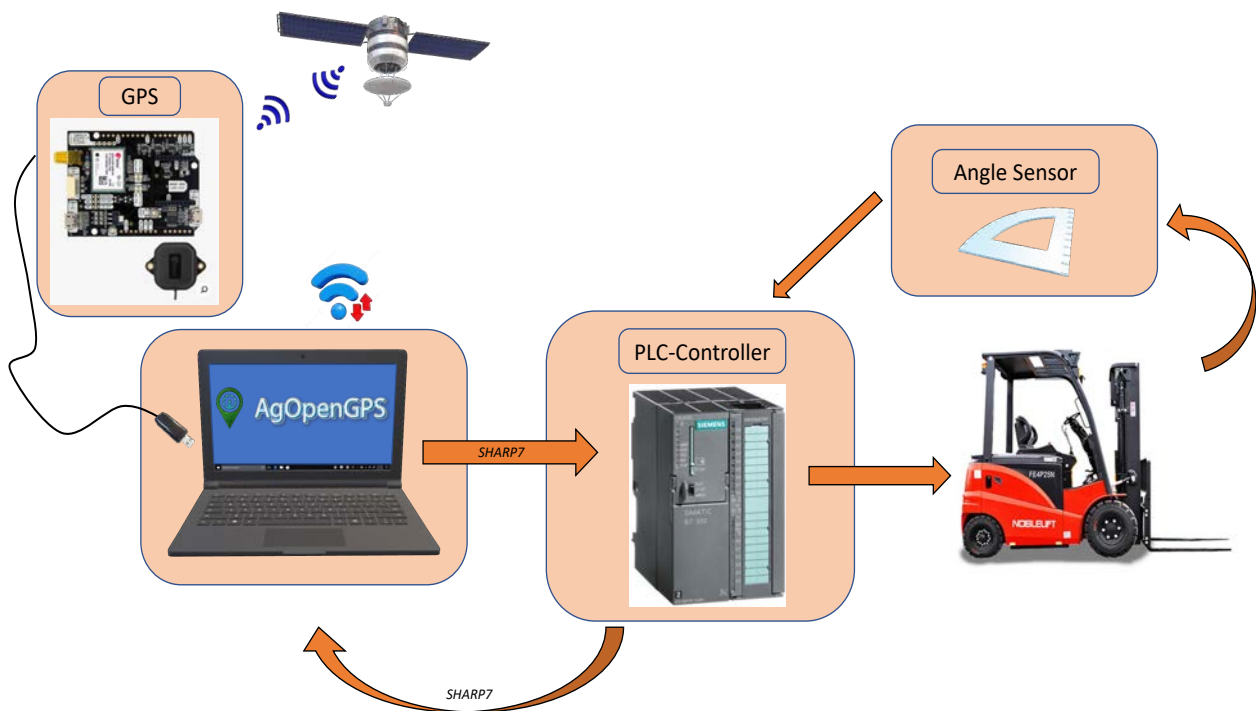


Figure 12. Project's system block diagram.

3. SOFTWARE AND METHODOLOGY

3.1. SOFTWARE

3.1.1. AgOpenGPS.

Quoted verbatim from the AgOpenGPS official YouTube webpage video presentation: "AgOpenGPS started as an experiment by a farmer in Alberta (Canada), Brian Tishler, to test some ideas he had for a DIY tractor auto-steer. Its name stands for Agricultural Open-Source GPS and it is a precision agricultural project built by farmers worldwide who are interested in learning how it works by building it."

It uses advanced GPS or GNSS technology to perform precision tasks such as auto-steer auto section control, mapping, automatic turn-around (U-turns) and much more. It is also fully open source meaning free to use and modify and highly customizable."

The items you would need to run the project for what it was created for are the following as we can also see in Figure 13:

- A portable Windows device running Windows 7 or newer to run the main AgOpenGPS software.
- An auto-steer box which contains the printed circuit board with Arduino microcontroller and other components for reading sensors and controlling the steering or an implement.
- The IMU (Inertial Measurement Unit) which is a device used to calculate the roll and heading of the tractor used to correct the GPS position for tilt.

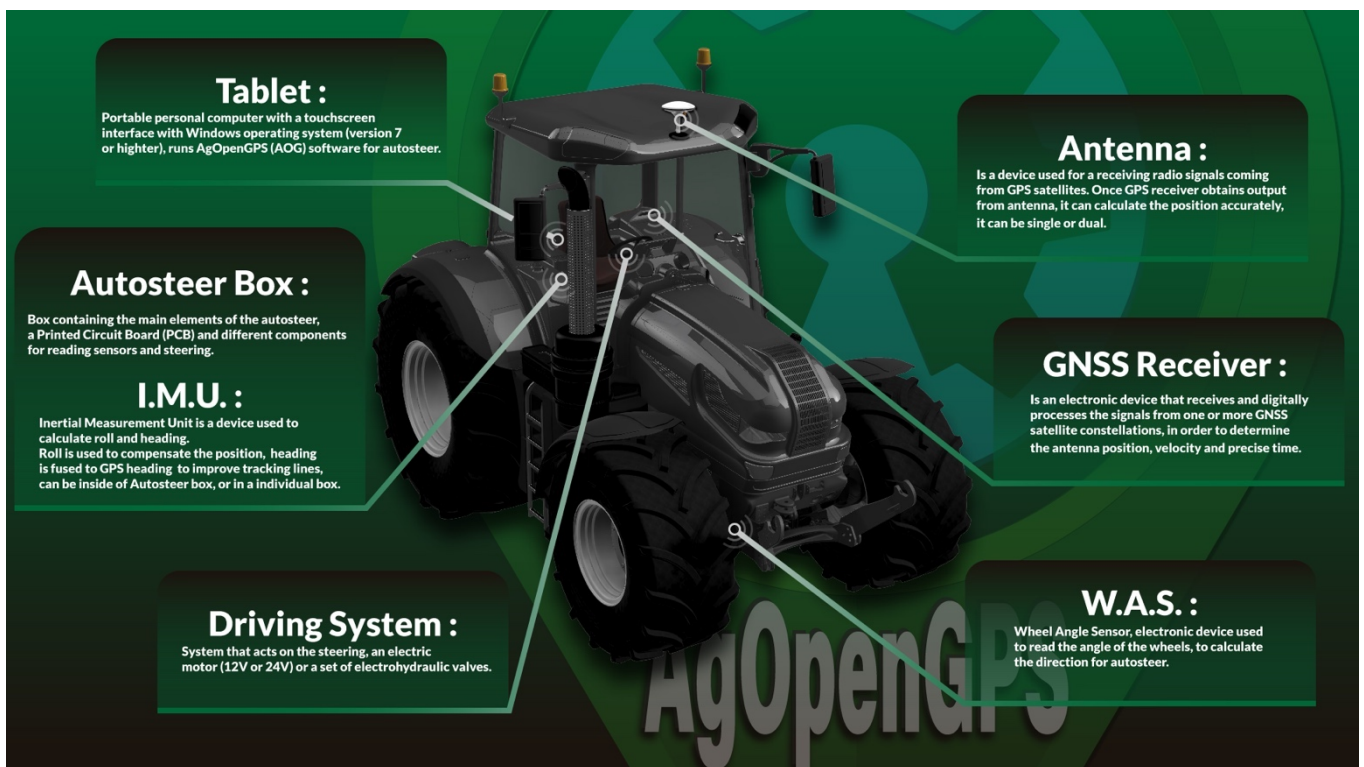


Figure 13. Original AgOpenGPS project's components diagram.

- The steering system, method of controlling the steering wheel angle, usually a 12- or 24-volt electric motor that turns the physical steering wheel or a set of factory or aftermarket integrated electrohydraulic valves.

- The wheel angle sensor (WAS): an electronic sensor used by the auto-steer box to sense the angle of the steering wheels and calculate the needed changes in steering angle.
- FINALLY, a GNSS receiver –an antenna that receives radio signals from the global navigation satellite systems/constellations (GNSS) in orbit and it connects to the GNSS receiver via a coax cable in order to determine the position, velocity and precise time.

In the AGV Geplant Fruit project, we had to work with the already-integrated system in the , as for example, the PLC that controls the hydraulic system, system that runs the forward-backwards driving, left-right steering, and the up-down movement of the pallet forks. Also, the already-integrated sensors such as the steering-angle sensor built in the steering piston that it's indispensable for the algorithm to run a close-loop and be able to autosteer as we will explain later. This means that most of the elements and features of the initial AgOpenGPS project from Brian Tischler will not be used or will have to be modified in order to use them and work in our system.

With that said, now I am going to explain how the elements that can be implemented directly have been configured:

4.1.1.1. GPS-NTRIP CONNECTION.

One of the key features of AgOpenGPS is its ability to connect to a GPS device and receive real-time GPS data. The program supports a range of GPS devices, including standalone GPS receivers, as well as GPS modules that can be integrated into other agricultural machinery. AgOpenGPS can connect to GPS devices using a variety of communication protocols, including serial, USB, and Bluetooth. Once connected to a GPS device, AgOpenGPS can use the GPS data to

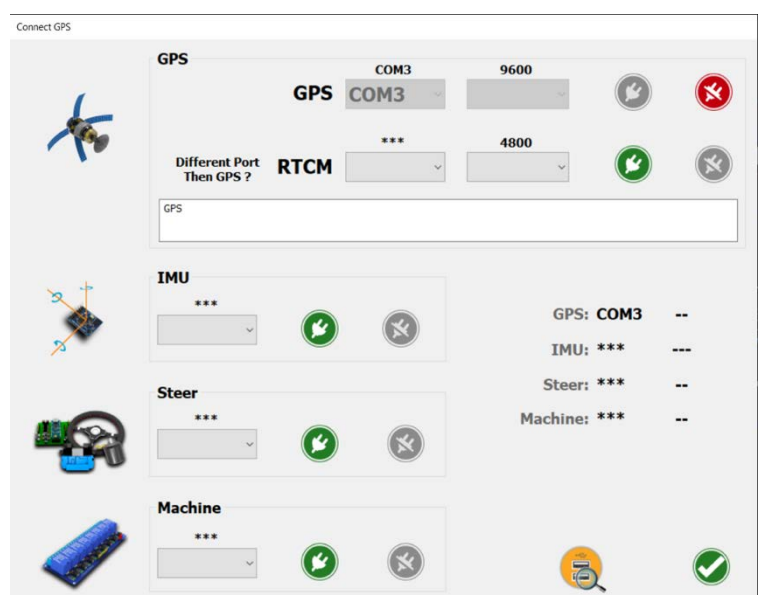


Figure 14. GPS ports communication menu.

provide a range of precision agriculture and guidance features. These may include real-time mapping and positioning, automatic guidance and steering, and field data collection and analysis. AgOpenGPS also supports a range of additional sensors and devices, such as depth sensors, yield monitors, and weather stations, which can be used to provide additional data for precision agriculture applications.

Only with the GPS-satellite connection we would not have the necessary precision to perform the self-driving functions efficiently. This is due to time the signal travels to our GPS receiver and also the signal disturbances by the local air layers that equals a margin of error on position (about meters) that it's not acceptable.

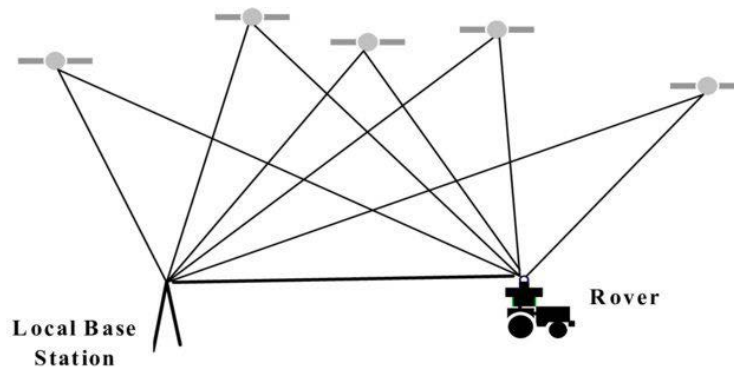


Figure 15. Satellite-Antenna-GPS device communication diagram.

That is why we need to use a fixed base station in the same local air layer so that the information flows from the base station to the rover GPS, via radio, mobile internet, whatever is necessary using the so-called 'NTRIP' protocol. Luckily Flanders offers free NTRIP service (FLEPOS network) (see Figure 16)

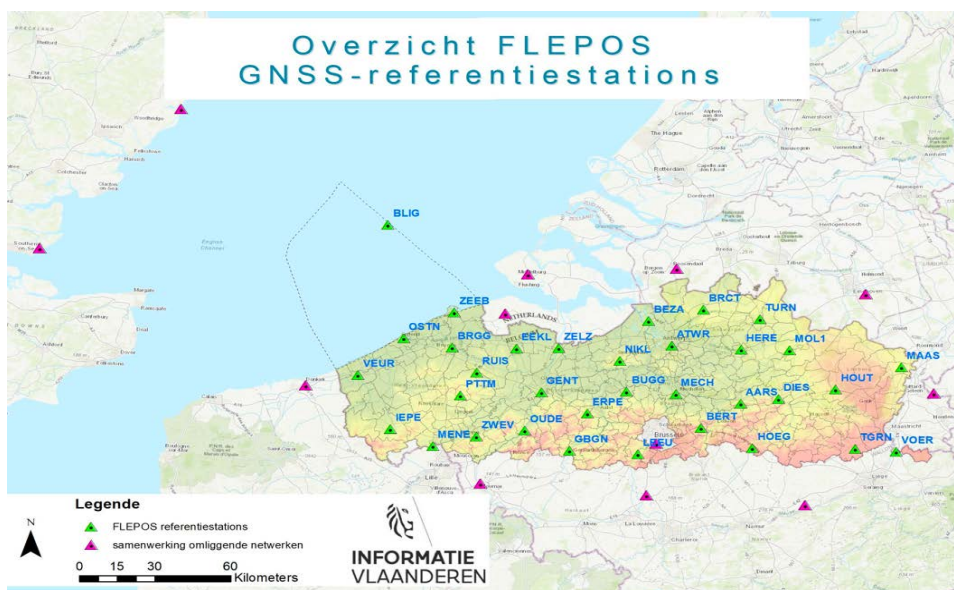


Figure 16. FLEPOS network on flanders.

The accuracy of the rises with:

- Number of satellites in view.
- The proximity base station (<20 km).
- GPS: meters error.

As shown in Figure 17

FLOAT: error < 1m and often < 10 cm

FIX: error < 2 cm

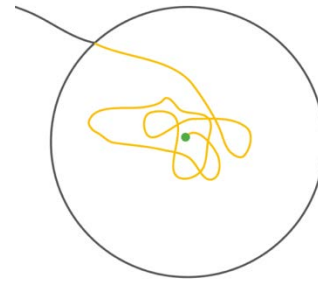


Figure 17. GPS error demonstration.

An error between 1m and 10cm is the minimum we need to have an efficient autosteering.

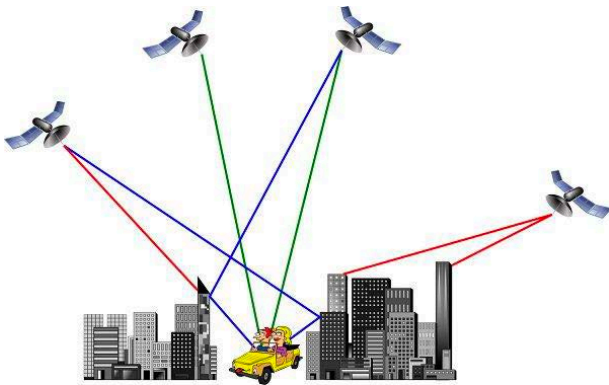


Figure 18. GPS Connection problems examples.

The main problems that can disturb the signal are:

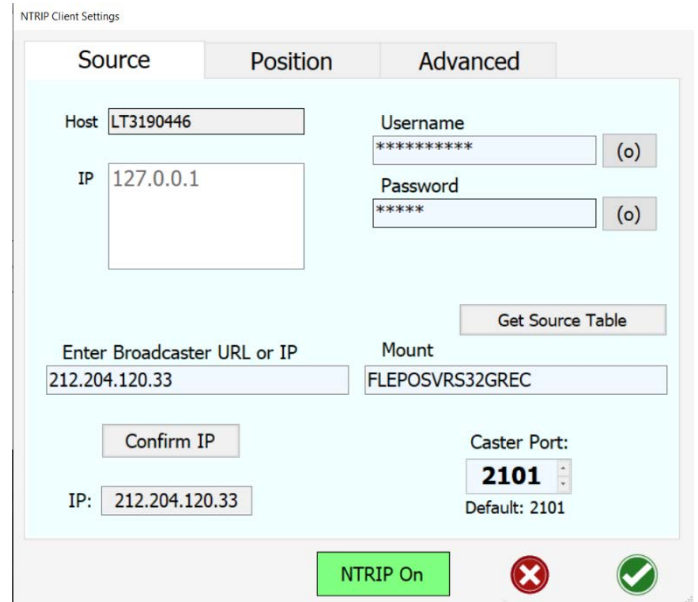
- Obstructions and reflections of signals in the environment generating multipath effects.
- Disturbance factors in the environment: concrete, metal, objects with water (like trees)

As represented in Figure 18.

Inside the program AgOpenGPS we can set the configuration to make the connection to the NTRIP web.

You must first request it so that they can send you all the information for the connection to the FLEPOS network (as has already been said, free of charge). This information consists of a username, a password, an IP address, etc. (as we can see in Figure 19 below).

Once you establish the connection from AgOpenGPS, the computer must always remain connected to internet so as not to lose this connection. Obviously, this project is being designed to carry out tasks abroad, so the best option so far is to use the personal mobile device as a portable router and use the 4G or 5G network (depending on your area).



The screenshot shows the 'NTRIP Client Settings' window with three tabs: 'Source', 'Position', and 'Advanced'. The 'Advanced' tab is active. It contains the following fields and controls:

- Host:** LT3190446
- IP:** 127.0.0.1
- Username:** ***** (o)
- Password:** ***** (o)
- Get Source Table:** Button
- Enter Broadcaster URL or IP:** 212.204.120.33
- Mount:** FLEPOSVRS32GREC
- Confirm IP:** Button
- IP:** 212.204.120.33
- Caster Port:** 2101 (Default: 2101)
- NTRIP On:** Green button
- Close:** Red button with 'X'
- Check:** Green button with checkmark

Figure 19. FLEPOS network's connection menu.

4.1.1.2. VEHICLE CONFIGURATION.

We insert in the AgOpenGPS all the information about the vehicle. The type of vehicle, the measurements, (taken with a meter) and type and position of the antenna. All this information helps the program be more efficient when making the necessary calculations for the autosteer.

a) Type of vehicle.



Figure 20. Articulated tractor.

As we have already said, the AgOpenGPS program is designed for farming tractors. The program has a preselection of three types as shown in Figure 21, luckily in our AGV vehicle corresponds to one of them as we can see in **¡Error! No se encuentra el origen de la referencia..** This type it's called **articulated tractor** or, in our situation, **articulated AGV vehicle**

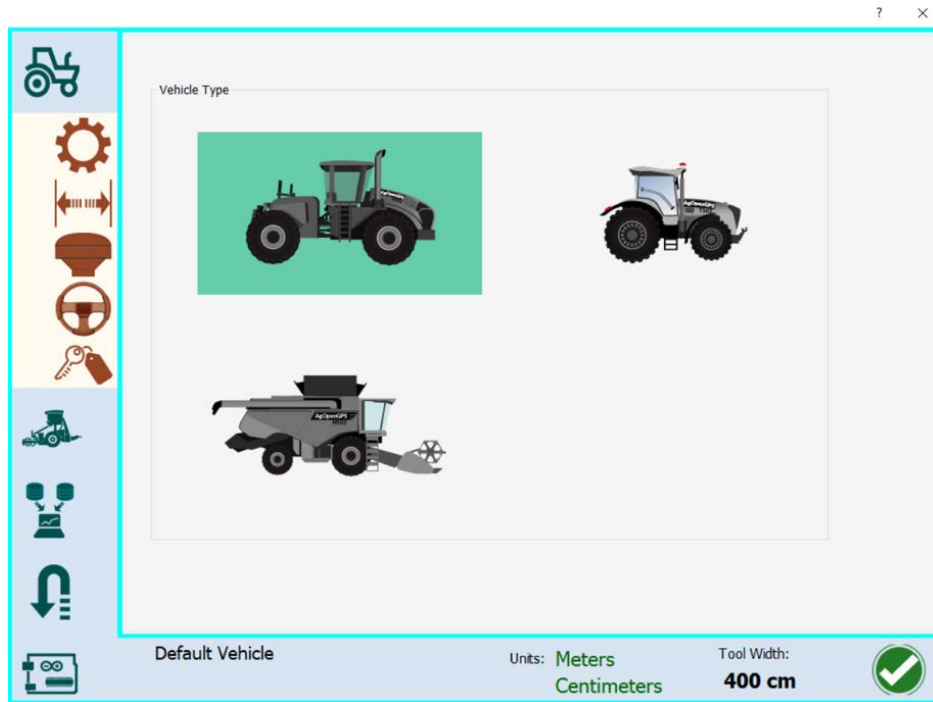


Figure 21. AgOpenGPS vehicle selection menu.

An articulated vehicle is a type of vehicle that consists of a main unit and a trailer unit minimum (it can also have more units connected) that are connected by a hinge. This hinge allows the two parts to pivot, which allows the vehicle to make sharp turns and maneuver in tight spaces. Articulated vehicles are commonly used in agriculture and construction to haul heavy loads, and they are also used in snow removal and other applications where the ability to make sharp turns is important.

In addition to articulated tractors, other types of articulated vehicles include buses, trains, and some military vehicles.



Figure 22. AGV.

b) Measures and capacities of the vehicle

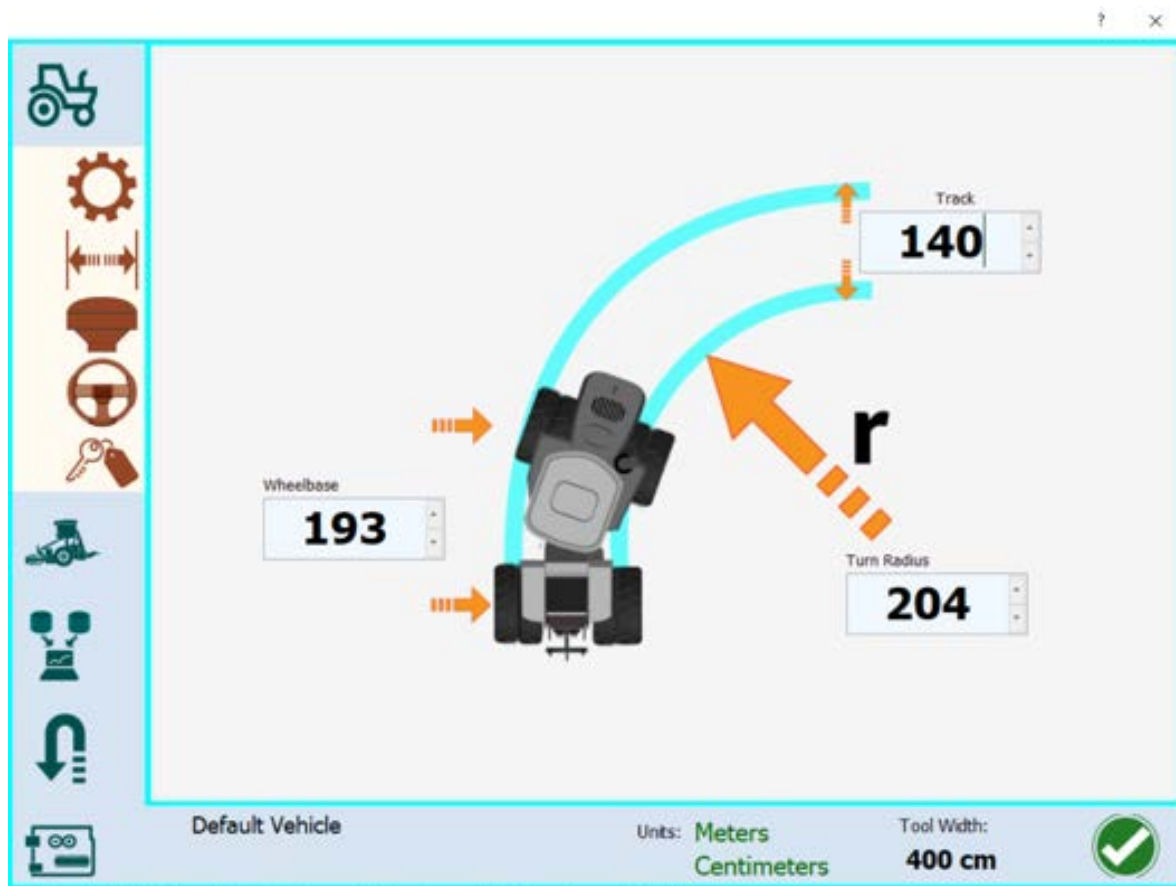


Figure 23. AGV measures.

As we can see in Figure 23, there are three important measures of the vehicle that are necessary for efficient calculations: the distance between the two wheel axes (Wheelbase as the program calls it), the distance (track) between two wheels on the perpendicular axis of the same plane as the wheels axes and the turn radius.

On Table 4 we can see the three measures:

Table 4. Vehicle measures.

VEHICLE MEASURES	
Distance between axes	193 cm
Distance between wheels	140 cm
Turn radius	380 cm

The approximation of the turn radius has been calculated in Section 5.3 TURNING RADIUS. inside. Calculations and Measures

c) Antenna position and configuration.

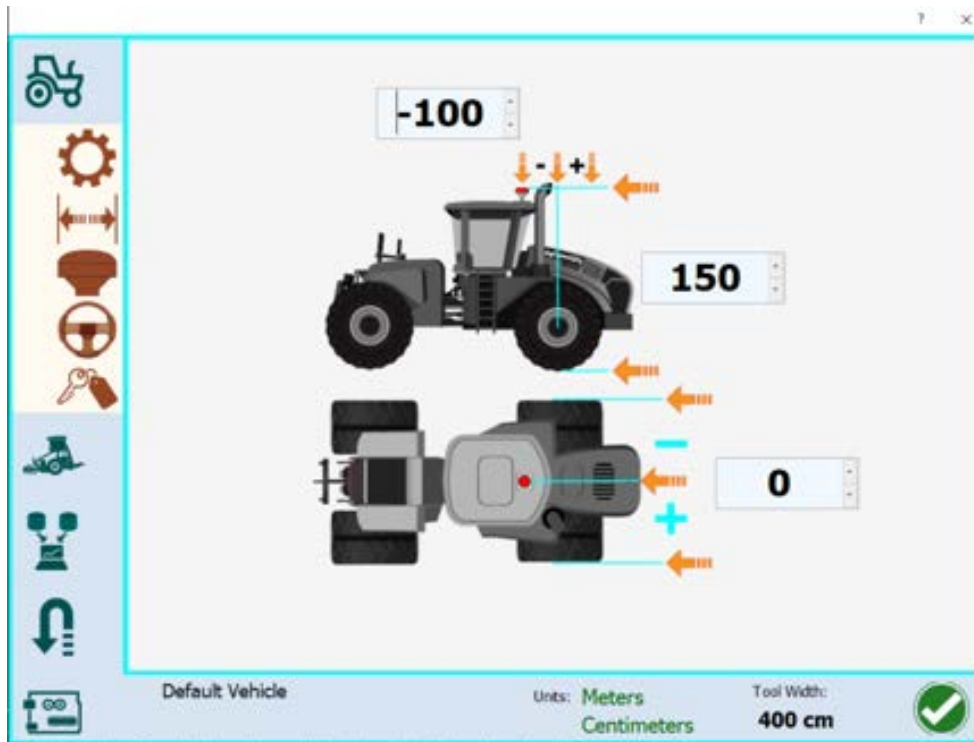


Figure 24. Antenna position relative to the back axis.

For the first distance that we have chosen, as we can see in Figure 24, the antenna is located 100 cm in front of the rear wheel axle (first parameter -100) and in the middle axis of the vehicle (second parameter 0).

According to the article **Sensors, A Simple Method to Improve Autonomous GPS Positioning for Tractors** written in 2011 by Jaime Gómez-Gil, Sergio Alonso-García,

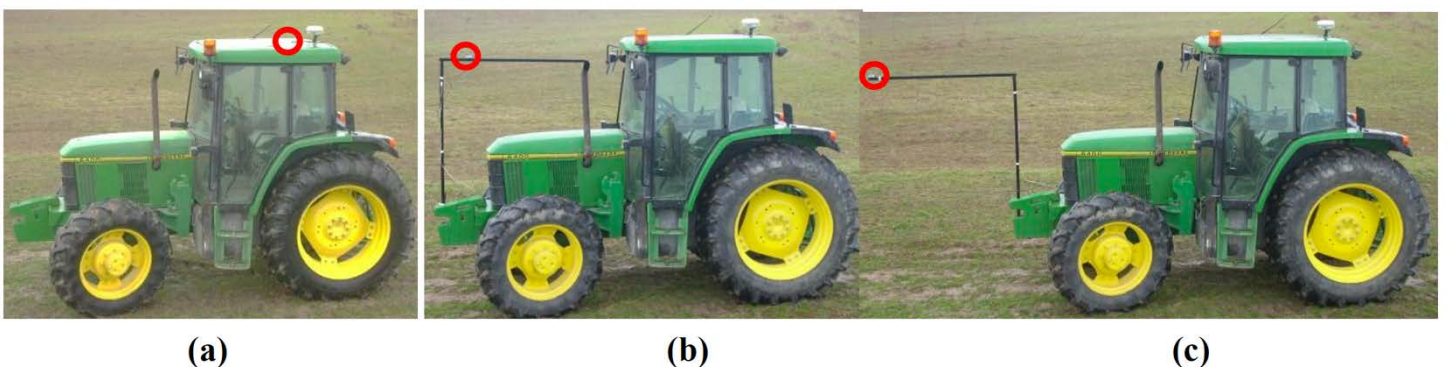
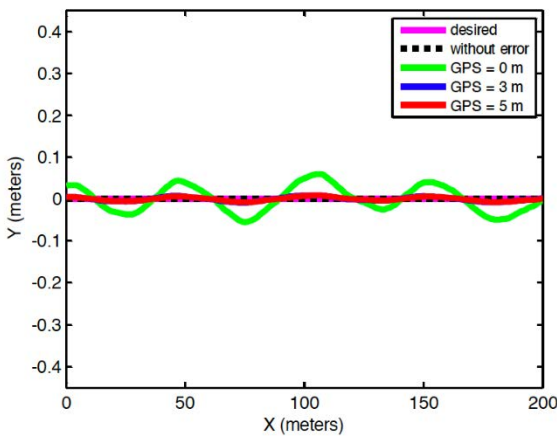


Figure 25. Different GPS positioning.

at the University of Valladolid, they have come to the conclusion that by placing the GPS ahead of the tractor, it is possible to devise a simple and low-cost method a more precise tractor positioning, and thus improve the autonomous GPS guidance performance. This

more precise tractor positioning is achieved because the proposed method uses the received GPS data and the kinematic model of the tractor together for tractor positioning, whereas positioning is only obtained with the received GPS data when the receiver is placed on the rear axle. The proposed method improves tractor positioning when only a GPS receiver is used as the positioning sensor. The improvement in the positioning occurs in the tractor orientation and in the positioning with respect to the transverse axis of the tractor's orientation. It is expected that the improvement ratio will decrease when a tilt sensor is employed and that no improvement will be achieved when the tractor positioning system includes an orientation sensor.

It is possible to observe in Figure 26 that the error is reduced when the GPS receiver is



placed ahead of the tractor like in case b) or c) of Figure 25 with not much difference between both of them.

In our case, the antenna is a Single Fix antenna as we can see in Figure 27 . Which is not the most precise system and needs a lot of complements such as an IMU system that allows to have information about the inclination and rotation of the vehicle.

Figure 26. GPS signal in three different cases.

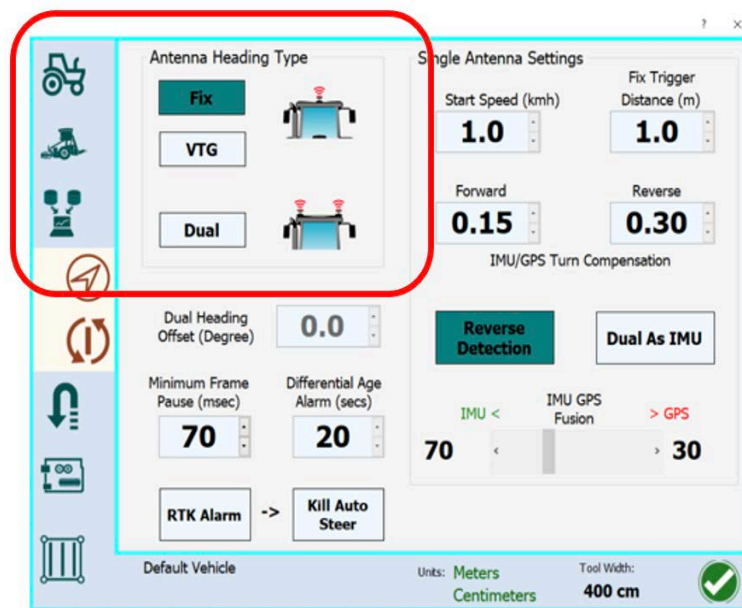


Figure 27. Antenna configuration menu.

A good solution and improvement to the GPS system would be a 'Dual GPS autosteer' with a double antenna as we can in Figure 28 installed on the top of the vehicle that would deliver sublime position, slope, direction... essential information for the best performance.



Figure 28. Double antenna.

4.1.1.3. RECORDED PATH AND AUTOSTEER CONFIGURATION.

The most powerful feature that the AgOpenGPS program presents is the Recorded-Path option. This option consists of driving the vehicle with the GPS activated (actually with the GPS connected to the computer it already works) and recording the route that has been made with the real coordinates.

Once the tour is done, it is recorded in the memory, and you can select it from the menu of the Figure 29 to start going through it.

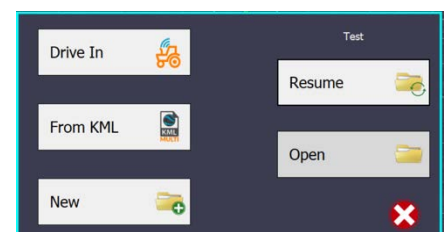


Figure 29. Path selection menu.

The pre-recorded path appears on screen as we can see in Figure 30. Obviously, when starting the route, the vehicle is never going to be exactly on the path, so the program generates an auxiliary path (see Figure 31) and gives instructions for it to be directed to the record path.

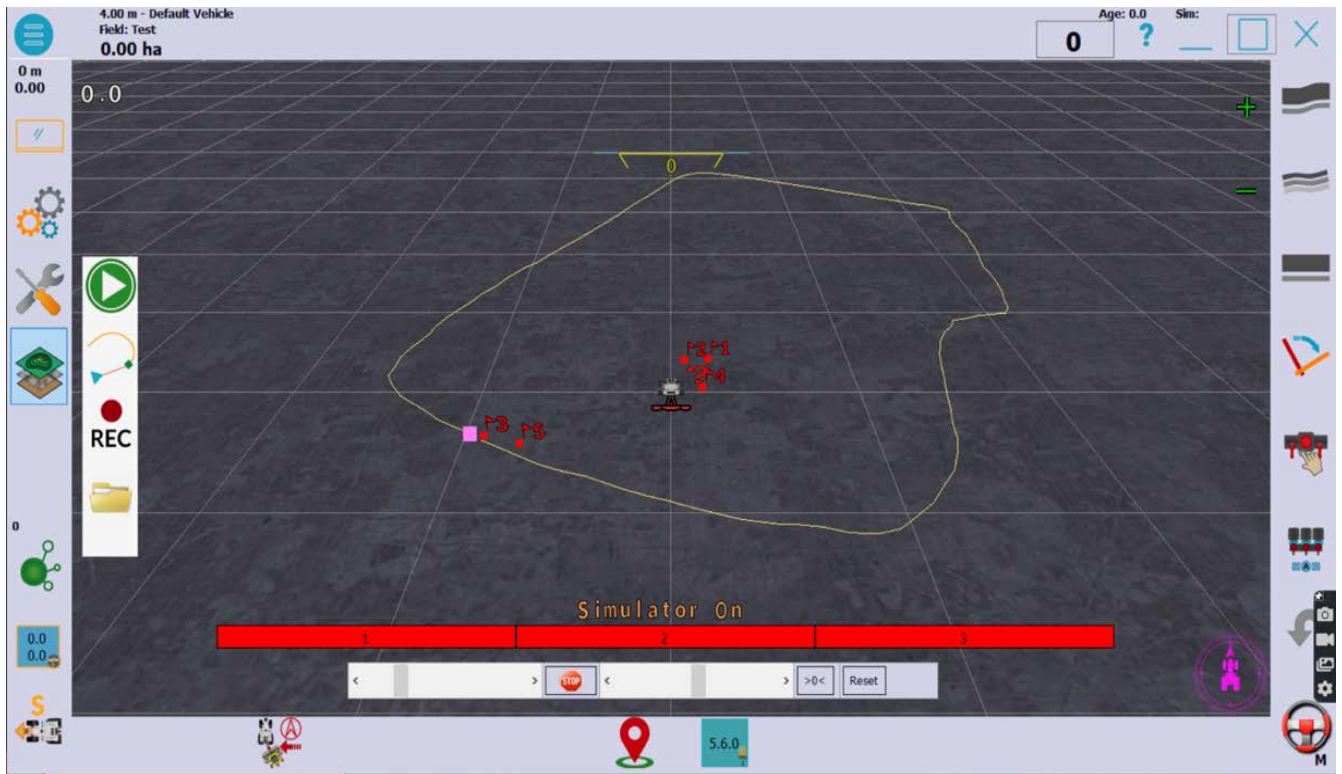


Figure 30. Pre-Recorded path around UCLL technologie building.

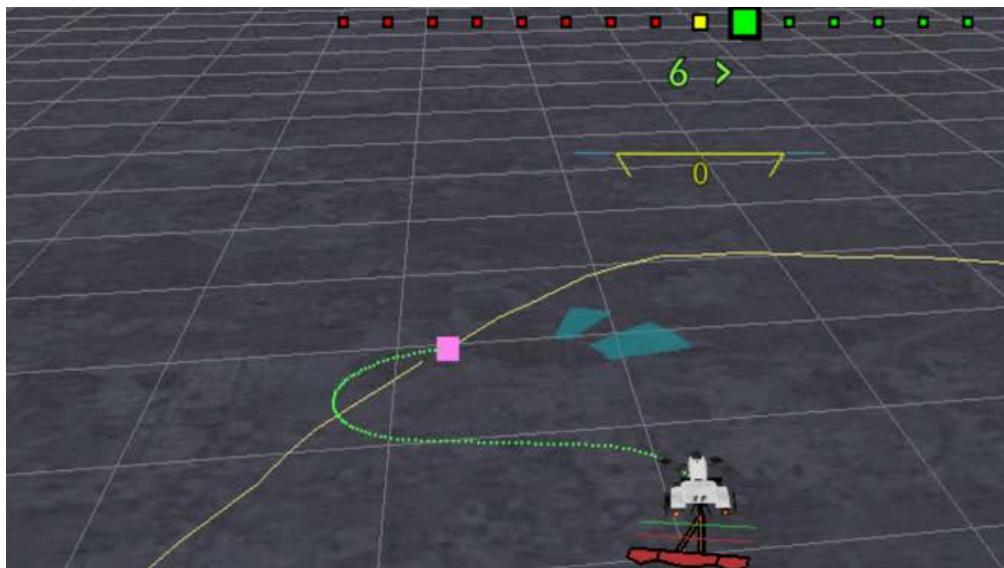


Figure 31. Dubins path to the beginning of the recorded path.

Another alternative to the *Recorded Path* function is to obtain the path to follow from a KML file that you create yourself by drawing from the GoogleEarth system. It is a good alternative but less accurate.

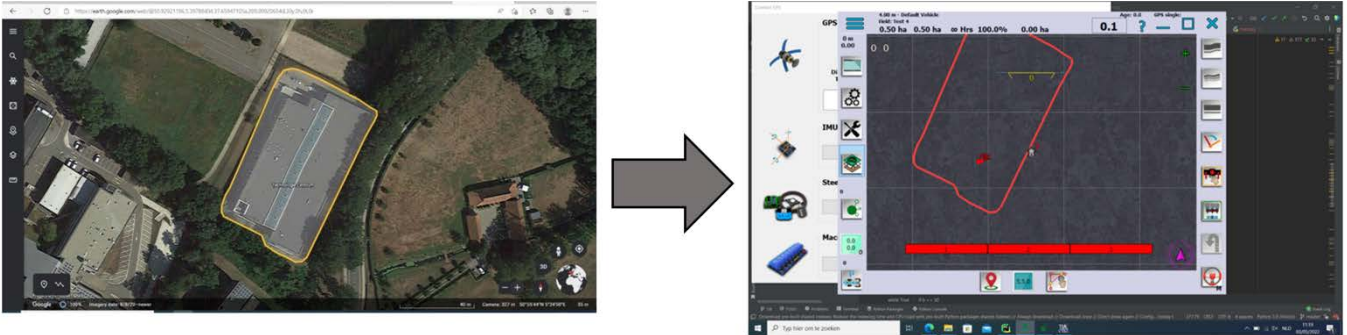


Figure 32. KML file from GoogleEarth to path.

According to the creator of the program, when configuring the autosteering it is necessary to do it by trial and error until the parameters are adjusted to your needs and the driving is as smooth as possible.

The only parameter that we know for sure, the max steering angle which in our case is 35° (see

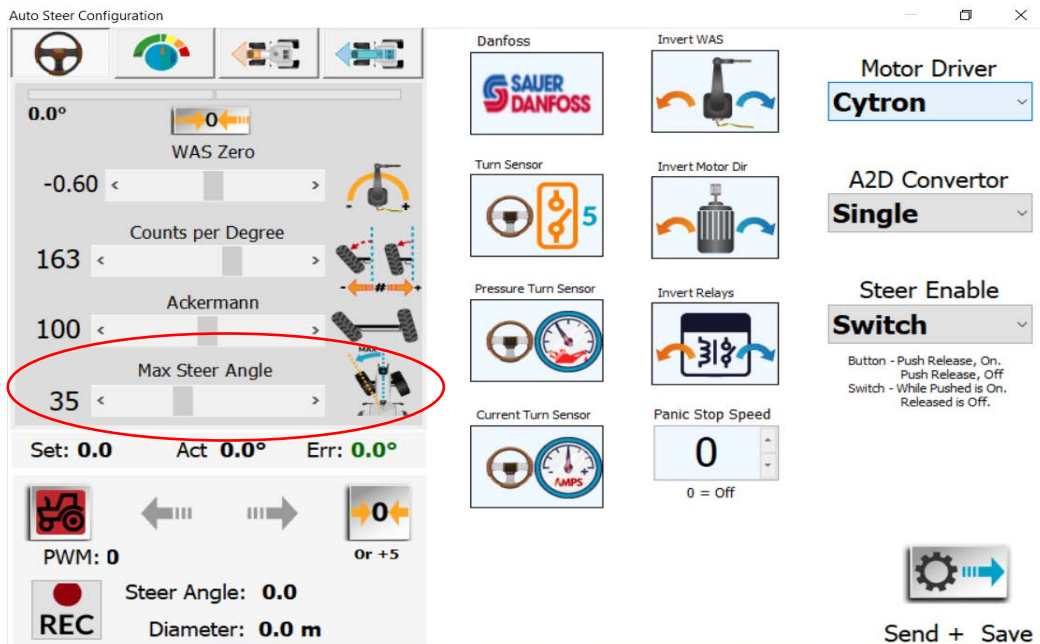


Figure 33. Auto steer configuration menu.

3.1.2. TIA-Portal

General Info:

The S7-300 can be programmed using the Siemens STEP 7 software, called TIA-Portal (Totally Integrated Automation Portal), which is a comprehensive programming environment that supports a variety of programming languages, including Ladder Logic, Function Block Diagram and Structured Text.

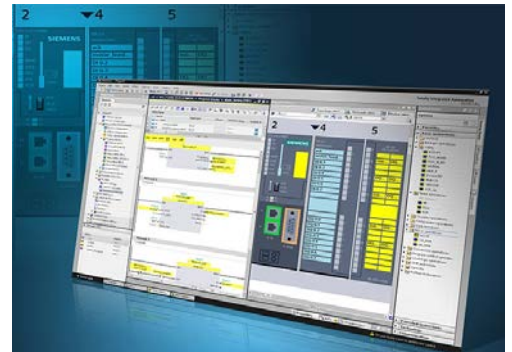


Figure 34. TIA Portal interface.

TIA Portal is a software platform developed by Siemens that is used for programming, configuring, and commissioning automation systems, including programmable logic controllers (PLCs) and as will be seen later, where almost all the functions of the vehicle are being programmed.

One of the main characteristics of TIA Portal is the ability to program and configure PLCs using the Siemens STEP 7 software. It also provides a range of powerful debugging and testing tools, as well as support for industry-standard communication protocols such as PROFIBUS and PROFINET.

TIA Portal also provides a range of tools for configuring and commissioning automation systems, including support for configuration and commissioning of human-machine interfaces (HMIs), drives and other automation components. It also provides tools for diagnosing and troubleshooting automation systems, including support for online monitoring, and debugging of PLCs and other devices.

In conclusion, TIA Portal is a powerful and flexible software platform that is widely used in the automation industry for programming and configuring automation systems.

3.2. METHODOLOGY, EXPERIMENTAL PROCEDURE.

3.2.1. IMPLEMENTED AND MODIFIED CODE.

4.2.1.1. SHARP7



Figure 35. Sharp7 logo.

Sharp7 is the main tool that has been used to modify the code. Sharp7 is an open-source sub-library from Snap7 (which works for many programming languages) written in C# that allows you to connect to Siemens PLCs using the S7 communication protocol. The library provides an easy communication interface for sending and receiving data from the PLCs.

The main features of Sharp7 according to its official webpage are:

- Fully standard “safe managed” C# code without any dependencies.
- Packed protocol headers to improve performances.
- Helper class to access to all S7 types without worrying about Little-Big endian conversion.
- Compatible with Universal Windows Platform including Win10 IoT for Raspberry.
- One single file.
- Compatible with Snap7.net.cs, plug and play replaceable.
- No additional libraries to deploy.
- Virtually every hardware with an Ethernet adapter able to run a .NET Core can be connected to an S7 PLC.

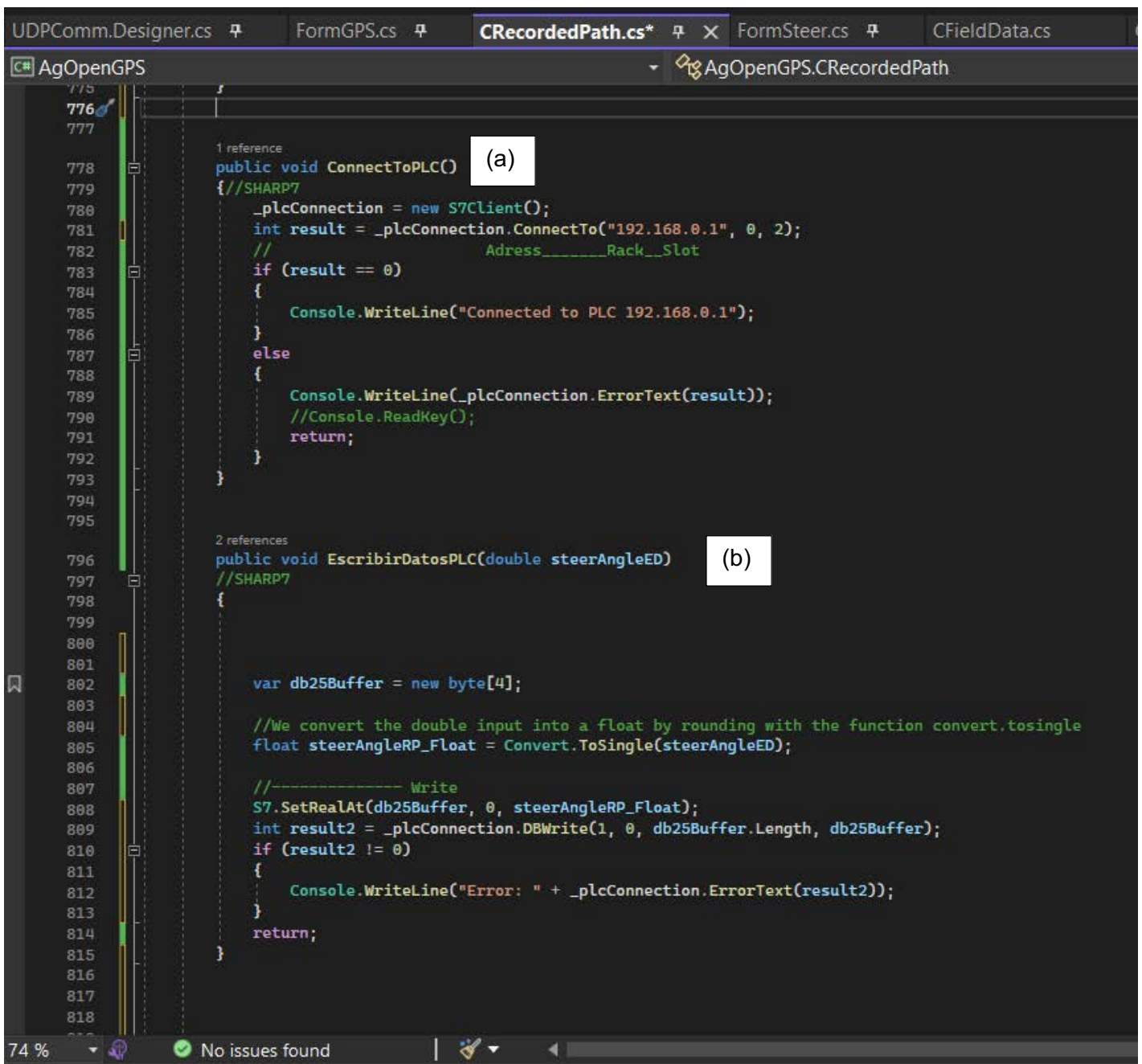
In this project we use in fact the .NET framework developed by Microsoft that runs primarily on Microsoft Windows. It includes a large library of pre-coded solutions to common programming problems, and a virtual machine that manages the execution of programs written in various programming languages.

.NET applications are programs that are built using the .NET framework and can run on any device or operating system that has the .NET runtime installed. These applications can be created using a variety of programming languages, such as C#, VB.NET, and F#,

and can be used for a wide range of purposes, including building desktop applications, web applications, mobile apps, and games.

4.2.1.2. CODE IMPLEMENTED IN AgOpenGPS.

- i. Sending the variable to the PLC



```

775
776
777
778 1 reference
779  public void ConnectToPLC() (a)
780  { //SHARP7
781    _plcConnection = new S7Client();
782    int result = _plcConnection.ConnectTo("192.168.0.1", 0, 2);
783    // Address _____ Rack __ Slot
784    if (result == 0)
785    {
786      Console.WriteLine("Connected to PLC 192.168.0.1");
787    }
788    else
789    {
790      Console.WriteLine(_plcConnection.ErrorText(result));
791      //Console.ReadKey();
792      return;
793    }
794  }
795
796 2 references
797  public void EscribirDatosPLC(double steerAngleED) (b)
798  { //SHARP7
799  {
800
801
802    var db25Buffer = new byte[4];
803
804    //We convert the double input into a float by rounding with the function convert.tosingle
805    float steerAngleRP_Float = Convert.ToSingle(steerAngleED);
806
807    //----- Write
808    S7.SetRealAt(db25Buffer, 0, steerAngleRP_Float);
809    int result2 = _plcConnection.DBWrite(1, 0, db25Buffer.Length, db25Buffer);
810    if (result2 != 0)
811    {
812      Console.WriteLine("Error: " + _plcConnection.ErrorText(result2));
813    }
814    return;
815  }
816
817
818
819

```

Figure 36. Writing the variable methods.

The first modification that has been made to the source code of the program is the implementation of a function that allows to send the information on the calculation of the steering angle that the vehicle must take to the PLC in real time.

```

454 steerAngleRP = glm.toDegrees(Math.Atan2 * (((goalPointRP.easting - pivotAxlePosRP.easting) * Math.Cos(LocalHeading))
455 + ((goalPointRP.northing - pivotAxlePosRP.northing) * Math.Sin(LocalHeading))) * mf.vehicle.wheelbase / goalPointDistanceSquared));
456
457 if (steerAngleRP < -mf.vehicle.maxSteerAngle) steerAngleRP = -mf.vehicle.maxSteerAngle;
458 if (steerAngleRP > mf.vehicle.maxSteerAngle) steerAngleRP = mf.vehicle.maxSteerAngle;

```

Figure 37. Angle calculation function.

This calculation of the angle is done from lines 454 to 458 of code on Figure 37. The value of this variable called 'steerAngleRP' is constantly changing in real-time according to other parameters received such as the GPS position, the actual value of the steering angle read by the sensor, the speed at which the vehicle is moving, etc.

To send the information about this variable to the PLC we use the *Sharp 7* library as we saw on section 4.2.1.1.

First, we open a client connection using a *c#* method (method (a) from Figure 36) with the PLC using its IP address, and the rack-slot information from the PLC. After that, we must check if the connection between the program and the PLC was successful sending a message through the console whether it was successful or not (if not, an error description is shown)

```

1 reference
778 public void ConnectToPLC()
779 { //SHARP7
780     _plcConnection = new S7Client();
781     int result = _plcConnection.ConnectTo("192.168.0.1", 0, 2);
782     // Address _____ Rack _____ Slot
783     if (result == 0)
784     {
785         Console.WriteLine("Connected to PLC 192.168.0.1");
786     }
787     else
788     {
789         Console.WriteLine(_plcConnection.ErrorText(result));
790         //Console.ReadKey();
791         return;
792     }
793 }
794

```

Figure 38. (a) Connection to the PLC method.

This connection to the SIEMENS client to send data, only **must be done once and not with each iteration** (which generated errors in the first versions of the code implementation).

Looking through the code, it has been decided that the connection should be made when the user starts the *RecordedPath* option in the application. That's why this method/function is called inside the *StartDrivingRecordedPath()* function as we can see in Figure 39.

```

92 | 1 reference
93 | public bool StartDrivingRecordedPath()
94 | {
95 |     //SHARP7
96 |     ConnectToPLC();
97 | }

```

Figure 39. Calling of the connection method.

The next step is to send to the PLC the value of the variable we need (value of *steerAngleRP*). First, we create a buffer called 'db24Buffer' with the size we want, in this case 4 bits (using the 'var' declaration that in C# is used to automatically detect the type of variable it is).

After that, because the variable *steerAngleRP* was declared *double*, we must transform it to a float type of variable (in essence we are truncating the number from 8 bytes (64 bits) to 4 bytes (32 bits)). To transform the variable, we use the command 'Convert.ToSingle()' from the c# library.

```

795 |
796 | 2 references
797 | public void EscribirDatosPLC(double steerAngleED)
798 | {
799 |     //SHARP7
800 |
801 |     var db25Buffer = new byte[4];
802 |
803 |     //We convert the double input into a float by rounding with the function convert.tosingle
804 |     float steerAngleRP_Float = Convert.ToSingle(steerAngleED);
805 |
806 |     //----- Write
807 |     S7.SetRealAt(db25Buffer, 0, steerAngleRP_Float);
808 |     int result2 = _plcConnection.DBWrite(1, 0, db25Buffer.Length, db25Buffer);
809 |     if (result2 != 0)
810 |     {
811 |         Console.WriteLine("Error: " + _plcConnection.ErrorText(result2));
812 |     }
813 |     return;
814 | }

```

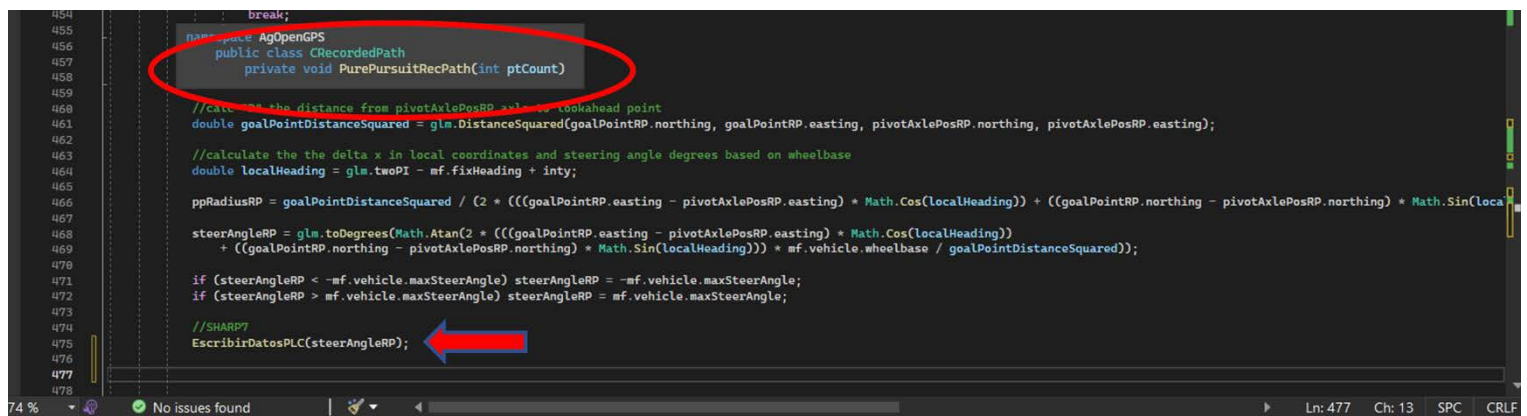
Figure 40. (b) Sending the variable to the PLC.

Then, we write the float value on the buffer we initially created 'db24Buffer' specifying its position inside the byte (position 0 in this case) with the command `S7.SetRealAt()` from the Sharp7 library. When this is done, we send the whole buffer to the PLC through the connection we already made, and we write it on the DataBlock we want (in our case DataBlock number 24) with the command `_plcConnection.DBWrite()`.

The final step is to check if the writing of the value in the DataBlock is being successful. Normally we would have to program the disconnection from the PLC client using `client.Disconnect()`, but this disconnection causes problems, which is why it has been decided not to program so it would be disconnected when the program is closed.

Then, after the method is program, we must call it in to places in order to pass the information correctly to the PLC.

The first call is made inside the function `PurePursuitRecPath()`, as we can see in Figure 41, this function calculates the steering angle once it is inside the recorded path. When making the call, we must pass to our method the value of the variable that we are interested in sending to the PLC, as it has already been said, it is the variable (`steerAngleRP`)



```

454         break;
455     namespace AgOpenGPS
456     {
457         public class CRecordedPath
458         {
459             private void PurePursuitRecPath(int ptCount)
460
461             //calculate the distance from pivotAxlePosRP to the goalPointRP
462             double goalPointDistanceSquared = glm.DistanceSquared(goalPointRP.northing, goalPointRP.easting, pivotAxlePosRP.northing, pivotAxlePosRP.easting);
463
464             //calculate the the delta x in local coordinates and steering angle degrees based on wheelbase
465             double localHeading = glm.twoPI - mf.fixHeading + inty;
466
467             ppRadiusRP = goalPointDistanceSquared / (2 * (((goalPointRP.easting - pivotAxlePosRP.easting) * Math.Cos(localHeading)) + ((goalPointRP.northing - pivotAxlePosRP.northing) * Math.Sin(localHeading))));
468
469             steerAngleRP = glm.toDegrees(Math.Atan2 * (((goalPointRP.easting - pivotAxlePosRP.easting) * Math.Cos(localHeading)) + ((goalPointRP.northing - pivotAxlePosRP.northing) * Math.Sin(localHeading)))) * mf.vehicle.wheelbase / goalPointDistanceSquared);
470
471             if (steerAngleRP < -mf.vehicle.maxSteerAngle) steerAngleRP = -mf.vehicle.maxSteerAngle;
472             if (steerAngleRP > mf.vehicle.maxSteerAngle) steerAngleRP = mf.vehicle.maxSteerAngle;
473
474             //SHARP7
475             EscribirDatosPLC(steerAngleRP);
476
477
478

```

Figure 41. `EscribirDatosPLC` call inside `PurePursuitRecPath()`.

The second call is made in the same way inside the function `PurePursuitDubins()` (see Figure 42) that calculates the steering angle necessary to get to the path creating/calculating what's called *Dubins*.

```

647
648
649 namespace AqOpenCDS
650     public class CRecordedPath
651     private void PurePursuitDubins(int ptCount)
652
653 //calculate the distance from pivotAxlePosRP to lookahead point
654 double goalPointDistanceSquared = glm.DistanceSquared(goalPointRP.northing, goalPointRP.easting, pivotAxlePosRP.northing, pivotAxlePosRP.easting);
655
656 //calculate the the delta x in local coordinates and steering angle degrees based on wheelbase
657 //double localHeading = glm.twoPI - mf.fixHeading;
658
659 double localHeading = glm.twoPI - mf.fixHeading + inty;
660
661 ppRadiusRP = goalPointDistanceSquared / (2 * (((goalPointRP.easting - pivotAxlePosRP.easting) * Math.Cos(localHeading)) + ((goalPointRP.northing - pivotAxlePosRP.northing) * Math.Sin(localHeading)));
662
663 steerAngleRP = glm.toDegrees(Math.Atan2 * (((goalPointRP.easting - pivotAxlePosRP.easting) * Math.Cos(localHeading))
664 + ((goalPointRP.northing - pivotAxlePosRP.northing) * Math.Sin(localHeading)))) * mf.vehicle.wheelbase / goalPointDistanceSquared);
665
666 if (steerAngleRP < -mf.vehicle.maxSteerAngle) steerAngleRP = -mf.vehicle.maxSteerAngle;
667 if (steerAngleRP > mf.vehicle.maxSteerAngle) steerAngleRP = mf.vehicle.maxSteerAngle;
668
669 //SHARP?
670 EscribirDatosPLC(steerAngleRP);
671

```

Figure 42. EscribirDatosPLC call inside PurePursuitRecDubins().

Dubins-paths are a type of mathematical path that is used in control theory and robotics to model the movement of vehicles with limited turning radius, such as cars and airplanes. They are characterized by a series of short straight segments separated by smooth turns and are used to find the shortest or most efficient path between two points while taking into account the constraints of the vehicle's turning radius. Consequently, it optimizes the arrival to the path by calculating the most efficient way to get there, as we can see in Figure 43 (green lines) where it calculates the path to the beginning of the Recorded Path.



Figure 43. Dubins path calculation.

- ii. Receiving the variable from the PLC.

The first thing to do, as when writing data to the PLC, is to establish the connection. This is done when starting the *FormGPS.cs* class which is basically started when the program is run and is kept open (as already mentioned, without opening and closing it with each iteration) until the class is closed.

```
365 //Initialize items before the form Loads or is visible
366 1 reference
367 private void FormGPS_Load(object sender, EventArgs e)
368 {
369     this.MouseWheel += ZoomByMouseWheel;
370
371     //SHARP7
372     var client = new S7Client();
373     this.client.ConnectTo("192.168.0.1", 0, 2);
374 }
375 //start udp server is required
376 StartLoopbackServer();
```

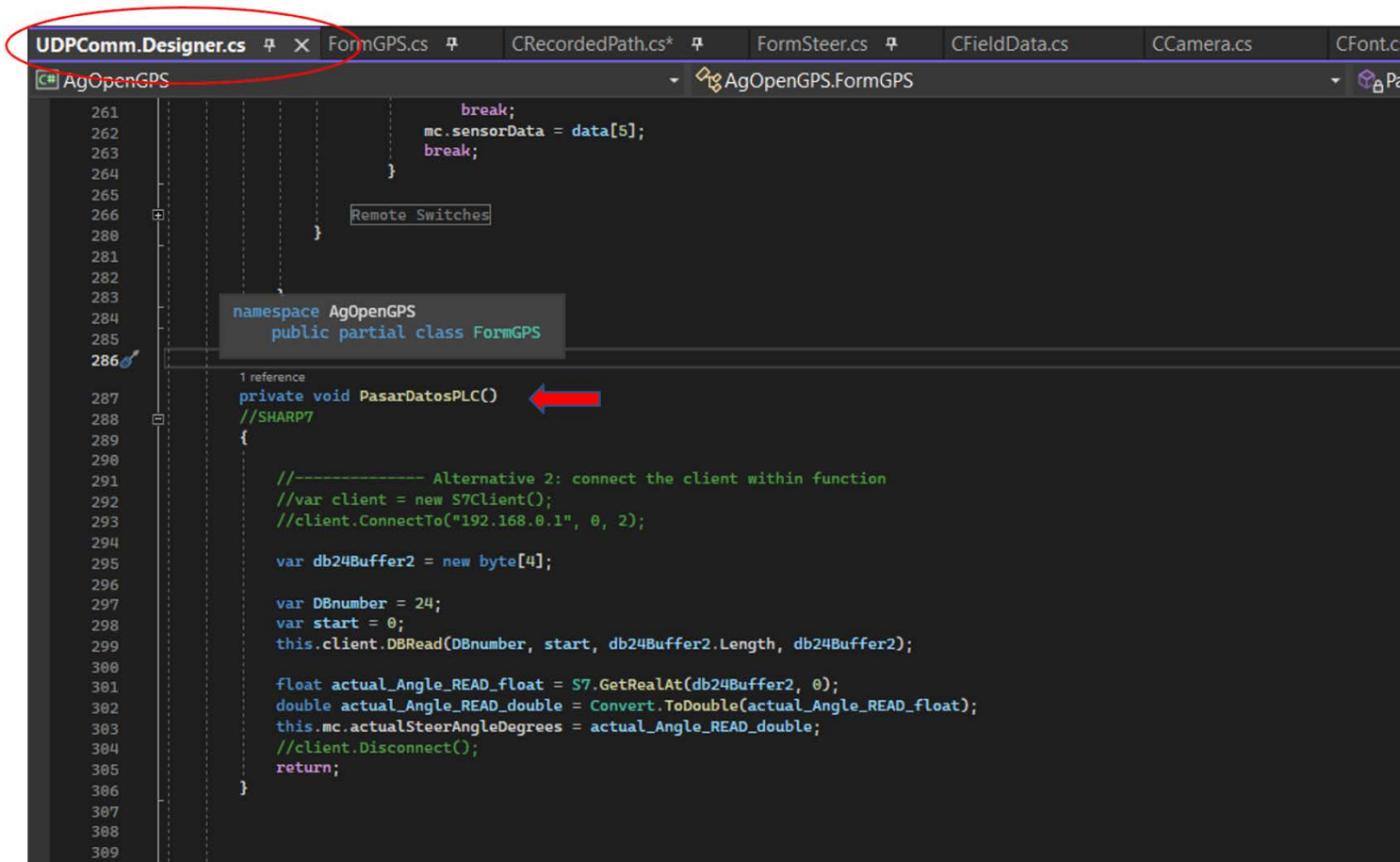
Figure 44. Connection to read form PLC

As you can see in Figure 44 the connection to the client is made inside a method that starts the *FormGPS* class and the disconnection (Figure 45) inside a method that closes the class.

```
568 }
569 namespace AgOpenGPS
570 public partial class FormGPS
571 private void FormGPS_FormClosing(object sender, FormClosingEventArgs e)
572 {
573     //save current vehicle
574     SettingsIO.ExportAll(vehiclesDirectory + vehicleFileName + ".XML");
575
576     //SHARP7
577     this.client.Disconnect();
578 }
579 }
```

Figure 45. Disconnection after reading from PLC.

To be able to do the calculations in a closed loop, it is necessary to read the current value of the angle of the axis that is measured, as has already been commented before, by a sensor. To read this value from the PLC, a method called `PasarDatosPLC()` has been created. (See Figure 46).



```

261         break;
262         mc.sensorData = data[5];
263         break;
264     }
265
266     Remote Switches
267
268
269
270
271
272
273
274
275
276
277     namespace AgOpenGPS
278     public partial class FormGPS
279     {
280
281         1 reference
282         private void PasarDatosPLC()
283         //SHARP7
284         {
285
286             //----- Alternative 2: connect the client within function
287             //var client = new S7Client();
288             //client.ConnectTo("192.168.0.1", 0, 2);
289
290             var db24Buffer2 = new byte[4];
291
292             var DBnumber = 24;
293             var start = 0;
294             this.client.DBRead(DBnumber, start, db24Buffer2.Length, db24Buffer2);
295
296             float actual_Angle_READ_float = S7.GetRealAt(db24Buffer2, 0);
297             double actual_Angle_READ_double = Convert.ToDouble(actual_Angle_READ_float);
298             this.mc.actualSteerAngleDegrees = actual_Angle_READ_double;
299             //client.Disconnect();
300             return;
301         }
302     }
303
304
305
306
307
308
309

```

Figure 46. Reading the angle's sensor data from the PLC method.

This method is inside a class of the program called `UDPCComm.Designer.cs` that rules the communication between all the peripheral devices you can connect such as the GPS. The reading from a PLC is quite similar to the writing. First, you create a buffer where you are going to store the data from the PLC. In this case it is called `db24Buffer2` and has a 4-bite capacity. Then with the command `Client.DBRead()`. Then we read the data from de DataBlock number 24 of the PLC and we store it in to the buffer.

After that we store the data again but this time in a variable and not a buffer with the command `S7.GetRealAt()`. That translates the information in a real-type variable witch then, must be transformed in a double variable that is finally going to be used by the program passing it to a variable that already exists in the program called `'actualSteerAngleDegrees'`.

4.2.1.3. CODE PROGRAMMED IN TIA PORTAL.

To begin with, a power up sequence is programmed so it allows entering the GPS driving mode called *Driving on GPS*. This power up sequence is a sequence of buttons to be pressed on the remote control.

The sequence that should be pressed is the following:

- 1° Lever all the way to the left on position 1 to make contact.
- 2° Lever all the way to the right on position 2 to start the engine.
- 3° Lever on the middle on position 1+2 and press the main button for starting the program.

This sequence is translated into the following TIA Portal program as shown in Figure 47. On *network 1* the first two steps are programmed and then on network 2 we can see the instruction where you press the main button.

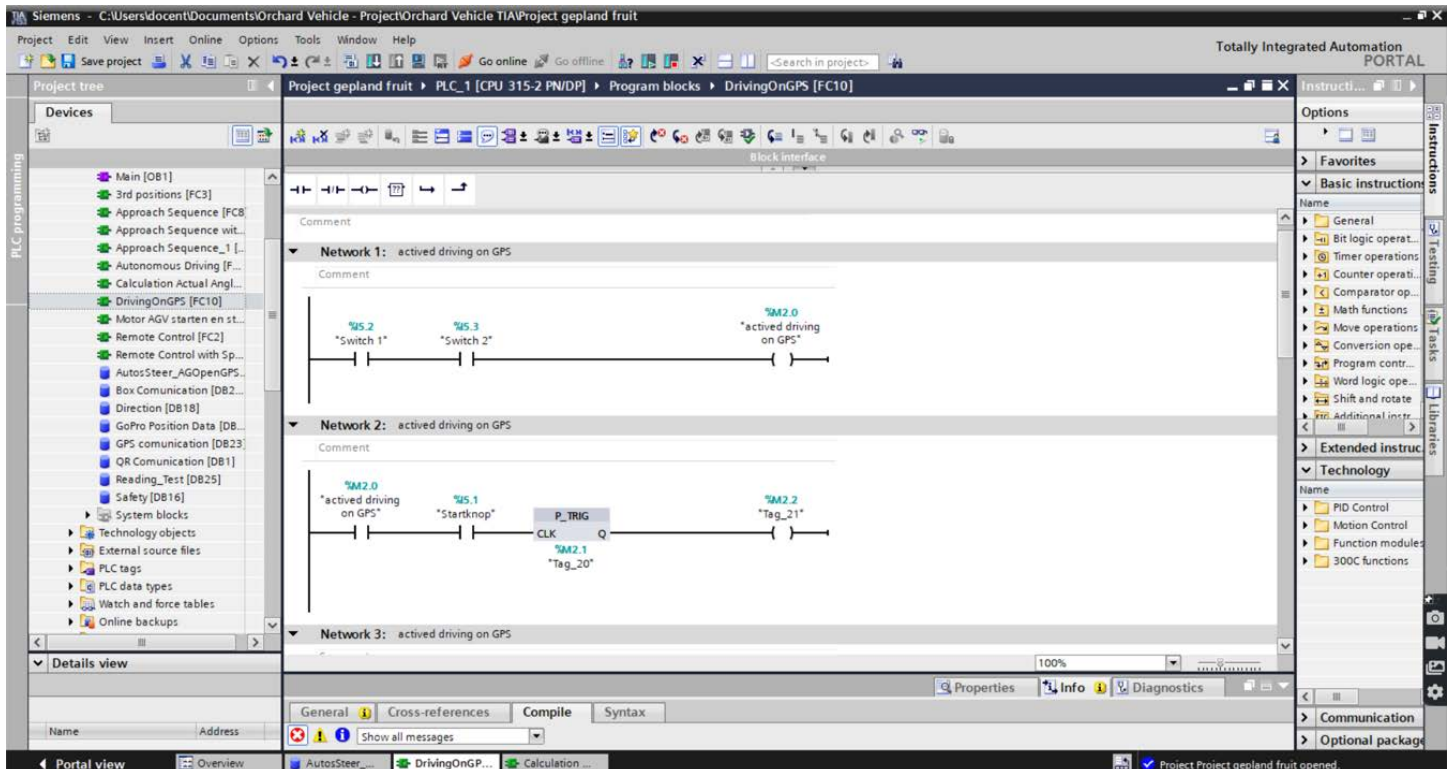


Figure 47. Start driving on GPS sequence code.

This causes a problem because just with these two networks the button must stay pressed all the time. That is why on network 3 we use a flip-flop command as we can

see on *Network 3* from Figure 48 that allows us to run the program once the button has been pressed and end it once you press it again. (see Figure 49)

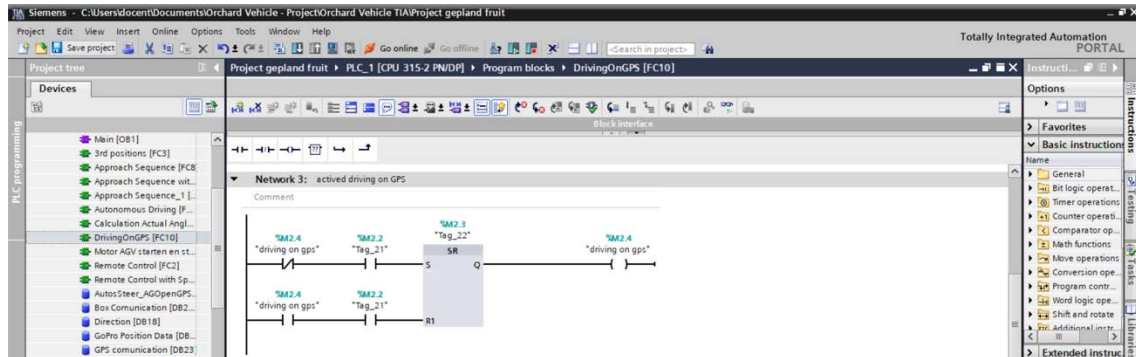


Figure 48. Flip-Flop.

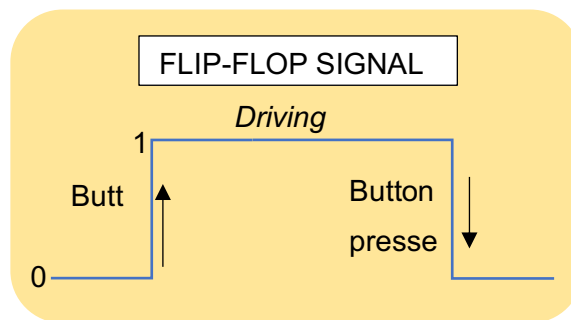


Figure 49. Flip-Flop square signal.

For the steering code programmed on TIA Portal, first we had to compare the two Angle signals. The one calculated by the AgOpenGPS program (goal sign) and the signal read by the angle sensor (actual value of the angle) in order to generate an error value that depending on whether it is positive or negative we steer right or left.

On Network 5 of Figure 50. Comparison between the calculated angle and the real one. we program the comparison with a 'SUB' block that subtracts the two values previously mentioned and generates the error signal.

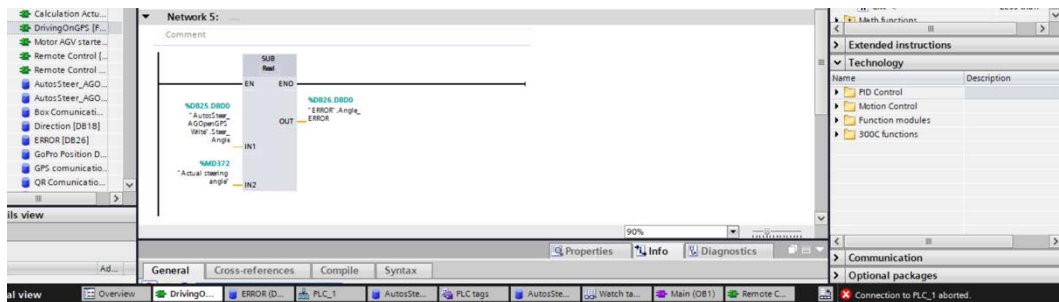


Figure 50. Comparison between the calculated angle and the real one.

After that on Network 6 and 7 using a Comparator operation block we steer right if the error value is greater than 0.0, or left if the error value is less than 0.0 as we can see on Figure 51.

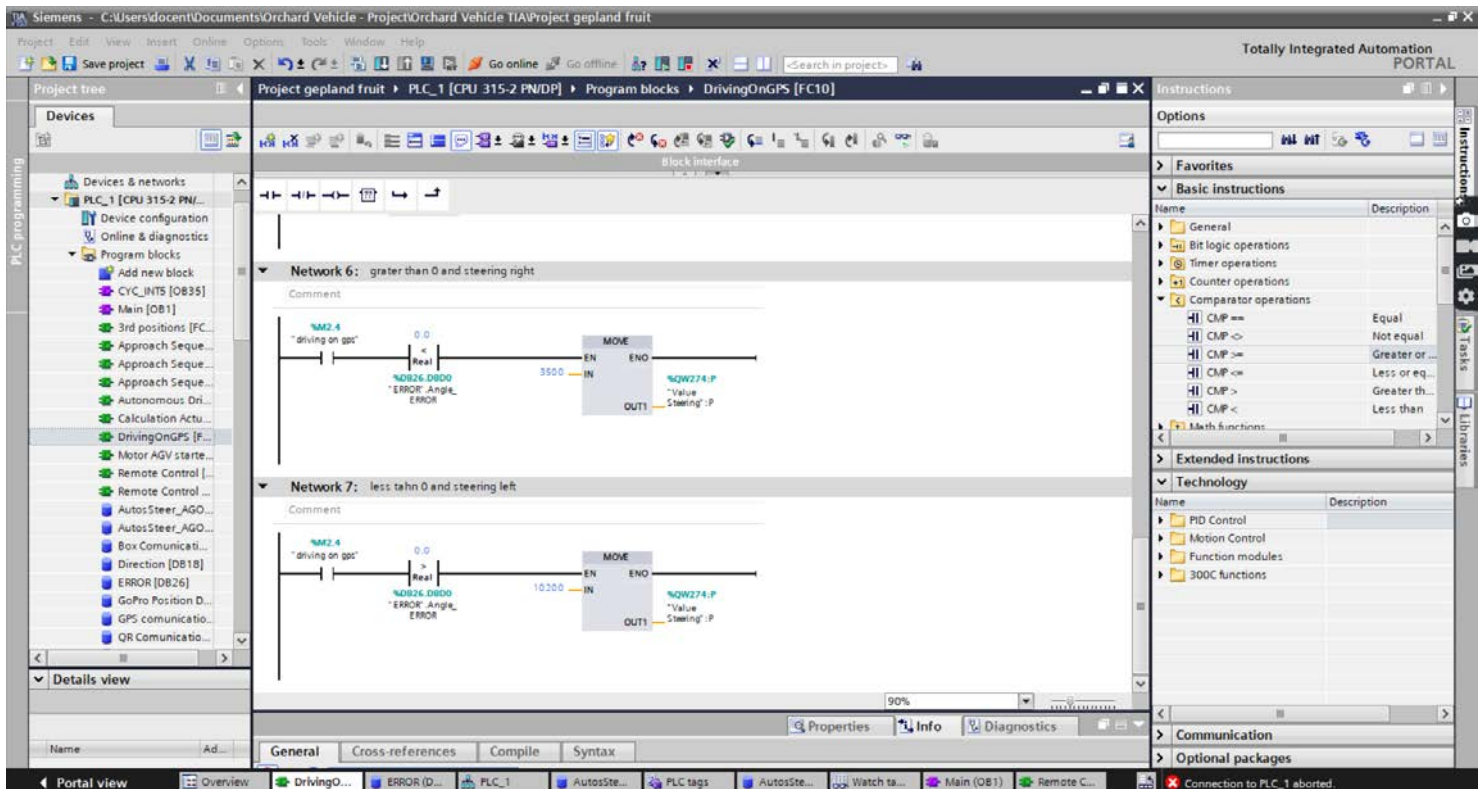


Figure 51. Steering right or left code.

3.2.2. EXPERIMENTAL PROCEDURE

i. INITIAL APPROACH.

In order to try to make the vehicle capable of autosteering around the Technology building, a linear procedure has been followed in which the bases have been built in stages, managing to solve the problems in order to move on to the next point in the system creation process.

It began with the study of the extensive AgOpenGPS project and everything that surrounded it. This study was done mostly by reading the official forums within the project's GitHub, where thousands of users asked and answered questions about their personal implementation of the system. Also, when downloading the program, all kinds of documents are included, such as plans, electric maps, purchase guides and user manuals.

ii. MODIFYING THE CODE

The methodology to be followed was established, which, as already mentioned, consists of modifying the program code so that it sends and receives signal to the PLC through an ethernet connection and then, with the necessary data, programming the PLC controller of the vehicle to control steering. Then, step by step, it starts with the different tests that in theory will lead to the final result.

First of all, a test program was created to be able to experiment with the connection to an external test PLC. The aim was to experiment and understand the functioning of the Sharp7 library. Afterwards, the reading and writing of data to a PLC was tested, for which a more complex program was created that allowed values to be entered from the keyboard so that these were later transmitted to the PLC, thus simulating the future writing of the value of the angle calculated by the program AgOpenGPS.

The next step was to study the extensive internal code of the AgOpenGPS program in order to find the key variables that calculate the angle at which the vehicle must steer and that stores the data received from the sensors.

Once the key variables have been identified, the AgOpenGPS base-code has been implemented and modified. The implemented code must now make the connection with

the definitive PLC of the AGV. Afterwards, many tests and small modifications are made in the code, seeking, in a first instance, to be able to send the steering-angle data calculated by the program to the PLC. (It should be mentioned that the implemented code has gone through many phases and versions seeking to solve bugs that appeared, so that the transmission of information is as efficient as possible).

For the reading of the actual angle from the sensor, it was a more complicated procedure. This is due to several reasons. The first one is that the identification of the variable that stored that value in the code, was much more complicated because it was hidden. On the other hand, to do all the checks, you must be permanently connected to the vehicle's PLC, with the engine running and operate the steering pistons to check the information captured by the sensor.

iii. FIRST STEERING TEST.

For the first steering tests a simple program was made on TIA Portal. This program consisted of a *MOVE* command to de motor piston so that the AGV would drive with a constant speed (Figure 52) Where the '*IN*' input (*now 11500*) must be modified to have a higher or lower constant speed.

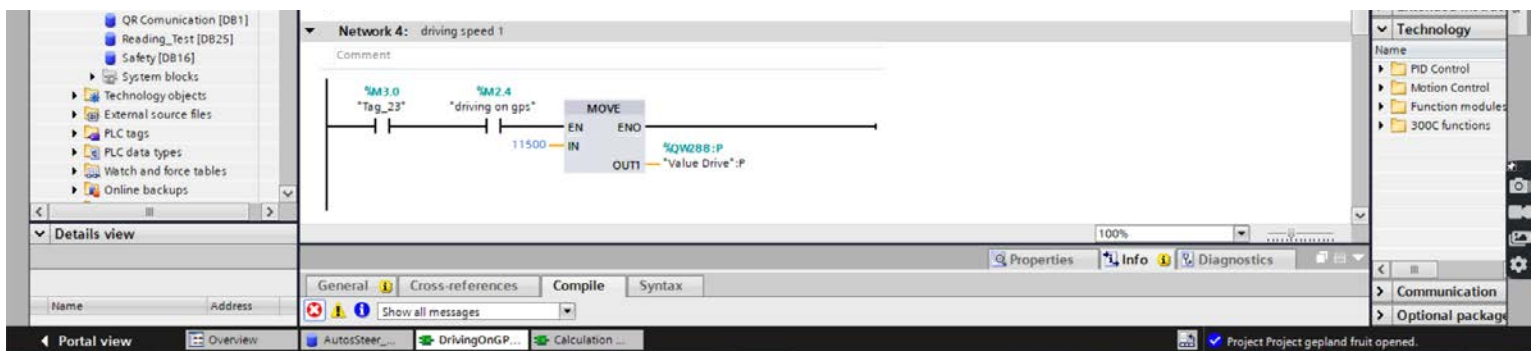


Figure 52. Driving at constant speed.

Then other two commands that were programmed that made the vehicle steer left or right depending if the steering angle that the AgOpenGPS is calculating was positive (right) or negative (left). (see Figure 54). This can be summed up in the next diagram where the most important part is the lack of a closed loop (see Figure 53):

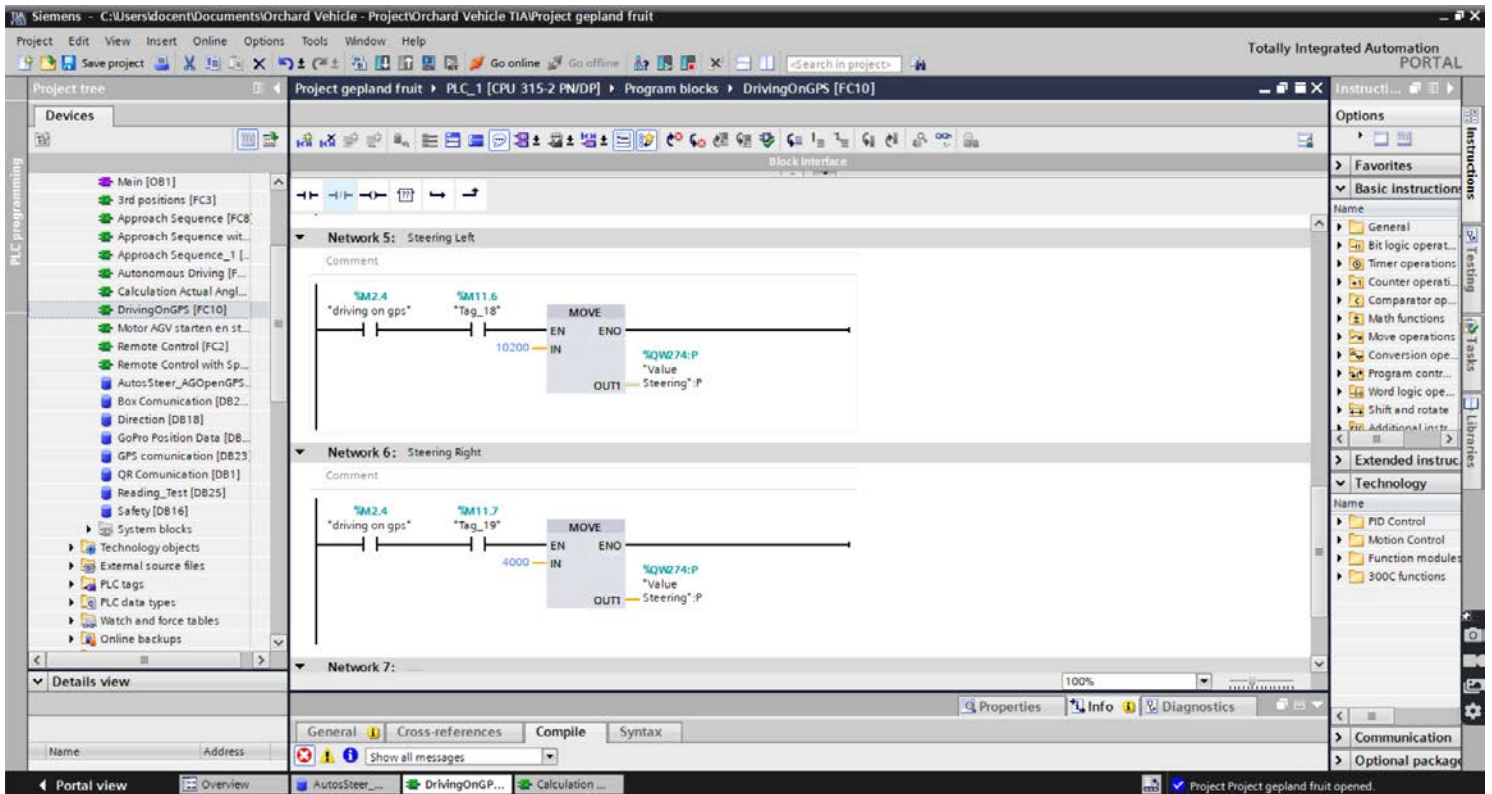
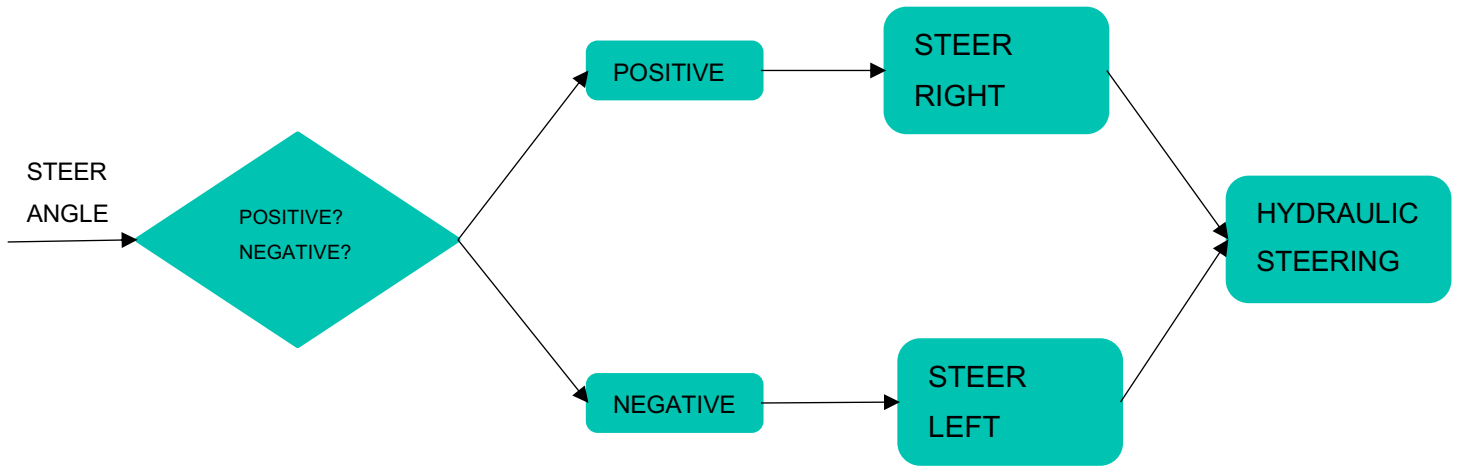


Figure 54. Steering program without feedback

Because it was not a closed loop and had only entry and exit, the steering was very abrupt and uncontrollable. Added to this, there were problems with the controller when programming the constant speed at which the vehicle was meant to go. The AGV was not able to move smoothly and was stumbling.

iv. FURTHER STEERING TESTS.

In order to succeed we need to create a closed loop and compare the Objective Signal (*Steering angle* calculated by the AgOpenGPS) and the recirculated signal sent by the angle sensor called *Actual steer angle*. Following the next diagram (Figure 55):

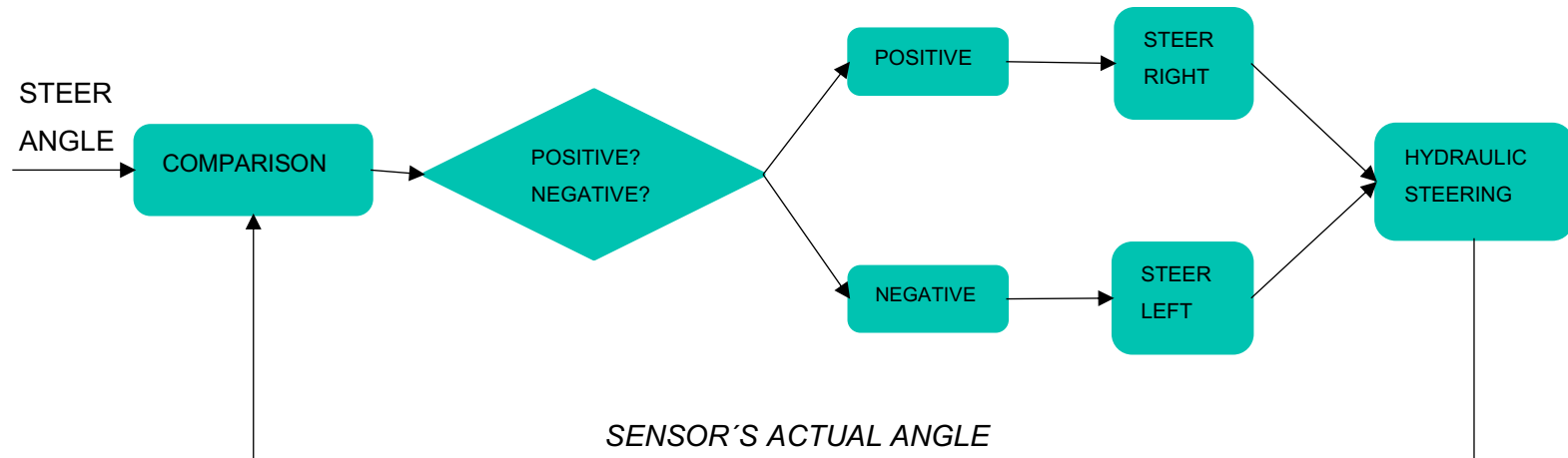


Figure 55. Steering program's sequence without the sensor's feedback.

This type of control system feedback is called Relative Position Control: that is a control logic used to calculate the difference between the current value of the angular input and the previous value. If the difference is positive, a control signal is sent to make the vehicle turn left, and if the difference is negative, a control signal is sent to make the vehicle turn right.

Following this scheme and system, the final tests were carried out outdoors, which are specified in the section 6. Results.

4. CALCULATIONS AND MEASUREMENTS

4.1. ROTATION ANGLE RANGE OF THE VEHICLE AXLE.

According to the specifications of the piston that controls the steering and the design of the vehicle, the axis is supposed to have a 70° motion range.

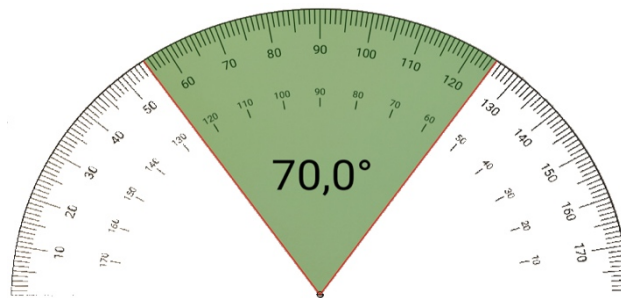


Figure 56. Steering range of the vehicle.

To prove this, we made an experiment where we turned manually the axis of the AGV all the way to one side, in this case, the left side. After that, with the help of two straight wooden planks we aligned the front axis of the vehicle with the back axis to measure manually the angle that they formed. As we can see in Figure 57.

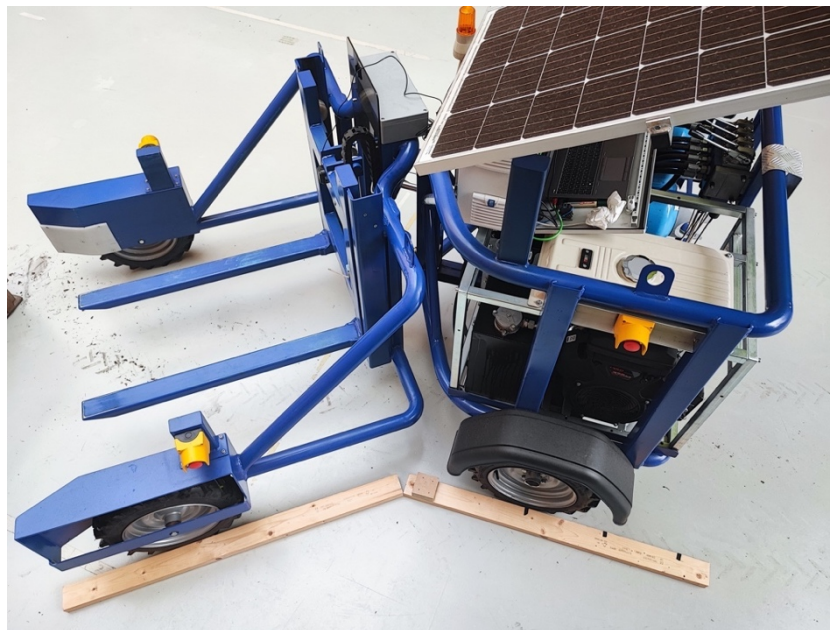


Figure 57. Wooden planks aligned with the axi.

Once the wooden planks were aligned, we used the mobile app called *Angel Meter* that allows you to measure an angle with a virtual transposer using the phone camera. As we can see on Figure 58. Using basic trigonometry, we realize the small angle formed by the wooden plank is the direct angle that the two axes form of approximately 35° .

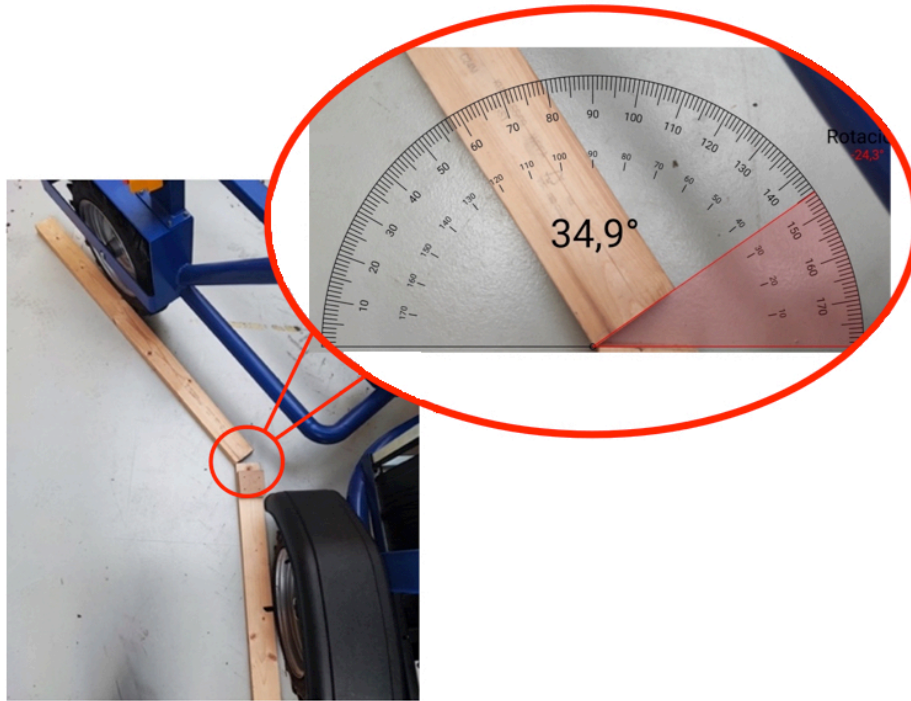


Figure 58. steering angle's range measurement.

If we establish the zero of our reference system when the vehicle is completely straight, we get a 35° range to steer to the left [-35° to 0°] and another 35° range to steer to the right [0° to 35°].

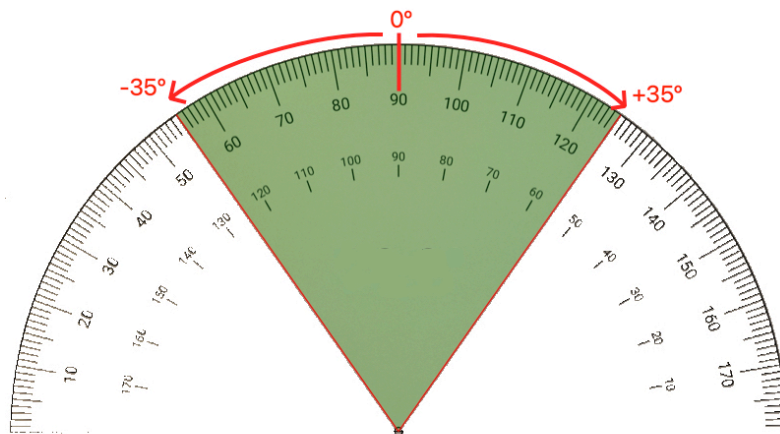


Figure 59. Steering angle's reference system.

4.2. STEER ANGLE DETECTION.

The angle sensor (previously mentioned in section 1.3.3 ANGLE SENSOR) attached to the “Steering piston” which provides the right-left movement to the vehicle, is connected to the PLC. This wire connection will involve connecting the sensor's output wires to the appropriate input terminals on the PLC, using the wiring diagram provided by the manufacturer see Figure 60. Also, the KTC-300-P sensor needs to be connected to a power source to work. This power source, as explained in section 1.2.2 ENERGY STORAGE SYSTEM. is provided by a Li-ion battery (that also powers more components of the AGV).

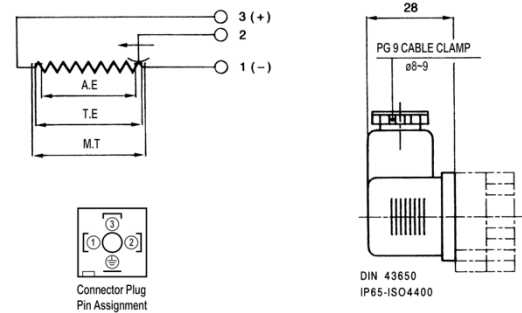


Figure 60. Wiring diagram of the angle sensor.

The angle sensor emits an electrical signal that is read by the PLC that can be monitored via the TIA Portal interface. This signal is a value that varies between 29 840 and 2 300 throughout the range of the angle capable of moving by the AGV.

Table 5. Value range conversion of the actual angle.

VALUE RANGE CONVERSION			
MOTION DESCRIPTION	SENSOR VALUE	RECEIVED VALUE	IN DEGREES
Axis all the way to the left	29824		-35°
Vehicle aligned to the center	18084		0°
Axis all the way to the right	2304		+35°

As we saw in the previous section, the range (in degrees) of the AGV steering-motion was 70°, from -35° to +35°. We must transform the data received from the sensor to degrees. The main problem is that the two functions are not linear. We need to program a code inside TIA-Portal to make that conversion in real time.

First, we calculate how much one degree equals in each of the two ranges, left and right.

a) Left range [-35° to 0°]:

$$\text{Left equivalence to } 1^\circ = \frac{\text{Eq. } 0^\circ - \text{Eq. } (-35^\circ)}{\text{Left range in degrees}}$$

Equation 1.

$$\text{Left equivalence to } 1^\circ = \frac{18084 - 29824}{35^\circ} = -335.4285$$

Equation 2.

b) Right range [0° to 35°]:

$$\text{Right equivalence to } 1^\circ = \frac{\text{Eq. } 0^\circ - \text{Eq. } 35^\circ}{\text{Right range in degrees}}$$

Equation 3.

$$\text{Right equivalence to } 1^\circ = \frac{18084 - 2304}{35^\circ} = +450.8571$$

Equation 4.

In second instance, we program on TIA Portal the algorithm that is going to do the conversion in real time. First, we must write a condition to read if we are on the left range [-35° to 0°] or on the right range [0° to 35°]. (see Figure 61)

- If the in-put of the angle read by the sensor is greater than the value 18084 (Eq. 0°) (Actual_Angle > 18084) the program enters the algorithm for the left range. *Network 1.*
- If the in-put of the angle read by the sensor is if value less or equal than the value 18084 (Eq. 0°) (Actual_Angle ≤ 18084) the program enters the algorithm for the right range. *Network 2.*

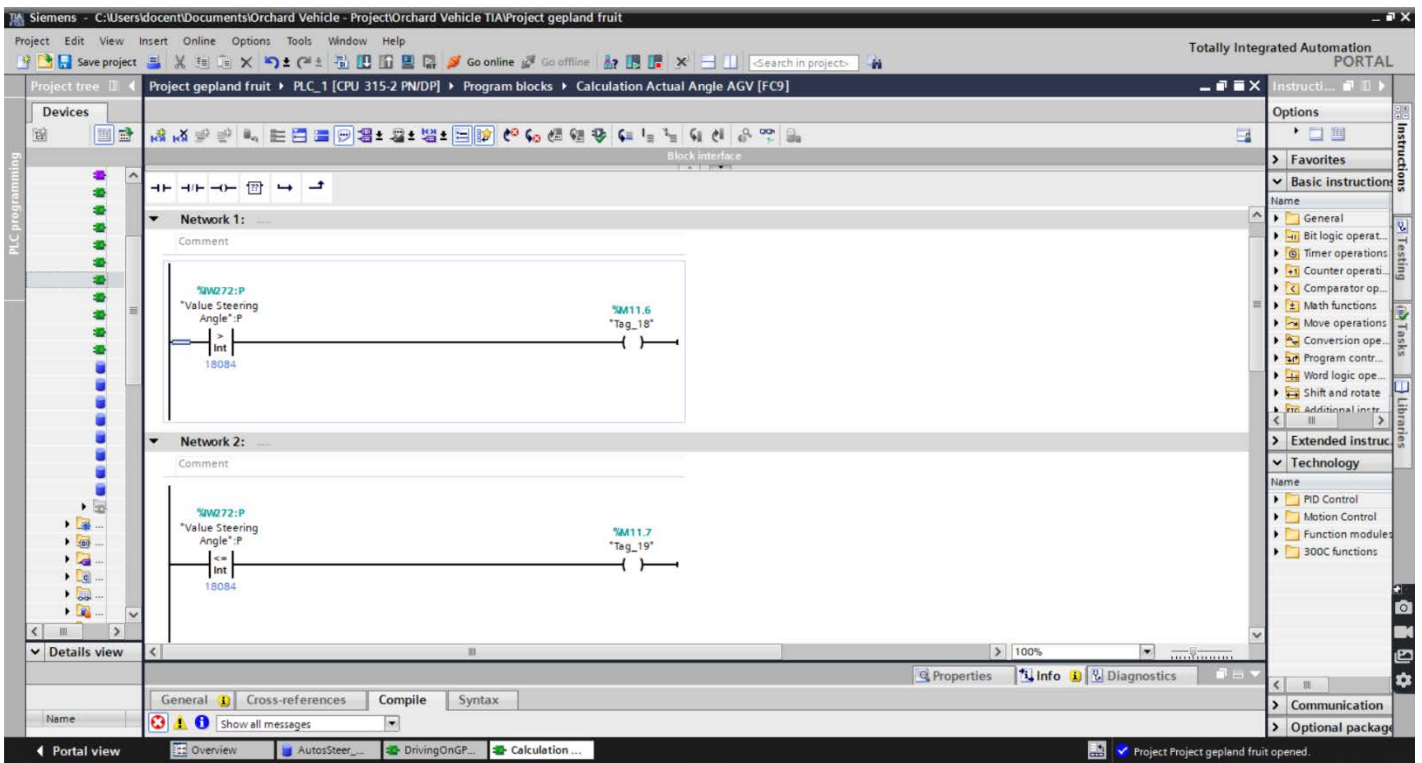


Figure 61. Angle range condition.

Once we are on the left range $[-35^\circ \text{ to } 0^\circ]$, we program the algorithm using the following formula. *Network 3*:

$$\text{Angle in degrees } ^\circ = \frac{\text{Eq. } 0^\circ - \text{Value_detected}}{\text{Left equivalence to } 1^\circ}$$

Equation 5.

$$\text{Angle in degrees } ^\circ = \frac{\text{Value_detected} - 18084}{-335.43}$$

Equation 6.

The value of the variable *Angle in degrees* will always be **negative** for the left interval because the fraction's numerator is strictly a positive number, and the denominator is constantly negative.

That translates to the program in the following sequence:



Figure 62. Angle conversion's sequence for the left interval.

- 1) We use a subtraction block to subtract the equivalence to 0°, to value detected by the sensor and sent to the PLC.
- 2) Then we use a two-conversion block to convert the value obtained in the first operation to a *real* value, first from a *Int* value to a *Dint* and then from the *Dint* value to a *Real*.
- 3) Finally, we use a *DIV* block to divide last value by the left equivalence to 1°, in this case -335.43

If we have entered the right range [0° to 35°], we program the algorithm using the following formula. *Network 4*:

$$\text{Angle in degrees } ^\circ = \frac{\text{Eq. } 0^\circ - \text{Value_detected}}{\text{Right equivalence to } 1^\circ}$$

Equation 7.

$$\text{Angle in degrees } ^\circ = \frac{18084 - \text{Value_detected}}{+450.8571}$$

Equation 8.

The value of the variable *Angle in degrees* will always be **positive** for the right interval because the fraction's numerator is strictly a positive number, and the denominator is also constantly positive.

That translates to the program in the same sequence than the left:



Figure 63. Angle conversion's sequence for the right interval.

- 1) We use a subtraction block to subtract the equivalence to 0°, to value detected by the sensor and sent to the PLC.
- 2) Then we use a two-conversion block to convert the value obtained in the first operation to a *real* value, first from a *Int* value to a *Dint* and then from the *Dint* value to a *Real*.
- 3) Finally, we use a *DIV* block to divide last value by the left equivalence to 1°, in this case +4.69.

The ultimate step of the program, *Network 5* (Figure 64), is to send the value that is obtained and converted to degrees in real time, to a *DataBlock* where it can be manipulated and used in the *AgOpenGPS* modified program to calculate on a *Close Loop* the angle that the vehicle must steer in order to follow correctly the recorded path.



Figure 64. Sending the actual angle's value to a *DataBlock*.

4.3. TURNING RADIUS.

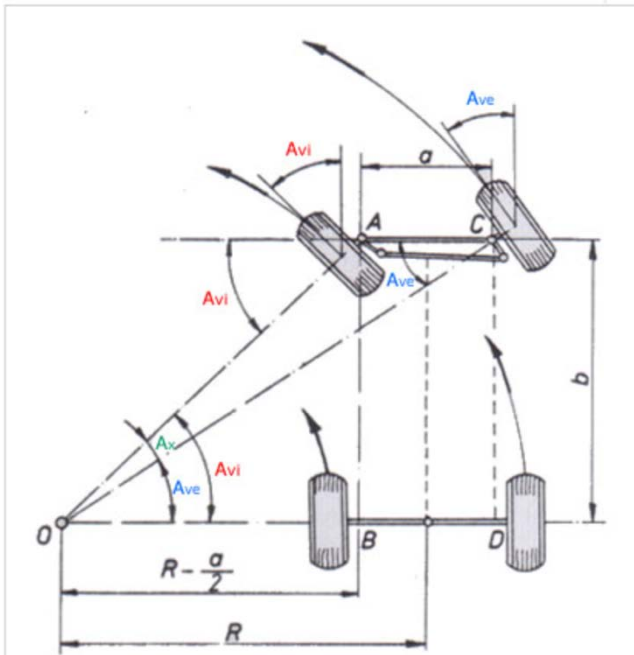


Figure 65. Turnig radius of a vehicle.

According to the source “*Diseño, implementación y análisis de sistemas de propulsión humana y dirección para vehículo solar*” page 17, the angle of turn (A_{vi}) for a given radius of gyration (R), according to the triangles rectangles OAB and OCD of the Figure 65, by trigonometric functions are obtained the angles formed by the wheels depending on the wheelbase (b) and the width track (a). Considering that the maximum turning radius in vehicles is usually about twice the wheelbase ($R_{max} \approx 2b$), steering angle maximum between the wheels is given by the equations.

$$\tan(A_{vi}) = \frac{2b}{4b - a}$$

$$\tan(A_{ve}) = \frac{2b}{4b + a}$$

Assuming $R_{max} \approx 2b \rightarrow R_{max} \approx 386 \text{ cm}$

Solving the equations we get:

$$A_{vi} = 31,41^\circ$$

$$A_{ve} = 23^\circ$$

A_{ve} = steering angle of the outer wheel.

A_{vi} = steering angle of the inner wheel.

A_x = angular deviation.

a = width of the vehicle.

b = width of the vehicle.

$$b = 193\text{cm}$$

$$a = 140\text{cm}$$

$$A_{ve_max} = A_{vi_max}$$

5. RESULTS.

After all the experimental process in which we have worked, carrying out different tests in the open air, adjusting parameters, rewriting programming code to solve errors and bugs... It can be seen that the system designed, itself, works. In the sense that when an input angle, calculated by the AgOpenGPS program, together with the feedback from the sensor and the program implemented in the PLC to control the steering movements, it works. The problem is that the vehicle is not able to follow the recorded path efficiently and that is due to a number of different things. To begin with, AgOpenGPS finds it too difficult to know the orientation of the vehicle, which makes it very difficult to follow the path. The solution to this would be to include an IMU controller as we will be mentioned below. Afterwards, the GPS signal is not as accurate as it could be and this makes it difficult for the program to make the necessary calculations efficiently (this could be because the tests are being done outdoors but surrounded by buildings, trees... that interfere with the signal). Lastly, and linked to the two points exposed above, even though the program detects that the vehicle is just above the recorded path, for some reason (possibly because it is programmed that way) it always tries to guide the vehicle along an additional path (dubin) as we can see in Figure 66 that goes off the road and makes it impossible for the vehicle to follow the road.



Figure 66. Guidance problem.

6. FURTHER IMPROVEMENTS.

For the further development of the project, some improvements are suggested.

The first improvement the system needs is to enhance the autosteer TIA Portal program with a PID controller instead of only comparing the to signals. This PID controller could adjust parameters such as the speed of the vehicle or the steering speed to minimize the error and maximize the steering efficiency and smoothness.

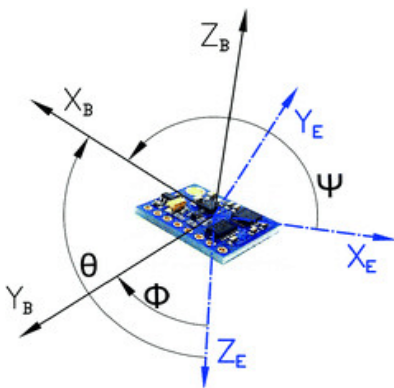


Figure 67. IMU sensor and Euler's angles.

The second improvement that can be done is to add to the system an IMU sensor there can be more feedback about the orientation and balance of the vehicle, essential if the terrain where the AGV is going to operate is not going to be exclusively flat. In order to implement such device, a similar procedure to the reading of the actual angle sent by the angle sensor must be followed. First (*this is a theory that must be corroborated, it hasn't been tested*) you would have to connect the IMU device to the PLC and then calibrate it so that the signal it sends can be interpreted correctly. After that, the

AgOpenGPS source code would have to be modify in a similar way to the reading of the angle. You would have to "delete" the part where it tries to read the data from the PCB motherboard inside *CModuleComm.Designer* and write new code to read the information from the PLC.

The final improvement proposal would be to get a RTK local antenna to improve the local GPS connection and become more accurate. This would be to do instead or at the same time than the FLEPOS connection the compatibility must be studied.



Figure 68. RTK antenna.

7. CONCLUSIONS.

In conclusion, this thesis has highlighted the importance and growing popularity of auto steering technology, not only in precision agriculture but also in fields such as transportation with companies like Tesla implementing it in their vehicles in a much larger scale. The system designed in this project and the modifications made (when finished), will result in an increased efficiency and accuracy reducing the need for manual input (which will translate in more safety) and augmenting productivity. This technology not only improves productivity but also has a positive impact on the environment by allowing the vehicle to operate with greater precision, it can reduce the amount of overlap and unnecessary passes in a field. This results in less fuel being used and fewer emissions.

Additionally, this work highlights the importance of understanding controllers, feedback control systems, programming in the development of GPS autosteering systems, as it is an essential skill for those working in the field of precision agriculture and related industries. This project demonstrates the potential in the industry and the need for further research to improve and expand its capabilities.

8. REFERENCES.

- *ArduSimple - Centimeter GPS.* (2020, julio 24). ArduSimple. <https://www.ardusimple.com/about-us/>
- Álvarez Salazar, J., & Mejía Arango, J. G. (2017). *TIA PORTAL. Aplicaciones de PLC.* Instituto Tecnológico de Medellín.
- Dasgupta, K., Mukherjee, A., & Maiti, R. (1996). Theoretical and experimental studies of the steady state performance of an orbital rotor low-speed high-torque hydraulic motor. *Proceedings of the Institution of Mechanical Engineers, Part A: Journal of Power and Energy*, 210(6), 423–429. https://doi.org/10.1243/pime_proc_1996_210_070_02
- Dewaele, K. (s/f). *Workshop AgOpenGPS PDF.*
- Gómez-Gil, J., & Alonso-García, S. (Eds.). (s/f). *A Simple Method to Improve Autonomous GPS Positioning for Tractors* written in 2011. Universidad de Valladolid. *Sensors.*
- Hernández, H. S. S. (2015). *DISEÑO, IMPLEMENTACIÓN Y ANÁLISIS DE SISTEMAS DE PROPULSIÓN HUMANA Y DIRECCIÓN PARA VEHÍCULO SOLAR* [UNIVERSIDAD TECNOLÓGICA DE PEREIRA, ESCUELA DE TECNOLOGÍAMECÁNICA.]. <https://repositorio.utp.edu.co/server/api/core/bitstreams/f9f23a84-bb31-42f2-a2c0-f26b78ae6b40/content>
- *Hydraulic247 home page - English Hydraulic247.* (s/f). Hydraulic247.com. <https://hydraulic247.com/>
- *Hydrauliek aggregaat powerpack met 26pk V-twin benzinemotor.* (s/f). Hydrauliek24 | Hydrauliek cilinders, hydromotoren en onderdelen. Recuperado el 21 de enero de 2023, de <https://www.hydrauliek24.nl/hydrauliek-aggregaat-powerpack-met-26-pk-v-twin-benzinemotor>

- *Lineaire positieensor.* (s/f). Distrelec.nl. <https://www.distrelec.nl/nl/lineaire-positieensor-05-kohm-spanningsverdeler-variohm-eurosensor-ktc-300/p/11089723?origPos=60&q=&pos=3&origPageSize=50&track=true>
- Panchenko, A., Voloshina, A., Luzan, P., Panchenko, I., & Volkov, S. (2021). Kinematics of motion of rotors of an orbital hydraulic machine. *IOP conference series. Materials science and engineering*, 1021(1), 012045. <https://doi.org/10.1088/1757-899x/1021/1/012045>
- PRIMAGAZ. Europages.es. <https://www.europages.es/PRIMAGAZ/BEL084051-00101.html>
- Nardella, D. Snap7 Homepage. Sourceforge.net. <https://snap7.sourceforge.net/>
- SIMATIC S7-300 -. (2022, julio 13). Siemens.com Global Website. <https://new.siemens.com/global/en/products/automation/systems/industrial/plc/simatic-s7-300.html>
- Tischler, B. (s/f). *AgOpenGPS: Ag Precision Mapping, Section Control and Guidance Software.*
- Various authors. (s/f). *AgOpenGPS_Manual.es.pdf version 5.7.*
- *Wij maken je bad even warm, maar jij begint niet te zweten bij de afrekening.* (s/f). Primagaz.be. <http://www.primagaz.be>

Hasselt, Belgium. 27/01/2023

Jorge Arribas Molina.

